International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 6, ISSN No. 2455-2143, Pages 85-110
Published Online October 2020 in IJEAST (http://www.ijeast.com)

A COMPARATIVE PERFORMANCE STUDY OF MACHINE LEARNING ALGORITHMS, FOR EFFICIENT DATA MINING MANAGEMENT OF INTRUSION DETECTION SYSTEMS

Salihu Alhasan
Department of Computer Science, Kebbi State Polytechnic, Dakingari
Kebbi State, Nigeria

Ajayi Ebenezer Akinyemi
Faculty of Science, Information Technology
Multimedia University, Melaka, Malesia

Daniel Dauda Wisdom
Department of Mathematics
Computer Unit, Usmanu Danfodiyo University
Sokoto (UDUS), Nigeria

ABSTRACTS - Data mining provide decision support for intrusion management, and also help in Intrusion Detection System (IDS) in detecting of new vulnerabilities and intrusions by discovering unknown patterns of attacks or intrusions. In this paper, we have compared four algorithms of Machine Learning models which are namely: Naives Bayes (NB), Decision trees (J48), Support Vector machines (SVN), and Sequential Machine optimization (SMO). The realistic models were evaluated and compared using data sets as obtained from NSL-KDDCup. The method takes into consideration the relative sizes of the classes to each other in the dataset. which allows the user of the IDS to evaluate how well they will predict the classes given the distribution of the dataset. In addition, graphs were plotted in order to efficiently analyze the results obtained of the various models depicted in Figures. The simulation results were obtained using WEKA. The simulation parameters were filtered into various tables as depicted also in Figures so as to achieve a visual conception. Finally, we carried out various computational analyses to give us semblance of graphical constructions that are related to some parameters (time, kappa characteristics, ROC etc.) of our experiments respectively.

I. INTRODUCTION

It’s a well-known fact that the evolution of modern network computer connectivity through the Internet as brought about great security challenge to computer network systems. Intrusion Malware by codes is becoming a major threat to the usability, security and privacy of computer systems and networks worldwide. This malicious threats has brought a serious concern to commercial, industries, and military organizations from financial activities and power system operations to Internet information communication and aircraft reconnaissance and attack activities (Holloway et al., 2009). Thus, network security is a very serious concern in military environment and other enterprises (such as government bodies, academic institutions and large corporations). However, outsider intrusion detection systems, insider covert network detection and system anomaly detection techniques are important security tools in Cyber space. So many such systems have been proposed with the use of standard hierarchical management structures with identification of features employing classical pattern recognition algorithms. Evolved in this detection anomalies are Machine learning techniques that range from Naïve Bayes, Decision Trees (J48), Support Vector Machine (SVM) and Sequential Model optimization (SMO) which is thrust of this thesis write-up.

A. Security Threats and its Impact

There are many security threats that pose serious challenge towards the progress of IT economy. Amongst many attacks like Man in the Middle Attack, Session Hijacking, Cross site scripting, Spamming etc. (Handley, 2004), Malicious activities on the Internet is considered to be the most deadly weapon (Handley, 2004). In the year 2009, there were several series of malicious attacks that were carried out against the US information systems and South Korea IT databases. The attack is so powerful from several countries like Canada, Japan, Australia and China. In other attacks, many government websites were brought down including the Federal trade commission and Department of

Transport (Liu, 2009). We here–under give categorized element of network security as addressed:

B. Nature of Threats

Web based insider attack: These are malware code written and embedded in javascripts that become executable code when excited by innocent clients. These web proliferating malware which are of various types are called zero-day malware. These zero-days are difficult to detect using current intrusion detection system (IDS) mechanisms. Thus, according to the multitude of internal analysis, we are fighting a losing battle against those who create malware (Holloway *et al.*, 2009). Today, for example, the stealthy worm threat is currently under control, the computer virus signature types have grown to increasingly numbers of polymorphism structural deployment. Malware are easily encoded into obfuscated malware which cannot be detected by some current detectors. In most cases, any computer that is connected to the internet is guaranteed to attack within 15 minutes; most attacks are effective. As a result of this, the malware contemporary effort has moved to the client side, embedding exploits in web pages and emails (Holloway *et al.*, 2009).
In order words, border control is no longer adequate. Intrusion detection must look both into the network traffic and host activity. Therefore any kind of defense system should look inwardly into the internal threats by identifying it, guarantying it, and eliminating the malicious entities involved.

Denial of Service Attacks: this is a serious malicious attack that is worst than worms and virus infiltration, the malicious of worms and virus are currently examined to be lower than that of DoS attack. DoS attacks are mostly effectual; attacking computers brought by internal instruction from outside targeted network. DoS attack target come in various forms such as Trojan Horses. The objective of a DoS Malware is to render an organization work flow useless with intermittent proliferation of the malware packages until the server can no longer have any space capacity to accept new information by the users.

Information Exploitation and Corruption: The dilapidation of network performance as a result of the effects of these threats result in the corruption and destruction of information. Malicious agents (intruders) exploit and remove confidential information from the networks.

Counter Defense: is a process in which the Network security officers establish measures to counteract threats. These are achieved through the use of system devices called "Intrusion Detectors" (IDs); these detectors could alert the network administrator of the malware presence in the network system of such an organization.

C. Defensive Network

Secure Middleware: Defending network attack is possible using intrusion detection systems (IDS). This IDS is very reliable, In that the detectors allow security to be quickly used to detect and report back incident. Thus, IDS is a middleware which forms a computing system that all user of computer network interact with, in detecting malware. For instance, in detecting anomalies, Aircraft has developed middleware called Cyber craft (Karrels *et al.*, 2007).

Malware (Malicious Software):

They are software designed purposely to cause damage or discomfort in computer operations. Malware is also defined as software designed to infiltrate or damage a computer system without the owner's informed consent (Mihai *et al.*, 2005). Malware includes viruses, worms, Trojans, adware and spyware. One common feature amongst these codes is their ability to install themselves on your machine without your approval. Effects of malware can range from unnoticeable to annoying to install mental wreckage and steal important documents.

- **Viruses**

Viruses are malicious software programs that are designed to cause disorder to legitimate programs. These viruses spread using a host from one computer to another and to interfere with computer operation. A virus might corrupt or delete data on your computer, use your e-mail program to spread itself to other computers, or even erase everything on your hard disk (Mihai *et al.*, 2005).

- **Worm**

Worms are another variant of malicious software programs that are self-replicate computer program. It uses a network to send copies of itself to other computers on the network and it may do so without any user intervention. Worms usually exploit a known or zero-day vulnerability that allows them to execute their copies on computers on the same network (Mihai *et al.*, 2005).

- **Trojan Horse**

A Trojan horse is non-self-replicating malware that appears to perform a desirable function for the user but instead facilitates unauthorized access to the user's computer system. Nowadays, they are usually dropped as payloads by computer worms in order to give the attacker total control of the victim's PC (Mihai *et al.*, 2005).

- **Backdoor**

As their name implies, backdoor software allows an attacker to access a machine using an alternative entry method. Normal users log in through front doors, such as login screens with user IDs and passwords. Attackers use backdoors to bypass these normal system security controls that act as the front door and its associated locks. Once attackers install a backdoor on a machine, they can access the system without using the passwords, encryption, and account structure associated with normal users of the machine (Mihai *et al.,* 2005).

- **Rootkit**

A rootkit is a software system that consists of one or more programs designed to obscure the fact that a system has been compromised. An attacker may use a rootkit to replace vital system files, which may then be used to hid processes and files the attacker has installed. Rootkits often modify parts of the operating system or install themselves as drivers or kernel modules, depending on the internal details of an operating system's mechanisms. Kernel rootkits can be especially difficult to detect and remove because they operate at the same level as the operating system itself and are thus able to intercept or subvert any operation made by the operating system. Any software such as antivirus software, running on the compromised system is equally easily subverted. The fundamental problem with rootkit detection is that if the operating system currently running has been subverted, it cannot be trusted, including to find unauthorized modifications to itself or its components (Mihai *et al.,* 2005).

- **Spyware**

Spyware sneaks into your computer without your permission. It extract the personal information or details from the computers. This information is sent to specific locations without permission of owner which can be very dangerous. The attacker uses the spywares to steal the personal information of users like password or credit card number (Mihai *et al.,* 2005).

- **Adware**

Adware usually try to sell something to the users which automatically appear as pop up window even if users don't open these. Normally this program comes to the systems in the form of the gambling advertisements and these advertisements are related to the websites which you open. There will many windows open and users will not be able to close these windows in case of adware attack (Mihai *et al.,* 2005).

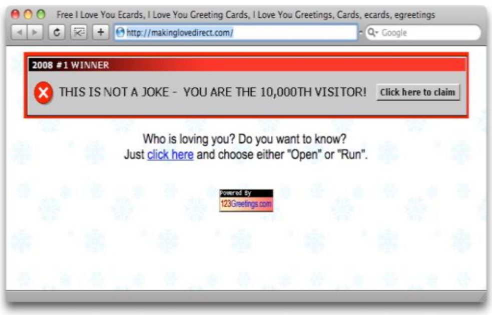
2.2 Related work

The first person to introduce Intrusion detection concept was James Anderson in 1980. To him, he see intrusion attempt or threat to be potential possibility of a deliberate unauthorized attempt to access information, manipulate or render a system unreliable or unusable (Anderson.J.P, 1980). In the latter part of 1990, data mining consisting of NIDS began to gain more attraction. Researchers suddenly recognized the need for existence of standardized dataset to train IDS tool. Minnesota Intrusion Detection System (MINDS) combines signature based tool with data mining for anomaly detection. In an early study applying GAs to intrusion detection, emphasise were based on being able to continuously learn user behaviour, to keep track of user drift (Balajinath and Raghavan, 2001). Similarly to (Balajinath and Raghavan, Neri, 2000) adopts a distributed GA, REGAL (Giordana and Neri 1995), to determine patterns of normal network behaviour. (Leon et al., 2004a, 2004b) demonstrate the potential of GAs to perform network based anomaly detection by means of clustering, which they achieve by incorporating a niching mechanism. Furthermore, Bankovi´c et al. 2008, proposed a GA over other clustering algorithms, to obtain more robustness, reduce the problem of 'getting stuck' in local optima and to exploit the parallel nature of the algorithm. They utilize the clustering potential of the GA to perform unsupervised, network based, anomaly detection. The approach does not require a predefined number of clusters, such as the popular *k*-means algorithm, and new data that is introduced to the cluster model does not need to be assigned to existing clusters; instead, new clusters may be created, thus, giving more flexibility. In a different application, by (Lin and Wang 2008), a GA is hybridised with *k*-means clustering, which allows for the value of *k* to be optimised. There are several applications of ACO based clustering to intrusion detection. Ramos and Abraham, 2004, apply an unsupervised ant clustering model, referred to as ACLUSTER, to network based intrusion detection. They argue that it is a desirable approach in this domain as the parallel and distributed nature of the ant model offers real time online training, and there is no need for complete retraining. The same benefits are argued by (Feng et al., 2006), who propose ACO clustering as a part of an agent system. Other benefits of their system include that it facilitates unsupervised and supervised learning, and that it is self organising. (Feng et al., 2007) later propose a new ACO based clustering system, hybridised with a SOM for network based anomaly detection. In addition Tsang (2005.) improve on an existing ant clustering model by (Lumer and Faieta 1994), to better deal with high dimensional data. They adopt the KDD Cup '99 data to evaluate the performance of their ant clustering model, and compare with *k*-

means clustering, SOM, another ant based clustering technique, and a multiple classifier (from other studies). Their algorithm obtained the highest detection rates on R2L and DoS, and second best for U2R, Probing and Normal. The ACO clustering system proposed by (Feng et al., 2006), as mentioned above, was also validated on a small subset of the KDD Cup '99 data set, and outperformed a DT, SVM, LGP (Linear Genetic Programming), and *k*-NN. There are several studies that demonstrate the success of GP for intrusion detection. (Abraham et al., 2007) and (Hansen et al., 2007) have both obtained high detection rates on the KDD Cup '99 data set. However, both studies used small subsets of the data, which prevents direct comparisons with other studies adopting the full data set. (Abraham et al., 2007) examined three types of GP algorithms: Linear Genetic Programming (LGP), Multi-Expression Programming (MEP), and Gene Expression Programming (GEP). They found that the different algorithms obtained better detection rates on certain classes. For example, MEP obtained the highest detection rates on U2R and R2L, whilst LGP detected Probing and DoS intrusions with higher accuracy. Similarly, (Alan Bivens et al., 2002) invented an NIDS using classifying self organizing maps for data clustering. MLP neural network is an efficient way of creating uniform, grouped input for detection when a dynamic number of inputs are present. An ensemble approach (Srinivas Mukkamala et al., 2004) help to indirectly combine the synergistic & complementary features of the different learning paradigms without any complex hybridization. The ensemble approach outperforms both SVMs, MARs, & ANNs. SVMs outperform MARs & ANN in respect of Scalability, training time running time & prediction accuracy.

D. Malware Based JavaScript

This document is an example of malware codes infused in an HTML structure highlighted in color.


Free I Love You Ecards, I Love You Greeting Cards, I Love You Greetings, Cards, ecards, egreetings

http://makinglovedirect.com/

2008 #1 WINNER

THIS IS NOT A JOKE - YOU ARE THE 10,000TH VISITOR! Click here to claim

Who is loving you? Do you want to know?
Just click here and choose either "Open" or "Run".

Powered By
123Greetings.com

Fig 1. example of malware codes infused in an HTML

From figure 1 above, Note the bottom image, which claims that the site is "powered by" 123 greetings.com, What one can't tell from Fiigure 1's static screen capture is that the image at the top of the page flashes the red border and red "x" icon as an animated .gif image, in an annoying throbbing look. Clicking on the image or on the "click here" text link would download two different executable files.

Unseen to the naked eye is an invisible <iframe> element that runs a ton of Javascript whose goal is to load additional software onto a vulnerable PC. The iframe element delivers a classic "drive-by" attack, so-called because all actions occur just by visiting the page, requiring no further action by the victim.

II. METHODOLOGY

This research compares the different algorithms of some machine learning on Intrusion Detection Systems. The Algorithms are: Naïve Bayes, Decision Trees (J48), Support Vector Machine (SVM) and Sequential Model Optimization (SMO). In view of this assertion, we here give the theoretical exposition of each model. In the design stage of this Project, we are going to highlight the theories of the four (4) Data mining algorithms we are going to deploy or run on the WEKA software.

A. A Pretty Conception of the Naives Bayes

The naïve Bayesian classifier is based on Bayes theorem, and is a relatively simple algorithm for machine learning. The Bayesian classifier has proved itself, and according to research it has performed in line with decision tree and neural network classifiers. The Bayesian classifier demands a lot of training data in order to be effective in classification of real data. In Bayes classification, the probability of a given hypothesis is calculated to be true given that the Data belong to a certain class. This method scans through the dataset the collected dataset and then re-scans as the case may be in order to re-calculate the probability to be more or less. It is strictly the work of an uncertainty as it relates to their Domains.

Let us take for an example an attacker aiming at a particular enterprise network domain, such an attacker will either have a positive action when he gets the target or negative when he failed to succeed. Here two variables are established: positive when he hits the target or negative when he misses it.

B. The Naïve Bayes Model

The Naïve Bayes method is based on the work of Thomas Bayes (1702-1761). In Bayesian classification, we have a hypothesis that the given data belongs to a particular class. We 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

Naïve Bayes is a form of Bayesian model in its simplistic form. It considers the probability of an end result given a set of evidences or variables independently given the end results. Network Intrusion Design may be liken to an Alarm. Assume that we have a set of examples that monitor some attributes such as whether it is raining, whether an earthquake has occurred etc. Let us also assume that we know, using the monitor, about the behavior of the alarm under these conditions. In addition, having knowledge of these attributes, we record whether or not a theft actually occurred. We will consider the category of whether a theft occurred or not as the class for the naïve Bayes classifier. This is the knowledge that we are interested in. The other attributes will be considered as knowledge that may give us evidence that the theft has occurred (Mrutyunjaya Panda and Manas Ranjan Patra, 2007).

A more detail of Car theft scenario can be given below as enumerated by (Christina Lee, 2007):

Now we will consider an example. Suppose that a hypothetical car alarm that responds correctly 99% of the time. The other 1% is divided into two categories, false positives, and false negatives. False positives make up all the situations in which the car alarm goes off, but where there is no criminal activity occurring. Assume that 1% of the time that the alarm rings, that this is the case. False negatives make up all of the situations in which the car alarm does not go off, but there is an attempted theft. Assume that this event also makes up 1% of all cases in which the alarm does not go off. Now, assume that the probability of criminal activity occurring with this particular car to be 1% in any given hour. Over a period of 1 hour, the car is left unsupervised. The alarm goes off once in this time– what is the probability that a theft occurred when the alarm went off? What is the probability that a theft did not occur when the alarm went off?

One way to approach this problem is to use the concept of natural frequencies. Natural frequencies translate the probability into concrete whole numbers before transferring them back into probabilities. For example, a probability that a fair coin gives heads can be thought of as the idea that out of 1000 cases, 500 will be heads.

Examining the car burglary case, we know that the probability that a theft occurred is 1% in any hour. Therefore, in considering the natural frequency, we can assume that over a period of 10,000 hours, 100 hours will have thefts (since there is a 1% probability for theft in any hour). This period of 10,000 hours can therefore be divided into two categories: those that have thefts, and those that do not. The number of hours having thefts, as stated earlier is 100. The number of hours not having thefts is 9,900. Therefore, in the number of hours having thefts, 100, the car alarm will, on average, go off 99 times. The other 1 time it will not go off. In the 9,900 hours in which no thefts occur, the alarm will go off 99 times. 9,801 times, it will not go off. Therefore, the total number of hours with alarms is 198. The total number of hours without alarms is 9,802. So the probability that a theft occurred when the alarm went off is 99/198, or 50%. The probability that a theft did not occur given that the alarm went off is 99/198, also 50%. Note that despite the fact that the false negatives only occur 1% of the time, the alarm is nonetheless incorrect 50% of the time that it goes off due to the fact that thefts occur much less commonly than non-thefts.

The above problem can also be expressed as follows:

Let P(correct) = .98
Let P(alarm|event) = P(falsepos) = .01
Let P(alarm|event) = P(falseneg) = .01
Let P(event) = .01
Therefore,
P(correct|event) = P(falseneg) = .99
P(alarm) =
P(correct|event)×P(event)+P(falsepos)×P(event) =
.99×.01+.01×.99 = .0099+.0099 =
.0198
P(alarm|event) = P(falsepos) = P(alarm|event) = .99
P(event|alarm) = P(alarm|event)×P(event)
P(alarm) = .99×.01
.0198 = .0099
.0198 = .5 (using Bayes's rule)
P(event|alarm) = P(alarm|event)×P(event)
P(alarm) = .01×.99
.0198 = .5 (using Bayes's rule) (Christina Lee; 2007)

As can be seen from the above example, the number of false positives must be reduced to significantly to prevent the alarm from becoming more annoying than helpful.

The naïve Bayes classifier works with assumptions. These further explain why the Probability of one attribute does not have any effect on the probability of the other. Given a series of n attributes, the naïve Bayes classifier makes 2n! Independent assumptions. Nevertheless, the results of the naïve Bayes classifier are often correct. (P. Domingos, and M. J. Pizzani, 1997) examines the circumstances under which the naïve bayes classifier performs well and why. It states that the error is as 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. nIn the training phase, the naïve bayes algorithm Compute the probabilities of a theft given a particular attribute and then stores this probability. This is repeated for each attribute. In the testing phase, the amount of time taken to calculate the probability of the given class for each example, in the worst case is proportional to n, the number of attributes. However, in worst case, the time taken for testing phase is same as that for the training phase. Figure 3.1 below shows the framework for a Naïve Bayesian model to perform intrusion detection.

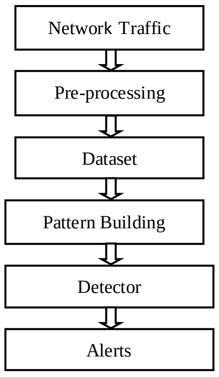


Fig. 2 the Framework of An Intrusion Detection Model

Bayes networks are among the most widely used graphical models to represent and handle uncertain information. They are specified by two main components:

A graphical component composed of a directed acyclic graph (DAG) where vertices represent events and edges are relations between events.

A numerical component is consisting of different links in the DAG by a conditional probability distribution of each node in the context of its parents.

As simple as Naïve Bayes networks, they are consisting of DAGs with only one root node which is called a Parent. The Parent represents the unobserved node with her several Children corresponding to observed nodes, with strong assumption of independence among Child nodes in the context of their parent. This thus means in the presence of training set we can only compute the conditional probabilities since the structure is unique.

Once the network is quantified, it is possible to classify any new object giving its attributes' values using the bayes' rule express as:

$$P(C_i/A) = \frac{P(A/C_i) * P(C_i)}{P(A)} \quad 3.1$$

Where C_i is a possible value in the session class and A is the total evidence on attributes the nodes. The evidence A can be distributed into pieces of evidence, for example a_1, a_2… a_n relative to the attributes A_1, A_2, ……., A_n, respectively. Since naïve baye's work under the assumption that these attributes are independent giving the parent node c, their combine probability is obtained as follows:

$$P(C_i/A) = \frac{P(a_1/c_i).P(a_2/c_i).......P(a_n/c_i) * P(c_i)}{P(A)} \quad 3.2$$

Note that there is no need to explicitly compute the denominator P(A) since it is determine by normalization condition**.**

C. Decision Trees.

Decision trees are Machine learning algorithm which comprise of three major elements or components. These components are as stated below (G.V Nadiammai, 2003):

1. ***Decision node*** which specify a test attribute
2. ***Edge or a branch***, corresponding to the one of the possible attributes values with the best attributes outcomes.
3. ***Leaf***, which is also known as an *answer node* contains the class to which the object belongs.

A decision tree is built in two phases:

1. **Building the tree**: Based on a given training set, a decision tree is built. It consists of selecting for each decision node the appropriate test attribute and also to define the class labeling each leaf.
2. **The second one is classification**, which is done in order to classify a new instance. In order to actualize this, we begin by the root of the decision tree, after which we test the attribute specified by the node. The result is allowed to move down the tree branch relative to the attribute value of the given instance. It is then allow to be repeated until a leaf is located. The instance is then classified in the same class as the characterized Leaf.

ID3 and and C4.5 algorithms (Quinlan, J.R, 1993) where among the earliest and popular work in the construction of decision trees and it well known applications in the classification world. These algorithms were use to built a model from the root to the Leaves through the use of the following parameters:

The attribute selection measure use the discriminative of each attribute over classes in order to choose the best as the root of the decision tree. In order words, this measure should consider the ability of each attribute, A_k to determine training objects' classes. Based on the gain ratio (Quinlan, J.R, 1993) and the Shannon entropy; an Attribute A_k and a set of objects T is define as follows:

$$Gain(T,A_k)=Info(T)–InfoA_k(T) \quad 3.3$$

where

$$Info(T)=-\sum_{i=1}^{n} \frac{freq(C_i,T)}{|T|} \log2 \frac{freq(C_i,T)}{|T|} \quad 3.4$$

Info A_k (T) =

$$\sum_{ak \in D\ (Ak)}^{n} \frac{|T_{ak} A_k|}{|T|} Info(T_{ak} A_k) Info\left(T_{ak} A_k\right) \quad 3.5$$

And $freq(C_i,T)$ denotes the number of objects in the set T belonging to the class C_i and T^{Ak}_{ak} is the subset of the objects for which the attribute A_k has the value a_k (belonging to the domain of A_k Denoted $D(A_k)$).

Then split Info (A_k) is define as the information content of the attribute A_k as in (Quinlan, J.R, 1993):

Split Info(T,A_k) =

$$-\sum_{ak \in D\ (Ak)}^{n} \frac{freq(C_i,T)}{|T|} \log2 \frac{freq(C_i,T)}{|T|} \quad 3.6$$

So, the gain ratio is the information gain calibrated by split Info:

$$Gainratio(T,A_k)=\frac{Gain\ (T,A_k)}{Split\ Info\ (A_k)} \quad 3.7$$

The partitioning strategy having as objective to divide the current training set by taking into account the selected test attribute. The stopping criteria, dealing with the condition (s) of stopping the growth of a part of the decision tree or even all the decision tree. In other words, they determine whether or not a training subset will be further divided.

D. Support Vector Machines (SVM)

Several extensions have been proposed to make SVMs suitable to deal with multi-class classification problems. Although none of the multi-class approaches known in the literature is accepted as a solution to generic problems, SVMs techniques are nowadays mature enough to be applicable to many classification problems (Chen et al., 2005).
The SVM approach transforms data into a feature space F that usually has a huge dimension. It is interesting to note that SVM generalization depends on the geometrical characteristics of the training data, not on the dimensions of the input space. Training a support vector machine leads to a quadratic optimization. Problem with bound constraints and one linear equality constraint. Vapnik (Joachims, 1998) shows how training a SVM for the pattern recognition problem leads to the following quadratic optimization problem (Buntod et al., 2010):

Minimize:

$$W(a)=-\sum_{i=1}^{1} a_i + \frac{1}{2}\sum_{i=1}^{1} \sum_{j=1}^{1} y_i y_j a_i a_j k(x_i,x_j) \quad 3.8$$

Subject to

$$\sum_{i=1}^{1} y_i a_i \forall i \le a_i \le c \quad 3.9$$

Where:
l = the number of training examples
a = A vector of l variables and each component i a corresponds to a training example (x_i , y_i)
The solution of (1) is the vector a* for which (1) is minimized and (2) is fulfilled.
Intrusion detection using support vector machines are being researched in universities (Harley Kozushko, 2003). The construction of SVM intrusion detection systems consists of three phases. The first is preprocessing, which uses automated parsers to process the randomly selected raw TCP/IP dump data into machine readable form. The second phase consists of training SVMs on different types of attacks and normal data. The data have 42 input features and fall into two categories: normal (+1) or attack (-1). The SVMs are trained with normal and intrusive data. The final phase involves measuring the performance on the testing data. In theory, SVMs are learning machines that plot the training vectors in high dimensional feature space, labeling each vector by class. Furthermore, SVMs classify data by determining a set of support vectors, which are members of the set of training inputs that outline a hyper plane in feature space. The SVMs are based on the concept of structural risk minimization, which recognizes true error on unseen examples. The process to which the data is classified involves partitioning the data into two classes: normal and attack, where attack represents

a collection of 22 different attacks belonging to the four classes, either: DOS attacks, unauthorized access from a remote machine, unauthorized access to a local super user privileges, or surveillance and other probing. The object is to separate normal (1) and intrusive (-1) patterns. The SVMs are trained with normal and intrusive data.

The primary advantage of SVMs is binary classification and regression which implies low expected probability of generalization errors; however there are many more advantages. Another advantage is speed as real-time performance is of primary importance to intrusion detection systems. In addition, the SVMs are very scalable. They are relatively insensitive to the number of data points and the classification complexity does not depend on the dimensionality of feature space. A final advantage is that because attack patterns are dynamic in nature, SVM can dynamically update training patterns.

E. Support Vector Classifier Algorithm using Sequential Minimal Optimization (SMO)

The math model of support vector classifier can be considered as a quadratic problem with some constraints. Traditional algorithms always involve matrix operation so, the expenses of saving and computing are large; especially the large scale data sets (C. J. C. Burges, 1998). The data sets we chose for our intrusion detection experiments are large, so after analysis and comparison we choose Sequential Minimal Optimization (SMO) algorithm as one of the Algorithm used. It was brought up by Platt in 1998. Its foundation is aimed to decompose a large quadratic problem to a serial of minimal quadratic problems.

The SMO algorithm is derived by taking the idea of the decomposition method to its extreme and optimizing a minimal subset of just two points at each iteration. The power of this technique resides in the fact that the optimization problem for two data points admits an analytical solution, eliminating the need to use an iterative quadratic programme optimizer as part of the algorithm
(Nello Cristianini, John Shawe-Taylor, 2005).

The requirement that the condition $\sum_{i=1}^{l} a_i y_i = 0$ is enforced throughout the iterations implies that the smallest number of multipliers that can be optimized at each step is 2: whenever one multiplier is updated, at least one other multiplier needs to be adjusted in order to keep the condition true.

At each step SMO chooses two elements α_i and α_j to jointly optimize, finds the optimal values for those two parameters given that all the others are fixed, and updates the α vector accordingly. The choice of the two points is determined by a heuristic, while the optimization of the two multipliers is performed analytically. Despite needing more iteration to converge, each iteration uses so few operations that the algorithm exhibits an overall speed-up of some orders of magnitude. Besides convergence time, other important features of the algorithm are that it does not need to store the kernel matrix in memory, since no matrix operations are involved, that it does not use other packages, and that it is fairly easy to implement. Notice that since standard SMO does not use a cached kernel matrix, its introduction could be used to obtain a further speed-up, at the expense of increased space complexity.

1. Population of the Study

In our work, the KDDCUP 99 was downloaded and filtered according to the type of Attacks as shown in the Table and figure. The KDDCUP'99 is about 1,048,575 connections which make it very bulky; to this end we again relied on the refined Dataset by (Mahbod Tavallaee et al., 2009) and posted on the website http://nsl.cs.unb.ca/NSL-KDD on March 2009. The website contains the following Datasets: 11850 connections representing 21% of test data, 22544 connections representing the whole of test data, 125973 connections of train data set and 25192 representing 20% train data. We carry out the experiments on the train and test data using four Data mining algorithms on Weka software.

2. Sampling Procedure

The Datasets taken from section 3.3 above were prepared and ran on WEKA software. For each of train or test dataset we conducted a four (4) data mining algorithms on it. When we ran each of the algorithms, a collected the output on our System is done and on the same Machine we again collect the ROC graphs for proper analysis.

3. Instrumentation

In order to perform this experiment we use the WEKA (Waikato environment for knowledge analysis). This software train the data collected from the KDDCup '99 after which the information obtained from the training was analyzed with the use of MALAP9.0. The experiment was carried out with the use of a windows XP service pack 3 Operating System Software running on an Intel ® Core™2 duo processor CPU T7100 1.8 GHz 2.0 GB of RAM.

III. DATA COLLECTION PROCEDURE

A. KDDCup '99 Data Collection

Network Attacks has become a major phenomenon in the field of computing and Information

Technology. Security has virtually become a confidential issue for any organization and as such Organizations conceal this piece information as classified. Since it is very difficult getting Attack information of any organize enterprise network, we strictly and wholly relied on the available samples collected by NSL- KDD. Before NSL KDD data set most of the investigators or researchers used KDD'99 data set for the investigation or detection of the intrusion, but the outcome of the KDD'99 data could not satisfy to the investigator or researchers. There are many problems in KDD'99 data set which has overcome by NSL KDD data set (Mahbod Tavallaee et al., 2009). The NSL-KDD data set has the following advantages over the original KDD data set:

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

The website (http://nsl.cs.unb.ca/NSL-KDD) contains the following Datasets: 11,850 connections representing 21% of test data, 22544 connections representing the whole of test data, 125,973 connections of train data set and 25,192 representing 20% train data. A connection is a sequence of TCP packets starting and ending at some well defined times, between source IP address to a target IP address with some well defined protocol. Each connection is categorized as normal, or as an attack, with one specific attack type. The training dataset is classified into five subsets namely **Denial of service attack, Remote to Local attack, User to Root attack, Probe attacks and normal data**. Each record is categorized as normal or attack, with exactly one particular attack type.

B. Weka program data collection

WEKA is open source software that was developed using Java at Waikato University and it is available on the website "http://www.cs.waikato.ac.nz/ml/weka/."

For each machine learning algorithm, the algorithm was always trained with the training data, and testing was performed with either the testing data or the training data. In addition, the option to output detailed statistics was selected and to output the model. The model consists of all the information necessary to reproduce the trained machine learning data structure (e.g. the decision tree, naive bayes, SMO or SVM trained on the 20% dataset). The option -Xmx1024m was used to increase the memory available to the JRE to 1024 MB for SVM and SMO. Naive Bayes algorithm outputs the results of training and testing, as well as the model for the naive Bayes. Besides the options mentioned above, the Weka naive Bayes and J48 decision tree algorithms were run using the defaults, and no other options were selected for them.

C. Data Analysis Techniques.

The data files used are from the University of California, Irvine Knowledge Discovery and Data Mining (UCI KDD) website (http://www.kdd.ics.uci.edu/databases/kddcup99/task.html) as amended by (Mahbod Tavallaee et al., 2009). The data files give the necessary information to create and train the algorithms. The kddcup names file lists the class types, including 'normal.' which signifies that no attack is in progress. The attack types are back, buffer_ overflow, ftp_write, guess_passwd, imap, ipsweep, land, loadmodule, multihop, neptune, nmap, normal, perl, phf, pod, portsweep, rootkit, satan, smurf, spy, teardrop, warezclient, and warezmaster. In the test data, the attacks mentioned above are present with other novel attacks, which include: xsnoop, xterm, Apache2, httptunnel, mailbomb, mscan, named, processtable, ps, saint, sendmail, snmpgetattack, snmpguess, sqlattack, udpstorm, worm and xlock. Table 3.1 shows the four classes of attacks and the different classifications they belong to.

While there are a large variety of attacks, most of these attacks above fit into one of four categories (Christina Lee, 2007):

Table 1. Different types of Attacks and their classes

Attack Class(4 main Classes)	Different Attacks (22 misused attacks)
Denial of Service(DOS)	back, land, neptune, pod, smurt, teardrop
Remote to Local(R2L)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
User to Root(U2R)	buffer_overflow, perl, loadmodule, rootkit
Probe	ipsweep, nmap, portsweep, satan

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data which make the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the signature of known attacks can be sufficient to catch novel variants (Mahbod Tavallaee et al., 2009)
The names file also lists the attribute names. Each attribute name states whether it is a continuous or symbolic variable. A symbolic variable has a finite number of possible values and can be completely enumerated. A continuous variable cannot be enumerated.
Each example uses forty-one attributes, and the testing data contains 23 different classes. The attributes are as given in table 3.2 to 3.4

Table 2: Basic Features of individual TCP connections

feature name	Description	Type
duration	length (number of seconds) of the connection	Continuous
protocol type	type of protocol, e.g. tcp, udp, etc	Symbolic
service	network service on the destination, e.g., http, telnet, etc.	Symbolic
flag	normal or error status of the connection	Symbolic
src_bytes	number of data bytes from source to destination	Continuous
dst_bytes	number of data bytes from destination to source	Continuous
land	1 if connection is from/to the same host/port; 0 otherwise	Symbolic
wrong_fragment	number of "wrong" fragments	Continuous
urgent	number of urgent packets	Continuous

Table 3: Content features suggested by domain knowledge

feature name	Description
hot	number of "hot" indicators
num_failed_logins	number of failed login attempts
logged_in	1 if successfully loggin in, 0 otherwise
num_compromised	number of "compromised" conditions
root_shell	1 if root shell obtained, 0 otherwise
su_attempted	1 if "su root" command attempted, 0 otherwise
num_root	number of "root" accesses
num_file_creations	number file creation operations
num_shells	number of shell prompts
num_access_files	number of operations on access control files
num_outbound_cmds	number of outbound commands in an ftp session
is_hot_login	1 if the login belongs to the "hot" list, 0 otherwise
is_guest login	1 if the login is a "guest" login, 0 otherwise

Table 4: A two-second window where various traffic features were computed

feature name	Description	Type
count	# of connections to the same host as this one in the past two seconds	Continuous
	Note: the following features refer to these same-host connections	
serror_rate	% of connections that have "SYN" errors	Continuous
rerror_rate	% of connections that have "REJ" errors	Continuous
same_srv_rate	% of connections to the same service	Continuous
diff_srv_rate	% connections to different services	Continuous
srv_count	# of connections to the same service as this one in the past two seconds	Continuous
	Note: The following features refer to these same-service connections	
srv_serror_rate	% of connections that have "SYN" errors	Continuous
srv_rerror_rate	% of connections that have "REJ" errors	Continuous
srv_diff_host_rate	% of connections to different hosts	Continuous

The NSL –KDDCup'2009 data file lists the value of the class and the value of the attributes. The testing data set contains 22,543 examples. These examples contain 9,711 normal items and 12,832 attacks. Therefore, this data is most likely atypical because it contains more attacks than normal data. The attack connections make up 56.92% of the dataset. The training dataset contains 125,972 items. There are 67,342 normal connections and 58,630 attack connections. The attacks make up 46.54% of the dataset. All the datasets has 41 attributes. Fig 4.18 shows the datasheet containing this analysis.

For the Weka algorithms, the dataset was converted to arff format, which is a standard data mining format used by Weka. This was accomplished by first converting the file to csv format using Microsoft excel, The CSV format is a standard comma-separated format. The version of the csv format read by WEKA has a row of entries at the top that lists the name of each attribute. In this step, a line with the feature names as given above was added, along with the classification name. Then the lines were processed using only the classes ’normal’ and ’anomaly’ (i.e. any class that wasn’t normal was changed to read ’anomaly’). Next, the Weka CSVLoader was used to convert both the training and the testing dataset (now in csv format) to arff format. The two arff headers were manually compared and merged to form a single arff header with all the possible attribute values from both the testing and the training phase (Weka requires that the arff headers for the testing and training data match).

D. Neural Network Java Codes Implementation for Intrusion Detection.

This work is strictly on the application of WEKA for comparison of the IDS algorithms. However, we are obliged to give a simple and workable code program in Java that could be applied in for any IDS using the neural networks, thus, code is given in Appendix A.

he Methods are based on the following machine and data mining algorithms models: using WEKA software

1. Naïve Bayesian,
2. Support Vector Machines (SVM)
3. Decision Trees (J48) and
4. Sequential Minimal Optimization (SMO).

E. Research Questions

The question we will answer in this study is; which is the best data mining or Machine learning Techniques among the four to be studied for the detection of the intrusion, such as:

1. Denial of Service (DoS), 2. Remote to Local (R2L),

3. User to Root (U2R) and 4. Probe ?

The question will be answered in terms of performance, efficiency and scalability in order to make an appropriate conclusion in chapter five.

Network Intrusion Detection System: is a specialized tool that knows how to read and interpret the contents of log files from routers, firewalls, servers, and other network devices so as to compare patterns of activity, traffic, or behavior to indentify and separate the normal from the unauthorized traffic.

Intrusion Prevention System: is a tool deployed to preventing or averting the unauthorized traffic into any Computer System or Network.

Data Mining: is defined as the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantage, usually an economic one.

Machine Learning: is the prediction, based on known properties learned from the training data or is a scientific discipline concerned with the design and allow computers to evolve behaviors based on empirical data such as from sensor data or database.

False positive: is when you have a specific vulnerability in your program but in fact you don’t or when an IDS sound an alarm signifying intrusion but in really there is no any intrusion.

True positive: when an IDS sound an alarm when really there is an unauthorized traffic in the network or System.

Receiver Operating Characteristic (ROC): curves identify how the true positives vary with the false positives. ROC curves show how well a test does at distinguishing between classes without taking the relative frequency of the classes into account.

Support Vector Machines (SVM): transforms data into a feature space F that usually has a huge dimension. The algorithms generalization depends on the geometrical characteristics of the training data, not on the dimensions of the input space.

Sequential Minimal Optimization (SMO) is aimed to decompose a large quadratic problem to a serial of minimal quadratic problems.

IV. RESULTS AND DISCUSSIONS

A. Introduction

This works compares four algorithms of Machine Learning models; viz: Naives Bayes, Decision trees (J48), Support Vector machines (SVN), and

Sequential Machine optimization (SMO). These pragmatic models were evaluated and compared using data sets as obtained from NSL-KDDCup. This method takes into consideration the relative sizes of the classes to each other in the dataset. This allows the user of the Intrusion Detection Systems (IDS) to evaluate how well they will predict the classes given the distribution of the dataset. In these comparative data analyses, we plotted various graphs and charts to analyze the results of the various models which give us experimental parameters as obtained in section (4.4). The itemized below are vital in our simulations undertakings:

1. Time taken to built the results of each Algorithms;
2. The Kappa Statistics of the Algorithms;
3. Area under the curve(AUC);
4. False positives
5. Receiver Operating Characteristics(ROC) Graphs; and
6. Experimenter results of train and test data on WEKA.

The simulated outputs for the aforementioned models were executed using open source software named WEKA. The output parameters were collected and all the various parameters filtered into various tables as shown. In order to give visual conception, we carried out various computational analyses to give us semblance of graphical constructions that are related to some parameters (time, kappa characteristics, ROC etc.) of our experiments.

B. Visualization of Experimental works

We give clear visualization for our experiments as followings:

Receiver Operating Characteristics (ROC)

Our Data sets were acquired from 1998 DARPA. Using this huge data set, we carried out various evaluations which enable us to obtain various patterns of intrusion detection parameters. Detected results were compared with the total number of network sessions to give two summary measures of an IDS's performance**: Detection rate** (intrusions detected divided by intrusions attempted) and false alarm rate (false alarms divided by total network sessions). These summary measures were taken as an estimate of one point on the IDS's receiver operating characteristic (ROC) curve which is hereunder explained. A ROC curve is a plot of detection probability of positive detection versus false alarm detection probability. It shows the probability of detection provided by the IDS at a given false alarm probability. Alternatively, it shows the false alarm probability provided by the IDS at a given probability of detection.

Receiver Operating Characteristic (ROC) curves; identify how the true positives vary with the false Positives. The area under these curves signifies how well the test used can distinguish between the examples. The more the example classes overlap relative to the test, the less the area under the ROC curve will be. ROC curves show how well a test does at distinguishing between classes without taking the relative frequency of the classes into account. The authors have observed that in the ROC curves created by the Decision Tree in WEKA, there are few points. This is most likely because many of the decision tree branches are discrete. Whether an edge is followed is therefore a binary decision (e.g. something either is or isn't an ftp connection).On continuous valued attributes, the numbers can be changed to affect the number of examples classified as normal and as an attack). The followings are the realities of what ROC is and it uses:

1. The ROC curve allows us to see, in a simple visual display, how sensitivity and specificity vary as our threshold varies
2. The shape of the curve also gives us some visual clues about the overall strength of association between the underlying test statistic
3. Area under curve of ROC represent the total area of the grid represented by an ROC curve is 1, since both TPR and FPR range from 0 to 1
4. The portion of this total area that falls below the ROC curve is known as the **area under the curve**(AUC)
5. An AUC of 1.0 would mean that the test statistic could be used to perfectly classified between cases and controls
6. An AUC of 0.5 (reflected by the diagonal 45° line) is equivalent to simply guessing
7. The following define the values of AUC and their corresponding interpretations:

.90-1.0 = excellent
.80-.90 = good
.70-.80 = fair
.60-.70 = poor
.50-.60 = fail
<.50 is worse than guess work

C. The kappa statistic

The Kappa statistics measures the agreement of prediction with the true class. 1.0 signifies complete agreement. For the example taken from fig. 4.1 the Kappa characteristics is 0.7906

D. The confusion matrix

The confusion matrix is more commonly named *contingency table*. In our case we have two classes, and therefore a 2x2 confusion matrix, the matrix could be arbitrarily large. The number of correctly classified instances is the sum of diagonals in the matrix; all others are incorrectly classified (class "a" gets misclassified as "b" exactly twice, and class "b" gets misclassified as "a" three times. From fig.4.1 the Matrix is as given below.

```
=== Confusion Matrix ===

     a       b     <-- classified as
 12272    1177 |    a = normal
  1445   10298 |    b = anomaly
```

E. The *True Positive (TP)*

The *True Positive (TP)* rate is the proportion of examples which were classified as class *x*, among all examples which truly have class *x*, i.e. how much part of the class was captured. It is equivalent to *Recall*. In the confusion matrix, this is the diagonal element divided by the sum over the relevant row.

$$TP = \frac{Diagonal\,Element}{sum\,over\,the\,relevant\,row} = \frac{12273}{13449} = 0.912 \quad for\ normal;and$$

$$= \frac{10298}{11743} = 0.877 \quad for\ Anomaly$$

F. The *False Positive (FP)*

The *False Positive (FP)* rate is the proportion of examples which were classified as class *x*, but belong to a different class, among all examples which are not of class *x*. In the matrix, this is the column sum of class *x* minus the diagonal element, divided by the rows sums of all other classes.

$$FP = \frac{Column\,sum - Diagonal\,Element}{Rows\,sums\,of\ all\,classes} = \frac{1445}{11743} = 0.123 \quad for\ Normal;$$

$$= \frac{1177}{13449} = 0.088\ for\ Anomaly$$

G. The *Precision*

The *Precision* is the proportion of the examples which truly have class *x* among all those which were classified as class *x*. In the matrix, this is the diagonal element divided by the sum over the relevant column.

$$PRECISION = \frac{Diagonal\,Element}{umoover\,the\,relevant\,Column} = \frac{12272}{13717} = 0.895\ for\ Normal;and$$

$$= \frac{10298}{11475} = 0.897\ for\ Anormal$$

H. The *F-Measure*

The *F-Measure* is simply 2*Precision*Recall/ (Precision+ Recall), a combined measure for precision and recall

I. Research Question/Hypothesis Analysis

There are so many potential Stakeholders for the results of quantitative evaluations of IDS accuracy. Acquisition managers need such information to improve the process of system selection, which is too often based only on the claims of the vendors and limited-scope reviews in trade magazines. Security analysts who review the output of IDSs would like to know the likelihood that alerts will result when particular kinds of attacks are initiated. Finally, Research & Development program managers need to understand the strengths and weaknesses of currently available systems so that they can effectively focus research efforts on improving systems, and measure their progress. Therefore the security of any enterprise network become a concern of any Systems Administrator as Intruders over time invades and targeted networks of interest. Attempts have been made to get a solution to this security menace. Of interest to us in this research is to Compare and get a more suitable Data mining techniques that could be used in the Analysis of NSL-KDDCUP data and other related intrusion detection experiments. We apply the data set as obtained from NSL- KDDCUP'99 for our research because we lack the necessary equipment to generate our own experimental works; which is a ban of all researchers will experience in Nigeria for lack of effective network Laboratories. Data collected for experimentations for training set normally take a long period of time. This data so collected are use for training before we latter have data for testing abnormalities intrusions; thus acquire abnormal detects. Significant results of training are hereby given in Section 4.4, below.

V. RESULTS

List of Results Tables Showing Tabulated Parameters Taken From Weka Software Output

```
Time taken to build model: 1.66 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly  Classified  Instances          22570
89.5919 %
Incorrectly  Classified  Instances          2622
10.4081 %
Kappa statistic                     0.7906
Mean absolute error                  0.1034
Root mean squared error             0.3152
Relative absolute error             20.7817 %
Root relative squared error        63.1897 %
Coverage of cases (0.95 level)      90.9654 %
Mean rel. region size (0.95 level)  51.2385 %
Total Number of Instances          25192
=== Detailed Accuracy By Class ===
                 TP Rate      FP Rate      Precision
Recall      F-Measure    ROC Area   Class
                   0.912         0.123           0.895
0.912          0.903       0.968       normal
                 0.877          0.088            0.897
0.877          0.887        0.963      anomaly
```

```
Weighted Avg.    0.896          0.106          0.896
0.896         0.896        0.966
=== Confusion Matrix ===
    a              b   <-- classified as
 12272          1177 |    a = normal
  1445         10298 |    b = anomaly
```

Fig 4.1 An Example of an output for Naïve Bayesian model

TABLEB1: 42 Attributes Naïve Bayes train data

```
Time taken to build model: 1.59 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly  Classified  Instances                  18200
80.731  %
Incorrectly  Classified  Instances                 4344
19.269  %
Kappa statistic                          0.623
Mean absolute error                     0.1924
Root mean squared error                 0.4371
Relative absolute error            39.2297 %
Root relative squared error      88.2712 %
Coverage of cases (0.95 level) 81.1657 %
Mean rel. region size (0.95 level)      50.3903 %
Total Number of Instances             22544
=== Detailed Accuracy By Class ===
                         TP  Rate      FP  Rate      Precision
Recall   F-Measure   ROC Area   Class
                           0.95          0.301              0.705
0.95       0.809            0.958       normal
                           0.699       0.05                 0.949
0.699     0.805            0.949       anomaly
Weighted  Avg.          0.807       0.158               0.844
0.807     0.807           0.953
=== Confusion Matrix ===
   a             b   <-- classified as
 9225        486 |    a = normal
 3858       8975 |    b = anomaly
```

TABLEB2: 42 Attributes Naïve Bayes for 20 percent test data

```
Time taken to build model: 0.78 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances                      7783
65.6793 %
Incorrectly Classified Instances                    4067
34.3207 %
Kappa statistic                                  0.2798
Mean absolute error                              0.3343
Root mean squared error                         0.562
Relative absolute error                            112.4401
%
Root relative squared error                        145.7894
%
Coverage of cases (0.95 level)                      73.5527
%
Mean rel. region size (0.95 level)          55.4684 %
Total Number of Instances                     11850
=== Detailed Accuracy By Class ===
                                    TP Rate   FP Rate   Precision
                              Recall     F-Measure    ROC Area
                              Class
                                 0.83             0.382          0.326
0.83        0.468          0.848          normal
                                 0.618          0.17             0.943
0.618      0.747          0.844          anomaly
Weighted  Avg.            0.657          0.208           0.831
0.657      0.696          0.845
=== Confusion Matrix ===
   a             b   <-- classified as
 1787          365 |    a = normal
 3702         5996 |    b = anomaly
```

TABLEB3: 42 Attributes for train data using trees – j48

```
Time taken to build model: 13.55 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances                       25081
99.5594 %
Incorrectly Classified Instances                        111
0.4406 %
Kappa statistic                               0.9911
Mean absolute error                          0.0064
Root mean squared error                     0.0651
Relative absolute error                     1.2854 %
Root relative squared error               13.059  %
Coverage of cases (0.95 level)          99.6229 %
Mean rel. region size (0.95 level)      50.2997 %
Total Number of Instances                 25192
=== Detailed Accuracy By Class ===
                                    TP     Rate           FP     Rate
Precision    Recall    F-Measure   ROC Area   Class
                                 0.996       0.004          0.996
0.996     0.996          0.998          normal
                                 0.996       0.004          0.995
0.996     0.995          0.998        anomaly
Weighted Avg.                0.996      0.004          0.996
0.996     0.996          0.998
=== Confusion Matrix ===
    a                   b   <-- classified as
 13389              60 |     a = normal
    51          11692 |     b = anomaly
```

TABLEB4: 42 attributes for test data using trees -j48

```
Number of Leaves  :          500
Size of the tree :   590
Time taken to build model: 4.05 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly   Classified   Instances                   11509
97.1224 %
Incorrectly   Classified   Instances                  341
2.8776 %
Kappa statistic                              0.9022
```

```
Mean absolute error                  0.0389
Root mean squared error              0.1552
Relative absolute error             13.1002 %
Root relative squared error         40.2591 %
Coverage of cases (0.95 level)      98.557 %
Mean rel. region size (0.95 level)  53.0338 %
Total Number of Instances           11850
=== Detailed Accuracy By Class ===
                              TP  Rate        FP  Rate
Precision   Recall   F-Measure   ROC Area   Class
                                 0.908              0.015
0.932       0.908    0.92          0.971       normal
                                 0.985    0.092     0.98
0.985    0.982        0.971       anomaly
Weighted  Avg.                       0.971       0.078
0.971       0.971    0.971         0.971
=== Confusion Matrix ===
    a              b   <-- classified as
 1953          199 |    a = normal
  142         9556 |    b = anomaly
```

TABLEB5: 42 attributes for train data using libsvm

```
Test mode:    2-fold cross-validation
=== Classifier model (full training set) ===
LibSVM wrapper, original code by Yasser EL-
Manzalawy (= WLSVM)
Time taken to build model: 1234.28 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly   Classified   Instances             23686
94.0219 %
Incorrectly   Classified   Instances             1506
5.9781 %
Kappa statistic                      0.8789
Mean absolute error                  0.0598
Root mean squared error              0.2445
Relative absolute error             12.0113%
Root relative squared error         49.0128%
Coverage of cases (0.95 level)      94.0219%
Mean rel. region size (0.95 level)   50%
Total Number of Instances           25192
=== Detailed Accuracy By Class ===
                        TP  Rate    FP  Rate     Precision
Recall    F-Measure   ROC Area   Class
                        0.999       0.128              0.9
0.999     0.947            0.936    normal
                        0.872      0.001             0.999
0.872     0.932            0.936    anomaly
Weighted Avg.          0.94       0.068             0.946
0.94      0.94             0.936
=== Confusion Matrix ===
     a                b   <-- classified as
 13442                7 |     a = normal
  1499            10244 |     b = anomaly
```

TABLEB6: 42 attributes for train data using libsvm

```
=== Classifier model (full training set) ===
LibSVM wrapper, original code by Yasser EL-
Manzalawy (= WLSVM)
Time taken to build model: 293.98 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly   Classified   Instances             10802
91.1561 %
Incorrectly   Classified   Instances             1048
8.8439 %
Kappa statistic                      0.6457
Mean absolute error                  0.0884
Root mean squared error              0.2974
Relative absolute error             29.7488%
Root relative squared error         77.1396%
Coverage of cases (0.95 level)      91.1561%
Mean rel. region size (0.95 level)  50%
Total Number of Instances           11850
=== Detailed Accuracy By Class ===
                 TP  Rate      FP  Rate      Precision
Recall   F-Measure   ROC Area   Class
                   0.551          0.008           0.936
0.551     0.693           0.771    normal
                   0.992          0.449           0.909
0.992     0.948           0.771    anomaly
Weighted  Avg.       0.912         0.369          0.914
0.912     0.902           0.771
=== Confusion Matrix ===
    a             b   <-- classified as
 1185           967 |    a = normal
   81          9617 |    b = anomaly
```

TABLEB7: 42 attributes for test data using function SMO

```
Time taken to build model: 523.19 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly   Classified   Instances             10880
91.8143 %
Incorrectly   Classified   Instances              970
8.1857 %
Kappa statistic                       0.6868
Mean absolute error                   0.0819
Root mean squared error               0.2861
Relative absolute error              27.5347%
Root relative squared error          74.2135%
Coverage of cases (0.95 level)       91.8143%
Mean rel. region size (0.95 level)    50%
Total Number of Instances            11850
=== Detailed Accuracy By Class ===
                          TP Rate   FP Rate   Precision
Recall      F-Measure   ROC Area      Class
                            0.619      0.016       0.898
0.619        0.733        0.802          normal
                            0.984      0.381       0.921
0.984        0.952        0.802         anomaly
Weighted Avg.                 0.918      0.314       0.917
0.918        0.912        0.802
=== Confusion Matrix ===
    a          b   <-- classified as
```

```
 1333      819 |   a = normal
  151     9547 |   b = anomaly
```

TABLEB8: 25 attributes Naïve Bayes train data

```
Time taken to build model: 6.72 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances              24234
96.1972 %
Incorrectly Classified Instances              958
3.8028 %
Kappa statistic                          0.9233
Mean absolute error                      0.0411
Root mean squared error                  0.1822
Relative absolute error                  8.2601 %
Root relative squared error             36.5231 %
Coverage of cases (0.95 level)          97.7612 %
Mean rel. region size (0.95 level)      52.4571 %
Total Number of Instances               25192
=== Detailed Accuracy By Class ===
                         TP Rate  FP Rate  Precision
Recall   F-Measure  ROC Area   Class
                         0.987    0.067      0.944
0.987    0.965       0.996     normal
                         0.933    0.013      0.985
0.933    0.958       0.996     anomaly
Weighted Avg.            0.962    0.042      0.963
0.962    0.962       0.996
=== Confusion Matrix ===
    a        b   <-- classified as
 13277     172 |   a = normal
   786   10957 |   b = anomaly
```

TABLEB9: 25 Attributes Decision Trees-J48 for train data

```
Number of Leaves:        262
Size of the tree:   314
Time taken to build model: 7.41 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances              25070
99.5157 %
Incorrectly Classified Instances              122
0.4843 %
Kappa statistic                          0.9903
Mean absolute error                      0.0075
Root mean squared error                  0.0687
Relative absolute error                  1.5046 %
Root relative squared error             13.7643 %
Coverage of cases (0.95 level)          99.5554 %
Mean rel. region size (0.95 level)      50.1707 %
Total Number of Instances               25192
=== Detailed Accuracy By Class ===
                    TP Rate   FP Rate    Precision
Recall   F-Measure  ROC Area   Class
                    0.996     0.006        0.995
0.996    0.995       0.996     normal
                    0.994     0.004        0.996
0.994    0.995       0.996     anomaly
Weighted  Avg.      0.995     0.005        0.995
0.995    0.995       0.996
=== Confusion Matrix ===
    a          b   <-- classified as
 13398        51 |   a = normal
    71     11672 |   b = anomaly
```

TABLE B10: 25 Attributes function LibSVM of train data

```
Test mode:    2-fold cross-validation
=== Classifier model (full training set) ===
LibSVM wrapper, original code by Yasser EL-
Manzalawy (= WLSVM)
Time taken to build model: 2757.88 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances              23023
91.3901 %
Incorrectly Classified Instances             2169
8.6099 %
Kappa statistic                          0.825
Mean absolute error                      0.0861
Root mean squared error                  0.2934
Relative absolute error                 17.2991 %
Root relative squared error             58.8202 %
Coverage of cases (0.95 level)          91.3901 %
Mean rel. region size (0.95 level)      50 %
Total Number of Instances               25192
=== Detailed Accuracy By Class ===


                  TP    Rate         FP    Rate
            Precision    Recall    F-Measure
            ROC Area    Class
                  1          0.184        0.861
1        0.925         0.908    normal
               0.816           0            1
0.816    0.898         0.908    anomaly
Weighted  Avg.    0.914       0.099        0.926
0.914    0.913         0.908
=== Confusion Matrix ===
    a       b   <-- classified as
 13446       3 |   a = normal
  2166    9577 |   b = anomaly
```

TABLE B11: 25 Attributes function SMO of train data

```
Number of kernel evaluations: 107619127
(45.871% cached)
Time taken to build model: 671.05 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances              24510
97.2928 %
Incorrectly Classified Instances              682
2.7072 %
Kappa statistic                          0.9455
Mean absolute error                      0.0271
Root mean squared error                  0.1645
Relative absolute error                  5.4394 %
Root relative squared error             32.9829 %
```

```
Coverage of cases (0.95 level)        97.2928 %
Mean rel. region size (0.95 level)     50 %
Total Number of Instances          25192
=== Detailed Accuracy By Class ===
                    TP  Rate     FP  Rate     Precision
Recall   F-Measure   ROC Area    Class
                      0.985          0.041            0.965
0.985    0.975            0.972         normal
                      0.959          0.015            0.983
0.959    0.971            0.972         anomaly
Weighted  Avg.         0.973        0.029           0.973
0.973    0.973            0.972
=== Confusion Matrix ===
    a          b   <-- classified as
 13250        199 |    a = normal
   483      11260 |    b = anomaly
```

TABLE B12: 19 Attributes Naïve Bayes train data

```
Time taken to build model: 0.69 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly   Classified   Instances              23216
92.1562 %
Incorrectly   Classified   Instances              1976
7.8438 %
Kappa statistic                          0.8418
Mean absolute error                      0.0793
Root mean squared error                  0.2757
Relative absolute error                 15.9327 %
Root relative squared error             55.2604 %
Coverage of cases (0.95 level)          92.9144 %
Mean rel. region size (0.95 level)       51.0162 %
Total Number of Instances               25192
=== Detailed Accuracy By Class ===
                    TP  Rate     FP  Rate     Precision
Recall    F-Measure   ROC Area     Class
                     0.952       0.114             0.906
0.952     0.928           0.973          normal
                     0.886       0.048             0.942
0.886     0.913           0.967         anomaly
Weighted  Avg.      0.922       0.083             0.923
0.922     0.921            0.97
=== Confusion Matrix ===
    a            b   <-- classified as
 12809        640 |    a = normal
  1336      10407 |    b = anomaly
```

TABLE B19: 19 Attributes Decision Trees-J48 for train data

```
Number of Leaves  :          189
Size of the tree :   237
Time taken to build model: 4.63 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly  Classified  Instances              25046
99.4205 %
Incorrectly   Classified   Instances              146
0.5795 %
Kappa statistic                        0.9884
Mean absolute error                    0.009
Root mean squared error                0.0745
Relative absolute error                 1.8057 %
Root relative squared error            14.9359 %
Coverage of cases (0.95 level)         99.5435 %
Mean rel. region size (0.95 level)     50.6828 %
Total Number of Instances              25192
=== Detailed Accuracy By Class ===
                     TP  Rate      FP  Rate      Precision
Recall    F-Measure   ROC Area      Class
                    0.996          0.008             0.993
0.996     0.995           0.996          normal
                    0.992          0.004             0.995
0.992     0.994           0.996         anomaly
Weighted  Avg.       0.994         0.006             0.994
0.994     0.994           0.996
=== Confusion Matrix ===
    a           b   <-- classified as
 13394         55 |    a = normal
    91      11652 |    b = anomaly
```

TABLE B14: 19 Attributes function LibSVM of train data

```
=== Classifier model (full training set) ===
LibSVM  wrapper,  original  code  by  Yasser  EL-
Manzalawy (= WLSVM)
Time taken to build model: 1187.69 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly   Classified   Instances              22743
90.2787 %
Incorrectly   Classified   Instances              2449
9.7213 %
Kappa statistic                         0.8021
Mean absolute error                     0.0972
Root mean squared error                 0.3118
Relative absolute error                19.5322 %
Root relative squared error            62.5016 %
Coverage of cases (0.95 level)         90.2787 %
Mean rel. region size (0.95 level)   50 %
Total Number of Instances           25192
=== Detailed Accuracy By Class ===
                      TP  Rate      FP  Rate      Precision
Recall    F-Measure   ROC Area       Class
                           1       0.208      0.846            1
0.917           0.896           normal
                       0.792          0                        1
0.792     0.884           0.896           anomaly
Weighted  Avg.         0.903        0.111            0.918
0.903     0.901           0.896
=== Confusion Matrix ===
    a          b   <-- classified as
 13446          3 |    a = normal
  2446       9297 |    b = anomaly
```

TABLE B15: 19 Attributes function SMO of train data

```
Number    of    kernel    evaluations:    117638676
(44.952% cached)
```

```
Time taken to build model: 631.05 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly   Classified   Instances                   24503
97.265  %
Incorrectly   Classified   Instances                   689
2.735  %
Kappa statistic                          0.945
Mean absolute error                      0.0273
Root mean squared error                  0.1654
Relative absolute error                  5.4952 %
Root relative squared error              33.1517 %
Coverage of cases (0.95 level)           97.265  %
Mean rel. region size (0.95 level)       50 %
Total Number of Instances                25192
=== Detailed Accuracy By Class ===
                      TP  Rate     FP  Rate     Precision
Recall   F-Measure   ROC Area    Class
                        0.985        0.041              0.965
0.985     0.975          0.972          normal
                        0.959        0.015              0.982
0.959     0.97           0.972          anomaly
Weighted  Avg.          0.973        0.029              0.973
0.973     0.973          0.972
=== Confusion Matrix ===
     a          b   <-- classified as
 13245       204 |    a = normal
   485     11258 |    b = anomaly
```

A. Discussion of Results

We now discuss the output obtained in fig 4.1. Our first experimental object was truncated from the original KDDCup'99 with 42 attributes. Each object is defined in 42 attributes, and belongs to one of the five classes: normal and any of these attacks: probe, denial-of service (DOS), unauthorized access to root (U2R) and unauthorized access from remote machine (R2L) referring to an Anomaly. The different classes of Attacks are as giving below:

1. **Denial of Service Attacks:**

In denial of service the attacker develops some computing or memory resource available or unavailable to manage valid requirements, or reject valid user's rights to use a machine.

2. **User to Root Attacks:**

In User to Root attack, the attacker initiate by using a normal user account on the system and take advantage of some vulnerability to achieve root access to the system.

3. **Remote to User Attacks:**

Remote to User attack takes place when an attacker has the ability to send packets to a machine over a network but does not have an account on that machine, performing some vulnerability to access as a user of that machine.

4. **Probes:**

Probing is a kind of attacks that takes place when an attacker checks a network to collect information or find out well-known threats. This information is helpful for an attacker who plans to make an attack in future. There are different types of probes such as abusing the system's legitimate features, using social engineering methods. However this type of attack requires few technical expertises.

Objects in the normal class are harmless connections, whereas objects in the other Anomaly class are different types of attacks. The training set contains 125,972 connections; the test data includes 22,543. The KDD Cup 1999 data set is the only large-scale, publicly available data for evaluating intrusion detection tools. A detailed description of the data set is available on the Attack Analysis sheet. We have used a subset of the 20% of the new NSL-KDDCup 2009, otherwise known as refined KDDCup '99 dataset as our train dataset and 21% of the test data. The test dataset is the same as that, which was used in evaluating classification algorithms in KDD-Cup 99 contest.

We normalized the train and test data sets, where each numerical value in the data set is normalized between 0.0 and 1.0. Fig.4.1 shows a complete output of Naïve Bayesian algorithm from WEKA software. All other parameters are taken from the output as the experiments were carried out for all the algorithms.

We will now continue with the discussion of the results using the following parameters:

Table 5. Cross Validation Summary.
Summary of train data for 42 attributes

Summarize Properties for 42 attributes	Naïve Bayes	J48 decision Trees	Lib SVM (SVM)	Optimized SVM(SMO)	
				2 fold cross validation	10 fold cross validation
Time taken to build the model(s)	1.6	13.55	1234.28	850.42	501.38
Incorrectly classified Instances	2622	111	1506	677	970
Correctly classified instances	22570	25081	23686	24515	10880
Total number of instances	25192	25192	25192	25192	11850
% of correctly classified instances	89.5919	99.5594	94.0219	97.3126	91.8143
% of incorrectly classified instances	10.4081	0.4406	5.9781	2.6874	8.1857
Kappa Statistics	0.7906	0.9911	0.8789	0.9459	0.6868

Fig. 3. A Graph of Different types of Algorithm as against time taken to build the model(s) for 42 Attributes

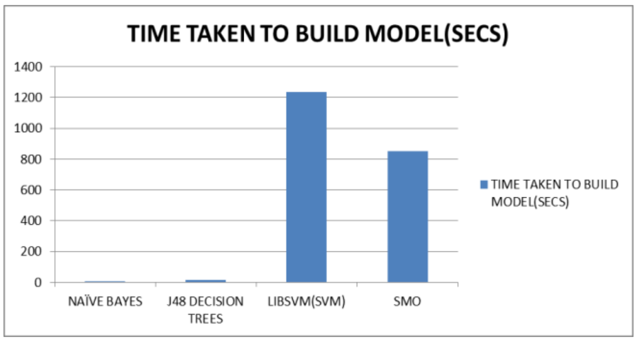


Table 6: Summary of train data for 25 attributes

Summarize Properties for 25 attributes	Naïve Bayes	J48 decision Trees	LibSVM (SVM)	Optimized SVM(SMO)
Time taken to build the model(s)	0.95	7.41	2757.88	671.05
Incorrectly classified Instances	2782	122	2169	682
Correctly classified instances	22410	25070	23023	24510
Total number of instances	25192	25192	25192	25192
% of correctly classified instances	88.9568	99.5157	91.3901	97.2928
% of incorrectly classified instances	11.0432	0.4843	8.6099	2.7072
Kappa Statistics	0.9233	0.9903	0.825	0.9455

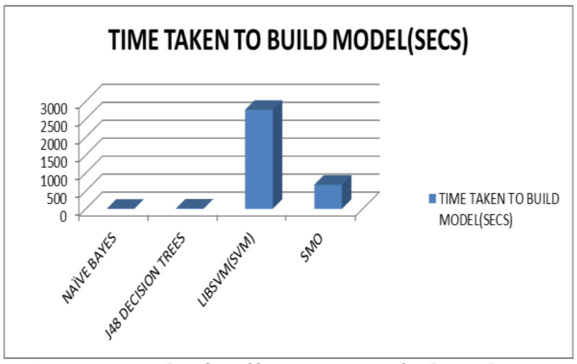


Fig 4. A Graph of Different types of Algorithm as against time taken to build the model (secs) for 25 attributes

Table 7. Summary of train data for 19 attributes

Summarize Properties for 19 attributes	Naïve Bayes	J48 decision Trees	Lib SVM (SVM)	Optimized SVM(SMO)
Time taken to build the model(s)	0.69	4.09	1187.69	631.05
Incorrectly classified Instances	1976	146	2449	689
Correctly classified instances	23216	25046	22743	24503
Total number of instances	25192	25192	25192	25192
% of correctly classified instances	92.1562	99.4205	90.2787	97.265
% of incorrectly classified instances	7.8438	0.5795	9.7213	2.735
Kappa Statistics	0.8418	0.9884	0.8021	0.945

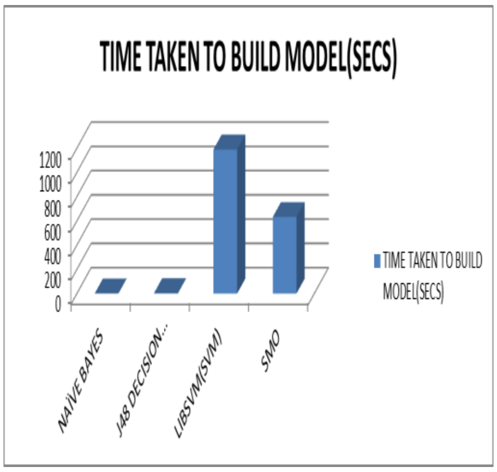


Fig 5. A Graph of Different types of Algorithm as against time taken to build the model (secs) for 19 attributes

A. Time taken to build the model.

Fig 4.2 - 4.4 shows time taken to build the model for every algorithm above. From the tables and charts, it is clear that Naïve bayes algorithm takes a shorter time to build a model, averaging between 1.66s for 42 attributes to 0.95s for 25 attributes and 0.69 for 19 attributes. This is closely followed by J48. LibSVM took longer time to build a model, this is due to the time it takes to build a quadratic functions. SMO took shorter time than LibSVM because the functions in SMO are decomposed into serial minimal Quadratic problems.

Kappa Statistics Algorithms Charts

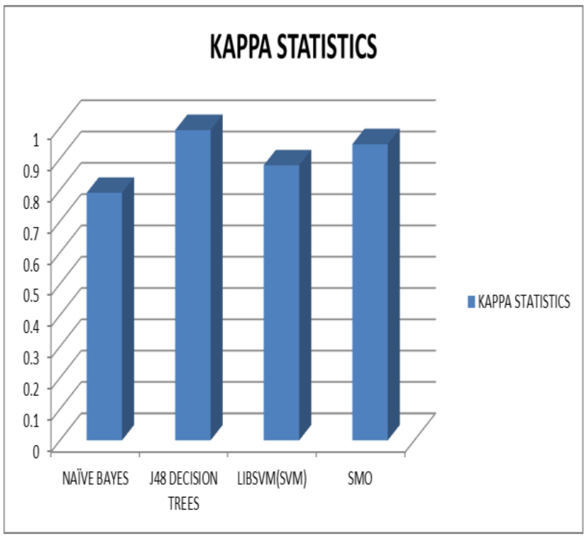


Fig 6. A Graph of different Algorithms against Kappa Statistic for 42 attributes

A. **Kappa Characteristics**

From the above Chart, The Kappa characteristic measures the agreement of a prediction to the true class. The higher the value of prediction; the better the prediction. Fig 4.5 shows that J48 has the highest Kappa characteristics hovering around 0.98 to 0.99, competitively followed by SMO and Naive Bayesian. The highest value that Kappa characteristics can go is a value of one.

DETAILED ACCURACY BY CLASS

Table 8. for Detailed accuracy by class for 42 attributes

Accuracy description	Naïve Bayes			J48			LibSVM			SMO		
	Nor.	Ano.	Aver	Nor.	Ano.	Aver	Nor.	Ano.	Aver	Nor.	Ano.	Aver
True positive	0.83	0.618	0.657	0.908	0.985	0.971	0.551	0.992	0.912	0.619	0.984	0.918
False Positive	0.382	0.17	0.208	0.015	0.092	0.078	0.008	0.449	0.369	0.016	0.381	0.314
Precision	0.326	0.943	0.943	0.932	0.98	0.971	0.936	0.909	0.914	0.898	0.921	0.917
Recall	0.83	0.618	0.618	0.908	0.985	0.971	0.551	0.992	0.912	0.619	0.984	0.918
F- measure	0.468	0.747	0.747	0.92	0.982	0.971	0.693	0.948	0.902	0.733	0.952	0.912
ROC Area	0.848	0.844	0.696	0.971	0.971	0.971	0.771	0.771	0.771	0.802	0.802	0.802

Table 9. for Detailed accuracy by class for 25 attributes

Accuracy description	Naïve Bayes			J48			LibSVM			SMO		
	Nor.	Ano.	Aver	Nor.	Ano.	Aver	Nor.	Ano.	Aver	Nor.	Ano.	Aver
True positive	0.911	0.865	0.89	0.996	0.994	0.995	1	0.816	0.914	0.985	0.959	0.973
False Positive	0.135	0.089	0.114	0.006	0.004	0.005	0.184	0	0.099	0.041	0.015	0.029
Precision	0.885	0.895	0.89	0.995	0.996	0.995	0.861	1	0.926	0.965	0.983	0.973
Recall	0.911	0.865	0.89	0.996	0.994	0.995	1	0.816	0.914	0.985	0.959	0.973
F- measure	0.898	0.88	0.889	0.995	0.995	0.995	0.925	0.898	0.913	0.975	0.971	0.973
ROC Area	0.968	0.962	0.965	0.996	0.996	0.996	0.908	0.908	0.908	0.972	0.972	0.972

Area under curve (AUC) chart

B. **Area under the curve (AUC)**

Fig7 show Bar graphs of AUC for all the set of experiments we carried out. For each set of attributes, J48 shows a higher value over and above other algorithms. Just like the Kappa characteristic is gauge with a value 1 as the highest, so is AUC. This again suggests that J48 is a better algorithm for data classifications.

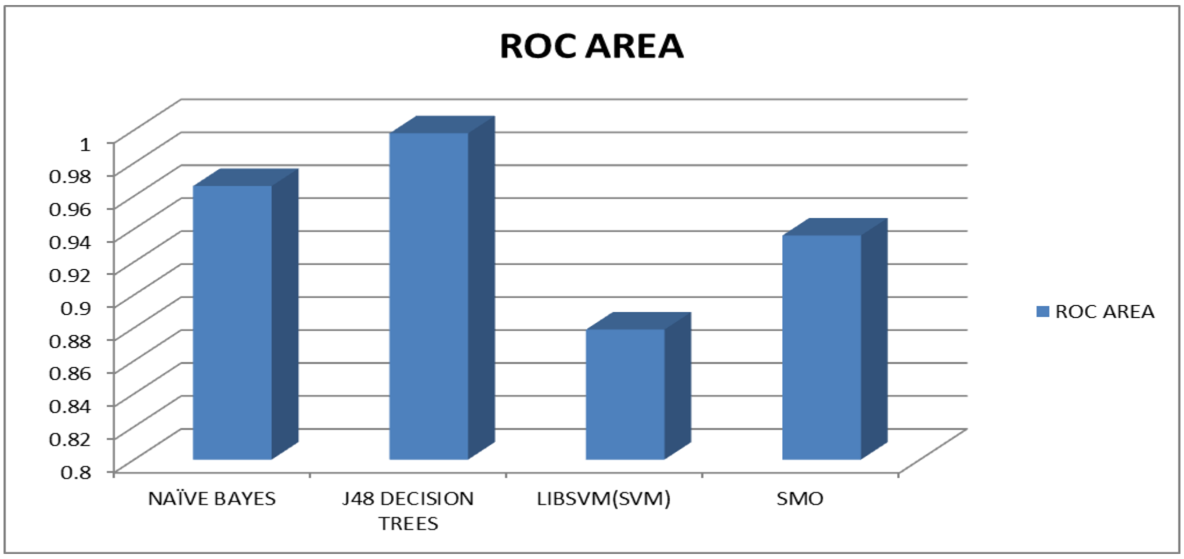


Fig 7. A Graph of different Algorithms against ROC

Table for Detailed accuracy by class for 42 attributes test data

Accuracy description	Naïve Bayes			J48			LibSVM			SMO		
	Nor.	Ano.	Aver	Nor.	Ano.	Aver	Nor.	Ano.	Aver	Nor.	Ano.	Aver
True positive	0.83	0.618	0.657	0.908	0.985	0.971	0.551	0.992	0.912	0.619	0.984	0.918
False Positive	0.382	0.17	0.208	0.015	0.092	0.078	0.008	0.449	0.369	0.016	0.381	0.314
Precision	0.326	0.943	0.943	0.932	0.98	0.971	0.936	0.909	0.914	0.898	0.921	0.917
Recall	0.83	0.618	0.618	0.908	0.985	0.971	0.551	0.992	0.912	0.619	0.984	0.918
F- measure	0.468	0.747	0.747	0.92	0.982	0.971	0.693	0.948	0.902	0.733	0.952	0.912
ROC Area	0.848	0.844	0.696	0.971	0.971	0.971	0.771	0.771	0.771	0.802	0.802	0.802

False positive rate chart

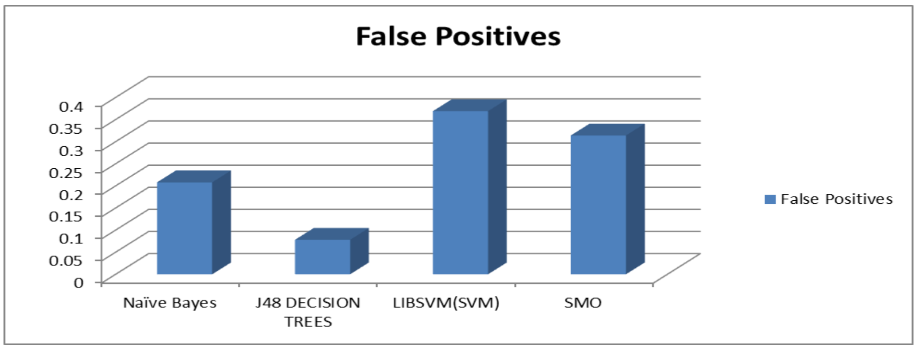


Fig 8. A Graph of Different Algorithms against False Positives 42 attributes test data

E. **False positive rate**

False positives rate gives the value of events that are wrongly classified. From the graphs in Fig 4.7, J48 has the lowest false positive rate as low as 0.078 as against LibSVM that recorded 0.37 at a time when we ran it on 42 attributes. This suggests that J48 algorithm has lower rates of errors than the other data mining Machines.

Receiver Operating Characteristics (ROC) Graph

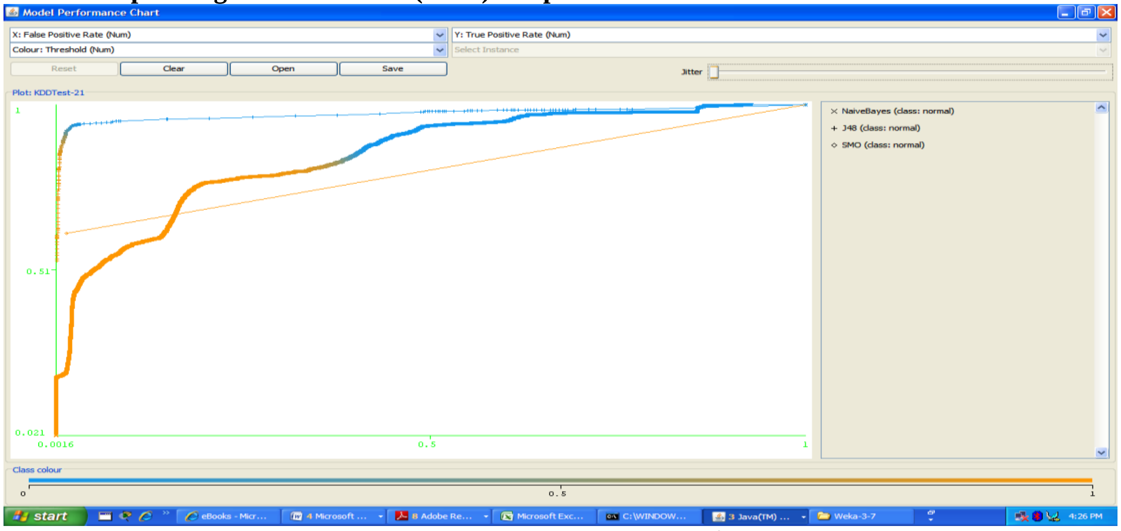


Fig 9. A graph comparing the ROC characteristics for Naïve Bayes, J48 and SMO

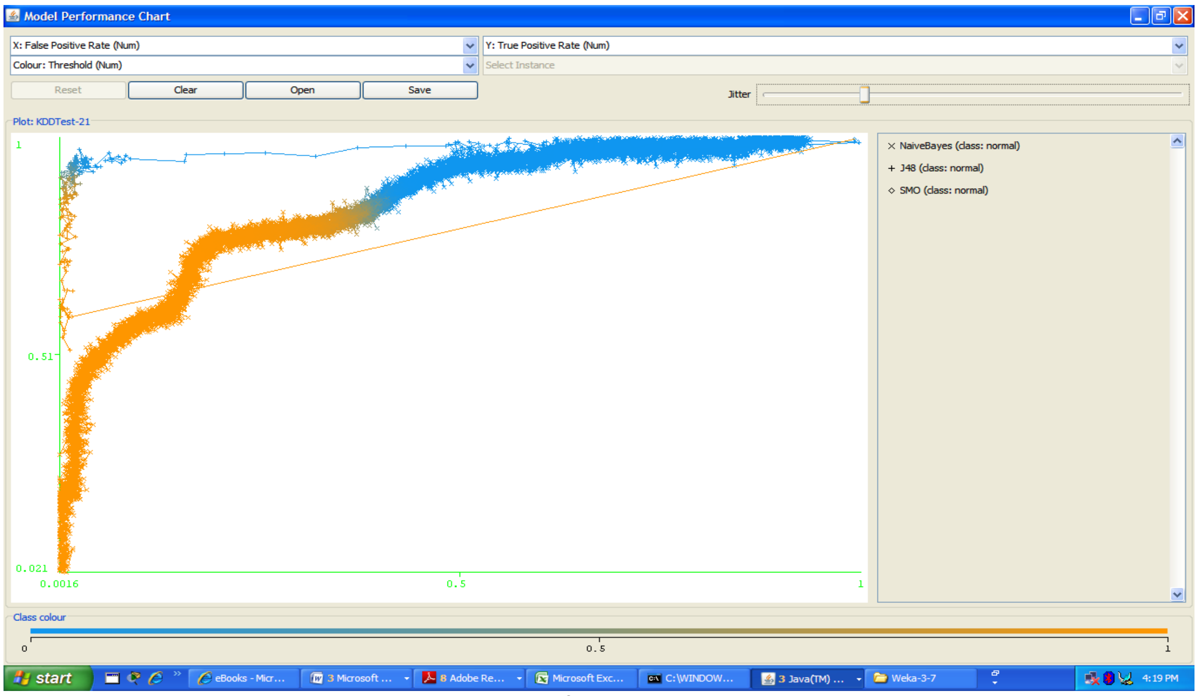


Fig 10. A graph comparing the ROC characteristics for Naïve Bayes, J48 and SMO.

F. Receiver Operating Characteristics (ROC)

Fig 4.14 shows the characteristics curve of three algorithms. We did not include LibSVM because of its scaling capability. As a quadratic function, it has linear graphs just as SMO, but SMO load faster than LibSVM. We are sure it is because of its serial minimal characteristics. It kept giving heap size error when ran the entire four algorithms together, not until we remove the LibSVM algorithm on WEKA knowledge flow. Since SMO and LibSVM are both quadratic functions one will suffice to use for the purpose of this comparative work.

The intersection between the Naïve Bayes and the linear graph of SMO is described as the point of equal error rate for the two algorithms. Equal error rate is the point where the usability of the two algorithms is as good as the security of the IDS for the network. Fig 4.8a is shows the graph for the three algorithm in a lower jitter setting, while in fig 4.8b we increase the jitter setting so as to magnified the graph for better clarity of understanding. The linear graph is the graph of SMO, the thicker line is for the Naïve bayes while the last on top with the + sign wind together is for J48 algorithm suggesting the best ever among the algorithm.

G. Experimenter

We set up the test bed for the experimenter using both train and test data to run two folds to be repeated twice. We tried to run it for ten folds to repeat 10 times and it took more than 24hrs to complete, even at completion it was not successfully. This, we attributed it to the quadratic nature of LibSVM function, because the log showed that it was counting for this algorithm.

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   1
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       2/21/12 11:43 PM
Dataset                  (1) trees.J4 | (2) bayes (3) funct
(4) funct
--------------------------------------------------------------
--------
KDDTest                  (4)   98.43 |   81.16 *   94.62
*   90.66 *
--------------------------------------------------------------
--------
                         (v/  /*) |      (0/0/1)      (0/0/1)
(0/0/1)
Key:
(1) trees.J48 '-C 0.25 -M 2' -217733168393644444
(2) bayes.NaiveBayes '' 5995231201785697655
(3) functions.SMO '-C 1.0 -L 0.0010 -P 1.0E-12 -N
0      -V      -1      -W      1      -K
\"functions.supportVector.PolyKernel -C 250007 -
E 1.0\"' -6585883636378691736
(4) functions.LibSVM '-S 0 -K 2 -D 3 -G 0.0 -R 0.0
-N 0.5 -M 40.0 -C 1.0 -E 0.0010 -P 0.1 -model
\"C:\\\\Program Files\\\\Weka-3-7\"' 14172
```

Fig 4.8a Output of the experimenter using KDDTest Dataset

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   1
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       2/22/12 1:46 AM
Dataset                  (1) trees.J4 | (2) bayes (3) funct
(4) funct
--------------------------------------------------------------
--------
KDDTrain-20Percent       (4)    99.47 |    89.73 *
97.31 *   94.00 *
--------------------------------------------------------------
--------
                         (v/  /*) |      (0/0/1)      (0/0/1)
(0/0/1)
Key:
(1) trees.J48 '-C 0.25 -M 2' -217733168393644444
(2) bayes.NaiveBayes '' 5995231201785697655
(3) functions.SMO '-C 1.0 -L 0.0010 -P 1.0E-12 -N
0      -V      -1      -W      1      -K
\"functions.supportVector.PolyKernel -C 250007 -
E 1.0\"' -6585883636378691736
(4) functions.LibSVM '-S 0 -K 2 -D 3 -G 0.0 -R 0.0
-N 0.5 -M 40.0 -C 1.0 -E 0.0010 -P 0.1 -model
\"C:\\\\Program Files\\\\Weka-3-7\"' 14172
```

Fig 4.8b Output of the experimenter using KDDTrain Data

We make our comparism using the percent correct statistic for the four methods which are displayed horizontally, numbered (1), (2), (3) and (4), as the heading of a little table. The labels for the columns are repeated at the bottom—trees.J48, bayes. Naivebayes, functions.SMO and functions. LibSVM, in case there is insufficient space for them in the heading the value in brackets at the beginning of the KDDTest as in fig 4.8a row (4) is the number of experimental runs: 2 times 2fold cross-validation. The percentage correct for the four schemes is shown in Figure 4.8a for test data and fig 4.15b. For the test data showed in fig 4.8a, 98.43% for method 1, 81.16% for method 2, 94.62% for method 3 and 90.66% for method 4. The symbol placed beside a result indicates that it is statistically better (*v*) or worse (*) than the baseline scheme. In this case *J48* at the specified significance level (0.05, or 5%) is better than other methods.
As shown, method 2 is significantly worse than method 1 because its success rate is followed by an asterisk. At the bottom of columns 2, 3 and 4 are counts (*x/y/z*) of the number of times the scheme was better than (*x*), the same as (*y*) or worse than (z the baseline scheme on the datasets used in the experiment. In this case there is only one dataset i.e. fig 4.8a method 2 was worse than method 1 (the baseline) once, method 3 was worse than it once, in same way method 4 was worse than the baseline once. (The annotation (*v/ /**) is placed at the bottom of column 1 to help you remember the meanings of the three counts (*x/y/z*). The explanation is applied to fig 4.8b.

VI. CONCLUSION

We have studied four well known Algorithm using WEKA work bench, including Experimenter and ROC, the performance or the choice as regards a learning algorithm for data mining in Intrusion detection systems was also achieved in the study. The key motivation for the use of data mining method in intrusion detection is enhance automation. Data mining technologies, such as decision tree (DT), naïve Bayesian classifier (NB), Sequential minimal optimization (SMO), support vector machine (SVM), k-nearest neighbors

(KNN), fuzzy logic model, and genetic algorithm. Which is widely used to analyze network logs in order to enhance intrusion related knowledge and improve the performance of IDS. Data mining provide decision support for intrusion management, and also help IDS in the detecting of new vulnerabilities and intrusions by discovering unknown patterns of attacks or intrusions.

We have achieved the following in this paper, from the design and implementation of IDS: Firstly, the ability of IDS to be able to detect attacks and the percentage of false alarms, ease of use, most secure and interoperability J48 version of the decision trees is better. This is because it has the lowest false positives, highest ROC areas and detection rates. Another approach to this parlance is the analysis using Kappa characteristics which by implication shows that J48 has the higher characteristics. We went further to plot a graph of false positives against true positives for the algorithms and confirm that J48 graph is higher and better than other Algorithms. Efforts were also in using Experimenter in the same WEKA bench, which again proved J48 to have a better (higher) value, and by implication the best of the four. It is only in efficiency (time taken to process information) that Naïve Bayes shows that it can be relied on, than other Algorithm. In all the combination of attributes Naïve bayes was faster than J48 in about 8 times.

Finally, J48 algorithm proved to be the best of the four methods of classifiers investigated, it was closely and competitively followed by SMO and Naïve Bayesian. In all parameters employed on the work bench, SVM shows the worst of the three algorithms in the entire different test carried out. Thus, we conclude that J48 is a reliable approach for generating a high good classification system for a given data. The only aspect that Naïve Bayes showed superiority over others is in the area of the time taken to build model.

VII. REFERENCES

[1] Adetunmbi, A. O., Adeola S. O. & Daramola, O. A. (2010). "Analysis of KDD '99 Intrusion Detection Dataset for Selection of Relevance Features," *Proceedings of the World Congress on Engineering and Computer Science. San Francisco, USA.*3(4); 101-105

[2] Alan, B., Chandrika, P., Rasheda, S. & Boleslaw S. (2002), "Network-Based Intrusion Detection Using Neural Networks," *In Proceedings of the Intelligent Engineering Systems through Artificial Neural Networks, St.Louis, ASME Press, New York.* 1(2);579-584,

[3] Aly, Ei-S., Janica, E., Jesus, G-P. & Mauricio P. (2006). "Applying Data Mining of Fuzzy Association Rules to Network Intrusion Detection", *In the Proceedings of Workshop on Information Assurance United States Military Academy, IEEE Communication Magazine, West Point, NY,DOI:10.1109/IAW/652083.*22-31

[4] Amir, A., Alasti, A., Ahmad, H. N. & Hadi, B. (2011). "A New System for Clustering & Classification of Intrusion Detection System Alerts Using SOM," *International Journal of Computer Science & Security,* 4(6);589-597.

[5] Anderson, J. P. (1980). "Computer Security Threat Monitoring & Surveilance", *Technical Report, Fort Washington, Pennsylvania.*3(4);86-92.

[6] Buntod, P. C., Suksringam, P., & Singseevo, A. (2010). "Effects of learning environmental Education on science process skills and critical thinking of mathayomsuksa 3 students with different learning achievements," *J.Soc. Sci.,* 6(1);60-63.

[7] Chen, P. H., Lin, C. J., & Schkopf, B. (2005). "A tutorial on m-support vector machines," *National Taiwan University.*22-32.

[8] Christopher, J. C. Burges (1998). "A Tutorial on Support Vector Machines for Pattern Recognition," *Retrived on June 27, 2012 from http://www.burges@lucent.com/Bell Laboratories, Lucent Technologies. Kluwer Academic Publishers, Boston. Manufactured in the Netherlands.*

[9] Denning, D.E (1987). "An Intrusion Detection Model," *Transactions on Software Engineering, IEEE Communication Magazine,* 13(1);222-232.

[10] Dewan, Md. F. & Mohammed, Z. R. (2010). "Anomaly Network Intrusion Detection Based on Improved Self Adaptive Bayesian Algorithm," *Journal of Computers,* 5(1);23-31.

[11] Dewan, Md. F., Nouria, H. & Mohammad, Z. R. (2010). "Combining Naive Bayes and Decision Tree for Adaptive Intrusion Detection," *International journal of network security & its applications (ijnsa),* 2(2).

[12] Diego, Z. (2001). "Using internal sensors for computer intrusion detection," *Ph.D. Dissertation, Purdue University.* 22-32

[13] Han, J. & Kamber, M. (2000). "Data Mining: Concepts and Techniques,"(1st ed.). *San Francisco:Morgan Kaufmann Publisher.*223-231

[14] Harley K. (2003). "Intrusion Detection: Host-Based and Network-Based Intrusion Detection Systems," *An Independent Study.*56-60

[15] Jacob, W., Ulvila & John, E. G. (2003). "Evaluation of Intrusion Detection Systems," *Journal of Research of the National Institute of Standards and Technology. J. Res. Natl Jacob W. Ulvila and John E. Gaffney, Jr.. Inst.Stand. Technol.* 108(6);453-473.

[16] Jake, R., Lin, M- J. & Risto, M. (1998). "Intrusion Detection With Neural Networks," *Advances In Neural Information Processing System 10, Cambridge,* MA:MIT Press, DOI:10.1.1.31. 3570. 32-44.

[17] Jin-Ling, Zhao, Jiu-fen, & Zhao, Jian-Jun Li (2005). "Intrusion Detection Based on Clustering Genetic Algorithm," *In Proceedings of International Conference on Machine Learning & Cybernetics (ICML) IEEE Communication Magazine,* ISBN:0-7803-9091-1,DOI: 10.1109/ICML.1527621. 23(4); 24 - 33

[18] Knowledge discovery in databases (1999). " DARPA archive. Task Description". *Retrived on May 22, 2012 from* http://www.kdd.ics.uci.edu/databases/kddcup99/task.html

[19] Mahbod, T., Ebrahim, B., Wei Lu, & Ali, A. G. (2009). "A Detailed Analysis of the KDD CUP 99 Data Set," *proceedings of the IEEE Symposium on Computational intelligence in security and defense applications.* 73 - 119

[20] Matthew, V. M. & Philip, K. C. (2003). "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," *Computer Science Department, Florida Institute of Technology,150 W. University Dr., Melbourne, Florida 329012,* (3);686-692.

[21] Mahoney, M. & Chan, P. (2003). "An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection," *Proceedings of Recent Advances in intrusion detection (RAID) Pittsburg, USA.* 3(4); 55-76.

[22] MeeraGandhi, G., Kumaravel, A., Srivatsa, S. K. (2010). **"**Effective Network Intrusion Detection using Classifiers Decision Trees and Decision rules," *Int. J. Advanced Networking and Applications.*2(3);686-692.

[23] MIT Lincoln Laboratory (2011). "DARPA Intrusion detection Evaluation," *Retrived on May 7, 2012 from* http://www.ii.mit.edu.org

[24] Mohammad, S. A. & Jafar, H. (2008). "Computer Intrusion Detection Using an Iterative Fuzzy Rule Learning Approach" 10 – 4.

[25] Mrutyunjaya, P. & Manas, R. P. (2007). "Network intrusion detection using Naïve Bayes," *Department of E &TC Engineering, G.I.E.T., Gunupur, India and Department of Computer Science, Berhampur University, Berhampur, India IJCSNS International Journal of Computer Science and Network Security,*7(12).

[26] Mr. Hemant, H. P. & Ketan, J. S. (2011). "Analysis of Data mining Algorithm in Intrusion Detection," *International Journal of Emerging Technology and Advanced Engineering.* 2(3);686-692.

[27] Muamer, N. M., Norrozila, S. & Emad T. K. (2011). "A Novel Local Network Intrusion Detection System Based on Support Vector Machine," *Journal of Computer Science Publications by Faculty of Computer Systems and Software Engineering, University Malaysia Pahang, Kuantan 26300, Malaysia Muamer.* 7 (10);1560-1564.

[28] Nadiammai, G.V., Krishnaveni, S. & Hemalatha, M. (2011). "A Comprehensive Analysis and study in Intrusion Detection System using Data Mining Techniques," *International Journal of Computer Applications.* 35(8);0975 – 8887.

[29] Nahla, B. A., Salem, B. & Zied, E. (2003). "Naive Bayes vs Decision Trees in Intrusion Detection Systems," *Institute Supérieur de Gestion41 Avenue de la libert´e 2000 Le Bardo, Tunisie* 3(5); 60 – 87.

[30] Nashville, T. L. (2007). "An evaluation of machine learning techniques in intrusion Detection," 31 - 40

[31] Nello, C. & John, S-T. (2005). "An Introduction to Support Vector Machines and Other Kernel –based Learning Methods," *China Machine Press.* 1 - 123

[32] Nsl-kdd (2009). "Data set for network-based intrusion detection systems". *Retrived on July 25, 2012 from* http://nsl.cs.unb.ca/NSL-KDD

[33] Oswais, S., Snasel, V., Kromer, P. & Abraham, A. (2008). "Survey: Using Genetic Algorithm Approach in Intrusion Detection Systems Techniques," *In the Proceedings of 7th International Conference on Computer Information & Industrial Management Applications (CISIM), IEEE Communication Magazine,*49(1)300-307.

[34] Paxson, V. Bro. (1999). "A System for Detecting Network Intruders in Real-Time," *Computer Networks,* 31(14);2435-2463.

[35] Peter, M., Vincent, H. L., Josh, H. & Marc, Z. (2004). "An Overview of Issues in Testing Intrusion Detection Systems," *National Institute of Standards and Technology ITL and Massachusetts Institute of Technology Lincoln Laboratory.3(4);334-338.*

[36] Provost, F. & Fawcett, T. (2001). "Robust classification for imprecise environment," *Machine Learning,* 42(3);203-231.

[37] Richard, P. L., David, J., Fried, Isaac, G., Joshua, W. H., Kristopher, R. K., David, McClung,

Dan W., Seth E. W., Dan W., Robert K. C. & Marc A. Z. (2000). "Evaluating Intrusion Detection Systems": The 1998 DARPA Off-line Intrusion Detection Evaluation. Lincoln Laboratory MIT, 244 Wood Street, Lexington, MA 02173-9108. Institute of Electrical and Electronics Engineers. *Reprinted from Proceedings DARPA Information Survivability Conference and Exposition (DISCEX) 2000*, IEEE Computer Society Press, Los Alamitos, CA.

[38] Sapna, S., Kaushik & Deshmukh, P. R. (2011). "Comparisons of approaches to implement intrusion detection system," *International Journal of Computer Science and Communication* 2(1);45–48.

[39] Sathyabama, S., Irfan A.M.S, Saravanan, A. (2011). "Network Intrusion Detection Using Clustering: A Data Mining Approach," *International Journal of Computer Application* 30(4);0975-8887. ISBN: 978-93-80864-87-5, DOI: 10.5120/3670-5071.

[40] Sekeh, M. A. & Bin Maarof, M. A. (2009). "Fuzzy Intrusion Detection System Via Data Mining with Sequence of System Calls," *In the Proceedings of International Conference on Information Assurance & security* (IAS), IEEE Communication Magazine, 32(1);154-158.

[41] Shilendra, K., Shrivastava & Preeti, J. (2011). "Effective Anomaly Based Intrusion Detection Using Rough Set Theory & Support Vector Machine," 18(3);0975-8887, DOI: 10.5120/2261-2906.

[42] Srinivas, M., Andrew, H. Sung & Ajith A.(2004). "Intrusion Detection Using an Ensemble of Intelligent Paradigms," *Journal of Network & Computer Applications*,1- 15.

[43] Taeshik, S. & Jong, S. Moon (2007). "A Hybrid Machine Learning Approach to Network Anomaly Detection," *Information Sciences,* 177(18);3799-3821, *Publisher: USENIX Association, ISSN:00200255,DOI:10.1016/j.ins.*

[44] Tang,Y. & Lixin, XU (2008). "Research of the Network Intrusion Detection Method Based on Support Vector Machine School of Astronautic Science and Technology," *Beijing Institute of Technology, 100081 International Symposium on Photo electronic Detection and Imaging: Image Processing, P*roc. of SPIE 18(1);6623 - 6631

[45] Teng, H. S., Chen, K. & Lu, S. C. (1990). "Adaptive Real-Time Anomaly Detection using Inductively Generated Sequential Patterns," *In the Proceedings of Symposium on research in Computer Security & Privacy, IEEE Communication Magazine*,278-284.

[46] Weka 3: "Data mining software in java". *Retrived on June 27, 2012 from http://www.cs.waikato.ac.nz/ml/weka/.*

[47] Ahmad M.A, Woodhead S. (2015), Containment of fast scanning computer network worms, international conference on internet and distributed computing systems, 235-247, 2015.

[48] Ahmad M.A,Woodhead S. Gan D. (2016), The V-network testbed for malware analysis 2016 international conference on advanced communication control and technology, 2016.

[49] Ahmad M.A,Woodhead S. Gan D. (2016), A safe guard against fast self-propagating malware, proceedings of the 6th international conference on communication and networks, 2016.

[50] Ahmad M.A,Woodhead S. Gan D. (2016), Early containment of fast network worm malware, IEEE 2016.