International Journal of Education and Science Research Review
ISSN 2348-6457
Volume-1, Issue-5
October- 2014
www.ijesrr.org
Email- editor@ijesrr.org

AI Based Classifiers-An Empirical Analysis

Jabarweer Singh , Gulshan Kumar
Assistant professor
SBS State Technical campus
Ferozepur(Punjab)-India

Krishan Kumar
Associate professor
SBS State Technical campus
Ferozepur(Punjab)-India

ABSTRACT:
Intrusion Detection Systems (IDS) are implemented over host sonnet work to categorize the activities taking place as normal or malevolent. The detection unit of IDS employs various methods for classification of these activities in to one of the five categories: Normal, Probe, DoS, User to root(U2R)and Remote to local(R2L).These techniques are mainly statistical techniques, knowledge based techniques or artificial intelligence(AI)based techniques.AI based classifiers are flexible, capable of learning, adaptive and speedy, for these reasons AI based techniques are more appropriate for intrusion detection than conventional approaches. There is no existing critical analysis of AI based classifiers in literature to highlight promising classifiers for intrusion detection. The existing studies evaluated few classifiers by using either different bench mark data sets or different subsets of the data set.
In this paper, first we study various AI based techniques. Then, we evaluate performance of these classifiers on the NSL-KDD data set and compare them empirically using various measures. We find promising classifiers for each class of instances present in the bench mark data set.

Keywords–Intrusion Detection System, ensemble, combination, weighted voting, artificial intelligence.

I. INTRODUCTION:

Intrusion can be defined as any set of actions that attempt to compromise the security objectives [1]. These attempts have to be detected, so that necessary action could be taken. One method is to classify all of the activities into normal and attack. The classification of the network traffic in computer networks is done by using Intrusion Detection Systems. Intrusion Detection (ID) is the method of inspection of the events taking place on a system or network and investigating them for intrusions, like non legitimate access, activity, or file manipulation. This includes three main courses of action: Monitoring and analyzing traffic; detection the intrusive activity; alarming the network administrator or taking some predefined actions. Intrusion Detection System (IDS) is software that computerizes the intrusion detection procedure and sense potential intrusions. Intrusion Detection Systems provide three crucial safety measures: they examine, detect, and react to illegal activity by insiders and outsiders of the network [2]. The network traffic is examined for distrustful activity and a signal is generated by the IDS. For examining the network, a huge amount of activity data is collected from the network generating large log files and raw network traffic data (in which human inspection is impossible) and then these activity data are compressed into high-level events, called attributes. Over it, a set of attributes is obtained and monitored by the IDS in order to detect intrusion attempts [1]. Intrusion Detection Systems can be classified into various types based upon different criteria. IDS can be classified based upon five criteria: Based upon information source (host based, network Based); Based upon Analysis strategy (anomaly detection, misuse detection); Based upon Time aspects (real-time prediction, offline prediction); Based upon Architecture (centralized, distributed); Based upon Response (active, passive) [3].

The detection approach used is implemented in the brain of the IDS i.e. Detection and analysis unit [1]. The accuracy of classification of the network traffic by IDS depends upon the accuracy of the approach implemented in the detection and analysis unit. There are two primary approaches for analyzing events to detect attacks; namely misuse detection and anomaly detection. Misuse detection is based on extensive knowledge of known attacks and system vulnerabilities provided by human experts. The misuse detection approaches look for hackers that attempt to perform these attacks and/or to exploit known vulnerabilities. Although the misuse detection can be very accurate in detecting known attacks, misuse detection approaches cannot detect unknown and emerging cyber threats. Anomaly detection, on the other hand, is based on the analysis of profiles that represent normal behavior of users, hosts, or network connections. Anomaly detectors characterize normal legitimate computer activity using different techniques and then use a variety of measures to detect deviations from de-fined normal behavior as potential anomaly. The major benefit of anomaly detection algorithms is their ability to potentially recognize unforeseen attacks. However, the major limitation is potentially high false alarm rate. Note that deviations detected by anomaly detection algorithms may not necessarily represent actual attacks, as they may be new or unusual, but still legitimate, network behavior [3].The main types of techniques used in intrusion detection are statistical based techniques, knowledge based techniques and artificial intelligence based techniques etc. Ponce (2004) has listed several advantages of using AI based techniques over other approaches [4]. The major advantages include Flexibility; Adaptability; Pattern recognition; fast computing; high detection accuracy of new attacks. AIs can learn new rules automatically, whereas in traditional systems the security administrator must add new rules for each new attack type or each new allowed program.

Most researchers used a single technique for the classifications of all five classes of traffic [5]. It is significant to query this method that attempts to recognize a single classifier that can detect instances of all five classes accurately. These five classes are: normal and other attack categories including DoS, Probing, R2L, and U2R, have noticeably exclusive execution dynamics and signatures, which motivates to find classifiers that are likely to exhibit superior performance for a given class type. Considering this option, we perform training and testing of a comprehensive set of machine learning algorithms and select best performing algorithms for each category present in the dataset.

II. AI BASED TECHNIQUES APPLIED TOINTRUSION DETECTION

In this section an overview of some major AI based techniques is given, in this work seventeen AI based classifiers evaluated, hence a brief introduction to those techniques is given:

A. Bayesian classifiers

1) **Bayes** Net: A Bayesian network (BN) consists of a directed acyclic graph G and a set P of probability distributions, where nodes and arcs in G stand for arbitrary variables and direct associations among variables correspondingly, and P is the set of local distributions for every node. A local distribution is usually specified by a conditional probability table (CPT). BNs are frequently used for the classification problem. In the classification learning problem, a learner tries to build a classifier from a certain set of labelled training instances that are characterized by a tuple of attribute variables used together to forecast the value of the class variable [6].

2) **Naive Bayes:** The Nave Bayes classifier method is derived from the Bayesian theorem and is mainly appropriate when the dimensionality of inputs is large. It is an uncomplicated classifier but can frequently do better than more complex techniques. The Nave Bayesian classifier presents a straightforward approach, with comprehensible semantics, to represent, using, and learning probabilistic knowledge. The technique is designed

in support of supervised induction tasks, where the performance objective is to precisely forecast, the class of test instances and in which the training instances include class information. In a way, it is a specialized form of a Bayesian network. It is named nave for the reason that it relies on two hypotheses. It supposes that the predictive attributes are conditionally independent given the class, and it posits that no hidden or latent attributes manipulate the prediction procedure. These assumptions maintain very proficient algorithms for both classification and learning [7].

B. Functions based classifiers

1) **SMO (Sequential Minimal Optimization**): Sequential Minimal Optimization or SMO is an algorithm for training Support Vector Machines. Training a Support Vector Machine (SVM) needs the solution of a very big quadratic programming (QP) optimization problem. SMO splits this outsized QP problem into a chain of minimum achievable QP problems. These tiny QP problems are resolved rationally, which keeps away the need of using a time-consuming numerical QP optimization as an inner loop. The quantity of memory needed for SMO is linear in the training set size, which permits SMO to work with very huge training sets. SMO's computation time is subjugated by SVM evaluation; hence SMO is best for linear SVMs and sparse data sets in terms of speed [8].

2) **Simple logistic:** Simple logistic is used for the construction of linear logistic regression models. Logit Boost with simple regression functions as base learners are used for fitting the logistic models. The optimal quantity of Logit Boost iterations to execute is cross-validated and automatic attribute selection is done. Firstly, Logit Boost is executed on the entire set of data to construct a logistic regression model for the root node. The number of iterations to be used is calculated by a fivefold cross-validation. In each fold, Logit Boost is run on the training set up to a maximum number of iterations (200). The number of iterations generating the minimum sum of errors on the test set over all five folds is used in Logi tBoost, for all the data to produce the model for the root node and is also used to build logistic regression models at all nodes in the tree and data are split by using C4.5 algorithm. Logistic regression models are then built at the child nodes on the corresponding subsets of the data using Logit Boost. As long as at least 15 instances are present at a node and a useful split is found, then splitting and model building is continued in the same fashion. The CART cross-validation-based pruning algorithm is applied to the tree [9].

3) **MLP (Multilayer Perception):** MLP uses back propagation for the purpose of classification of the examples. These networks can by generated by human, by algorithm or by using both approaches. At the time of their training these networks can be modified [10]. MLP is widely used in various pattern recognition problems. A MLP network consists of an input layer including a set of sensory nodes as input nodes, one or more hidden layers of computation nodes, and an output layer of computation nodes. Each interconnection has associated with it a scalar weight which is adjusted during the training phase. In addition, the back propagation learning algorithm is usually used to train a MLP, which are also called as back propagation neural networks. First of all, random weights are given at the beginning of training. Then, the algorithm performs weights tuning to define whatever hidden unit representation is most effective at minimizing the error of misclassification [11].

4) **RBF (Radial Basis Function**) Network: Radial Basis Function (RBF) neural networks are extensively used category of feed forward neural networks. They carry out the classification by calculating the distances between the inputs and the centers of the RBF hidden neurons, this leads to higher speed far better than back-propagation, and also they solve problems with large sample size more precisely [10]. RBF Networks are derived from the theory of function approximation, but they take a little unusual approach. They are two-layer feed-forward networks. The hidden nodes employ a set of radial basis functions (e.g. Gaussian functions). The

output nodes implement linear summation functions as in an MLP (Multilayer Perception). The network training is divided into two stages: first the weights from the input to hidden layer are determined, and then the weights of the hidden to output layer. The training/learning is very fast. The networks are very good at interpolation [12].

C. Lazy (Instance-based Learner)

1) IBK: Instance based K-nearest neighbors (IBK) use instance-based learning that generates classification predictions using only specific instances. Instance-based learning algorithms do not maintain a set of abstractions derived from specific instances. This is a K-nearest neighbor classifier. IBK can select appropriate value of K based on cross-validation, and distance weighting can also be done using IBK [13]. It works on the principle that first plot each training instance and then measure the distance of each test instance to the training instances. The class of the training instance with the least distance between it and the test instance is the class that we assign to the test instance. Basically k is chosen to be an odd number, and we take the smallest average distance of the k instances [14].

2) K Star: K Star is an instance-based learner which uses en-tropy as a distance measure. K* is an instance-based classifier, that is the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function. The underlying assumption of instance-based classifiers such as K*, IB1, PEBLS, etc., is that similar instances will have similar classes [15].

D. Miscellaneous: VFI (Voting Feature Intervals)

This method is a new algorithm called VFI. A concept is represented by a set of feature intervals on each feature dimension separately. Each feature participates in the classification by distributing real-valued votes among classes. The class receiving the highest vote is declared to be the predicted class. VFI is compared with the Naive Bayesian Classifier, which also considers each feature separately. Experiments on real-world datasets show that VFI achieves comparably, and even better than NBC in terms of classification accuracy. Moreover, VFI is faster than NBC on all datasets [16].

E. Rule based

1) J Rip: This class implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which was proposed by Cohen (1995) as an optimized version of IREP [17]. The algorithm is briefly described as follows as[18]: Initialize RS = , and for each class from the less prevalent one to the more frequent one, DO:

Building stage: Repeat a and b until the description length (DL) of the ruleset and examples is 64 bits greater than the smallest DL met so far, or there are no positive examples, or the error rate ¿= 50 percent (a) Grow phase: Grow one rule by greedily adding antecedents (or conditions) to the rule, until the rule is perfect. The procedure tries every possible value of each attribute and selects the condition with highest information gain: p (log (p/t) -log (P/T)). (b) Prune phase: Incrementally prune each rule and allow the pruning of any final

Sequences of the antecedents; The pruning metric is (p-n) / (p+n) – but it's actually 2p/ (p+n) -1, so in this implementation we simply use p/ (p+n).

Optimization stage: after generating the initial rule set Ri, generate and prune two variants of each rule Ri from randomized data using procedure a and b. But one variant is generated from an empty rule while the other is generated by greedily adding antecedents to the original rule. Moreover, the pruning metric used here is (TP+TN) / (P+N). Then the smallest possible DL for each variant and the original rule is computed. The variant with the minimal DL is selected as the final representative of Ri in the rule set. After all the rules in Ri have been examined and if there are still residual positives, more rules are generated based on the residual positives

using Building Stage again.
Delete the rules from the rule set that would increase the DL of the whole rule et if it were in it and add resultant rule set to RS.
END DO.

2) **PART:** Frank et al. (1998) had shown how good rule sets can be learned one rule at a time, without any need for global optimization. They present an algorithm (PART) for inferring rules by repeatedly generating partial decision trees, thus combining the two major paradigms for rule generation: creating rules from decision trees and the separate-and-conquer rule-learning technique. The algorithm is straightforward and elegant, despite this, experiments on standard datasets show that it produces rule sets that are as accurate as and of similar size to those generated by C4.5, and more accurate than RIPPERs. Moreover, it operates efficiently, and because it avoids post processing, does not suffer the extremely slow performance on pathological example sets for which the C4.5 method has been criticized. It adopts the separate-and-conquer strategy in that it builds a rule, removes the instances it covers, and continues creating rules recursively for the remaining instances until none are left. In essence, to make a single rule a pruned decision tree is built for the current set of instances, the leaf with the largest coverage is made into a rule, and the tree is discarded. This avoids hasty generalization by only generalizing once the implications are known [19].

3) **Ridor (Ripple-Down Rules):** Ripple down rules form a binary decision tree that differs from standard decision trees in that compound clause are used to determine branching, and these clauses need not exhaustively cover all cases so that it is possible for a decision to be reached at an interior node. This contrasts with standard trees where all decisions are made at root nodes. However, the feature of standard decision trees is retained that one, and only one, decision node is activated for each case. This makes maintenance simple because if the decision reach is erroneous then only node, and the past cases that have fallen under it, need be considered. It generates a default rule first and then the exceptions for the default rule with the least (weighted) error rate. Then it generates the "best" exceptions for each exception and iterates until pure. Thus it performs a tree-like expansion of exceptions. The exceptions are a set of rules that predict classes other than the default. IREP is used to generate the exceptions [20].

F. Decision Trees

1) J48: The J48 Decision tree classifier follows the following simple algorithm. In order to classify a new item, it first needs to create a decision tree based on the attribute values of the available training data. So, whenever it encounters a set of items (training set) it identifies the attribute that discriminates the various instances most clearly. This feature that is able to tell us most about the data instances so that we can classify them the best is said to have the highest information gain. Now, among the possible values of this feature, if there is any value for which there is no ambiguity, that is, for which the data instances falling within its category have the same value for the target variable, then we terminate that branch and assign it to the target value that we have obtained. For the other cases, we then look for another attribute that gives us the highest information gain. Hence we continue in this manner until we either get a clear decision about what combination of attributes gives us a particular target value, or we run out of attributes. In the event that we run out of attributes, or if we cannot get an unambiguous result from the available information, we assign this branch a target value that the majority of the items under this branch possess. Now that we have the decision tree, we follow the order of attribute selection as we have obtained from the tree. By checking all the respective attributes and their values with those seen in the decision tree model, we can assign or predict the target value of this new instance [21].

2) LAD Tree (Logical Analysis of Data): Logical Analysis of Data is the method for classification proposed in optimization literature. It builds a classifier for binary target variable based on learning a logical expression that

can distinguish between positive and negative samples in a data set. The basic assumption of LAD model is that a binary point covered by some positive patterns, but not covered by any negative pattern is positive, and similarly, a binary point covered by some negative patterns, but not covered by positive pattern is negative. The construction of Lad model for a given data set typically involves the generation of large set patterns and the selection of a subset of them that satisfies the above assumption such that each pattern in the model satisfies certain requirements in terms of prevalence and homogeneity [22].

3) Random Forest: Random Forests grow many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes. Each tree is grown as follows: If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree; if there are M input variables, a number m¡¡M is specified such that at each node, m variables are selected at random out of the M and the best split on this m is used to split the node. The value of m is held constant during the forest growing; each tree has grown to the largest extent possible. There is no pruning. It is unexcelled in accuracy among current algorithms. It can handle thousands of input variables without variable deletion. It generates an internal unbiased estimate of the generalization error as the forest building progresses. It computes proximities between pairs of cases that can be used in clustering, locating outliers or (by scaling) give interesting views of the data [23].

4) REP Tree: A decision tree is a tool for carrying out classification of data instances input to it. Decision trees have production rules of the type IF THEN. Rep Tree is a fast decision tree learner and builds a decision/regression tree using information gain/variance reduction and prunes. It uses reduced-error pruning with back fitting, only sorts values for numeric attributes once. Since this a fast algorithm so the pruned tree reduces the complexity in the classification process. Moreover pruning is used to find the best sub-tree of the initially grown tree with the minimum error for the test set [14].

5) Simple Cart (Simple classification and regression tree):It builds both classification and regression trees. The classification tree construction by CART is based on binary splitting of the attributes. It uses gini index splitting measure in selecting the splitting attribute. Pruning is done in CART by using a portion of the training data set. CART uses both numeric and categorical attribute for building the decision tree and has inbuilt features that deal with missing attributes [22].

III. RELATED WORK

Sabhnani et al. (2003) performed a simulation study to evaluate the performances of some machine learning algorithms on the KDD 1999 Cup intrusion detection dataset. They have evaluated Multilayer perceptron, Gaussian classifier (GAU), K-means clustering (K-M), nearest cluster algorithm (NEA), Incremental radial basis function, Leader algorithm (LEA), Hyper sphere algorithm (HYP), Fuzzy ARTMAP (Adaptive Resonance Theory mapping), and C4.5 decision tree. They demonstrated that for a given attack category certain classifier algorithms performed better. They evaluated that MLP, GAU, K-M, NEA, and RBF detected more than 85% of attack records for probing category and for attack records in DoS category, MLP, K-M, NEA, LEA, and HYP scored a 97% detection rate. GAU and K-M, the two most successful classifiers for U2R category, detected more than 22% of attack records. In case of R2L category, only GAU could detect around 10% of attack records. They have shown that MLP performs the best for probing, K-M for DoS as well as U2R, and GAU for R2L attack categories. As a result, they built a multi-classifier model using most promising classifiers for a given attack category that was evaluated for probing, denial-of-service, user-to-root, and remote-to-local attack categories. Their multi classifier system has shown precision of detection as follow: 0.887 for Probe instances,

0.973 for DoS instance, 0.298 for U2R instances and 0.096 for R2L instances. They reported that the machine learning algorithms working as classifiers for the KDD 1999 Cup data set cannot detect U2R and R2L attacks with sufficient accuracy within the misuse detection context [24].

Panda et al. (2008) evaluated three well known machine learning based classifiers namely ID3, J48 and Nave Bayes. They evaluated these classifiers based on the 10-fold cross validation test. They have also used the KDDCup99 IDS data set for all experiments. They evaluated the overall error rate of these classifiers and shown that Nave Bayes gave 3.56%, J48 gave 3.47% and ID3 gave 3.47% overall error rates. Their results illustrated that the Nave Bayes classifier is very interesting because of its simplicity, elegance, robustness and effectiveness. They also have shown that the decision trees have high efficiency in both generalization and detection of new attacks [25].

Tara pore et al. (2012) reviewed several AI based techniques used for the intrusion detection. They have evaluated Decision trees, back propagation neural networks, support vector ma-chine, hierarchical self-organizing maps (SOM). They have compared the results of different authors who applied these techniques by using different data sets and platforms. They showed the key concept of each methodology, advantages and disadvantages of each approach, and the dataset(s) used. They reported that the R2L and U2R attack classes have low detection rates since there is less number of training instances [5].

IV. TOOL AND DATASET

A. Tool: Weka (Waikato Environment for Knowledge Analysis)

Simulation work for proposed Network Intrusion Detection System is done using WEKA. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes [26]. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. WEKA consists of an Explorer, Experimenter, Knowledge flow, Simple Command Line Interface, Java interface.

The performance of the classifiers can be measured with the help of some performance metrics used in Weka. Consider the classifier classify the traffic into attack and normal (two classes only). Usually the performance of classifiers is measured by using confusion matrix and measures derived from the confusion matrix for n number of instances in the dataset.

TP: Number of True positives (Attack traffic classified as Attack);
FP: Number of False positives (Normal traffic classified as Attack);
TN: Number of True Negatives (Normal traffic classified as Normal);
FN: Number of False Negative (Attack traffic classified as Normal)

Table I: Confusion Matrix

	Predicted class	Predicted class
Actual Class	Normal	attack
Normal	TN	FP
Attack	FN	TP

Other measures derived using confusion matrix are: Ac-curacy is how accurately a classifier (Technique)

classifies instances; True Positive Rate (TPR) or Sensitivity or recall is the how well the classifier classifies the positive instances; False Positive Rate (FPR) rate of negative instances are inaccurately classified by the classifier; Precision is the probability of correctness of a positive prediction; F-measure c is the harmonic mean of precision and recall and can be used as a single measure of performance.; Receiver Operating Characteristics (ROC) graphs have long been used in signal detection theoryto depict the trade-off between hit rates and false alarm rates over noisy channel.

$$\text{Accuracy} = \frac{TP + TN}{n} \quad (1)$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{False Alarm} = \frac{FP}{TN + TP} \quad (3)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (4)$$

$$F \text{ measure} = \frac{2 \text{ precision recall}}{\text{precision} + \text{recall}} \quad (5)$$

B. Dataset: NSL-KDD

Knowledge Discovery and Data Mining Competition - KDD Cup 99 is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 the Fifth International Conference on Knowledge Discovery and Data Mining. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. A connection is a sequence of TCP packets starting and ending at some well-defined times, between which data flows to and from a source IP address to a target IP address under some well-defined protocol. Each connection is labelled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

In the KDD99 database, any network connection (or in-stance) is comprised of 41 attributes and each instance is labelled either as normal or as an attack-specified type. In KDD99 database, there are 4,898,430 labelled and 311,029 unlabeled connection records in the dataset. The labelled connection records consist of 22 different attack types categorized into 04 classes namely (DoS: denial of service, Probe, U2R: user to root, R2l: Remote to local)[?].Whereas unlabeled dataset consists of 20 known and 17 unknown attacks. The labelled dataset is used for training and unlabeled dataset is used for testing of the classifiers. The 17 unknown attacks used for testing helps in determining accuracy of the classifiers for unknown attacks. It is clear that total number of connection records to be used for training and testing of the classifiers is very large and also the number of connection records related to U2R and R2L is very less as compared to other attack classes. Moreover, KDD99 is built based on the data captured in DARPA98 which has been criticized by McHugh (2000), mainly because of the characteristics of the synthetic data. As a result, some of the existing problems in DARPA98 remain in

KDD99 [27].

Due to shortcomings and large number of records on which test was very difficult to perform Tavallace et al. (2009) included 125,973 and 22,544 records in a training dataset and test dataset respectively by randomly selecting connections from KDD99 dataset. They named this dataset NSL-KDD [28]. Further, in order to reduce non-uniformity in the dataset, we randomly selected maximum of 10,000 connection records of each attack type from labelled dataset for the purpose of training the classifiers in an unbiased manner. Total 66,961 (including normal) connection records are selected from entire labelled KDD dataset for training of classifiers. In order to test the classifiers, we randomly selected 5000 connection records of all attack types from unlabeled dataset. There are 40,603 connection records in the test dataset [29].

For using the dataset in training and testing purposes we must first pre-process the dataset. The pre-processing includes two phases: 1) Mapping of symbolic value features to numeric value; 2) Normalization of continuous features. We have done the pre-processing as suggested by Gulshan et al.(2010) as follow: 1) Symbolic features like protocol type, (3 different symbols), service (70 different symbols), and flag (11 different symbols) were mapped to integer values ranging from 0 to N-1 where N is the number of symbols; 2) The attack type feature is mapped to one of attack class namely Probe, DoS, U2R and R2L; 3) The normalization of continuous features is done as per equation shown below:

$$\text{Normalized Value}_i = \text{normalize}(\ln(\text{val}_i + 1)) \quad (6)$$

$$\text{Normalize}(X_i) = \frac{x_i \ln(\text{Min}_i + 1)}{\ln(\text{Max}_i + 1) \quad \ln(\text{Min}_i + 1)} \quad (7)$$

V. EXPERIMENT RESULTS AND ANALYSIS

In our experiments we have trained and tested seventeen AI classifiers. The training dataset consists of 37791 instances containing 12533 labelled Normal instances, 11656 labelled Probe instances, 12555 labelled DoS instances, 52 U2R in-stances and 995 R2L instances. Our test dataset consists of 6763 instances containing 1609 unlabeled Normal instances, 1607 unlabeled Probe instances, 1628 unlabeled DoS in-stances, 200 unlabeled U2R instances and 1719 unlabeled R2L instances. All experiments are performed using Weka 3.6.8 with 10-fold cross validation, 32 bit Windows XP platform, 2GB RAM, Intel core 2 Duo CPU, 2.00 GHz processor. The following figures show the results of experiments. We have calculated four performance metrics of various AI based classifiers for all class types.

The first metric is True Positive Rate (TPR) of seventeen AI based classifiers for different classes. Figure 4.1 depicts that RandomForest (0.97) have the highest TPR for Normal class followed by JRip, also JRip (0.863) have a highest TPR for DoS class. For the detection of Probe class, U2R class and R2L class Naive Bayes have highest TPRs which are 0.944, 0.275, and 0.501 respectively. Our second performance metric for comparison is False Positive Rate (FPR). Figure 4.2 illustrates the FPRs of tested classifiers for all attack classes. The classifier having lower FPR is better (low FPR=low false alarm).

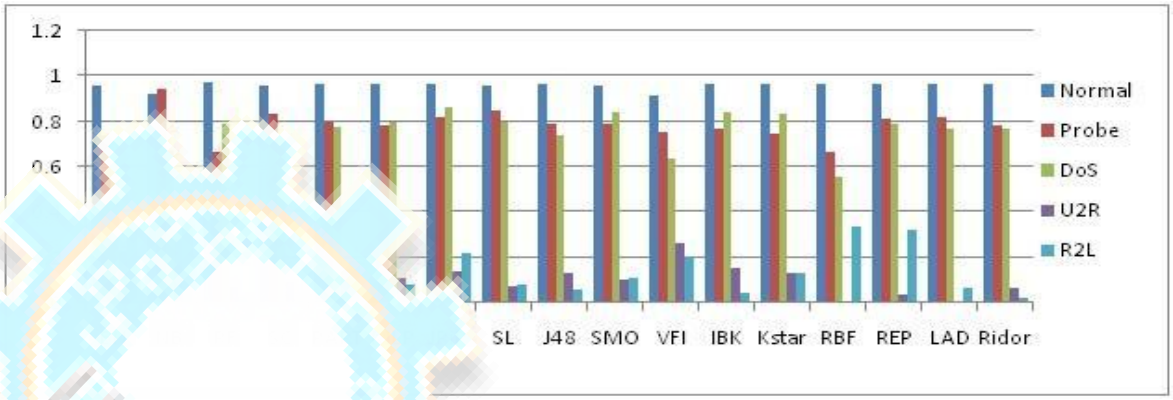

Fig. 1. TPR of classifiers for different attack classes

Naive Bayes (0.215) has lowest FPR for Normal type instances. JRip (0.013) and RBFnetwork (0.006) have lowest FPRs for Probe and DoS type instances respectively. In case of U2R instances every classifier shown low FPR but detection is also very low as seen in Figure 4.1. Jrip and REPtree have shown values of FPR near to zero for R2L class.

Fig. 2. FPR of classifiers for different attack classes

The third metric used for comparison is the F - measure which is very precise as it includes the precision and recall values in its calculation.

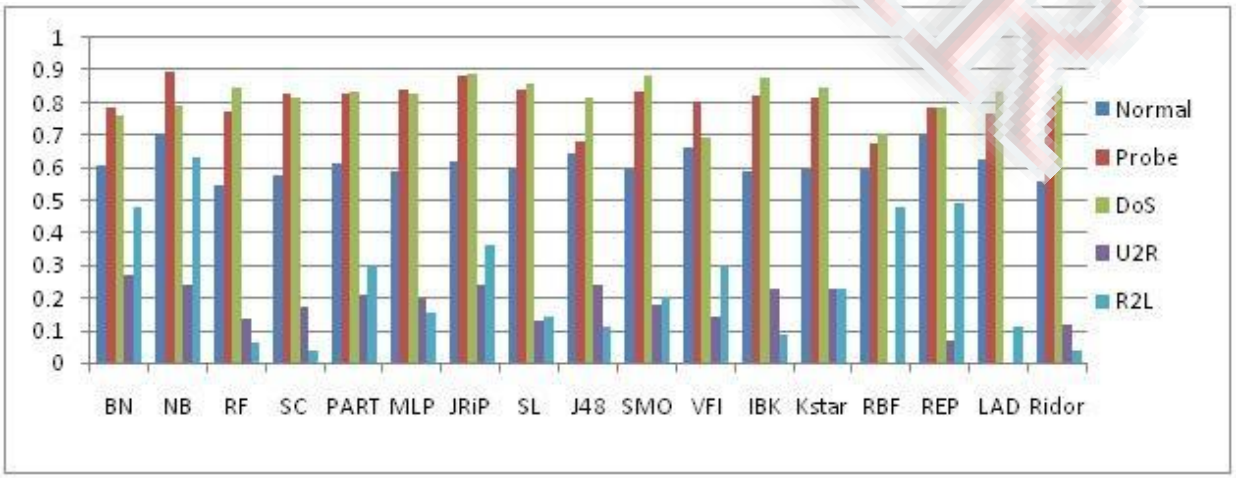

Fig. 3. F-Measure of classifiers for different attack classes

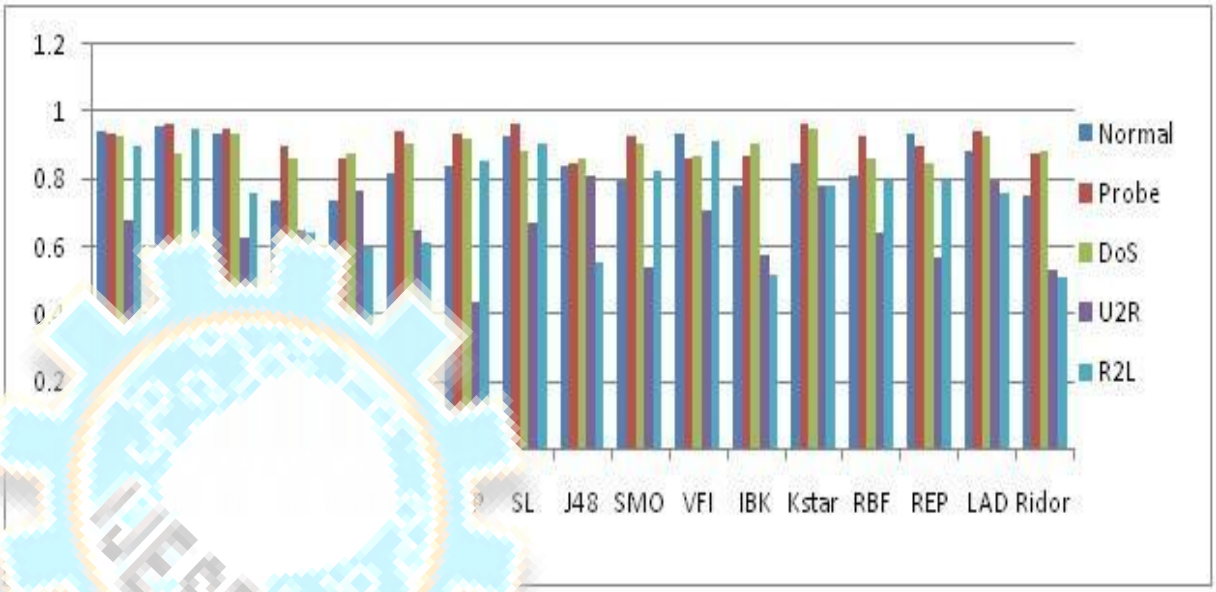

Fig. 4. ROC area of classifiers for different attack classes

On the basis of F-measure (as shown in Figure 4.3) the promising classifiers are: Naive Bayes for Normal (0.708), Probe (0.898) and R2L (0.634), JRip for DoS (0.887), BayesNet for U2R (0.271) class. Last metric we have used is ROC area which is also a common metric used for comparison of classifiers. The promising classifiers based upon the ROC area (Figure 4.4) are: Naive Bayes for Normal (0.955), Probe (0.967) and R2L (0.949); KStar for DoS (0.953) and J48 for U2R (0.813).

VI. CONCLUSION

AI based classifiers are highly suitable when used for Intrusion detection. They have learning capabilities which make them highly robust. Moreover these classifiers are very fast hence providing real time detection. We have analyzed performance of seventeen AI based classifiers for different class type instances. In our experiments we concluded that the Naive Bayes classifier is very suitable for intrusion detection, because of its high detection accuracy. For detection of Normal and Probe type class Naive Bayes performed best among the tested classifiers but it showed low performance for DoS type instances. JRip and SMO have shown good performances in detection of DoS class type instances. The training set contain very low number of U2R instances hence all classifiers have shown low detection rates for U2R type instances but Naive bayes and Bayes Net performed better than other classifiers for this class type. R2L instances were best classified using Naive Bayes and RE Ptree.

During our experiments we have seen that no single classifier is able to generate detection accuracy to an acceptance level. Different classifiers perform better for detection of a different class; hence more than one classifier should be combined to detect intrusions. Our future work will include combining these promising classifiers to build a multiple classifier system which uses strength of each classifier and tends to achieve detection accuracy to an acceptance level.

REFERENCES

1. G. Kumar, K. Kumar, and M. Sachdeva, "The use of artificial intelligence based techniques for intrusion detection: a review," Artificial Intelligence Review, vol. 34, no. 4, pp. 369–387, 2010.
2. V. Marinova-Boncheva, "A short survey of intrusion detection systems," Problems of Engineering Cybernetics and Robotics, vol. 58, pp. 23–30, 2007.
3. A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion detection: A survey," in Managing Cyber Threats. Springer, 2005, pp. 19–78.
4. M. C. Ponce, "Intrusion detection system with artificial intelligence," in FIST Conference-June, 2004.
5. K. D. Tarapore N.Z. and K. P.V., "Intrusion detection using an ensemble of intelligent paradigms," Journal of Artificial Intelligence, vol. 3, pp. 111–116, 2012.
6. J. Su and H. Zhang, "Full bayesian network classifiers," in Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 897–904.
7. G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.
8. J. Platt et al., "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.
9. M. Sumner, E. Frank, and M. Hall, "Speeding up logistic model tree induction," in Knowledge Discovery in Databases: PKDD 2005. Springer, 2005, pp. 675–683.
10. S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," Applied Soft Computing, vol. 10, no. 1, pp. 1–35, 2010.
11. C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," Expert Systems with Applications, vol. 36, no. 10, pp. 11 994–12 000, 2009
12. J. A. Bullinaria, "Introduction to neural networks," 2004, lecture 12.
13. D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," Machine learning, vol. 6, no. 1, pp. 37–66, 1991.
14. S. Ali, B. Pang, and K. Tackle, "Rule based base classifier selection for bagging algorithm." in DMIN, 2008, pp. 26–29
15. www.sourceforge.net.Class kstar. sourceforge. [Online]. Available: http://weka.sourceforge.net/doc.dev/weka/classifiers/lazy/KStar.html
16. G. Demiroz¨ and H. A. Guvenir,¨ "Classification by voting feature intervals," in Machine Learning: ECML-97. Springer, 1997, pp. 85–92.
17. W. W. Cohen, "Fast effective rule induction," in ICML, vol. 95, 1995, pp. 115–123.
18. www.sourceforge.net. Class jrip. sourceforge. [Online]. Available: http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/JRip.html
19. E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," 1998.
20. B. R. Gaines and P. Compton, "Induction of ripple-down rules applied to modeling large databases," Journal of Intelligent Information Systems, vol. 5, no. 3, pp. 211–228, 1995.
21. A. PADHYE. Chapter 5: Classification methods. Universityof Minnesota, Duluth. http://www.d.umn.edu/. [Online]. Available: http://www.d.umn.edu/ padhy005/Chapter5.html
22. Amudha.J, Soman.K.P, and Kiran.Y, "Article: Feature selection in top-down visual attention model using weka," International Journal of Computer Applications, vol. 24, no. 4, pp. 38–43, June 2011, published by Foundation of Computer Science.
23. L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001.
24. M. Sabhnani and G. Serpen, "Application of machine learning algorithms to kdd intrusion detection dataset within misuse detection context." in MLMTA, 2003, pp. 209–215.
25. M. Panda and M. R. Patra, "A comparative study of data mining algorithms for network intrusion detection," in Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on. IEEE, 2008, pp. 504–507.
26. Weka. Weka. The University of Waikato. [Online]. Available: http://www.cs.waikato.ac.nz/ ml/weka/
27. J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," ACM transactions on Information and system Security, vol. 3, no. 4, pp. 262–294, 2000.
28. M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009, 2009.
29. G. K. Ahuja, K. K. Saluja, and M. Sachdeva, "An empirical comparative analysis of feature reduction methods for intrusion detection," International Journal of Information and Telecommunication Technology (ISSN: 0976-5972), vol. 1, no. 1, 2010.