Int. J. Advance. Soft Comput. Appl., Vol. 2, No. 1, March 2010
ISSN 2074-8523; Copyright © ICSRS Publication, 2010
www.i-csrs.org

Intelligent Network Intrusion Detection Using DT and BN Classification Techniques

P. Srinivasulu, R. Satya Prasad and I. Ramesh Babu

Faculty of Computer Science and Engineering Department
V R Siddhartha Engineering College
Vijayawada, Andhra Pradesh, India
e-mail: srinivasulupamidi@yahoo.co.in

Faculty of Computer Science and Engineering Department
Dept of CSE, Acharya Nagarjuna University
Guntur, Andhra Pradesh, India
e-mail : profrsp@gamil.com, csnu@nagarjunauniversity.ac.in

Abstract

Security is becoming a critical part of organizational information systems. Network Intrusion Detection System (NIDS) is an important detection that is used as a countermeasure to preserve data integrity and system availability from attacks. The detection of attacks against computer networks is becoming a harder problem to solve in the field of Network security. Intrusion Detection is an essential mechanism to protect computer systems from many attacks. The success of an intrusion detection system depends on the selection of the appropriate features in detecting the intrusion activity. In NIDS electing unnecessary features may cause computational issues and decrease the accuracy of detection. This paper describes a technique of applying Genetic Algorithm (GA) to choose features (attributes) of KDDCUP99 Dataset. We have chosen the standard dataset KDDCUP from MIT, U.S.A, which is used for IDS research oriented projects. In this paper a brief overview of the Intrusion Detection System and genetic algorithm is presented. We used Bayes Network (BN), and Decision Tree (DT) Tree approaches for classifying the network attacks for chosen attribute dataset. These models gave better performance compared with the all features of dataset.

Keywords: *Genetic Algorithms, Classification methods, Confusion matrix, Fitness function, and Crossover.*

1 Introduction

Computer based Information Systems are becoming an integral part of our organizations. An Information System is a computerized system which contains organization information which serves the organization in its various activities and functions. Computer Security is the ability to protect a computer system and its resources with respect to confidentiality, integrity, and availability. Various protocols, firewalls are in existence to protect these systems from computer threats. Intrusion is a type of cyber attack that attempts to bypass the security mechanism of a computer system. Such an attacker can be an outsider who attempts to access the system, or an insider who attempts to gain and misuse non-authorized privileges.

Data Mining is assisting various applications for required data analysis. Recently, data mining is becoming an important component in intrusion detection system. Different data mining approaches like classification, clustering, association rule, and outlier detection are frequently used to analyze network data to gain intrusion related knowledge. In this paper we elaborate on neural network and CART data mining classification techniques and will describe how they are used in the classification of intrusion detection attacks.

Data Mining is an analytic process designed to explore data (usually large amounts of data - typically business or market related) in search of consistent patterns and/or systematic relationships between variables, and then to validate the findings by applying the detected patterns to new subsets of data. The ultimate goal of data mining is prediction. Predictive data mining is the most common type of data mining and one that has the most direct business applications. The process of data mining consists of three stages: (1) the initial exploration, (2) model building or pattern identification with validation/verification, and (3) deployment (i.e., the application of the model to new data in order to generate predictions). These steps are explained below

Stage 1: Exploration. This stage usually starts with data preparation which may involve cleaning data, data transformations, selecting subsets of records and - in case of data sets with large numbers of variables ("fields") - performing some preliminary feature selection operations to bring the number of variables to a manageable range (depending on the statistical methods which are being considered). Then, depending on the nature of the analytic problem, this first stage of the process of data mining may involve anywhere between a simple choice of straightforward predictors for a regression model, to elaborate exploratory analyses using a wide variety of graphical and statistical methods (see *Exploratory Data Analysis (EDA)*) in order to identify the most relevant variables

and determine the complexity and/or the general nature of models that can be taken into account in the next stage.

Stage 2: Model building and validation. This stage involves considering various models and choosing the best one based on their predictive performance (i.e., explaining the variability in question and producing stable results across samples). This may sound like a simple operation, but in fact, it sometimes involves a very elaborate process. There are a variety of techniques developed to achieve that goal - many of which are based on so-called "competitive evaluation of models," that is, applying different models to the same data set and then comparing their performance to choose the best. These techniques - which are often considered the core of predictive data mining - include: Bagging (Voting, Averaging), Boosting, Stackin (Stacked Generalizations), and Meta-Learning.

Stage 3: Deployment. That final stage involves using the model selected as best in the previous stage and applying it to new data in order to generate predictions or estimates of the expected outcome.

We used DARPA Intrusion Detection KDDCUP99 dataset. This was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. It includes a wide variety of intrusions simulated in a military network environment. The 1999 KDD intrusion detection dataset contains 41 features and one target class feature. The success of an intrusion detection system depends on the selection of the appropriate features in detecting the intrusion activity. Selecting unnecessary features may cause computational issues and decrease the accuracy of detection. We used Genetic Algorithm for reduction of attributes set as a data preprocessing step. The result indicates that the feature selected in the research has a good influence and may be useful in detecting the intrusion activity.

After preprocessing, we applied Regression Tree (CART) and Neural Network approaches for the classification purposed. The performances of the classification models were also shown.

2 KDDCUP99 Dataset Description

The KDDCUP99 data set is having 41 features and one target class feature (Table 1). The dataset contains six million records. 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. 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 labeled as either

normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes. Attacks fall into four main categories:

- **DOS**: denial-of-service, e.g. SYN flood;
- **R2L**: unauthorized access from a remote machine, e.g. guessing password;
- **U2R**: unauthorized access to local super user (root) privileges, e.g., various ``buffer overflow'' attacks;
- **Probing**: surveillance and other probing, e.g., port scanning.

Table 1: Features of KDDCUP99 Dataset

No	***feature name***	***Type***	***No***	***feature name***	***Type***
1	duration	continuous	21	is_hot_login	discrete
2	protocol_type	discrete	22	is_guest_login	discrete
3	service	discrete	23	count	continuous
4	src_bytes	continuous	24	serror_rate	continuous
5	dst_bytes	continuous	25	rerror_rate	continuous
6	flag	discrete	26	same_srv_rate	continuous
7	land	discrete	27	diff_srv_rate	continuous
8	wrong_fragment	continuous	28	srv_count	continuous
9	urgent	continuous	29	srv_serror_rate	continuous
10	hot	continuous	30	srv_rerror_rate	continuous
11	num_failed_login	continuous	31	srv_diff_host_rate	continuous
12	logged_in	discrete	32	dst_host_count	continuous
13	num_compromised	continuous	33	dst_host_srv_count	continuous
14	root_shell	discrete	34	dst_host_same_srv_ rate	continuous
15	su_attempted	discrete	35	dst_host_diff_srv_ rate	continuous
16	num_root	continuous	36	dst_host_same_src_ port_rate	continuous continuous
17	num_file_creations	continuous	37	dst_host_srv_diff_ host_rate	continuous
18	num_shells	continuous	38	dst_host_serror_ rate	continuous
19	num_access_files	continuous	39	dst_host_srv_ serror_rate	continuous
20	num_outbound_cm ds	continuous	40	dst_host_rerror_ rate	continuous
21	is_hot_login	discrete	41	dst_host_srv_rerror_ rate	continuous

3 Genetic Algorithm for selecting the main feature of KDDCUP99 Dataset

Genetic algorithm is a family of computational models based on principles of evolution and natural selection. These algorithms convert the problem in a specific domain into a model by using a chromosome-like data structure and evolve the chromosomes using selection, recombination, and mutation operators. The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representations of the problem to be solved. The process is shown in fig.1.

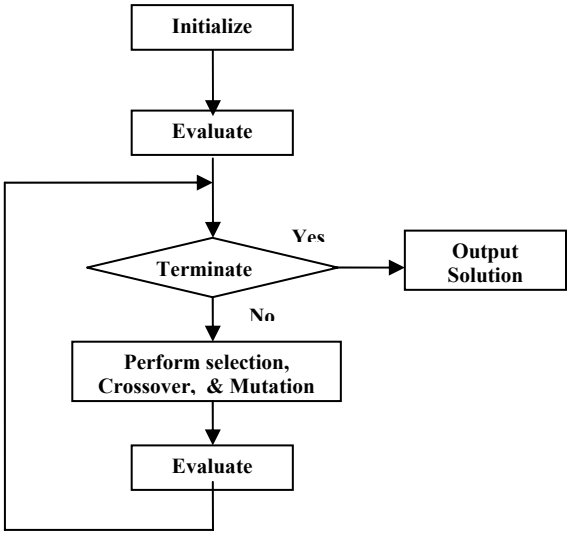

Fig.1: Genetic Algorithm

The procedure starts from an initial population of randomly generated individuals. Then the population is evolved for a number of generations while gradually improving the qualities of the individuals in the sense of increasing the fitness value as the measure of quality. During each generation, three basic genetic operators are sequentially applied to each individual that is Selection, Crossover and Mutation. Those chromosomes with a higher fitness value are more likely to reproduce offspring. If the new generation contains a solution that produces an output that is close enough or equal to the desired answer then the problem has been solved. If this is not the case, the new generation will go through the same process. This will continue until a solution is reached. The result of this algorithm on the KDDCUP99 dataset features were shown in Table 2.

Table 2: Initial Population of attributes-set

Merit	**Scaled**	**Attributes-Subset**
0.3289	0.26439	1 2 4 9 21 26 27 30 39 41
0.34959	0.30656	1 4 5 6 7 8 9 10 12 14 15 18 20 21 23 24 25 26 28 29 35 36 38 40 41
0.45835	0.5282	2 3 13 16 20 21 23 28 29 32 35 36 39 40
0.28381	0.17249	1 6 7 8 11 14 16 17 18 20 21 25 26 27 28 29 31 34 37 38 39 40 41
0.2375	0.07814	4 8 22 31 39
0.38589	0.38053	2 3 4 5 6 7 8 11 12 13 14 15 16 17 18 20 21 22 23 24 27 28 30 33 34 35
0.47399	0.56008	13 16 17 22 24 40 41
0.46616	0.54412	2 5 6 7 9 10 11 22 24 29 32 40 41
0.37359	0.35546	4 6 7 8 11 12 13 14 17 19 21 23 26 28 30 32 35 37 38 39
0.41159	0.4329	1 3 6 7 10 11 15 19 23 30 31 38 40
0.29318	0.19161	2 9 12 21 22 30 32 33
0.40489	0.41926	14 19 24 29 38
0.2396	0.0824	1 2 4 8 9 11 13 15 18 21 24 25 28 38
0.35648	0.32059	5 7 17 19 23 27 28 38
0.58294	0.7821	1 4 6 18 22 23 24 27 29 36
0.4998	0.61267	2
0.37717	0.36275	2 4 5 6 7 8 12 15 21 23 29 31 34 36
0.44225	0.49538	2 4 5 6 9 10 12 13 14 15 18 20 21 23 24 25 26 29 32 33 35 38 41

The parameters of Genetic search have been initialized and given below:

Population size *: 20*
Number of generations *: 20*
Probability of crossover *: 0.6*
Probability of mutation *: 0.033*
Report frequency *: 20*

In the genetic search the crossover and mutation the result is shown in Table 3.

Table 3: Generated Attributes set

Merit	**Scaled**	**Attributes-Subset**
0.82877	1.01422	2 3 5 6 10 12 13 22 23 24 29 30 31 37 41
0.82877	1.01422	2 3 5 6 10 12 13 22 23 24 29 30 31 37 41
0.81738	0.97681	2 3 5 6 10 12 13 22 23 24 25 29 30 31 33 35 37 41
0.78981	0.88623	2 3 5 6 10 13 17 18 24 27 31 32 33 36 37 39
0.63719	0.3849	2 3 5 6 12 13 18 20 22 23 24 29 30 31 37 41
0.81029	0.95351	2 3 5 6 10 12 13 22 23 24 28 29 31 34 37 41
0.6199	0.32811	3 5 6 8 10 13 19 22 23 24 29 30 31 37 41
0.56325	0.14203	3 5 6 9 10 13 16 17 18 22 24 27 31 33 36 37 39
0.80827	0.94689	3 5 6 10 13 22 23 24 29 30 31 37 41
0.82572	1.0042	2 3 4 5 6 10 12 13 22 23 24 29 30 31 37 41
0.81946	0.98364	2 3 5 6 12 13 18 22 23 24 29 31 36 37 38 40
0.82572	1.0042	2 3 4 5 6 10 12 13 22 23 24 29 30 31 37 41
0.78415	0.86765	2 3 5 10 13 17 18 22 24 25 32 36 37 41
0.76192	0.79463	3 5 6 11 12 13 17 18 24 27 31 33 36 37 39
0.52001	0	2 3 5 6 7 8 12 13 18 22 23 24 28 29 31 37 41
0.78474	0.8696	3 5 6 11 12 13 18 22 24 25 32 36 37 41
0.77406	0.8345	3 5 6 13 18 22 23 24 30 31 37
0.78054	0.85579	2 3 5 6 10 12 13 14 22 23 24 28 29 31 40 41
0.7955	0.90493	2 3 5 6 10 12 19 22 23 24 36 37 41
0.57665	0.18606	3 5 6 11 12 13 14 17 18 20 22 24 25 29 30 31 33 37 41

The selected attributes with fitness is given below:
On this set of attributes one more time genetic search is applied and results were shown in Table 4.

Table 4: Initial population of 20 records

Merit	Scaled	Attribute subset
0.72512	0.73436	4 5 10 11 16
0.66143	0.51309	2 6 10
0.80237	1.00274	1 4 8 9 11 12 13 14
0.7461	0.80724	2 6 8 12 14
0.60266	0.30891	4 11 12 16
0.81703	1.05368	2 3 4 6 8 9 11 13 14 16
0.75179	0.827	3 5 6 8 9 10 12 13 15 16
0.79277	0.96941	2 3 4 6 8 9 13 16
0.77901	0.92159	1 3 7 8 9 13 14 15 16
0.73537	0.76997	3
0.73482	0.76804	3 8 13 14
0.7571	0.84548	1 5 6 7 8 9 10 11 12 13 15 16
0.77986	0.92455	2 3 6 7 10 11 12 13 15
0.55274	0.13547	8 10 12
0.70012	0.6475	1 5 7 8 11 12 15
0.51375	0	4
0.65629	0.49523	1 6 7 8 11 12 16
0.76951	0.88858	2 4 5 7 8 9 10 11 14 16
0.79065	0.96204	2 4 6 10 13 14 15
0.75922	0.85282	1 2 3 6 7 12 16

Table 5: The generated attributes set with crossover and mutation operations

merit	scaled	Subset of attributes
0.87718	1.26957	1 2 3 4 5 12 13 14 15
0.87718	1.26957	1 2 3 4 5 12 13 14 15
0.87718	1.26957	1 2 3 4 5 12 13 14 15
0.86443	0.93517	1 3 4 5 13 14 15
0.86802	1.02943	1 2 3 4 12 13 14 15
0.86779	1.02327	1 2 3 4 6 12 13 14 15
0.85176	0.60297	1 3 5 13 14 15
0.85395	0.66022	1 3 4 12 13 14 15
0.85626	0.72103	1 3 4 6 12 13 14 15
0.8695	1.06818	1 3 4 5 12 13 14 15
0.86779	1.02327	1 2 3 4 6 12 13 14 15
0.85166	0.60024	1 2 3 4 12 13 15
0.82877	0	1 3 4 6 12 13 15
0.84742	0.48909	1 2 3 4 6 12 13 14 15 16
0.86802	1.02943	1 2 3 4 12 13 14 15
0.84714	0.48178	1 2 3 4 6 12 13 15
0.85822	0.77232	1 2 3 4 8 12 13 14 15
0.86779	1.02327	1 2 3 4 6 12 13 14 15
0.87718	1.26957	1 2 3 4 5 12 13 14 15
0.85526	0.69458	1 3 4 6 12 13 14 16

Selected attributes: 1,2,3,4,5,6,7,8,9,10 : 10

protocol_type
service
src_bytes
dst_bytes
count
dst_host_same_src_port_rate
dst_host_srv_diff_host_rate
dst_host_serror_rate
dst_host_srv_serror_rate
dst_host_rerror_rate

This is the final set of attributes selected for classification. In the next section we describe how to apply the CART and Neural Network techniques for the reduced dataset.

4 Induction Decision (ID) Tree Technique

Decision Trees

Decision trees are a popular structure for supervised learning. A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or *decision*. We wish to be able to induce a decision tree from a set of data about instances together with the decisions or classifications for those instances.

Choosing Attributes and ID

- The order in which attributes are chosen determines how complicated the tree is.
- ID uses information theory to determine the most informative attribute.

Tree Induction Algorithm

- The algorithm operates over a set of training instances, T.
- If all instances in T are in class *P*, create a node *P* and stop, otherwise select a *feature* or *attribute* **F** and create a decision node.
- Partition the training instances in T into subsets according to the values of *V*.
- Apply the algorithm recursively to each of the subsets T.

Entropy

- Different messages have different probabilities of arrival.
- Overall level of uncertainty (termed entropy) is:
- $-\Sigma_i P_i \log_2 P_i$
- Frequency can be used as a probability estimate.

Information and Learning

- We can think of learning as building many-to-one mappings between input and output.

- Learning tries to reduce the information content of the inputs by mapping them to fewer outputs.
- Hence we try to minimize entropy.
- The simplest mapping is to map everything to one output.

The Model using ID Tree Show in Fig. 2.

```
src_bytes <= 35195
|  dst_host_srv_diff_host_rate <= 0.48
|  |  dst_host_srv_serror_rate <= 0.04
|  |  |  protocol_type = tcp
|  |  |  |  dst_host_rerror_rate <= 0.02: normal.
|  |  |  |  dst_host_rerror_rate > 0.02
|  |  |  |  |  dst_host_srv_diff_host_rate <= 0
|  |  |  |  |  |  dst_host_serror_rate <= 0
|  |  |  |  |  |  |  service = http
|  |  |  |  |  |  |  |  src_bytes <= 7012: normal.
|  |  |  |  |  |  |  |  src_bytes > 7012: back.
|  |  |  |  |  |  |  service = smtp: normal. (11.0)
|  |  |  |  |  |  |  service = private: ipsweep.
|  |  |  |  |  |  dst_host_serror_rate > 0
|  |  |  |  |  |  |  service = telnet: guess_passwd.
|  |  |  |  |  |  |  service = private: portsweep.
|  |  |  |  |  dst_host_srv_diff_host_rate > 0: normal. (1276.0/4.0)
|  |  |  protocol_type = udp
|  |  |  |  src_bytes <= 28
|  |  |  |  |  service = domain_u: normal. (28.0)
|  |  |  |  |  service = private: teardrop. (99.0)
|  |  |  |  src_bytes > 28
|  |  |  |  |  service = ntp_u: normal. (146.0)
|  |  |  |  |  service = domain_u: normal.
|  |  |  |  |  service = private: nmap. (26.0/1.0)
|  |  |  protocol_type = icmp
|  |  |  |  src_bytes <= 548
|  |  |  |  |  src_bytes <= 14: nmap. (3.0/1.0)
|  |  |  |  |  src_bytes > 14: normal. (195.0)
|  |  |  |  src_bytes > 548
|  |  |  |  |  src_bytes <= 1255: smurf. (9142.0)
|  |  |  |  |  src_bytes > 1255: pod. (20.0)
|  |  dst_host_srv_serror_rate > 0.04
|  |  |  count <= 5
|  |  |  |  dst_host_srv_serror_rate <= 0.84
|  |  |  |  |  dst_host_srv_diff_host_rate <= 0.02
|  |  |  |  |  |  dst_bytes <= 251: guess_passwd.
|  |  |  |  |  |  dst_bytes > 251: normal. (6.0)
|  |  |  |  |  dst_host_srv_diff_host_rate > 0.02: normal. (16.0)
```

```
|  |  |  |  dst_host_srv_serror_rate > 0.84: nmap.
|  |  |  count > 5
|  |  |  |  dst_host_rerror_rate <= 0: neptune.
|  |  |  |  dst_host_rerror_rate > 0: portsweep.
|  dst_host_srv_diff_host_rate > 0.48
|  |  src_bytes <= 8
|  |  |  service = http
|  |  |  |  dst_host_rerror_rate <= 0.97: ipsweep.
|  |  |  |  dst_host_rerror_rate > 0.97: normal.
|  |  |  service = finger: normal. (7.0/1.0)
|  |  |  service = eco_i: ipsweep. (563.0)
|  |  |  service = private: ipsweep. (54.0)
|  |  src_bytes > 8: normal. (20.0/1.0)
src_bytes > 35195
|  dst_bytes <= 913: normal. (51.0/2.0)
|  dst_bytes > 913: back. (1992.0)
```

Fig. 2: Induction Decision Tree for IDS.

5 Bayesian Network Technique for IDS

Bayesian networks (BNs), also known as *belief networks* (or Bayes nets for short), belong to the family of probabilistic *graphical models* (GMs). These graphical structures are used to represent knowledge about an uncertain domain. In particular, each node in the graph represents a random variable, while the edges between the nodes represent probabilistic dependencies among the corresponding random variables. These conditional dependencies in the graph are often estimated by using known statistical and computational methods.

A Bayesian network *B* is an annotated acyclic graph that represents a Joint Probability Distribution (JPD) over a set of random variables **V**. The network is defined by a pair

B = (*G,* Θ *)*, where *G* is the Directed Acyclic Graph (DAG) whose nodes *X*1*, X*2*, . . ., Xn* represents random variables, and whose edges represent the direct dependencies between these variables. The graph *G* encodes independence assumptions, by which each variable *Xi* is independent of its no`ndescendents given its parents in *G*. The second component Θ denotes the set of parameters of the network. This set contains the parameter $\theta_{xi|\pi i} = P_B(xi|\pi i)$ for each realization *xi* of *Xi* conditioned on *πi*, the set of parents of *Xi* in *G*. Accordingly, *B* defines a unique JPD over **V**, namely:

$$P_B(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P_B(X_i \mid \Pi_i) \quad (1)$$

If X_i has no parents, its local probability distribution is said to be *unconditional*, otherwise it is *conditional*. If the variable represented by a node is *observed*, then

the node is said to be an evidence node, otherwise the node is said to be hidden or latent. The probability distribution for protocol_type attribute is shown in table6. Similarly we can compute for remaining attributes.

5.1 BNs Learning

In many practical settings the BNs is unknown and one needs to learn it from the data. This problem is known as the *BNs learning problem*, which can be stated informally as follows: Given training data and prior information (e.g., *expert knowledge, casual relationships*), estimate the graph topology (network structure) and the parameters of the JPD in the BNs.

One of the BNs learning methods is the *Equivalent Sample Size (ESS)*. In this method assign a prior *probability density function* to each parameter vector and use the training data to compute the posterior parameter distribution and the Bayes estimates. To compensate for zero occurrences of some sequences in the training dataset, one can use appropriate (mixtures of) conjugate prior distributions.

In the first and simplest case the goal of learning is to find the values of the BNs parameters in each Conditional Probability Distribution (CPD) that maximize the (log) likelihood of the training dataset. This dataset contains *m* cases that are often assumed to be independent. Given the training dataset of

$$\sum = \{x_1, x_2, ...x_n\} \text{ and } x_l = (x_{l1,} x_{l2,} \ldots x_{ln})^T, \text{ and}$$

the parameter set $\Theta = (\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_n})$, where $\boldsymbol{\theta}_i$ is the vector of parameters for the conditional distribution of variable X_i (represented by one node in the graph), the log-likelihood of the training dataset is a sum of terms, one for each node. The probability table for the attribute protocol_type is shown in the table 6. Similar tables will be computed for the remaining attributes using expression 2.

$$\log L(\Theta|\Sigma) == \sum_m \sum_n \log P(x_{li} \mid \Pi_i, \theta_i) \qquad (2)$$

Table 6: Probability distribution for protocol_type attribute

target_class	***tcp***	***udp***	***icmp***
normal	0.957504	0.0365	0.005916
buffer_overflow	0.967213	0.016393	0.016393
perl	0.6	0.2	0.2
guess_passwd	0.981651	0.009174	0.009174
pod	0.069767	0.023255	0.906976
teardrop	0.014925	0.98009	0.004975
ipsweep	0.149355	0.002274	0.848369
ftp_write	0.894736	0.052631	0.052631
back	0.999500	2.50E-04	2.50E-04
imap	0.6	0.2	0.2
phf	0.6	0.2	0.2
nmap	0.816091	0.172413	0.01149
multihop	0.846153	0.076923	0.076928
satan	0.953023	0.045477	0.001499
neptune	0.99968	1.60E-04	1.60E-04
portsweep	0.975903	0.01204819	0.012048
smurf	9.89E-04	1.10E-04	0.9989

6 Performance Evaluation.

The confusion matrix is more commonly named *contingency table*. In our case we have 15 classes, and therefore a 15x15 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.

The *True Positive (TP) rate (TPR)* 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*** or ***Sensitivity.***

The *False Positive (FP)* rate (FPR) 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.

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.

$$\Pr = \frac{TP}{TP + FP} \tag{3}$$

F-Measure (F_v) combines the true positive rate (recall) and precision Pr into a single utility function which is defined as ν-weighted harmonic mean.

$$F_y = \frac{1}{\sqrt{(1/TPR) + (1-y)(1/\Pr)^3}}\ , \tag{4}$$

6.1 Receiver Operating Characteristic (ROC)

ROC is used to evaluate the predictive ability of classifiers and provide an intuitive and convenient way of dealing with asymmetric costs of the two types of errors. ROCs are plotted in coordinates spanned by the rates of false positive and true positive classifications. These rates, denoted by TPR and FPR are defined as follows

$$TPR = \frac{TP}{TP + FN}, \text{ and } \quad FPR = \frac{FP}{FP + TN}\ , \tag{5}$$

where,

FN is False Negative,
TN is True Negative,
TP is True Positive, and
FP is False Positive

These measures are useful for comparing classifiers. The performance for the classification models ID and Bayes Networks is shown in Table 9 and Table 10.

7 Results

The results for both classification models Induction Decision (ID) and Bayes Network models are shown in the tables 7-10. These results are showing that ROC area for both models is almost one for all class types. The accuracy is 99.9% for ID and 99.6% for Bayes network. This is because we chosen only ten attributes out of 41 features using genetic search algorithm.

Table 7: Confusion Matrix for ID Tree Technique

	Normal	**Neptune**	**Smurf**	**Guess_Passwd**	**Pod**	**Teardrop**	**Portsweep**	**Ipsweep**	**Back**	**nmap**
normal	32290	0	0	5	0	0	2	1	1	2
neptune	0	3539	0	0	0	0	0	0	0	1
smurf	0	0	9142	0	0	0	0	0	0	0
guess_passwd	1	0	0	51	0	0	0	0	1	1
pod	0	0	0	0	20	0	0	0	0	0
teardrop	0	0	0	0	0	99	0	0	0	0
portsweep	4	0	0	0	0	0	33	0	0	0
ipsweep	15	0	0	0	0	0	0	642	1	1
back	4	0	0	0	0	0	0	0	1998	0
nmap	1	0	0	0	0	0	1	1	0	126

Table 8: Confusion Matrix Bayes Net

	Normal	**Neptune**	**Smurf**	**Guess_Passwd**	**Pod**	**Teardrop**	**Portsweep**	**Ipsweep**	**Back**	**nmap**
normal	32079	0	9	5	81	2	0	110	0	6
neptune	0	3357	0	0	0	0	0	0	1	2
smurf	0	0	9142	0	0	0	0	0	0	0
guess_passwd	1	0	0	51	0	0	0	0	0	1
pod	0	0	0	0	20	0	0	0	0	0
teardrop	0	0	0	0	0	99	0	0	0	0
portsweep	0	0	0	0	0	0	40	0	0	0
ipsweep	6	0	0	0	0	0	0	652	0	0
back	0	0	0	0	0	0	0	0	2002	0
nmap	0	0	0	0	0	4	0	2	0	124

Table 9: Detailed Accuracy by class using using ID Tree Technique

TP-Rate	FP-Rate	Precision	Recall	F-Measure	ROC-Area	Class
1	0.002	0.999	1	0.999	1	normal
1	0	1	1	1	1	neptune
1	0	1	1	1	1	smurf
0.943	0	0.909	0.943	0.926	0.972	guess_passwd
1	0	1	1	1	1	pod
1	0	1	.1	1	1	teardrop
0.875	0	0.897	0.875	0.886	0.975	portsweep
0.974	0	0.988	0.974	0.981	0.998	ipsweep
0.998	0	1	0.998	0.999	0.999	back
0.969	0	0.962	0.969	0.966	0.996	nmap

Table 10: Detailed Accuracy by class Bayes Net

TP-Rate	FP-Rate	Precision	Recall	F-Measure	ROC-Area	Class
0.993	0	1	0.993	0.996	1	normal
0.999	0	1	0.999	1	1	neptune
1	0	0.999	1	1	1	smurf.
0.962	0	0.911	962	0.936	1	guess_passwd
1	0.002	0.198	1	0.331	1	pod
1	0	0.943	1	0.971	1	teardrop
1	0	0.976	1	0.988	1	portsweep
0.991	0.002	0.853	0.991	0.917	1	ipsweep
1	0	1	1	1	1	back
0.954	0	0.867	0.954	0.908	1	nmap

References

[1] P. Clark and R. Bosewell, "Rule Induction with CN2: Some Recent Improvements in Machine Learning": *Proc. Of the 5th European Conf. (EWSL- 91)*, pages 151-163, 1991.

[2] W.W. Cohen, "Fast Effective Rule Induction". *In Proc. Of the 12th Intl. Conf. on Machine Learning*, pages 115-123, Tahoe City, July 1995.

[3] P. Cheeseman and J. Stutz, "Bayesian classification: Theory and results". In U.M. Fayyad, G. Piatetsky, P. Smith, and R. Uthuruswamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153-180, MIT Press, 1996.

[4] M. Dash and H. Liu, Feature selection methods for classification, *Intelligent Data Analysis*, 1(3):131-156, 1997.

[5] M. Dash and H. Liu, and J.Yao, "Dimensionality Reduction of unsupervised data." *In Proc. 1997 IEEE Int. Conf. Tools with AI (ICTAI)*, pages 532-539, IEEE Computer Society, 1997.

[6] P. Srinivasulu, D. Nagaraju, P. Ramesh Kumar, K. Nageswara Rao, "Classifying the Network Intrusion Attacks using Data Mining Classification Methods and their Performance Comparison", *International Journal of Computer Science and Network Security*, Vol. 9 No. 6 pp. 11-18, 2009.

[7] P Srinivasulu, R Ranga Rao, and I Ramesh Babu, "Intrusion Detection System using FP Tree", *International Journal of Advanced Networking Applications*, Vol. 1 No. 1 pp. 30-39, 2009.

[8] http://kdd.ics.uci.edu/databases/kddcup99

[9] Stich, T. (2004). Bayesian networks and structure learning, Diploma Thesis, Computer Science and Engineering, University of Mannheim.

[10] Jiawei Han and Micheline Kamber, *Data Mining Concepts and Techniques*, Elsevier publication, second edition.

[11] Richard J. Roiger, and Michael W. Geatz, *Data Mining A Tutorial-based Primer*, Pearson Education publications.

[12] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to Data Mining*, Pearson publications

[13] S. Selvakani, and R.S.Rajesh, "Genetic Algorithm for framing rules for Intrusion Detection", *International Journal of Computer Science and Network Security*, Vol.7 No.11, November 2007.

[14] S.SELVAKANI, and R.S.RAJESH, "Escalate Intrusion Detection using GA – NN", *Int. J. Open Problems Compt. Math.*, Vol. 2, No. 2, June 2009.

[15] Thaer Alibaisi, ABD El Latif ABU Dalhoum, Mohammed Al Rawi, Manuel Alfonseca, Alfonso Ortega, Network Intrusion Detection Using Genetic Algorithm to find Best DNA Signature, *WSEAS Transactions on Systems*.

Author's Biography

P. Srinivasulu received his B. Tech from Acharya Nagarjuna University, Guntur, Andhra Pradesh in 1994 and completed post graduation from Jawaharlal Nehru Technological University, Hyderabad in 1998. He is currently pursuing Ph. D from Acharya Nagarjuna University, Guntur, and working as Assistant Professor in V R Siddhartha Engineering College, in the Department of Computer Science and Engineering, Vijayawada,

Andhra Pradesh. His research interests include Data Mining and Data Warehousing, Computer Networks, Network security and Parallel Computing. He has more than thirteen years of experience in teaching in many subjects, industry and in research. He is the member of Indian Society of Technical Education (ISTE) and also member of Computer Society of India (CSI). He has many publications in National and International conferences. He was selected for the Journal.

R Satya Prasad received Ph. D from Acharya Nagarjuna University in Computer Science and Engineering. Presently working as Asst. Professor in the department of Computer Science and Engineering. He completed M. Phil and M.Sc. He is member of Post Graduate Board of Studies in the CSE Dept. of ANU. He is also member of Computer Society of India (CSI). He has four international journals paper publications and many internationational conference papers. He is getting PhD thesis for evaluation.

Dr. I Ramesh Babu received Ph. D in commuter science and engineering from Achary Nagarjuna University, Gunture. M.E in Computer Engineering from Andhra University, B.E in Electronics and Communication Engineering from University of Mysore. He is currently working as Professor in the department of Computer Science Nagarjuna University.

Also he is Senate member of the same university from 2006. He held many positions in Acharya Nagarjuna University as Head, Director-Computer Center, chairman board of studies. He was a special officer, convener of ICET, Andhra Pradesh. He is also member of board of studies for the other universities. His areas of interest include image processing, computer graphics, cryptography, artificial intelligence, and network security. He is a member of IEEE, CSI, ISTE, IETE, IGISS, and Amateur Ham Radio (VU2 IJZ). He is currently supervising ten PhD students who are working in different areas of image processing and artificial intelligence. He has published 35 papers in international journals and conferences.