computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

SELECTING KDD FEATURES AND USING RANDOM CLASSIFICATION TREE FOR PREDICTING ATTACKS

Rachid Beghdad

Faculty of sciences, 12 boulevard Bouaouina, Béjaïa 06000, Algeria.
email: rbeghdad@yahoo.fr

Abstract: *The purpose of this study is to identify some higher-level KDD features, and to train the resulting set with an appropriate machine learning technique, in order to classify and predict attacks. To achieve that, a two-steps approach is proposed. Firstly, the Fisher's ANOVA technique was used to deduce the important features. Secondly, 4 types of classification trees: ID3, C4.5, classification and regression tree (CART), and random tree (RnDT), were tested to classify and detect attacks. According to our tests, the RndT leads to the better results. That is why we will present here the classification and prediction results of this technique in details. Some of the remaining results will be used later to make comparisons. We used the KDD'99 data sets to evaluate the considered algorithms. For these evaluations, only the four attack categories' case was considered. Our simulations show the efficiency of our approach, and show also that it is very competitive with some similar previous works.*

Keywords: *Intrusion detection systems, Data mining, Misuse intrusion detection, Fisher's ANOVA ranking, Knowledge Discovery and Data mining (KDD) dataset, Classification Trees.*

1. INTRODUCTION

Detection of attempts to compromise the integrity, confidentiality, or availability of computing and communication networks is an extremely challenging problem. Most current approaches to the design of intrusion detection systems (IDS) are based on the premise that the actions used in an attempted intrusion can be differentiated from the actions executed by users or processes during the normal operation of the computing and communication networks. An effective IDS logs actions executed by users or processes for investigation, alerts the system administrator when the monitored activities are indicative of attempted intrusion, and, if appropriate, takes corrective measures e.g., expelling the intruder [1]. Since the amount of audit data that an IDS needs to examine is very large even for a small network, analysis is difficult even with computer assistance because extraneous features can make it harder to detect suspicious behaviour patterns [2]. That is why supervised learning techniques such as classification trees are so successful in detecting network intrusions; they are also capable of identifying new attacks.

There are two basic types of intrusion detection: host-based and network-based. Each has a distinct approach to monitoring and securing data, and each has distinct advantages and disadvantages. In short, host-based IDSs examine data held on individual computers that serve as hosts, while network-based IDSs examine data exchanged between computers.

In addition to that, intrusion detection techniques can be mapped into four classes: anomaly detection, misuse detection, specification-based detection, and model-based detection. Anomaly detection consists of establishing normal behavior profile for user and system activity and observing significant deviations of actual user activity with respect to the established habitual pattern. Misuse detection, refers to intrusions that follow well defined attack patterns that exploit weaknesses in system and application software. In specification-based detection, the correct behaviours of critical objects are manually abstracted and crafted as security specifications, which are compared with the actual behaviour of the objects. Intrusions, which usually cause object to behaviour in an incorrect manner, can be detected without exact knowledge about them. Model-based intrusion detection compares a process's execution against a program model to detect intrusion attempts.

In this paper, a two-steps approach is presented to efficiently classify and predict attacks. In the first step, an appropriate method will be used to reduce the amount of data managed by an IDS. In the second step, the RndT classification tree will be applied on the resulting subset to classify and predict attacks.

The rest of this paper is organized as follows. Section 2 presents a brief survey of the data mining methods used in intrusion detection. All the studied classification trees are presented in section 3. The statistical method used for reducing data is presented in section 4. Section 5 describes the evaluation dataset. Our experiments and their results are detailed in section 6. Section 7 concludes the paper.

2. RELATED RESEARCH

In this section, some intrusion detection models are presented.

In [3] the authors present decision tree techniques that are used to automatically learn intrusion signatures and classify activities in computer network systems as normal or intrusive for intrusion detection. They show the design of decision tree classifiers for intrusion detection, using different features of raw activity data in computer network systems and different sizes of observation windows. The performance of decision tree classifiers is discussed. They also present the impact of noises in data on the detection performance of the decision tree classifiers. Computer audit data from the Basic Security Module of the Solaris operating system are used to train and test the decision tree classifiers.

In [4] an intrusion detection algorithm based on GP ensembles is proposed. The algorithm runs on a distributed hybrid multi-island model-based environment to monitor security-related activity within a network. Each island contains a cellular genetic program whose aim is to generate a decision-tree predictor, trained on the local data stored in the node. Every genetic program operates cooperatively, yet independently by the others, by taking advantage of the cellular model to exchange the outmost individuals of the population. After the classifiers are computed, they are collected to form the GP ensemble. Experiments on the KDD Cup 1999 Data show the validity of the approach.

[5] addresses the issue of identifying important input features in building an intrusion detection system (IDS). Since elimination of the insignificant and/or useless inputs leads to a simplification of the problem, faster and more accurate detection may result. Feature ranking and selection, therefore, is an important issue in intrusion detection. The authors apply the technique of deleting one feature at time to perform experiments on SVMs and neural networks to rank the importance of input features for the DARPA collected intrusion data. Important features for each of the 5 classes of intrusion patterns in the DARPA data are identified. It is shown that SVM-based and neural network based IDSs using a reduced number of features can deliver enhanced or comparable performance. An IDS for class-specific detection based on five SVMs is proposed.

In [6] the authors present a study to identify important input features in building an IDS that is computationally efficient and effective. They investigated the performance of two feature selection algorithms involving Bayesian networks (BN) and Classification and Regression Trees (CART) and an ensemble of BN and CART. They used Markov blanket (MB) technique to reduce the features to respectively 17 and 12 features (variables). The training and test sets comprised only 5092 and 6890 records, respectively. Empirical results indicated that significant input feature selection is important to design an IDS that is lightweight, efficient and effective for real world detection systems. They also proposed an hybrid architecture for combining different feature selection algorithms for real world intrusion detection.

In [7] the authors demonstrate that poor performance can be improved using a combination of discriminative training and generic keywords. Generic keywords are selected to detect attack preparations, the actual break-in, and actions after the break-in. Discriminative training weights keyword counts to discriminate between the few attack sessions where keywords are known to occur and the many normal sessions where keywords may occur in other contexts. This approach was used to improve the baseline keyword intrusion detection system used to detect user-to-root attacks in the 1998 DARPA Intrusion Detection Evaluation. It reduced the false alarm rate by two orders of magnitude (to roughly 1 false alarm per day) and increased the detection rate to roughly 80%. The improved keyword system detects new as well as old attacks in this data base and has roughly the same computation requirements as the original baseline system. Both generic keywords and discriminant training were required to obtain this large performance improvement.

The authors of [8] investigate the use of a hybrid genetic algorithm/k-nearest neighbour approach to features selection and apply this approach to an intrusion detection data set. They have found that this feature selection process is able to identify features that are important for identifying different types of attacks present in the data set leading to improved classification accuracy.

In [9], two machine-learning paradigms, artificial neural networks and fuzzy inference system, are used to design an intrusion detection system. SNORT is used to perform real time traffic analysis and packet logging on IP network during the training phase of the system. Then a signature pattern database is constructed using protocol analysis and neuro-fuzzy learning method. Using 1998 DARPA Intrusion Detection Evaluation Data and TCP dump raw data, the experiments are deployed and discussed.

2.1. CRITICS

- Even if the results of Chebrolu and al.[6] using 17 and 12 features are interesting, we will try to propose a better approach for the features deduction, and the attacks prediction.
- Even if some supervised learning techniques lead to some results that are competitive with the "winning strategy" ones, the classification rates (CRs) of "User To Root" and "Remote To Local" attack categories remain poor. In addition to that the CR of the "Probing" category is still less than 90%.
- Even if the four attacks categories case was treated, some works did not detail and compare the performances of the classification trees used.

That is why, we focused in this study on the improvement of the Chebrolu and al.[6] approach. To achieve that, instead of using Markov blanket model, we will use another technique, for the deduction of important features. After that, instead of using Bayesian nets or CART, we will use another classification tree for the classification and the prediction of attacks.

3. CLASSIFICATION TREES

Decision or classification tree is a predictive model; that is, a mapping of observations about an item to conclusions about the item's target value. Each interior node corresponds to a variable; an arc to a child represents a possible value of that variable. A leaf represents the predicted value of target variable given the values of the variables represented by the path from the root.

Classification tree analysis is a term used when the predicted outcome is the class to which the data belongs. Regression tree analysis is a term used when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital). C-RT analysis is a term used to refer to both of the above procedures. The name C-RT or CART is an acronym from the words Classification And Regression Trees, and was first introduced by Breiman et al. [10]. In decision trees, two major phases should be ensured:

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 labelling each leaf.

2. Classification. In order to classify a new instance, we start by the root of the decision tree, then we test the attribute specified by this node. The result of this test allows moving down the tree branch relative to the attribute value of the given instance. This process will be repeated until a leaf is encountered. The instance is then being classified in the same class as the one characterizing the reached leaf.

Several algorithms have been developed in order to ensure the construction of decision trees and its use for the classification task. The ID3 and C4.5 algorithms developed by Quinlan [11] are probably the most popular ones. We can also mention the CART algorithm of Breiman and al. [10]. The majority of these algorithms use a descendent strategy, i.e. from the root to the leaves. To ensure this procedure, the following generic parameters are required:

– The attribute selection measure taking into account the discriminative power of each attribute over classes in order to choose the 'best' one as the root of the (sub) decision tree. In other words, this measure should consider the ability of each attribute A_k to determine training objects' classes. In the literature many attribute selection measures are proposed. We mention the gain ratio, used within the C4.5 algorithm [11] and based on the Shannon entropy, where for an attribute A_k and a set of objects T, it is defined as follows:

$$Gain(T, A_k) = Info(T) - Info_{A_k}(T) \qquad (1)$$

Where:

$$\mathrm{Info(T)} = -\sum_{i=1}^{n} \frac{\mathrm{freq(c_i, T)}}{|\mathrm{T}|} \log_2 \frac{\mathrm{freq(c_i, T)}}{|\mathrm{T}|} \qquad (2)$$

$$Info_{A_k}(T) = \sum_{a_k \in D(A_k)} \frac{|T_{a_k}^{A_k}|}{|T|} Info(T_{a_k}^{A_k}) \qquad (3)$$

and *freq(c* —

and $freq(c_i, T)$ denotes the number of objects in the set T belonging to the class c_i and $T_{a_k}^{A_k}$ is the subset of 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 defined as the information content of the attribute A_k itself:

$$Split\ Info(T,A_k) = -\sum_{a_k \in D(A_k)} \frac{|T_{a_k}^{A_k}|}{|T|} \log 2 \frac{|T_{a_k}^{A_k}|}{|T|} \qquad (4)$$

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

$$\textit{Gain ratio}(T,A_k) = \frac{Gain(T,A_k)}{SplitInfo(A_k)} \quad (5)$$

– 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.

3.1. RANDOM CLASSIFICATION TREE (RNDT)

In machine learning, a random forest is a classifier that consists of many decision trees and outputs the class that is the mode of the classes output by individual trees. The algorithm for inducing a random forest was developed by Leo Breiman and Adele Cutler [12], and "Random Forests" is their trademark. The term came from random decision forests that was first proposed by Tin Kam Ho [13] of Bell Labs in 1995. The method combines Breiman's "bagging" idea and Ho's "random subspace method" to construct a collection of decision trees with controlled variations.

Each tree is constructed using the following algorithm:

1. Let the number of training cases be *N*, and the number of variables in the classifier be *M*.
2. We are told the number *m* of input variables to be used to determine the decision at a node of the tree; *m* should be much less than *M*.
3. Choose a training set for this tree by choosing *N* times with replacement from all *N* available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
4. For each node of the tree, randomly choose *m* variables on which to base the decision at that node. Calculate the best split based on these *m* variables in the training set.
5. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

4. FEATURE SELECTION

4.1 FISHER ANALYSIS OF VARIANCE (ANOVA)

Analysis of variance (ANOVA) is the most commonly used technique for comparing the means of groups of measurement data. There are lots of different experimental designs that can be analyzed with different kinds of ANOVA. We're only going to talk here about single-classification ANOVA and two-way ANOVA.

- In a single-classification ANOVA (also known as a one-way ANOVA), there is one measurement variable and one attribute variable. Multiple observations of the measurement variable are made for each value of the attribute variable. For example, you could measure the amount of transcript of a particular gene for multiple samples taken from arm muscle, heart muscle, brain, liver, and lung. The transcript amount would be the measurement variable, and the tissue type (arm muscle, brain, etc.) would be the attribute variable.

The basic idea of one-way ANOVA is to calculate the mean of the observations within each group, then compare the variance among these means to the average variance within each group. Under the null hypothesis that the observations in the different groups all have the same mean, the among-group variance will be the same as the within-group variance. As the means get further apart, the variance among the means increases. The test statistic is thus the ratio of the variance among means divided by the average variance within groups, or F_s. This statistic has a known distribution under the null hypothesis, so the probability of obtaining the observed F_s under the null hypothesis can be calculated.

The shape of the F-distribution depends on two degrees of freedom, the degrees of freedom of the numerator (among-group variance) and degrees of freedom of the denominator (within-group variance). The among-group degree of freedom is the number of groups minus one. The within-groups degree of freedom is the total number of observations, minus the number of groups. Thus if there are *n* observations in *a* groups, numerator degrees of freedom is *a*-1 and denominator degrees of freedom is *n*-*a*.

Formally, the steps of one-way ANOVA can be summarized like this:

1. Tabulate the data. Each column in this table represents the occurrences of an attribute α.

2. For each attribute α, compute :

- the sum of the occurrences, Σx,
- the mean of the occurrences $\bar{x}=\Sigma x/\mathrm{n}$, where *n* is the number of occurrences.
- the square value of each occurrence x2, and the sum of these square values Σx^2
- the value $(\sum x)^2/n$
- the sum of squares of the deviation *d* for an observation *x* from the mean $\bar{x}$, $\sum d^2= \sum x^2 - (\sum x)^2/n$

3. For each attribute divide Σd^2 by *n*-1 to obtain the variance, σ^2. Divide the highest value of σ^2 by

the lowest value of σ^2 to obtain a variance ratio (F). Then look up a table of F_{max} for the number of treatments in the table of data and the degrees of freedom (number of replicates per treatment -1). If our variance ratio *does not exceed* the F_{max} value then we are safe to proceed. If not, the data might need to be transformed.

4. Sum all the values of Σx^2 and call the sum A.

5. Sum all the values for $(\sum x)^2/n$ and call the sum B.

6. Sum all the values for Σx to obtain the grand total.

7. Square the grand total and divide it by total number of observations; call this D.

8. Calculate the Total sum of squares (SS_{Total}) = A - D

9. Calculate the Between-treatments sum of squares ($SS_{Treatments}$) = B - D

10. Calculate the Residual sum of squares (SS_{Error}) = A - B

11. Construct a table as follows, where *** represents items to be inserted, and where:

u = number of treatments and *v* = number of replicates.

Source of variance	**Sum of squares (SS)**	**Degrees of freedom (df)**	**Mean square = SS / df**
Between treatments	***	u - 1	***
Residual	***	u(v-1)	***
Total	***	(uv)-1	

[The total *df* is always one fewer than the total number of data entries]

12**.** Using the *mean squares* in the final column of this table, do a *variance ratio test* to obtain an F value:

F = Between treatments mean square / Residual mean square (6)

13. Go to a table of F ($p = 0.05$) and read off the value where n_1 is the *df* of the between treatments mean square and n_2 is *df* of the residual mean square. If the calculated F value exceeds the tabulated value there is significant difference between treatments. If so, then look at the tabulated F values for $p = 0.01$ and then_0.001, to see if the treatment differences are more highly significant.

- In comparing 2 or more population means, there are often two or more factors of simultaneous interest. A two-way ANOVA include two factors in modelling the mean response. The Randomized Block Design (RBD) is one common experimental design which lends itself to a two-way ANOVA.

4.2 APPLYING FISHER FILTER

To apply the fisher's ANOVA to the intrusion detection problem, we consider that we have only one variable B representing the connection behavior. This variable B is characterized by a set of attributes A_i represented by the KDD connection features (defined in 1). So, we can write:

B = ψ (A1 A2 …A41) (7)

Where:

- B stands for the behaviour variable.

- A_i stand for the attributes (KDD connection features).

According to our assumptions, we used the univariate Fisher's ANOVA ranking technique to solve our problem. We applied the Fisher filter by using the Tanagra software [14]. While using this software, the inputs included all the KDD attributes (41 connection features) of the KDD data sets (defined in Table II). After filtering, only 14 features among the 41 were deduced using the univariate Fisher's ANOVA. The resulting subset "Set2" was

Table 1. KDD connection features (selected features are in bold characters).

TCP connections basic features		**Content features**		**Traffic features**		**Other features**	
A	**duration**	**J**	**Hot**	**W**	**count**	**AF**	**dst_host_count**
B	protocol-type	K	num_falied_logins	**X**	**srv_count**	**AG**	**dst_host_srv_count**
C	service	**L**	**logged_in**	Y	serror_rate	AH	dst_host_same_srv_rate
D	flag	**M**	**num_compromised**	Z	srv_serror_rate	AI	dst_host_diff_srv_rate
E	**src_bytes**	**N**	**root_shell**	AA	rerror_rate	AJ	dst_host_same_src_port_rate
F	dst_bytes	O	su_attempted	AB	srv_rerror_rate	AK	dst_host_srv_diff_host_rate
G	land	**P**	**num_root**	AC	same_srv_rate	AL	dst_host_serror_rate
H	wrong_fragment	**Q**	**num_file_creations**	AD	Diff_srv_rate	AM	dst_host_srv_serror_rate
I	urgent	**R**	**num_shells**	AE	srv_diff_host_rate	AN	dst_host_rerror_rate
		S	num_access_files			AO	dst_host_srv_rerror_rate
		T	num_outbound_cmds				
		U	is_host_login				
		V	**is_guest_login**				

used as the input of five classification trees to predict attacks. The deduced features are labelled according to table 1: A, E, J, L, M, N, O, P, Q, V, W, X, AF, AG. Most of the deduced features (8 among 14) here are content features within a connection suggested by domain knowledge.

This concludes the first phase of our approach. The second and last phase will be the use of an appropriate classification tree to classify and detect attacks.

5. EVALUATION DATASET

The 1999 version of MIT Lincoln Laboratory – DARPA (Defense Advanced Research Projects Agency) intrusion detection evaluation data was used in this research [15]. This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition (KDD-99). In one hand, the sample version of the dataset included 494021 connection records. In this version, there are four different known categories of computer attacks including Denial of Service (DoS) attacks, User to Root attacks (U2R), Remote to Local (R2L) attacks and probing (PRB) attacks. These four cited categories contain 22 training attack types. In the other hand, the KDD testing set contains 311029 records belonging to 38 attack types. In DARPA dataset each event (connection) is described with 41 features. These features are either continuous or discrete. 22 of these features describe the connection itself and 19 of them describe the properties of connections to the same host in last 2 seconds. These are called *same host features*. Some features examine only the connections in the past 2 seconds that have the same service as the current connection and are called *same service features*. Some other connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host, instead of a time window. This yields a set of *host-based traffic features*. Unlike DOS and PRB attacks, the U2R and R2L attacks do not have any sequential patterns. The later have the attacks embedded in the data packets. So, some features that look for suspicious behaviour are constructed and these are called *content features*.

Table 2. Distribution of normal and attack connections in KDD sets.

Set type Connection ty	Training set		Testing set	
Normal	97278	19,69%	60593	19,48%
DoS	391458	79,24%	229853	73,90%
PRB	4107	0,83%	4166	1,34%
R2L	1126	0,22%	16189	5,20%
U2R	52	0,01%	228	0,07%

6. EXPERIMENTS

This section describes the last step of our approach. After deducing the appropriate features, 4 types of classification trees: ID3, C4.5, CART, and RnDT, were tested to classify and detect attacks. According to our tests the RndT leads to better results. That is why we will present here only the results of RndT in details. We will then describe some other results to make comparisons.

Tanagra software [14] was used for the implementation of all the classification trees, on a Pentium 4 (2.88 GHz), with 512 Mb of memory. All the experiments and results will be presented and discussed according to some performance measures, among them we can cite:

- The confusion matrix: A confusion matrix (CM) is defined by associating classes as labels for the rows and columns of a square matrix: in the KDD dataset, there are five classes, {Normal, PRB, DoS, U2R, R2L}, and therefore the matrix has dimensions of 5×5. An entry at row *i* and column *j*, CM(i,j), represents the number of misclassified patterns, which originally belong to class *i* yet mistakenly identified as a member of class *j*.
- The Percent of Correct Classification (PCC) will be used to evaluate the classification efficiency of the instances belonging to the KDD testing set.
- In addition to the two mentioned measures, both the learning time (LT) and the computation time (CT), are also computed.

6.1 EXPERIMENTS WITHOUT ANOVA

First of all, as the data set has 5 different classes, the dimension of the confusion matrixes is (5×5). All the matrixes resulting from the training and testing steps are described in the following tables:

Training step results

For the training step, 7178 records were manually selected from the KDD training set. Each record is composed of 41 features. This leads to the following result (table 3):

According to table 3, all the connection types are practically well classified (CR>95%) with a very low error rate (0,005). The very high value of the PCC means that RndT has learned well the training set, and therefore can easily predict any instance of attack belonging to this set. It means also that the learning set is coherent. This learning step consumes 1,312 second.

Testing step results

For the testing step, 2570 records were manually selected from the KDD testing set. Each record is composed of 41 features. This leads to the following result (table 4):

According to table 4, except the CR of the R2L category, all the other CRs are low. In addition to that, the resulting PCC (71,71%) is very far from the one resulting from the training step. This can be due to the fact that some useless features were used and contribute to the degradation of the CRs accuracy. In addition to that, some new attacks belonging to the testing set, and that were not learned in the training step, may contribute also to the degradation of the obtained results.

6.2 EXPERIMENTS USING ANOVA

Training step results

The training set here comprises the 7178 records of the first experiment, but, each record is composed of only 14 features. This leads to the following result (table 5):

According to tables 3 and 5, RndT leads to better results when only 14 features are used. In fact, except the CR of U2R, all the CRs reached in table 5, are better than the ones obtained in table 3. In addition to that the obtained PCC value is lightly higher than the one obtained in table 3. Unlike the first experiment, this training step consumed only 0,125 s.

Testing step results

The testing set here comprises the 2570 records of the first experiment, but, each record is composed of only 14 features. This leads to the following result (table 6):

Table 3. The confusion matrix of the RndT in the training step.

Error rate = 0,0050									
Values prediction			**Confusion matrix**						
Value	**Recall**	**1-Precision**		**normal.**	**DoS.**	**PRB.**	**R2L**	**U2R.**	**Sum**
normal	0,9930	0,0040	**normal**	1986	2	1	11	0	2000
DoS.	0,9970	0,0010	**DoS.**	5	1994	0	1	0	2000
PRB.	0,9950	0,0010	**PRB.**	1	0	1990	9	0	2000
R2L	0,9964	0,0201	**R2L**	2	0	1	1122	1	1126
U2R.	0,9615	0,0196	**U2R.**	0	0	0	2	50	52
PCC=99,47%	**LT**=1,312 s		**Sum**	1994	1996	1992	1145	51	7178

Table 4. The confusion matrix of the RndT in the testing step.

Error rate = 0,2829									
Values prediction			**Confusion matrix**						
Value	**Recall**	**1-Precision**		**normal.**	**DoS.**	**PRB.**	**R2L.**	**U2R.**	**Sum**
normal	0,7340	0,2874	**normal**	734	3	188	75	0	1000
DoS.	1,0000	0,0050	**DoS.**	0	1000	0	0	0	1000
PRB.	0,3160	0,8078	**PRB.**	170	0	79	1	0	250
R2L	0,0880	0,8087	**R2L**	95	2	130	22	1	250
U2R.	0,1143	0,1111	**U2R.**	31	0	14	17	8	70
PCC=71,71%	**CT**=0,0016 s		**Sum**	1030	1005	411	115	9	2570

Table 5. The confusion matrix of the RndT in the training step.

Error rate = 0,0018									
Values prediction			**Confusion matrix**						
Value	**Recall**	**1-Precision**		**normal.**	**DoS.**	**PRB.**	**R2L.**	**U2R.**	**Sum**
normal	0,9970	0,0010	**normal**	1994	0	1	5	0	2000
DoS.	1,0000	0,0000	**DoS.**	0	2000	0	0	0	2000
PRB.	0,9985	0,0020	**PRB.**	2	0	1997	0	1	2000
R2L	0,9982	0,0053	**R2L**	0	0	2	1124	0	1126
U2R.	0,9615	0,0196	**U2R.**	0	0	1	1	50	52
PCC=99,82%	**LT**=0,125 s		**Sum**	1996	2000	2001	1130	51	7178

Table 6. The confusion matrix of the RndT in the testing step.

Error rate =0,0969									
Values prediction			**Confusion matrix**						
Value	**Recall**	**1-Precision**		**normal.**	**DoS.**	**PRB.**	**R2L.**	**U2R.**	**Sum**
normal	0,9500	0,1029	**normal**	950	6	27	11	6	1000
DoS.	1,0000	0,0060	**DoS.**	0	1000	0	0	0	1000
PRB.	1,0000	0,2733	**PRB.**	0	0	250	0	0	250
R2L	0,4200	0,2446	**R2L**	100	0	45	105	0	250
U2R.	0,2286	0,2727	**U2R.**	9	0	22	23	16	70
PCC=90,31%	**CT**=0 s		**Sum**	1059	1006	344	139	22	2570

According to the previous table, all the CRs here (corresponding to 14 features), are better than the ones obtained with 41 features (table 4). In addition to that, the obtaind PCC (90,31%) is higher than one obtained with 41 features. Nevertheless, the CRs of both R2L and U2R attack categories remain poor. This may be due to the fact that the training set is very different from the testing set, especially concerning the “new attacks” that do not appear in the former set.

Table 7 summarizes the performances of the RndT without/with the use of ANOVA.

Table 7: Performances of RndT with/without ANOVA.

	Without ANOVA	**Using ANOVA**
normal	0,7340	0,9500
DoS.	1,0000	1,0000
PRB.	0,3160	1,0000
R2L	0,0880	0,4200
U2R.	0,1143	0,2286
PCC	71,71%	90,31%
CT	0,0016 s	0 s

7. CONCLUSION

In this study we have performed data reduction and evaluated the performances of the RndT on the DARPA benchmark intrusion data. Firstly, to reduce data we used the Fisher’s ANOVA technique. After that, we deduced that the RndT used as intrusion detection model, leads to better results than some other tested classification trees.

Table 8 shows that the results of our approach are very competitive with both the results of Ben Amor and al.[16], and also with the results of Chebrolu and al.[6]. In fact, the CR of the normal category obtained using our approach is largely higher than the one obtained by Chebrolu[6] and Ben Amor[16]. In addition to that, the PCC obtained using our approach is also largely higher than the one obtained with the two other approaches.

Table 8. Comparison between our approach and some known IDSs.

	Our approach	**Chebrolu[6]**	**Chebrolu[6]**	**Ben Amor [16]**
	14 variables	**17 variables**	**12 variables**	**41 variables**
normal	**0,9500**	0,8080	0,7660	0,7970
DoS.	**1,0000**	1,0000	1,0000	1,0000
PRB.	**1,0000**	1,0000	0,9960	0,9880
R2L	0,4200	**0,6000**	0,5520	**0,6000**
U2R.	**0,2286**	0,2000	0,1429	0,1000
PCC	**90,31%**	86,46%	84,16%	85,64%

As a future work, it will be interesting to enhance both ANOVA and RndT in order to reach a higher CR of R2L and U2R categories.

8. REFERENCES

[1] D.-K. Kang, D. Fuller, V. Honavar, “Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation », in Proceedings of the 2005 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, pp. 118-125, 2005.

[2] W. Lee W, S. Stolfo, K. Mok. “A data mining framework for building intrusion detection models”. In Proceedings of the IEEE symposium on security and privacy; pp. 120-132,1999.

[] L. Xiangyang, Y. Nong, “Decision tree classifiers for computer intrusion detection”, book chapter, *Real-time system security*, Nova Science Publishers, pp. 77-93, 2003.

[4] G. Folino, C. Pizzuti and G. Spezzano, “GP Ensemble for Distributed Intrusion Detection Systems”, book chapter, *Pattern Recognition and Data Mining*, Volume 3686/2005, Springer Verlag, pp. 54-62, 2005.

[5] A. H. Sung, S. Mukkamala, “Identifying important features for intrusion detection using support vector machines and neural networks”.

In: Proceedings of International Symposium on Applications and the Internet (SAINT 2003); pp. 209-217, 2003.

[6] S. Chebrolu, A. Abraham, J. P. Thomas, “Feature deduction and ensemble design of intrusion detection systems”, Journal of Computers and Security, Elsevier, Volume 24, Issue 4 , pp. 295-307, 2005.

[7] R. Lippmann, S. Cunningham, “Improving intrusion detection performance using keyword selection and neural networks.” *Computer Networks*;34(4):594-603, 2000.

[8] M. Middlemiss, G. Dick, “Feature selection of intrusion detection data using a hybrid genetic algorithm/KNN approach”, book chapter, *Design and application of hybrid intelligent systems*, IOS press, pp. 519-527, 2003.

[9] S. Khusbu, and al., “Adaptive neuro-fuzzy intrusion detection system”. In: IEEE International Conference on Information Technology: Coding and Computing (ITCC’04), Portugal, vol. 1. USA: IEEE Computer Society; p. 70-74, 2004.

[10] L. Breiman, J. Friedman, R. A. Olshen and C. J. Stone, "Classification and regression trees". Monterey, CA Wadsworth & Brooks, 1984.

[11] Quinlan, J. R.: C4.5, Programs for machine learning. Morgan Kaufmann San Mateo Ca, 1993.

[12] L. Breiman, "Random Forests". Machine Learning 45 (1), 5-32, 2001.

[13] H. T. Kam, "Random Decision Forest". Proc. of the 3rd Int'l Conf. on Document Analysis and Recognition, Montreal, Canada, August 14-18, pp. 278-282, 1995.

[14] Ricco Rakotomalala, "Tanagra : a free software for training and research", in Actes de EGC'2005, RNTI-E-3, vol. 2, pp.697-702, 2005.

[15] KDD data set, 1999; http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, accessed on July 2006.

[16] N. Ben Amor, and al., “Naïve Bayes vs Decision Trees in Intrusion Detection Systems”, in the Proceeding of the ACM Symposium on Applied Computing, Cyprus, pp. 420-424, 2004.

__Rachid BEGHDAD__ received his computer science engineer degree in 1991 from the Polytechnical school of engineers, Algiers, Algeria. He received his Master computer science degree from Clermont-Ferrand University, France, in 1994. He earned his Ph.D. computer science degree from Toulouse University, France, in 1997.
He is a reviewer for some journals, such as the Computer Communications journal, Elsevier, UK
His main current interest is in the area of computer communication systems including intrusion detection methods, unicast and multicast routing protocols, real-time protocols, and wireless LAN protocols.