Asian Journal of Information Technology 9 (6): 300-306, 2010
ISSN: 1682-3915

A Comparative Study of Bagging, Boosting and C4.5: The Recent Improvements in Decision Tree Learning Algorithm

Subrata Pramanik, Utpala Nanda Chowdhury, Bimal Kumar Pramanik and Nazmul Huda
Department of Computer Science and Engineering,
University of Rajshahi, Rajshahi, Bangladesh

Abstract: Decision tree learning algorithm has been successfully used in expert systems in capturing knowledge and presents a powerful method of inferring classification rules from a set of labeled examples. ID3 is a well known and the most basic decision tree-learning algorithm that is based on information gain theory. Improvements are made to this decision tree induction algorithm by Quinlan's C4.5 algorithm that uses gain ratio as opposed to information gain. Breiman's Bagging and Freund and Schapire's Boosting are recent methods of improving the predictive power of any classifier learning system. Both form a set of classifiers that are combined by voting, bagging by generating samples with replication of the data and boosting by adjusting the weights of training instances. In this research work both bagging and boosting have been applied to C4.5 algorithm and the corresponding predictive accuracies are computed by testing on a representative dataset. While both approaches substantially improve predictive accuracy of C4.5, boosting shows the greater benefit.

Key words: Data mining, decision tree, decision tree induction, ID3, C4.5, bagging, boosting

INTRODUCTION

The amount of data kept in computer files and databases is growing at a phenomenal rate. At the same time, the users of these data are expecting more sophisticated information from them. A marketing manager is no longer satisfied with a simple listing of marketing contacts but wants detailed information about customers' past purchases as well as predictions of future purchase. Simple structured or query languages are not adequate to support these increased demands for information. Data mining steps in to solve these needs. Data mining is often defined as finding hidden information in a database. Alternatively, it has been called data analysis, data driven discovery and deductive learning. Classification and Prediction are two forms of data analysis that can be used to extract patterns describing important data classes or to predict future data trends. Researchers in machine learning, experts systems, statistics and neurobiology have proposed many classification and prediction methods.

Recent database mining research has built scalable classification and prediction techniques capable of handling large disk-resident data and Decision Tree Induction (DTI) is one of them. The most well-known algorithm in the literature for building decision trees is the C4.5 (Quinlan, 1993). C4.5 is an extension of Quinlan's earlier ID3 algorithm. One of the latest studies that compare decision trees and other learning algorithms has been done by Lim *et al.* (2000). The study shows that C4.5 has a very good combination of error rate and speed. In 2002, Ruggieri presented an analytic evaluation of the runtime behavior of the C4.5 algorithm which highlighted some efficiency improvements (Ruggieri, 2002). Based on this analytic evaluation, he implemented a more efficient version of the algorithm, called EC4.5. He argued that his implementation computed the same decision trees as C4.5 with a performance gain of up to five times.

There has recently been renewed interest in increasing accuracy by generating and aggregating multiple classifiers. Although, the idea of growing multiple trees is not new, the justification for such methods is often empirical. In contrast, two new approaches for producing and using several classifiers are applicable to a wide variety of learning systems and are based on theoretical analysis of the behavior of the composite classifier. They are Bagging and Boosting.

Goal and overview of this research: The goal of this research work was to extract the meaningful knowledge lied in the database and transform them into meaningful

Corresponding Author: Subrata Pramanik, Department of Computer Science and Engineering, University of Rajshahi, Rajshahi, Bangladesh

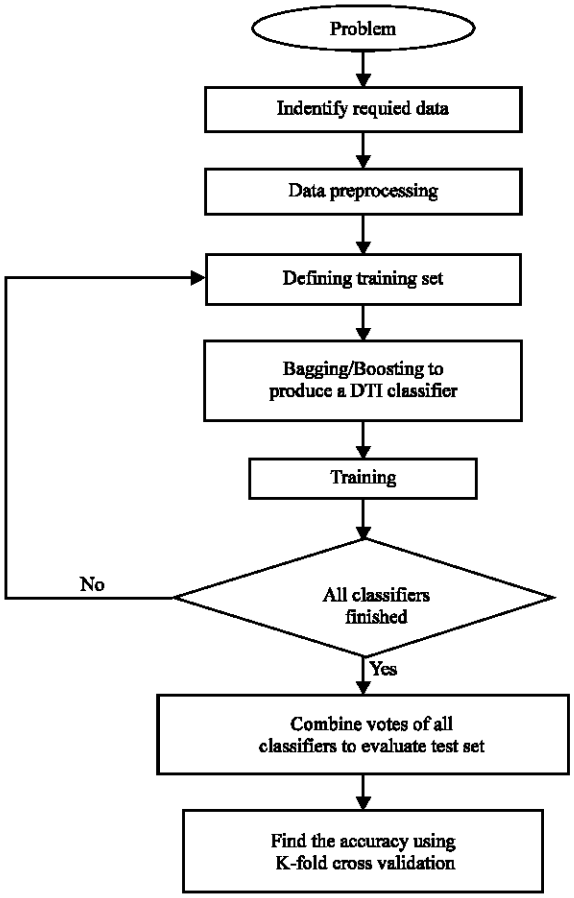

Fig. 1: Block diagram of the research work process

rules. Then the rules are used to predict the class labels of unknown data. Finally we introduced Bagging and Boosting to improve the accuracy of this whole process.

Keeping the aimed goal of this research in mind we constructed the whole research process as shown in the block diagram in Fig. 1. Here the decision tree induction algorithms are used to turn the hidden knowledge into a large dataset into decision rules. Again enhancements are made to these algorithms to extract and use the rules more precisely to improve the accuracy. In the research we have used heart disease dataset which is collected from UCI machine learning repository. At first, ID3 algorithm is used to extract rules from the dataset and to use the rules to classify new data which is implemented in MATLAB7. C4.5, the successor of ID3 is then used to classify data more accurately. Finally, two new approaches named Bagging and Boosting are introduced to improve the predictive accuracy of C4.5.

Background study

Classification and prediction: Data classification is a two-step process (Han and Kamber, 2007; Dunham, 2004). In the first step, a model is built describing predetermined set of data classes or concepts. The model is constructed by analyzing database tuples described by attributes. Each tuple is assumed to belong to a predefined class as determined by one of the attributes, called the class label attribute. In the context of classification, data tuples are also referred to as samples or objects. The data tuples analyzed to build the model collectively form the training dataset. The individual tuples making up the training set are referred to as training samples and are randomly selected from the sample population (Han and Kamber, 2007). Since the class label of each training sample is provided, this step is also known as supervised learning. It contrasts with unsupervised learning, in which the class label of each training sample is not known and the number or set of classes to be learned may not be known in advance (Han and Kamber, 2007; Dunham, 2004).

Prediction can be viewed as the construction and use of a model to assess the class of an unlabeled sample or to asses the value or value ranges of an attribute that a given sample is likely to have. In this view, classification and regression are the two major types of prediction problems where classification is used to predict discrete or nominal values, while regression is used to predict continuous or ordered values. In the view, however, refer to the use of prediction to predict class label as classification and the use of prediction to predict continuous values as prediction.

Decision tree induction: Decision tree induction is a greedy algorithm that constructs decision tree in a top-down recursive divide and conquer manner. A decision tree is a tree in which each branch node represents a choice between a numbers of alternatives and each leaf node represents a decision. Decision trees are commonly used for gaining information for the purpose of decision-making. It starts with a root node and forms this node; users split each node recursively according to decision tree learning algorithm. The final result is a decision tree in which each branch represents a possible scenario of decision and its outcome.

For extracting rules, information gain measure is used to select the test attribute at each node in the tree. The attribute with the highest information gain is chosen as the test attribute for the current node and the path from the root node to each leaf node in the tree is tracked to construct rules from the dataset.

They use induction in order to provide an appropriate classification of objects in terms of their attributes, inferring decision tree rules. In their learning phase, explicit rules or interactions among relevant features are induced. Such a learning method differs from non-linear classifiers such as support vector machines or

neural networks where the learning phase is to determine the parameters of the non-linear kernel functions.

ID3 algorithm: The ID3 (Iterative Dichotomiser 3) technique (Quinlan, 1986; Han and Kamber, 2007; Dunham, 2004) to building a decision tree is based on information theory and attempts to minimize the expected number of comparisons. The basic idea of the induction algorithm is to ask questions whose answers provide the most information. The first question divides the search space into two large search domains while the second performs little division of the space. The basic strategy used by ID3 is to choose splitting attributes with the highest information gain first. The amount of information associated with an attribute value is related to the probability of occurrence.

Let node N represents or hold the tuples of partition D. The attribute with the highest information gain is chosen as the splitting attribute for node N. This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or impurity in these partitions. To calculate the gain (Quinlan, 1986; Han and Kamber, 2007; Dunham, 2004) of an attribute, at first we calculate the entropy of that attribute by the following formula:

$$\text{Entropy}(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (1)$$

where, P_j is the probability that an arbitrary tuple in S belongs to class C_j and estimated by $|C_{i,D}| / |D|$. A log function to the base 2 is used because the information is encoded in bits. Entropy (S) is just the average amount of information needed to identify the class label of the tuple in S. Now, the gain of an attribute is calculated by the formula (Han and Kamber, 2007):

$$\text{Info}_A(S) = \sum_{i=1}^{n} \frac{|S_i|}{|S|} \text{Entropy}(S_i) \quad (2)$$

where, $S_i = \{S_1, S_2 \ldots\ldots S_n\}$ = partitions of S according to values of attribute A:

n = Number of attributes A
$|S_i|$ = Number of cases in the partition S_i
|S| = Total number of cases in S

Information gain is defined as the difference between the original information requirement and new requirement. That is (Han and Kamber, 2007):

$$\text{Gain}(A) = \text{Entropy}(S) - \text{Info}_A(S) \quad (3)$$

In other words, Gain (A) tell us how much would be gained by branching on A. It is the expected reduction in the information requirement caused by knowing the value of A. The attribute A with highest information gain is chosen as the splitting attribute at node N.

MATERIALS AND METHODS

New decission tree learning algorithms: The C4.5 algorithm is (Quinlan, 1993; Han and Kamber, 2007; Dunham, 2004) extension of his own ID3 algorithm for generating decision trees. Bagging and Boosting are general strategies for improving classifier and predictor accuracy. Suppose that we are a patient and would like to have a diagnosis made based on the symptoms. Instead of asking one doctor, we may choose to ask several. If a certain diagnosis occurs more than any others, we may choose this as the final or best diagnosis. That is the final diagnosis is made based on a majority vote where each doctor gets an equal vote. Now replace each doctor by a classifier, we have the basic idea behind bagging.

In boosting, we assign weights to the value of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made. The final diagnosis is then a combination of the weighted diagnoses.

C4.5 Algorithm: Just as with CART, the C4.5 algorithm recursively visits each decision node, selecting the optimal split, until no further splits are possible. The steps of C4.5 algorithm (Han and Kamber, 2007; Dunham, 2004; Beck *et al*., 2007) for growing a decision tree is given below:

- Choose attribute for root node
- Create branch for each value of that attribute
- Split cases according to branches
- Repeat process for each branch until all cases in the branch have the same class

A question that, how an attribute is chosen as a root node? At first, we calculate of the gain ratio of each attribute. The root node will be that attribute whose gain ratio is maximum. Gain ratio is calculated by the formula (Beck *et al*., 2007; Quinlan, 1993):

$$\text{Gain Ratio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}(A)} \quad (4)$$

where, A is an attribute whose gain ratio will be calculated. The attribute A with the maximum gain ratio is selected as the splitting attribute. This attribute minimizes the information needed to classify the tuples in the resulting partitions. Such an approach minimizes the expected number of tests needed to classify a given tuple and guarantees that a simple tree if found. To calculate the gain of an attribute, at first we calculate the entropy of that attribute by the following formula (Beck *et al*., 2007; Quinlan, 1993):

$$\text{Entropy}(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (5)$$

where, P_i is the probability that an arbitrary tuple in S belongs to class C_i and estimated by $|C_{i,D}|/|D|$. A log function to the base 2 is used because the information is encoded in bits. Entropy (S) is just the average amount of information needed to identify the class label of the tuple in S. Now gain of an attribute is calculated by the formula (Beck *et al*., 2007; Quinlan, 1993):

$$\text{Gain}(A) = \text{Entropy}(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \text{Entropy}(S_i) \quad (6)$$

where, $S_i = \{S_1, S_2 ..... S_n\}$ = partitions of S according to values of attribute A:

n = Number of attributes A
|Si| = Number of cases in the partition S_i
|S| = Total number of cases in S

The gain ratio divides the gain by the evaluated split information. This penalizes splits with many outcomes. (Beck *et al*., 2007; Quinlan, 1993):

$$\text{SplitInfo}(A) = -\sum_{i=1}^{n} \frac{|Si|}{|S|} \log_i \frac{|Si|}{|S|} \quad (7)$$

The split information is the weighted average calculation of the information using the proportion of cases which are passed to each child. When there are cases with unknown outcomes on the split attribute, the split information treats this as an additional split direction. This is done to penalize splits which are made using cases with missing values. After finding the best split, the tree continues to be grown recursively using the same process.

Bagging: We first take an intuitive look at how bagging (Han and Kamber, 2007) researchers as a method of increasing accuracy. Suppose that we are a patient and would like to have a diagnosis made based on the symptoms. Instead of asking one doctor, you may choose to ask several. If a certain diagnosis occurs more than any others, you may choose this as the final or best diagnosis. That is the final diagnosis is made based on a majority vote where each doctor gets an equal vote. Now replace each doctor by a classifier, you have the basic idea behind bagging. Intuitively, a majority vote made by a large group of doctors may be more reliable than a majority vote made by a small group.

Given a set, D, of d tuples, bagging works as follows (Han and Kamber, 2007). For iteration i (I =1, 2, 3,....., k), a training set, D_i of d tuples is sampled with replacement from the original set of tuples, D. Note that the term bagging stands bootstrap aggregation. Each training set is a bootstrap sample. Because sampling with replacement is used, some of the original tuples of D may not be included in D_i, whereas others may occurs more than once. A classifier model M_i is learned for each training set, D_i. To classify an unknown tuple, X, each classifier, Mi, returns its class prediction, which counts as one vote. The bagged classifier, M*, counts the votes and assigns the class with the most vote to X. Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple.

Algorithm: Bagging: The bagging algorithm (Han and Kamber, 2007) creates an ensemble of models (classifiers or predictors) for a learning scheme where each model gives an equally-weighted prediction.

Input:

- D, a set of training tuples
- k, the number of models in the ensemble
- A learning scheme (e.g., decision tree algorithm, backpropagation, etc.)

Output: A composite model, M*

Method:

- For I = 1 to k do// create k models
- Create bootstrap sample, Di by sampling D with replacement
- Use D_i to derive a model, M_i
- Endfor

To use the composite model on a tuple, X:

- If classification then
- Let each of the k models classify X and return the majority vote

- If prediction then
- Let each of the k models predict a value for X and return the average predicted value

The bagged classifier often has significantly greater accuracy than a single classifier derived from D, the original training data. It will not be considerably worse and is more robust to the effects of noisy data. The increased accuracy occurs because the composite model reduces the variance of the individual classifiers. For prediction, it was theoretically proven that a bagged predictor will always have improved accuracy over a single predictor derived from D.

Boosting: Boosting is a general method for improving accuracy of any given learning algorithm. It is an effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb. In the research we have focused especially on the AdaBoost algorithm (Freund and Schapire, 1995; Han and Kamber, 2007).

Adaboost algorithm: In AdaBoost, the input includes a dataset D of d class-labeled tuples, an integer k specifying the number of classifiers in the ensemble and a classification-learning scheme.

Each tuple in the dataset is assigned a weight. The higher the weight is the more it influences the learned theory. Initially, all weights are assigned a same value of 1/d. The algorithm repeats k times. At each time, a model M_i is built on current dataset D_i which is obtained by sampling with replacement on original training dataset D. The framework (Han and Kamber, 2007) of this algorithm is as follows:

Algorithm: AdaBoost
Input:

- D, a set of d class-labeled training tuples
- K, the number of rounds
- A classification learning scheme

Output: A composite model
Method:

- Initialize the weight of each tuple in D to 1/d
- For I = 1-k do
- Sample D with replacement according to the tuple weights to obtain D_i
- Use training set Di to drive a model, M_i
- Compute the error rate error(Mi) of M_i
- If error(M_i) >0.5 then
- Reinitialize the weights to 1/d
- Go back to step 3 and try again
- Endif
- Update and normalize the weight of each tuple;
- Endfor

The error rate of M_i is the sum of the weights of all tuples in D_i that of the tuples in D_i that M_i misclassified:

$$\text{error}(M_i) = \sum_{j=1}^{d} w_j \times \text{err}(X_j) \quad (8)$$

Where, err (X_j) = 1, if X_j is misclassified and err (X_j) = 0 otherwise. Then the weight of each tuple is updated so that the weights of misclassified tuples are increased and the weights of correctly classified tuples are decreased. This can be done by multiplying the weights of each correctly classified tuple by error (M_i)/(1- error (M_i)). The weights of all tuples are then normalized so that the sum of them of them is equal to 1. In order to keep this constraint, the weight of each tuple is divided by the sum of the new weights.

After K rounds, a composite model will be generated, or an ensemble of classifiers which is then used to classify new data. When a new tuple X comes, it is classified through these steps:

- Initialize weight of each class to 0
- For i = 1-k do
- Get weight w_i of classifier M_i
- Get class prediction for X from M_i:c = M_i (X_i)
- Add β_i to weight for class c
- endfor
- Return the class with the largest weight

The weight w_i of each classifier M_i is calculated by this Eq. 9:

$$w_i = \log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)} \quad (9)$$

Requirements for bagging and boosting: These two methods for utilizing multiple classifiers make different assumptions about the learning system. As above, bagging requires that the learning system should not be stable, so that small changes to the training set should lead to different classifiers. Breiman also notes that, poor predictors can be transformed into worse ones by bagging. Boosting, on the other hand, does not preclude the use of learning systems that produce poor predictors, provided that their error on the given distribution can be kept below 50%. However, boosting implicitly requires the same instability as bagging; if C_t is the same C_{t-1} the weight adjustment scheme has the probability that error (M_t) = 0.5.

RESULTS AND DISCUSSION

To experiment the research concept on a representative dataset, a system is developed using Matlab7 which is powerful tool for complex calculation and high-level programming. The developed system has four individual sections: the first part implements and evaluates the ID3 algorithm in the second part ID3 algorithm is extended to evaluate C4.5, then bagging technique is used in the third portion and the resultant accuracy is calculated and finally, in the fourth part, boosting method is applied on C4.5 to show the improvements.

Experimental data: In the research as experimental data we have used a biomedical dataset for detecting heart disease and it is collected from the UCI Machine Learning Repository. The repository database is freely available and can be obtained from the following link: http://b-course.cs.helsinki.fi/obc/cl_readymade.html.

The heart disease dataset of 270 patients is used in this experiment This dataset contains 13 attributes and a class variable with two possible values which are shown in Table 1. Figure 2 shows obtained original dataset.

This data contains some attributes (such as age, resting blood pressure, Serum cholesterol in mg dL^{-1}, maximum heart rate achieve, ST depression induced by exercise relative to rest and The slope of the peak exercise ST segment) which contains continuous values. So, before using this dataset in this experiment, those continuous valued attributes are divided into ranges. The resultant transformed dataset is shown in Fig. 3. Both the datasets are transformed into Matlab readable text files.

Development method: The result of this research work was obtained through four different sections:

- At first, calculate the accuracy of the classification by applying decision tree induction using ID3 algorithm
- Secondly, calculate the accuracy of decision tree induction by applying C4.5 classification algorithm
- Then apply the Bagging method of generating multiple C4.5 classifiers with the same dataset and calculate the accuracy
- Finally, introduce Boosting method to C4.5 and calculate the improvement in the predictive accuracy

Each of the sections contains the basic three steps:

- Extract rules from training dataset called training process

Table 1: Description of the features in the heart disease dataset

Attributes	Values
Age	Numarical
Sex	Male, female
Chest pain type	1, 2, 3, 4
Resting blood pressre	Numarical
Serum cholesterol in mg dL^{-1}	Numarical
Fasting blood sugar>120 mg d^{-1}	Yes, no
Resting electrocardiographic results	0, 1, 2, 3
Maximum heart rate achieve	Numarical
Exercise induced angina	Yes, no
ST deprssion induced by exercise relative to test	Numarical
The slope the peak exercise ST segment	Numarical
Number of major vessels colored by fluoroscopy	0, 1, 2, 3
Thal	Normal, fixed defect, Reversible defect
Absence or presence of heart disease	Absence, presence

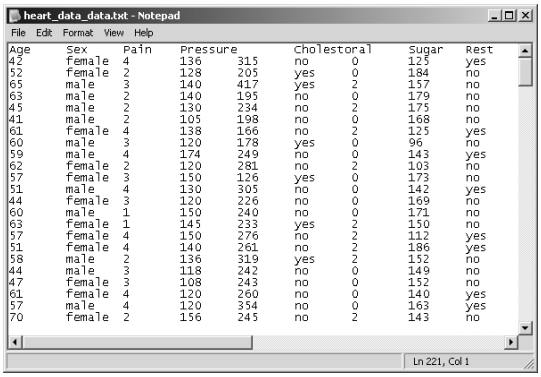

Fig. 2: The original obtained dataset

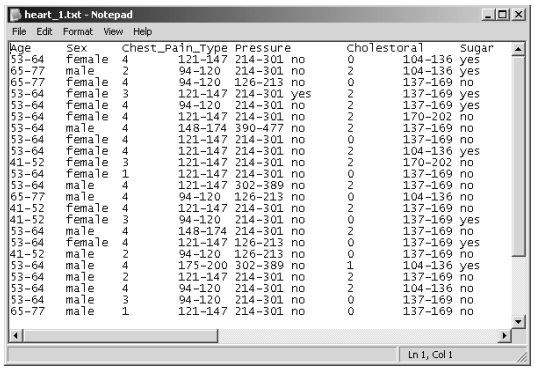

Fig. 3: The transformed dataset for learning

- Apply rules to testing dataset to predict classes name testing
- Estimate the accuracy of the testing process

Accuracy estimation: The goal of this research work was to improve the predictive accuracy. Calculated the accuracy of each approache by 10 fold cross validation method. This is a method for estimating generalization error based on resampling. About 10 fold cross validation

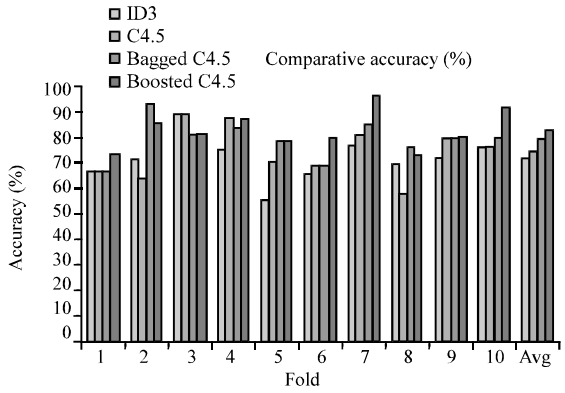

Fig. 4: Comparative accuracy of ID3, C4.5, Bagged C4.5 and boosted C4.5 algorithm

breaks the dataset into 10 datasets of size n/10 each where n denotes size of the entire train dataset. Training is performed on 9 data subsets entirely and remaining dataset is used for testing purpose. The process is repeated 10 times and overall accuracy of the system is calculated as the average of 10 calculated accuracies. It can be used to estimate the generalization error of a given model or it can be used for model selection by choosing one of several models that has the smallest estimated generalization error.

Result: The Comparative result of the implemented methods is shown in Fig 4. Following the above results we can conclude that the steps taken in this research work gradually increase the predictive accuracy of decision tree induction classification algorithm.

Experiment over the heart disease dataset has confirmed that boosted and bagged versions of C4.5 produce noticeably more accurate classifiers than the standard version. Boosting and bagging both have a sound theoretical base and also have the advantage that the extra computation they require is known in advance-if T classifiers are generated then both require T times the computational effort of C4.5. In this experiment, a 10 fold increase in computation buys a healthy average increase (between 7 and 12%) in the classification accuracy. In many applications, improvements of this magnitude would be well worth the computational cost.

CONCLUSION

The building and testing of new or less frequently applied algorithms is always worth doing, since they can have a positive effect when combined with popular models. Consider the fact that we managed to build a competitive model which significantly increase the classification accuracy. Having rich feature representation of the problem (which permits a feature set split and recombine procedure) often turns out to be just as important as the choice of the learning method.

The result demonstrated that Boosting seems to be more effective than Bagging when applied to C4.5, though the performance of bagged C4.5 is less variable than its boosted counterpart. Although, for some folds, bagging shows better performance but overall, boosting method seems to be more accurate. What is more, this research well to extract meaningful rules from a dataset and classify new data more accurately.

There are of course many ways in which the system can be improved. Perhaps the two most obvious ones are to implement the system to learn from and to classify data with more than one class and also to add post-pruning technique in the decision tree induction phase to produce more meaningful rules from the dataset as well as to enlarge the size and improve the quality of out training data.

REFERENCES

Beck, J.R., M.E. Garcia, M. Zhong, M. Georgiopoulos and G. Anagnostopoulos, 2007. A backward adjusting strategy for the C4.5 decision tree classifier. AMALTHEA REU SITE; Beck, *et al.* 2007. http://www.machine-learning.net/amalthea-reu.org/pubs/amalthea_tr_2007_01.pdf.

Dunham, M.H., 2004. Data Mining Introductory and Advanced Topics. Pearson Education (Singapore) Pvt. Ltd., Delhi, India.

Freund, Y. and R.E. Schapire, 1995. A decision-theoretic generalization of on-line learning and an application to boosting. Proceedings of the 2nd European Conference on Computational Learning Theory, March 1995, Murray Hill, New Jersey, USA., pp: 23-37.

Han, J. and M. Kamber, 2007. Data Mining Concepts and Techniques. 2nd Edn., Morgan Kaufmann Publishers, San Francisco, CA. USA.

Lim, T.S., W.Y. Loh and Y.S. Shih, 2000. A comparison of prediction accuracy, complexity and training time of thirty-tree old and new classification algorithms. Machine Learn., 40: 203-228.

Quinlan, J.R., 1986. Induction of decision trees. Machine Learn., 1: 81-106.

Quinlan, R.J., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publisher Inc., San Francisco, CA. USA., ISBN: 1-55860-238-0.

Ruggieri, S., 2002. Efficient C4.5. IEEE Trans. Knowledge Data Eng., 14: 438-444.