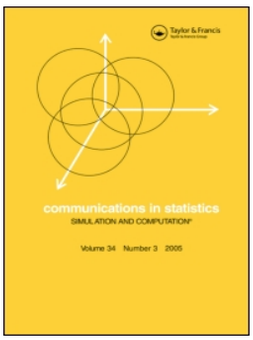

Taylor & Francis
Taylor & Francis Group

Communications in Statistics - Simulation and Computation

ISSN: 0361-0918 (Print) 1532-4141 (Online) Journal homepage: https://www.tandfonline.com/loi/lssp20

Three Steps Strategy to Search for Optimum Classification Trees

Muhammad Azam, Muhammad Aslam & Karl Peter Pfeiffer

To cite this article: Muhammad Azam, Muhammad Aslam & Karl Peter Pfeiffer (2016) Three Steps Strategy to Search for Optimum Classification Trees, Communications in Statistics - Simulation and Computation, 45:2, 548-565, DOI: 10.1080/03610918.2013.867991

To link to this article: https://doi.org/10.1080/03610918.2013.867991

Accepted author version posted online: 23 Jun 2014.
Published online: 30 Oct 2015.

Communications in Statistics—Simulation and Computation®, 45: 548–565, 2016

ISSN: 0361-0918 print / 1532-4141 online
DOI: 10.1080/03610918.2013.867991

Taylor & Francis
Taylor & Francis Group

Three Steps Strategy to Search for Optimum Classification Trees

MUHAMMAD AZAM,[1] MUHAMMAD ASLAM,[2]
AND KARL PETER PFEIFFER[3]

[1]Department of Statistics, Forman Christian College University, Lahore, Pakistan
[2]Department of Statistics, Faculty of Sciences, King Abdulaziz University, Jeddah, Saudi Arabia
[3]Department of Medical Statistics, Informatics and Health Economics, Innsbruck Medical University, Austria

*This article presents a new strategy to construct classification trees. According to the proposed scheme, we focused on keeping the record of sequences of each constructed classification tree; both in terms of splitting predictors and their splitting values in an array. So overall we have as many arrays as we have drawn samples. At this stage, a three steps strategy is introduced, which is used to search for the optimum classification tree. The proposed strategy provides comparable or improved results in terms of generalized error rates than tree and rpart (packages available for classification purposes in the **R**) using four of the well-known evaluation functions, that is, the Gini, the Entropy, the Twoing, and the Exponent-based function to split nodes for many real-life datasets.*

Keywords Evaluation functions; Optimum classification trees; Three steps strategy; Value-wise tree sequence; Variable-wise tree sequence.

Mathematics Subject Classification 62-XX; 62H30

1. Introduction

In data mining research, classification and regression trees, collectively named as decision trees, are non-parametric computationally intensive methods that have greatly increased in popularity during the last two decades (Rao et al., 2005).

A decision tree is structured as a sequence of simple questions, and the replies to these questions trace a path down the tree (Bremner and Taplin, 2002). This technique uses binary trees methodology, which was used in AID (Automatic Interaction Detection) and THAID (Theta AID) by Morgan and Sonquist (1963) and Morgan and Messenger (1973) for the construction of regression and classification trees, respectively. But these methods could not get popularity because of their spurious results noticed by Doyle (1973) and Doyle and Fenwick (1975). The only difference between classification and regression trees is the response variable. The decision trees are named as classification trees if the response variable is categorical, otherwise they are said to be regression trees.

Received January 28,2013; Accepted November 12,2013
Address correspondence to Muhammad Azam, Department of Statistics, Forman Christian College University, Lahore, Pakistan; E-mail: mazam72@yahoo.com

In the 1980s, a group of statisticians Breiman, Friedman, Olshen, and Stone introduced this non-parametric approach to address the problem of discrimination and regression called *classification and regression trees* (Taylor and Silverman, 1993). These methods were later implemented in the CART computer software by Salford (1995). Decision trees can now be constructed using many different software packages, some of which are relatively costly and are marketed for commercial usage (such as MATLAB code for generating trees by Martinez and Martinez, 2002 and SPSS) while others are freely available (e.g., tree and rpart libraries in the **R** software). In the following subsections, node splitting evaluation functions, their selection criterion, goodness of split criterion, and misclassification rates are discussed.

1.1. *Node Splitting Functions*

This function splits an intermediate node t with class proportions P_j, into two sub-nodes $t_L$and t_R. Some commonly used families of evaluation functions (i.e., impurity-based and non-impurity-based functions) are described as below.

1.1.1. *Gini function.* This function was proposed and used as an evaluation function by Breiman et al. (1984). For a node t, it is given by

$$i\,(t)_{\text{Gini}} = 1 - \sum_{j=1}^{J} P_j^2,$$

where j is total number of classes or categories, P_j is the proportion of jth class in a node t, such that

$$\sum_{j=1}^{J} P_j = 1.$$

we can define the Gini function for two descendent nodes as

$$i\,(t_L)_{\text{Gini}} = 1 - \sum_{j=1}^{J} P_{j/L}^2 \text{ and } i\,(t_R)_{\text{Gini}} = 1 - \sum_{j=1}^{J} P_{j/R}^2,$$

where t_L and t_R correspond to left and right descendents, and $P_{j/L}$ and $P_{j/R}$are the proportions of jth class on left and right descendent node, respectively, such that

$$\sum_{j=1}^{J} P_{j/L} = \sum_{j=1}^{J} P_{j/R} = 1.$$

1.1.2. *Entropy function.* Evaluation function proposed by Quinlan (1986) is based on the definition of entropy from information theory. For a node t, the entropy function is defined by

$$i\,(t)_{\text{Entropy}} = -\sum_{j=1}^{J} P_j \ln P_j,$$

provided that 0ln0 is taken to be 0. The logarithm may be taken to any convenient base (Fayyad and Irani, 1992).

1.1.3. *Exponent-based function.* Azam et al. (2009) proposed an evaluation function for the measurement of node impurity. It is defined by

$$i\,(t)_{\text{Exponent}} = 1 - \frac{1}{e}\sum_{j=1}^{J} P_j e^{P_j}.$$

The values of $i\,(t_L)$ and $i\,(t_R)$ are computed in a similar way as computed by Gini using function.

1.1.4. *Twoing rule.* A non-impurity-based node splitting criterion that measures the goodness-of-split value directly was introduced by Breiman et al. (1984). It measures the difference in probabilities that a class appears in the left descendant rather than in the right descendant node. The criterion is thus based on a concept of class separation rather than node heterogeneity. The criterion selects the best splitting value, which maximizes

$$g(x, s, t)_{\text{Towing}} = \frac{P_L P_R}{4}\left[\sum\nolimits_{J=1}^{J} |P_{j/L} - P_{j/R}|\right]^2,$$

where $P_L = N_L/N$and $P_R = N_R/N$, such that $P_L + P_R = 1$ Similarly, N_L, N_R, P_L, and P_R are numbers and proportions of data objects in the left and right descendent node, respectively, while N is the total number of data objects in a root node. The aim is to get a probability that a class j unit goes to the left as different as possible from the probability that it goes to the right. The factor (P_L $P_{R)}$ is designed to favor relatively even splits. This factor takes a maximum value of 0.25, when $P_L = P_R = 0.5$ and it declines if any of the proportions is close to 0 or 1.

1.2. *A Goodness-of-split Criterion*

At each intermediate node t, the split selected is the one that split s^*, which maximizes $g(x, s, t)$. It is called the "goodness-of-split" or "decrease-of-impurity" measure. Its value is computed by using the formula:

$$g\,(x, s, t) = i\,(t) - P_L i\,(t_L) - P_R i\,(t_R)\,.$$

1.3. *Selection Criterion to Split a Node*

Once an evaluation function to split a node is selected, the next step is to choose the best splitting predictor and its splitting point. The general objective of split selection rule is to divide the data objects at each node into two sub-nodes in such a way that both external sub-nodes are heterogenous, but internally they are as homogenous as possible. This process continues until each data object represents any of a specified class label.

A common approach to split selection in classification trees is to search through all possible splits generated by predictor variables. A splitting criterion is then used to evaluate these splits and select the one which maximizes the "goodness-of-split measure" and transfer data objects to corresponding sub nodes. There are $l = L-1$ (L is the number of distinct

Table 1
Confusion matrix

	Predicted class						
True class	*1*	*2*	*3*	*j*	...	*C*	Total
1	n_{11}	n_{12}	n_{13}			n_{1C}	$n_{1.}$
2	n_{21}	n_{22}	n_{23}			n_{2C}	$n_{2.}$
3	n_{31}	n_{32}	n_{33}			n_{3C}	$n_{3.}$
i	n_{ij}						
...							
C	n_{C1}	n_{C2}	n_{C3}			n_{CC}	$n_{C.}$
Total	$n_{.1}$	$n_{.2}$	$n_{.3}$			$n_{.C}$	N

values of a continuous variable) possible number of split points for a continuous predictor and $l = 2^{(k-1)} - 1$ (where k is the possible number of classes) is for a categorical predictor.

1.4. Misclassification Rates (Loss Function)

In general, one can divide the error rates committed by the classification tree T into two types, training error rate $R^{\text{tr}}(T)$ and generalization error rate $R^*(T)$. The training error is computed by dividing the number of misclassified data objects by the number of data objects in a training set.

Let m_i be the possible number of misclassified data objects in the ith node of the constructed classification tree T and N^{tr} is the number of data objects in the training set. Then the training error rate is given by

$$R^{\text{tr}}(T) = \frac{\text{total number of misclaffication data objects in a classification tree T}}{\text{total number of data objects in the training data}}$$

$$R^{\text{tr}}(T) = \frac{\sum_i m_i}{N^{\text{tr}}}.$$

Furthermore, for the comparison of prediction accuracies of constructed classifiers, one has to compute the misclassification rate of a particular tree for data objects that are new to the constructed tree (Rao et al., 2005). This value of the true misclassification rate is known as the generalization error. A confusion matrix is constructed for a C class problem to measure the generalization error rate (GER) or the true error rate denoted by $R^*(T)$.

In Table 1, n_{ij} represents number of data objects that are common between ith true class and jth predicted class. Diagonal elements $n_{11}, n_{22}, \ldots, n_{CC}$ represent the correctly classified data objects (i.e., data objects whose predicted and true class labels are the same). While off-diagonal elements represent the misclassified/dissimilar data objects. Similarly, $n_{.1}, n_{.2}, \ldots, n_{.C}$ and $n_{1.}, n_{2.}, \ldots, n_{C.}$ are totals of data objects of the corresponding predicted and true class label, respectively. In case, all data objects are classified correctly, then all off-diagonal entries in the table become zero. Similarly, if none of the data objects are classified correctly, then all diagonal entries would become zero.

A confusion matrix is "similar" to a matching counts matrix. However, in a supervised setting, where the number of classes are established by the training dataset, we do not find any need for using a different number of classes in the classifier (i.e., C should always

be equal to K if $i = 1, 2, ..., C$ and $j = 1, 2, ..., K$). In some situations, we "binarize" our classification problem and build classifiers for each class (vs. the rest), but we end up with C 2×2 confusion matrices.

Since error rates and accuracy rates are complementary, we can see accuracy as a measure of similarity between the class labels in our dataset and our predicted class labels (resulting from applying a classification model to a test set). Likewise, error rate can be seen as a measure of dissimilarity (in the same sense that a distance is a measure of dissimilarity when we cluster data).

In order to evaluate a classifier, we should focus on measuring the performance of the classifier using a suitable measure for the problem at hand (accuracy, precision, recall, F-score, etc.), all of these could be viewed as measures of similarity between two sets of labels, albeit focusing on different aspects of the problem (dissimilarities in case we use error rates). Now the true error rate or an indicator for dissimilarity is given by

$$R^*(T) = 1 - \frac{\text{correctly classified data objects in confusion matrix}}{\text{total number of data objects in confusion matrix}},$$

$$R^*(T) = 1 - \frac{n_{11} + n_{22} + n_{33} + \ldots + n_{\text{cc}}}{N},$$

$$R^*(T) = 1 - \frac{\sum_{i=1}^{\text{C}} n_{ii}}{N}.$$

The presence of noise in the training data and/or the lack of representative samples might lead to (lucky) correct predictions. However that could not be explained by the input data. The so-called bias-variance tradeoff is also a related issue.

If we have C classes, a completely random assignment would lead to 100% classifier accuracy. Typically, we can do even better if we always assign any input data to the most common class in our classifier, (i.e., $> = 50\%$ for binary classification problems, even $> 99\%$ in anomaly detection settings). Hence, the need for cost-sensitive measures and alternative evaluation metrics.

The remainder of this article is organized as follows.

In Section 2, a three steps strategy is introduced, which is used to search for the optimum classification tree. Section 3 considers the alternative algorithms to construct decision trees. In Section 4, the results for classification trees using the proposed strategy and some other strategies described in Section 3 will be presented. Section 5 discusses the experimental results while the last Section 6 presents the conclusion.

2. Three Steps Strategy (3SS) to Search for Optimum Classification Trees

Initially, an example is considered to explain briefly the proposed strategy.

2.1. *An Example*

Consider the Wine dataset along with sufficient characteristics regarding the predictors (X_i; $i = 1, 2, \ldots, 13$) and the response variable (integer-type values are assumed against each class with class labels 1, 2, and 3) that is given in the table. The distribution of each class label in the Wine dataset is as under:

Class label	1	2	3
Frequency	59	71	48

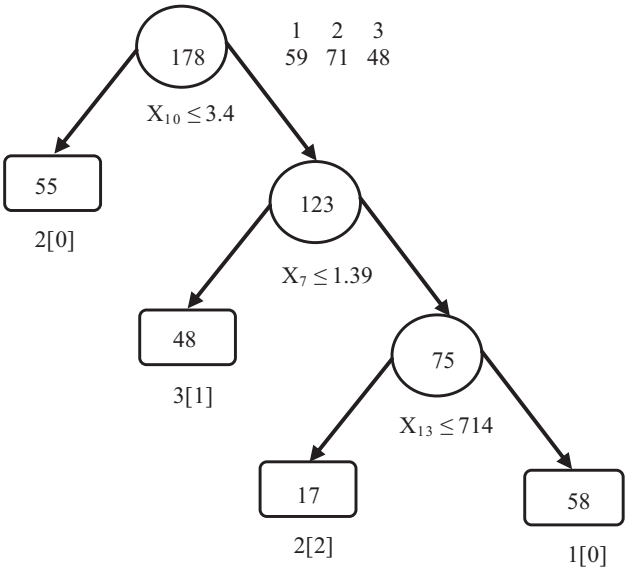

Figure 1. Classification tree of the Wine data using 3SS.

A classification tree is constructed using the bootstrap sample and then the out of bag (OOB) data are applied as testing set to observe the performance of the constructed tree and to obtain the true misclassification rate. The resulting classification tree is shown by Fig. 1. The values beneath the squares are the class labels while the values in the square brackets are the number of misclassified data objects. Each terminal node contains a sequence of splitting variables as well as splitting values.

These sequences start from the top node and goes through the end node. If only a single tree has to be constructed, two matrices are required: one for the variable-wise sequences and another for the corresponding splitting value-wise sequences. Our main focus is to keep the record of variable-wise sequences in one matrix and the record of corresponding value-wise sequences in another matrix. Each matrix is consisting of r-rows and c-columns (where r is the possible number of terminal nodes and c is the maximum depth of a tree).

On this way, by employing B bootstrap samples to generate classification trees, there are as many such tree structures (both variable-wise and their corresponding splitting value-wise) as there are the employed bootstrap samples.

From Fig. 1, one can depict both kinds of the tree structures simultaneously. For example, the tree structure given below is formulated by using a variable-wise sequence of each terminal node of the above tree starting from the left to right as

Node 1: X_{10}

Node 2: $X_{10} \rightarrow X_7$

Node 3: $X_{10} \rightarrow X_7 \rightarrow X_{13}$

Node 4: $X_{10} \rightarrow X_7 \rightarrow X_{13}$

Similarly, the tree structure using the corresponding splitting value-wise sequence of each terminal node of the tree starting from left to right is given as under:

Node 1: 3.40

Node 2: $3.40 \rightarrow 1.39$

Node 3: $3.40 \rightarrow 1.39 \rightarrow 714$

Node 4: 3.40 → 1.39 → 714

Note that the splitting at any node depends upon best selected variable and its value: A value that provides maximum measure of goodness of fit as compared to other variables and their values.

2.2. *Three Steps Strategy (3SS)*

Now, the proposed strategy will be described in the light of above example. The strategy is divided into three steps, which are explained below:

Step 1. B number of bootstrap samples is taken and classification trees $T_1, T_2, \ldots, T_B$ are constructed against each bootstrap sample. Each tree T_b; $(1 \leq b \leq B)$ is the joint structure of n_b terminal nodes.

Let $Q_1^{\text{var}}, Q_2^{\text{var}}, \cdots, Q_B^{\text{var}}$ be the variable-wise sequences of constructed tree structures. Similarly, $Q_1^{\text{val}}, Q_2^{\text{val}}, \cdots, Q_B^{\text{val}}$ are the corresponding value-wise sequences of the same tree structures. The sequences Q_b^{var} and Q_b^{val} of each tree T_b are stored in two separate matrices. Each matrix consists of r_b-rows and c_b-columns. Therefore, overall we have ($B \times 2$) number of such matrices for both types of sequences Q_b^{var} and Q_b^{val}. The search task is executed to acquire the most repeated variable-wise tree structure from the set of B matrices (i.e., a matrix having maximum frequency $f_{m(\text{var})}$).

Step 2. In this step, only those constructed trees are conceived that have the same variable-wise structure, that is, a set of matrices of sequences Q_b^{var} having maximum frequency $f_{m(\text{var})}$ obtained in the step 1. Now, the search is performed to obtain the most repeated value-wise tree structure Q_b^{val} having frequency $f_{m(\text{val})}$.

Step 3. In the last step, only those constructed trees are permitted having the same value-wise structure but they have different training error rates due to different training sets from the bootstrap samples. The search operation determines the one, which incorporates minimum training error rate out of all having the similar value-wise structure. On this way, final classification tree is incurred, which is the result of the three steps strategy and the OOB data are applied to get the generalized error rate.

The proposed algorithm explained above in three steps can be presented with the help of a flowchart shown by Fig. 2. The following frequency distribution has been formed to understand the mechanism of the proposed strategy. Let Q and q be the possible number of distinct variable-wise and value-wise sequences, respectively, then their distributions are

Distribution 1		Distribution 2	
Variable-wise sequence	F_{variable}	Value-wise sequence	F_{value}
S1	2	s1	1
S2	5	s2	3
S3	30 *fm*(variable)	s3	10 *fm*(value)
.	.	.	.
.	.	.	.
.	.	.	.
SQ	6	sq	2
Sum	$B = 200$ (say)	sum	30 (f_m)

Final Tree Selection: In the above table, Distribution 1 gives frequency distribution of $\underline{B} = 200$ (say) variable-wise tree structures with the label variable-wise sequence and

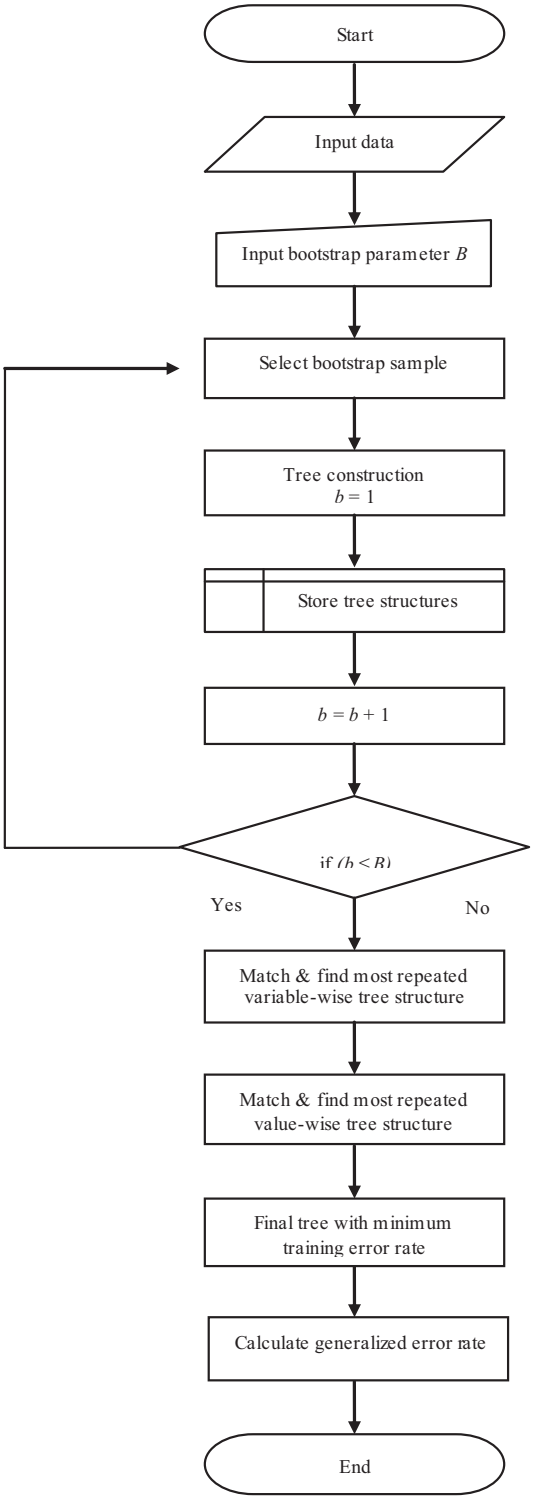

Figure 2. Flowchart of three steps strategy to search for the optimum classification tree.

Algorithm: Classification trees using the tree

Input: Let D be the original training data, B be the number of base classifiers, and K be the number of folds in cross validation.

- **for** *(b* in 1 : *B) {*
- Create training data D_b from D.
- **for** *(k* in 1 : *K) {*
- Construct a base classifier $C_{b(k)}$ from $D_{b(k)}$.
- *}*
- Obtain un-pruned tree & cross-validated best size (bs).
- *}*
- Obtain un-pruned & pruned trees with most frequent value of bs.

Output: Calculate generalized error rate

Figure 3. Classification trees algorithm using the "tree" package.

corresponding frequency (i.e., F_{variable}), while Distribution 2 gives frequency distribution of $f_{m(\text{variable})} = 30$ (the most repeated tree structure or sequence) and searches for a value-wise sequence appearing maximum number of times. We will consider $f_{m(\text{value})} = 10$ trees in Distribution 2. Finally, we will select the one having minimum number of misclassified units.

3. Alternative Algorithms to Construct Decision Trees

We have discussed above the main theme of the proposed strategy to acquire the optimum classification tree. There are few other tree structured algorithms (e.g., tree and rpart packages) with some differences. A brief description of algorithms used in this study for comparison purposes is given below; a detailed version of each algorithm can be found in the given references.

3.1. *Tree*

The "tree" package was first implemented as S-plus routines of Clark and Pregibon (1992) by the S-plus developers. Later on, it was implemented in **R** software by Ripley (1999) and had many versions with some kind of modifications, afterward by Ripley (2009). It may generate classification as well as regression trees. The tree software may consider two types of splitting criteria, that is, the deviance and the Gini function. The deviance is applicable for the construction of regression trees, while the Gini function is used for generating classification trees. The cost complexity pruning scheme of Breiman et al., (1984) is used for the pruning purpose. The tree package follows the strategy implemented in CART by Salford (1995). Figure 3 explains the un-pruned and pruned classification tree algorithm using the "tree" package.

3.2. *Rpart*

The "rpart" package, which is the abbreviation of recursive partitioning and regression trees, is used to construct decision trees. It was first implemented in **R** software in the

Algorithm: Classification trees using the tree

Input: Let D be the original training data, B be the number of base classifiers, and K be the number of folds in cross validation.

- **for** $(b$ in $1 : B)$ {
- Create training data D_b from D.
- **for** $(k$ in $1 : K)$ {
- Construct a base classifier $C_{b(k)}$ from $D_{b(k)}$.
- }
- Obtain un-pruned tree & cross-validated best size (bs).
- }
- Obtain un-pruned & pruned trees with most frequent value of bs.

Output: Calculate generalized error rate

light of the technical report of Therneau and Atkinson (2009) using the source code of Clark and Pregibon (1992) and many updated versions afterward. The latest version available to date is of Therneau and Atkinson (2009). It uses the Gini (Breiman et al., 1984) and the Entropy (Quinlan, 1986) function for splitting intermediate nodes into binary sub-nodes. By default, it performs cross-validation and provides the un-pruned trees on the basis of cross-validated estimates. The cost-complexity pruning scheme of Breiman et al. (1984) is used for the pruning purpose. The complexity parameter (cp) (which is the size of tree) plays a vital role in the selection of resultant tree. The value of cp penalizes models that are too complex. A small penalty leads to generate more splits. Figure 4 demonstrates the building decision trees algorithm using the "rpart" package.

Algorithm: Classification trees using the rpart

Input: Let D be the original training data, B be the number of base classifiers, and K be the number of folds in cross validation.

- **for** $(b$ in $1 : B)$ {
- Create training data D_b from D.
- **for** $(k$ in $1 : K)$ {
- Construct a base classifier $C_{b(k)}$ from $D_{b(k)}$.
- }
- Obtain un-pruned tree & complexity parameter (cp).
- }
- Obtain un-pruned & pruned trees with most frequent value of cp.

Output: Calculate generalized error rate

Figure 4. Classification trees algorithm using the "rpart" package.

Algorithm: Classification trees using the rpart

Input: Let D be the original training data, B be the number of base classifiers, and K be the number of folds in cross validation.

- **for** *(b* in 1 : *B)* {
- Create training data D_b from D.
- **for** *(k* in 1 : *K)* {
- Construct a base classifier $C_{b(k)}$ from $D_{b(k)}$.
- }
- Obtain un-pruned tree & complexity parameter (cp).
- }
- Obtain un-pruned & pruned trees with most frequent value of cp.

Output: Calculate generalized error rate

4. Comparison of Various Classification Tree Algorithms

Experiments are conducted to investigate and analyze the performance of the proposed three steps strategy to construct classification trees. First, the classification trees along with their main features obtained for variant evaluation functions including the Gini function (Breiman et al., 1984), the Entropy function (Quinlan, 1986), the Exponent-based function (Azam et al., 2009), and the Twoing rule (Breiman et al., 1984) are investigated and analyzed. Then the resultant trees obtained by proposed strategy are compared to other decision tree algorithms including "rpart" using the Gini as a node splitting function. Finally, the resultant trees of the proposed strategy are compared to some other decision tree algorithms, for example, "tree" and "rpart" using the Entropy as a node splitting function. The decision tree softwares including "tree," and "rpart" mentioned above are all available in **R**.

4.1. *Real-life Datasets*

There are number of methods to evaluate the performance of a method. Whether a particular method seems to accomplish better than other methods can be evaluated by using real-life data. In practice, real-life datasets are one of the ways to see the performance. Mathematical and Computer-based simulations are also helpful to observe the performance. Twelve benchmark datasets from the UCI (University of California Irvine) Machine Learning Repository in order to compare the performance of different node splitting methods are included. Table 2 summarizes the datasets those have been used in this study, which are freely available from the following URL: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

Some of the datasets have binary class labels while the other represents multi-class domain. Some of the problems contain a mixture of ordinal and continuous predictors while some are purely continuous, and few are purely ordinal. Only those datasets from the repository are included that have no missing values and contain a categorical response variable.

4.2. *Experimental Setup of Various Tree Algorithms*

The classification trees by "tree" package are obtained by using 10-fold cross-validation. Some information regarding the controlled parameters is to be provided. It includes the number of data objects in each training set, minimum size of data objects to be allowed in

Table 2
Dataset used in our experiments

Dataset	Sample size	Predictors	Continuous	Ordinal	Classes
Lymphography	148	18	0	18	4
Iris	150	4	4	0	3
Wine	178	13	13	0	3
Glass	214	9	9	0	6
Haberman	306	3	3	0	2
Bupa	345	6	6	0	2
Housing	506	13	12	1	9
Breast cancer	569	30	30	0	2
Bal. scale	625	4	0	4	3
Pima	768	8	8	0	2
Vehicle	846	18	18	0	4
Yeast	1,484	8	8	0	10

a node, minimum deviance needed to declare any node a terminal node, etc. An optimum size of the tree is achieved at each run on the basis of the point where the error rate bottoms out.

Finally, one such value of optimum size is searched, which appears maximum number of times in the list of B such values.

A similar procedure as described above is initiated for the "rpart" software to construct a decision tree. It provides the complexity parameter (cp) which is used to trim off the least important splits at each run. Note that cp, at this initial fit has to be small enough so that the minimum of the cross-validated error is attained. Furthermore, the most frequent value of cp is chosen from the list of B such values to generate un-pruned and pruned trees. The basic experiment is repeated for 200 times to get more reliable results while the generalized error rates are obtained for each method. To avoid over-fitting problem or over optimistic misclassification errors, we have utilized the OOB data objects left over while drawing each bootstrap sample. Tables 3–9 provides misclassification errors based on OOB units.

4.3. *Experimental Results of Various Tree Algorithms*

Tables 3 and 4 summarize the experimental results obtained from constructing binary classification trees using the proposed three steps strategy on datasets mentioned above. Table 3 shows generalized error rates and size of the constructed classification trees before pruning, while Table 4 presents the results after pruning. The accompanying tables include the results obtained by using four evaluation functions described earlier. Tables 6 and 7 summarize the results of building classification trees using two different approaches including tree and rpart. Entropy function is used as the splitting function in both tree building algorithms for classification and prediction purposes. Generalized error rates are tabulated both for un-pruned and pruned classification trees.

Similarly, Tables 8 and 9 summarize the results of constructing classification trees before and after pruning when the performance of 3SS, tree, and rpart is compared. This comparison is carried out when Gini function is used as the node splitting criterion for all three classification tree algorithms.

Table 3
Generalized error rates (GER) in % age and tree size (before pruning)

	Gini		Twoing		Entropy		Exponent	
Dataset	GER	Tree size	GER	Tree size	GER	Tree size	GER	Tree size
Wine	5.19	4	5.71	4	4.48	5	3.17	5
Breast cancer	6.76	10	4.07	11	6.00	10	7.08	12
Iris	5.56	4	3.70	4	3.92	5	3.33	5
Housing	23.28	14	16.67	19	25.65	14	14.74	19
Lymphography	20.37	5	26.67	4	22.41	4	30.36	4
Bal. scale	20.90	26	20.43	26	21.91	26	21.19	29
Vehicle	20.27	59	21.93	57	19.48	58	21.32	62
Pima	16.55	51	19.38	51	19.24	49	15.41	56
Bupa	29.66	28	25.62	31	29.32	29	22.83	29
Haberman	23.53	22	27.03	29	23.15	31	18.97	31
Glass	22.37	20	31.65	15	18.07	17	32.43	16
Yeast	43.56	18	49.54	7	44.22	10	40.48	35
Average	19.83	21.75	21.03	21.50	19.82	21.50	19.27	25.25

5. Discussion on Various Classification Tree Algorithms

All four evaluation functions used for splitting purpose demonstrate very similar behavior (see Tables 3 and 4) both in terms of error rates and the tree complexity both before and after pruning. It is generally observed that none of the evaluation criteria always perform better for

Table 4
Generalized error rates (GER) in % age and tree size (after pruning)

	Gini		Twoing		Entropy		Exponent	
Dataset	GER	Tree size	GER	Tree size	GER	Tree size	GER	Tree size
Wine	5.19	4	5.71	4	4.48	5	3.17	5
Breast cancer	6.76	5	4.07	10	6.00	7	7.55	8
Iris	5.56	4	3.70	4	3.92	4	3.33	4
Housing	34.92	8	39.44	4	25.65	14	22.63	13
Lymphography	20.37	5	26.67	4	22.41	4	30.36	4
Bal. scale	28.69	11	28.09	8	33.47	7	29.24	10
Vehicle	27.70	27	31.56	25	29.22	25	32.92	12
Pima	19.42	15	23.53	6	27.84	3	21.92	3
Bupa	35.59	7	33.06	10	36.84	4	31.50	4
Haberman	25.21	6	33.33	5	29.63	4	25.86	6
Glass	27.63	9	37.97	9	28.92	9	35.14	11
Yeast	43.56	13	49.73	6	44.40	8	42.31	13
Average	23.38	9.50	26.41	7.92	24.40	7.83	23.83	7.75

Table 5
Generalized error rates (GER) in % age

Dataset	Random forests GER	Average GER (Above 4 methods)
Wine	1.69	4.64
Iris	4.67	4.13
Breast cancer	3.51	6.10
Housing	3.75	30.66
Vehicle	24.35	30.35
Lymphography	14.86	24.95
Bal. scale	16.32	29.87
Pima	23.57	23.18
Haberman	27.45	28.51
Bupa	24.06	34.25
Glass	20.09	32.42
Yeast	36.73	45.00
Average	16.75	24.50

all datasets when these functions are applied to the proposed three steps strategy. The experimental results also support the theory that most of the decision tree analysis present connatural response regarding the selection of node splitting point, which returns almost the same structure in case of having small number of classes (Breiman et al., 1984). In the present study, the Gini function performed slightly better than others, though results are not significantly (Friedman rank sum test (Demsar, 2006) is used to observe the significance.) varying

Table 6
Generalized error rates (GER) in % age and tree size using Entropy function (before pruning)

	3SS		rpart	
Dataset	GER	Tree size	GER	Tree size
Wine	4.48	5	1.59	8
Breast cancer	6.00	10	7.11	7
Iris	3.92	5	5.26	5
Housing	25.65	14	17.04	20
Lymphography	22.41	4	25.93	15
Bal. scale	21.91	26	25.82	12
Vehicle	19.48	58	31.37	19
Pima	19.24	49	26.09	14
Bupa	29.32	29	35.87	35
Haberman	23.15	31	32.45	27
Glass	18.07	17	28.57	25
Yeast	44.22	10	41.65	8
Average	19.82	21.50	23.23	16.25

Table 7
Generalized error rates (GER) in % age and tree size using Entropy function (after pruning)

	3SS		rpart	
Dataset	GER	Tree size	GER	Tree size
Wine	4.48	5	11.11	8
Breast cancer	6.00	7	7.11	3
Iris	3.92	4	7.02	3
Housing	25.65	14	27.27	13
Lymphography	22.41	4	25.93	5
Bal. scale	33.47	7	30.98	7
Vehicle	29.22	25	39.54	5
Pima	27.84	3	26.09	14
Bupa	36.84	4	42.75	6
Haberman	29.63	4	27.19	4
Glass	28.92	9	32.47	6
Yeast	44.40	8	44.40	6
Average	24.40	7.83	26.82	6.67

(see Tables 3 and 4). For the iris and wine dataset, it is observed that that there is no change in the results in both un-pruned and pruned trees in all the four nodes splitting functions. The results with and without pruning are quite similar for some datasets (e.g., wine, yeast) in terms of GER because of top-down sequential pruning scheme (Azam et al., 2009). If

Table 8
Generalized error rates (GER) in % age and tree size using Gini function (before pruning)

	tree		3SS		rpart	
Dataset	GER	Tree size	GER	Tree size	GER	Tree size
Wine	28.12	18	5.19	4	4.22	8
Breast cancer	4.14	15	6.76	10	6.54	7
Iris	5.08	6	5.56	4	5.66	4
Housing	21.54	18	23.28	14	14.06	20
Lymphography	27.27	27	20.37	5	33.33	17
Bal. scale	22.97	83	20.90	26	20.53	11
Vehicle	55.73	175	20.27	59	31.25	10
Pima	34.64	77	16.55	51	24.74	16
Bupa	35.65	41	29.66	28	32.81	34
Haberman	36.28	40	23.53	22	23.52	8
Glass	40.25	22	22.37	20	34.93	25
Yeast	43.53	10	43.56	18	46.47	9
Average	29.60	44.33	19.83	21.75	23.17	14.08

Table 9
Generalized error rates (GER) in % age and tree size using Gini function (after pruning)

	tree		3SS		rpart	
Dataset	GER	Tree size	GER	Tree size	GER	Tree size
Wine	29.69	17	5.19	4	7.04	4
Breast cancer	5.52	3	6.76	5	10.28	3
Iris	5.08	3	5.56	4	5.66	3
Housing	21.54	18	34.92	8	19.79	9
Lymphography	29.09	16	20.37	5	33.33	3
Bal. scale	29.73	10	28.69	11	25.89	6
Vehicle	65.57	27	27.70	27	40.13	5
Pima	31.78	7	19.42	15	24.74	16
Bupa	35.65	3	35.59	7	37.50	3
Haberman	33.63	13	25.21	6	23.52	8
Glass	58.44	2	27.63	9	38.55	5
Yeast	43.53	8	43.56	13	47.56	7
Average	32.44	10.58	23.38	9.50	26.17	6.00

there will be no decrease in impurity in descendent nodes, then it will prune it backward and the number of descendent nodes will be decreased without affecting the misclassification rates.

While using the other datasets, sometimes even similar results are achieved for various evaluation functions, but it is not necessary that the whole of the tree structure is similar both in terms of chosen predictors and their splitting values.

It is observed that the proposed "3SS" outperforms for many (9 out of 12) datasets (before and after pruning) and it provides minimum generalized error rates when the results are compared with other tree algorithms, for example, "rpart" using Gini as a node splitting function. The proposed strategy generated trees that are larger in size as compared to other tree building procedure (see Tables 6 and 7). The proposed 3SS also performed better than rpart after pruning but the results are not significant.

The proposed three steps strategy is also compared with two algorithms, that is, "tree" and "rpart" using Gini function as a node splitting function. The proposed strategy again showed better performance (5 out of 12 datasets) in both cases of un-pruned and pruned trees. On average, the proposed strategy showed lower GER as compared to tree and rpart classification trees. The Gini function is observed to be more suitable than the Entropy function as a node splitting criterion when the classification trees are constructed using the proposed strategy (see Tables 8 and 9). An ensemble method "Random Forests" (Breiman, 2001) has also been implemented (see Table 5). It shows better predictive accuracy than just one classification tree constructed by any classification method. This is perhaps similar finding about ensemble classifier versus single classification tree as quoted by Breiman (1996) "what one loses is the simple and interpretable tree structure and what one gains is the increased accuracy." So in the proposed strategy we are not losing tree structure even we tried to get more stable classifier.

6. Comments and Conclusion

The experimental results tabulated in Section 4.3 and some discussion presented in Section 5, it can be concluded that the proposed three steps strategy performs equally well for all node splitting functions used in this study. The proposed strategy shows better performance with respect to other tree building algorithms, for example, "tree" and "rpart." More specifically, it performs well when the Gini function is used as a node splitting function in all these algorithms. The proposed three steps strategy seems to be more suitable and appealing than other tree building procedures. Because, main theme of the proposed strategy is based on the idea of taking majority votes (as discussed in the bagging strategy) in terms of whole tree structure (both in the selection of predictor-wise and the splitting values-wise sequences). On this way, it provides the tree structure that appears maximum number of times in the set of B such structures. The strategy needs larger value of B when the size of the dataset is large enough or too many terminal nodes are involved, so that it can search for the optimum tree structure.

We have applied the three steps strategy to the un-pruned trees first and then obtained the pruned trees. As a result, very few trees are selected in the steps 1 and 2, respectively, because of the tree complexity. This issue can be overcome by increasing B or by initiating the three steps strategy to the pruned trees first instead of un-pruned trees. It will strengthen the classification as well as predictive power of the resulting decision tree by taking more votes for the most repeated tree sequences.

Acknowledgment

The authors are deeply thankful to the two reviewers and the editor for their valuable suggestions to improve the quality of this manuscript.

References

Azam, M., Zaman, Q. , Pfeiffer, K. P. (2007). Improved classification trees with two or more classes. In: *Proceedings of The 9th Islamic Countries Conference on Statistical Sciences 2007*, pp. 608–626.

Azam, M., Zaman, Q., Salahuddin, Pfeiffer, K. P. (2009). Evaluation criteria for the construction of binary classification trees with two or more classes. *Pakistan Journal of Statistics* 25(3):241–249.

Berzal, F., Cubero, J. C., Cuenca, F., Martin-Bautista, M. J. (2003). On the quest of easy to understand splitting rules. *Data and Knowledge Engineering* 44:31–44.

Breiman, L. (1996). Bagging predictors. *Machine Learning* 24:123–140.

Breiman, L. (2001). Random forests. *Machine Learning* 45:5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group.

Bremner, A. P., Taplin, R. H. (2002). Modified classification and regression tree splitting criteria for data with interactions. *Australian New Zealand Journal of Statistics* 44(2):169–176.

Chang, Y. (2012). Multiple-step classification trees. *Communications in Statistics: Simulation and Computations* 41(9):1728–1744.

Clark, L. A., Pregibon, D. (1992). Tree-based models. In: Chambers, J. M., Hastie, T. J., eds. *Statistical Models in S*. Pacific Grove, CA: Wadsworth and Brooks/ Cole.

Demsar, J. (2006). Statistical comparison of classifiers over multiple datasets. *Journal of Machine Learning Research* 7:1–30.

Doyle, P. (1973). The use of automatic interaction detector and similar search procedures. *Operational Research Quarterly* 24:465–467.

Doyle, P., Fenwick, I. (1975). The pitfalls of AID analysis. *Journal of Marketing Research* 12:408–413.

Fayyad, U. M., Irani, K. B. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning* 8:87–102.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 7(2):179–188.

Hornik, K., Buchta, C., Hothorn, T., Karatzouglo, A., Meyer, D., Zeileis, A. (2009). *R/Weka Interface*. *CRAN Repository*. WU Wien, Austria: Department of Statistics and Mathematics.

Hsiao, W. C., Shih, Y. S. (2007). Splitting variable selection for multivariate regression trees. *Statistics &Probability Letters* 77:265–271.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics* 29:119–127.

Loh, W. Y. (2008). Classification and regression tree methods. In: *Encyclopedia of Statistics in Quality and Reliability,* pp. 315–323.

Loh, W. Y., Shih, Y. S. (1997). Split selection methods for classification trees. *Statistica Sinica* 7:815–840.

Machova, K., Barcak, F., Bednar, P. (2006). A bagging method using decision trees in the role of base classifiers. *Acta Polytechnica Hungarica* 3(2):121–132.

Martinez, W. L., Martinez, A. R. (2002). *Computational Statistics Handbook with MATLAB*. Boca Raton, FL: Chapman and Hall/CRC.

Morgan, J. N., Messenger, R. C. (1973). *THAID: A Sequential Search Program for the Analysis of Nominal Scale Dependent Variables*. (Technical Report). Institute for Social Research. Ann Arbor, MI: University of Michigan.

Morgan, J. N., Sonquist, J. A. (1963). Problems in the analysis of survey data and a proposal. *Journal of American Statistical Association* 58:415–434.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning* 1:81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Machine Studies* 27:221–234.

Quinlan, J. R. (1993). *C4.5 Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

Rao, C. R., Wegman, E. J., Solka, J. L. (2005). *Handbook of Statistics, Vol. 24: Data Mining and Data Visualization*. North Holland: Elsevier B.V.

Ripley, B. D. (1999). *Package 'tree'. CRAN Repository*. WU Wien, Austria: Department of Statistics and Mathematics.

Ripley, B. D. (2009). *Package 'tree'. CRAN Repository*. WU Wien, Austria: Department of Statistics and Mathematics.

Salford, S. (1995).*CART*. Salford Systems Company, San Diego, CA.

Shih, Y. S. (2004). A note on split selection bias in classification trees. *Computational Statistics & Data Analysis* 45:457–466.

Strobl, C., Boulesteix, A. L., Augustin, T. (2007). Unbiased split selection for classification trees based on the Gini index. *Computational Statistics & Data Analysis* 52:483–501.

Taylor, P. C., Silverman, B. W. (1993). Block diagrams and splitting criteria for classification trees. *Statistics and Computing* 3:147–161.

Therneau, T. M., Atkinson, E. J. (1997). *An Introduction to Recursive Partitioning Using rpart Routine*. Technical Report 61. Mayo Clinic, Section of Statistics.

Therneau, T. M., Atkinson, E. J. (2009). *Package "rpart". CRAN Repository*. WU Wien, Austria: Department of Statistics and Mathematics.

Therneau, T. M., Atkinson, B. (2009). rpart: Recursive partitioning [Computer software]. R package version 4.1-9. R port by Brian Riple. Available from http://CRAN.R-project.org/package=rpart