Accounting for Multiple Comparisons in Decision Tree Pruning

David Jensen
Department of Computer Science
Box 34610 LGRC
University of Massachusetts
Amherst, MA 01003-4610

Abstract

Overfitting is a widely observed pathology of induction algorithms. For induction algorithms that build decision trees, pruning is a common approach to correct overfitting. Most common pruning techniques, do not account for one potentially important factor — multiple comparisons. Multiple comparisons occur whenever an induction algorithm examines several candidate models and selects the one that best accords with the data. Making multiple comparisons produces systematic overestimates of accuracy. This paper empirically examines the importance of accounting for multiple comparisons when evaluating models. Specifically, it examines the effectiveness of one particular pruning method that does account for multiple comparisons – Bonferroni pruning. Based on experiments with artificial and realistic datasets, Bonferroni pruning produces trees that are smaller and at least as accurate as trees pruned using several other common approaches.

Keywords: Decision-tree learning, Classification, Bonferroni pruning, Multiple comparisons, Statistical hypothesis testing.

jensen@cs.umass.edu
413-545-3613

Accounting for Multiple Comparisons in Decision Tree Pruning

Abstract Overfitting is a widely observed pathology of induction algorithms. For induction algorithms that build decision trees, pruning is a common approach to correct overfitting. Most common pruning techniques, do not account for one potentially important factor — multiple comparisons. Multiple comparisons occur whenever an induction algorithm examines several candidate models and selects the one that best accords with the data. Making multiple comparisons produces systematic overestimates of accuracy. This paper empirically examines the importance of accounting for multiple comparisons when evaluating models. Specifically, it examines the effectiveness of one particular pruning method that does account for multiple comparisons – Bonferroni pruning. Based on experiments with artificial and realistic datasets, Bonferroni pruning produces trees that are smaller and at least as accurate as trees pruned using several other common approaches.

1 Introduction

Overfitting is a widely observed pathology of induction algorithms. Overfitted models contain unnecessary structure that reflects nothing more than random variation in the data sample used to construct the model. Portions of these models are literally wrong, and can mislead users. Overfitted models are less efficient to store and use than their correctly-sized counterparts. Finally, overfitting can reduce the accuracy of induced models on new data [2, 14].

For induction algorithms that build decision trees [4, 13, 15], *pruning* is a common approach to correct overfitting. Pruning techniques take an induced tree, examine individual subtrees, and remove those subtrees deemed to be unnecessary. While pruning techniques can differ in several respects, they primarily differ in the criterion used to judge subtrees. Many criteria have been proposed, including statistical significance tests [13], corrected error estimates [15], and minimum description length calculations [12].

Most common pruning techniques, however, do not account for one potentially important factor — multiple comparisons. Multiple comparisons occur whenever an induction algorithm examines several candidate models and selects the one that best accords with the data. Any search process necessarily involves multiple comparisons, and nearly all common induction algorithms involve implicit or explicit search through a space of candidate models. In the case of decision trees, search involves examining many possible subtrees and selecting the best one. Pruning techniques need to account for the number of subtrees examined, because such multiple comparisons affect the apparent accuracy of models on training data [1].

This paper examines the importance of accounting for multiple comparisons. Specifically, it examines the effectiveness of one particular pruning method — *Bonferroni pruning.* Bonferroni pruning adjusts the results of a standard significance test to account for the number of subtrees examined at a particular node of a decision tree. Evidence that Bonferroni pruning leads to better models supports the hypothesis that multiple comparisons are an important cause of overfitting.

The next section briefly surveys several approaches to decision tree pruning and explains the basic rationale for Bonferroni pruning. Section 3 presents the results of two experiments that use artificial data to compare Bonferroni pruning to three other pruning techniques. Section 4 introduces a full tree-building algorithm that employs Bonferroni pruning — Tree-building with Bonferroni Adjustment (TBA) — and compares its performance to that of C4.5, a widely-used algorithm. The final section discusses the wider implications of multiple comparisons.

2 Bonferroni Pruning

Pruning techniques have been developed based on several different criteria. Criteria based on statistical significance [13] have generally been rejected based on their empirical performance. Error-based criteria, such as the approach used in C4.5 [15], estimate true error rates by deflating the accuracy of subtrees on the training data. Minimum Description Length (MDL) criteria [12] characterize both datasets and models by the number of bits needed to encode them. The best tree is the one with the smallest total "description length" for the data, that is, the smallest sum of model description and description of the exceptions to the model's predictions.

However, none of these approaches is intended to account for multiple comparisons. An occasional induction algorithm has taken account of its effects [8, 2, 9] and several researchers have called attention to multiple comparisons when *evaluating* induction algorithms [6, 7, 16]. However, nearly all commonly-used tree-building algorithms do not account for multiple comparisons.

Why should multiple comparisons affect decision tree pruning? Consider the simplest case of pruning — deciding between a small subtree and a leaf node. Suppose the subtree partitions the data based on the values of a binary variable (A) and predicts the class + for one subsample and - for the other subsample. Suppose the leaf node always predicts the most prevalent class +. One way of deciding whether to retain the tree is to produce a *score* for the tree, denoted x. The score could be the classification accuracy alone, or some more complex function.

For the tree to be retained, x must exceed some threshold — either the score for the leaf or some higher threshold. We will call this threshold the *critical value*, denoted x_c. One way of setting x_c is to employ statistical hypothesis testing; x_c can be based on the reference distribution for the evaluation function under the null hypothesis H_0 that A is unrelated to the class variable. There are also numerous *ad hoc* approaches.

The score x is an estimate of some true score x_t. Because x is based on a single sample, it will vary somewhat from the true value. Because of this, there is some probability that $x \geq x_c$ even though $x_t < x_c$. If we use statistical hypothesis testing, we can determine this probability under the null hypothesis H_0. The probability that x will exceed x_c under the null hypothesis — $p(x \geq x_c | H_0)$ — is commonly denoted α. The value of x_c is usually set such that α is fairly small (e.g., 10%).

This hypothesis testing approach, and nearly every *ad hoc* approach to setting thresholds, ignores an important aspect of how decision trees are built: the variable A was selected from a pool of potential variables. *Each* of these variables' scores had an opportunity to exceed x_c. If the potential variables are independently distributed, then $\alpha \neq 0.10$. Instead:

$$\alpha_t = 1 - (1 - \alpha_i)^n \tag{1}$$

where α_t is the probability that *at least one of* the n variables' scores will equal or exceed x_c, and α_i is the probability that a *particular* variable's score will equal or exceed x_c. If α_i is 0.10 and 30 variables are examined, then $\alpha_t = 0.96$. Rather than a 10% probability of incorrectly selecting the tree, such selection becomes nearly certain. In practice, induction algorithms evaluate dozens or even hundreds of variables when selecting a variable for a test node. As a result, accounting for these multiple comparisons becomes essential for an accurate pruning.

Equation 1 is one of a class of Bonferroni equations [11], commonly used to adjust statistical tests for multiple comparisons. The adjustment is necessary because nearly all conventional reference distributions are constructed under the assumption of a single comparison. Multiple comparisons render these reference distributions inaccurate [1].

The Bonferroni adjustment can be used to account for multiple comparisons. Specifically, Equation 1 can be solved for α_i, so that, for a given overall α_t and number of comparisons n, an appropriate critical value can be selected. The next two sections present both experiments with artificial and realistic data to evaluate the effect of this approach.

3 Experiments with Artificial Data

The two experiments reported in this section involve generating artificial data and applying a simple tree-building algorithm. The training set is created by randomly generating instances with 30 uniformly, independently, and identically distributed binary attributes. Next, a binary classification is created by applying a boolean function to one or more of the attributes. Finally, the class labels are systematically corrupted by complementing each label with a probability of 0.1. Thus, on average, 10% of the class labels are incorrect, producing a theoretical upper bound of 90% on classification accuracy.

The tree-building algorithm uses information gain [15] to choose the best attribute for a particular decision node. A leaf node is created when no partition can improve accuracy on the training set. The class label of the leaf node is determined by the majority class of the training instances present at that node. After constructing a tree, several bottom-up pruning techniques are applied. The techniques differ only by their pruning criterion. The pruning criterion is applied to all non-leaf subtrees of the original tree. If the criterion judges the subtree to be valid, the node is retained. If not, the subtree is converted to a leaf node and labeled with the majority class of the training instances present at that node. For any given tree, pruning continues until all subtrees are judged to be valid.

The pruning criteria are:

- FISHERS Significance testing using a conventional hypothesis test —

Fisher's exact test [11] — with $\alpha = 0.10$. This test does not account for multiple comparisons.

- ERROR-BASED The technique used in C4.5 [15].
- MDL Minimum length encoding, using Utgoff's MDL formulation [17].
- BONFERRONI Fisher's exact test with α adjusted to account for the number of comparisons using the Bonferroni adjustment. The significance level for each individual comparison is $\alpha_i = 1 - (1 - \alpha_t)^{1/n}$, where α_t is the overall significance level desired and n is the number of comparisons.

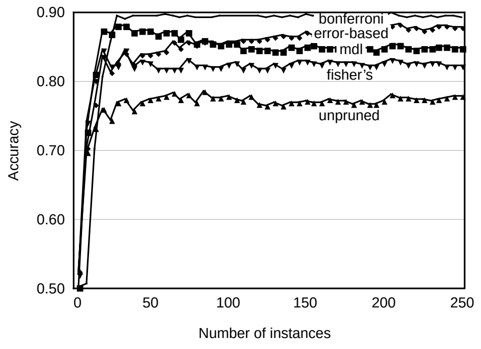
Finally, these methods are compared to trees that have not been pruned at all (UNPRUNED).

Each experiment involves varying the size of training sets from 5 to 200 by increments of 5 instances. For each value of training set size, a training set of that size is generated, an unpruned tree is induced, that tree is provided to each pruning technique, and the accuracy of the resulting pruned trees is evaluated on a test set of 1000 freshly generated instances. In addition, the complexity of each pruned tree is determined by counting the total number of nodes. For each training set size, 100 trials are conducted and the results for accuracy and complexity are each averaged.

3.1 Finding a Simple Tree

In the first experiment, the value of the classification variable is set to that of the first attribute ($C = A_1$). Recall that noise is introduced into the values of the classification variable, so the maximum theoretical accuracy is 90%. For this experiment, the theoretically correct tree has three nodes — a decision node and two leaf nodes.

The results are shown in Figure 1. In general, BONFERRONI produces the least complex and most accurate trees, but Figure 1 has some interesting details. For very small numbers of instances ($N < 20$), the average accuracy is quite low because no technique reliably produces the correct tree. BONFERRONI, however, is the most conservative, producing less complex and less accurate trees than the other pruning techniques. For slightly larger samples ($20 < N < 30$), the accuracy of trees from FISHERS, ERROR-BASED,



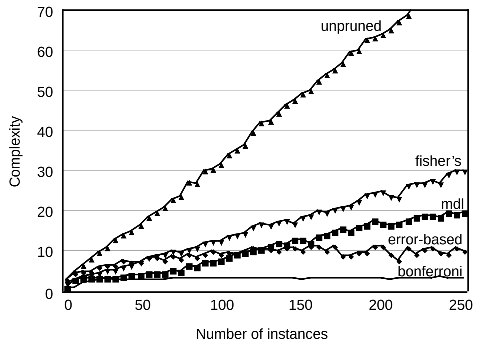


Figure 1: Accuracy and Complexity for Simple Relationship

MDL, and BONFERRONI are roughly the same, and the average accuracy of UNPRUNED trees is much lower.

For training sets with more than 40 instances, BONFERRONI achieves nearly maximum average accuracy, while the other techniques have lower average accuracy. This drop in accuracy for methods besides BONFERRONI is due entirely to overfitting. Recall that all pruning methods begin with the same trees. Because BONFERRONI produces pruned trees with nearly optimal accuracy, it is clear that the true structure is present in the original trees.

In addition to the overall variation in accuracy among the pruning methods, there is some interesting variation of average accuracy as training set size increases past $N = 40$. MDL initially produces highly accurate trees, but its average accuracy decreases as training set size grows. In contrast, ERROR-BASED initially has relatively low average accuracy, but it increases as training set size grows.

The graph of complexity indicates the reasons for these variations in average accuracy. For training sets of between 40 and 75 instances, MDL creates trees of roughly correct size, and thus their accuracy is fairly high. For $N > 75$, however, MDL adds spurious structure and average accuracy declines. Conversely, ERROR-BASED builds relatively large trees for $N < 100$. Then, however, its average complexity levels out. For $N > 100$, the training set contains enough instances that increasingly accurate labels can be assigned to the leaves of the (relatively fixed-size) tree. Thus the accuracy of ERROR-BASED trees continues to increase. With the exception of BONFERRONI and ERROR-BASED, all the techniques continue to add unnecessary structure, and this impairs accuracy.

Only BONFERRONI effectively avoids overfitting. The other techniques

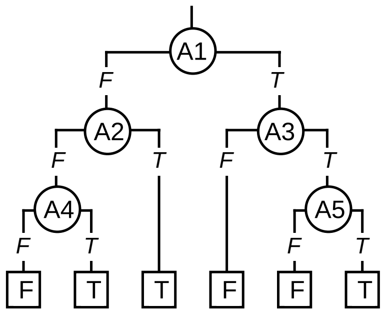


Figure 2: Tree Representing Correct Relationship

reduce the complexity of induced models somewhat, but they all produce models with excess structure. Complexity is strongly associated with training set size for UNPRUNED, FISHERS, and MDL. A related paper [3] explores this behavior for a wide variety of realistic datasets and several pruning approaches.

3.2 Finding a Complex Tree

In the second experiment, a much more complex tree was used to generate the class labels. The theoretically correct tree (shown in Figure 2) has eleven nodes — five decision nodes and six leaf nodes.

The results are shown in Figure 3. BONFERRONI produces the best trees if the number of instances in the training set exceeds 125. Prior to that threshold, there is a period where BONFERRONI creates trees whose accuracy is virtually indistinguishable, on average, from those of other pruning techniques ($75 < N < 125$), as well as a period where it produces trees that are less accurate ($N < 75$). Only BONFERRONI produces trees that converge to the correct tree size as the number of instances in the training set increases. The other techniques continue to add unwarranted complexity.

4 TBA: Tree-building with Bonferroni Adjustment

The experiments in section 3 provide some reason to believe that Bonferroni pruning is an effective approach to limiting the complexity of decision trees, but the experiments ignore a set of issues that arise in more realistic datasets. For example, datasets in many induction tasks contain attributes with multiple discrete values (e.g., eye-color) and real values (e.g., temperature), a mixture of attribute types, missing values, and non-tree-structured relationships. How does Bonferroni pruning perform in these cases and how does its performance compare to existing algorithms? These questions are addressed by evaluating the performance of a new algorithm — Tree-building with Bonferroni Adjustment (TBA) — on 13 datasets from the UCI repository.

4.1 How TBA Builds Trees

TBA's basic algorithm is similar to nearly all other algorithms for top-down induction of decision trees [13]. TBA differs from other algorithms in its evaluation function, how it selects partitions during tree construction, and how it selects between possible subtrees during tree pruning.

TBA uses the G statistic to evaluate contingency tables during tree construction and pruning.

$$G = 2 \sum_{cells} f_{ij} \ln \left(\frac{f_{ij}}{\hat{f}_{ij}} \right), \quad (2)$$

where f_{ij} is the number of occurrences, or frequency, in the cell i, j and $\hat{f}_{ij}$ is the expected value of that cell. In this case, the expected value is $f_{i\cdot} f_{\cdot j} / f_{\cdot\cdot}$, where $f_{i\cdot}$ is the total frequency in row i, $f_{\cdot j}$ is the total frequency in column j, and $f_{\cdot\cdot}$ is the total of all cells in the table.

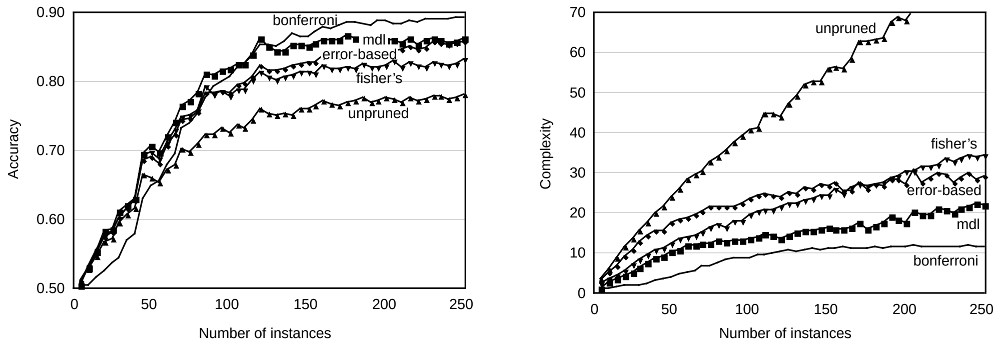


Figure 3: Accuracy and Complexity of Pruned Trees

The G statistic is used because it has a known reference distribution and because it is additive, allowing portions of contingency tables to be evaluated individually. This latter quality is important to how TBA builds trees.

During tree construction, attributes are selected using an approach suggested by Kass [9] and Kerber [10]. For each attribute, a contingency table is constructed with a row for each class value and a column for each of k attribute values — every possible value for discrete attributes or every unique interval for discretized continuous attributes. Then, the pair of columns in the table with the least significant difference is merged. The merging process is repeated until all columns are significantly different. For continuous attributes, only adjacent columns, corresponding to adjacent numeric intervals, can be merged. The result is a node that partitions the sample into j subsamples, where $1 \leq j \leq k$.

TBA compares attributes based on their probability values. These values are calculated by comparing the G value for the merged contingency table to an appropriate reference distribution, and applying a Bonferroni adjustment. For a table with r rows and c columns, the appropriate reference distribution is chi-square with $(r-1)(c-1)$ degrees of freedom. Following Kass, the Bonferroni exponent n is the number of possible ways of combining j initial categories into k final categories. The calculations differ by attribute type because only adjacent intervals can be merged in continuous attributes:

$$n_{continuous} = \begin{pmatrix} j-1 \\ k-1 \end{pmatrix}; \qquad n_{discrete} = \textstyle\sum_{i=0}^{k-1} (-1)^i \dfrac{(k-i)^j}{i!(k-i)!} \tag{3}$$

While these estimates of the exponents are approximate, at best, they provide a rough balance between the total number of possible tables (certainly an overestimate because the tables are highly correlated) and an exponent of 1 (certainly an underestimate).

TBA forms a decision node using the attribute with the lowest probability value, regardless of whether that value exceeds some threshold α. The algorithm uses the decision node to partition the sample into j subsamples based on the attribute's values, and repeats the attribute selection process for each subsample. Tree growth stops when no partition can improve accuracy on the training set.

After constructing a tree, TBA prunes the tree by examining the probability values calculated during tree construction. Recall that those probability values were adjusted to account for multiple comparisons *within* an attribute, but were not adjusted to account for multiple comparisons *among* the many attributes that could be used at an individual node. The latter adjustment is made at this stage, where the Bonferroni exponent n is the number of attributes considered at that node.

TBA examines each frontier node of the tree — decision nodes that have only leaf nodes as children. Frontier nodes where $p \leq \alpha_t$ are retained; frontier nodes where $p > \alpha_t$ are converted to leaf nodes and labeled with the majority class of the appropriate training subsample. The process continues until all frontier nodes are significant. Note that this process cannot eliminate non-frontier nodes for which $p > \alpha$. This could potentially "trap" insignificant nodes in the interior structure of the tree, but it ensures that potentially useful structure is not eliminated. For all of TBA's significance tests, $\alpha_t = 0.10$.

TBA handles missing values by assigning a default class to each decision node (the majority class of the training instances at that node). This class is assigned to any test instance that reaches a decision node for which it lacks a value for the appropriate attribute. While this approach is easy to implement, it lacks the sophistication of the approaches in other algorithms. For example, C4.5 sends instances that lack attribute values down *all* relevant branches of a node, weighted according to overall frequency in the training set, and then makes a prediction based on the weighted class labels of all branches. This difference appears to affect performance in at least one experiment reported below.

4.2 Results

Table 1 shows the results of applying TBA, C4.5, and a version of TBA without Bonferroni adjustment to thirteen datasets from the UCI repository. Each value in the table is the result of a ten-fold cross-validation where the folds are held constant across the runs of all three algorithms. Size indicates the total number of nodes in the pruned trees, and error indicates the frequency with which the pruned trees incorrectly classified instances in the test set.

Values in the table appear in italics when they are significantly different from the results of TBA. Values appear in bold when they are significantly different from the results for TBA and are inconsistent with the conclusion that TBA produces trees that are smaller and at least as accurate as those of C4.5. Significance was judged using a two-tailed, paired t-test on the ten-fold cross-validation results. The tests used a significance level of 0.10 and a

Table 1: Comparison of TBA with C4.5 and TBA-lesion

	TBA		C4.5		TBA-lesion	
Dataset	size	error	size	error	size	error
breast	7	0.280	12	0.266	*33*	0.300
cleveland	8	0.254	*46*	0.252	*23*	0.236
crx	22	0.167	*50*	0.154	*110*	0.196
glass	14	0.342	*51*	0.289	*32*	0.297
heart	11	0.244	*48*	0.248	*71*	0.274
hepatitis	7	0.161	*16*	0.212	*24*	0.175
hypothyroid	12	0.020	11	**0.008**	*57*	0.022
iris	4	0.047	*9*	0.067	*8*	0.060
lymphography	4	0.261	*26*	0.211	*20*	0.243
pima	20	0.267	*123*	0.285	*209*	0.327
votes	6	0.046	*14*	0.055	*27*	0.062
wine	14	0.068	**10**	0.085	*21*	0.057
wisconsin	14	0.060	20	0.052	*52*	0.082

Bonferroni correction to account for the 13 independent comparisons.

In nearly all cases, the results are consistent with the conjecture that TBA produces smaller trees than C4.5 without sacrificing accuracy. In nine of the thirteen datasets, TBA produced significantly smaller trees, in some cases dramatically smaller. Trees built by TBA are half the size, on average, of trees built by C4.5. In only one case was TBA's accuracy significantly lower than C4.5's. This dataset, `hypothyroid`, is the only set of the thirteen with substantial numbers of missing values. Nearly all of the over 3000 instances in the `hypothyroid` dataset contain one or more missing values. As noted earlier, TBA uses a simple, but potentially inferior method of handling missing values. In addition to comparisons with C4.5, the table presents results for TBA-lesion, a version of the TBA algorithm that does not apply the Bonferroni adjustment. In all cases, this algorithm produces trees that are significantly larger, but not significantly more accurate, than trees produced by TBA.

5 Implications

The experiments reported here provide strong empirical indications that accounting for multiple comparisons reduces overfitting. A related paper [1], provides strong theoretical reasons to account for multiple comparisons. Together, these results indicate that multiple comparisons are an important cause of overfitting, and that failing to account for its effects can lead to models with excessive structure.

In addition, the experiments indicate that accounting for multiple comparisons is an improvement over alternative methods such as error-based pruning, unadjusted significance tests, and pruning based on minimum description length. The experiments with artificial data show a disturbing increase in tree complexity with training set size, a relationship explored in a related paper [3]. Of the techniques evaluated here, only Bonferroni pruning avoids this pathology. In addition, the artificial experiments clearly show that overfitting can reduce accuracy.

However, substantial work remains to be done. Only the experiments with artificial data indicate that avoiding overfitting can improve accuracy. This effect did not emerge from experiments with realistic data. In addition, only the experiments with artificial data controlled for other potential causes of performance differences. While TBA's performance is impressive, several potential effects may contribute to that performance, including representation (TBA constructs multi-way partitions on continuous attributes), handling of missing values, and the criterion for attribute selection. Finally, more extensive comparisons of TBA to other pruning methods are needed.

References

[1] [names removed for anonymous review] (1997). Overfitting explained. Submitted to the Fourteenth International Conference on Machine Learning.

[2] Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees.* Pacific Grove, CA: Wadsworth and Brooks.

[3] Clark, P. and T. Niblett (1989). The CN2 induction algorithm. *Machine Learning* 3(4):261–283.

[4] Feelders, A. and W. Verkooijen (1995). Which method learns the most from data? Methodological issues in the analysis of comparative studies. *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics,* January 4-7, 1995, Ft. Lauderdale, Florida. 219–225.

[5] Gascuel, O. and G. Caraux (1992). Statistical significance in inductive learning. *ECAI92: Proceedings of the 10th European Conference on Artificial Intelligence.* B. Neumann (Ed.), Chichester: Wiley. 435–439.

[6] Gaines, B. (1989). An ounce of knowledge is worth a ton of data: Quantitative studies of the trade-off between expertise and data based on statistically well-founded empirical induction. *Proceedings of the Sixth International Workshop on Machine Learning.* 156–159.

[7] [citation removed for anonymous review]

[8] Kass, G.V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics* 29(2):199–127.

[9] Kerber, R. (1992). ChiMerge: Discretization of numeric attributes. Proceedings of the Tenth National Conference on Artificial Intelligence. AAAI Press/MIT Press. Menlo Park. pp.123-128.

[10] Kotz, S. and N.L. Johnson (Eds.) (1982-1989). *Encyclopedia of Statistical Sciences.* New York: Wiley.

[11] [names removed for anonymous review] (1997). The effects of training set size on decision tree complexity. Submitted to the Fourteenth International Conference on Machine Learning.

[12] Quinlan, J.R. and R. Rivest 1989. Inferring decision trees using the minimum description length principle. *Information and Computation* 80:227–248.

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

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

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

[16] Salzberg, S. (1995). On Comparing Classifiers: A Critique of Current Research and Methods. Technical Report JHU-95/06, Department of Computer Science, Johns Hopkins University, May 1995.

[17] Utgoff, P. (1995). Decision tree induction based on efficient tree restructuring. Technical Report 95-18. Department of Computer Science, University of Massachusetts, Amherst. March 17, 1995.