Lazy Learning for Improving Ranking of Decision Trees

Han Liang[1⋆], Yuhong Yan[2]

[1]Faculty of Computer Science, University of New Brunswick
Fredericton, NB, Canada E3B 5A3
han.liang@unb.ca
[2] National Research Council of Canada
Fredericton, NB, Canada E3B 5X9
yuhong.yan@nrc.gc.ca

Abstract. Decision tree-based probability estimation has received great attention because accurate probability estimation can possibly improve classification accuracy and probability-based ranking. In this paper, we aim to improve probability-based ranking under decision tree paradigms using AUC as the evaluation metric. We deploy a lazy probability estimator at each leaf to avoid uniform probability assignment. More importantly, the lazy probability estimator gives higher weights to the leaf samples closer to an unlabeled sample so that the probability estimation of this unlabeled sample is based on its similarities to those leaf samples. The motivation behind it is that ranking is a relative evaluation measurement among a set of samples, and it is reasonable to yield the probability for an unlabeled sample with reference to its extent of similarities to its neighbors. The proposed new decision tree model called *LazyTree*, outperforms C4.5, its recent improvement C4.4 and their *state-of-the-art* variants in AUC on a large suite of benchmark sample sets.

1 Introduction

A learning model is induced from a set of labeled samples represented by the vector of an attribute set $\mathbf{A} = \{A_1, A_2, \ldots, A_n\}$ and a class label C. Classic decision trees are typical decision boundary-based models. When computing class probabilities, decision trees use the observed frequencies at leaves for estimation. For instance, if a leaf contains 100 samples, 60 of which belong to the positive class and the others are in the negative class, then for any unlabeled sample that falls into the leaf, decision trees will assign the same positive probability of $\hat{p}(+|\mathbf{A_p} = \mathbf{a_p})$=60%, where $\mathbf{A_p}$ is the set of attributes from the leaf to the root. This incurs two problems: high bias (traditional tree inductive algorithm tries to make leaves homogeneous, therefore, the class probabilities are systematically shifted toward zero or one) and high variance (if the number of samples at a leaf is small, the class probabilities are unreliable) [11].

⋆ This work was done when the author was a visiting worker at Institute for Information Technology, National Research Council of Canada.

However, accurate probability estimation is important for many tasks. For classification problems, an unlabeled sample $\mathbf{e_t}$ is classified into the class c with the maximum posterior class probability $p(c|\mathbf{e_t})$ (or simply class probability), as shown below:

$$h(\mathbf{e_t}) = \arg\max_{c \in C} p(c|\mathbf{e_t}) = \arg\max_{c \in C} p(c, \mathbf{e_t})/p(\mathbf{e_t}), \tag{1}$$

where $h(\mathbf{e_t})$ in (1) is the classification function $h : \{\mathbf{e_t}\} \mapsto \{c\}$. Accurate probability estimation can possibly improve classification accuracy. For probability-based ranking problems, the rank of a set of samples $\{\mathbf{e_t}\}$ in class c is a sorted list based on the assigned class probabilities $\{p(c|\mathbf{e_t})\}$. Again, precise probability estimation can improve probability-based ranking. Various methods have been proposed to improve the probability estimation of decision trees (c.f. the next section).

The main objective of this paper is to improve the ranking performance of decision trees. The improvement comes from two aspects. Firstly, decision trees work better when the sample set is large. After several splits of attributes, the samples at the subspaces can be too few on which to base the probability. Therefore, although employing a traditional tree inductive process, we stop the splits once the samples are reduced to some extent and deploy probability estimators at leaves. The probability estimators assign distinct probabilities to different samples. Thus, the probability generated by such a tree is more accurate than assigning a uniform probability for the samples falling into the same leaf. Secondly, and more importantly, we observe that probability-based ranking is indeed a relative evaluation measurement where the correctness of ranking depends on the relative position of a sample among a set of other samples. In our paper, a lazy probability estimator that calculates the probability of an unlabeled sample based on its neighbors is designed for better ranking quality. The lazy probability estimator finds m closest neighbors (leaf samples) at a leaf for the unlabeled sample and calculates a weight for each neighbor using a newly proposed similarity score function. We generate the probability estimates for this unlabeled sample by normalizing all weights of the neighbors at the leaf in terms of their class values. The new model is called *LazyTree*. AUC [4] is used to evaluate our method. On a large suite of 36 standard sample sets, empirical results indicate that *LazyTree* performs substantially better than C4.5, C4.4 and their variants with other methods designed for optimal ranking, such as m-Branch, *Ling*&*Yan*'s algorithm and a voting strategy–*bagging*, in yielding accurate ranking.

The rest of the paper is organized as follows. Section 2 introduces some related works in learning decision trees with precise probability estimation and ranking. Section 3 presents our new model. We describe in detail the experiment setup and results in Section 4. We conclude in Section 5.

2 Related Work

A trained decision tree can be easily adapted to be a probability estimator by using the absolute class frequencies at each leaf of the tree. For instance, if a node has absolute frequencies $n_1, n_2, \ldots, n_c$ (obtained from the training set) the estimated probabilities for that node are calculated as $p_i = n_i / \sum n_i$. But this estimation is poor. Several meta methods have been proposed to improve the probability estimation of decision trees. The resulting trees are called *Probability Estimation Trees* (PETs).

Frequency smoothing of the leaf probability estimation deals with the pure nodes that contain samples from the same class. Instead of assigning a probability of 1 or 0, smoothing methods try to give a more modest estimation. Smoothing considers the cardinality of nodes so that nodes with small cardinality where all the samples are of class i will not have the same probability as nodes with much more cardinality in which all the samples also belong to class i. *Laplace* smoothing [8], where $p_i = (n_i + 1)/(\sum_{c \in C} n_i + |C|)$, is the most commonly used method. Ferri et al. [3] introduced another smoothing approach, call m-Branch. Given a leaf v^d and its branch from root v^1, p_i^{j-1} is the probability at level $j-1$ for class i. p_i^j is the probability smoothed from the root to level j as described in (2), where the parameter m is adjusted by the depth and cardinality of a node:

$$p_i^j = \frac{n_i^j + m * p_i^{j-1}}{\sum_{i \in C} n_i^j + m}. \tag{2}$$

Ling and Yan [6] proposed a novel algorithm. For a given unlabeled sample, instead of using a single leaf that the sample falls into, the class probability estimates are produced by averaging the probability estimates of all the leaves of a tree. The contribution of each leaf is decided by the number of unequal parent attribute values that the leaf has, compared to the unlabeled sample. Here, parent attributes of a leaf are defined as the attributes on the path from the root to this leaf. Therefore, Ling and Yan's algorithm needs to go through the whole tree in order to calculate different degrees of contributing to the final probabilities of the unlabeled sample for each leaf.

Unpruning trees are proposed by Provost and Domingos [8] based on the observation that pruning (or related techniques such as C4.5 collapsing) is not helpful for increasing probability estimation. By turning off pruning and collapsing in C4.5, decision trees can keep some branches that may not be useful for classification but are crucial for accurate probability estimation. The final version is called C4.4. However, there is a contradiction that turning off pruning could result in a larger tree so that within some leaves containing a small number of samples the probability estimation will be highly variant. In addition, large trees usually overfit the training set, therefore, the corresponding probability estimation could be still unreliable even using *Laplace* correction. Some researchers [1] [9] suggested using *bagging*, which takes an ensemble of sub-classifiers into account and generate probability estimates by averaging across the members of the

ensemble, to improve decision trees for better ranking. But the results produced by *bagging* are not easily interpretable.

Note that the above methods mentioned assign a uniformed probability to any unlabeled sample that is dispatched into the same leaf, moreover, they actually use only the path attributes $\mathbf{A_p}(l)$, which are the attributes from the root to a leaf l, for probability estimation:

$$p(c|\mathbf{e_t}) \approx p(c|\mathbf{A_p}(l)). \tag{3}$$

Using a probability density estimator at each leaf is another improvement to tackle the "uniform probability distribution" problem of decision trees. Kohavi [5] proposed an Naïve Bayes Tree (NBTree), in which a naïve Bayes is deployed at each leaf to produce probabilities. The intuition behind it is to take advantage of leaf attributes $\mathbf{A_l}(l)$ for probability estimation. Therefore, $p(c|\mathbf{e_t}) \approx p(c|\mathbf{A_p}(l), \mathbf{A_l}(l))$. Zhang and Su [12] presented the encoding of $p(\mathbf{A}, C)$ for NBTree:

$$p(\mathbf{A}, C) = \alpha p(C|\mathbf{A_p}(l))p(\mathbf{A_l}(l)|\mathbf{A_p}(l), C), \tag{4}$$

where α is a normalization factor. The term $p(C|\mathbf{A_p}(l))$ is the joint conditional distribution of path attributes and the term $p(\mathbf{A_l}(l)|\mathbf{A_p}(l), C)$ is the leaf attributes presented by naïve Bayes. According to the conditional independence assumption of naïve Bayes, $p(\mathbf{A_l}(l)|\mathbf{A_p}(l), C) = \prod_{i=1}^{n} p(A_{li}|\mathbf{A_p}(l), C)$.

Our method is inspired by the probability estimator method and smoothing techniques, but we want to make the estimation explicitly associated with more samples. We use the **Area Under the Curve of Receiver Operating Characteristics** (AUC) to evaluate ranking performances. A simple approach to calculate the AUC of a boolean model Γ is as follows [4]:

$$AUC = \frac{S_1 - n_1(n_1 + 1)/2}{n_1 n_0}, \tag{5}$$

where n_0 and n_1 are the numbers of negative and positive samples respectively, and $S_1 = \sum r_i$, where r_i is the rank of ith positive sample in the ranking list. For more classes, we can compute AUC separately for each pair of classes and then average all values [4].

3 Using Lazy Learner to Improve Tree-Based Ranking

3.1 The Lazy Learner

Our work aims to calibrate probability-based ranking of decision trees. Improvement comes from two aspects. Firstly, we want to trade-off between *bias* and *variance* of decision trees by deploying a ***probability estimator*** at each leaf. A decision tree partitions a sample space into regions. Using class frequencies to estimate conditional probability $p(c|\mathbf{e_t})$ suffers from *high bias* that the inductive algorithm makes leaves pure, therefore the class probabilities are systematically

shifted toward zero or one, and *high variance* that the observed frequencies are not statistically reliable when the number of samples at a leaf is small [11].

Although using a traditional tree heuristic search algorithm, we stop the splits if the samples are reduced to some extent and deploy probability estimators at leaves. *Probability Estimator* is defined as:

Definition 1. *Given a set of unlabeled samples E and a set of class labels $C = \{c_i\}$, a **Probability Estimator** is a set of functions $p_i : E \mapsto [0, 1]$, such that $\forall \mathbf{e_t} \in E$, $\sum p_i(\mathbf{e_t}) = 1$.*

The probability estimators give distinct probabilities to different samples. Thus, the probability estimation generated by such trees is more accurate than the uniform probability assignment for the samples falling into the same leaf.

Secondly, and more essentially, we observe that probability-based ranking is indeed a ***relative evaluation measurement*** where the correctness of ranking depends on the relative position of a sample among a set of other samples. For instance, for a binary-class problem, if assigned class probabilities of a positive sample $\mathbf{e}_+$ and a negative sample $\mathbf{e}_-$ satisfy $p(+|\mathbf{e}_+) > p(+|\mathbf{e}_-)$, it is a correct ranking. For multi-classes, the right ranking means $p_i(\mathbf{e} \in c_i) > p_i(\mathbf{e}' \notin c_i)$.

These inspire us to use a ***lazy probability estimator*** which calculates the probability of a sample based on its neighbors. The lazy probability estimator finds m closest neighbors at a leaf (here m means all the samples at this leaf) for an unlabeled sample and calculate a weight for each neighbor using a newly proposed similarity metric.

Assume that sample $\mathbf{e}$ can be represented by an attribute vector as $< a_1(\mathbf{e}), a_2(\mathbf{e}), ..., a_n(\mathbf{e}) >$, where $a_i(\mathbf{e})$ denotes the value of ith attribute. The distance between two samples $\mathbf{e_1}$ and $\mathbf{e_2}$ is calculated in (6):

$$d(\mathbf{e_1}, \mathbf{e_2}) = \sqrt{\sum_{i=1}^{n} \delta(a_i(\mathbf{e_1}), a_i(\mathbf{e_2}))}, \tag{6}$$

$\delta(a_i(\mathbf{e_1}), a_i(\mathbf{e_2}))$ outputs zero if $a_i(\mathbf{e_1})$ is equivalent to $a_i(\mathbf{e_2})$, otherwise it outputs one.

For an unlabeled sample $\mathbf{e_t}$ and a set of labeled samples $\{\mathbf{e_i}|i = 1, ..., n\}$ falling in a leaf, we assign a weight to each of the labeled sample $\mathbf{e_i}$ based on its distance to $\mathbf{e_t}$ (as in 7):

$$w_i = 1 - \frac{d_i}{\sum_{i=1}^{n} d_i}. \tag{7}$$

In (7), $d_i = d(\mathbf{e_t}, \mathbf{e_i})$ is the distance of an unlabeled sample $\mathbf{e_i}$ to $\mathbf{e_t}$. Notice that for any two labeled samples $\mathbf{e_i}$ and $\mathbf{e_j}$, the shorter the distance to $\mathbf{e_t}$ (assuming that $d_i \leq d_j$), the larger weight the sample ($w_i \geq w_j$). That implies a labeled sample nearest to $\mathbf{e_t}$ contributes most when calculating the probability for $\mathbf{e_t}$.

We generate the probability estimates of the unlabeled sample by normalizing all weights of the labeled samples at a leaf in terms of their class values, as in (8):

$$p(c_j|\mathbf{e_t}) = \frac{\sum_{k=1}^{m} w_k^j + \frac{1}{|C|}}{\sum_{k=1}^{n} w_k + 1}, \tag{8}$$

where n represents the number of samples at the leaf and m is the number of samples that belong to c_j.

3.2 *LazyTree* Induction Algorithm

We deploy a lazy probability estimator at each leaf of a decision tree and call this model *LazyTree*. To induce the model, we adopt a heuristic search process, in which we exhaustively build all possible trees in each step and keep only the best one for the next level expansion. Suppose that finite k attributes are available. When expanding the tree at level q, there are k-q+1 attributes to be chosen. On each iteration, each candidate attribute is chosen as the root of the (sub) tree, the generated tree is evaluated, and we select the attribute that achieves the highest *gain ratio* as the next level node to grow the tree. We consider two criteria for halting the search process. We could stop splitting when none of the alternative attributes can statistically significantly upgrade the classification accuracy. Or, to avoid the "fragmentation" problem, there are at least 30 samples at the current node. Besides, we still permit splitting if the relative increment in accuracy is not a negative value, which is greedier than C4.5. The tree model is represented as T. An unlabeled sample $\mathbf{e_t}$ is assigned a set of class probabilities as in algorithm 1.

Algorithm 1 $LazyTrees(T, \mathbf{e_t})$ return $\{p(c_j|\mathbf{e_t})\}$

T: a model with a set of leaves L
S_l: a set of labeled samples at a leaf l
$\mathbf{e_t}$: an unlabeled sample
$\{p(c_j|\mathbf{e_t})|c_j \in C\}$: a set of probability estimates of $\mathbf{e_t}$

Dispatch $\mathbf{e_t}$ into one leaf l according to its attributes
for each labeled sample $\mathbf{e_{train}} \in S_l$ **do**
  Calculate the distance d_{train} between $\mathbf{e_{train}}$ and $\mathbf{e_t}$, by utilizing Equation 6
  Calculate the sample weight w_{train}, in terms of its similarity to $\mathbf{e_t}$, by utilizing Equation 7
for each class value $c_j \in C$ **do**
  Use Equation 8 to compute $p(c_j|\mathbf{e_t})$
Return a set of probability estimates $\{p(c_j|\mathbf{e_t})\}$ for the unlabeled sample $\mathbf{e_t}$

4 Empirical Study

We used 36 well-recognized sample sets from the UCI repository [2] and conducted three groups of experiments in terms of ranking, within the *Weka* [10] Platform. These sample sets come from real-world problems and vary in characteristics. Table 1 lists the properties of the sample sets. The preprocessing stages of sample sets include the following four steps:

Table 1. Brief description of sample sets used in our experiments.

Data Set	Size	Classes	Missing	Numeric	Sample Set	Size	Classes	Missing	Numeric
anneal	898	6	Y	Y	ionosphere	351	2	N	Y
anneal.ORIG	898	6	Y	Y	iris	150	3	N	Y
audiology	226	24	Y	N	kr-vs-kp	3196	2	N	N
autos	205	7	Y	Y	labor	57	2	Y	Y
balance	625	3	N	Y	letter-2000	2000	26	N	Y
breast	286	2	Y	N	lymph	148	4	N	Y
breast-w	699	2	Y	N	mushroom	8124	2	Y	N
colic	368	2	Y	Y	p.-tumor	339	21	Y	N
colic.ORIG	368	2	Y	Y	segment	2310	7	N	Y
credit-a	690	2	Y	Y	sick	3772	2	Y	Y
credit-g	1000	2	N	Y	sonar	208	2	N	Y
diabetes	768	2	N	Y	soybean	683	19	Y	N
glass	214	7	N	Y	splice	3190	3	N	N
heart-c	303	5	Y	Y	vehicle	846	4	N	Y
heart-h	294	5	Y	Y	vote	435	2	Y	N
heart-s	270	2	N	Y	vowel	990	11	N	Y
hepatitis	155	2	Y	Y	waveform-5000	5000	3	N	Y
hypoth.	3772	4	Y	Y	zoo	101	7	N	Y

1. Applying the filter of *ReplaceMissingValues* in *Weka* to replace the missing values of attributes.
2. Applying the filter of *Discretize* in *Weka* to make numeric attributes discrete. Therefore, all the attributes are treated as nominal.
3. It is well known that, if the number of values of an attribute is almost equal to the number of samples in a sample set, this attribute does not contribute any information to classification. So we use the filter of *Remove* in *Weka* to delete these attributes. Three occurred within the 36 sample sets, namely *Hospital Number* in sample set *Horse-colic.ORIG*, *Instance Name* in sample set *Splice* and *Animal* in sample set *Zoo*.
4. Due to the relatively high time complexity of *LazyTree*, we apply the filter of unsupervised *Resample* in *Weka* to reselect sample set *Letter* and generate a new sample set named *Letter-2000*. The selection rate is 10%.

To avoid the zero-frequency problem, we used the *Laplace* estimation. More precisely, assume that there are n_c samples that have the class label as c, t total samples, and k class values in a sample set. The *Laplace* estimation calculates the estimated probability $p(c)$ as $p(c) = \frac{n_c+1}{t+k}$. Similarly, $p(a_i|c)$ is calculated by $p(a_i|c) = \frac{n_{ic}+1}{n_c+v_i}$, where v_i is the number of values of attribute A_i and n_{ic} is the number of samples in class c with $A_i = a_i$.

In the first group of our experiments, *LazyTree* was compared to C4.5 and its PET variants including C4.5-L (C4.5 with *Laplace* estimation), C4.5-M (C4.5 with m-Branch) and C4.5-LY (C4.5 with *Ling*&*Yan*'s algorithm). In the second group, we compared *LazyTree* with C4.4 and its PET variants, which contain C4.4-nLa (C4.4 without *Laplace* estimation), C4.4-M (C4.4 with m-Branch) and C4.4-LY (C4.4 with *Ling*&*Yan*'s algorithm). In the last group, we made a comparison between LazyTree-B (*LazyTree* with *bagging*) and C4.5-B (C4.5 with *bagging*) and C4.4-B (C4.4 with *bagging*). We implemented *LazyTree*, AUC metric, m-Branch method, *Ling*&*Yan*'s algorithm, C4.5-M, C4.5-LY, C4.4-M and

C4.4-LY within the *Weka* [10], and used the current implementations of other learning models and *bagging* method in *Weka*. Note that some sample sets have more than two class values and we deal with this multi-class AUC situation by M-measure[4]. The AUC value on each sample set was measured via a ten-fold cross validation ten times. Runs with various models were carried out on the same training sets and evaluated on the same testing sets. In particular, the cross-validation folds were the same for all the experiments on each sample set. Finally, we performed two-tailed t-tests with a significantly different probability of 0.95 to compare our model with others. That is, we speak of two results for a sample set as being "significantly different" only if the difference is statistically significant at the 0.05 level according to the corrected two-tailed t-test[7]. Now, our observations are highlighted as follows.

1. *LazyTree* achieves remarkably good performance on AUC among C4.5 and its variants in Table 2 and Table 3. *LazyTree* performs significantly better than C4.5 (31 wins and 0 loss). As the results show, C4.5 variants can improve the AUC values of C4.5. However, *LazyTree* considerably outperforms these models in AUC (compared with C4.5-L, 15 wins and 1 loss; compared with C4.5-M, 9 wins and 4 losses; compared with C4.5-LY, 17 wins and 0 loss). In addition, decision trees with m-Branch or with *Ling&Yan*'s algorithm can not generate multiple probabilities for the samples falling into the same leaf.
2. In Table 4 and Table 5, *LazyTree* is the best model among C4.4 and its variants in AUC. *LazyTree* significantly outperforms C4.4 (10 wins and 1 loss). Since C4.4 is the *state-of-art* decision tree model designed for yielding accurate ranking, this comparison provides strong evidence to the ranking performance of *LazyTree*. *LazyTree* also outperforms most of C4.4 variants in AUC (compared with C4.4-nLa, 28 wins and 0 loss; compared with C4.4-M, 2 wins and 3 losses; compared with C4.4-LY, 15 wins and 1 loss).
3. LazyTree-B performs greatly better than C4.5-B and C4.4-B. C4.5-B has good ranking performance compared with other typical models. However, *LazyTree*, which uses a new similarity metric at leaves, with *bagging* applied, produces better AUC values than C4.5-B (15 wins and 0 loss). Our model also performs better than C4.4-B (7 wins and 2 losses). Due to space limitation, we did not present the experiment results in this paper.
4. Besides having good performance on ranking, *LazyTree* also has better robustness and stability than other models. The average standard deviation of *LazyTree* in AUC is 4.37, which is lowest among all models.

Generally speaking, *LazyTree* is a trade-off between the quality of probability-based ranking and the comprehensibility of results when selecting the best model. Furthermore, we observed that *Ling&Yan*'s algorithm works better in a relatively small tree and m-Branch works consistently in any tree size.

5 Conclusion

In this paper, we analyzed that traditional decision trees have inherent defects in achieving precise ranking, and proposed to resolve those issues by representing

the similarity between each sample at a leaf and an unlabeled sample. One key observation is that for a leaf, deploying a lazy probability estimator is an optimal alternative to produce a unique probability estimate for a specific sample, compared with directly using frequency-based probability estimation based on the leaf samples. Experiment results prove our expectation that *LazyTree* outperforms or is competitive with typical decision tree models in ranking quality. In our future work, there is much room to make further progress. For instance, after learning a tree structure using the *LazyTree* algorithm, other parameter-learning methods can be used to tune the probability estimation at a leaf. Additionally, we can find the right tree size for our model, i.e. possibly grow a tree into C4.5 and use the *leave-one-out* technique to learn a weight for each sample at leaves, then continue fully splitting the tree, and lastly, for the samples at each leaf, we can normalize the weights and output the normalized values as the probability estimation for that leaf.

References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting amd variants. *Artificial Intelligence*, 36, 1999.
2. C. Blake and C.J. Merz. Uci repository of machine learning database.
3. P. A. Flach C. Ferri and J. Hernandez-Orallo. Improving the auc of probabilistic estimation trees. In *Proceedings of the 14th European Conference on Machine Learning (ECML2003)*. Springer, 2003.
4. D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45, 2001.
5. Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
6. C. X. Ling and R. J. Yan. Decision tree with better ranking. In *Proceedings of the 20th International Conference on Machine Learning (ICML2003)*. Morgan Kaufmann, 2003.
7. C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52(40), 2003.
8. F. J. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(30), 2003.
9. F. J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, 1999.
10. I. H. Witten and E. Frank. *Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.
11. B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the 18th International Conference on Machine Learning (ICML2001)*. Springer, 2001.
12. H. Zhang and J. Su. Conditional independence trees. In *Proceedings of the 15th European Conference on Machine Learning (ECML2004)*. Springer, 2004.

Table 2. Results of AUC & standard deviation: comparing *LazyTree* with C4.5 and its variants

Sample Set	LazyTree	C4.5	C4.5-L	C4.5-M	C4.5-LY
anneal	94.94±4.60	83.49±5.05 •	89.12±4.34 •	89.48±5.93 •	90.94±4.77 •
anneal.ORIG	93.75±6.76	85.93±5.12 •	89.23±6.61 •	90.07±7.77 •	90.19±5.08 •
audiology	69.93±1.51	61.50±1.55 •	62.04±2.36 •	62.70±2.42 •	68.94±2.01
autos	94.17±2.28	73.70±5.77 •	91.98±3.78 •	93.59±2.65	87.97±4.33 •
balance-scale	54.89±7.58	52.67±7.13 •	56.56±8.43	55.04±7.03	61.07±7.37
breast-cancer	67.19±12.29	61.34±10.14•	62.26±9.25	62.34±9.22	66.95±11.35
breast-w	98.39±1.20	96.42±3.04 •	97.44±2.24	97.40±2.26	96.20±3.53 •
colic	86.12±7.49	81.30±8.24 •	84.86±7.06	85.45±6.70	83.84±7.61
colic.ORIG	85.45±6.89	83.30±7.50 •	84.38±6.34	85.13±6.03	76.79±9.31 •
credit-a	91.19±3.74	88.39±4.34 •	89.94±3.74	90.09±3.68	89.71±3.82
credit-g	71.40±5.14	69.36±5.44 •	72.18±4.47	72.67±4.51	70.14±5.52
diabetes	78.80±6.34	76.33±6.88 •	77.59±6.51	77.95±6.39	76.14±6.13
glass	81.88±6.33	77.29±5.27 •	80.00±5.96	80.36±6.07	78.00±8.04
heart-c	83.53±0.63	83.12±0.74 •	83.25±0.65	83.27±0.66	83.59±0.65
heart-h	83.73±0.61	81.08±4.56	81.15±4.59	81.18±4.60	81.61±4.01
heart-statlog	84.18±8.77	81.25±9.47 •	84.91±7.88	85.03±7.92	84.04±8.49
hepatitis	82.67±12.14	69.09±15.74•	70.16±14.77•	70.31±14.85•	71.56±15.70•
hypothyroid	83.65±6.76	68.18±6.85 •	63.68±6.65 •	68.25±6.74 •	78.87±7.56 •
ionosphere	92.03±4.81	88.71±5.94 •	89.87±5.15 •	90.14±5.01	80.39±8.96 •
iris	98.79±2.22	98.94±1.65	98.58±2.09	98.58±2.09	98.72±2.08
kr-vs-kp	99.92±0.10	99.81±0.26	99.88±0.12	99.88±0.12	99.71±0.22 •
labor	93.83±13.01	79.65±20.97•	84.15±18.11•	84.15±18.11•	84.02±18.81•
letter-2000	90.72±1.51	86.45±1.68 •	85.84±1.92 •	92.08±1.27 ○	85.62±2.19 •
lymph	86.33±4.97	71.44±9.71 •	86.21±5.30	86.29±5.31	86.53±5.15
mushroom	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.44±0.19 •
primary-tumor	72.26±3.37	64.19±3.03 •	68.38±3.04 •	70.79±2.94	72.41±3.53
segment	99.09±0.40	98.30±0.71 •	98.88±0.45 •	98.94±0.43 •	94.01±1.10 •
sick	99.12±1.70	93.63±4.41 •	93.79±4.19 •	94.43±3.68 •	92.32±4.71 •
sonar	75.42±10.32	70.07±10.39•	73.80±10.90	74.16±10.82	70.95±10.79
soybean	99.09±1.02	98.07±1.82	97.89±1.64 •	98.94±1.36	98.45±2.67
splice	98.11±0.71	96.85±1.10 •	98.09±0.84	98.17±0.80	98.14±0.65
vehicle	86.23±2.88	82.69±3.73 •	87.02±2.70	87.32±2.58 ○	79.00±3.70 •
vote	98.46±1.96	96.86±3.08 •	97.59±2.26	97.59±2.26	98.49±1.74
vowel	93.74±2.03	92.79±2.05 •	90.56±2.50 •	95.63±1.43 ○	85.05±3.35 •
waveform-5000	86.64±1.22	84.48±1.52 •	86.95±1.20 ○	88.54±1.16 ○	84.87±1.67 •
zoo	87.90±3.88	79.57±5.53 •	80.10±6.28 •	80.29±5.91 •	85.67±4.36
Average	87.32±4.37	82.12±5.29	84.40±4.84	85.17±4.74	84.18±5.31

•, ○ statistically significant degradation or improvement compared with *LazyTree*

Table 3. Summary on t-test of AUC comparisons on *LazyTree*, C4.5 and its variants. An entry $w/t/l$ means the number of wins, ties and loses for the model at the corresponding row, compared to the model at the corresponding column.

Models	C4.5	C4.5-L	C4.5-M	C4.5-LY
C4.5-L	12/22/2			
C4.5-M	18/18/0	8/28/0		
C4.5-LY	13/19/4	5/23/8	5/22/9	
LazyTree	31/5/0	15/20/1	9/23/4	17/19/0

Table 4. Results of AUC & standard deviation: comparing *LazyTree* with C4.4 and its variants

Sample Set	LazyTree	C4.4	C4.4-nLa	C4.4-M	C4.4-LY
anneal	94.94±4.60	93.67±6.25	83.12±5.48 •	93.74±6.10	92.64±3.78 •
anneal.ORIG	93.75±6.76	91.01±8.07 •	85.57±5.21 •	92.07±8.27	91.30±6.62 •
audiology	69.93±1.51	64.04±2.38 •	62.61±1.42 •	64.96±2.20 •	69.36±1.90
autos	94.17±2.28	91.33±4.13 •	73.94±5.86 •	94.06±2.37	87.30±4.40 •
balance-scale	54.89±7.58	61.40±6.73 ○	56.12±5.49	56.35±6.06	63.36±6.66 ○
breast-cancer	67.19±12.29	60.53±10.08	57.53±11.17•	61.85±10.78	61.93±12.25
breast-w	98.39±1.20	98.22±1.25	96.04±2.66 •	98.23±1.26	89.62±5.54 •
colic	86.12±7.49	83.96±7.41	79.14±7.56 •	87.00±7.30	83.38±7.48
colic.ORIG	85.45±6.89	83.00±6.55	78.08±8.22 •	83.98±6.16	75.18±10.36•
credit-a	91.19±3.74	89.59±3.81	84.85±4.98 •	91.26±3.57	89.03±3.97
credit-g	71.40±5.14	70.07±4.70	63.95±5.59 •	73.48±4.41	69.64±5.62
diabetes	78.80±6.34	76.20±5.10	70.29±5.84 •	78.95±5.35	70.62±7.71 •
glass	81.88±6.33	80.11±6.91	74.70±6.44 •	81.57±6.56	79.11±8.57
heart-c	83.53±0.63	83.27±0.75	82.78±0.91 •	83.46±0.70	83.17±0.90
heart-h	83.73±0.61	83.30±0.64 •	82.50±0.93 •	83.66±0.63	82.98±0.97 •
heart-statlog	84.18±8.77	82.81±8.28	78.17±9.59 •	85.29±7.98	82.45±8.75
hepatitis	82.67±12.14	79.50±14.28	71.34±17.24	80.93±13.76	72.85±17.17
hypothyroid	83.65±6.76	80.62±8.24	85.47±7.85	85.64±7.81	80.87±10.73
ionosphere	92.03±4.81	93.43±4.44	86.56±7.55 •	93.11±4.68	82.04±6.86 •
iris	98.79±2.22	98.67±1.98	97.40±3.00	98.69±1.99	97.87±2.79
kr-vs-kp	99.92±0.10	99.93±0.08	99.81±0.27	99.93±0.08	99.83±0.17
labor	93.83±13.01	87.17±18.05	84.60±18.31	89.35±16.49	85.81±20.46
letter-2000	90.72±1.51	85.26±2.05 •	86.03±1.68 •	92.59±1.25 ○	84.51±2.21 •
lymph	86.33±4.97	86.30±4.58	71.71±8.87 •	87.03±4.49	86.35±5.54
mushroom	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.44±0.19 •
primary-tumor	72.26±3.37	68.53±3.05 •	63.40±2.82 •	71.59±2.89	72.79±3.57
segment	99.09±0.40	99.08±0.42	98.11±0.71 •	99.19±0.39	94.60±1.25 •
sick	99.12±1.70	99.03±0.66	96.37±3.09	99.16±0.62	95.07±3.67
sonar	75.42±10.32	78.38±9.04	72.33±9.56 •	79.24±9.15	73.32±11.87
soybean	99.09±1.02	98.02±1.62 •	97.70±1.89 •	98.94±1.36	98.38±2.69
splice	98.11±0.71	98.06±0.71	94.96±1.23 •	98.49±0.64 ○	97.08±1.04 •
vehicle	86.23±2.88	85.96±2.75	78.11±3.71 •	87.34±2.57	79.29±3.80 •
vote	98.46±1.96	97.43±2.37	96.50±3.31 •	97.49±2.38	98.53±1.67
vowel	93.74±2.03	91.57±2.34 •	92.14±2.13 •	96.07±1.41 ○	85.25±3.58 •
waveform-5000	86.64±1.22	81.36±1.41 •	79.03±1.61 •	87.15±1.21	82.78±1.91 •
zoo	87.90±3.88	80.26±6.35 •	79.76±5.60 •	80.52±6.00 •	85.62±4.37
Average	87.32±4.37	85.59±4.65	81.69±5.22	87.01±4.41	83.98±5.58

•, ○ statistically significant degradation or improvement compared with *LazyTree*

Table 5. Summary on t-test of AUC comparisons on *LazyTree*, C4.4 and its variants. An entry $w/t/l$ means the number of wins, ties and loses for the model at the corresponding row, compared to the model at the corresponding column.

Models	C4.4	C4.4-nLa	C4.4-M	C4.4-LY
C4.4-nLa	1/11/24			
C4.4-M	16/18/1	26/10/0		
C4.4-LY	5/20/11	13/18/5	4/16/16	
LazyTree	10/25/1	28/8/0	2/31/3	15/20/1