# Diagnosing Cancerous Abnormalities with Decision Tree Learning

J. P. Mennicke
joerg.mennicke@stud.uni-bamberg.de

T. Hecker
thomas.hecker@stud.uni-bamberg.de

***Abstract*— This paper evaluates two decision tree learning algorithms, C4.5 and CAL5, for diagnosing cancerous abnormalities from pre-processed digital images. For this purpose, the former algorithms were extended to allow for unattended optimization of their training parameters and to deal with imbalanced datasets. An evaluation of these extended algorithms produced encouraging results, if sufficient data is available.**

## I. Introduction

One discipline of machine learning, inductive inference, is concerned with generating models for a problem in the light of exemplary datasets. Whenever the underlying structure of a problem cannot be modeled mathematically, the approach of inducing general regularities that are implicitly given in historical cases shows good results in most problem situations. An evaluation of common induction methods is given in Michie et al. (1994).

Often the task is to predict a class for so far unclassified instances, which are described by feature-vectors $x \in X$, where $X \subseteq A_1 \times A_2 \times \ldots \times A_m$ with each $A_i$ defining one of the vector's *attributes* or *dimensions*. The problem of predicting a class for such instances can be described as to find an approximation $h$ (*hypothesis*) for an unknown target function $f : X \rightarrow C$, where $C$ is a finite set of class-labels (Mitchell, 1997).

Learning algorithms (or: *inducers*) that build decision trees are efficient and robust methods for providing a hypothesis for a variety of classification problems. For example, Michie et al. (1994) evaluated the most common decision tree inducers on different domains. Many decision tree inducers are available for data represented by feature vectors consisting of both discrete- and continuous-valued attributes.

A decision tree is equivalent to a set of if-then-rules and therefore, unlike some other function representations, both comprehensible to humans and suitable to describe disjunctive concepts. Furthermore, decision trees are robust to noise in the data and can also be capable of handling missing attribute values in feature vectors (Mitchell, 1997).

The present paper evaluates two decision tree inducers for diagnosing cancerous abnormalities from medical images. For this purpose, the images—e.g. microscopic or endoscopic exposures— were pre-classified by human experts and then transformed into numerical feature vectors (cf. Wittenberg et al., 2001). These instances, consisting of the images' feature vectors and their assigned classifications, were used to train a decision tree that approximates the target function and is therefore capable of classifying unseen images by their feature representations.

Several machine learning methods like *artificial neural networks (ANN)* or *k-nearest neighbor (k-NN)* have already been applied to this domain (cf. Wittenberg et al., 2001; Zhou, Jiang, Yang, & Chen, 2002). By deploying a decision tree inducer for the given classification problem, one may expect several advantages over the former methods.

### A. Advantages of Decision Trees

*1) Comprehensible Hypotheses:* The rules that can be extracted from an induced decision tree capture the essence of the classification problem at hand while still being comprehensible to human experts. Hence, decision support should be improved:
Human experts will be able to base their decisions not only on a class identified by the hypothesis, but also on the criteria that led to this decision. Furthermore, the ability to extract and compare criteria resulting from different classification approaches of several experts could, for example, be used to both analyze systematic misclassifications of particular approaches and to identify the most common and successful classification rules for training purposes.

This however, would require to reverse the pre-processing phase of the original images to a certain degree in order to transform the numerical values that are present in the rule conditions to (fuzzy) linguistic terms (e.g. 'dark' or 'light' instead of a numerical luminance value in a certain region). Such rules could then easily be integrated with *fuzzy expert systems*.

*2) High Prediction Accuracy Across Domains:* One may expect the decision tree inducers to produce hypotheses with higher accuracy compared to *k-NN*, because *k-NN* has several shortcomings which decision tree inducers do not share.

One problem of *k-NN* classifiers is, that the distance of two instances is always based on *all* dimensions of the instance space. This fact becomes an important issue in high-dimensional instance spaces when the amount of relevant attributes is low in comparison to the overall attribute count.

In such scenarios, the distance between two instances is dominated by the large amount of irrelevant attributes (Mitchell, 1997), affecting the *k-NN* classifier's ability to find the truly most similar instances (based on the relevant attributes) in order to classify unseen instances. This phenomenon is also known as the *curse of dimensionality*.

To overcome this issue, *k-NN* classifiers assign appropriate weights to the individual attributes according to their relevance in the classification process. When Wittenberg et al. (2001) conducted their experiments with *k-NN*, which dealt with similar data, they assumed that among the 100+ feature vector dimensions, some of them would be redundant or irrelevant. Finding the optimal weights for the dimensions, however, requires a significant computational effort (Friedman, 1994). Therefore, various approaches exist that use heuristic search for assigning weights to attributes or to completely remove irrelevant ones. It is however evident, that the heuristic search principles underlying these approaches, may not produce optimal results. Wettschereck and Aha (1995) give an evaluation of the effects of some of these approaches on *k-NN*'s prediction accuracy.

Decision tree inducers do not face the problems resulting from the *curse of dimensionality* as they aim to select only relevant attributes when forming the hypothesis. However, it is to be mentioned, that under certain conditions decision tree inducers may also suffer from the presence of irrelevant attributes. This may happen for example, when data is sparse and only few examples can be used to grow certain regions of the decision tree. In this case, irrelevant attributes may exhibit accidental regularities that outweigh the patterns contained in the relevant attributes. The decision tree inducer will then fail to select the relevant attributes in these subtrees and thus overfit the training data (Mitchell, 1997). Kohavi and John (1997) showed that removing irrelevant features from the training data may significantly improve the accuracy of decision inducers on some problems.

Although decision tree inducers may suffer from irrelevant attributes under certain conditions, this issue is not as important to decision tree learning as it is to *k-NN* methods. The reason for this is, that the feature selection process for *k-NN* yields a *global approximation* of the most relevant attributes, while decision tree inducers grow subtrees by *locally* selecting the most relevant attributes according to a specific region in the feature space. The problem of globally selecting/weighting attributes for *k-NN* classifiers exposes another problem, which lies in the assumption that feature relevancy is invariant across the whole instance space (Wettschereck & Aha, 1995).

However, it may seem obvious that this assumption does not hold for many problems. Figure 1 visualizes an exemplary scenario where the complete instance space is projected onto a single dimension $A_1$. From the distribution of the classes (+) and (-) alongside the $A_1$ dimension, one would assume that the attribute $A_1$ is relevant for instances which have a high value on this attribute. For instances with a low value of $A_1$ though, this attribute seems irrelevant and thus counterproductive to the classification process.

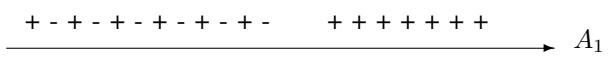


Fig. 1. Relevance of a single Attribute in the Feature Space

This issue becomes even more important when considering the dependency of an instance's class on different attributes. Given there are four attributes $A, B, C, D$ and two classes $c_1, c_2$. If the class depends on these attributes such that the classification rules

$$A \geq v_1 \wedge B \geq v_2 \rightarrow c_1$$

$$C \geq v_3 \wedge D \geq v_4 \rightarrow c_2$$

always hold, then all four attributes are relevant, but only two of them at a time, while the other two are being counterproductive. Since classes are disjunctive, the above rules mandate that the rules

$$c_1 \rightarrow C < v_3 \vee D < v_4$$

$$c_2 \rightarrow A < v_1 \vee B < v_2$$

also hold. If many attributes are relevant in such a way, then this would have a major impact on the accuracy of a *k-NN* classifier, due to the curse of dimensionality.

Since there is reason to believe that such dependencies exist within the current domain, a *k-NN* classifier may address this issue basically in two ways:

- increasing the number of instances in the dataset so that the $k$ nearest neighbors for any given instance are very close with respect to any dimension
- using *locally adaptive* distance metrics, based on the unseen case's position in the instance space

The first option may not be desirable, as the amount of required instances grows exponentially with the dimensions. The second option which is discussed by Friedman (1994) seems more feasible, but adds further computational effort to the classification phase (see also I-A.3). Furthermore, it increases the degrees of freedom and therefore the risk of overfitting (Mitchell, 1997).

Decision trees, on the other side, are inherently able to deal very well with such dependencies and therefore do not share this problem, although problems may arise if many of the attributes are simultaneously interacting, as demonstrated by Quinlan (1993).

Nevertheless, when Michie et al. (1994) compared several classifiers on datasets of pre-processed image data (*Vehicle* and *Chrom*), *k-NN* outperformed the decision tree methods discussed in this paper. It must be noted, however, that the datasets had only few dimensions (16 and 18 attributes respectively), while the datasets discussed in this paper have considerably higher dimensionalities (up to 180).

At last, there is no reason to believe that the prediction accuracy of *ANN* methods should be lower than that of decision tree inducers. However, if both should perform

equally well, decision trees should be preferred for the sake of comprehensible hypotheses, as discussed earlier.

*3) Fast Classification:* Decision trees and *ANN*, on the one hand, belong to the class of *eager learners*, as they generalize over the training data during the training phase, requiring low computational effort for classification. *k-NN* methods, on the other hand, are *lazy learners*, since they postpone the generalization task to the classification phase itself, requiring them to generalize each time a new unseen instance is to be classified (Mitchell, 1997). Although the feature-selection process represents a certain form of generalization for *k-NN*, the computational effort to retrieve the $k$ nearest neighbors in high dimensional feature spaces may be significant, especially for values $k > 1$ in large databases (Michie et al., 1994).

### B. Overview

The paper is organized as follows. Section II provides a short revision of decision tree learning in general and describes the algorithms used in this paper, including the extensions for parameter optimization and balancing of the training data. Section III describes the experiment settings, including a description of the datasets. Section IV outlines the results of the two algorithms on the domain in question followed by section V which gives an interpretation of the experiments' findings and identifies areas of improvement. Finally, section VI summarizes the conclusions drawn from this paper and gives recommendations for further work.

## II. Evaluated Inducers

The decision tree inducers C4.5 (Quinlan, 1993) and CAL5 (Unger & Wysotzki, 1981), that were evaluated for this work, both underlie some general principles which are discussed in the following two sections, while the subsequent sections are used to describe their specific properties as well as the extensions for handling imbalanced datasets and parameter optimization.

### A. Top-Down Induction Process

Decision trees in general are constructed *top-down* using a simple *divide-and-conquer* approach that was first introduced by Hunt et al. (1966). Given a set of instances $S$, the inducer chooses an *appropriate* attribute test with $k$ outcomes in order to partition the instances into subsets $S_1, \ldots, S_k$. In the decision tree, this attribute test is represented by an inner node, where each outcome links to a child node. This procedure is then applied recursively to all resulting childs/partitions $S_i$ until one of the following conditions holds:

- $S$ contains only instances of the same class $c$
- $S$ does not contain any instances
- $S$ contains instances of multiple classes, but there are no attributes left to test

In these cases, $S$ will be represented by a leaf node, to which a class-label $c \in C$ is assigned. While in the first scenario the class assignment can be made unambiguously, the other two usually require to choose the majority class in $S$ or the majority class of the parent node. However, the two latter scenarios are less relevant for the domain in question, due to the manner in which the inducers evaluated here perform tests on continuous attributes. However, choosing a majority class for a leaf node becomes important for the *pruning* of decision trees, which will be discussed in subsequent sections.

### B. Selection of Attribute Tests

Decision trees as hypothesis representation language are so expressive that there exists a multitude of different hypotheses/trees that are consistent with the given training data. *Occam's Razor* emphasizes that the simplest of those hypotheses is to be preferred, because it is more likely that such a hypothesis identifies the true patterns contained in the training data and may therefore also be capable of classifying unseen instances (Mitchell, 1997). Hence, decision tree inducers generally attempt to find the smallest possible decision tree that is still consistent with the given data (although consistency is partly traded for simplicity). Since finding such a minimal decision tree is a *NP-complete* problem, inducers employ search heuristics in order to identify the best attribute test in a given decision tree node.

A popular search heuristic, that was first employed by the ID3 algorithm (Quinlan, 1986), is based on *Information Theory*. It assesses the suitability of an attribute test to discriminate the instances of different classes at a certain node by the *Information Gain* measure. Mitchell (1997) describes this measure as follows:

Let there be $c$ classes contained in a set of instances $S$. The *Entropy* of $S$ is then defined as

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \ log_2 \ p_i$$

where $p_i$ is the proportion of class $i$ in $S$. If an attribute $A$ is to be tested on the instances in $S$, the information gain of this attribute test is computed as

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where $Values(A)$ are the possible values of attribute $A$ and $S_v$ are the resulting partitions of $S$ containing all instances with the attribute value $A = v$. Hence, the information gain is the reduction in entropy when $S$ is partitioned according to attribute $A$. Decision tree inducers that use the information gain measure select the attribute with the highest information gain in a given node, in order to quickly find leaf nodes that contain only instances of a single class. This local greedy search may quickly produce small trees, however it is not guaranteed to find the smallest tree possible (Mitchell, 1997).

### C. Description of the Algorithm C4.5

The C4.5 inducer (Quinlan, 1993) is an extension of the ID3 inducer (Quinlan, 1986) that includes several improvements such as pruning facilities, as well as support for continuous-valued attributes, and missing attribute values. The following sections describe the relevant features of C4.5. However, features relating to discrete-valued attributes and missing attribute

values were omitted, since they do not apply to the data that was used for evaluation.

*1) Attribute Selection:* As a descendant of ID3, C4.5's attribute selection test is based on the information gain measure described earlier, but uses by default an extended measure —*gain ratio*— that addresses one problem of the traditional gain measure related to attribute tests with (infinitely) many outcomes (e.g. a timestamp), where each partition contains only a single instance. The information gain measure for such an attribute would achieve the highest value possible, since it creates perfectly pure partitions. This attribute test would, however, be rather impractical for classifying unseen instances, as each new instance may have a distinct value on this attribute, that was not observed in the training data. Hence, the gain ratio measure penalizes attributes that produce many small partitions. In Mitchell (1997), the gain ratio measure is defined as

$$GainRatio(S,A) \equiv \frac{Gain(S,A)}{SplitInformation(S,A)}$$

where *SplitInformation* is defined as

$$SplitInfomration(S,A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} log_2 \frac{|S_i|}{|S|}$$

Although gain ratio is more sophisticated than the information gain measure, there is little difference between the two when testing continuous-valued attributes in C4.5, as these attribute tests have only two different outcomes (see next point). Note that since C4.5's remaining features do not depend on a specific measure, the following parts will simply refer to them as the *gain* measure.

*2) Tests on Continuous-Valued Attributes:* When considering a continuous-valued attribute $A$ for testing in a decision tree node, C4.5 orders all instances in $S$ according to their value of $A$. If two adjacent instances have different classes, a threshold

$$s = \frac{v_i + v_{i+1}}{2}$$

is determined, where $v_i$ and $v_{i+1}$ are the values of attribute $A$ for these two instances. Hence, if the instances in $S$ contain $m$ different values of $A$, there exist $m-1$ possible thresholds. Each of these thresholds is then considered for splitting $S$ into two partitions $S_1, S_2$, where $S_1$ contains all instances with $A < s$ and $S_2$ the remaining instances. The gain for these splits is then computed and the threshold with the highest gain is selected.

Note that C4.5 only allows to perform a binary test for continuous-valued attributes. It is obvious that such a test may be insufficient when there are more than two adjacent class regions in $A$. Therefore, C4.5 allows to further test the same attribute in descendant nodes, in order to refine the test on this attribute.

*3) Probabilistic Thresholds:* Although the construction of decision trees is robust to noise, the classification of unseen instances that contain noise may be prone to error under certain conditions. When a decision tree node tests a continuous-valued attribute $A$ on an unseen instance, even minor noise in that attribute value can cause the instance to be sent into the wrong subtree, if the instance's true value of $A$ was close to the threshold $s$. Therefore, C4.5 allows *probabilistic thresholds* — or: *soft thresholds*— to be used for testing continuous-valued attributes. When applying soft thresholds, an instance that is tested on $A$ is sent into *both* subtrees of the respective node, but weighted with a value $w \in [0,1]$ in one subtree and $1-w$ in the other. $w$ depends on the instance's assigned value of $A$ and its distance to the threshold. An instance may, therefore, be weighted several times as it is tested in multiple nodes. Since a single instance reaches multiple leaf nodes, the final class assignment is made by selecting the leaf node in which this instance experiences the highest weight.
Note that soft thresholds are not used in hypotheses that were generated using rule-postpruning (see section II-C.6).

*4) Prepruning:* Decision tree inducers that build hypotheses from noisy datasets tend to produce overly complex trees in which many leaf nodes represent only few instances of the training set. Such hypotheses usually overfit the training data and are therefore less capable of classifying unseen instances. To avoid overfitting, smaller decision trees are preferred, even though they are not perfectly consistent with the training data. Two approaches exist in order to prevent overfitting: *prepruning* (or: *stopping*) and *postpruning*. C4.5 allows to use both. For prepruning, C4.5 deploys a user-defined parameter $m$ ("minobjs"), which requires each attribute test to obtain at least *two* outcomes with $|S_i| \geq m$. Consequently, this means for continuous-valued attributes that both partitions produced by the split must contain at least $m$ instances. If no such test exists for a set of instances $S$, then $S$ is not partitioned any further and replaced by a leaf node to which the majority class in $S$ is assigned. If not specified, C4.5 uses the default value $m = 2$.

*5) Tree-Postpruning:* The prepruning approach described above bears the risk that the inducer stops too early when growing the tree. Therefore, growing the tree until it overfits the training data and subsequently identifying and replacing irrelevant subtrees by leaf nodes or one of their branches is considered to be more successful. C4.5 adopts by default an *error-based pruning* approach that considers each inner tree node for pruning. It, therefore, estimates the errors for classifying unseen instances that would result from replacing the associated subtree either by a leaf node or one of its branches. For this purpose, C4.5 relies on a statistical test that is solely performed on the training data: For each of the subtree's leaf nodes a confidence interval for the true error on unseen instances can be computed based on the errors on the training data, assuming a binomial distribution and a user-defined confidence level $CF_t$ (default: $CF_t = 25\%$). The upper bound of this confidence interval is then used as a pessimistic estimate for the true error. The estimates are then multiplied with the amount of instances classified in the respective nodes and summed for the whole subtree. The resulting estimate for the whole subtree is then compared

against the estimated error of the pruned subtree. Subtrees are pruned in this manner until further pruning would increase the estimated error.
Although this pruning algorithm makes several simplifying assumptions in order to calculate an error estimate over the entire population of instances, it has shown to increase hypothesis accuracy in practice.

*6) Rule-Postpruning:* A shortcoming of the tree-postpruning approach discussed above, is the fact that attribute tests at inner nodes cannot be removed without affecting the whole subtree. Therefore, C4.5 offers another postpruning method that abandons the tree structure converting it to a set of rules. Each of these rules corresponds to a path from the root to some leaf node. The attribute tests of the inner nodes form the rule's preconditions, while the class assigned to the leaf node represents the rule's postcondition. The *rule-postpruning* facility of C4.5 considers each rule precondition as a candidate for removal, using again a pessimistic estimate of the rule's true error on a given confidence level $CF_r$ (default: $CF_r = 25\%$), which is similar to the statistical test performed during tree-postpruning. Among all rules, C4.5 greedily removes the precondition for which the highest decrease in error is estimated, then starting all over again and repeating the process until no more preconditions can be removed.

In addition to the test based on the pessimistic error estimate, C4.5 allows to test each rule precondition for statistical significance, removing all preconditions which cannot be determined significant on a user-defined confidence level $CF_s$. This additional significance test is, however, turned off by default.

Finally, the remaining rules are grouped into sets each predicting only a single class. These class-sets are further reduced by determining an optimal subset of rules according to the *Minimum Description Length Principle*. Hereby, the sum of costs for encoding a set of rules (theory costs) and for encoding the misclassified instances (exception costs) is to be minimized. A further option enables C4.5 to take into account possible redundancies among the attributes, weighting theory costs up by a redundancy factor $W$ (default: $W = 1.0$).

Although rule-postpruning may produce shorter and more accurate rules, it has two disadvantages: Class-concepts are no longer disjunctive and a classifier must process the rules sequentially until it identifies a rule that holds for the given instance. The fact that class-concepts are no longer disjunctive means implies that several rules with different class-assignments may hold for a given instance. Hence, the order in which rules are processed becomes important. For this purpose, the class-sets are ranked and ordered according to the amount of *false positives* they predict on the training data (i.e. the instances belonging to other classes for which the rule also holds). Finally, a default rule is created that assigns some default class to all instances for which no other rule holds.

*7) Windowing:* With C4.5's *windowing* facility, trees are no longer grown from the entire training set. Instead, an initial window of the training set is formed by randomly selecting a defined number of instances (default: 20%), but ensuring a uniform class distribution. From this window an initial tree is grown and tested against the remaining instances of the training set. Then, the window is extended by including a subset of all misclassified instances (default: 20% of the initial window size, but at least half of the misclassified instances) and a new tree is trained. This process is repeated until all remaining instances are correctly classified or accuracy does not further improve. The windowing facility further allows to perform multiple trials with different initial windows, always retaining the final tree (default: $t = 10$ trials). Then, the tree with the lowest predicted error is selected for post-pruning.

The fact that windowing randomly selects instances for the initial window adds a certain form of non-determinism to to the C4.5 inducer: subsequent runs of C4.5 over the same dataset may produce different hypotheses. This effect may not always be desired, although it can be reduced by increasing the number of trials.

Nevertheless, windowing also increases the chance of finding smaller and more accurate hypotheses, due to the fact that multiple trials use different initial windows. A serious disadvantage of C4.5's windowing facility is, however, that it can significantly increase the time needed to learn a hypothesis.

### *D. Description of the Algorithm CAL5*

The decision tree inducer, CAL5 (Unger & Wysotzki, 1981; Müller & Wysotzki, 1994) is specifically designed for handling continuous-valued attributes. A comparison of the performance of CAL5 with classical statistical methods and a variety of other learning methods is given by Müller and Wysotzki (1997) and Michie et al. (1994).

In the light of statistical considerations, CAL5 automatically converts continuous attributes into intervals, so that an optimal discrimination of the classes in the feature space is to be achieved.

In other words, CAL5 separates the feature space into areas in which a class $c_i$ dominates with the probability $p(c_i) \geq s$, with $s$ being a user-defined threshold. While $s$ provides a lower boundary for the estimated class probabilities within an interval, it is used to automatically pre-prune the tree during its construction process whenever the conditional probability of a class at a certain node exceeds $s$. This allows the user to control the accuracy of the approximation of class regions in the feature space.

The class probability $p(c_i)$ itself is estimated on a user-defined confidence level $1 - \alpha$, which allows to adjust the demanded quality of estimation. A higher confidence interval $1 - \alpha$ requires a larger amount of data within each interval to obtain sufficient statistics on which to base the estimation of $p(c_i)$.

CAL5 constructs the decision tree top-down starting with one attribute, recursively branching out the nodes with other attributes until a sufficient classification probability has been achieved.

*1) Selection of the Best Attribute:* Before each branching step, a selection process is performed that identifies the best

attribute on which to base the next test in the decision tree. Therefore, each attribute is evaluated by CAL5 using one of two different discrimination measures — the entropy (Quinlan, 1986) or a statistical measure (Meyer-Broetz & Schuermann, 1970) — both indicating the local ability of an attribute to discriminate classes at the current node.

As in first attempts the entropy measure seemed to produce better or at least equal results compared to the statistical measure, the tests in this paper are solely based on the entropy measure. Therefore, the statistical measure is not described here. A description of this measure is contained in Müller and Wysotzki (1994).

Calculating the entropy in order to identify the best discriminator at a certain node requires the considered attribute tests to have discrete outcomes. Thus, continuous-valued attributes used in CAL5 need to be converted into intermediate intervals first, which can then be handled like values of ordered discrete attributes. At *each* node *all* attributes must be split into intervals using the CAL5 splitting procedure described below. For these intervals the best local discriminator can then be selected on the basis of the respective information gain, as described in section II-B.

Note that *all* attributes are tested in each node for their discrimination ability. This procedure may cause an attribute to be selected multiple times within the same path of the tree, resulting each time in a refinement of the preceding interval.

*2) Splitting Continuous-valued Attributes into Intervals:* Let $S$ be the set of all $n$ examples reaching the current node $x$. If $A$ is the next continuous-valued attribute for further branching the tree, the instances contained in $S$ are ordered along the axis of the attribute $A$. Now intervals $I$ are formed collecting examples from left to right. After an interval has been closed according to one the following conditions, the next interval is opened until no more examples are available:

- *A class decision can be made with confidence:* There exists a class $c_i$ in $I$ for which an estimation $p(c_i|x) \geq s$ can be made on a confidence level $1-\alpha$. In this case $c_i$ dominates in $I$. Then, $I$ can be closed and is labeled with the class symbol $c_i$. The classification error on the data is $e < 1-s$ with a confidence $1-\alpha$.
- *No class decision can be made with confidence:* For all classes $c_i$ occurring in $I$ the inequality $p(c_i|x) < s$ holds on a given confidence level $1-\alpha$. In this case no class dominates in $I$. Then, $I$ will be closed and is labeled for further branching.
- *No more samples are available for further extending $I$:* In this case $I$ is closed and labeled with the most common class $c_i$ in $I$.

To test these conditions whilst considering the confidence level, one assumes that $n_i$ (the number of instances with class label $i$) has a Bernoulli distribution. This assumption requires that the set of examples is created by an independent, stationary source in terms of its classes. Note that no specific class distribution on the current attribute axis is needed.

Thus, a confidence interval $[d_1, d_2]$ can be computed using the formula

$$d_{1,2} = \frac{2\alpha n_i + 1}{2\alpha n + 2} \mp \frac{1}{2\alpha n + 2}\sqrt{4\alpha n_i(1 - \frac{n_i}{n}) + 1}$$

where $\alpha$ is the significance level set by the user and $1-\alpha$ the corresponding confidence level. Given the Bernoulli distribution of class labels for each class $c_i$ this formula can be derived from the Tschebyschev inequality.

As the true value of the conditional probability $p(c_i|x)$ for a sequence of examples lies between the boundaries $d_1$ and $d_2$ of the confidence interval with a probability of $1-\alpha$, one can test the dominance of a class in $I$ with respect to the given confidence level, by testing the following cases for all classes occurring in $I$:

- Class $c_i$ dominates in $I$ on the confidence level $1-\alpha$, if $d_1 \geq s$, where $d_1$ is the lower bound of the confidence interval
- No class $c_i$ occurring in $I$ dominates on the confidence level $1-\alpha$, if $d_2 < s$, where $d_2$ is the upper bound of the confidence interval, holds for all $c_i$.

As a result several intervals $I_l$ are obtained, which are labeled with either a class $c_i$ (leaves) or an indication for further branching (intermediate nodes).

*3) Merging adjacent intervals:* In a final step adjacent intervals $I_l$ and $I_{l+1}$ with same labels can be merged. Also adjacent intervals where no class dominates can be merged according to a heuristic elimination procedure:

Eliminate each class in the intervals for which the inequality

$$d_2(c_i) < \frac{1}{n_I}$$

holds. Here, $n_I$ is the number of different class labels in $I$ and $d_2(c_i)$ is again the upper bound of the confidence interval calculated for class $c_i$. If —after applying the elimination procedure— all remaining classes of adjacent intervals are equal, intervals can be merged.

After this final step all intervals, which are labeled for further branching (intermediate nodes), become the next starting node $x$ and are recursively expanded in the same way until all branches of the tree are closed and labeled with a unique class $c_i$.

### *E. Handling of Imbalanced Datasets*

Imbalanced datasets may pose a problem to decision tree inducers, as they tend to optimize classification accuracy toward over-represented classes. This phenomenon was previously observed by many researchers (cf. Provost & Fawcett, 1997; Kubat & Matwin, 1997). Therefore, an additional parameter was introduced that enabled the algorithms to balance the training set before starting the induction process. Drummond and Holte (2003) investigate the two different approaches for balancing datasets:

- *under-sampling* removes instances of over-represented classes from the training data

- *over-sampling* duplicates existing instances of under-represented classes for inclusion in the training data, or creates synthetic instances from them

Findings of their experiments show that under-sampling techniques tend to yield better results. Therefore, such an approach was also adopted for the experiments in this paper: The algorithm first counts the instances for each class and then determines the smallest class-count $m > 0$. Then, the algorithm iterates over the training set, removing all instances of a class that exceed this minimum count.

### F. Parameter Optimization

Most inducers allow to parametrize the induction process in order to adapt to specific properties of the given data. As most inducers have a multitude of possible parameters, finding optimal configurations manually is a tedious task. John (1994) investigated the possibility to unattendedly optimize induction parameters using *k-fold cross-validation*. However, his work focused only on a single parameter of C4.5 and the hypothesis' accuracy estimates were optimistically biased since parameter optimization and error estimation were performed on the same set of instances.

To ensure that the choice for the best configuration of inducer-specific parameters does not bias the results of the experiments, parameter optimization and error estimation were performed on disjoint test- and validation sets. This section describes the parameter optimization process. The evaluation method for the generalization errors on unseen cases is described in section III-B.

*a) K-fold Cross-Validation in General:* k-fold cross-validation in general splits the available data $D$ into $k$ equally sized partitions. The inducer runs $k$ times over the data, while each time a different partition is withheld from the training process. This holdout partition, also referred to as the test- or validation set (depending on the respective purpose), is used to evaluate the generalization error of the resulting *hypothesis*. Finally, these errors are averaged over the $k$ runs to provide an estimate of the *inducer's* generalization error. The advantage of k-fold cross-validation is that it makes best use of the available data, since each instance is used once for evaluation and $k-1$ times for training.

*b) K-fold Parameter Optimization:* In order to determine the best parameters for an inducer on a given dataset, a set of potential parameter configurations is created and each configuration is evaluated using k-fold cross-validation. The configuration with the lowest average error performs best on the given data and is therefore selected to generate a final hypothesis over the whole data.

Note that the average error on the validation sets is only used to determine the best configuration. This error cannot be used as an estimate for the true generalization error, since the configuration was optimized on the whole training data. Therefore, the actual error estimate for the extended inducer algorithms including parameter optimization is determined on a separate test set, as described in III-B.

## III. Experiments

The performance of the two inducers —including their extensions— was tested on eight datasets, each belonging to one of five distinct subdomains within the domain of diagnosing cancerous abnormalities from pre-processed images. Tests were performed with CAL5 v2.3[1] and C4.5 R8[2] with the OFAI patch[3].

### A. Description of the Data

Instances of the datasets are described by vectors consisting solely of continuous attributes which represent features extracted from the original images. The feature vectors did not contain any missing attribute values. Table I lists the properties of the individual datasets. The amount of distinct classes varied between 2 and 32. Feature vectors consisted of 48–180 attributes, which did even vary within a single subdomain. Most datasets were relatively small with a size of approx. 300 instances. The remaining datasets contained between 482 and 749 instances, dataset I499P892 even conatined 9610 instances.

### B. Evaluation Method

The performance of the inducers were evaluated using 10-fold cross-validation. To avoid biased results due to an unequal distribution of instances across the dataset partitions created for cross-validation, the instances have been randomly shuffled prior to the evaluations.
Note that the cross-validation method used for evaluation of the inducers is distinct from the cross-validation that is used for parameter optimization. The evaluation is performed on the inducers *including* their extension for parameter optimization.

Thus, within each of the $k_e$ evaluation runs, the parameter optimization procedure is performed with a complete cross-validation cycle (using $k_o$ runs) on the reduced dataset —excluding the instances of the test set used for the evaluation procedure described here.

The holdout partitions used during cross-validation in the parameter optimization process are therefore referred to as 'validation sets', in order to allow distinction from the test sets of the evaluation procedure. Hence, the complete evaluation process comprises two nested cross-validation loops. This approach is quite time-consuming, since the evaluation of each dataset requires

$$total\ hypotheses\ generated = k_e \cdot k_o \cdot n_c$$

hypotheses to be generated, where $n_c$ indicates the amount of potential parameter configurations that are considered by the parameter optimization process.

This approach was chosen despite its high computational costs, because it makes best use of the available data and identifies optimal parameter configurations. As some datasets

[1] http://ki.cs.tu-berlin.de/software/DIPOL_CAL5.html – Accessed: 02/09/06
[2] http://www.rulequest.com/Personal/ – Accessed: 02/09/06
[3] http://www.ofai.at/~johann.petrak/c45ofai.html – Accessed: 02/09/06

TABLE I
STRUCTURES OF DATA SETS

| **I183P102** | | **I183P1035** | | **I186P102** | | **I233P1017** | |
|---|---|---|---|---|---|---|---|
| Subdomain: 1 | | Subdomain: 1 | | Subdomain: 1 | | Subdomain: 1 | |
| 482 Cases | | 482 Cases | | 300 Cases | | 300 Cases | |
| 48 Attributes | | 180 Attributes | | 48 Attributes | | 48 Attributes | |
| Class | # Cases | Class | # Cases | Class | # Cases | Class | # Cases |
| 1 | 182 | 1 | 182 | 1 | 97 | 1 | 97 |
| 2 | 122 | 2 | 122 | 2 | 80 | 2 | 80 |
| 5 | 178 | 5 | 178 | 5 | 123 | 5 | 123 |

| **I023P891** | | **I003P886** | | **I001P003** | | **I499P892** | |
|---|---|---|---|---|---|---|---|
| Subdomain: 2 | | Subdomain: 3 | | Subdomain: 4 | | Subdomain: 5 | |
| 320 Cases | | 749 Cases | | 499 Cases | | 9610 Cases | |
| 96 Attributes | | 48 Attributes | | 180 Attributes | | 48 Attributes | |
| Class | # Cases | Class | # Cases | Class | # Cases | Class | # Cases |
| 1–32 | 10 each | 1 | 559 | 1 | 116 | 1 | 5611 |
| | | 2 | 190 | 2 | 74 | 3 | 1214 |
| | | | | 3 | 44 | 4 | 563 |
| | | | | 4 | 103 | 5 | 247 |
| | | | | 5 | 91 | 6 | 1975 |
| | | | | 6 | 71 | | |

contain only around 300 instances, withholding fixed sets of instances for both evaluation and parameter optimization appeared prohibitive. Also, optimizing parameters on a completely separated validation set of relatively small size is likely to yield a suboptimal decision.

In order to keep evaluation times at a reasonable level, $k_e$ and $k_o$ were both set to 10. The main problem, however, was the amount of potential parameter configurations resulting from the combination of the individual parameter values. Especially C4.5 suffers from this combinatorial explosion, due to its large amount of parameters. Therefore, the potential values for each learning parameter were restricted to a few reasonable values.

### C. Optimized Parameter Ranges for C4.5

The following list contains the values for each of the learning parameters to which the parameter optimization process of C4.5 was restricted to:

- training set balancing: yes , no
- attribute selection criterion: gain ratio only
- thresholds: soft , hard
- windowing: yes , no
- windowing parameters:
  - initial window size: auto
  - window increment: auto
  - trials: 2
- minobjs (pre-pruning): 1 , 2 , 4
- postpruning: tree , rule , none
- tree-postpruning parameters:
  - $CF_t$: 10 , 25 , 40
- rule-postpruning parameters:
  - $CF_r$: 10 , 25 , 40
  - significance testing: no
    - $CF_s$: n/a
  - redundancy factor: 1.0 , 2.0

These parameter values can be combined to 168 possible configurations. Note that some parameter values restrict other parameters, e.g. soft thresholds cannot be used with rule postpruning.

The attribute selection criterion was limited to the gain ratio measure, since there is no significant difference to the information gain values when testing continuous attributes. Settings for the initial window size and window increment values were left to C4.5's default values, as optimizing predetermined values increases the amount of possible configurations considerably. The windowing trials were limited to 2, since windowing significantly increases training times. Although more trials are generally desirable, this compromise had to be made since evaluation times of more than a week — even for the smallest datasets— with C4.5's default of 10 trials was prohibitively long. Also, significance testing for rule preconditions was disabled, since early tests did not show any significant increases in performance.

### D. Optimized Parameter Ranges for CAL5

The CAL5 algorithm has three parameters that are relevant for optimization on the current datasets. These parameters, plus the extension for training set balancing, were restricted as follows:

- training set balancing: yes , no
- attribute selection criterion: information gain only
- alpha: 11 values from the interval [0,0.5] in steps of 0.05
- threshold: 11 values from the interval [0.5,1.0] in steps of 0.05

These parameter values yield 242 possible configurations. The attribute selection criterion was limited to the information gain measure, since it outperformed the statistical measure

in early tests. The values for alpha and threshold were restricted to the intervals above, since values $alpha > 0.5$ and $threshold < 0.5$ seem not desirable due to their meaning in the training process.

## IV. Results

This section summarizes the results for both algorithms and outlines general findings that can be drawn from the empirical observations. Table II shows the generalization errors of the two algorithms on the various datasets. More detailed results on the individual datasets are contained in the Appendix, where Tables VI–XIV state the averaged classification matrices.

TABLE II
Comparison of Generalization Errors between CAL5 and C4.5

| Dataset | Size | CAL5 | C4.5 | $\Delta error$ |
|---|---|---|---|---|
| I183P102 | 482 | 30.5% | 26.1% | 4.4% |
| I183P1035 | 482 | 26.7% | 20.5% | **6.2%** |
| I186P102 | 300 | 30.7% | 30.3% | 0.4% |
| I233P1017 | 300 | 29.3% | 22.7% | **6.6%** |
| I023P891 | 320 | 47.8% | 17.8% | **30.0%** |
| I003P886 | 749 | 17.1% | 14.8% | 2.3% |
| I001P003 | 499 | 29.4% | 20.6% | **8.8%** |
| I499P892 | 9610 | 20.5% | 20.0% | 0.5% |

Significant differences of the "one-tailed binomial test for the difference in proportionvalues (normal approximation)" are highlighted using **boldface**.

TABLE III
Comparison of Generalization Errors on Reduced Datasets

| Dataset | Size | CAL5 | C4.5 |
|---|---|---|---|
| I499P892L | 2074 | 20.9% | 21.8% |
| I499P892S | 306 | 34.6% | 33.0% |
| $\Delta error$ | | **13.7%** | **11.2%** |

Datasets I499P892L and I499P892S were generated from independent samples of dataset I499P892. Significant differences of the "one-tailed binomial test for the difference in proportionvalues (normal approximation)" are highlighted using **boldface**.

### A. General Findings

Upon first inspection of the results two evident findings can be derived:

First, C4.5 showed considerably better results than CAL5 over all datasets. Hence the hypothesis:

**H1:** C4.5 performs better than CAL5 on the domains in question.

This hypothesis can be supported with evidence by considering the statistical significance of the results, although differences in performace are less evident for the larger datasets. Significant differences in the errors of the two inducers are highlighted using **boldface** in Table II (one-tailed binomial test for the difference in proportionvalues using the normal approximation, confidence level 0.05).

Second, for both algorithms the best generalization errors are observed on the two largest datasets. Therefore one can generate the hypothesis:

**H2:** Larger datasets yield lower generalization errors for both C4.5 and CAL5.

To support this hypothesis with evidence another experiment was performed on disjoint subsets of dataset I499P892 of size 2074 and 306 respectively. The results of this test were statistically significant to support hypothesis **H2** with evidence (one-tailed binomial test for the difference in proportionvalues using the normal approximation, confidence level 0.05). Significant differences in the two datasets are again highlighted using **boldface** in Table III, which shows the performance of the two inducers on the reduced datasets. Note that the error on the larger subset (2074 instances) increased only marginally in comparison to the original dataset (9610 instances). Thus, it can be inferred for dataset I499P892, that approx. 2000 instances are sufficient to produce stable results. The significant drop in accuracy on the smaller subset (306 instances) shows that a dataset of this size cannot warrant an accurate hypothesis to be generated by these inducers. In the light of these findings and by considering the sizes of the other datasets, the error estimates for these datasets should be treated with caution, since it is likely that the generated hypotheses were not sufficiently supported by the training data.

A closer inspection of the classification matrices on the two largest datasets shows that both inducers minimize the classification error for majority classes at the expense of the classification accuracy for under-represented classes. Table IV shows this behavior on dataset I499P892, where error rates increase considerably as the number of instances per class decreases.

Hence, the hypothesis,

**H3:** Both algorithms bias their classification behavior toward over-represented classes.

In order to determine whether the unequal distribution of the classes is solely accountable for the high variance in error rates among the different classes, another experiment was performed on dataset I499P892. In this experiment the algorithmes were forced to balance the training set in order to ensure an equal distribution of the classes. The results are given in Table V. Although the error on the smaller classes improved slightly, while errors on larger classes increased, the variance in error rates among classes is still rather high.

### B. Other Findings

- The lowest error for both algorithms was obtained on dataset I003P886 (15.8% and 17.1% respectively). This is, however, not suprising, as this dataset contains only two classes with a clear majority for one class (74.6%). Even a simple majority classifier would therefore have an average classification error of only 25.4%.

TABLE IV
CLASSIFICATION MATRIX FOR DATASET I499P892

| **CAL5** | | | | | | | | **C4.5** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Average Error: | | 20.5% | | | | | | Average Error: | | 20.0% | | | | | |
| **Average Classification Matrix** | | | | | | | | **Average Classification Matrix** | | | | | | | |
| Actual Class | Avg. Cases per Class | Classified as | | | | | Misses in % | Actual Class | Avg. Cases per Class | Classified as | | | | | Misses in % |
| | | 1 | 3 | 4 | 5 | 6 | | | | 1 | 3 | 4 | 5 | 6 | |
| 1 | 561.1 | 540.8 | 8.5 | 0.9 | 0.2 | 10.7 | 3.6% | 1 | 561.1 | 527.4 | 8.2 | 3.9 | 0.1 | 21.5 | 6.0% |
| 3 | 121.4 | 19.8 | 62.2 | 3.0 | 0.3 | 36.1 | 48.8% | 3 | 121.4 | 5.7 | 65.5 | 3.4 | 0.2 | 46.6 | 46.0% |
| 4 | 56.3 | 17.2 | 9.2 | 2.0 | 0.9 | 27.0 | 96.4% | 4 | 56.3 | 6.4 | 6.1 | 5.3 | 0.2 | 38.3 | 90.6% |
| 5 | 24.7 | 5.5 | 2.7 | 1.5 | 0.6 | 14.4 | 97.6% | 5 | 24.7 | 2.1 | 2.6 | 0.6 | 0.2 | 19.2 | 99.2% |
| 6 | 197.5 | 21.9 | 13.2 | 3.2 | 0.7 | 158.5 | 19.7% | 6 | 197.5 | 11.7 | 9.9 | 3.7 | 1.9 | 170.3 | 13.8% |

Results of the average classification performance per test set on dataset I499P891. Results were obtained by 10-fold cross-validation. The average size of the test sets was 961 cases.

TABLE V
CLASSIFICATION MATRIX FOR DATASET I499P892 WITH FORCED BALANCED TRAINING SETS

| **CAL5** | | | | | | | | **C4.5** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Average Error: | | 22.9% | | | | | | Average Error: | | 22.6% | | | | | |
| **Average Classification Matrix** | | | | | | | | **Average Classification Matrix** | | | | | | | |
| Actual Class | Avg. Cases per Class | Classified as | | | | | Misses in % | Actual Class | Avg. Cases per Class | Classified as | | | | | Misses in % |
| | | 1 | 3 | 4 | 5 | 6 | | | | 1 | 3 | 4 | 5 | 6 | |
| 1 | 561.1 | 533.2 | 8.8 | 0.4 | 0.9 | 17.8 | 5.0% | 1 | 561.1 | 531.2 | 6.6 | 8.5 | 0.3 | 14.5 | 5.3% |
| 3 | 121.4 | 23.2 | 50.0 | 2.9 | 1.2 | 44.1 | 58.9% | 3 | 121.4 | 15.8 | 56.0 | 11.6 | 0.7 | 37.3 | 53.9% |
| 4 | 56.3 | 16.2 | 7.2 | 3.1 | 1.1 | 28.7 | 94.5% | 4 | 56.3 | 10.4 | 7.0 | 8.4 | 0.7 | 29.8 | 85.1% |
| 5 | 24.7 | 4.5 | 2.2 | 1.2 | 0.3 | 16.5 | 98.8% | 5 | 24.7 | 5.5 | 2.2 | 1.2 | 1.0 | 14.8 | 96.0% |
| 6 | 197.5 | 20.1 | 15.8 | 6.0 | 1.0 | 154.6 | 21.7% | 6 | 197.5 | 22.3 | 16.9 | 10.4 | 1.0 | 146.9 | 25.6% |

Average classification performance on dataset I499P892 with forced balancing. Results were obtained by 10-fold cross-validation.
Note that the avg. class count relates to the test sets, which were not balanced. The avg. class count on the training sets was $0.9 \cdot 247 = 222.3$ for each class.

The following findings can not be explained that easily and would therefore require a more detailed inspection of the datasets in question:

- Surprisingly the performance of the two inducers differs remarkably on the dataset I023P891. CAL5's generalization error on this dataset was the highest over all datasets with 47.8%, while C4.5 achieved one of it's best errors with 17.8%.
- Both inducers yield considerably worse results on the two datasets I186P102 and I183P102 compared to the others (with exception of CAL5's performance on the dataset discussed in the previous point).

## V. DISCUSSION

In the previous section three hypotheses were introduced, for which the reasons will be discussed here in more detail as well as their implications for areas of improvement.

The first hypothesis —C4.5 performs better than CAL5 on the domains in question— can be explained by the way how C4.5 tackles various shortcomings of decision trees associated with handling noise and the algorithms' greedy search during tree construction:

1) C4.5's soft thresholds are a sophisticated means for handling noise in continuous-valued attributes. This prevents instances that are located close to the decision surface from being classified in the wrong subtree, when the value of a relevant attribute is distorted by minor noise, which is inherent to image-processing.
2) The greedy top-down search through hypothesis space is a general problem of decision trees. The assessment of suitable attributes based on a local partition of the training set may neither yield the smallest, nor the best decision tree (Mitchell, 1997). Selecting the wrong attribute at the upper levels of the decision tree may fragment the data in a way that makes further testing difficult for classes that are sparsely represented in the resulting partitions. As already mentioned in section I-A.2, this increases the risk of overfitting. The problem of early fragmentation of the training data is more imminent for the CAL5 algorithm, since CAL5 may create many small partitions with a single attribute test, while C4.5 uses only binary splits. Also, C4.5's windowing facility can alleviate the effects of local greedy search by growing multiple trees and selecting the best one.
3) There are various approaches that address the issue of overfitting in decision trees. CAL5 solely uses pre-pruning to prevent overfitting. However, two independent findings show that growing the tree until it overfits the training data and then post-prune it, generally produces more accurate hypotheses (Breiman et al., 1984; Quinlan, 1993). C4.5 uses both pre- and postpruning, while pre-pruning is usually very conservative (mostly $m = 1$ or 2) in favor of stronger postpruning. Additionally, the rule-postpruning facility of C4.5 allows to

remove irrelevant attributes at the tree's upper levels, while preserving relevant attributes that were selected in descendant nodes.

Hence, one may expect CAL5's performance — especially on smaller datasets — to improve by extending the algorithm with postpruning, windowing and soft thresholds.

The second hypothesis —larger datasets yield lower generalization errors for both C4.5 and CAL5— may again be explained by the fact that some subtrees are grown with small partitions of the training data, which may cause accidental patterns to dominate the true patterns, hence the risk of overfitting. Removing irrelevant attributes can be expected to alleviate this effect on smaller datasets. Apart from this, it is possible that the smaller datasets did not even contain all patterns necessary for accurate class predictions.

However, one must also add that the estimated error in the experiments is expected to be pessimistically biased by the 10-fold cross-validation method, which used 10% of the data for evaluating the error and thus withholding these instances from the induction process (Kohavi, 1995). The impact of this reduction in the training data is expected to be stronger for small datasets than for larger ones. Using less data for the evaluation by increasing the number of folds (or even using leave-one-out cross-validation) should give a lower error estimate that is closer to the true error. This would, however, further increase the computational effort.

The last hypothesis —both algorithms bias their classification behavior toward over-represented classes— is a problem that many induction algorithms (not only decision trees) are susceptible to and which is therefore widely discussed (see e.g. Provost & Fawcett, 1997; Kubat & Matwin, 1997).

One explanation for this behavior of C4.5 and CAL5 may again be the greedy selection of attribute tests. If one class is overrepresented in the dataset, the attribute that separates most of the instances from this class will be selected first, although some instances of the other classes may fall into the same partition. Hence the under-represented classes will be fragmented early between several subtrees, reducing the chance to determine the truly relevant attribute tests for these instances. Fragmented instances of the under-represented classes also pose a problem for C4.5's post-pruning facilities, which are inclined to replace (potentially correct) subtrees, that were only created from a small set of instances, by a single leaf node in favor of a simpler hypothesis.

It is also noticeable that balancing was hardly used due to the way parameter optimization was performed. The naive under-sampling method that was used for balancing generally reduced the overall size of the training set in favor of equally distributed classes, without taking into account that some instances may be more important for the training process than others. As the second test showed, balancing may have reduced the differences in error rates among classes to a certain degree, it would also have increased the overall error on the (unbalanced) validation set. Since the parameter optimization process was solely determined to minimize this error, those parameter configurations that balanced the training sets had a low probability of being selected as optimal.

Additionally, C4.5's windowing facility already uses a balanced subset of the training data for the initial window, which causes the classes to be treated equally when the first tree is grown. Since misclassified instances of the remaining training set are included in subsequent windows, C4.5 can still achieve better overall error rates than with training sets that were balanced beforehand.

To address the problem of unequal treatment of minority classes, several improvements are conceivable:

1) One solution may be not to use a single decision tree in order to classify all instances, but to subsequently modify the training set in order to grow an ensemble of several decision trees, where each of the trees is used to predict a single class with high accuracy and combining their weighted votes for a class prediction. Common approaches for this include Bagging (Breiman, 1996), Boosting (Freund & Schapire, 1996) and Randomization (Dietterich, 1999). Empirical evaluations of these methods show that the accuracy of decision tree classifiers can be significantly improved with such ensembles (cf. Bauer & Kohavi, 1999; Dietterich, 1999).
2) Although the generalization error is a commonly used measure to assess the quality of an inducer, for certain problems —especially cancer diagnosis— minimizing the costs for misclassifying instances may be more desirable. Various approaches exist to incorporate misclassification costs into the training process (Bradford et al., 1998; Chawla et al., 2002; Domingos, 1999; Pazzani et al., 1994). Since CAL5 was originally designed to directly incorporate such costs, it may be expected that its performance will improve on this task. However, the optimization of an inducer to minimize misclassification costs will not improve the problem of misclassifying under-represented instances in general, it may only do so for classes with high misclassification costs (i.e. critical classes) that are also under-represented.
3) A simpler means to tackle this issue could be to base the parameter optimization process on a balanced validation set, thus eliminating the preference for parameter configurations that minimize the overall error in favor of majority classes.

The first improvement can be expected to reduce the generalization error for each class individually and therefore the overall generalization error. In contrast, the other two improvements are expected to reduce variance in prediction errors between the classes at the expense of higher overall generalization errors.

Although balancing reduced the differences in error rates for dataset I499P892 to a certain degree, the variance is still high. Therefore, another reason must exist which causes the uneven error rates between classes.

On the one hand, certain classes may generally be harder to classify because the instances form many small and complex class regions. If these class regions are represented only by few instances within the training data, the inducer may not be

capable to determine the relevant patterns within these regions, causing the tree to overfit the data. This may be the case for classes 3, 4 and 5 in dataset I499P892 and can only be prevented by providing more instances of these classes for the training process.
On the other hand, classes that form few simple class regions in the instance space are easy to classify independent from the amount of instances of this class or the proportion of instances compared to other classes. Classes 1 and 6 in dataset I499P892 seem to form such class regions.

Apart from the suggestions derived from the findings of the experiments, there are further potential areas for improvement that apply to most decision tree inducers in general:

1) One major shortcoming of many decision tree inducers is the way how continuous attributes are tested in the decision tree nodes. Quinlan (1993) showed that such an attribute test corresponds to splitting the instance space by a hyperplane that is orthogonal to the dimension of the tested attribute (for CAL5, this may be multiple orthogonal hyperplanes). The class regions in the instance space are therefore approximated by hyperrectangles. In most scenarios, however, class regions are too complex to be bounded by a single hyperrectangle. Hence the decision tree must approximate these class regions through many small hyperrectangles, which requires many instances close to the decision surfaces for a sufficiently accurate approximation. This however may yield very complex hypotheses.
   The CART (Breiman et al., 1984) decision tree inducer partially addresses this issue by testing linear combinations of attributes at a single node instead of a single attribute. Other algorithms have also been proposed that use such multivariate tests (Murthy, Kasif, & Salzberg, 1994; Utgoff & Brodley, 1990).
   These attribute tests, however, come at the cost of less comprehensible hypotheses.
2) Another issue that has already been mentioned is that of irrelevant attributes which exhibit accidental patterns in small partitions of the datasets. Hence the reduction in attributes might further improve the accuracy of both decision tree inducers. The *Wrapper* approach (John, Kohavi, & Pfleger, 1994), which attempts to remove irrelevant attributes from the dataset showed significant improvements in C4.5's performance on some domains (Kohavi & John, 1997).
   Other methods to reduce dimensionality without loss of information are *Singular Value Decomposition (SVD)* or *Principal Component Analysis (PCA)*. These however seem less common for use with decision trees. Fradkin and Madigan (2003) conducted an experiment where C4.5 was tested on datasets whose dimensions where reduced by PCA, but without improvement in C4.5's performance. Apart from the computationally expensive SVD/PCA processing step, decision trees that are built from such processed data become hard to interpret, as the original dimensions are merged into artificial ones, whose meaning is not clearly defined.
3) Another area of improvement are the pruning facilities of both algorithms. Both are based on statistical tests on the training data itself. On the one hand, this has the advantage that no data must be spared for a separate validation set (not to be confused with the validation set that is used here for parameter optimization). On the other hand, such tests may fail to distinguish attribute tests that overfit the training data from those that are relevant for the classification of unseen instances. For example, Kohavi and John (1997) observed in an experiment how C4.5 initially built a correct decision tree, but then pruned back some relevant nodes because the small dataset did not warrant for such a large tree. Several approaches exist that prune decision trees based on a separate validation set. Breslow and Aha (1997) give an overview of these approaches. One may expect that these approaches would further improve the inducers' ability to generalize over the data. Additionally, a balanced validation set for post-pruning could also improve the problem of under-represented classes.

## VI. Conclusions

In this paper an extension for the CAL5 and C4.5 decision tree inducers was proposed that allows for unattended optimization of algorithm-specific parameters and to handle imbalanced datasets. An evaluation of the extended algorithms for the task of diagnosing cancerous abnormalities from preprocessed image data shows that C4.5 outperforms CAL5 on this domain. Although, the difference in performance decreases with larger datasets.

The error rates on small datasets, however, should be treated with care, as it must be assumed that these datasets cannot warrant accurate hypotheses. Additionally, the error estimates are pessimistically biased due to the 10-fold cross-validation method deployed for evaluation. Therefore, the error rates on the two largest datasets encourage to perform further experiments with larger datasets. It is likely that this would further improve error rates on all but the largest dataset, where results are already stable.

The results of the experiments also show that the extension for handling imbalanced datasets has several shortcomings, which prevent the algorithms from properly handling such datasets. Among the recommendations suggested in this paper, Bagging and Boosting techniques for creating decision tree ensembles, incorporation of misclassification costs, alternative postpruning methods based on validation sets and removal of irrelevant attributes using Wrapper-methods, are most promising to achieve further improvements in performance.

Desirable additions to the parameter optimization process include the use of search heuristics to replace the current "brute-force" approach in order to reduce the computational effort required to explore larger sets of potential parameter combinations.

Another interesting approach that should be further investigated for this domain is the use of hybrid decision tree and k-

NN algorithms (Cardie, 1993). In this approach, a decision tree is used to predict the relevant attributes that should be included in a k-NN distance metric. Also, Friedman et al. (1996) propose an algorithm for *Lazy Decision Trees* which attempts to alleviate the problems with decision trees associated to their local greedy search.

## ACKNOWLEDGEMENTS

The authors would like to thank Dr.-Ing. Thomas Wittenberg and Dipl.-Inf. Christian Münzenmayer (Fraunhofer-Institute for Integrated Circuits) for permission to use the comprehensive, real-world data for the experiments.

The authors are, in particular, very grateful for the much appreciated support and tutoring by Prof. Dr. Ute Schmid (Dept. of Information Systems and Applied CS, Bamberg University).

Additional thanks to Nils Nawrot for refreshing our memories in the field of statistics.

## REFERENCES

Bauer, E., & Kohavi, R. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants. *Machine Learning*, 105–142.

Bradford, J. P., Kunz, C., Kohavi, R., Brunk, C., & Brodley, C. E. (1998). Pruning Decision Trees with Misclassification Costs. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning* (pp. 131–136). London: Springer.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *26*(2), 123–140.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Monterey: Wadsworth and Brooks.

Breslow, L. A., & Aha, D. W. (1997). Simplifying Decision Trees: A Survey. *Knowledge Engineering Review*, *12*(1), 1–40.

Cardie, C. (1993). Using Decision Trees to Improve Case–Based Learning. In *Proceedings of the 10th International Conference on Machine Learning* (pp. 25–32). San Francisco: Morgan Kaufmann.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Oversampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Dietterich, T. (1999). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*.

Domingos, P. (1999). MetaCost: A General Method for Making Classifiers Cost-Sensitive. In *KDD '99: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 155–164). New York: ACM Press.

Drummond, C., & Holte, R. C. (2003). *C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-sampling.* In: Workshop on Learning from Imbalanced Data Sets II (2003).

Fradkin, D., & Madigan, D. (2003). Experiments with Random Projections for Machine Learning. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 517–522). New York: ACM Press.

Freund, Y., & Schapire, R. (1996). Experiments with a new Boosting Algorithm. In *Proc. 13th International Conference on Machine Learning* (pp. 148–146). San Francisco: Morgan Kaufmann.

Friedman, J. H. (1994). *Flexible Metric Nearest Neighbor Classification* (Tech. Rep. No. 113). Stanford: Department of Statistics, Stanford University.

Friedman, J. H., Kohavi, R., & Yun, Y. (1996). Lazy Decision Trees. In H. Shrobe & T. Senator (Eds.), *Proceedings of the 13th National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference* (pp. 717–724). Menlo Park: AAAI Press.

Hunt, E., Marin, J., & Stone, P. (1966). *Experiments in Induction*. New York: Academic Press.

John, G. H. (1994). *Cross-Validated C4.5: Using Error Estimation for Automatic Parameter Selection* (Tech. Rep. No. CS-TN-94-12). Stanford: Standford University Computer Science Department.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant Features and the Subset Selection Problem. In *11th International Conference on Machine Learning* (pp. 121–129). San Francisco: Morgan Kaufmann.

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1137–1145).

Kohavi, R., & John, G. H. (1997). Wrappers for Feature Subset Selection. *Artificial Intelligence*, *97*(1-2), 273–324.

Kubat, M., & Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In D. H. Fisher (Ed.), *ICML '97: Proceedings of the 14th International Conference on Machine Learning* (pp. 179–186). San Francisco: Morgan Kaufmann.

Meyer-Broetz, G., & Schuermann, J. (1970). *Methoden der Automatischen Zeichenerkennung*. Berlin: Akademie Verlag.

Michie, D., Spiegelhalter, D. J., Taylor, C. C., & Campbell, J. (Eds.). (1994). *Machine Learning, Neural and Statistical Classification*. Upper Saddle River: Ellis Horwood.

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill.

Müller, W., & Wysotzki, F. (1994). Automatic Construction of Decision Trees for Classification. *Annals of Operations Research*, *52*(4), 231–247.

Müller, W., & Wysotzki, F. (1997). The Decision-Tree Algorithm CAL5 Based on a Statistical Approach to Its Splitting Algorithm. In R. Nakeiazadeh & C. Taylor (Eds.), (pp. 45–65). Wiley.

Murthy, S. K., Kasif, S., & Salzberg, S. (1994). A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, *2*, 1–32.

Pazzani, M. J., Merz, C. J., Murphy, P. M., Ali, K., Hume, T., & Brunk, C. (1994). Reducing Misclassification Costs. In *Proceedings of the 11th International Conference on Machine Learning* (pp. 217–225). San Francisco: Morgan Kaufmann.

Provost, F. J., & Fawcett, T. (1997). Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (pp. 43–48). Menlo Park: AAAI Press.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1), 81–106.

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

Unger, S., & Wysotzki, F. (1981). *Lernfähige Klassifizierungssysteme*. Berlin: Akademie-Verlag.

Utgoff, P. E., & Brodley, C. E. (1990). An Incremental Method for finding Multivariate Splits for Decision Trees. In *Proceedings of the 7th International Conference on Machine Learning* (pp. 58–65). San Francisco: Morgan Kaufmann.

Wettschereck, D., & Aha, D. W. (1995). Weighting Features. In *ICCBR '95: Proceedings of the 1st International Conference on Case-Based Reasoning Research and Development* (pp. 347–358). London: Springer-Verlag.

Wittenberg, T., Neubauer, K., Küblbeck, C., Permanyer, I., & Schmidt, R. (2001). Automatische Tumorerkennung bei unterschiedlichen Organen mittels Berechnung und Klassifikation von Texturmerkmalen. In T. Lehmann, H. Handels, A. Horsch, & H.-P. Meinzer (Eds.), *5. Workshop Bildverarbeitung für die Medizin* (pp. 377–381). Berlin, Heidelberg, New York: Springer.

Zhou, Z.-H., Jiang, Y., Yang, Y.-B., & Chen, S.-F. (2002). Lung Cancer Cell Identification Based on Artificial Neural Network Ensembles. *Artificial Intelligence in Medicine*, *24*(1), 25–36.

## APPENDIX

TABLE VI

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I001P003

| **CAL5** | | | | | | | | **C4.5** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Average Error: | 29.4% | | | | | | | Average Error: | 20.6% | | | | | | |
| **Average Classification Matrix** | | | | | | | | **Average Classification Matrix** | | | | | | | |
| Actual Class | Classified as | | | | | | Misses in % | Actual Class | Classified as | | | | | | Misses in % |
| | 1 | 2 | 3 | 4 | 5 | 6 | | | 1 | 2 | 3 | 4 | 5 | 6 | |
| 1 | 8.7 | 1.1 | 0.2 | 1.3 | 0.2 | 0.1 | 25.0% | 1 | 10 | 0.7 | 0 | 0.8 | 0.1 | 0 | 13.8% |
| 2 | 0.4 | 6.2 | 0 | 0.1 | 0.4 | 0.3 | 16.2% | 2 | 0.6 | 6.2 | 0 | 0.3 | 0.1 | 0.2 | 16.2% |
| 3 | 0 | 0.2 | 3.2 | 0.6 | 0.1 | 0.3 | 27.3% | 3 | 0 | 0 | 3.9 | 0.1 | 0 | 0.4 | 11.4% |
| 4 | 1.1 | 0.1 | 0.5 | 8.6 | 0 | 0 | 16.5% | 4 | 1.2 | 0.2 | 0.2 | 8.5 | 0.2 | 0 | 17.5% |
| 5 | 0.1 | 0.3 | 0 | 0.2 | 5.2 | 3.3 | 42.9% | 5 | 0.1 | 0 | 0 | 0.3 | 6.8 | 1.9 | 25.3% |
| 6 | 0.2 | 0.2 | 0.6 | 0.3 | 2.5 | 3.3 | 53.5% | 6 | 0.2 | 0.1 | 0.1 | 0.3 | 2.2 | 4.2 | 40.8% |

TABLE VII

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I003P886

| **CAL5** | | | | **C4.5** | | | |
|---|---|---|---|---|---|---|---|
| Average Error: | 17.1% | | | Average Error: | 14.8% | | |
| **Average Classification Matrix** | | | | **Average Classification Matrix** | | | |
| Actual Class | Classified as | | Misses in % | Actual Class | Classified as | | Misses in % |
| | 1 | 2 | | | 1 | 2 | |
| 1 | 52.4 | 3.5 | 6.3% | 1 | 51.6 | 4.3 | 7.7% |
| 2 | 9.3 | 9.7 | 48.9% | 2 | 6.8 | 12.2 | 35.8% |

TABLE VIII

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I183P102

| **CAL5** | | | | | **C4.5** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Average Error: | 30.5% | | | | Average Error: | 26.1% | | | |
| **Average Classification Matrix** | | | | | **Average Classification Matrix** | | | | |
| Actual Class | Classified as | | | Misses in % | Actual Class | Classified as | | | Misses in % |
| | 1 | 2 | 5 | | | 1 | 2 | 5 | |
| 1 | 17.3 | 0.3 | 0.6 | 4.9% | 1 | 16.2 | 0.6 | 1.4 | 11.0% |
| 2 | 1.3 | 5.7 | 5.2 | 53.3% | 2 | 0.5 | 6.4 | 5.3 | 47.5% |
| 5 | 1.6 | 5.7 | 10.5 | 41.0% | 5 | 0.8 | 4.0 | 13 | 27.0% |

TABLE IX

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I183P1035

| **CAL5** | | | | | **C4.5** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Average Error: | 26.7% | | | | Average Error: | 20.5% | | | |
| **Average Classification Matrix** | | | | | **Average Classification Matrix** | | | | |
| Actual Class | Classified as 1 | 2 | 5 | Misses in % | Actual Class | Classified as 1 | 2 | 5 | Misses in % |
| 1 | 17.5 | 0.2 | 0.5 | 3.8% | 1 | 17.4 | 0.1 | 0.7 | 4.4% |
| 2 | 0.2 | 6.4 | 5.6 | 47.5% | 2 | 0.4 | 7.8 | 4.0 | 36.1% |
| 5 | 0.9 | 5.5 | 11.4 | 36.0% | 5 | 0.4 | 4.3 | 13.1 | 26.4% |

TABLE X

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I186P102

| **CAL5** | | | | | **C4.5** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Average Error: | 30.7% | | | | Average Error: | 30.3% | | | |
| **Average Classification Matrix** | | | | | **Average Classification Matrix** | | | | |
| Actual Class | Classified as 1 | 2 | 5 | Misses in % | Actual Class | Classified as 1 | 2 | 5 | Misses in % |
| 1 | 8.5 | 0.5 | 0.7 | 12.4% | 1 | 8.5 | 0.1 | 1.1 | 12.4% |
| 2 | 0.7 | 3.9 | 3.4 | 51.3% | 2 | 0.3 | 4.1 | 3.6 | 48.8% |
| 5 | 1.4 | 2.5 | 8.4 | 31.7% | 5 | 0.7 | 3.3 | 8.3 | 32.5% |

TABLE XI

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I233P1017

| **CAL5** | | | | | **C4.5** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Average Error: | 29.3% | | | | Average Error: | 22.7% | | | |
| **Average Classification Matrix** | | | | | **Average Classification Matrix** | | | | |
| Actual Class | Classified as 1 | 2 | 5 | Misses in % | Actual Class | Classified as 1 | 2 | 5 | Misses in % |
| 1 | 8.5 | 0.4 | 0.8 | 12.4% | 1 | 8.9 | 0.3 | 0.5 | 8.2% |
| 2 | 0.6 | 3.6 | 3.8 | 55.0% | 2 | 0.2 | 5.2 | 2.6 | 35.0% |
| 5 | 0.9 | 2.3 | 9.1 | 26.0% | 5 | 1.1 | 2.1 | 9.1 | 26.0% |

TABLE XII

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I499P892

| **CAL5** | | | | | | | **C4.5** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Average Error: | 20.5% | | | | | | Average Error: | 20.0% | | | | | |
| **Average Classification Matrix** | | | | | | | **Average Classification Matrix** | | | | | | |
| Actual Class | Classified as 1 | 3 | 4 | 5 | 6 | Misses in % | Actual Class | Classified as 1 | 3 | 4 | 5 | 6 | Misses in % |
| 1 | 540.8 | 8.5 | 0.9 | 0.2 | 10.7 | 3.6% | 1 | 527.4 | 8.2 | 3.9 | 0.1 | 21.5 | 6.0% |
| 3 | 19.8 | 62.2 | 3.0 | 0.3 | 36.1 | 48.8% | 3 | 5.7 | 65.5 | 3.4 | 0.2 | 46.6 | 46.0% |
| 4 | 17.2 | 9.2 | 2.0 | 0.9 | 27.0 | 96.4% | 4 | 6.4 | 6.1 | 5.3 | 0.2 | 38.3 | 90.6% |
| 5 | 5.5 | 2.7 | 1.5 | 0.6 | 14.4 | 97.6% | 5 | 2.1 | 2.6 | 0.6 | 0.2 | 19.2 | 99.2% |
| 6 | 21.9 | 13.2 | 3.2 | 0.7 | 158.5 | 19.7% | 6 | 11.7 | 9.9 | 3.7 | 1.9 | 170.3 | 13.8% |

TABLE XIII

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I023P891 (CAL5)

| **CAL5** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Average Error: | 47.8% | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Average Classification Matrix** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Actual | Classified as | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Misses |
| Class | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | in % |
| 1 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50.0% |
| 2 | 0 | 0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 30.0% |
| 3 | 0 | 0.1 | 0.1 | 0 | 0.3 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.3 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 90.0% |
| 4 | 0 | 0 | 0 | 0.3 | 0 | 0.1 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.1 | 0.1 | 0.1 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 70.0% |
| 5 | 0.2 | 0 | 0 | 0 | 0.4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.4 | 0 | 0 | 0 | 0 | 60.0% |
| 6 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0.2 | 0 | 0 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50.0% |
| 7 | 0 | 0 | 0 | 0 | 0 | 0.3 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0.2 | 0 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 70.0% |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 9 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 10 | 0 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 50.0% |
| 11 | 0 | 0.4 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 80.0% |
| 12 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0.2 | 0 | 0 | 0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 30.0% |
| 13 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.3 | 0 | 0.1 | 0.2 | 0 | 0 | 0.1 | 0 | 0 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 70.0% |
| 14 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0.2 | 0 | 0.1 | 0.1 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 0 | 80.0% |
| 15 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.1 | 0.1 | 0.2 | 0 | 0.2 | 0 | 0 | 0.1 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 90.0% |
| 16 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.2 | 0.4 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 60.0% |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.2 | 0.1 | 0.5 | 0 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50.0% |
| 18 | 0.1 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0.4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 60.0% |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 20 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 50.0% |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.8 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 23 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 40.0% |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.1 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 30.0% |
| 25 | 0 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.4 | 0 | 0 | 0 | 0.2 | 0 | 0 | 0 | 60.0% |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| 27 | 0.1 | 0 | 0.1 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0.1 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.3 | 0 | 0 | 0 | 0 | 90.0% |
| 28 | 0 | 0 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0 | 0 | 0 | 0 | 50.0% |
| 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0.5 | 0 | 0.3 | 0 | 50.0% |
| 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.8 | 0 | 0.2 | 20.0% |
| 31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.7 | 0.2 | 30.0% |
| 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.3 | 0 | 0.7 | 30.0% |

TABLE XIV

GENERALIZATION ERRORS AND CLASSIFICATION MATRIX FOR DATA SET I023P891 (C4.5)

| **C4.5** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Average Error: | 17.8% | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Average Classification Matrix** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Actual | Classified as | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Misses |
| Class | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | in % |
| 1 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| 2 | 0 | 0.4 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 60.0% |
| 3 | 0.1 | 0.2 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 40.0% |
| 4 | 0 | 0 | 0 | 0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 5 | 0 | 0 | 0.2 | 0 | 0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 6 | 0 | 0 | 0 | 0 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 8 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 9 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.8 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.8 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.3 | 0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 30.0% |
| 12 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 13 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0.6 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 40.0% |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.8 | 0.1 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.3 | 0.3 | 0 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 70.0% |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.0% |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10.0% |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 30.0% |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.2 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 30.0% |
| 25 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.8 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 20.0% |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0% |
| 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0% |
| 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0% |
| 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.8 | 0 | 0.1 | 20.0% |
| 31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0.1 | 0 | 0.8 | 0 | 20.0% |
| 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0% |