A Robust Decision Tree Algorithm for Imbalanced Data Sets

Wei Liu∗and Sanjay Chawla∗
David A. Cieslak†and Nitesh V. Chawla†

Abstract

We propose a new decision tree algorithm, Class Confidence Proportion Decision Tree (CCPDT), which is robust and insensitive to class distribution and generates rules which are statistically significant.

In order to make decision trees robust, we begin by expressing information gain, the metric used in C4.5, in terms of confidence of a rule. This allows us to immediately explain why information gain, like confidence, results in rules which are biased towards the majority class. To overcome this bias, we introduce a new measure, Class Confidence Proportion (CCP), which forms the basis of CCPDT. To generate rules which are statistically significant we design a novel and efficient top-down and bottom-up approach which uses Fisher’s exact test to prune branches of the tree which are not statistically significant. Together these two changes yield a classifier that performs statistically better than not only traditional decision trees but also trees learned from data that has been balanced by well known sampling techniques. Our claims are confirmed through extensive experiments and comparisons against C4.5, CART, HDDT and SPARCCC.

1 Introduction

While there are several types of classifiers, rule-based classifiers have the distinct advantage of being easily interpretable. This is especially true in a “data mining” setting, where the high dimensionality of data often means that apriori very little is known about the underlying mechanism which generated the data.

Decision trees are perhaps the most popular form of rule-based classifiers (such as the well-known C4.5 [16]). Recently however, classifiers based on association rules have also become popular [20] which are often called associative classifiers. Associative classifiers use association rule mining to discover interesting and significant rules from the training data, and the set of rules discovered constitute the classifier. The canonical example of an associative classifier is CBA (classification based on associations) [15], which uses the minimum support and confidence framework to find rules. The accuracy of associative classifiers depends on the quality of their discovered rules. However, the success of both decision trees and associate classifiers depends on the assumption that there is an equal amount of information for each class contained in the training data. In binary classification problems, if there is a similar number of instances for both positive and negative classes, both C4.5 and CBA generally perform well. On the other hand, if the training data set tends to have an imbalanced class distribution, both types of classifier will have a bias towards the majority class. As it happens, an accurate prediction is typically related to the minority class – the class that is usually of greater interest.

One way of solving the imbalance class problem is to modify the class distributions in the training data by oversampling the minority class or under-sampling the majority class. For instance, SMOTE [6] uses oversampling to increase the number of the minority class instances, by creating synthetic samples. Further variations on SMOTE [8] have integrated boosting with sampling strategies to better model the minority class, by focusing on difficult samples that belong to both minority and majority classes.

Nonetheless, data sampling is not the only way to deal with class imbalanced problems: some specifically designed “imbalanced data oriented” algorithms can perform well on the original unmodified imbalanced data sets. For example, a variation on associative classifier called SPARCCC [20] has been shown to outperform CBA [15], CMAR [14] and CCCS [1] on imbalanced data sets. The downside of SPARCCC is that it generates a large number of rules. This seems to be a feature of all associative classifiers and negates many of the advantages of rule-based classification.

In [9], the Hellinger distance (HDDT) was used as the decision tree splitting criterion and shown to be insensitive towards class distribution skewness. We will compare and discuss CCPDT and HDDT more extensively in Section 3.4. Here it will be suffice to state that while HDDT is based on likelihood difference, CCPDT is based on likelihood ratio.

In order to prevent trees from over-fitting the data, all decision trees use some form of pruning. Traditional pruning algorithms are based on error estimations - a node is pruned if the predicted error rate is decreased. But this pruning technique will not always perform well on imbalanced data sets. [5] has shown that pruning in C4.5 can have a detrimental effect on learning from imbalanced data sets, since lower error rates can be achieved by removing the
Main Insight The main insight of the paper can be summarized as follows. Let $X$ be an attribute and $y$ a class. Let $X \rightarrow y$ and $\neg X \rightarrow y$ be two rules, with confidence $p$ and $q$ respectively. Then we can express information gain (IG) in terms of the two confidences. Abstractly,

$$IG_{C4.5} = F(p, q)$$

We will show that a splitting measure based on confidence will be biased towards the majority class. Our innovation is to use Class Confidence Proportion (CCP) instead of confidence. Abstractly CCP of the $X \rightarrow y$ and $\neg X \rightarrow y$ is $r$ and $s$. We define a new splitting criterion

$$IG_{CCPD} = F(r, s)$$

The main thrust in the paper is to show that $IG_{CCPD}$ is more robust to class imbalance than $IG_{C4.5}$ and behave similarly when the classes are balanced.

The approach of replacing a conventional splitting measure by CCP is a generic mechanism for all traditional decision trees that are based on the “balanced data” assumption. It can be applied to any decision tree algorithm that checks the degree of impurity inside a partitioned branch, such as C4.5 and CART etc.

The rest of the paper is structured as follows. In Section 2, we analyze the factors that causes CBA and C4.5 to perform poorly on imbalanced data sets. In Section 3, we introduce CCP as the measure for selecting attributes to split during decision tree construction. In Section 4 we present the full decision tree algorithm which details how we incorporate CCP and use Fisher’s exact test (FET) for pruning.. A wrapper framework utilizing sampling techniques is introduced in Section 5. Experiments, Results and Analysis are presented in Section 6. We conclude in Section 7 with directions for research.

2 Rule-based Classifiers

We analyze the metrics used by rule-based classifiers in the context of imbalanced data. We first show that the ranking of rules based on confidence is biased towards the majority class, and then express information gain and Gini index as functions of confidence and show that they also suffer from similar problems.

2.1 CBA The performance of Associative Classifiers depends on the quality of the rules it discovers during the training process. We now demonstrate that in an imbalanced setting, confidence is biased towards the majority class.

Suppose we have a training data set which consists of $n$ records, and the antecedents (denoted by $X$ and $\neg X$) and class ($y$ and $\neg y$) distributions are in the form of Table 1. The rule selection strategy in CBA is to find all rule items that have support and confidence above some predefined thresholds. For a rule $X \rightarrow y$, its confidence is defined as:

$$Conf(X \rightarrow y) = \frac{Supp(X \cup y)}{Supp(X)} = \frac{a}{a + c}$$

Similarly, we have:

$$Conf(X \rightarrow \neg y) = \frac{Supp(X \cup \neg y)}{Supp(X)} = \frac{c}{a + c}$$

Equation 2.1 suggests that selecting the highest confidence rules means choosing the most frequent class among all the instances that contains that antecedent (i.e. $X$ in this example). However, for imbalanced data sets, since the size of the positive class is always much smaller than the negative class, we always have: $a + b \ll c + d$ (suppose $y$ is the positive class). Given that imbalanced data do not affect the distribution of antecedents, we can, without loss of generality, assume that $X$’s and $\neg X$’s are nearly equally distributed. Hence when data is imbalanced, $a$ and $b$ are both small while $c$ and $d$ are both large. Even if $y$ is supposed to occur with $X$ more frequently than $\neg y$, $c$ is unlikely to be less than $a$ because the positive class size will be much smaller than the negative class size. Thus, it is not surprising that the right-side term in Equation 2.2 always tends to be lower bounded by the right-side term in Equation 2.1. It appears that even though the rule $X \rightarrow \neg y$ may not be significant, it is easy for it to have a high confidence value.

In these circumstances, it is very hard for the confidence of a “good” rule $X \rightarrow y$ to be significantly larger than that of a “bad” rule $X \rightarrow \neg y$. What is more, because of its low confidence, during the classifier building process a “good” rule may be ranked behind some other rules just because they have a higher confidence because they predict the majority class. This is a fatal error, since in an imbalanced class problem it is often the minority class that is of more interest.

2.2 Traditional decision trees Decision trees such as C4.5 use information gain to decide which variable to split
The information gain from splitting a node $t$ is defined as:

$$\text{InfoGain}_{\text{split}} = \text{Entropy}(t) - \sum_{i=1,2} \frac{n_i}{n} \text{Entropy}(i)$$

where $i$ represents one of the sub-nodes after splitting (assume there are 2 sub-nodes), $n_i$ is the number of instances in subnote $i$, and $n$ stands for the total number of instances. In binary-class classification, the entropy of node $t$ is defined as:

$$\text{Entropy}(t) = -\sum_{j=1,2} p(j|t) \log p(j|t)$$

where $j$ represents one of the two classes. For a fixed training set (or its subsets), the first term in Equation 2.3 is fixed, because the number of instances for each class (i.e. $p(j|t)$ in equation 2.4) is the same for all attributes. To this end, the challenge of maximizing information gain in Equation 2.3 reduces to maximizing the second term $- \sum_{i=1,2} \frac{n_i}{n} \text{Entropy}(i)$.

If the node $t$ is split into two subnodes with two corresponding paths: $X$ and $\neg X$, and the instances in each node have two classes denoted by $y$ and $\neg y$, Equation 2.3 can be rewritten as:

$$\text{InfoGain}_{\text{split}} = \text{Entropy}(t) - \frac{n_1}{n} [-p(y|X) \log(y|X) - p(\neg y|X) \log p(\neg y|X)] - \frac{n_2}{n} [-p(y|\neg X) \log(y|\neg X) - p(\neg y|\neg X) \log p(\neg y|\neg X)]$$

Note that the probability of $y$ given $X$ is equivalent to the confidence of $X \rightarrow y$:

$$p(y|X) = \frac{\text{Support}(X \cup y)}{\text{Support}(X)} = \text{Conf}(X \rightarrow y)$$

Then if we denote $\text{Conf}(X \rightarrow y)$ by $p$, and denote $\text{Conf}(\neg X \rightarrow y)$ by $q$ (hence $\text{Conf}(X \rightarrow \neg y) = 1 - p$ and $\text{Conf}(\neg X \rightarrow \neg y) = 1 - q$), and ignore the “fixed” terms $\text{Entropy}(t)$ in equation 2.5, we can obtain the relationship in Equation 2.7.

The first approximation step in Equation 2.7 ignores the first term $\text{Entropy}(t)$, then the second approximation transforms the addition of logarithms to multiplications.

Based on Equation 2.7 the distribution of information gain as a function of $\text{Conf}(X \rightarrow y)$ and $\text{Conf}(\neg X \rightarrow y)$ is shown in Figure 1. Information gain is maximized when $\text{Conf}(X \rightarrow y)$ and $\text{Conf}(\neg X \rightarrow y)$ are both close to either 0 or 1, and is minimized when $\text{Conf}(X \rightarrow y)$ and $\text{Conf}(\neg X \rightarrow y)$ are both close to 0.5. Note that when $\text{Conf}(X \rightarrow y)$ is close to 0, $\text{Conf}(X \rightarrow \neg y)$ is close to 1; when $\text{Conf}(\neg X \rightarrow y)$ is close to 0, $\text{Conf}(\neg X \rightarrow \neg y)$ is close to 1. Therefore, information gain achieves the highest value when either $X \rightarrow y$ or $X \rightarrow \neg y$ has the highest confidence, and either $\neg X \rightarrow y$ or $\neg X \rightarrow \neg y$ also has the highest confidence.

$$\text{InfoGain}_{\text{split}} = \text{Entropy}(t) + \sum_{i=1,2} \frac{n_i}{n} \text{Entropy}(i)$$

$$= \text{Entropy}(t) + \frac{n_1}{n} [p \log p + (1 - p) \log(1 - p)] + \frac{n_2}{n} [q \log q + (1 - q) \log(1 - q)]$$

$$\propto \frac{n_1}{n} [p \log p + (1 - p) \log(1 - p)] + \frac{n_2}{n} [q \log q + (1 - q) \log(1 - q)]$$

$$\propto \frac{n_1}{n} \log p^p (1 - p)^{1-p} + \frac{n_2}{n} \log q^q (1 - q)^{1-q}$$

Therefore, decision trees such as C4.5 split an attribute whose partition provides the highest confidence. This strategy is very similar to the rule-ranking mechanism of association classifiers. As we have analyzed in Section 2.1, for imbalanced data set, high confidence rules do not necessarily imply high significance in imbalanced data, and some significant rules may not yield high confidence. Thus we can assert that the splitting criteria in C4.5 is suitable for balanced but not imbalanced data sets.

We note that it is the term $p(j|t)$ in Equation 2.4 that is the cause of the poor behavior of C4.5 in imbalanced situations. However, $p(j|t)$ also appears in other decision tree measures. For example, the Gini index defined in

![Figure 1: Approximation of information gain in the formula formed by $\text{Conf}(X \rightarrow y)$ and $\text{Conf}(\neg X \rightarrow y)$ from Equation 2.7. Information gain is the lowest when $\text{Conf}(X \rightarrow y)$ and $\text{Conf}(\neg X \rightarrow y)$ are both close to 0.5, and is the highest when both $\text{Conf}(X \rightarrow y)$ and $\text{Conf}(\neg X \rightarrow y)$ reaches 1 or 0.](image-url)
Table 2: Confusion Matrix for the classification of two classes

<table>
<thead>
<tr>
<th>Actual positive</th>
<th>Predicted positive</th>
<th>Predicted negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>True positive (tp)</td>
<td>False positive (fp)</td>
<td>True negative (tn)</td>
</tr>
<tr>
<td>False negative (fn)</td>
<td>True negative (tn)</td>
<td>True negative (tn)</td>
</tr>
</tbody>
</table>

CART[3] can be expressed as:

\[ Gini(t) = 1 - \sum_j p(j|t)^2 \]  

This decision tree based on CART will too suffer from the imbalanced class problem. We now propose another measure which will be more robust in the imbalanced data situation.

3 Class Confidence Proportion and Fisher’s exact test

Having identified the weakness of the support-confidence framework and the factor that results in the poor performance of entropy and Gini index, we are now in a position to propose new measures to address the problem.

3.1 Class Confidence Proportion

As previously explained, the high frequency with which a particular class \( y \) appears together with \( X \) does not necessarily mean that \( X \) “explains” the class \( y \), because \( y \) could be the overwhelming majority class. In such cases, it is reasonable that instead of focusing on the antecedents (\( X \)s), we focus only on each class and find the most significant antecedents associated with that class. In this way, all instances are partitioned according to the class they contain, and consequently, instances that belong to different classes will not have an impact on each other. To this end, we define a new concept, Class Confidence (CC), to find the most interesting antecedents (\( X \)s) from all the classes (\( y \)s):

\[ CC(X \rightarrow y) = \frac{\text{Supp}(X \cup y)}{\text{Supp}(y)} \]  

The main difference between this CC and traditional confidence is the denominator: we use \( \text{Supp}(y) \) instead of \( \text{Supp}(X) \) so as to focus only on each class.

In the notation of the confusion matrix (Table 2) CC can be expressed as:

\[ CC(X \rightarrow y) = \frac{\text{TruePositiveInstances}}{\text{ActualPositiveInstances}} = \frac{tp}{tp + fn} \]  

\[ CC(X \rightarrow \neg y) = \frac{\text{FalsePositiveInstances}}{\text{ActualNegativeInstances}} = \frac{fp}{fp + tn} \]

While traditional confidence examines how many predicted positive/negative instances are actually positive/negative (the precision), CC is focused in how many actual positive/negative instances are predicted correctly (the recall). Thus, even if there are many more negative than positive instances in the data set \( (tp + fn \ll fp + tn) \), Equations 3.10 and 3.11 will not be affected by this imbalance. Consequently, rules with high CC will be the significant ones, regardless of whether they are discovered from balanced or imbalanced data sets.

However, obtaining high CC rules is still insufficient for solving classification problems – it is necessary to ensure that the classes implied by those rules are not only of high confidence, but more interesting than their corresponding alternative classes. Therefore, we propose the proportion of one CC over that of all classes as our measure of how interesting the class is – what we call the CC Proportion (CCP). The CCP of rule \( X \rightarrow y \) is defined as:

\[ CCP(X \rightarrow y) = \frac{CC(X \rightarrow y)}{CC(X \rightarrow y) + CC(X \rightarrow \neg y)} \]

A rule with high CCP means that, compared with its alternative class, the class this rule implies has higher CC, and consequently is more likely to occur together with this rule’s antecedents regardless of the proportion of classes in the data set. Another benefit of taking this proportion is the ability to scale CCP between \([0,1]\), which makes it possible to replace the traditional frequency term in entropy (the factor \( p(j|t) \) in Equation 2.4) by CCP. Details of the CCP replacement in entropy is introduced in Section 4.

3.2 Robustness of CCP

We now evaluate the robustness of CCP using ROC-based isometric plots proposed in Flach [11] and which are inherently independent of class and misclassification costs.

The 2D ROC space is spanned by false positive rate (x-axis) and the true positive rate (y-axis). The contours mark out the lines of constant value, of the splitting criterion, conditioned on the imbalanced class ratio. Metrics which are robust to class imbalance should have similar contour plots.
for different class ratios.

In Figure 2, the contour plots of information gain are shown for class ratios of 1:1 and 1:10, respectively. It is clear, from the two figures, that when the class distributions become more imbalanced, the contours tend to be flatter and further away from the diagonal. Thus, given the same true positive rate and false positive rate, information gain for imbalanced data sets (Figure 2b) will be much lower than for balanced data sets (Figure 2a).

Following the model of relative impurity proposed Flach in [11], we now derive the definition for the CCP Impurity Measure. Equation 3.12 gives:

\[
\text{CCP}(X \rightarrow y) = \frac{tp}{tp + fn} + \frac{fp}{fp + tn} = \frac{tpr}{tpr + fpr}
\]

where \(tpr/fpr\) represents true/false positive rate. For each node-split in tree construction, at least two paths will be generated, if one is \(X \rightarrow y\), the other one will be \(\neg X \rightarrow \neg y\) with CCP:

\[
\text{CCP}(\neg X \rightarrow \neg y) = \frac{tn}{tp + fn} + \frac{fn}{tp + fn} = \frac{1}{2 - tpr - fpr}
\]

The relative impurity for C4.5 proposed in [11] is:

\[
\text{InfoGain}_{C4.5} = \text{Imp}(tp + fn, fp + tn) - (tp + fp) * \text{Imp}(\frac{tp}{tp + fp}, \frac{fp}{tp + fp}) - (fn + tn) * \text{Imp}(\frac{fn}{fn + tn}, \frac{tn}{fn + tn})
\]

where \(\text{Imp}(p,n) = -p \log_2 p = -n \log_2 n\). The first term in the right side represents the entropy of the node before splitting, while the sum of the second and third terms represents the entropy of the two subnodes after splitting. Take the second term \((tp + fp) * \text{Imp}(\frac{tp}{tp + fp}, \frac{fp}{tp + fp})\) as an example, the first frequency measure \(\frac{tp}{tp + fp}\) is an alternative way of interpreting the confidence of rule \(X \rightarrow y\); similarly, the second frequency measure \(\frac{fp}{tp + fp}\) is equal to the confidence of rule \(X \rightarrow \neg y\). We showed in Section 2.2 that both terms are inappropriate for imbalanced data learning.

To overcome the inherent weakness in traditional decision trees, we apply CCP into this impurity measure and thus rewrite the information gain definition in Equation 3.15 as the CCP Impurity Measure:

\[
\text{InfoGain}_{CCP} = \text{Imp}(tp + fn, fp + tn) - (tp + fp) * \text{Imp}(\frac{tp}{tp + fp}, \frac{fp}{tp + fp}) - (2 - tpr - fpr) * \text{Imp}(\frac{1 - tpr}{2 - tpr - fpr}, \frac{1 - fpr}{2 - tpr - fpr})
\]

where \(\text{Imp}(p,n)\) is still \(-p \log_2 p\), while the original frequency term is replaced by CCP.

The new isometric plots, with the CCP replacement, are presented in Figure 3 (a,b). A comparison of the two figures clearly shows that contour lines remain unchanged. This clearly demonstrates that that CCP is unaffected by the changes in the class ratio.

### 3.3 Properties of CCP

If all instances contained in a node belong to the same class, its entropy is minimized (zero). The entropy is maximized when a node contains equal number of elements from both classes.

By taking all possible combinations of elements in the confusion matrix (Table 2), we can plot the entropy surface as a function of \(tpr\) and \(fpr\) as shown in Figure 4. Entropy (Figure 4a) is the highest when \(tpr\) and \(fpr\) are equal, since “\(tpr = fpr\)” in subnodes is equivalent to elements in the subnodes being equally split between the two classes. On the other hand, the larger the difference between \(tpr\) and \(fpr\), the purer the subnodes and the smaller their entropy. However, as stated in Section 3.2, when data sets are imbalanced, the pattern of traditional entropy will become distorted (Figure 4b).

Since CCP-embedded “entropy” is insensitive to class skewness, its will always exhibit a fixed pattern, and this pattern is the same as traditional entropy’s balanced data situation. This can be formalized as follows:

By using the notations in the confusion matrix, the frequency term in traditional entropy is \(p_{\text{traditional}} = \frac{tp}{tp + fp}\), while in CCP-based entropy it is \(p_{CCP} = \frac{tpr}{tpr + fpr}\). When
classes in a data set are evenly distributed, we have $tp + fn = fp + tn$, and by applying it in the definition of CCP we obtain:

$$p_{CCP} = \frac{tpr}{tpr + fpr} = \frac{\frac{tp}{tp + fn}}{\frac{tp}{tp + fn} + \frac{fp}{fp + tn}} = \frac{tp}{tp + fp}$$

Thus when there are same number of instances in each class, the patterns of CCP-embedded entropy and traditional entropy will be the same. More importantly, this pattern is preserved for CCP-embedded entropy independent of the imbalance in the data sets. This pattern is confirmed in Figure 4c and which is always similar to that of entropy Figure 4a regardless of the class distributions.

### 3.4 Hellinger Distance and its relationship with CCP

The divergence of two absolutely continuous distributions can be measured by Hellinger distance with respect to the parameter $\lambda$ [18, 12], in the form of:

$$d_H(P, Q) = \sqrt{\int_{\Omega} (\sqrt{P} - \sqrt{Q})^2 d\lambda}$$

In the Hellinger distance based decision tree (HDDT) technique [9], the distribution $P$ and $Q$ are assumed to be the normalized frequencies of feature values (“X” in our notation) across classes. The Hellinger distance is used to capture the propensity of a feature to separate the classes. In the tree-construction algorithm in HDDT, a feature is selected as a splitting attribute when it produces the largest Hellinger distance between the two classes. This distance is essentially captured in the differences in the relative frequencies of the attribute values for the two classes, respectively.

The following formula, derived in [9], relates HDDT with the true positive rate ($tpr$) and false positive rate ($fpr$).

$$\text{Impurity}_{HD} = \sqrt{(\sqrt{tpr} - \sqrt{fpr})^2 + (\sqrt{1-tpr} - \sqrt{1-fpr})^2}$$

This was also shown to be insensitive to class distributions in [9], since the only two variables in this formula are $tpr$ and $fpr$, without the dominating class priors.

Like the Hellinger distance, CCP is also just based on $tpr$ and $fpr$ as shown in Equation 3.13. However, there is a significant difference between CCP and Hellinger distance. While Hellinger distance take the square root difference of $tpr$ and $fpr$ ($\sqrt{tpr} - \sqrt{fpr}$) as the divergence of one class distribution from the other, CCP takes the proportion of $tpr$ and $fpr$ as a measurement of interest. A graphical difference between the two measures is shown in Figure 5.

If we draw a straight line (Line 3) parallel to the diagonal...
in Figure 5, the segment length from origin to cross-point 
between Line 3 and the y-axis is \( |tpr_o - fpr_o| \) (\( tpr_o \) 
and \( fpr_o \) can be the coordinates of any point in Line 3), 
is proportional to the Hellinger distance \( \sqrt{|tpr - \sqrt{fpr}|} \). 
From this point of view, HDDT selects the point on those 
parallel lines with the longest segment. Therefore, in Figure 
5, all the points in Line 3 have a larger Hellinger distance 
than those in Line 4; thus points in Line 3 will have higher 
priority in the selection of attributes. As \( CCP = \frac{tpr}{tpr+fpr} \) 
can be rewritten as \( tpr = \frac{CCP}{1-CCP} fpr \), CCP is proportional 
to the the slope of the line formed by the data point and 
the origin, and consequently favors the line with the highest 
slope. In Figure 5, the points in Line 1 are considered by 
CCP as better splitting attributes than those in Line 2.

By analyzing CCP and Hellinger distances in terms of 
lines in a \( tpr \text{ versus } fpr \) reference frame, we note that CCP 
and Hellinger distance share a common problem. We give 
an example as follows. Suppose we have three points, A 
\( (fpr_A, tpr_A) \), B \( (fpr_B, tpr_B) \) and C \( (fpr_C, tpr_C) \), where 
A is one Line 1 and 3, B on Line 2 and 3, and C on 
Line 2 and 4 (shown in Figure 5). Then A and B are 
on the same line (Line 3) that is parallel to the diagonal 
(i.e., \( |fpr_A - tpr_A| = |fpr_B - tpr_B| \)), while B and 
C are on the same line (Line 2) passing through the origin 
(i.e., \( \frac{tpr_B}{fpr_B} = \frac{tpr_C}{fpr_C} \)). Hellinger distances will treat A and 
B as better splitting attributes than C, because as explained 
above all points in Line 3 has longer Hellinger distances 
than Line 4. By contrast, CCP will consider A has higher 
splitting priorities than both B and C, since all points in 
Line 1 obtains greater CCP than Line 2. However, on 
points in Line 3 such as A and B, Hellinger distance fails 
to distinguish them, since they will generate the same \( tpr \) 
vs. \( fpr \) difference. In this circumstance, HDDT may make 
an noneffective decision in attribute selection. This problem 
will become significant when the number of attributes is 
large, and many attributes have similar \( |tpr - fpr| \) (or more 
precisely \( \sqrt{tpr} - \sqrt{fpr} \)) difference. The same problem 
occurs in the CCP measurement on testing points in Line 2 
such as B against C.

Our solution to this problem is straightforward: when 
choosing the splitting attribute in decision tree construction, 
we select the one with the highest CCP by default, and if 
there are attributes that possess similar CCP values, we 
prioritize them on the basis of their Hellinger distances. 
Thus, in Figure 5, the priority of the three points will be 
\( A > B > C \), since Point A has a greater CCP value than Points 
B and C, and Point B has higher Hellinger distance than 
Point C. Details of these attribute-selecting algorithms are 
in Section 4.

3.5 Fisher’s exact test While CCP helps to select which 
branch of a tree are “good” to discriminate between classes, 
we also want to evaluate the statistical significance of each 
branch. This is done by the Fisher’s exact test (FET). For a 
rule \( X \rightarrow y \), the FET will find the probability of obtaining 
the contingency table where \( X \) and \( y \) are more positively 
associated, under the null hypothesis that \( \{X, \neg X\} \) and 
\( \{y, \neg y\} \) are independent [20]. The \( p \) value of this rule is 
given by:

\[
p(a, b, c, d) = \sum_{i=0}^{min(b,c)} \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!(a+i)!(b-i)!(c-i)!(d+i)!}
\]

During implementation, the factorials in the \( p \)-value def-
inition can be handled by expressing their values logarithmi-
cally.

A low \( p \) value means that the variable independence null 
hypothesis is rejected (no relationship between \( X \) and \( y \)); 
in other words, there is a positive association between the 
upper-left cell in the contingency table (true positives) and 
the lower-right (true negatives). Therefore, given a threshold 
for the \( p \) value, we can find and keep the tree branches that 
are statistically significant (with lower \( p \) values), and discard 
those tree nodes that are not.

4 CCP-based decision trees (CCPDT)

In this section we provide details of the CCPDT algorithm. 
We modify the C4.5 splitting criterion based on entropy and 
replace the frequency term by CCP. Due to space limits, 
we omit the algorithms for CCP-embedded CART, but the 
approach is identical to C4.5 (in that the same factor is 
replaced with CCP).

Algorithm 1 (CCP-C4.5) Creation of CCP-based C4.5

Input: Training Data: \( TD \)
Output: Decision Tree
1: if All instances are in the same class then
2: Return decision tree with one node (root), labeled as the 
instances’ class,
3: else
4: // Find the best splitting attribute (\( AttrI \)),
5: \( AttrI = \max \{ \text{CCPGain}(TD), \}
6: Assign \( AttrI \) to the tree root (root = \( AttrI \)),
7: for each value \( v_i \) of \( AttrI \) do
8: Add a branch for \( v_i \),
9: if No instance is \( v_i \) at attribute \( AttrI \) then
10: Add a leaf to this branch.
11: else
12: Add a subtree \( CCP - C4.5(TD_{v_i}) \) to this branch,
13: end if
14: end for
15: end if

4.1 Build tree The original definition of entropy in deci-
sion trees is presented in Equation 2.4. As explained in Sec-
Algorithm 2 (MaxCCPGain) Subroutine for discovering the attribute with the greatest information gain

**Input:** Training Data: $TD$
**Output:** The attribute to be split: $Attri$

1. Let $MaxHellinger$, $MaxInfoGain$ and $Attri$ to 0;
2. for Each attribute $A_j$ in $TD$ do
3. Calculate $Hellinger$ distance: $A_j.Hellinger$, $Hellinger$ before splitting: $A_j.oldEnt$, $Hellinger$ after splitting: $A_j.newEnt$ to 0,
4. for Each value $V^j_i$ of attribute $A_j$ do
5. // $|T_{x,y}|$ means the number of instance that have value $x$ and class $y$,
6. $tpr = \frac{T_{x=V^j_i,y=+}}{T_{x=V^j_i,y=+} + T_{x=V^j_i,y=+}}$;
7. $fpr = \frac{T_{x=V^j_i,y=+}}{T_{x=V^j_i,y=+} + T_{x=V^j_i,y=+}}$;
8. $A_j.newEnt + = T_{x=V^j_i,v=+} \times \left( - \frac{tpr}{tpr+fpr} \log_2 \frac{tpr}{tpr+fpr} - \frac{fpr}{tpr+fpr} \log_2 \frac{fpr}{tpr+fpr} \right)$;
9. end for
10. $CurrentInfoGain = A_j.oldEnt - A_j.newEnt$;
11. if $MaxInfoGain < CurrentInfoGain$ then
12. $Attri = j$;
13. $MaxInfoGain = CurrentInfoGain$;
14. $MaxHellinger = A_j.Hellinger$;
15. else
16. if $MaxHellinger < A_j.Hellinger$ AND $MaxInfoGain == CurrentInfoGain$ then
17. $Attri = j$;
18. $MaxHellinger = A_j.Hellinger$;
19. end if
20. end if
21. end if
22. end for
23. Return $Attri$.

Algorithm 3 (Prune) Pruning based on FET

**Input:** Unpruned decision tree $DT$, $p$-value threshold ($pVT$)
**Output:** Pruned $DT$

1. for Each leaf $Leaf_i$ do
2. if $Leaf_i.parent$ is not the Root of $DT$ then
3. $Leaf_i.parent.pruneable = true$, // ‘true’ is default
4. SetPruneable($Leaf_i.parent$, $pVT$),
5. end if
6. end for
7. Obtain the root of $DT$,
8. for Each $child(i)$ of the root do
9. if $child(i)$ is not a leaf then
10. if $child(i).pruneable == true$ then
11. Set $child(i)$ to be a leaf,
12. else
13. PrunebyStatus($child(i)$),
14. end if
15. end if
16. end for

Algorithm 4 (SetPruneable) Subroutine for setting pruneable status to each branch node by a bottom–up search

**Input:** A branch node $Node$, $p$-Value threshold ($pVT$)
**Output:** Pruneable status of this branch node

1. for each $child(i)$ if $Node$ do
2. if $child(i).pruneable == false$ then
3. $Node.pruneable = false$,
4. end if
5. end for
6. if $Node.pruneable == true$ then
7. Calculate the $p$ value of this node: $Node.pValue$,
8. if $Node.pValue < pVT$ then
9. $Node.pruneable = false$,
10. end if
11. end if
12. if $Node.parent$ is not the Root of the full tree then
13. $Node.parent.pruneable = true$ // ‘true’ is default
14. SetPruneable($Node.parent$, $pVT$),
15. end if

Algorithm 5 (PrunebyStatus) Subroutine for pruning nodes by their pruneable status

**Input:** A branch represented by its top node $Node$
**Output:** Pruned branch

1. if $Node.pruneable == true$ then
2. Set $Node$ as a leaf,
3. else
4. for Each $child(i)$ of $Node$ do
5. if $child(i)$ is not a leaf then
6. PrunebyStatus($child(i)$),
7. end if
8. end for
9. end if

In Algorithm 2, the factor $p(j|t)$ in Equation 2.4 is not a good criterion for learning from imbalanced data sets, so we replace it with CCP and define the CCP-embedded entropy as:

$$\text{Entropy}_{\text{CCP}}(t) = - \sum_j \text{CCP}(X \rightarrow y_j) \log_2 \text{CCP}(X \rightarrow y_j)$$

Then we can restate the conclusion made in Section 2.2 as: in CCP-based decision trees, $IG_{\text{CCP DT}}$ achieves the highest value when either $X \rightarrow y$ or $X \rightarrow \neg y$ has high CCP, and either $\neg X \rightarrow y$ or $\neg X \rightarrow \neg y$ has high CCP.

The process of creating CCP-based C4.5 (CCP-C4.5) is described in Algorithm 1. The major difference between CCP-C4.5 and C4.5 is the the way of selecting the candidate-splitting attribute (Line 5). The process of discovering the attribute with the highest information gain is presented in the subroutine Algorithm 2. In Algorithm 2, Line 4 obtains the entropy of an attribute before its splitting, Lines 6 – 11 obtain the new CCP-based entropy after the splitting of that attribute, and Lines 13 – 22 record the attribute with the highest information gain.
highest information gain. In information gain comparisons of different attributes, Hellinger distance is used to select the attribute whenever InfoGain value of the two attributes are equal (Lines 18–21), thus overcoming the inherent drawback of both Hellinger distances and CCP (Section 3.4).

In our decision tree model, we treat each branch node as the last antecedent of a rule. For example, if there are three branch nodes (BranchA, BranchB, and BranchC) from the root to a leaf LeafY, we assume that the following rule exists: \( \text{BranchA} \wedge \text{BranchB} \wedge \text{BranchC} \rightarrow \text{LeafY} \). In Algorithm 2, we calculate the CCP for each branch node; using the preceding example, the CCP of BranchC is that of the previous rule, and the CCP of BranchB is that of rule \( \text{BranchA} \wedge \text{BranchB} \rightarrow \text{LeafY} \), etc. In this way, the attribute we select is guaranteed to be the one whose split can generate rules (paths in the tree) with the highest CCP.

4.2 Prune tree After the creation of decision tree, Fisher’s exact test is applied on each branch node. A branch node will not be replaced by a leaf node if there is at least one significant descendant (a node with a lower \( p \) value than the threshold) under that branch.

Checking the significance of all descendants of an entire branch is an expensive operation. To perform a more efficient pruning, we designed a two-staged strategy as shown in Algorithm 3. The first stage is a bottom–up checking process from each leaf to the root. A node is marked “pruneable” if it and all of its descendants are non-significant. This process of checking the pruning status is done via Lines 1–6 in Algorithm 3, with subroutine Algorithm 4. In the beginning, all branch nodes are set to the default status of “pruneable” (Line 3 in Algorithm 3 and Line 13 in Algorithm 4). We check the significance of each node from leaves to the root. If any child of a node is “unpruneable” or the node itself represents a significant rule, this node will be reset from “pruneable” to “unpruneable”. If the original unpruned tree is \( n \) levels deep, and has \( m \) leaves, the time complexity of this bottom–up checking process is \( O(nm) \).

After checking for significance, we conduct the second pruning stage – a top-down pruning process performed according to the “pruneable” status of each node from root to leaves. A branch node is replaced by a leaf, if its “pruneable” status is “true” (Line 10 in Algorithm 3 and Line 1 in Algorithm 5).

Again, if the original unpruned tree is \( n \) levels deep and has \( m \) leaves, the time complexity of the top–down pruning process is \( O(nm) \). Thus, the total time complexity of our pruning algorithm is \( O(n^2) \).

This two-stage pruning strategy guarantees both the completeness and the correctness of the pruned tree. The first stage checks the significance of each possible rule (path through the tree), and ensures that each significant rule is “unpruneable”, and thus complete; the second stage prunes all insignificant rules, so that the paths in the pruned tree are all correct.

5 Sampling Methods

Another mechanism of overcoming the imbalance class distribution is to synthetically delete or add training instances and thus balance the class distribution. To achieve this goal, various sampling techniques have been proposed to either remove instances from the majority class (aka under-sampling) or introduce new instances to the minority class (aka over-sampling).

We consider a wrapper framework that uses a combination of random under-sampling and SMOTE [6, 7]. The wrapper first determines the percentage of under-sampling that will result in an improvement in AUC over the decision tree trained on the original data. Then the number of instances in majority class is under-sampled to the stage where the AUC does not improve any more, the wrapper explores the appropriate level of SMOTE. Then taking the level of under-sampling into account, SMOTE introduces new synthetic examples to the minority class continuously until the AUC is optimized again. We point the reader to [7] for more details on the wrapper framework. The performance of decision trees trained on the data sets optimized by this wrapper framework is evaluated against CCP-based decision trees in experiments.

6 Experiments

In our experiments, we compared CCRDT with C4.5 [16], CART[3], HDDT [9] and SPARCCC [20] on binary class data sets. These comparisons demonstrate not only the efficiency of their splitting criteria, but the performance of their pruning strategies.

Weka’s C4.5 and CART implementations [21] were employed in our experiments, based on which we implemented CCP-C4.5, CCP-CART, HDDT and SPARCCC so that we can normalize the effects of different versions of the implementations.

All experiments were carried out using 5 \( \times \) 2 folds cross-validation, and the final results were averaged over the five runs. We first compare purely on splitting criteria without applying any pruning techniques, and then comparisons between pruning methods on various decision trees are presented. Finally, we compare CCP-based decision trees with state-of-the-art sampling methods.

6.1 Comparisons on splitting criteria The binary-class data sets were mostly obtained from [9] (Table 3) which were pre-discretized. They include a number of real-world data sets from the UCI repository and other sources. “Estate”

Table 3: Information about imbalanced binary-class data sets. The percentages listed in the last column is the proportion of the minor class in each data set.

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Instances</th>
<th>Attributes</th>
<th>Min.Class %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary</td>
<td>3505</td>
<td>175</td>
<td>3.8%</td>
</tr>
<tr>
<td>Breast</td>
<td>569</td>
<td>30</td>
<td>37.3%</td>
</tr>
<tr>
<td>Cam</td>
<td>28374</td>
<td>132</td>
<td>5.0%</td>
</tr>
<tr>
<td>Covtype</td>
<td>38500</td>
<td>10</td>
<td>7.1%</td>
</tr>
<tr>
<td>Estate</td>
<td>5322</td>
<td>12</td>
<td>12.0%</td>
</tr>
<tr>
<td>Fourclass</td>
<td>862</td>
<td>2</td>
<td>35.6%</td>
</tr>
<tr>
<td>German</td>
<td>1000</td>
<td>24</td>
<td>30.0%</td>
</tr>
<tr>
<td>Ism</td>
<td>11180</td>
<td>6</td>
<td>2.3%</td>
</tr>
<tr>
<td>Letter</td>
<td>20000</td>
<td>16</td>
<td>3.9%</td>
</tr>
<tr>
<td>Oil</td>
<td>937</td>
<td>50</td>
<td>4.4%</td>
</tr>
<tr>
<td>Page</td>
<td>5473</td>
<td>10</td>
<td>10.2%</td>
</tr>
<tr>
<td>Pendigits</td>
<td>10992</td>
<td>16</td>
<td>10.4%</td>
</tr>
<tr>
<td>Phoneme</td>
<td>2700</td>
<td>5</td>
<td>29.3%</td>
</tr>
<tr>
<td>PhosS</td>
<td>11411</td>
<td>481</td>
<td>5.4%</td>
</tr>
<tr>
<td>Pima</td>
<td>768</td>
<td>8</td>
<td>34.9%</td>
</tr>
<tr>
<td>Satimage</td>
<td>6430</td>
<td>37</td>
<td>9.7%</td>
</tr>
<tr>
<td>Segment</td>
<td>2310</td>
<td>20</td>
<td>14.3%</td>
</tr>
<tr>
<td>Splice</td>
<td>1000</td>
<td>60</td>
<td>48.3%</td>
</tr>
<tr>
<td>SVMguide</td>
<td>3089</td>
<td>4</td>
<td>35.3%</td>
</tr>
</tbody>
</table>

contains electrotopological state descriptors for a series of compounds from the US National Cancer Institute’s Yeast Anti-Cancer drug screen. “Ism” (f6) is highly unbalanced and records information on calcification in a mammogram. “Oil” contains information about oil spills; it is relatively small and very noisy [13]. “Phoneme” originates from the ELENA project and is used to distinguish between nasal and oral sounds. “Boundary”, “Cam”, and “PhosS” are biological data sets from [17]. “FourClass”, “German”, “Splice”, and “SVMGuide” are available from LIBSVM [4]. The remaining data sets originate from the UCI repository [2]. Some were originally multi-class data sets, and we converted them into two-class problems by keeping the smallest class as the minority and the rest as the majority. The exception was “Letter”, for which each vowel became a member of the minority class, against the consonants as the majority class.

As accuracy is considered a poor performance measure for imbalanced data sets, we used the area under ROC curve (AUC) [19] to estimate the performance of each classifier.

In our imbalanced data sets learning experiments, we only wanted to compare the effects of different splitting criteria; thus, the decision trees (C4.5, CCP-C4.5, CART, CCP-CART, and HDDT) were unpruned, and we used Laplace smoothing on leaves. Since SPARCCC has been proved more effective in imbalanced data learning than CBA [20], we excluded CBA and included only SPARCCC. Table 4 lists the “Area Under ROC (AUC)” value on each data set for each classifier, followed by the ranking of these classifiers (presented in parentheses) on each data set.

We used the Friedman test on AUCs at 95% confidence level to compare the different classifiers [10]. In all experiments, we chose the best performance classifier as the “Base” classifier. If the “Base” classifier is statistically significantly better than another classifier in comparison (i.e. the value of Friedman test is less than 0.05), we put a “✓” sign on the respective classifier. (shown in Table 4).

The comparisons revealed that even though SPARCCC performs better than CBA [20], its overall results are far less robust than those from decision trees. It might be possible to obtain better SPARCCC results by repeatedly modifying their parameters and attempting to identify the optimized parameters, but the manual parameter configuration itself is a shortcoming for SPARCCC.

Because we are interested in the replacement of original factor $p(j\mid t)$ in Equation 2.4 by CCP, three separate Friedman tests were carried out: the first two between conventional decision trees (C4.5/CART) and our proposed decision trees (CCP-C4.5/CCP-CART), and the third on all the other classifiers A considerable AUC increase from traditional to CCP-based decision trees was observed, and statistically confirmed by the first two Friedman tests: these small $p$ values of 9.6E-5 meant that we could confidently reject the hypothesis that the “Base” classifier showed no significant differences from current classifier. Even though CCP-C4.5 was not statistically better than HDDT, the strategy to combine CCP and HDDT (a analyzed in Section 3.4) provided an improvement on AUC with higher than 91% confidence.

Table 4: Splitting criteria comparisons on imbalanced data sets where all trees are unpruned. “Fr.T” is short for Friedman test. The classifier with a ✓ sign in the Friedman test is statistically outperformed by the “Base” classifier. The first two Friedman tests illustrate that CCP-C4.5 and CCP-CART are significantly better than C4.5 and CART, respectively. The third Friedman test confirms that CCP-based decision trees is significantly better than SPARCCC.

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Area Under ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C4.5</td>
</tr>
<tr>
<td>Boundary</td>
<td>0.933(4)</td>
</tr>
<tr>
<td>Breast</td>
<td>0.919(5)</td>
</tr>
<tr>
<td>Cam</td>
<td>0.707(3)</td>
</tr>
<tr>
<td>Covtype</td>
<td>0.928(4)</td>
</tr>
<tr>
<td>Estate</td>
<td>0.601(1)</td>
</tr>
<tr>
<td>Fourclass</td>
<td>0.955(5)</td>
</tr>
<tr>
<td>German</td>
<td>0.631(4)</td>
</tr>
<tr>
<td>Ism</td>
<td>0.9805(4)</td>
</tr>
<tr>
<td>Letter</td>
<td>0.972(3)</td>
</tr>
<tr>
<td>Oil</td>
<td>0.641(6)</td>
</tr>
<tr>
<td>Page</td>
<td>0.9065(4)</td>
</tr>
<tr>
<td>Pendigits</td>
<td>0.966(4)</td>
</tr>
<tr>
<td>Phoneme</td>
<td>0.8245(5)</td>
</tr>
<tr>
<td>PhosS</td>
<td>0.543(4)</td>
</tr>
<tr>
<td>Pima</td>
<td>0.702(4)</td>
</tr>
<tr>
<td>Satimage</td>
<td>0.774(4)</td>
</tr>
<tr>
<td>Segment</td>
<td>0.961(1)</td>
</tr>
<tr>
<td>Splice</td>
<td>0.913(4)</td>
</tr>
<tr>
<td>SVMguide</td>
<td>0.976(4)</td>
</tr>
<tr>
<td>Avg. Rank</td>
<td>3.95</td>
</tr>
</tbody>
</table>

F.T (C4.5) ✓ 9.6E-5 Base
F.T (CART) ✓ 9.6E-5 Base
F.T (Other) ✓ 0.8396 ✓ 1.3E-5
Table 5: Pruning strategy comparisons on AUC. “Err.ESt” is short for error estimation. The AUC of C4.5 and CCP-C4.5 pruned by FET are significantly better than those pruned by error estimation.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Err.ESt</th>
<th>FET</th>
<th>Err.ESt</th>
<th>FET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary</td>
<td>0.50(1)</td>
<td>0.560(2)</td>
<td>0.500(4)</td>
<td>0.613(1)</td>
</tr>
<tr>
<td>Breast</td>
<td>0.954(1)</td>
<td>0.951(3)</td>
<td>0.953(2)</td>
<td>0.951(3)</td>
</tr>
<tr>
<td>Cam</td>
<td>0.545(3)</td>
<td>0.747(2)</td>
<td>0.513(4)</td>
<td>0.755(1)</td>
</tr>
<tr>
<td>Cotype</td>
<td>0.977(4)</td>
<td>0.979(2)</td>
<td>0.979(2)</td>
<td>0.980(1)</td>
</tr>
<tr>
<td>Estate</td>
<td>0.505(3)</td>
<td>0.539(2)</td>
<td>0.505(3)</td>
<td>0.595(1)</td>
</tr>
<tr>
<td>Fourclass</td>
<td>0.964(3)</td>
<td>0.961(4)</td>
<td>0.965(2)</td>
<td>0.969(1)</td>
</tr>
<tr>
<td>German</td>
<td>0.708(3)</td>
<td>0.715(2)</td>
<td>0.706(4)</td>
<td>0.719(1)</td>
</tr>
<tr>
<td>Ism</td>
<td>0.870(3)</td>
<td>0.891(1)</td>
<td>0.848(4)</td>
<td>0.887(2)</td>
</tr>
<tr>
<td>Letter</td>
<td>0.985(3)</td>
<td>0.993(1)</td>
<td>0.982(4)</td>
<td>0.989(2)</td>
</tr>
<tr>
<td>Oil</td>
<td>0.776(4)</td>
<td>0.791(3)</td>
<td>0.812(2)</td>
<td>0.824(1)</td>
</tr>
<tr>
<td>Page</td>
<td>0.967(4)</td>
<td>0.975(1)</td>
<td>0.969(3)</td>
<td>0.973(2)</td>
</tr>
<tr>
<td>Pendigits</td>
<td>0.984(4)</td>
<td>0.986(3)</td>
<td>0.988(2)</td>
<td>0.989(1)</td>
</tr>
<tr>
<td>Phonomew</td>
<td>0.856(4)</td>
<td>0.860(2)</td>
<td>0.858(3)</td>
<td>0.868(1)</td>
</tr>
<tr>
<td>PhonS</td>
<td>0.694(1)</td>
<td>0.649(3)</td>
<td>0.595(4)</td>
<td>0.688(2)</td>
</tr>
<tr>
<td>Pima</td>
<td>0.751(4)</td>
<td>0.769(1)</td>
<td>0.758(2)</td>
<td>0.755(3)</td>
</tr>
<tr>
<td>Satimage</td>
<td>0.897(4)</td>
<td>0.912(2)</td>
<td>0.907(3)</td>
<td>0.917(1)</td>
</tr>
<tr>
<td>Segment</td>
<td>0.987(3)</td>
<td>0.987(3)</td>
<td>0.988(1)</td>
<td>0.988(1)</td>
</tr>
<tr>
<td>Splice</td>
<td>0.955(1)</td>
<td>0.951(4)</td>
<td>0.954(1)</td>
<td>0.953(3)</td>
</tr>
<tr>
<td>SVMguide</td>
<td>0.982(4)</td>
<td>0.985(1)</td>
<td>0.984(2)</td>
<td>0.986(2)</td>
</tr>
<tr>
<td>Avg.Rank</td>
<td>3.0</td>
<td>2.15</td>
<td>2.65</td>
<td>1.55</td>
</tr>
</tbody>
</table>

Fr.T (C4.5) ✓ 0.0184 Base
Fr.T (CCP) ✓ 0.0076 Base

6.2 Comparison of Pruning Strategies
In this section, we compared FET-based pruning with the pruning based on error estimation as originally proposed in C4.5 [16]. We reuse the data sets from previous subsection, and apply error-based pruning and FET-based pruning separately on the trees built by C4.5 and CCP-C4.5, where the confidence level of FET was set to 99% (i.e. the p-Value threshold is set to 0.01). Note that the tree constructions differences between C4.5 and CCP-C4.5 are out of scope in this subsection; we carried out separate Friedman test on C4.5 and CCP-C4.5 respectively and only compare the different pruning strategies. HDDD was excluded from this comparison since it provides no separate pruning strategies.

Table 6: Pruning strategy comparisons on number of leaves. The leaves on trees of C4.5 and CCP-C4.5 pruned by FET are not significantly more than those pruned by error estimation.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Err.ESt</th>
<th>FET</th>
<th>Err.ESt</th>
<th>FET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary</td>
<td>2.70(2)</td>
<td>33.9(3)</td>
<td>2.70(2)</td>
<td>87.6(4)</td>
</tr>
<tr>
<td>Breast</td>
<td>7.74(4)</td>
<td>6.4(2)</td>
<td>7.03(3)</td>
<td>6.2(1)</td>
</tr>
<tr>
<td>Cam</td>
<td>54.5(2)</td>
<td>445.9(3)</td>
<td>7.9(1)</td>
<td>664.6(4)</td>
</tr>
<tr>
<td>Cotype</td>
<td>156.0(1)</td>
<td>175.3(3)</td>
<td>166.9(2)</td>
<td>189.0(4)</td>
</tr>
<tr>
<td>Estate</td>
<td>2.2(2)</td>
<td>5.2(4)</td>
<td>2.0(1)</td>
<td>4.6(3)</td>
</tr>
<tr>
<td>Fourclass</td>
<td>13.9(3)</td>
<td>13.0(1)</td>
<td>14.1(4)</td>
<td>13.3(2)</td>
</tr>
<tr>
<td>German</td>
<td>41.3(3)</td>
<td>40.0(1)</td>
<td>47.1(4)</td>
<td>40.1(2)</td>
</tr>
<tr>
<td>Ism</td>
<td>19.8(1)</td>
<td>26.3(3)</td>
<td>21.9(2)</td>
<td>31.0(4)</td>
</tr>
<tr>
<td>Letter</td>
<td>30.2(1)</td>
<td>45.1(3)</td>
<td>34.3(2)</td>
<td>47.4(4)</td>
</tr>
<tr>
<td>Oil</td>
<td>8.0(1)</td>
<td>10.7(3)</td>
<td>8.5(2)</td>
<td>12.0(4)</td>
</tr>
<tr>
<td>Page</td>
<td>29.4(2)</td>
<td>28.9(1)</td>
<td>32.6(3)</td>
<td>34.0(4)</td>
</tr>
<tr>
<td>Pendigits</td>
<td>35.1(3)</td>
<td>37.8(4)</td>
<td>31.5(1)</td>
<td>32.8(2)</td>
</tr>
<tr>
<td>Phonomew</td>
<td>32.6(4)</td>
<td>29.7(2)</td>
<td>30.3(1)</td>
<td>27.0(1)</td>
</tr>
<tr>
<td>PhonS</td>
<td>68.2(2)</td>
<td>221.5(3)</td>
<td>37.7(1)</td>
<td>311.4(4)</td>
</tr>
<tr>
<td>Pima</td>
<td>15.9(4)</td>
<td>12.7(2)</td>
<td>13.3(3)</td>
<td>11.8(1)</td>
</tr>
<tr>
<td>Satimage</td>
<td>83.7(1)</td>
<td>107.8(3)</td>
<td>94.6(2)</td>
<td>119.2(4)</td>
</tr>
<tr>
<td>Segment</td>
<td>8.3(3)</td>
<td>8.4(4)</td>
<td>6.4(1)</td>
<td>7.2(2)</td>
</tr>
<tr>
<td>Splice</td>
<td>32.3(3)</td>
<td>18.0(1)</td>
<td>22.6(4)</td>
<td>18.4(2)</td>
</tr>
<tr>
<td>SVMguide</td>
<td>14.5(3)</td>
<td>12.3(2)</td>
<td>15.3(4)</td>
<td>11.9(1)</td>
</tr>
<tr>
<td>Avg.Rank</td>
<td>2.3</td>
<td>2.45</td>
<td>2.25</td>
<td>2.7</td>
</tr>
</tbody>
</table>

Fr.T (C4.5) ✓ 0.4913 Base
Fr.T (CCP) ✓ 0.2513 Base

6.3 Comparisons with Sampling Techniques
We now compare CCP-based decision trees against sampling based methods discussed in Section 5. Note that the wrapper
is optimized on training sets using 5 × 2 cross validation to determine the sampling levels. The algorithm was then evaluated on the corresponding 5 × 2 cross validated testing set.

The performances of three pairs of decision trees are shown in Table 7. The first pair “Original” has no modification on either data or decision tree algorithms. The second pair “Sampling based” uses the wrapper to re-sample the training data which is then used by original decision tree algorithms to build the classifier; and “CCP based” shows the performance of CCP-based decision trees learned from original data. Friedman test on the AUC values shows that, although the “wrapped” data can help to improve the performance of original decision trees, using CCP-based algorithms can obtain statistically better classifiers directly trained on the original data.

7 Conclusion and future work
We address the problem of designing a decision tree algorithm for classification which is robust against class imbalance in the data. We first explain why traditional decision tree measures, like information gain, are sensitive to class imbalance. We do that by expressing information gain in terms of the confidence of a rule. Information gain, like confidence, is biased towards the majority class. Having identified the cause of the problem, we propose a new measure, Class Confidence Proportion (CCP). Using both theoretical and geometric arguments we show that CCP is insensitive to class distribution. We then embed CCP in information gain and use the improved measure to construct the decision tree algorithms can obtain statistically better classifiers directly trained on the original data.
Table 7: Performances comparisons on AUC generated by original, sampling and CCP-based techniques. “W+CART/W+C4.5” means applying wrappers to sample training data before it’s learned by CART/C4.5. In this table, CCP-based CART decision tree is significantly better than all trees in “Original” and “Sampling based” categories.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Original CART</th>
<th>CL4.5</th>
<th>Sampling based W+CART/W+C4.5</th>
<th>CCP based CART/CCP-CART/CCP-C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary</td>
<td>0.514(5)</td>
<td>0.501(6)</td>
<td>0.582(4)</td>
<td>0.614(2)</td>
</tr>
<tr>
<td>Breast</td>
<td>0.928(6)</td>
<td>0.954(2)</td>
<td>0.955(1)</td>
<td>0.953(4)</td>
</tr>
<tr>
<td>Cam</td>
<td>0.701(5)</td>
<td>0.545(6)</td>
<td>0.660(5)</td>
<td>0.670(4)</td>
</tr>
<tr>
<td>Covtype</td>
<td>0.910(6)</td>
<td>0.977(4)</td>
<td>0.974(5)</td>
<td>0.980(2)</td>
</tr>
<tr>
<td>Estate</td>
<td>0.583(3)</td>
<td>0.505(6)</td>
<td>0.560(5)</td>
<td>0.580(4)</td>
</tr>
<tr>
<td>Fourclass</td>
<td>0.978(1)</td>
<td>0.964(5)</td>
<td>0.943(6)</td>
<td>0.965(4)</td>
</tr>
<tr>
<td>German</td>
<td>0.630(6)</td>
<td>0.708(2)</td>
<td>0.668(5)</td>
<td>0.690(4)</td>
</tr>
<tr>
<td>Ion</td>
<td>0.799(6)</td>
<td>0.870(5)</td>
<td>0.905(3)</td>
<td>0.909(1)</td>
</tr>
<tr>
<td>Letter</td>
<td>0.964(6)</td>
<td>0.985(4)</td>
<td>0.977(5)</td>
<td>0.989(1)</td>
</tr>
<tr>
<td>Oil</td>
<td>0.666(6)</td>
<td>0.776(5)</td>
<td>0.806(3)</td>
<td>0.789(4)</td>
</tr>
<tr>
<td>Page</td>
<td>0.920(6)</td>
<td>0.967(5)</td>
<td>0.970(4)</td>
<td>0.978(1)</td>
</tr>
<tr>
<td>Pendigits</td>
<td>0.963(6)</td>
<td>0.984(4)</td>
<td>0.982(5)</td>
<td>0.987(3)</td>
</tr>
<tr>
<td>PhoNem</td>
<td>0.838(6)</td>
<td>0.856(5)</td>
<td>0.890(2)</td>
<td>0.894(1)</td>
</tr>
<tr>
<td>Phoneme</td>
<td>0.542(6)</td>
<td>0.694(1)</td>
<td>0.665(5)</td>
<td>0.670(4)</td>
</tr>
<tr>
<td>Pima</td>
<td>0.695(6)</td>
<td>0.751(4)</td>
<td>0.742(5)</td>
<td>0.755(2)</td>
</tr>
<tr>
<td>Salimage</td>
<td>0.736(6)</td>
<td>0.897(4)</td>
<td>0.887(5)</td>
<td>0.904(3)</td>
</tr>
<tr>
<td>Segment</td>
<td>0.996(5)</td>
<td>0.982(5)</td>
<td>0.980(5)</td>
<td>0.982(4)</td>
</tr>
<tr>
<td>Splice</td>
<td>0.885(5)</td>
<td>0.954(1)</td>
<td>0.829(6)</td>
<td>0.942(3)</td>
</tr>
<tr>
<td>SVMguide</td>
<td>0.973(6)</td>
<td>0.982(5)</td>
<td>0.985(3)</td>
<td>0.987(2)</td>
</tr>
</tbody>
</table>

Avg.Rank: 3.05 3.85 3.15 2.7 1.75 2.3 1.96

References