2013 IEEE 13th International Conference on Data Mining

Conformal Prediction Using Decision Trees

Ulf Johansson*, Henrik Boström†, Tuve Löfström*
*School of Business and IT
University of Borås, Sweden
Email: {ulf.johansson, tuve.lofstrom}@hb.se
†Department of Computer and System Sciences, Stockholm University, Sweden
Email: henrik.bostrom@dsv.su.se

***Abstract*—Conformal prediction is a relatively new framework in which the predictive models output sets of predictions with a bound on the error rate, i.e., in a classification context, the probability of excluding the correct class label is lower than a predefined significance level. An investigation of the use of decision trees within the conformal prediction framework is presented, with the overall purpose to determine the effect of different algorithmic choices, including split criterion, pruning scheme and way to calculate the probability estimates. Since the error rate is bounded by the framework, the most important property of conformal predictors is efficiency, which concerns minimizing the number of elements in the output prediction sets. Results from one of the largest empirical investigations to date within the conformal prediction framework are presented, showing that in order to optimize efficiency, the decision trees should be induced using no pruning and with smoothed probability estimates. The choice of split criterion to use for the actual induction of the trees did not turn out to have any major impact on the efficiency. Finally, the experimentation also showed that when using decision trees, standard inductive conformal prediction was as efficient as the recently suggested method cross-conformal prediction. This is an encouraging results since cross-conformal prediction uses several decision trees, thus sacrificing the interpretability of a single decision tree.**

Keywords*—*Conformal prediction, Decision trees

I. Introduction.

One can often get a fairly good estimate of the general predictive performance of a model, e.g., by estimating the error rate on a validation set or through cross-validation. However, in many situations it is not sufficient to know how well a model performs in general, but an assessment of the (un)certainty of each individual prediction is required. For example, we may choose to rely only on some of the individual predictions, typically those that have an acceptable level of certainty. Many classification models do not only output a single, most likely, class, but instead output a probability distribution over the possible classes. As an example, standard decision trees [1], [2] are referred to as probability estimation trees (PETs) [3], when producing probabilities instead of just the class label. Obviously, probability distributions do indeed provide means to filter out unlikely or uncertain predictions, but often the output distributions are not well-calibrated, i.e., the predicted class probabilities do not reflect the true, underlying probabilities, see e.g., [4]. Although there have been several attempts to improve, or calibrate, predicted class probabilities [5], the proposed methods do not provide any guarantees for their correctness or any bound on the corresponding errors.

Conformal prediction [6] is a relatively new framework that addresses the problem of assessing the (un)certainty of each individual prediction in a slightly different way than by calibrating predicted class probabilities. Instead of predicting a single class label, or a distribution over the labels, a conformal predictor outputs a set of labels, with a bounded error, i.e., the probability of excluding the correct class label is guaranteed to be smaller than a predetermined significance level. So, the price you pay for the bounded error rate is that not all predictions are singletons, i.e., they are less informative. The framework has been applied in conjunction with several popular learning algorithms, such as ANNs [7], kNN [8], [9], [10], SVMs [10], [11], and random forests [9], [10]. Each learning algorithm requires a specific adaptation of the framework, and design choices as well as parameter settings that are well suited for the standard setting, i.e., when maximizing classification accuracy, may need to be reconsidered. In other words, lessons learned from applying an algorithm in the standard framework may, or may not, carry over to the conformal prediction framework.

Decision tree learning is one of the most widely used machine learning techniques, and its popularity can be explained by its relative efficiency, effectiveness and ability to produce interpretable models. In addition, sets of decision trees are often combined into ensembles, typically using bagging [12] or boosting [13]. In particular the random forest technique [14], combining bagging with random subspacing [15], is often referred to as a state-of-the-art ensemble technique. With this in mind, we argue that decision trees are still important to investigate, both as single models and as parts of ensembles.

For decision trees and PETs, which, as described above, generalize the former by returning class distributions instead of single class predictions, it has been observed that different algorithmic choices, such as split metric, probability estimate and whether or not to prune, may have a significant impact on the predictive performance, see e.g., [3], [16]. However, no such corresponding results have been reported within the conformal prediction framework. Hence, it is not known what algorithmic choices should be preferred when learning decision trees (or PETs) in this novel framework. To counter this, we present an extensive empirical investigation on the effect of different algorithmic choices for decision tree learning within the conformal prediction framework. This is not only the first investigation of conformal prediction using decision trees, but also one of the most comprehensive empirical investigations of conformal prediction presented for any algorithm.

In the next section, we provide some background on the conformal prediction framework and decision trees, in

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.85

particular on how the framework is adapted to decision tree learning, and the different algorithmic choices which must be considered. In section III we provide the details of the empirical study, before presenting and analyzing the results in section IV. In section V, we discuss the results in relation to previous findings. Finally, the key conclusions are presented in section VI.

II. Background.

In this section, we first briefly describe the conformal prediction framework, and then discuss the algorithmic choices for decision tree learning that will be considered in the study.

A. Conformal prediction

The conformal prediction framework [6] was originally developed for numerical prediction (regression), but was later adapted to classification, which is what we focus on in this paper. The framework is general in that it can be used in conjunction with any learning algorithm. A central component of the framework is the (non-)conformity function, which gives a score to each instance and class label pair. When classifying a novel instance, scores are calculated for all possible class labels, and these scores are compared to scores obtained from instances with known labels. Labels that are found to not conform, i.e., when the corresponding conformity value is lower than some predetermined fraction of the calibration examples (here called significance level), are excluded. This means that for each instance to be classified the conformal prediction framework outputs a set of predictions, which may contain one, several, or even no class labels, i.e., the set may be empty. Under certain, but very general, assumptions, it can be guaranteed that the probability of excluding the true class label is bounded by the chosen significance level, independently of the chosen conformity function [6]. The original formulation of the framework assumes a transductive setting, i.e., the example to be classified has to be included in the calibration set, thus requiring recalculation of all (non-)conformity scores relative to each test example. The more efficient inductive setting, which is adopted here, instead assumes that conformity scores can be calculated for a calibration set without including the test example. The inductive setting requires the available examples to be split into a proper training set, used to train the model, and the calibration set, used to calculate the (non-)conformity scores. In this study, we assume that the conformity function C is defined relative to a trained predictive model M:

$$C(\langle \bar{x}, c \rangle) = F(c, M(\bar{x})) \tag{1}$$

where $\bar{x}$ is a vector of feature values (representing the example to be classified), c is a class label, $M(\bar{x})$ returns the class probability distribution predicted by the model, and the function F returns a score calculated from the chosen class label and predicted class distribution. A possible choice for this function, which is adopted here, is to use the margin [17], which gives the difference between the probability of the chosen class label and the probability of the most likely of the other labels, thus ranging from -1 to 1.[1]

Using a conformity function, a p-value for an example $\bar{x}$ and a class label c is calculated in the following way:

$$p_{\langle \bar{x}, c \rangle} = \frac{|\{s : s \in S \ \& \ C(s) \leq C(\langle \bar{x}, c \rangle)\}|}{|S|} \tag{2}$$

where S is a calibration set. The prediction for an example $\bar{x}$, where $\{c_1, \ldots, c_n\}$ are the possible class labels, is:

$$P(\bar{x}, \sigma) = \{c : c \in \{c_1, \ldots, c_n\} \ \& \ p_{\langle \bar{x}, c \rangle} > \sigma\} \tag{3}$$

where σ is a chosen significance level, e.g., 0.05. Note that the resulting prediction hence is a (possibly empty) subset of the possible class labels.

B. Decision trees

As mentioned in the introduction, the popularity of techniques for learning decision trees can be explained by their ability to produce transparent yet fairly accurate models. Furthermore, they are relatively fast and require a minimum of parameter tuning. The two most famous decision tree algorithms are C4.5/C5.0 [1] and CART [2].

The generation of a decision tree is done recursively by splitting the data set on the independent variables. Each possible split is evaluated by calculating the resulting *purity gain* if it was to be used to divide the data set D into the new subsets $\{D_1, \ldots, D_n\}$. The purity gain Δ is the difference in impurity between the original data set and the subsets as defined in equation (4) below, where $I(\cdot)$ is an impurity measure of a given node and $P(D_i)$ is the proportion of D that is placed in D_i. Naturally, the split resulting in the highest purity gain is selected, and the procedure is then repeated recursively for each subset in this split.

$$\Delta = I(D) - \sum_{i=1}^{n} P(D_i) \cdot I(D_i) \tag{4}$$

Different decision tree algorithms apply different impurity measures. C4.5 uses *entropy*; see equation (5) while CART optimizes the *Gini index* see equation (6). Here, C is the number of classes and $p(c_i|t)$ is the the fraction of instances belonging to class c_i at the current node t.

$$Entropy(t) = -\sum_{i=1}^{C} p(c_i|t) log_2 p(c_i|t) \tag{5}$$

$$Gini(t) = 1 - \sum_{i=1}^{C} [p(c_i|t)]^2 \tag{6}$$

Although the normal operation of a decision tree is to predict a class label based on an input vector, decision trees

[1] The conformity functions that have been proposed for random forests, [9], [10], work for sets of trees and are not directly meaningful for single trees. However, those of the previously proposed functions that are based on the proportion of trees voting for a particular class are similar in spirit to using the predicted probabilities of single trees.

can also be used to produce class membership probabilities; in which case they are referred to as PETs [3]. For PETs, the easiest way to obtain a class probability is to use the *relative frequency*; i.e., the proportion of training instances corresponding to a specific class in the specific leaf where the test instance fall. In equation (7) below, the probability estimate $p_i^{c_j}$, based on relative frequencies, is defined as

$$p_i^{c_j} = \frac{g(i,j)}{\sum_{k=1}^{C} g(i,k)} \tag{7}$$

where $g(i,j)$ gives the number of instances belonging to class j that falls in the same leaf as instance i, and C is the number of classes.

Often, however, some kind of smoothing technique is applied. The most common is called the *Laplace estimate* or the *Laplace correction*. The main reason for using a smoothing technique is that the basic relative frequency estimate does not consider the number of training instances reaching a specific leaf. Intuitively, a leaf containing many training instances is a better estimator of class membership probabilities. With this in mind, the Laplace estimator calculates the estimated probability as:

$$p_i^{c_j} = \frac{1 + g(i,j)}{C + \sum_{k=1}^{C} g(i,k)} \tag{8}$$

It should be noted that the Laplace estimator in fact introduces a prior uniform probability for each class; i.e., before any instances have reached the leaf, the probability for each class is $1/C$. Yet another smoothing operation is the *m-estimate*

$$p_i^{c_j} = \frac{mp_{c_j} + g(i,j)}{m + \sum_{k=1}^{C} g(i,k)} \tag{9}$$

where *m* is a parameter, and p_{c_j} is a prior probability for the class c_j. p_{c_j} is either assumed to be uniform, i.e., $p = 1/c$ or estimated from the training distribution. In the uniform case, the Laplace correction is a special case of the m-estimate with $m = c$.

In order to obtain what they call well-behaved PETs, Provost and Domingos [3] changed the C4.5 algorithm by turning off both pruning and the collapsing mechanism, which obviously led to substantially larger trees. This, together with the use of Laplace estimates, however, turned out to produce much better PETs; for details see the original paper.

C. Related work.

Conformal prediction is very much a framework under development. Vovk keeps a large number of older working papers regarding the *transductive confidence machine* (TCM) at www.vovk.net, while continuously updated versions of the more recent working papers are found at www.alrw.net.

Inductive conformal prediction is introduced in the book [6], and is further developed and analyzed in [18]. A new working paper [19] introduces the method of cross-conformal prediction, which is a hybrid of inductive conformal prediction and cross-validation. It must be noted, however, that all of these papers are highly mathematical and often abstract away from specific machine learning techniques. Even when using a specific machine learning technique, the purpose is to demonstrate a specific property of the framework. One example is [7] where inductive conformal prediction using neural networks is described in detail.

Having said that, there are a number of other papers on conformal prediction, typically either using it for a specific application, or investigating the effect of varying some property, most often the conformity function.

Nguyen et al. use conformal prediction in [8] for indoor localization, i.e., to identify and observe a moving human or object inside a building. Another example is Lambrou et al. who use conformal prediction on rule sets evolved by a genetic algorithm [20]. The method is applied on two real-world datasets for medical diagnosis. Yang et al. use the outlier measure of a random forest to design a nonconformity measure, and the resulting predictor is tested on some medical diagnosis problems [21]. Papadopoulos et al. use neural networks as inductive conformal predictors to obtain predictions which have well-calibrated and practically useful confidence measures for the problem of acute abdominal pain diagnosis in [22]. They compare the accuracy of their neural networks with the accuracy achieved using CART, but never use the decision trees as conformal predictors.

In [10], Devetyarov and Nouretdinov use random forests as on-line and off-line conformal predictors, with the overall purpose to compare the efficiency of three different nonconformity measures. Bhattacharyya also investigates different nonconformity functions for random forests, but in an inductive setting [9]. Both these interesting studies, however, use a very limited number of data sets, so they serve mainly as proofs-of-concept. Finally, we have in a recent study [23], evaluated conformal predictors based on decision trees which were evolved using genetic programing.

III. Method.

As described above, most of the work on efficiency has targeted the conformity functions, but the efficiency is also heavily dependent on the underlying model, including factors like training parameters and how probability estimates are calculated internally. In addition, there is no fully accepted measure to use for comparing the efficiency of the conformal predictions. With this in mind, there is an apparent need for studies explicitly evaluating techniques for producing efficient conformal predictors utilizing a specific kind of predictive model. Such studies should preferably use a sufficiently large number of data sets to allow for statistical inference, thus making it possible to establish best practices. As mentioned in the introduction, to the best of our knowledge, no such comparative studies have been performed in the field of classification using standard decision trees as the underlying algorithm for conformal prediction.

The overall purpose of this study is to evaluate different algorithmic choices for decision trees used as conformal predictors. We first, however, demonstrate the conformal prediction framework using decision trees that are all trained with identical settings. After that, in the main experiment, a number

of different settings are evaluated. More specifically, the split criterion, pruning method and the method for calculating the probability estimates are all varied, leading to, in total 12 different setups. Finally, we include another experiment, where standard ICP is compared to cross-conformal prediction (CCP) as suggested in [19]. Simply put, CCP is very similar to standard (internal) cross-validation. First, the available training data is divided into a number of folds, typically five or ten. In this study, we use five folds, since some of the data sets are fairly small. Then, a model (here a decision tree) is built from all but one of the folds, which is instead used as the calibration set. This procedure is repeated so all folds are used as the calibration set once. So, the result is five conformal predictors, each having a separate calibration set. When using CCP on a novel test instance, all five conformal predictors are applied to that test instance, and the resulting p-value is found by averaging the five individual p-values. Naturally, the idea is that this simple ensemble approach should lead to more efficient conformal predictors. This is also verified in the original study where, using MART classifiers, CCP is found to be slightly more efficient than standard ICP, see [19]. Unfortunately only two data sets, both fairly large and easy, were used in the analysis. It must be noted that when using CCP, there are in fact five models, so even if each model (here tree) is interpretable, the conformal predictor becomes an opaque ensemble, i.e., one of the main reasons for using decision trees in the first place is no longer true. Naturally, the entire procedure is also more time consuming since five models have to be trained and five conformal predictors have to be applied to each instance. With this in mind, it becomes important to know how general the results from the original study are, i.e., should CCPs be expected to be more efficient than ICPs, and if so, how much efficiency has to be sacrificed to achieve the interpretability offered by using a single tree.

All experimentation was performed in MatLab, using the decision trees as implemented in the statistics toolbox. The two split criteria evaluated are in MatLab called *gdi* and *deviance*. The gdi measure is identical to the Gini index, while deviance is the same as entropy. When pruning is applied, this is done using the internal pruning procedure in MatLab, which optimizes the pruning based on internal cross-validation. Finally, the three different ways of calculating the probability estimates are the unadjusted relative frequency, the Laplace estimate and the m-estimate, as defined in (7) to (9). For the m-estimate, *m* was set to 2, and the prior probabilities were estimated from the training data.

Since the error rate is bounded, a natural point of comparison between different conformal predictors is their efficiency, i.e., to what extent the predictors manage to exclude (incorrect) labels. When evaluating the efficiency, two different metrics were used. Since high efficiency roughly corresponds to a large number of singleton predictions, *OneC*, i.e., the proportion of predictions that include just one singe class, is a natural choice. Similarly, *MultiC* and *ZeroC* are the proportions of predictions consisting of more than one class label, and no class labels at all, respectively. One way of aggregating these numbers is *AvgC*, which is the average number of class labels in the predictions.

As mentioned in the Background, we use a conformity function based on the well-known concept of *margin*. For an instance i with the true class Y, the higher the probability estimate for class Y the more conforming the instance, and the higher the other estimates the less conforming the instance. Specifically, the most important of the other class probability estimates is the one with the maximum value $max_{j=1,\ldots,C:j\neq Y} p_i^{c_j}$, which might be close to, or even higher than p_i^Y. From this, we define the following conformity measure for a calibration instance z_i:

$$\alpha_i = p_i^Y - max_{j=1,\ldots,C:j\neq Y} p_i^{c_j} \tag{10}$$

For a specific test instance z_i, we use the equation below to calculate the corresponding conformity score for *each* possible class label c_k.

$$\alpha_i^{c_k} = p_i^{c_k} - max_{j=1,\ldots,C:j\neq k} p_i^{c_j} \tag{11}$$

For the evaluation, 4-fold cross-validation[2] was used, so all results reported are averaged over the four folds. The training data was split 2:1; i.e., 50% of the available instances were used for training and 25% were used as the calibration set. The 36 two-class data sets used are all publicly available from either the UCI repository [24] or the PROMISE Software Engineering Repository [25].

IV. Results.

In the first part of the results section, we demonstrate the behavior of conformal predictors. The decision trees used for this demonstration were induced using the Gini split criterion and no pruning. The probability estimates were calculated using m-estimates. Figure 1 below shows some key results on the Wisconsin breast cancer data set.

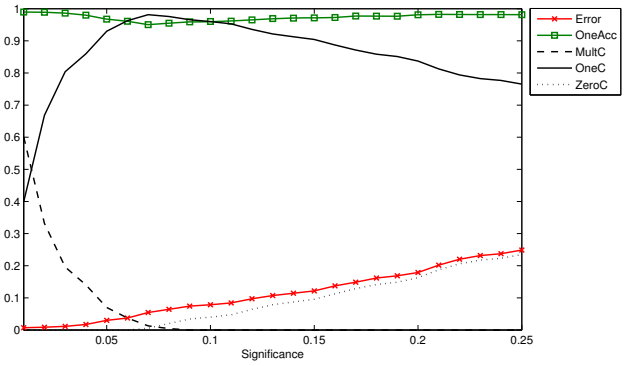

Fig. 1. Key characteristics for the conformal predictor. Breast cancer - w data set.

Looking first at the error, which in the conformal prediction framework is the proportion of predictions which does not contain the correct class, it is obvious that this conformal predictor is valid and well-calibrated. For every point in the graph, the actual error is very close to the corresponding significance

[2]This is of course a different folding than the internal folding used by CCP. The hold-out fold, which is used for the evaluation, is not used in any way when building or calibrating the model.

level. Analyzing the efficiency, the number of singleton predictions (OneC) starts at approximately 40% for $\epsilon = 0.01$, and then rises quickly to over 90% at $\epsilon \in [0.05, 0.15]$. The number of multiple predictions (MultiC), i.e., predictions containing both classes, of course, has the exact opposite behavior. For higher significance levels, OneC starts to decline since, for this rather easy data set, the number of empty predictions (ZeroC), quickly increases. OneAcc, which shows the accuracy of the singleton predictions, is fairly stable and always higher than the accuracy of the underlying tree model, i.e., the singleton predictions should be trusted more than predictions in general by the original model.

Figure 2 presents the same analysis, but now for the Diabetes data set, which is much harder, with an accuracy of the underlying model just over 70%. Here, OneC, consequently, is much smaller for low significance levels. As a matter of fact, for $\epsilon = 0.01$, less than 5% of the predictions are singletons, while the rest contain both classes. As the significance level increases, so does OneC, while MultiC decreases. For the significance levels plotted, there are no empty predictions. OneAcc, consequently decreases for higher significance levels, but is still always higher than the accuracy of the underlying model.

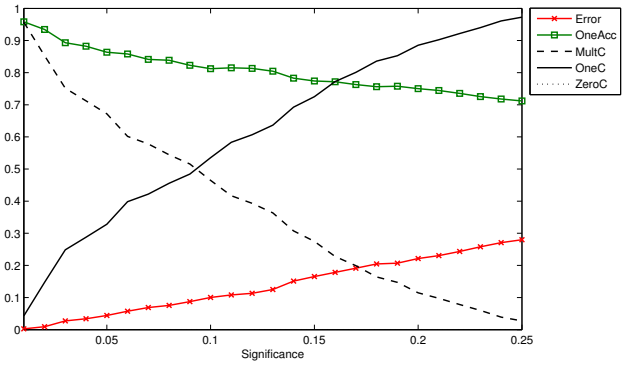

Fig. 2. Key characteristics for the conformal predictor. Diabetes data set.

Table I below shows similar results for five data sets, arranged in decreasing order of the accuracy of the underlying model, i.e., when used without the conformal prediction framework. For all data sets and significance levels, the error rate is very close to the significance level, indicating valid and well-calibrated conformal predictors. We can also observe the typical behavior of a conformal predictor, where MultiC decreases and OneC increases as the significance level increases. Once there are no multiple predictions, the number of empty predictions starts to rise. Naturally, the more accurate the underlying model is to start with, the higher OneAcc, and consequently ZeroC, tend to be.

TABLE I. CONFORMAL PREDICTION - DEMONSTRATION

kr-vs-kp (acc .992)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.01$	.012	.997	.000	.003	.992
$\epsilon = 0.05$	.054	.953	.000	.047	.992
$\epsilon = 0.1$	.102	.904	.000	.096	.994
$\epsilon = 0.2$	.212	.792	.000	.208	.996
breast - w (acc .941)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.01$	.007	.394	.606	.000	.989
$\epsilon = 0.05$	.031	.926	.074	.000	.966
$\epsilon = 0.1$	.083	.957	.000	.043	.959
$\epsilon = 0.2$	.173	.844	.000	.156	.980
credit - a (acc .836)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.01$	.012	.181	.819	.000	.962
$\epsilon = 0.05$	.051	.665	.335	.000	.928
$\epsilon = 0.1$	.104	.858	.142	.000	.881
$\epsilon = 0.2$	.177	.949	.000	.051	.867
diabetes (acc .706)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.01$	.003	.049	.951	.000	.947
$\epsilon = 0.05$	.038	.320	.680	.000	.884
$\epsilon = 0.1$	.099	.539	.461	.000	.816
$\epsilon = 0.2$	.221	.880	.120	.000	.749
bCancer (acc .633)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.01$	.000	.000	1.000	.000	N/A
$\epsilon = 0.05$	.053	.217	.783	.000	.783
$\epsilon = 0.1$	.088	.392	.608	.000	.786
$\epsilon = 0.2$	.207	.703	.297	.000	.711

Before looking at the efficiency results, Table II, on the next page, shows accuracies and sizes of the underlying models. It must be noted that due to space limitations, only accuracies from using relative frequencies are given in the table. When the trees are pruned, there is rarely anything to gain from using one of the smoothing techniques. For the unpruned trees, on the other hand, some minor improvement in accuracies were observed, especially on the unbalanced data sets, and when using the m-estimate. These differences are, however, much smaller than the differences between using pruned and unpruned models, and even between the two split criteria.

The most important result in Table II is that pruned models in general are more accurate than the unpruned. This is quite reassuring for the pruning technique applied, since the overall purpose of pruning of course is to produce trees that generalize better. To determine if there are any statistically significant differences, we use the statistical tests recommended by Demšar [26] for comparing several classifiers over a number of data sets; i.e., a Friedman test [27], followed by a Nemenyi post-hoc test [28]. With four setups and 36 data sets, the critical distance (for $\alpha = 0.05$) is 0.78, so based on these tests using pruning and the Gini split criterion did actually result in significantly higher accuracy, compared to both setups using unpruned trees. In addition, the setup using no pruning and the entropy split criterion produced significantly higher accuracy than unpruned trees induced using Gini. Comparing model sizes, presented as total number of nodes, the unpruned trees were, of course, significantly larger.

TABLE II. ACCURACY AND SIZE OF THE UNDERLYING TREES

	Accuracy				Size			
	Prun on		Prun off		Prun on		Prun off	
	Ent	Gini	Ent	Gini	Ent	Gini	Ent	Gini
ar1	.926	.926	.835	.885	1.0	1.0	6.0	6.0
ar4	.813	.804	.794	.795	3.0	3.5	7.5	7.5
breast-cancer	.699	.706	.629	.633	7.0	4.5	35.5	29.0
breast-w	.937	.944	.943	.941	13.0	11.5	19.5	17.0
colic	.818	.845	.813	.807	7.5	11.0	25.5	22.5
credit-a	.839	.855	.835	.835	6.0	6.5	40.0	43.5
credit-g	.718	.727	.695	.672	19.5	12.0	107.5	97.5
diabetes	.723	.734	.703	.706	11.5	10.0	73.0	74.0
heart-c	.733	.749	.749	.729	5.5	13.5	26.0	26.5
heart-h	.789	.772	.793	.782	7.5	9.0	28.5	23.5
heart-s	.763	.796	.745	.778	12.5	13.5	23.0	21.5
hepatitis	.793	.806	.787	.793	3.5	4.5	11.0	7.5
ionosphere	.860	.903	.855	.880	6.5	5.0	18.5	19.0
jEdit4042	.689	.694	.708	.730	11.0	20.5	32.5	27.5
jEdit4243	.656	.634	.602	.631	17.0	27.5	45.0	45.5
jm1	.812	.810	.754	.751	18.5	10.0	944.5	877.5
kc1	.853	.844	.813	.818	8.0	10.5	139.5	143.0
kc2	.818	.820	.801	.764	8.5	8.0	33.0	33.5
kc3	.906	.906	.860	.860	1.0	1.0	17.0	22.0
kr-vs-kp	.991	.987	.989	.992	43.5	54.0	48.5	53.0
labor	.754	.700	.789	.843	5.0	3.5	5.0	6.0
letter	.977	.986	.982	.977	12.5	21.0	21.5	28.5
liver	.644	.646	.655	.621	19.5	13.5	42.0	37.5
mozilla4	.943	.944	.928	.926	36.5	43.0	519.5	517.5
mw1	.923	.918	.901	.893	1.0	2.5	14.0	18.5
pc1_req	.622	.663	.609	.628	11.5	7.5	37.5	26.0
pc3	.898	.898	.858	.845	1.0	1.0	84.0	70.0
pc4	.888	.877	.878	.882	27.5	24.5	60.5	61.0
promoters	.774	.727	.783	.690	6.0	6.0	6.5	7.0
sick	.987	.984	.986	.985	25.5	20.5	36.5	27.5
sonar	.712	.716	.678	.716	10.0	5.5	16.0	19.0
spambase	.919	.903	.913	.906	114.5	95.0	221.0	220.5
spect	.782	.797	.789	.801	10.5	7.0	24.0	20.5
spectf	.764	.770	.802	.767	13.0	15.0	25.0	28.0
tic-tac-toe	.932	.907	.890	.904	47.5	48.5	54.5	52.0
vote	.961	.954	.956	.942	5.5	4.0	8.5	10.0
Mean	**.823**	**.824**	**.808**	**.809**	**15.5**	**15.4**	**79.4**	**76.3**
MeanRank	**2.15**	**2.00**	**2.86**	**2.99**	**1.51**	**1.56**	**3.46**	**3.47**

Turning to the efficiency results, Table III, on the next page, shows the efficiency for trees induced using Gini. The efficiency is measured as AvgC, i.e., the average number of elements (class labels) in a prediction. When the significance level is $\epsilon = 0.01$, most predictions of course contain both class labels. Comparing the different setups, we see that using unpruned trees with smoothing is the best option. With six setups, the critical distance for another Friedman test, followed by a Nemenyi post-hoc test, is (for $\alpha = 0.05$) as high as 1.26. So, the only statistical significant difference is that using unpruned trees with smoothing is more efficient than unpruned trees using relative frequencies. When $\epsilon = 0.05$, unpruned trees with smoothing are again clearly the most efficient. Now, these setups are significantly better than both setups using relative frequencies. In addition, both setups using smoothed pruned trees were significantly more efficient than unpruned trees using relative frequencies. For $\epsilon = 0.1$, the differences in mean ranks are generally smaller. Unpruned trees using m-estimates were, however, still significantly more efficient than both setups utilizing relative frequencies. Summarizing these results, the best choice is obviously to use unpruned trees, with either Laplace or m-estimate corrections. The second best group is pruned trees using smoothing.

Table IV also shows efficiency results for trees induced using Gini, but now the efficiency is measured using OneC; i.e., the proportion of singleton predictions. The overall impression is that these OneC results are very similar to the AvgC results. Again, we can see four different groups; the best option is unpruned and smoothed trees, followed by pruned smoothed trees. Using relative frequencies is generally the worst option, with unpruned and unsmoothed trees being the least efficient choice overall. There are relatively few statistically significant differences, even though the results seem to be fairly consistent over all data sets. Actually, the only difference that is statistically significant when $\epsilon = 0.01$ is that unpruned trees using Laplace correction are significantly more efficient than unpruned trees using no smoothing. When $\epsilon = 0.05$, both setups using unpruned trees with smoothing obtained a significantly higher OneC than both setups using no smoothing. For $\epsilon = 0.1$, finally, the only difference from when $\epsilon = 0.05$ is that unpruned trees using the Laplace estimate is no longer significantly more efficient than unpruned and unsmoothed trees.

Table V below summarizes the efficiency result for trees induced using the entropy criterion by showing values and ranks averaged over all data sets.

TABLE V. EFFICIENCY RESULTS FOR ENTROPY

	Prun On			Prun Off		
AvgC	Rf	LP	mE	Rf	LP	mE
$\epsilon = 0.01$						
Mean	1.87	1.85	1.85	1.88	1.79	1.79
Mean Rank	4.08	3.72	3.72	4.33	2.67	2.47
$\epsilon = 0.05$						
Mean	1.57	1.53	1.54	1.61	1.46	1.46
Mean Rank	4.31	3.75	3.72	4.58	2.26	2.38
$\epsilon = 0.1$						
Mean	1.32	1.30	1.30	1.37	1.26	1.26
Mean Rank	3.78	3.46	3.60	4.83	2.79	2.54
	Prun On			Prun Off		
OneC	Rf	LP	mE	Rf	LP	mE
$\epsilon = 0.01$						
Mean	.133	.152	.152	.120	.213	.214
Mean Rank	4.08	3.69	3.75	4.31	2.69	2.47
$\epsilon = 0.05$						
Mean	.424	.460	.457	.384	.533	.534
Mean Rank	4.22	3.49	3.60	4.72	2.43	2.54
$\epsilon = 0.1$						
Mean	.655	.677	.674	.602	.710	.710
Mean Rank	3.78	3.38	3.49	4.53	2.97	2.86

The first impression is probably that these results are very similar to the ones presented above for Gini, and indeed, the same four distinct groups can be identified. Here too, the best choice is unpruned trees using any type of smoothing, while the second best is pruned trees, also using smoothing. So, again, trees using no smoothing is the least efficient choice. A deeper analysis, however, shows that when using entropy as the split criterion, the advantage of using unpruned and smoothed trees is even larger. Specifically, for $\epsilon = 0.05$, the two setups utilizing unpruned and smoothed trees actually obtained significantly lower AvgC than all other setups.

Table VI on the next page, shows a direct pairwise comparison between the best setup identified, i.e., unpruned trees that were smoothed using the m-estimate, and all other setups. The numbers tabulated are wins for the setup using no pruning and m-estimate smoothing.

TABLE III. EFFICIENCY RESULTS FOR GINI - AVGC

	$\epsilon = 0.01$						$\epsilon = 0.05$						$\epsilon = 0.1$					
	Prun On			Prun Off			Prun On			Prun Off			Prun On			Prun Off		
	Rf	LP	mE	Rf	LP	mE	Rf	LP	mE	Rf	LP	mE	Rf	LP	mE	Rf	LP	mE
ar1	2.00	2.00	2.00	2.00	2.00	2.00	1.77	1.80	1.81	1.75	1.55	1.57	1.15	1.13	1.19	1.33	1.09	1.06
ar4	2.00	2.00	2.00	2.00	2.00	2.00	1.90	1.71	1.70	1.89	1.71	1.76	1.57	1.34	1.37	1.46	1.34	1.32
bCancer	2.00	2.00	2.00	2.00	2.00	2.00	1.88	1.80	1.83	1.87	1.78	1.78	1.69	1.56	1.58	1.66	1.59	1.61
breast-w	1.89	1.88	1.86	1.95	1.60	1.61	1.21	1.19	1.20	1.37	1.08	1.07	0.97	0.97	0.98	0.97	0.96	0.96
colic	2.00	2.00	2.00	2.00	2.00	2.00	1.76	1.57	1.57	1.82	1.65	1.67	1.30	1.18	1.20	1.53	1.31	1.32
credit-a	1.97	1.93	1.94	1.97	1.82	1.82	1.44	1.43	1.45	1.79	1.34	1.33	1.21	1.16	1.16	1.48	1.14	1.14
credit-g	1.95	1.95	1.94	1.98	1.93	1.93	1.58	1.59	1.60	1.82	1.65	1.64	1.41	1.46	1.46	1.65	1.48	1.47
diabetes	1.97	1.96	1.96	1.98	1.95	1.95	1.73	1.72	1.73	1.83	1.69	1.68	1.53	1.51	1.53	1.64	1.49	1.46
heart-c	2.00	2.00	2.00	2.00	2.00	2.00	1.83	1.74	1.73	1.79	1.69	1.67	1.53	1.40	1.38	1.54	1.38	1.37
heart-h	2.00	2.00	2.00	2.00	2.00	2.00	1.80	1.66	1.68	1.79	1.57	1.58	1.57	1.38	1.43	1.45	1.32	1.36
heart-s	2.00	2.00	2.00	2.00	2.00	2.00	1.81	1.67	1.70	1.86	1.75	1.78	1.51	1.39	1.38	1.62	1.47	1.47
hepatitis	2.00	2.00	2.00	2.00	2.00	2.00	1.77	1.62	1.61	1.92	1.88	1.89	1.58	1.49	1.44	1.72	1.60	1.70
iono	2.00	2.00	2.00	2.00	2.00	2.00	1.54	1.47	1.42	1.49	1.30	1.33	1.05	1.05	1.05	1.06	1.05	1.05
jEdit4042	2.00	2.00	2.00	2.00	2.00	2.00	1.88	1.82	1.81	1.87	1.69	1.66	1.68	1.53	1.52	1.61	1.42	1.41
jEdit4243	2.00	2.00	2.00	2.00	2.00	2.00	1.90	1.85	1.85	1.86	1.73	1.71	1.75	1.67	1.65	1.70	1.58	1.60
jm1	1.93	1.94	1.94	1.96	1.84	1.84	1.64	1.66	1.65	1.81	1.54	1.54	1.33	1.34	1.34	1.60	1.33	1.33
kc1	1.94	1.92	1.91	1.97	1.70	1.70	1.53	1.49	1.49	1.84	1.38	1.37	1.18	1.18	1.18	1.40	1.17	1.17
kc2	1.98	1.97	1.98	1.99	1.93	1.90	1.56	1.54	1.54	1.87	1.35	1.35	1.24	1.19	1.20	1.57	1.21	1.20
kc3	1.99	1.98	1.99	2.00	1.58	1.58	1.53	1.57	1.53	1.82	1.17	1.18	1.03	1.04	1.02	1.09	1.05	1.02
kr-vs-kp	1.02	1.06	1.05	1.00	1.00	1.00	0.96	0.96	0.96	0.95	0.95	0.95	0.91	0.90	0.90	0.91	0.91	0.90
labor	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	1.81	1.80	1.81	1.64	1.56	1.56
letter	1.77	1.25	1.25	1.52	1.33	1.31	0.97	0.97	0.97	0.98	0.98	0.98	0.91	0.92	0.91	0.92	0.92	0.92
liver	2.00	2.00	2.00	2.00	2.00	2.00	1.88	1.90	1.91	1.83	1.83	1.83	1.71	1.72	1.75	1.69	1.65	1.65
mozilla4	1.53	1.79	1.79	1.80	1.35	1.35	1.03	1.03	1.03	1.06	1.04	1.04	0.94	0.94	0.95	0.95	0.93	0.93
mw1	2.00	2.00	2.00	2.00	2.00	2.00	1.42	1.37	1.35	1.42	1.13	1.12	1.01	1.00	1.00	1.03	1.03	0.99
pc1_req	2.00	2.00	2.00	2.00	2.00	2.00	1.89	1.90	1.88	1.85	1.91	1.90	1.73	1.74	1.74	1.75	1.77	1.76
pc3	1.92	1.92	1.93	1.96	1.63	1.63	1.52	1.51	1.50	1.64	1.22	1.22	1.09	1.10	1.08	1.11	1.11	1.10
pc4	1.81	1.48	1.48	1.93	1.35	1.35	1.44	1.26	1.25	1.53	1.16	1.15	1.07	1.08	1.07	1.02	1.02	1.01
promoters	2.00	2.00	2.00	2.00	2.00	2.00	1.69	1.73	1.71	1.95	1.84	1.83	1.47	1.53	1.55	1.68	1.57	1.62
sick	1.08	1.01	1.01	1.24	1.02	1.02	0.96	0.95	0.95	0.96	0.96	0.96	0.91	0.90	0.90	0.91	0.91	0.91
sonar	2.00	2.00	2.00	2.00	2.00	2.00	1.89	1.95	1.91	1.85	1.85	1.87	1.65	1.84	1.77	1.66	1.50	1.49
spambase	1.85	1.80	1.80	1.87	1.58	1.58	1.16	1.17	1.17	1.16	1.12	1.12	1.00	0.99	0.99	0.98	0.97	0.97
spect	2.00	2.00	2.00	2.00	2.00	2.00	1.87	1.66	1.61	1.68	1.51	1.50	1.48	1.44	1.45	1.55	1.26	1.29
spectf	2.00	2.00	2.00	2.00	2.00	2.00	1.82	1.65	1.64	1.84	1.46	1.44	1.48	1.31	1.33	1.64	1.28	1.28
tic-tac-toe	1.48	1.48	1.49	1.61	1.46	1.48	1.10	1.10	1.11	1.15	1.16	1.14	0.97	0.96	0.97	0.99	0.99	0.99
vote	1.82	1.83	1.82	1.91	1.79	1.81	1.08	1.06	1.03	1.17	1.00	1.00	0.97	0.97	0.97	0.94	0.91	0.92
Mean	**1.89**	**1.87**	**1.86**	**1.91**	**1.80**	**1.80**	**1.58**	**1.53**	**1.52**	**1.64**	**1.46**	**1.46**	**1.32**	**1.28**	**1.28**	**1.37**	**1.26**	**1.26**
Mean Rank	**3.89**	**3.61**	**3.53**	**4.58**	**2.65**	**2.74**	**4.28**	**3.40**	**3.39**	**4.85**	**2.71**	**2.38**	**3.99**	**3.18**	**3.38**	**5.14**	**2.90**	**2.42**

TABLE VI. WINS FOR THE SETUP USING NO PRUNING AND M-ESTIMATE SMOOTHING AGAINST ALL OTHER SETUPS

	AvgC					OneC				
	Prun On			Prun off		Prun On			Prun off	
Gini	Rf	LP	mE	Rf	LP	Rf	LP	mE	Rf	LP
$\epsilon = 0.01$	**25.5**	**24**	**24.5**	**26.5**	17.5	**25.5**	**24.5**	**25.5**	**25.5**	17.5
$\epsilon = 0.05$	**27.5**	**25**	**24.5**	**29.5**	23	**28.5**	**26**	**25.5**	**32.5**	20
$\epsilon = 0.1$	**27**	**24**	**24**	**33**	21	**26**	**24**	**25**	**28**	20.5
	Prun On			Prun off		Prun On			Prun off	
Entropy	Rf	LP	mE	Rf	LP	Rf	LP	mE	Rf	LP
$\epsilon = 0.01$	**27**	**26**	**26**	**27**	21	**27**	**26**	**26**	**27**	21
$\epsilon = 0.05$	**30.5**	**25.5**	**25.5**	**31.5**	17.5	**29.5**	**23.5**	**24.5**	**29.5**	17.5
$\epsilon = 0.1$	23	22	**26**	**32**	21.5	21	23	23	**28**	18

With 36 data sets, a standard one-sided sign test requires 24 wins for significance with $\alpha = 0.05$[3]. Numbers representing statistically significant differences are underlined and bold. When comparing the setups head-to-head, the use of unpruned trees and m-estimate smoothing is actually almost always significantly more efficient than all other setups, with the exception of unpruned trees smoothed by Laplace correction.

Yet another important measure is the *precision*, i.e., in this setting, the proportion of removed labels that are actually incorrect. A perfect conformal predictor should remove only incorrect labels, which would correspond to a precision of 1.0. Figure 3 below, shows the precision for all six different setups using Gini on the PC3 data set.

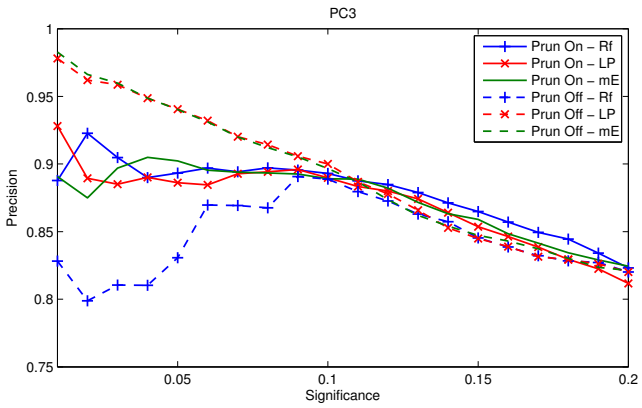

Fig. 3. Precision. PC3 data set.

[3]Since ties are divided equally between the two setups in the table, 23.5 wins is actually sufficient

TABLE IV. EFFICIENCY RESULTS FOR GINI - ONEC

	$\epsilon = 0.01$						$\epsilon = 0.05$						$\epsilon = 0.1$					
	Prun On			Prun Off			Prun On			Prun Off			Prun On			Prun Off		
	Rf	LP	mE	Rf	LP	mE	Rf	LP	mE	Rf	LP	mE	Rf	LP	mE	Rf	LP	mE
ar1	.000	.000	.000	.000	.000	.000	.232	.199	.191	.255	.447	.431	.853	.853	.812	.667	.909	.942
ar4	.000	.000	.000	.000	.000	.000	.104	.291	.300	.112	.289	.243	.426	.657	.630	.536	.656	.683
bCancer	.000	.000	.000	.000	.000	.000	.119	.199	.168	.126	.221	.217	.314	.438	.424	.343	.413	.392
breast-w	.113	.122	.139	.050	.396	.394	.789	.810	.800	.631	.924	.926	.948	.937	.936	.966	.957	.957
colic	.000	.000	.000	.000	.000	.000	.245	.432	.429	.185	.351	.332	.696	.815	.796	.473	.690	.682
credit-a	.035	.065	.064	.028	.183	.181	.562	.568	.552	.213	.659	.665	.793	.836	.836	.517	.862	.858
credit-g	.054	.047	.057	.019	.070	.070	.418	.410	.403	.176	.355	.364	.587	.545	.537	.354	.521	.530
diabetes	.033	.036	.042	.021	.051	.049	.272	.277	.272	.169	.314	.320	.467	.486	.473	.362	.514	.539
heart-c	.000	.000	.000	.000	.000	.000	.169	.261	.267	.211	.310	.330	.473	.598	.617	.465	.620	.630
heart-h	.000	.000	.000	.000	.000	.000	.201	.339	.322	.208	.428	.422	.426	.616	.575	.551	.676	.642
heart-s	.000	.000	.000	.000	.000	.000	.189	.325	.299	.145	.248	.218	.488	.606	.617	.382	.533	.526
hepatitis	.000	.000	.000	.000	.000	.000	.232	.379	.385	.077	.122	.110	.420	.508	.560	.276	.400	.296
iono	.000	.000	.000	.000	.000	.000	.461	.533	.578	.513	.698	.673	.903	.909	.920	.937	.946	.949
jEdit4042	.000	.000	.000	.000	.000	.000	.120	.178	.189	.128	.310	.343	.318	.473	.484	.395	.584	.588
jEdit4243	.000	.000	.000	.000	.000	.000	.100	.154	.146	.138	.271	.293	.252	.334	.352	.304	.415	.402
jm1	.071	.065	.063	.035	.161	.161	.356	.340	.347	.191	.460	.464	.667	.659	.665	.403	.668	.670
kc1	.060	.083	.091	.026	.299	.302	.465	.510	.508	.157	.625	.625	.821	.823	.824	.598	.826	.828
kc2	.017	.027	.025	.012	.065	.102	.435	.464	.464	.134	.653	.648	.763	.808	.795	.426	.793	.799
kc3	.009	.015	.009	.000	.420	.418	.469	.435	.472	.177	.826	.823	.965	.950	.958	.911	.950	.972
kr-vs-kp	.984	.939	.951	.998	.997	.997	.962	.962	.962	.951	.954	.953	.906	.902	.902	.907	.906	.904
labor	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.193	.196	.193	.356	.442	.442
letter	.229	.747	.752	.482	.674	.690	.966	.970	.967	.976	.977	.976	.911	.915	.912	.923	.923	.925
liver	.000	.000	.000	.000	.000	.000	.116	.096	.087	.174	.171	.174	.290	.284	.250	.308	.348	.348
mozilla4	.467	.208	.208	.201	.654	.652	.971	.971	.970	.940	.962	.965	.944	.944	.946	.950	.934	.933
mw1	.000	.003	.000	.000	.000	.000	.578	.630	.647	.584	.866	.878	.963	.953	.958	.950	.955	.965
pc1_req	.000	.000	.000	.000	.000	.000	.109	.100	.125	.150	.091	.100	.272	.259	.259	.247	.231	.244
pc3	.079	.078	.069	.045	.373	.368	.476	.491	.502	.356	.782	.782	.900	.887	.898	.891	.889	.903
pc4	.189	.520	.523	.073	.652	.654	.556	.744	.745	.471	.844	.850	.933	.916	.925	.977	.980	.991
promoters	.000	.000	.000	.000	.000	.000	.305	.275	.293	.047	.161	.172	.530	.472	.452	.322	.426	.378
sick	.918	.981	.980	.756	.980	.981	.957	.952	.954	.962	.963	.963	.909	.900	.901	.913	.909	.911
sonar	.000	.000	.000	.000	.000	.000	.111	.053	.091	.149	.149	.135	.346	.163	.231	.341	.505	.510
spambase	.150	.195	.201	.129	.416	.417	.843	.830	.829	.845	.884	.884	.982	.980	.982	.980	.974	.972
spect	.000	.000	.000	.000	.000	.000	.128	.344	.390	.321	.493	.500	.518	.563	.551	.453	.737	.707
spectf	.000	.000	.000	.000	.000	.000	.181	.353	.359	.164	.540	.557	.520	.690	.672	.362	.718	.718
tic-tac-toe	.523	.516	.512	.386	.544	.521	.903	.900	.887	.846	.844	.860	.968	.963	.969	.963	.961	.955
vote	.183	.167	.181	.092	.209	.193	.924	.938	.956	.801	.979	.972	.968	.968	.972	.938	.913	.924
Mean	**.114**	**.134**	**.135**	**.093**	**.198**	**.199**	**.417**	**.464**	**.468**	**.352**	**.533**	**.532**	**.656**	**.689**	**.688**	**.601**	**.714**	**.711**
Mean Rank	**3.89**	**3.61**	**3.61**	**4.53**	**2.65**	**2.71**	**4.31**	**3.54**	**3.47**	**4.90**	**2.46**	**2.32**	**3.85**	**3.65**	**3.43**	**4.50**	**3.00**	**2.57**

Interestingly enough, we see that the most efficient setups, i.e., unpruned and smoothed trees, also have the highest precision. Table VII, shows the precision for all setups and significance levels. Since precision is undefined when every prediction include all labels, only the data sets where all setups have at least some removed labels are included. The number of data sets used for the comparison is given after each significance level in the table. As we can see when considering all data sets, the graph shown for the PC3 data set above is actually quite representative. The setups using pruning and some kind of smoothing do indeed exhibit the highest precision. In particular for the significance levels $\epsilon = 0.01$ and $\epsilon = 0.05$, the differences in both precision values and the mean ranks are substantial.

So, one of the main findings of the empirical investigation presented above is that in order to maximize efficiency, i.e., minimize the number of elements in the predicted label sets, when using decision trees for conformal prediction, one should avoid pruning but employ smoothing, using either Laplace correction or the m-estimate.

TABLE VII. PRECISION RESULTS

	Prun On			Prun Off		
Gini	Rf	LP	mE	Rf	LP	mE
$\epsilon = 0.01(17)$						
Mean	.922	.936	.913	.873	.973	.970
Mean Rank	3.97	3.09	3.79	4.91	2.41	2.82
$\epsilon = 0.05(35)$						
Mean	.811	.869	.871	.816	.895	.894
Mean Rank	4.37	3.57	3.40	4.49	2.74	2.43
$\epsilon = 0.1(36)$						
Mean	.838	.842	.841	.816	.858	.857
Mean Rank	3.44	3.75	3.61	4.56	2.78	2.86
	Prun On			Prun Off		
Entropy	Rf	LP	mE	Rf	LP	mE
$\epsilon = 0.01(17)$						
Mean	.897	.905	.941	.848	.964	.970
Mean Rank	4.26	3.88	3.06	4.97	2.65	2.18
$\epsilon = 0.05(35)$						
Mean	.825	.873	.865	.766	.891	.898
Mean Rank	4.51	3.19	3.43	5.20	2.46	2.21
$\epsilon = 0.1(36)$						
Mean	.829	.841	.842	.807	.864	.865
Mean Rank	3.81	3.72	3.58	4.53	2.69	2.67

In addition, the investigation shows that the most efficient methods also have a higher precision when excluding elements,

i.e., a lower risk of incorrectly excluding the correct class label. These two results together make a very strong case for using trees with smoothing but no pruning in conformal prediction.

Comparing ICP to CCP, Figure 4 below shows the average OneC over all data sets for the different CCP variants and for ϵ-values between 0.01 and 0.2.

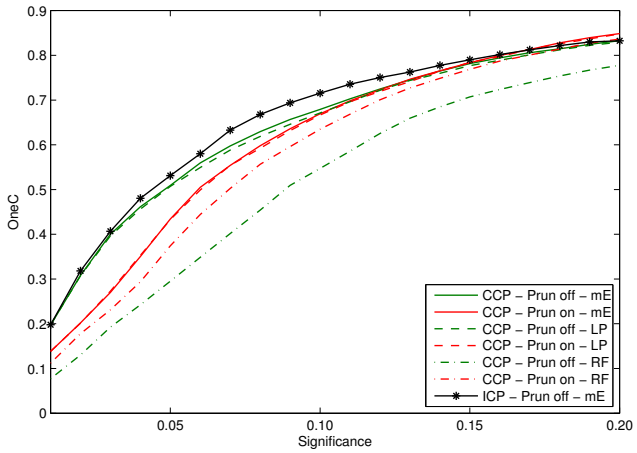

Fig. 4. Efficiency comparison ICP and CCP

It must be noted that just results for the Gini split criterion and only the best ICP from the previous experiment are included here, i.e., using no pruning and the m-estimate. Interestingly enough, the ICP is actually the most efficient, at least for small ϵ-values, which of course are the most important. Table VIII below, shows OneC and AvgC results for ICP and CCP.

TABLE VIII. EFFICIENCY RESULTS ICP AND CCP

			CCP				ICP
AvgC	Prun On			Prun Off			Prun Off
$\epsilon = 0.01$	Rf	LP	mE	Rf	LP	mE	mE
Mean	1.89	1.86	1.86	1.92	1.80	1.80	1.80
Mean Rank	4.72	4.38	4.35	5.31	2.88	3.10	3.28
$\epsilon = 0.05$							
Mean	1.62	1.56	1.56	1.70	1.49	1.49	1.46
Mean Rank	5.69	4.14	4.06	6.42	2.85	2.68	2.17
$\epsilon = 0.1$							
Mean	1.36	1.32	1.32	1.44	1.31	1.30	1.25
Mean Rank	5.07	3.76	3.67	6.44	3.83	3.22	2.00
			CCP				ICP
AvgC	Prun On			Prun Off			Prun Off
$\epsilon = 0.01$	Rf	LP	mE	Rf	LP	mE	mE
Mean	.112	.138	.138	.077	.198	.197	.198
Mean Rank	4.72	4.43	4.32	5.28	2.88	3.10	3.28
$\epsilon = 0.05$							
Mean	.374	.433	.434	.295	.507	.509	.531
Mean Rank	5.46	4.07	3.78	6.24	3.10	2.78	2.58
$\epsilon = 0.1$							
Mean	.635	.667	.669	.547	.671	.679	.715
Mean Rank	4.35	3.43	3.14	6.14	3.99	3.74	3.22

Looking at the values, the key result is that ICP is at least as efficient as the different CCP variants, which is in contrast to the results reported in [19]. As a matter of fact, comparing both average results and the mean ranks, ICP is often the most efficient setup, outperforming all CCP-variants. Also for CCP, using no pruning and smoothing is the best option. So, from this experiment, it is obvious that there is no need to sacrifice the interpretability of a single tree in order to produce more efficient conformal predictors through CCP.

V. DISCUSSION.

An immediate question is what the reasons for the observed effects are, i.e., why do the use of smoothing and the avoidance of pruning result in more efficient predictions? This can partly be explained by the fact that pruning leads to fewer nodes, which in turn leads to that the ranking produced by the conformity function will be less fine-grained, since all calibration examples that fall into the same node and have the same class label will obtain the same score. With more nodes, there will be more opportunities for the scoring function to place the example to be classified "in between" two calibration examples, i.e., the number of theoretically possible p-values increases with the number of nodes. Furthermore, when not using smoothing, all nodes for which the relative frequencies coincide will result in the same p-value. For example, it will make no difference if the example to be classified falls into a node with only one training example (of some class) or into a node with ten examples of the same class; the probability of that class will be 1.0 in both cases. If instead smoothing is employed, the class probability will in the former case be pushed closer to the *a priori* probability compared to what happens in the latter case. The corresponding p-value for the example to be classified will no longer be independent of which of the two nodes it falls within; the p-value for the particular class will obviously be higher in the latter case. This example does not only show why smoothing makes the conformity function more fine-grained, but also that extreme scores that simply are due to very few observations in a node can be avoided. Pruning should in principle also have a similar positive effect, since it increases the number of observations in each node. However, the experimental results indicate that the negative effect that comes from making the model less fine-grained is apparently stronger than this potential positive effect.

VI. CONCLUSION.

We have in this paper presented an empirical investigation of decision trees as conformal predictors. This is one of the first comprehensive studies where the effect of different algorithmic choices on conformal predictors is evaluated. The overall purpose was to analyze the effect of split criterion, pruning scheme and probability estimates, in order to produce a recommendation for how decision tress should be trained for conformal prediction. In the analysis, we focused on how to maximize the efficiency, but predictive performance, measured as OneAcc and Precision, was also investigated. In the experiments, it is shown that the best choice is to use no pruning but smoothed probability estimates, preferably the m-estimate. As a matter of fact, trees induced using these settings were not only the most efficient, but also obtained the highest precision.

Finally, the experimentation also shows that when using decision trees, CCP did not produce more efficient conformal predictors than ICP. This result is particularly interesting in

this context, since the implication is that there is no need to trade interpretability for efficiency. As a matter of fact, the conformal predictor built using just one interpretable decision tree was generally as efficient as all CCP variants evaluated.

REFERENCES

[1] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[2] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Chapman & Hall/CRC, January 1984.

[3] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Mach. Learn.*, vol. 52, no. 3, pp. 199–215, 2003.

[4] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers," in *Proc. 18th International Conference on Machine Learning*, 2001, pp. 609–616.

[5] H. Boström, "Calibrating random forests," in *Proc. of the International Conference on Machine Learning and Applications*. IEEE Computer Society, 2008, pp. 121–126.

[6] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic learning in a random world*. Springer, 2005.

[7] H. Papadopoulos, "Inductive conformal prediction: Theory and application to neural networks," *Tools in Artificial Intelligence*, vol. 18, pp. 315–330, 2008.

[8] K. Nguyen and Z. Luo, "Conformal prediction for indoor localisation with fingerprinting method," *Artificial Intelligence Applications and Innovations*, pp. 214–223, 2012.

[9] S. Bhattacharyya, "Confidence in predictions from random tree ensembles," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 71–80.

[10] D. Devetyarov and I. Nouretdinov, "Prediction with confidence based on a random forest classifier," *Artificial Intelligence Applications and Innovations*, pp. 37–44, 2010.

[11] L. Makili, J. Vega, S. Dormido-Canto, I. Pastor, and A. Murari, "Computationally efficient svm multi-class image recognition with confidence measures," *Fusion Engineering and Design*, vol. 86, no. 6, pp. 1213–1216, 2011.

[12] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[13] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, October 2001.

[15] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[16] C. Ferri, P. Flach, and J. Hernández-Orallo, "Improving the auc of probabilistic estimators trees," in *Proc. of the 14th European Conference on Artifical Intelligence*, vol. 2837. Springer, 2003, pp. 121–132.

[17] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The annals of statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[18] V. Vovk, "Conditional validity of inductive conformal predictors," *Journal of Machine Learning Research - Proceedings Track*, vol. 25, pp. 475–490, 2012.

[19] ——, "Cross-conformal predictors," arXiv:1208.0806, Tech. Rep., 2009.

[20] A. Lambrou, H. Papadopoulos, and A. Gammerman, "Reliable confidence measures for medical diagnosis with evolutionary algorithms," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 1, pp. 93–99, 2011.

[21] F. Yang, H. zhen Wang, H. Mi, C. de Lin, and W. wen Cai, "Using random forest for reliable classification and cost-sensitive learning for medical diagnosis," *BMC Bioinformatics*, vol. 10, no. S-1, 2009.

[22] H. Papadopoulos, A. Gammerman, and V. Vovk, "Reliable diagnosis of acute abdominal pain with conformal prediction," *Engineering Intelligent Systems*, vol. 17, no. 2, p. 127, 2009.

[23] U. Johansson, R. König, T. Löfström, and H. Boström, "Evolved decision trees as conformal predictors," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 1794–1801.

[24] A. Asuncion and D. J. Newman, "UCI machine learning repository," 2007.

[25] J. Sayyad Shirabad and T. Menzies, "The PROMISE Repository of Software Engineering Databases." School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[26] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[27] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of American Statistical Association*, vol. 32, pp. 675–701, 1937.

[28] P. B. Nemenyi, *Distribution-free multiple comparisons. PhD-thesis*. Princeton University, 1963.