Evolved Decision Trees as Conformal Predictors

Ulf Johansson*, Rikard König*, Tuve Löfström*, Henrik Boström†

*School of Business and IT, University of Borås, Sweden
{ulf.johansson, rikard.konig, tuve.lofstrom}@hb.se

†Department of Computer and Systems Sciences, Stockholm University, Sweden
henrik.bostrom@dsv.su.se

Abstract—In conformal prediction, predictive models output sets of predictions with a bound on the error rate. In classification, this translates to that the probability of excluding the correct class is lower than a predefined significance level, in the long run. Since the error rate is guaranteed, the most important criterion for conformal predictors is efficiency. Efficient conformal predictors minimize the number of elements in the output prediction sets, thus producing more informative predictions. This paper presents one of the first comprehensive studies where evolutionary algorithms are used to build conformal predictors. More specifically, decision trees evolved using genetic programming are evaluated as conformal predictors. In the experiments, the evolved trees are compared to decision trees induced using standard machine learning techniques on 33 publicly available benchmark data sets, with regard to predictive performance and efficiency. The results show that the evolved trees are generally more accurate, and the corresponding conformal predictors more efficient, than their induced counterparts. One important result is that the probability estimates of decision trees when used as conformal predictors should be smoothed, here using the Laplace correction. Finally, using the more discriminating Brier score instead of accuracy as the optimization criterion produced the most efficient conformal predictions.

I. INTRODUCTION

Conformal prediction is a fairly new framework introduced by Vovk, Gammerman and Shafer in [1]. The overall idea is that the conformal predictors produce set predictions instead of point predictions. In classification, this means that a prediction may consist of one class, several classes or no class at all. The method, however, in the long run guarantees an error rate lower than a predefined significance level; e.g. 0.05. Naturally, an error in this context means that the prediction set did not include the correct class. This property, called *validity*, applies under assumptions slightly weaker than the i.i.d. assumption that is standard in most machine learning scenarios, i.e., that the instances are generated independently from the same distribution.

At the heart of conformal prediction is the *conformity function*, which produces a *conformity score* for each possible prediction (class) of a test instance. The conformity score measures how well a specific prediction (class label) conforms to the training data. The prediction set for a specific test instance will only include classes with conformity scores above a certain threshold, as determined by the significance level.

Conformal prediction was originally designed for an on-line setting, in which test instances are predicted successively,

each prediction being reviewed before the next instance is predicted. In this transductive setting, conformal prediction, at least in theory, requires the learning of a new model for each new test instance to be predicted, which of course may be computationally prohibitive for many applications. Instead, inductive conformal prediction (ICP), which was also introduced in [1], may be used. In ICP, just one model is induced from the training data and then used for predicting all test instances. In the ICP setting, a separate data set (called the *calibration set*) is used for calculating the conformity scores. Consequently, models learned using any machine learning technique, e.g., decision trees, rule sets, neural networks, support vector machines, ensembles etc., can be transformed into ICPs, but it requires a calibration set that was not used for the model generation.

Since validity is guaranteed, the most important criterion for comparing conformal predictors is *efficiency*. A high efficiency, informally, means that the prediction sets contain as few classes as possible; or put in another way, that class predictions are as certain as possible. 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 the strengths associated with the predictions are calculated internally. As a matter of fact, until now, most published studies are either highly mathematical and abstract away from specific machine learning techniques, or use only a very limited number of data sets, thus serving mainly as proofs-of-concept.

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. In this study, we look at decision trees, either induced directly from the data using standard machine learning techniques, or evolved using genetic programming (GP). More specifically, we investigate how different optimization criteria and different ways of producing the probability estimates will affect the resulting models' efficiency as conformal predictors.

II. BACKGROUND

In this section we first describe the conformal prediction framework, before presenting basic decision tree theory and

summarizing the most important related work.

A. The conformal prediction framework

As described in the introduction, the conformal prediction framework [1] is very general and can be used with any learning algorithm. A key component of the framework is the conformity function, which produces a score for each instance and class label pair. When classifying a test instance in ICP, scores are calculated for all possible class labels, and these scores are compared to scores obtained from a calibration set consisting of instances with known labels. Each class is assigned a probability that it does conform to the calibration set, based on the fraction of calibration instances with a lower conformity score. For each test instance, the conformal prediction framework outputs a set of predictions with all class labels having a probability higher than some predetermined *significance level*. This prediction set may contain one, several, or even no class labels. Under 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 specific conformity function used [1].

In ICP, it is very natural to define the conformity function A relative to a trained model M :

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

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.

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

$$p_{\langle \bar{x}, c \rangle} = \frac{|\{s : s \in S \wedge A(s) \leq A(\langle \bar{x}, c \rangle)\}|}{|S|} \quad (2)$$

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

$$P(\bar{x}, \sigma) = \{c : c \in \{c_1, \dots, c_n\} \wedge p_{\langle \bar{x}, c \rangle} > \sigma\} \quad (3)$$

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

Decision trees are very popular since they produce transparent yet fairly accurate models. Furthermore, they are relatively fast to train and require a minimum of parameter tuning. The two most famous decision tree algorithms are C4.5/C5.0 [2] and CART [3].

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 used to divide the data set D into the subsets $\{D_1, \dots, D_n\}$. 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 the impurity 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.

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

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 fraction of training instances belonging to class c_i at the current node t .

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

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

It must be noted, however, that any top-down decision tree algorithm working greedily as described above, in fact is inherently suboptimal since each split is optimized only locally, without considering the global model; see [4].

Although the normal operation of a decision tree is to predict a class label based on an input vector, decision trees can also be used to produce class membership probabilities; in which case they are referred to as *probability estimation trees* (PETs)[5]. 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

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

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

Normally, however, a more refined smoothing technique, called *Laplace estimate* or *Laplace correction* is applied. The main reason 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:

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

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 $1/C$. In order to obtain what they call well-behaved PETs, Provost and Domingo [5] changed the C4.5 algorithm

by turning off both pruning and the collapsing mechanism, resulting in much larger trees. This, together with the use of Laplace estimates, turned out to produce much better PETs; for details see the original paper.

Although evolutionary algorithms may be used mainly for optimization, GP has also been proven capable of producing accurate classifiers. In this context, GP's key asset is probably the very general and quite efficient global search strategy. It is also very straightforward to specify an appropriate representation for the task at hand, just by tailoring the function and terminal sets. Remarkably, GP classification results are normally, at the very least, comparable to results obtained by the more specialized machine learning techniques. In particular, several studies show that evolved decision trees often are more accurate than trees induced by standard decision tree algorithms; see e.g., [6] and [7].

When using tree-based GP to evolve decision trees, each individual in the population is of course a direct representation of a decision tree. Technically, available functions and terminals constitute the literals of the representation language. Functions will typically be logical or relational operators, i.e., the operators in the splits, while the terminals are input variables or constants. Naturally, each leaf would represent a specific classification. Normally, GP classification tries to explicitly optimize accuracy; i.e., the fitness function minimizes some error metric directly related to accuracy. The most obvious choice would be to minimize the number of misclassifications on the training set, or on a specific validation set. When optimizing PETs, it is, however, also possible to use a more informed criterion, explicitly utilizing the probability estimates. The most obvious option is the *Brier score* metric.

For two-class problems, let y_i denote the response variable (class) of instance i , where $y_i = 0$ or 1 . Denote the probability estimate that instance i belongs to class 1 , by p_i . The Brier score is then defined as

$$\text{Brier score} = 2 \sum_{i=1}^N (y_i - p_i)^2 \quad (9)$$

which is exactly twice the sum of squares of the difference between the true class and the predicted probability over all instances. For the multi-class case, the generalized Brier score can be used. Let p_{ij} denote the probability estimate that instance i belongs to class j and y_{ij} be an indicator variable such that $y_{ij} = 1$ if $y_i = j$ and 0 otherwise. Then, the generalized Brier score is defined as

$$\text{Generalized Brier score} = \sum_{i=1}^N \sum_{j=1}^C (y_{ij} - p_{ij})^2 \quad (10)$$

The generalized Brier score is of course reduced to the Brier score when $C = 2$. A high (generalized) Brier score indicates poor predictive performance. Often, the fitness function also contains a length penalty, in order to put pressure on the evolution by rewarding smaller and thus more comprehensible trees.

C. Related work

Conformal prediction is very much 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 at www.alrw.net. Not all of these working papers have been published in scientific journals or proceedings. ICP is introduced in the book [1], and is further developed and analyzed in [8].

It must be noted, however, that all of these papers are highly mathematical and do not consider the implications of specific machine learning techniques. Even when using a specific machine learning technique, the purpose is most often to demonstrate a specific property of the framework. One example is [9] where inductive conformal prediction using neural networks is described in detail.

There are, however, a number of other papers on conformal prediction, typically either using it for a specific application, or evaluating some property, most often the conformity function.

Nguyen et al. use conformal prediction in [10] to identify and observe a moving human or object inside a building. Yang et al. use the outlier measure of a random forest to design a conformity measure, and the resulting predictor is tested on a few medical diagnosis problems [11].

In [12], Devetyarov and Nourtdinov use random forests as on-line and off-line conformal predictors, with the overall purpose to compare the efficiency of three different conformity measures. Bhattacharyya also investigates different conformity functions for random forests, but in an inductive setting [13]. Both these interesting studies, however, use a very limited number of data sets, so they serve mainly as proofs-of-concept.

Conformal prediction studies using evolutionary algorithms are very rare. One exception is Lambrou et al. who use conformal prediction on fuzzy rule sets evolved by a genetic algorithm [14]. In their paper the method is applied on two real-world datasets for medical diagnosis, and the authors point out the importance of comprehensible models.

III. METHOD

The approach presented in this paper is different from Lambrou et al. with regard to both the external representation language (we evolve trees instead of rule sets) and the internal representation language, where we use s-expressions directly mapping the tree, while Lambrou et al. use binary coded chromosomes. In addition, we evaluate different optimization criteria while Lambrou et al. use only training accuracy. Finally, we use a large number of data sets to allow for a more rigorous analysis of the results.

We have previously suggested a GP-based rule extraction algorithm called G-REX (Genetic Rule EXtraction) [15]. Since G-REX is a black-box rule extraction algorithm, it is able to extract rules from arbitrary opaque models, including ensembles. For a summary of the G-REX technique and a number of previous rule extraction studies, see [16]. Although G-REX was initially devised for rule extraction, it has evolved into a more general GP data mining framework, see [17].

Naturally, this includes the ability to evolve classifiers, in a wide variety of representation languages including decision trees and decision lists, directly from the data.

As described in the introduction, the overall purpose is to evaluate evolved decision trees as conformal predictors. In Experiment 1, the ICP framework is demonstrated using decision trees evolved while optimizing accuracy on the training data. In Experiment 2, the evolved decision trees are compared to standard decision trees induced from the data set using the SimpleCart (sCart) and J48 algorithms in the Weka workbench [18]. J48 and sCart are the Weka implementations of C4.5 and CART, respectively. All Weka settings were left at the default values, with the exception that J48 was restricted to binary splits, and that no pruning was allowed for either classifier. In addition, J48 used the Laplace correction when calculating the probabilities. With these settings, all models (i.e., G-REX, J48 and sCart) have identical representation languages. The G-REX representation language is presented using Backus-Naur form in Figure 1 below.

```

F = { if, ==, <, > }
T = { i1, i2, ..., in, c1, c2, ..., cm,  $\mathcal{R}$  }

DTree      :- (if RExp Dtree Dtree) | Class
RExp       :- (ROp ConI ConC) | (== CatI CatC)
ROp        :- < | >
CatI       :- Categorical input variable
ConI       :- Continuous input variable
CatC       :- Categorical attribute value
ConC       :-  $\mathcal{R}$ 
Class      :- c1 | c2 | ... | cm

```

Fig. 1. Representation language

For the actual evolution, G-REX used a variant, called *decision tree injection*, which not only automatically adjusts the size punishment to an appropriate level, but also guides the GP process towards good solutions in the case of an impractically large search space. In this variant, the population of genetic programs is enriched with decision trees induced using a standard technique, here J48. For more details see [19]. G-REX, just like J48, used Laplace corrections. The GP parameters are presented in Table I below.

TABLE I
GP PARAMETERS

Parameter	Value
Algorithm	Decision tree injection
Crossover rate	0.8
Mutation rate	0.01
Population size	500
Generations	50
Creation depth	6
Creation method	Ramped half-and-half
Selection	Tournament 4
Elitism	Yes

Accuracy and AUC are used for evaluating the predictive performance. AUC is the area under the ROC curve, which plots the true positive rate vs. false positive rate at various

threshold settings. AUC measures the ability to rank instances according to how likely they are to belong to the positive class; see e.g. [20]. AUC can be interpreted as the probability of ranking a (random) true positive instance ahead of a (random) false positive; see [21]. It must be noted, however, that comparing predictive performance, as measured by accuracy or AUC, is not the main focus of this study. These results are therefore presented mainly as part of the analysis of the conformal predictors.

When evaluating the efficiency of the conformal predictors, two different metrics are used. Since high efficiency roughly corresponds to a large number of predictions consisting of one single class, *OneC*, i.e., the proportion of predictions that include just one single class, is a natural choice. Similarly, *MultiC* and *ZeroC* are the proportions of predictions consisting of more than one class, and no classes at all, respectively. One way of aggregating these three measures is *AvgC*, which is the average number of classes in the predictions. It must be noted, however, that for minimizing *AvgC*, a prediction with no classes, is actually optimal, which may seem counterintuitive.

In this study, the conformity function is based on the well-known concept of *margin* [22]. 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 $\max_{j=1, \dots, C: j \neq Y} p_i^{c_j}$, 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 :

$$\alpha_i = p_i^Y - \max_{j=1, \dots, C: j \neq Y} p_i^{c_j} \quad (11)$$

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

$$\alpha_i^{c_k} = p_i^{c_k} - \max_{j=1, \dots, C: j \neq k} p_i^{c_j} \quad (12)$$

For the evaluation, 4-fold cross-validation 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 33 data sets used are all publicly available from either the UCI repository [23] or the PROMISE Software Engineering Repository [24].

IV. RESULTS

Experiment 1 illustrates how conformal prediction works on a selected number of datasets. Figure 2 below shows the behavior of evolved decision trees on the *Wisconsin breast cancer* data set. Looking first at the error, it is obvious that this conformal predictor is valid and well-calibrated. For each point in the graph, the measured error is very close to the corresponding significance level. Analyzing the efficiency, *OneC* is over 90% for $\epsilon = 0.05$ and $\epsilon = 0.1$, but declines after that. The explanation is that for this rather easy data set, where the accuracy of the underlying model is approximately 0.93,

the number of empty predictions (ZeroC), quickly starts to rise when the significance level increases. Multiple predictions (MultiC), are on the other hand only present for very low significance levels, and vanishes completely for significance levels larger than 0.1. OneAcc, finally, which shows the accuracy of the singleton predictions, is fairly stable and always higher than the model accuracy.

Fig. 2. Key characteristics for the conformal predictor. Breast-w data set.

Figure 3 presents the same results, but now for the *Diabetes* data set.

Fig. 3. Key characteristics for the conformal predictor. Diabetes data set.

Starting with the error, the conformal predictor is again well-calibrated since the observed errors are very close to the significance levels. This data set is, however, much harder, so the accuracy of the underlying evolved tree model is approximately 0.73. Here, OneC, consequently, is much smaller for low significance levels. As a matter of fact, for $\epsilon = 0.05$, less than 30% of the predictions are singletons, while the rest contain both classes. When the significance level increases, OneC increases and MultiC decreases until $\epsilon = 0.3$, where MultiC is just over 2% and OneC reaches its maximum of almost 98%. For even higher significance levels, OneC

decreases, due to the increasing number of empty predictions. For this data set too, OneAcc is rather stable and always higher than the accuracy of the underlying model.

Table II below shows similar results for six data sets. For most data sets and significance levels, the error rate is very close to (and most often slightly smaller than) 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 ZeroC increases as the significance level increases. Finally, it is of course reassuring to see that OneAcc is almost always higher than the accuracy of the underlying model. This should be expected unless singleton predictions are very rare.

TABLE II
EXPERIMENT 1: GP OPTIMIZING ACCURACY AS CONFORMAL PREDICTOR

breast-w (accuracy .927)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.05$	.052	.924	.063	.013	.958
$\epsilon = 0.1$	.090	.946	.000	.054	.962
$\epsilon = 0.2$	.206	.820	.000	.180	.968
$\epsilon = 0.3$	.279	.742	.000	.258	.971
credit-a (accuracy .836)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.05$	.045	.573	.427	.000	.922
$\epsilon = 0.1$	.094	.810	.190	.000	.886
$\epsilon = 0.2$	.190	.912	.009	.080	.879
$\epsilon = 0.3$	.291	.783	.000	.217	.906
diabetes (accuracy .727)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.05$	.056	.285	.715	.000	.750
$\epsilon = 0.1$	.104	.517	.483	.000	.781
$\epsilon = 0.2$	.195	.828	.172	.000	.760
$\epsilon = 0.3$	.272	.979	.004	.017	.740
heart-c (accuracy .782)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.05$	.043	.245	.755	.000	.809
$\epsilon = 0.1$	.086	.585	.415	.000	.849
$\epsilon = 0.2$	.205	.941	.046	.013	.800
$\epsilon = 0.3$	.248	.901	.016	.083	.815
iono (accuracy .860)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.05$	.057	.470	.530	.000	.884
$\epsilon = 0.1$	.103	.775	.225	.000	.873
$\epsilon = 0.2$	.188	.912	.009	.080	.882
$\epsilon = 0.3$	.274	.818	.000	.182	.891
vote (accuracy .927)	Error	OneC	MultiC	ZeroC	OneAcc
$\epsilon = 0.05$	.028	.837	.163	.000	.968
$\epsilon = 0.1$	.078	.982	.007	.011	.932
$\epsilon = 0.2$	.191	.828	.000	.172	.978
$\epsilon = 0.3$	.308	.706	.000	.294	.982

Table III below shows the performance results for the underlying models in Experiment 2. Comparing accuracies first, we see that the evolved trees are actually more accurate than the trees induced by J48 and sCart. Looking at AUCs, J48 models are generally the most accurate, while sCart models most often have the lowest AUC. Regarding model size, finally, there are fairly small differences when averaging over all data sets or comparing mean ranks. On individual data sets, like CMC or Cylinder, however, the differences are substantial. All in all, despite the fact that pruning was turned off for J48 and sCart, and that GP-Acc, optimizing accuracy, used a setting which is known to produce fairly complex trees, a large majority of the tree sizes must be considered as compact enough to enable analysis and understanding.

To determine if there are any statistically significant differ-

ences, we use the statistical tests recommended by Demšar [25] for comparing several classifiers over a number of data sets; i.e., a Friedman test [26], followed by a Nemenyi post-hoc test [27]. With three algorithms and 33 data sets, the critical distance (for $\alpha = 0.05$) is 0.58, so based on these tests the GP-evolved trees were significantly more accurate than J48 and sCart. Regarding AUC, both GP-Acc and J48 obtained significantly higher AUC than sCart. One possible explanation for this is the fact that sCart used relative frequencies when calculating the probability estimates, while both the GP and J48 used the more robust LaPlace correction.

TABLE III
EXPERIMENT 2: PERFORMANCE RESULTS

Data sets	Accuracy			AUC			Size		
	GP	J48	sCart	GP	J48	sCart	GP	J48	sCart
ar1	.909	.901	.926	.560	.586	.618	12.0	7.0	8.0
ar4	.757	.831	.766	.642	.716	.615	11.5	9.0	9.0
balanceS	.887	.877	.885	.917	.915	.917	46.5	47.5	62.5
breast-w	.927	.927	.943	.965	.956	.944	23.0	21.5	22.0
cmc	.530	.472	.496	.456	.437	.430	67.5	321.5	278.0
colic	.802	.802	.815	.853	.856	.814	31.5	35.0	20.5
credit-a	.836	.810	.836	.906	.901	.852	46.0	64.5	46.0
credit-g	.671	.667	.676	.678	.682	.633	74.5	169.5	97.5
cylinder	.678	.694	.596	.698	.731	.532	64.0	106.5	14.0
diabetes	.727	.719	.715	.745	.739	.706	56.0	48.5	74.5
ecoli	.807	.813	.798	.617	.622	.605	33.0	25.0	30.0
glass	.621	.631	.710	.528	.523	.546	42.5	35.5	28.0
haberman	.712	.677	.709	.667	.577	.607	26.0	45.5	31.5
heart-c	.782	.753	.752	.838	.815	.777	37.0	37.0	29.0
heart-h	.809	.792	.772	.820	.877	.791	25.0	34.0	23.0
heart-s	.811	.807	.800	.850	.855	.792	24.5	37.5	30.5
hepatitis	.806	.781	.774	.704	.789	.712	23.5	19.5	12.5
iono	.860	.872	.857	.853	.868	.840	15.0	14.5	17.5
jEdit4042	.679	.697	.679	.718	.737	.649	23.5	24.0	33.5
jEdit4243	.640	.591	.642	.688	.645	.655	31.5	27.0	61.5
kc1	.847	.835	.832	.764	.774	.608	74.0	138.5	117.0
kc2	.824	.808	.822	.769	.768	.684	28.5	25.5	49.0
kc3	.886	.869	.876	.733	.757	.680	19.0	23.5	24.0
letter	.986	.986	.981	.992	.997	.983	20.0	24.5	29.0
liver	.612	.603	.588	.584	.600	.578	40.0	25.5	43.0
mw1	.891	.876	.891	.674	.671	.641	11.5	20.5	16.5
sonar	.673	.668	.697	.701	.702	.681	19.0	18.5	15.0
spect	.823	.797	.789	.806	.740	.741	28.0	27.0	28.5
spectf	.787	.787	.779	.798	.837	.765	31.0	28.0	21.0
tic-tac-toe	.906	.895	.893	.966	.959	.914	73.0	76.5	86.0
wine	.865	.904	.882	.662	.679	.668	12.0	7.5	8.5
vote	.927	.927	.924	.970	.968	.933	10.0	12.5	22.0
vowel	.872	.856	.867	.867	.874	.894	12.5	11.0	9.5
Mean	.792	.786	.787	.757	.762	.721	33.1	47.5	42.4
Mean Rank	1.55	2.18	2.24	1.73	1.64	2.64	1.91	1.97	2.03

Turning to the specific results for the significance level $\epsilon = 0.05$ in Table IV below, we can confirm that all conformal predictors seem to be valid and well-calibrated. Only for a couple of data sets are the observed error rates actually larger than 0.05. Averaging over all data sets, the conformal predictors are, instead slightly conservative. Analyzing the efficiency results, we immediately see that GP-Acc and J48 are significantly more efficient than sCart, based on either the OneC or the AvgC measure. In addition, J48 and GP-Acc also have a significantly higher OneAcc, i.e., they produce not only more singleton predictions, but the accuracy of the singleton predictions is also higher. It may be noted that although the

average OneAcc for sCart is clearly affected by a couple of data sets where no singleton predictions at all are made, i.e., *ar4* and *hepatitis*, the statistical tests are based on the more robust mean ranks. Comparing J48 to GP-Acc, J48 is slightly more efficient, winning 18 of 33 data sets in a direct comparison.

Looking at individual data sets, OneC of course varies a lot. On some data sets, J48 and GP-Acc will only produce slightly more than 10% singleton predictions, when the error rate must be less than 0.05. On easier data sets, OneC may be as high as 90% or more. This is a very good illustration of the power inherent in conformal prediction; since the classifiers are required to produce predictions that keep the error rate smaller than the significance level, the number of singleton predictions is adjusted accordingly. At this level of significance, OneAcc is actually very poor, i.e., much lower than the accuracy of the underlying model on some data sets, like *ar1* and *ar4*. The reason is of course that for a two-class problem, the 5% erroneous instances consist of incorrect singleton predictions and empty predictions. So, when there are few singleton predictions, a much larger proportion of these may be incorrect, and the conformal predictor does still not exceed the overall error rate.

TABLE IV
EXPERIMENT 2: EFFICIENCY RESULTS ($\epsilon = 0.05$)

Data sets	Error			OneC			AvgC			OneAcc		
	GP	J48	sCart	GP	J48	sCart	GP	J48	sCart	GP	J48	sCart
ar1	.016	.008	.000	.245	.081	.042	1.76	1.92	1.96	.464	.225	.250
ar4	.000	.038	.000	.046	.266	.000	1.95	1.73	2.00	.250	.431	.000
balanceS	.042	.033	.024	.526	.517	.359	1.47	1.48	1.64	.921	.938	.938
breast-w	.056	.056	.046	.947	.940	.894	1.05	1.05	1.10	.946	.945	.954
cmc	.039	.038	.039	.112	.150	.018	2.62	2.65	2.83	.879	.789	.377
colic	.041	.031	.024	.356	.288	.125	1.64	1.71	1.88	.883	.905	.826
credit-a	.043	.036	.030	.596	.569	.266	1.40	1.43	1.73	.917	.929	.877
credit-g	.034	.040	.043	.251	.314	.115	1.75	1.69	1.89	.859	.873	.605
cylinder	.030	.032	.037	.174	.205	.085	1.83	1.80	1.91	.786	.864	.565
diabetes	.043	.038	.033	.268	.253	.124	1.73	1.75	1.88	.790	.820	.742
ecoli	.039	.045	.033	.479	.527	.036	4.10	3.38	6.40	.908	.936	.417
glass	.042	.049	.019	.131	.174	.038	5.47	5.46	6.07	.564	.547	.250
haberman	.016	.032	.016	.124	.165	.101	1.88	1.84	1.90	.852	.808	.785
heart-c	.010	.035	.020	.135	.156	.066	1.86	1.84	1.93	.956	.814	.725
heart-h	.014	.012	.017	.191	.344	.085	1.81	1.66	1.91	.882	.961	.808
heart-s	.022	.020	.041	.207	.270	.160	1.79	1.73	1.84	.920	.925	.700
hepatitis	.019	.006	.000	.135	.167	.000	1.87	1.83	2.00	.427	.482	.000
iono	.040	.047	.037	.437	.606	.463	1.56	1.39	1.54	.909	.925	.895
jEdit4042	.018	.013	.044	.113	.112	.080	1.89	1.89	1.92	.838	.673	.388
jEdit4243	.016	.020	.024	.138	.106	.073	1.86	1.89	1.93	.912	.768	.673
kc1	.046	.046	.045	.619	.630	.156	1.38	1.37	1.84	.926	.927	.711
kc2	.048	.042	.054	.440	.446	.199	1.56	1.55	1.80	.890	.902	.717
kc3	.033	.037	.033	.690	.768	.423	1.31	1.23	1.58	.956	.953	.918
letter	.041	.050	.054	.969	.959	.963	0.97	0.96	0.96	.990	.991	.983
liver	.061	.043	.052	.145	.113	.101	1.86	1.89	1.90	.564	.635	.440
mw1	.040	.041	.037	.757	.728	.362	1.24	1.27	1.64	.948	.946	.800
sonar	.005	.010	.048	.096	.045	.106	1.90	1.96	1.89	.979	.660	.617
spect	.034	.059	.026	.451	.542	.229	1.55	1.46	1.77	.923	.896	.878
spectf	.046	.037	.017	.460	.442	.144	1.54	1.56	1.86	.903	.919	.876
tic-tac-toe	.056	.062	.059	.898	.889	.780	1.10	1.11	1.22	.937	.930	.918
wine	.045	.037	.028	.554	.645	.139	1.65	1.48	2.55	.917	.909	.947
vote	.037	.037	.032	.890	.886	.716	1.11	1.11	1.28	.959	.959	.954
vowel	.011	.015	.044	.350	.391	.461	1.65	1.61	1.54	.969	.920	.900
Mean	.033	.035	.032	.392	.415	.240	1.82	1.78	2.06	.840	.821	.680
Mean Rank				1.64	1.55	2.82	1.67	1.52	2.82	1.61	1.61	2.79

When the significance level is instead $\epsilon = 0.1$, the number of singleton predictions is much higher, see Table V below. Averaging over all data sets, when allowed to commit 10% errors, almost 2 out of 3 predictions from J48 or GP-Acc are singletons. For sCart, the corresponding results is just over 50%. Again, J48 and GP-Acc are significantly more efficient, and have significantly higher OneAcc, compared to sCart.

TABLE V
EXPERIMENT 2: EFFICIENCY RESULTS ($\epsilon = 0.1$)

Data sets	Error			OneC			AvgC			OneAcc		
	GP	J48	sCart	GP	J48	sCart	GP	J48	sCart	GP	J48	sCart
ar1	.074	.091	.099	.958	.942	.967	1.04	0.99	0.97	.923	.940	.933
ar4	.029	.047	.019	.228	.371	.150	1.77	1.63	1.85	.456	.907	.900
balanceS	.076	.080	.075	.804	.812	.811	1.20	1.19	1.19	.906	.901	.912
breast-w	.102	.094	.097	.943	.961	.941	0.94	0.96	0.94	.953	.943	.959
cmc	.105	.091	.086	.272	.254	.093	2.36	2.42	2.63	.673	.698	.517
colic	.117	.095	.054	.821	.775	.353	1.18	1.22	1.65	.858	.877	.820
credit-a	.097	.080	.099	.835	.779	.572	1.17	1.22	1.43	.889	.900	.823
credit-g	.105	.098	.103	.507	.468	.284	1.49	1.53	1.72	.795	.791	.622
cylinder	.093	.096	.078	.406	.416	.220	1.59	1.58	1.78	.734	.770	.648
diabetes	.113	.100	.077	.516	.473	.279	1.48	1.53	1.72	.783	.788	.723
ecoli	.080	.079	.083	.699	.666	.301	2.60	2.43	3.67	.911	.917	.609
glass	.094	.092	.079	.342	.357	.113	4.37	4.60	4.70	.773	.783	.375
haberman	.065	.100	.079	.386	.453	.294	1.61	1.55	1.71	.822	.784	.723
heart-c	.066	.090	.056	.539	.516	.287	1.46	1.48	1.71	.892	.847	.805
heart-h	.054	.074	.102	.467	.721	.524	1.53	1.28	1.48	.874	.899	.812
heart-s	.063	.061	.063	.492	.472	.363	1.51	1.53	1.64	.871	.880	.817
hepatitis	.091	.043	.051	.691	.588	.335	1.31	1.41	1.66	.887	.926	.764
iono	.088	.098	.114	.719	.869	.722	1.28	1.13	1.23	.878	.889	.838
jEdit4042	.077	.074	.098	.313	.376	.233	1.69	1.62	1.77	.786	.758	.557
jEdit4243	.070	.072	.098	.298	.237	.252	1.70	1.76	1.75	.752	.683	.620
kc1	.092	.093	.097	.823	.808	.541	1.18	1.19	1.46	.888	.885	.803
kc2	.103	.102	.086	.833	.807	.515	1.17	1.19	1.48	.876	.873	.826
kc3	.085	.074	.085	.923	.903	.841	1.07	1.10	1.16	.910	.919	.900
letter	.093	.098	.091	.915	.907	.925	0.92	0.91	0.92	.991	.994	.983
liver	.125	.098	.087	.299	.261	.237	1.70	1.74	1.76	.579	.652	.634
mw1	.094	.103	.089	.975	.911	.849	1.02	1.09	1.15	.903	.887	.899
sonar	.087	.061	.101	.337	.224	.264	1.66	1.78	1.74	.731	.727	.622
spect	.064	.100	.083	.670	.727	.636	1.33	1.27	1.36	.908	.863	.871
spectf	.101	.075	.066	.658	.595	.371	1.34	1.41	1.63	.855	.874	.812
tic-tac-toe	.098	.117	.095	.978	.975	.925	1.00	0.99	1.02	.910	.899	.925
wine	.079	.079	.101	.750	.849	.581	1.34	1.14	1.49	.887	.892	.881
vote	.099	.109	.087	.938	.928	.991	0.94	0.94	0.99	.959	.956	.922
vowel	.067	.061	.089	.583	.613	.767	1.42	1.39	1.23	.887	.898	.894
Mean	.086	.086	.084	.634	.637	.501	1.50	1.49	1.65	.839	.855	.780
Mean Rank				1.58	1.82	2.61	1.64	1.70	2.67	1.79	1.58	2.64

The explanation for the very poor efficiency obtained by sCart is the way the probability estimates are calculated using raw relative frequencies instead of Laplace smoothing. When not using smoothing, all nodes for which the relative frequencies coincide will result in the same p -value. If instead smoothing is employed, the class probability will be pushed closer to the *a priori* probability. This reasoning does not only explain why smoothing increases the fine-grainedness of the conformity function, but also shows that extreme scores due to very few observations in a node, can be avoided. Finally, it should be noted that the recommendation of using a smoothing operator is in accordance with the findings in [5] and [28] for maximizing the AUC. The explanation is that both the AUC metric and conformal prediction rely on models' ability to rank instances. With this in mind, it is no surprise that the sCart models also obtained significantly lower AUCs than J48 and

G-REX, even if the accuracy was comparable to J48.

Table VI below summarizes the performance result for Experiment 3 by showing values and ranks averaged over all data sets. It must be noted that sCart results were excluded from this experiment based on the poor efficiency results in Experiment 2. Instead, GP optimized on Brier Score were added. Although there are few statistically significant differences, the results are quite clear. GP-Acc does indeed produce the most accurate trees, but also the trees evolved using the Brier fitness are more accurate than J48. When considering the ordering ability, as measured by AUC, GP-Brier is the best choice, followed by J48 and GP-Acc. Trees evolved using GP-Brier are, however, the most complex when comparing ranks. The very high value for mean size obtained by J48 is due to the induction of huge trees on a couple of data sets, as seen in Table III above.

TABLE VI
EXPERIMENT 3: PERFORMANCE RESULTS

	Accuracy		AUC		Size	
	Mean	Mean Rank	Mean	Mean Rank	Mean	Mean Rank
GP-Acc	.792	1.79	.757	2.24	33.1	1.73
GP-Brier	.790	1.83	.764	1.76	34.9	2.41
J48	.786	2.38	.762	2.00	47.5	1.86

Table VII presents the efficiency results for Experiment 3. For $\epsilon = 0.05$, GP-Brier is the most efficient setup, clearly outperforming both J48 and GP-Acc, when comparing mean ranks. For $\epsilon = 0.1$, however, the results are a bit mixed, even if the two GP setups are both more efficient than J48.

TABLE VII
EXPERIMENT 3: EFFICIENCY RESULTS

ϵ	Error	OneC		AvgC		OneAcc	
		Mean	Mean rank	Mean	Mean rank	Mean	Mean rank
$\epsilon = 0.05$	Mean						
GP-Acc	.033	.392	2.15	1.82	2.18	.840	2.00
GP-Brier	.037	.430	1.71	1.78	1.73	.822	2.03
J48	.035	.415	2.14	1.78	2.09	.821	1.97
$\epsilon = 0.1$	Mean						
GP-Acc	.086	.634	1.70	1.50	1.91	.839	2.18
GP-Brier	.087	.656	1.94	1.46	1.82	.859	1.91
J48	.086	.637	2.36	1.49	2.27	.855	1.91

Table VIII below, finally, shows pairwise comparisons between the three setups. With 33 data sets, a standard one-sided sign test where ties are divided equally, requires 24 wins for significance with $\alpha = 0.05$. It must be noted that in this comparison, no adjustment is made to avoid the increased risk of making a type I error due to multiple comparisons. Numbers representing statistically significant differences for single comparisons are underlined and bold. When comparing the setups head-to-head, GP-Brier is significantly more efficient than J48 with regard to OneC, and substantially more efficient with regard to AvgC. For $\epsilon = 0.05$, GP-Brier is also substantially more efficient than GP-Acc.

TABLE VIII
EXPERIMENT 3: EFFICIENCY - PAIRWISE COMPARISON

	$\epsilon = 0.05$		$\epsilon = 0.1$	
	AvgC	OneC	AvgC	OneC
	W/T/L	W/T/L	W/T/L	W/T/L
GP-Acc vs. GP-Brier	12/0/21	13/0/20	12/0/21	18/0/15
GP-Acc vs. J48	15/0/18	15/0/18	15/0/18	18/0/15
GP-Brier vs. J48	21/0/12	22/1/10	21/0/12	24/0/9

V. CONCLUSIONS

We have in this paper evaluated evolved decision trees as inductive conformal predictors. This is one of the first comprehensive studies where evolutionary algorithms are the basis of conformal predictors. The overall purpose was to analyze the effect of the optimization criterion while comparing the performance of the evolved decision trees to trees induced using standard techniques like J48 and sCart. In the analysis, we focused on the efficiency but predictive performance, measured as accuracy, AUC and OneAcc, was also investigated. From the experiments, it is obvious that all conformal predictors are valid and well-calibrated. Most importantly, the evolved trees are generally more accurate, and the corresponding conformal predictors more efficient, than their J48 and sCart counterparts. One key result identified is that when using decision trees as conformal predictors, the probability estimates should be smoothed, here using the Laplace correction. Finally, using the Brier score as the optimization criterion produced the most efficient conformal predictions. Naturally, this is a very strong argument for using evolutionary algorithms in conformal prediction since one obvious reason for using genetics based machine learning in the first place is the possibility to choose the optimization criterion depending on the application.

ACKNOWLEDGMENT

This work was supported by the Swedish Foundation for Strategic Research through the project High-Performance Data Mining for Drug Effect Detection (IIS11-0053) and the Knowledge Foundation through the project Big Data Analytics by Online Ensemble Learning (20120192).

REFERENCES

- [1] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. Springer-Verlag New York, Inc., 2005.
- [2] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.
- [3] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Chapman & Hall/CRC, January 1984.
- [4] S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining and Knowledge Discovery*, vol. 2, pp. 345–389, 1998.
- [5] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Mach. Learn.*, vol. 52, no. 3, pp. 199–215, 2003.
- [6] A. Tsakonas, "A comparison of classification accuracy of four genetic programming-evolved intelligent structures," *Information Sciences*, vol. 176, no. 6, pp. 691 – 724, 2006.

- [7] C. Bojarczuk, H. Lopes, and A. Freitas, "An innovative application of a constrained-syntax genetic programming system to the problem of predicting survival of patients," in *Genetic Programming*, ser. Lecture Notes in Computer Science, C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, Eds. Springer Berlin / Heidelberg, 2003, vol. 2610, pp. 11–59.
- [8] V. Vovk, "Conditional validity of inductive conformal predictors," *Journal of Machine Learning Research - Proceedings Track*, vol. 25, pp. 475–490, 2012.
- [9] H. Papadopoulos, "Inductive conformal prediction: Theory and application to neural networks," *Tools in Artificial Intelligence*, vol. 18, pp. 315–330, 2008.
- [10] K. Nguyen and Z. Luo, "Conformal prediction for indoor localisation with fingerprinting method," *Artificial Intelligence Applications and Innovations*, pp. 214–223, 2012.
- [11] 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.
- [12] D. Devetyarov and I. Nourtdinov, "Prediction with confidence based on a random forest classifier," *Artificial Intelligence Applications and Innovations*, pp. 37–44, 2010.
- [13] S. Bhattacharyya, "Confidence in predictions from random tree ensembles," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 71–80.
- [14] 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.
- [15] U. Johansson, R. König, and L. Niklasson, "Rule extraction from trained neural networks using genetic programming," in *ICANN, supplementary proceedings*, 2003, pp. 13–16.
- [16] U. Johansson, *Obtaining Accurate and Comprehensible Data Mining Models: An Evolutionary Approach*. PhD-thesis. Institute of Technology, Linköping University, 2007.
- [17] R. König, U. Johansson, and L. Niklasson, "G-REX: A versatile framework for evolutionary data mining," in *ICDM Workshops*. IEEE Computer Society, 2008, pp. 971–974.
- [18] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, June 2005.
- [19] R. König, U. Johansson, T. Löfström, and L. Niklasson, "Improving gp classification performance by injection of decision trees," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.
- [20] T. Fawcett, "Using rule sets to maximize roc performance," in *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM'01*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 131–138.
- [21] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.
- [22] 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.
- [23] A. Asuncion and D. J. Newman, "UCI machine learning repository," 2007.
- [24] 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>
- [25] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
- [26] 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.
- [27] P. B. Nemenyi, *Distribution-free multiple comparisons*. PhD-thesis. Princeton University, 1963.
- [28] C. Ferri, P. Flach, and J. Hernandez-Orallo, "Improving the auc of probabilistic estimators trees," in *Proc. of the 14th European Conference on Artificial Intelligence*, vol. 2837. Springer, 2003, pp. 121–132.