Proceedings of Machine Learning Research 105:1–18, 2019 Conformal and Probabilistic Prediction and Applications

Interpretable and Specialized Conformal Predictors

Ulf Johansson ULF.JOHANSSON@JU.SE
Tuwe Löfström TUWE.LOFSTROM@JU.SE
Dept. of Computer Science and Informatics, Jönköping University, Sweden

Henrik Boström BOSTROMH@KTH.SE
School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden

Cecilia Sönströd CECILIA.SONSTROD@HB.SE
Dept. of Information Technology, University of Borås, Sweden

Editor: Alex Gammerman, Vladimir Vovk, Zhiyuan Luo and Evgueni Smirnov

Abstract

In real-world scenarios, interpretable models are often required to explain predictions, and to allow for inspection and analysis of the model. The overall purpose of oracle coaching is to produce highly accurate, but interpretable, models optimized for a specific test set. Oracle coaching is applicable to the very common scenario where explanations and insights are needed for a specific batch of predictions, and the input vectors for this test set are available when building the predictive model. In this paper, oracle coaching is used for generating underlying classifiers for conformal prediction. The resulting conformal classifiers output valid label sets, i.e., the error rate on the test data is bounded by a preset significance level, as long as the labeled data used for calibration is exchangeable with the test set. Since validity is guaranteed for all conformal predictors, the key performance metric is efficiency, i.e., the size of the label sets, where smaller sets are more informative. The main contribution of this paper is the design of setups making sure that when oracle-coached decision trees, that per definition utilize knowledge about test data, are used as underlying models for conformal classifiers, the exchangeability between calibration and test data is maintained. Consequently, the resulting conformal classifiers retain the validity guarantees. In the experimentation, using a large number of publicly available data sets, the validity of the suggested setups is empirically demonstrated. Furthermore, the results show that the more accurate underlying models produced by oracle coaching also improved the efficiency of the corresponding conformal classifiers.

Keywords: Interpretability, Decision trees, Classification, Oracle coaching, Conformal prediction

1. Introduction

As AI is increasingly used not only for decision support, but also automated decision making, trust in the resulting decisions or recommendations becomes vital. Consequently, how to make AI solutions trustworthy is today a key question addressed by researchers from many disciplines. AI trustworthiness is also strongly manifested in the two vibrant areas Explainable AI (XAI) and Fairness, Accountability and Transparency (FAT). Professional associations such as the ACM (2017), FAT/ML (Diakopoulos et al., 2017) and IEEE (2017)

have proposed guidelines and frameworks for FAT/XAI, incorporating demands to be placed on AI solutions, as well as evaluation criteria for explainability and FAT. Specifically, humans interacting with, or affected by, AI must be able to make informed judgments about when to trust the system.

Consequently, interpretability is currently recognized as a key property of trustworthy predictive models. Only interpretable models make it possible to understand individual predictions, without the usage of specialized, and often very complex, explanation modules. In addition, with interpretable models, inspection and analysis of the model itself becomes straightforward. The importance of interpretable models is, of course, not new. In fact, it has been present in the AI discourse since the era of expert systems, and it is also the focus of recent high-impact publications within machine learning, such as the LIME framework (Ribeiro et al., 2016).

The DARPA Research Programme on Explainable AI (2016), targets the development of new AI solutions that produce "more explainable models, while maintaining a high level of learning performance (e.g., prediction accuracy)". Similarly, The FAT/ML Principles for Accountable Algorithms and a Social Impact Statement for Algorithms (Diakopoulos et al., 2017), include both *explainability* and *accuracy* as vital components of accountable algorithms. It is also interesting to note that one guiding question from the FAT/ML Principles for Accountable Algorithms is: "How confident are the decisions output by your system?" Thus, accountability puts demands on not only explainability and accuracy, but also requires an ability to, at the very least, report uncertainty. In fact, the ability for an algorithm to somehow reason about its own competence, specifically about confidence in individual recommendations, is deemed to be extremely valuable. So, the goal becomes algorithms producing explainable and accurate models, that are able to assess and clearly communicate a confidence measure for each prediction or recommendation. In our opinion, the prediction with confidence framework is uniquely well positioned to meet these demands.

While the potential benefits of prediction with confidence, i.e., conformal prediction and Venn predictors, are huge, these techniques have been slowly adopted by the machine learning community. This is despite the very solid theoretical foundation, and a fairly large number of papers published in top-tier machine learning venues. One potential explanation for this is that the techniques remain somewhat inaccessible; published papers are often theoretical in nature, typically lacking recommendations and best practices for usage in real-world data analytics projects. In addition, published case-studies and applications often focus on safety-critical scenarios, thus limiting the impact. In reality, the guarantees provided by conformal and Venn prediction would be extremely valuable also in many other situations. Finally, it must be noted that conformal and Venn predictors are not algorithms that can be used off-the-shelf, in a plug-and-play fashion. Instead, they contain a large number of parameters, and often some modifications are necessary based on, for instance, the kind of underlying model used and other constraints. Consequently, it becomes important to identify and analyze common situations where prediction with confidence could provide an advantage, in order to not only demonstrate the potential, but also provide recommendations for the usage.

With this in mind, we study one quite specific, but also very common scenario, in this paper. First of all, the predictive model must, for some reason, be interpretable, i.e., it should be possible to both follow the logic behind individual predictions, and to manually

inspect the model in order to obtain insights by analyzing overall relationships. In addition, in the targeted situation, we assume that the predictive model is explicitly built for a certain set of test instances, and that these input vectors are available when generating the model. While this initially may appear to be a very specialized scenario, it is in fact extremely common. One example is when recipients of a marketing campaign are found from analyzing historical data of customers. Here, the company would of course have access to the profiles of the potential recipients when generating the model. Another example is when using predictive analytics for in silico modeling in drug discovery, i.e., when using a data-driven approach to finding molecules with certain properties, e.g., non-toxic. In this scenario, the model would be built using a labeled data set consisting of molecule descriptions and a target variable showing if the molecule is toxic or not. However, the intended usage is of course to predict the toxicity of molecules not used when generating the model, and a set of such signatures would often be available already when the model is built, simply because the purpose of the modeling in the first place was to predict the toxicity of these specific substances.

2. Background

2.1. Decision trees

Decision trees are relatively accurate and comprehensible models with learning algorithms requiring a minimum of parameter tuning. The two most notable decision tree algorithms are C4.5/C5.0 (Quinlan, 1993) and CART (Breiman et al., 1984). Decision trees are inherently capable of reporting class membership probabilities; in which case they are referred to as *Probability Estimation Trees (PETs)* (Provost and Domingos, 2003). For PETs, the most straightforward way to obtain a class probability is to use the relative frequencies,

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

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

Intuitively, though, a leaf containing many training instances is a better estimator of class membership probabilities, so often, a *Laplace estimate* is used instead,

$$p_i^{c_j} = \frac{1 + g(i,j)}{C + \sum_{k=1}^{C} g(i,k)}\,. \tag{2}$$

It should be noted that the usage of a Laplace estimator also will lead to more fine-grained probability estimates since the number of instances is considered.

2.2. Oracle coaching

In predictive modeling, the available labeled data is normally split into different parts, where each part is used for a different purpose.

- The *training set*, $Z_{tr} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ is the part of the data set used to build one or more models.

- The *test set*, $Z_{te} = \{(\mathbf{x}_{l+1}, y_{l+1}), \ldots, (\mathbf{x}_{l+r}, y_{l+r})\}$ is used to evaluate the model(s).

In real-world scenarios, once the model building is finished, it would be applied to yet another data set, where correct target values are not known, for the actual predictions. This data set is sometimes referred to as the *production set*. So, the main usage of the test set is actually to estimate the performance on the production set. In order to make these estimations unbiased, the test set can not be used in any way when building the model.

Oracle coaching, introduced by Johansson and Niklasson (2009) is a methodology aimed at producing highly accurate interpretable models, with a specific set of production instances in mind. When empirically evaluating oracle coaching, a test set will normally be used to emulate the production set in the usual manner, i.e., using the correct target values to evaluate predictive performance. Oracle coaching utilizes the input vectors of the test set $Z_{te} = \{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+q}\}$, during model construction, but of course not the corresponding targets. This will produce a model that is specialized on the test set at hand. Consequently, oracle coaching requires that the test input vectors are available at the model building stage, as is often the case when performing batch predictions. Thus, oracle coaching is applicable to many but not all predictive tasks. Specifically, it can not be used in streaming scenarios where test instances arrive one by one.

The key idea of oracle coaching is to utilize a very accurate but opaque model, called the *oracle*, in the model building process. The simplest procedure for building an interpretable model using oracle coaching is:

1. the oracle is constructed, using the training set $Z_{tr} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$
2. the oracle is applied to the test set $Z_{te} = \{(\mathbf{x}_{l+1}, y_{l+1}), \ldots, (\mathbf{x}_{l+r}, y_{l+r})\}$, to produce the *oracle test set*, $Z_{te}^{O} = \{(\mathbf{x}_{l+1}, \hat{y}_{l+1}), \ldots, (\mathbf{x}_{l+r}, \hat{y}_{l+r})\}$, where $\hat{y}_i$ is the oracle prediction for input vector $\mathbf{x}_i$
3. a transparent model is constructed, using Z_{te}^{O} as training data

Oracle coaching resembles semi-supervised and transductive learning schemes, but explicitly focuses on situations where the final model must be interpretable. While the process where a weaker transparent model is built using a stronger opaque model is similar to *rule extraction*, the purpose is different. In oracle coaching, the intended result is a very accurate interpretable model, customized for predicting a specific set of instances. The way to achieve this is by utilizing knowledge about the corresponding input vectors, and having a stronger model coaching a weaker, but interpretable, model.

In previous studies, see e.g., Johansson and Niklasson (2009), it was shown that using only the oracle data set to generate the interpretable model could result in an overly specific model. This problem was, however, overcome by combining the original training data with the oracle data set. Oracle coaching has been evaluated for classification, using different combinations of oracle and transparent models, e.g., random forests and RBF neural network ensembles coaching decision trees and ordered rule sets (Johansson et al., 2012). It has also been demonstrated to work on real-world data sets. As an example, Crielaard and Papapetrou (2018) successfully used oracle coaching on five real-world data sets constructed from electronic health records to obtain explainable predictions of adverse drug effects.

2.3. Conformal prediction

Conformal prediction (Vovk et al., 2005) provides predictions with guarantees. Conformal predictors are exactly *valid*, which means that given a user-defined significance level $\epsilon \in (0, 1)$, the predictor is guaranteed to get an error rate equal to ϵ in the long run. In order to achieve this, a conformal classifier relies on a *nonconformity function* - a function $f(z, \zeta) \to \mathbb{R}$ that scores instance $z = (x, y)$ based on how well it complies with a sequence of instances $\zeta = z_1, ..., z_n$, in such a way that nonconforming (i.e., uncommon or unlikely) instances achieve higher values than more conforming instances. The standard approach is to base the nonconformity function on the predictions made by a regular classifier, often referred to as the *underlying model*. Nonconformity then becomes

$$f(z_i, \zeta) = \Delta[h(x_i), y_i], \tag{3}$$

where h is a classifier induced from ζ and Δ is a function measuring the prediction errors of h. Within classification, the hinge error function is commonly used as Δ,

$$\Delta\left[h\left(\boldsymbol{x}_i\right), y_i\right] = 1 - h(y_i \mid \boldsymbol{x}_i), \tag{4}$$

where $h(y_i \mid \boldsymbol{x}_i)$ is the probability estimate for the correct class y_i produced by h for input vector $\boldsymbol{x}_i$.

There are a few different ways to apply conformal prediction and one of the more popular is *inductive conformal prediction* (ICP) (Papadopoulos et al., 2002). One of its main benefits is the low computational cost, since only one underlying model has to be trained. ICP works in the following way:

1. Divide the training set Z_{tr} into two disjoint sets
   - A *proper training set* Z_t.
   - A *calibration set* Z_c, where $|Z| = q$.
2. Train a classifier h using the proper training set Z_t
3. Let $\alpha_1, ..., \alpha_q = f(z_i, Z_t) : z_i \in Z_c$.

The conformal predictor is defined and applied to a new test object x_j from Z_{te} in the following way:

1. Decide which significance level $\epsilon \in (0, 1)$ to use.
2. For each class label $\tilde{y} \in Y$:
   (a) Tentatively label x_j as $(x_j, \tilde{y})$.
   (b) Let $\alpha_j^{\tilde{y}} = f[(x_j, \tilde{y}), Z_t]$.
   (c) Calculate $p_j^{\tilde{y}}$ as

$$p_j^{\tilde{y}} = \frac{\left|\left\{z_i \in Z_c : \alpha_i > \alpha_j^{\tilde{y}}\right\}\right|}{q+1} + \theta_j \frac{\left|\left\{z_i \in Z_c : \alpha_i = \alpha_j^{\tilde{y}}\right\}\right| + 1}{q+1}, \tag{5}$$

   where $\theta_j \sim U[0, 1]$.

(d) Let the label set $\Gamma_j^\epsilon = \left\{\tilde{y} \in Y : p_j^{\tilde{y}} > \epsilon\right\}$ be the conformal prediction.

The probability that the true label y_j is not part of Γ_j^ϵ is ϵ, i.e., an error (meaning that $y_j \notin \Gamma_j^\epsilon$) occurs with probability ϵ.

3. Method

The overall purpose of this study is to design and evaluate setups combining oracle coaching with conformal prediction in order to create highly accurate, but interpretable, models. In the experimentation, random forests (Breiman, 2001) with 300 trees are used as the oracles, and CART (Breiman et al., 1984) decision trees, as implemented in the MatLab statistics toolbox, are used as the interpretable models.

Before generating transparent models, the random forest is trained using the proper training set Z_t. The resulting model (the oracle) is then applied to both the calibration instances and the test instances, thus creating predictions for these two data sets. Consequently, we have access to four different labeled data sets:

- The *proper training* set, Z_t, i.e., original training input vectors with corresponding true targets.
- The *proper calibration* set, Z_c, which is the original calibration data set, i.e., original calibration input vectors, with the corresponding true targets.
- The *oracle calibration* set, Z_c^O, consisting of the input vectors from the calibration set, but with the predictions from the oracle for these instances as the target.
- The *oracle test* set, Z_{te}^O, consisting of the input vectors of the test instances, with predictions from the opaque model as target values.

When oracle coaching was originally suggested, different combinations of data were described using made-up names, combining the terms *induction* (generating models from training data), *extraction* (building models on training data but with oracle outputs as targets) and *explanation* (generating models on test data with oracle outputs as targets). In this study, extraction is not included.

In ICP, the calibration data and the test data must be exchangeable, i.e., calibration and test instances must be treated identically. This would, however, not be the case for a majority of the setups suggested in the oracle coaching framework. As an example, in the *explanation* setup, i.e., where a transparent model is trained on Z_{te}^O, the test instances would have been part of the training set (although with predictions from the oracle as targets), but the calibration instances would not. Consequently, the nonconformity scores from the calibration set would not come from the same distribution as the nonconformity scores from the test set. Using that setup, the nonconformity scores from the test set would most certainly be lower, since the decision tree was optimized on these instances. This means that the validity guarantees of the conformal predictor would be lost.

One key contribution of this paper is therefore to suggest modified setups in the oracle coaching framework, so that when conformal predictors are built on top of these models, the conformal predictors are valid. For the conformal prediction, the calibration is always

performed on the proper calibration set Z_c. The difference between the three setups evaluated is therefore only which data that is used for generating the decision tree. We first describe the three setups below, before showing how two setups utilizing oracle coaching can be used as underlying models for valid conformal predictors.

- **Induction (I):** The decision tree is induced using the proper training set Z_t.
- **Explanation (X)**: The decision tree is induced on the union of the oracle calibration set and the oracle test set, i.e., $Z_c^O \cup Z_{te}^O$.
- **Indanation (IX)**: Uses the union of the proper training set, the oracle calibration set and the oracle test set for induction, i.e., $Z_t \cup Z_c^O \cup Z_{te}^O$.

We now argue that in the X and IX setups, as defined and described above, calibration and test instances are treated in an identical way. First of all, both the calibration instances and the test instances are kept hidden from the oracle. Second, all calibration instances and test instances are used to induce the final transparent model, but with the outputs from the oracle as targets. So, in these setups the nonconformity values of the calibration and test sets, e.g., the hinge errors, must come from the same distribution, as long as the calibration and test sets are exchangeable to start with. Consequently, these setups will produce valid conformal predictors.

In the experimentation, we will demonstrate empirically that these setups produce well-calibrated conformal classifiers. Most importantly, we will compare the efficiency of oracle coached trees, to trees induced using training data only. For this comparison, we will use the following two metrics:

- **AvgC**: The average number of class labels in the prediction sets, i.e., a direct measure of how good the conformal predictor is at rejecting class labels.
- **OneC**: The proportion of all predictions that are singletons. The motivation for OneC is that singleton predictions are the most informative, and often what is sought for.

In addition, we will look at the accuracy of the singleton predictions (OneAcc). Here we would expect OneAcc to be higher than the accuracy of the underlying model, i.e., showing that we have reason to trust the singleton predictions more than an arbitrary prediction from the underlying model.

The study consists of two experiments. In Experiment 1, we use only two-class data sets, while Experiment 2 uses multi-class data sets. It may be argued that conformal classification is uniquely suitable for the the multi-class scenario, since label sets can be very valuable in many situations, e.g., if the main purpose is to rule out certain hypotheses.

In the two-class scenario, the nonconformity measure used is the hinge error, see (4), i.e., a pattern is considered to be nonconforming when the probability estimate for the true class is low. In the multi-class scenario, we also evaluate a nonconformity measure based on the margin error,

$$\Delta\left[h\left(\boldsymbol{x}_i\right), y_i\right] = \underset{\tilde{y} \in Y \wedge \tilde{y} \neq y_i}{\arg\max}\ h(\tilde{y} \mid \boldsymbol{x}_i) - h(y_i \mid \boldsymbol{x}_i), \tag{6}$$

So, a nonconforming example has a low probability estimate for the true class label and/or a high probability estimate for an incorrect class label. Johansson et al. (2017) evaluated these two nonconformity functions for neural networks. The results showed that the usage of hinge errors as nonconformity function rejected the most labels, while the margin-based nonconformity function lead to the highest number of singleton predictions.

To be interpretable, we want the conformal trees to be fixed after the calibration step, thus allowing inspection and analysis. We argue that the resulting models are very informative, due to the label set predictions and the guaranteed validity. There is, however, one subtlety that must be considered. In conformal prediction, as seen in Section 2.3, ties among the nonconformity scores are normally handled by including a random number when calculating the p-values, see (5). When using trees, all instances in the same leaf will of course have the same nonconformity value, and if the randomized tie-breaker is used, this will sometimes lead to different label sets for different test instances falling in the same leaf, severely hampering the interpretability. At the same time, including all ties will lead to quite conservative trees, i.e., sacrificing efficiency. Using Laplace estimates, different leaves will rarely have the same nonconformity values, so in general there will be exactly one leaf, corresponding to the p-values just around the significance level, that suffers from the problem with different label sets. We investigate this issue further when showing some sample trees in Section 4.

To start with accurate decision trees, the trees were optimized using an internal cross-validation over *minleaf* values between 1 and 20. The *minleaf* parameter controls the depth of the tree, since every leaf node must contain at least *minleaf* training instances. For the actual evaluation, 10x4-fold cross-validation was employed. Model sizes are measured using the total number of nodes. The data sets used are all publicly available from either the UCI (Bache and Lichman, 2013) or the PROMISE (Sayyad Shirabad and Menzies, 2005) repositories.

4. Results

4.1. Experiment 1

In Figure 1, two conformal trees induced for the diabetes data set are shown. The first is induced using ordinary induction (I) and the second using indanation (IX). The significance level is $\epsilon = 0.2$. As can be seen, some of the leaves predict both classes while other leaves predict a single class. Both trees have one leaf with one of the class labels in bold (the lowest leaf for (a) and the second lowest leaf for (b)), which is the label that has been excluded for some test instances but not for others (see Section 3). In the example, IX results in more accurate trees (I: 75.5% vs IX: 78.1%), smaller leaves on average (AvgC for I: 1.2 vs IX: 1.1) and more instances being predicted as singleton predictions (OneC for I: 81.3% vs IX: 89.1%).

```
x2<127.5
|   x8<28.5
|   |   y = {1} {68/6}
|   x8>=28.5
|   |   x2<99.5
|   |   |   y = {1} {18/1}
|   |   x2>=99.5
|   |   |   x7<0.2915
|   |   |   |   y = {1, 2} {11/0}
|   |   |   x7>=0.2915
|   |   |   |   y = {1, 2} {16/0}
x2>=127.5
|   x6<30
|   |   y = {1} {19/8}
|   x6>=30
|   |   y = {1, 2} {60/16}
```

(a) Induction

```
x2<154.5
|   x6<29.85
|   |   y = {1} {64/8}
|   x6>=29.85
|   |   x2<127.5
|   |   |   x8<30.5
|   |   |   |   y = {1} {40/6}
|   |   |   x8>=30.5
|   |   |   |   x2<103.5
|   |   |   |   |   y = {1} {13/3}
|   |   |   |   x2>=103.5
|   |   |   |   |   y = {1, 2} {10/0}
|   |   x2>=127.5
|   |   |   y = {1, 2} {32/9}
x2>=154.5
|   y = {2} {33/6}
```

(b) Indanation

Figure 1: (a) Induction and (b) Indanation for the diabetes data set

Table 1 below shows the predictive performance and the model sizes. Starting with the accuracy, we see that the random forest, as expected, is the most accurate.

Table 1: Predictive performance and tree sizes for two-class data sets

	Accuracy				AUC				Size		
	RF	I	IX	X	RF	I	IX	X	I	IX	X
colic	.830	.842	.843	.834	.878	.846	.853	.850	5.4	7.4	7.2
creditA	.870	.845	.866	.863	.929	.895	.908	.908	8.1	17.9	12.2
diabetes	.756	.741	.751	.751	.818	.744	.771	.755	12.8	25.9	19.1
german	.668	.689	.677	.683	.608	.517	.562	.557	3.4	32.3	22.2
haberman	.686	.705	.688	.701	.641	.597	.607	.607	5.0	24.2	12.4
heartC	.811	.744	.802	.793	.896	.783	.839	.821	8.8	19.1	10.9
heartH	.817	.789	.803	.807	.885	.779	.820	.812	5.0	11.2	7.5
heartS	.818	.739	.796	.792	.893	.764	.838	.827	7.0	14.5	11.9
hepati	.819	.784	.817	.810	.817	.566	.644	.622	1.9	5.6	4.5
iono	.929	.879	.915	.918	.974	.886	.930	.920	7.6	15.3	8.4
je4042	.724	.703	.714	.708	.802	.737	.755	.742	7.9	17.3	9.5
je4243	.661	.638	.656	.657	.718	.654	.690	.682	11.5	33.0	17.6
kc1	.742	.734	.737	.741	.671	.600	.617	.606	6.2	34.6	19.9
kc2	.779	.778	.779	.784	.824	.743	.755	.732	5.3	10.6	6.4
kc3	.859	.852	.859	.864	.755	.553	.572	.530	3.4	5.4	2.2
liver	.702	.625	.655	.657	.730	.614	.646	.650	11.3	29.9	16.2
pc4	.893	.871	.886	.885	.918	.849	.806	.738	12.4	19.0	12.9
sonar	.763	.675	.737	.737	.857	.690	.763	.760	4.9	10.5	8.1
spect	.881	.887	.885	.890	.637	.502	.514	.500	1.2	2.9	1.0
spectf	.804	.777	.792	.795	.832	.657	.635	.565	5.4	6.5	3.3
transfusion	.718	.748	.737	.740	.671	.682	.691	.686	8.1	26.2	14.9
ttt	.947	.893	.929	.914	.992	.949	.974	.954	47.7	72.7	53.1
wbc	.954	.906	.938	.935	.982	.933	.951	.946	8.5	12.7	7.4
vote	.864	.837	.853	.857	.911	.865	.875	.842	15.0	20.6	8.2
Mean	**.804**	**.778**	**.796**	**.797**	**.818**	**.725**	**.751**	**.734**	**8.9**	**19.8**	**12.3**
Mean Rank	**1.73**	**3.46**	**2.48**	**2.33**	**1.13**	**3.63**	**2.13**	**3.13**	**1.21**	**3.00**	**1.79**

Interestingly enough, the two oracle-coached setups are almost as accurate as the random forest, clearly outperforming the induced tree. Regarding AUC, the differences are larger, and there is a clear ordering; RF, IX, X and I. When comparing model sizes, it must be noted that most trees are very compact, often containing fewer than 10 nodes.

In order to determine any statistically significant differences, we performed Friedman tests (Friedman, 1937), followed by Bergmann-Hommel's dynamic procedure (Bergmann and Hommel, 1988) to establish all pairwise differences at $\alpha = 0.05$. From this analysis, we see that for accuracy, I is significantly worse than all other setups. For AUC, the random forest had a significantly higher AUC than IX, which in turn significantly outperformed I and X. Regarding model sizes, all differences are significant. From this analysis, it is obvious that oracle coaching will produce stronger trees, compared to if they are induced directly from the data.

Turning to the conformal predictors, Table 2 below shows the error rates. From these results, where the empirical error rates are very close to the significance levels, it is obvious that IX and X produce valid conformal predictors.

Table 2: Empirical error rates - two-class data sets

	$\epsilon = 0.2$			$\epsilon = 0.1$			$\epsilon = 0.05$			$\epsilon = 0.01$		
	I	IX	X	I	IX	X	I	IX	X	I	IX	X
colic	.205	.206	.208	.107	.106	.106	.054	.051	.055	.009	.010	.013
creditA	.203	.202	.204	.104	.107	.105	.052	.058	.051	.012	.009	.010
diabetes	.205	.203	.201	.103	.097	.101	.052	.048	.050	.012	.010	.009
german	.205	.202	.198	.105	.097	.097	.057	.049	.049	.011	.012	.012
haberman	.198	.195	.187	.107	.102	.092	.049	.054	.042	.011	.009	.008
heartC	.205	.208	.211	.098	.102	.093	.054	.050	.043	.009	.011	.009
heartH	.192	.194	.189	.093	.095	.095	.045	.048	.049	.010	.010	.009
heartS	.188	.184	.196	.096	.094	.087	.049	.044	.041	.009	.008	.009
hepati	.196	.188	.186	.088	.085	.094	.041	.046	.045	.006	.013	.007
iono	.204	.208	.199	.110	.104	.102	.050	.056	.050	.010	.012	.009
je4042	.196	.196	.200	.099	.096	.091	.057	.045	.046	.009	.009	.008
je4243	.185	.188	.199	.087	.096	.094	.047	.050	.044	.011	.014	.006
kc1	.201	.203	.204	.103	.101	.102	.050	.052	.052	.010	.010	.011
kc2	.198	.198	.201	.099	.101	.095	.043	.052	.047	.009	.011	.010
kc3	.206	.203	.208	.106	.100	.099	.054	.052	.046	.010	.011	.011
liver	.206	.212	.196	.103	.112	.099	.051	.059	.050	.010	.012	.012
pc4	.197	.199	.199	.099	.099	.100	.051	.049	.047	.010	.009	.011
sonar	.205	.195	.186	.096	.076	.085	.052	.044	.042	.013	.009	.013
spect	.199	.200	.203	.094	.090	.090	.041	.047	.048	.009	.010	.010
spectf	.186	.193	.191	.090	.093	.087	.045	.051	.047	.007	.010	.007
transfusion	.197	.204	.204	.106	.097	.103	.055	.049	.049	.012	.009	.008
ttt	.205	.199	.202	.103	.103	.096	.052	.053	.049	.011	.010	.009
wbc	.203	.207	.219	.104	.106	.111	.055	.050	.049	.007	.012	.008
vote	.216	.212	.209	.112	.106	.102	.055	.050	.052	.014	.012	.009
Mean	**.200**	**.200**	**.200**	**.101**	**.099**	**.097**	**.050**	**.050**	**.048**	**.010**	**.011**	**.010**

Investigating the efficiency, Table 3 below shows AvgC. While the differences are rather small in absolute numbers, there is a clear ordering for all significance levels except $\epsilon = 0.01$, i.e., IX is the most efficient followed by X. Statistical tests show that for the significance

levels $\epsilon \in \{0.01, 0.05, 0.1\}$, IX obtained a significantly lower AvgC than I and X. For $\epsilon = 0.2$, IX and X were significantly more efficient than I.

Table 3: Efficiency AvgC - two-class data sets

	$\epsilon = 0.2$			$\epsilon = 0.1$			$\epsilon = 0.05$			$\epsilon = 0.01$		
	I	IX	X	I	IX	X	I	IX	X	I	IX	X
colic	0.93	0.92	0.93	1.24	1.23	1.23	1.62	1.61	1.58	1.92	1.93	1.91
creditA	0.91	0.89	0.89	1.12	1.08	1.08	1.38	1.32	1.34	1.85	1.86	1.85
diabetes	1.13	1.11	1.12	1.46	1.44	1.45	1.69	1.69	1.71	1.92	1.93	1.94
german	1.30	1.28	1.28	1.64	1.62	1.62	1.82	1.80	1.81	1.97	1.96	1.96
haberman	1.23	1.25	1.27	1.56	1.55	1.61	1.78	1.76	1.81	1.96	1.96	1.97
heartC	1.13	1.00	1.02	1.47	1.34	1.41	1.71	1.65	1.70	1.95	1.93	1.94
heartH	1.08	1.02	1.02	1.44	1.36	1.39	1.70	1.62	1.66	1.93	1.92	1.93
heartS	1.20	1.05	1.05	1.52	1.40	1.41	1.74	1.68	1.68	1.95	1.94	1.94
hepati	1.08	1.02	1.04	1.49	1.41	1.41	1.74	1.69	1.71	1.96	1.95	1.93
iono	0.88	0.84	0.86	1.06	0.98	0.99	1.37	1.19	1.24	1.84	1.76	1.81
je4042	1.26	1.23	1.22	1.57	1.57	1.59	1.76	1.77	1.78	1.96	1.95	1.95
je4243	1.43	1.37	1.36	1.71	1.66	1.67	1.85	1.82	1.84	1.96	1.95	1.97
kc1	1.15	1.15	1.14	1.53	1.51	1.53	1.77	1.75	1.76	1.95	1.95	1.95
kc2	1.05	1.04	1.05	1.33	1.35	1.37	1.62	1.63	1.66	1.92	1.92	1.93
kc3	0.91	0.90	0.90	1.19	1.18	1.21	1.54	1.52	1.58	1.90	1.90	1.91
liver	1.42	1.33	1.36	1.70	1.65	1.66	1.85	1.82	1.83	1.97	1.96	1.96
pc4	0.88	0.87	0.88	1.06	1.04	1.04	1.23	1.29	1.38	1.44	1.55	1.74
sonar	1.34	1.22	1.24	1.65	1.61	1.61	1.81	1.78	1.79	1.96	1.95	1.95
spect	0.90	0.90	0.90	1.19	1.19	1.19	1.60	1.57	1.59	1.91	1.91	1.92
spectf	1.09	1.06	1.08	1.39	1.41	1.50	1.64	1.64	1.72	1.93	1.93	1.95
transfusion	1.13	1.12	1.12	1.40	1.42	1.42	1.65	1.67	1.68	1.93	1.93	1.94
ttt	0.85	0.83	0.84	1.02	0.96	1.00	1.15	1.07	1.16	1.38	1.33	1.46
wbc	0.85	0.83	0.83	0.99	0.94	0.94	1.24	1.12	1.21	1.82	1.76	1.81
vote	0.90	0.89	0.90	1.12	1.12	1.14	1.37	1.37	1.47	1.75	1.71	1.87
Mean	**1.08**	**1.05**	**1.05**	**1.37**	**1.33**	**1.35**	**1.61**	**1.58**	**1.61**	**1.88**	**1.87**	**1.90**
Mean rank	**2.83**	**1.42**	**1.75**	**2.46**	**1.29**	**2.25**	**2.33**	**1.29**	**2.38**	**2.21**	**1.46**	**2.33**

Table 4 below shows the OneC results. As expected, these are very similar to AvgC. For two-class problems, the only difference between the two metrics is that AvgC rewards the ability to exclude labels, even if it results in empty label sets. Statistical testing shows that IX is significantly more efficient than I and X, for all ϵ-values but 0.2, where there are no significant differences.

Table 4: Efficiency OneC - two-class data sets

	$\epsilon = 0.2$			$\epsilon = 0.1$			$\epsilon = 0.05$			$\epsilon = 0.01$		
	I	IX	X	I	IX	X	I	IX	X	I	IX	X
colic	.922	.921	.918	.761	.769	.766	.380	.392	.421	.076	.073	.090
creditA	.914	.892	.889	.876	.925	.917	.622	.680	.660	.155	.141	.148
diabetes	.866	.887	.880	.544	.560	.546	.312	.313	.285	.076	.066	.056
german	.698	.719	.717	.355	.384	.384	.182	.202	.191	.035	.039	.039
haberman	.775	.752	.733	.441	.449	.386	.222	.236	.187	.040	.045	.033
heartC	.858	.927	.913	.531	.663	.591	.292	.347	.301	.051	.070	.060
heartH	.904	.927	.936	.559	.640	.605	.295	.379	.335	.069	.078	.067
heartS	.796	.910	.913	.477	.599	.591	.257	.318	.316	.048	.060	.064
hepati	.906	.923	.921	.514	.591	.587	.256	.314	.292	.043	.054	.070
iono	.883	.840	.858	.917	.958	.951	.634	.803	.761	.160	.235	.194
je4042	.744	.771	.776	.434	.434	.411	.241	.230	.217	.042	.047	.046
je4243	.575	.625	.639	.293	.343	.325	.149	.180	.162	.040	.052	.029
kc1	.854	.855	.864	.472	.486	.471	.229	.248	.240	.046	.050	.047
kc2	.920	.934	.928	.670	.651	.629	.378	.371	.339	.081	.081	.073
kc3	.908	.899	.904	.806	.816	.788	.462	.480	.417	.096	.098	.085
liver	.585	.673	.638	.300	.350	.335	.149	.179	.172	.028	.038	.036
pc4	.880	.871	.876	.935	.962	.958	.770	.711	.618	.564	.445	.264
sonar	.665	.785	.755	.348	.393	.391	.189	.222	.211	.042	.050	.047
spect	.897	.898	.895	.804	.800	.801	.403	.433	.411	.086	.088	.079
spectf	.899	.920	.899	.607	.593	.501	.357	.357	.276	.073	.069	.052
transfusion	.864	.874	.876	.603	.578	.584	.354	.329	.317	.070	.066	.059
ttt	.846	.829	.844	.947	.931	.948	.847	.927	.838	.625	.673	.543
wbc	.854	.832	.825	.954	.942	.942	.754	.873	.778	.177	.243	.195
vote	.897	.890	.897	.877	.881	.859	.632	.631	.532	.255	.289	.129
Mean	**.830**	**.848**	**.846**	**.626**	**.654**	**.636**	**.390**	**.423**	**.387**	**.124**	**.131**	**.104**
Mean rank	**2.21**	**1.83**	**1.96**	**2.33**	**1.40**	**2.27**	**2.33**	**1.29**	**2.38**	**2.21**	**1.46**	**2.33**

Looking at the OneAcc results in Table 5 below, it is reassuring to see that they are generally substantially higher than the accuracy of the models in Table 1. This means that we can trust the singleton predictions from the conformal predictors more than an arbitrary prediction from the tree. Comparing the setups, we see that IX, despite having the highest proportion of singleton instances, as seen by the OneC results in Table 4, also has the highest OneAcc. Here, the statistical tests show that for $\epsilon \in \{0.1, 0.2\}$ IX and X had a significantly higher OneAcc than I. When $\epsilon = 0.05$, the only statistically significant difference was between IX and I, and for $\epsilon = 0.01$, there were no significant differences.

Table 5: OneAcc - two-class data sets

	$\epsilon = 0.2$			$\epsilon = 0.1$			$\epsilon = 0.05$			$\epsilon = 0.01$		
	I	IX	X	I	IX	X	I	IX	X	I	IX	X
colic	.858	.861	.856	.861	.863	.863	.859	.869	.870	.878	.858	.857
creditA	.872	.894	.895	.882	.884	.886	.917	.915	.922	.925	.936	.932
diabetes	.764	.773	.774	.810	.827	.816	.832	.845	.823	.845	.849	.836
german	.707	.719	.724	.705	.747	.747	.689	.759	.741	.692	.690	.699
haberman	.745	.741	.745	.758	.772	.762	.778	.771	.775	.732	.802	.742
heartC	.769	.818	.803	.815	.846	.843	.814	.855	.857	.824	.839	.856
heartH	.796	.820	.821	.833	.852	.843	.846	.874	.852	.861	.873	.872
heartS	.765	.817	.805	.800	.844	.853	.811	.860	.870	.806	.863	.855
hepati	.792	.826	.818	.828	.856	.840	.841	.854	.845	.866	.759	.898
iono	.902	.943	.933	.892	.924	.926	.921	.935	.934	.939	.950	.951
je4042	.737	.745	.743	.773	.779	.778	.765	.805	.789	.781	.805	.823
je4243	.679	.699	.689	.702	.720	.709	.688	.723	.727	.724	.735	.779
kc1	.764	.763	.764	.783	.792	.783	.783	.788	.785	.793	.792	.770
kc2	.799	.801	.797	.852	.845	.849	.885	.860	.860	.883	.863	.866
kc3	.872	.886	.876	.871	.877	.876	.882	.892	.891	.898	.890	.874
liver	.647	.685	.692	.657	.680	.705	.656	.673	.706	.635	.695	.672
pc4	.912	.920	.915	.894	.897	.896	.934	.931	.923	.982	.980	.958
sonar	.692	.752	.754	.723	.806	.784	.723	.803	.800	.693	.819	.711
spect	.893	.891	.890	.888	.891	.891	.898	.892	.884	.898	.885	.872
spectf	.798	.803	.801	.851	.844	.827	.874	.858	.829	.907	.852	.856
transfusion	.773	.768	.771	.824	.833	.824	.844	.851	.847	.832	.867	.865
ttt	.936	.967	.946	.910	.947	.927	.940	.946	.943	.983	.985	.984
wbc	.934	.954	.946	.920	.948	.943	.928	.947	.943	.959	.950	.957
vote	.870	.885	.882	.874	.881	.882	.914	.920	.902	.945	.959	.928
Mean	**.803**	**.822**	**.818**	**.821**	**.840**	**.836**	**.834**	**.851**	**.847**	**.845**	**.854**	**.851**
Mean rank	**2.58**	**1.54**	**1.88**	**2.75**	**1.38**	**1.88**	**2.46**	**1.58**	**1.96**	**2.17**	**1.79**	**2.04**

Summarizing the two-class experiment, the two oracle-coached setups were shown to be valid, and more efficient than the standard choice of inducing a tree using training data only. In particular the IX setup, i.e., combining oracle data with standard training data, was the most successful, significantly outperforming I regarding efficiency, and also obtaining the highest OneAcc. While IX also produced the largest trees, it must be noted that these trees in general were small enough to be considered comprehensible.

4.2. Experiment 2

When considering multi-class problems, the glass data set is used in Figure 2 to compare both induction (I) vs indanation (IX) as well as the performance of margin and hinge as nonconformity measures. The trees induced using I have identical conditions for hinge (a) and margin (b), since they have been defined from the same underlying trees, the only differences are the predictions in the leaves. The same holds for IX in (c) and (d). In a comparison between I and IX, IX results in more accurate trees (I: 59.8% vs IX: 65.4%), smaller leaves on average and much more instances being predicted as singleton predictions, regardless of nonconformity measure. This clearly demonstrates the benefit of utilizing oracle data.

```
x4<1.78
|   x3<2.785
|   |   y = {2, 4, 5} {10/2}
|   x3>=2.785
|   |   x7<8.385
|   |   |   y = {2, 3} {14/4}
|   |   x7>=8.385
|   |   |   x2<13.4
|   |   |   |   x1<1.51819
|   |   |   |   |   y = {1, 2} {5/0}
|   |   |   |   x1>=1.51819
|   |   |   |   |   y = {1, 2} {4/1}
|   |   |   x2>=13.4
|   |   |   |   y = {1, 3} {11/1}
x4>=1.78
|   y = {4, 6} {10/3}
```

(a) Induction with Hinge

```
x4<1.78
|   x3<2.785
|   |   y = {2, 4, 5} {10/2}
|   x3>=2.785
|   |   x7<8.385
|   |   |   y = {2} {14/5}
|   |   x7>=8.385
|   |   |   x2<13.4
|   |   |   |   x1<1.51819
|   |   |   |   |   y = {1} {5/1}
|   |   |   |   x1>=1.51819
|   |   |   |   |   y = {1, 2} {4/1}
|   |   |   x2>=13.4
|   |   |   |   y = {1, 2, 3, 4, 5, 6} {11/1}
x4>=1.78
|   y = {2, 4, 6} {10/0}
```

(b) Induction with Margin

```
x8<0.335
|   x7<8.325
|   |   y = {2} {15/4}
|   x7>=8.325
|   |   x3<2.785
|   |   |   x2<13.71
|   |   |   |   y = {2, 4} {7/2}
|   |   |   x2>=13.71
|   |   |   |   y = {2, 5} {3/0}
|   |   x3>=2.785
|   |   |   x1<1.51692
|   |   |   |   y = {2, 3} {3/1}
|   |   |   x1>=1.51692
|   |   |   |   y = {1, 2} {18/5}
x8>=0.335
|   y = {6} {8/2}
```

(c) Indanation with Hinge

```
x8<0.335
|   x7<8.325
|   |   y = {2} {15/4}
|   x7>=8.325
|   |   x3<2.785
|   |   |   x2<13.71
|   |   |   |   y = {1, 2, 3, 4, 5, 6} {7/0}
|   |   |   x2>=13.71
|   |   |   |   y = {1, 2, 3, 4, 5, 6} {3/0}
|   |   x3>=2.785
|   |   |   x1<1.51692
|   |   |   |   y = {1, 2, 3} {3/0}
|   |   |   x1>=1.51692
|   |   |   |   y = {1, 2} {18/3}
x8>=0.335
|   y = {6} {8/2}
```

(d) Indanation with Margin

Figure 2: Induction with Hinge (a) and Margin (b) as well as Indanation with Hinge (c) and Margin (d) for the glass data set

When considering the different nonconformity measures, we can see that hinge results in smaller leaves on average (AvgC for I: 2.2 (a) and 2.5 (b) vs IX: 1.4 (c) and 2.2 (d)) but most often in fewer instances being predicted as singleton predictions (OneC for I: 0.0% (a) and 35.5% (b) vs IX: 65.6% (c) and 61.7% (d)). These differences are most clearly demonstrated when comparing the results for I using hinge (a) and margin (b), where hinge results in no leaves (and no instances) being predicted as a singleton prediction. Instead, all leaves predict two or three classes. For I with hinge (a), the tie determining whether a class label should be included or not occurred between two leaves rather than in a particular leaf, accounting for the absence of bold class labels in that tree. For I with margin (b), on the other hand, the leaf with the ties have three different classes that are included for some test instances but not for all. So for any particular test instance falling into that leaf, the number of predicted classes may range from three to six.

Turning to the evaluation over multiple data sets, Table 6 below shows the performance of the underlying trees. While these results are generally consistent with the results from the two-class experiment, it may be noted that IX here outperforms X on accuracy, and that the trees are substantially larger. The bigger models must be expected, simply because more leaves are necessary just to be able to make predictions of all possible labels. From the statistical testing, we see that for accuracy, I is significantly less accurate than all three other setups. In addition, the random forest is significantly more accurate than X. Regarding the

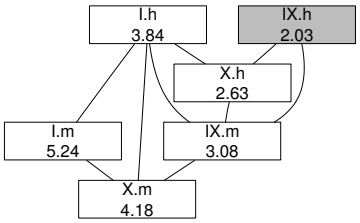
ranking ability, IX and RF obtained significantly higher AUC than I and X. For model sizes, IX trees were significantly larger than trees produced by I and X.

Table 6: Predictive performance and tree sizes for multiclass data sets

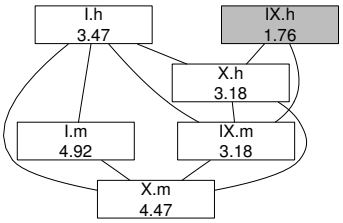
	Accuracy				AUC				Size		
	RF	I	IX	X	RF	I	IX	X	I	IX	X
balance	.838	.773	.810	.812	.940	.848	.896	.893	27.9	55.4	39.2
cars	.965	.936	.958	.950	.997	.985	.989	.986	46.8	65.5	53.4
cmc	.527	.524	.525	.528	.711	.699	.698	.695	40.5	192.3	109.0
cool	.940	.938	.937	.937	.994	.980	.984	.978	12.0	20.1	11.9
glass	.696	.590	.670	.651	.891	.772	.799	.774	11.3	21.5	13.9
heat	.980	.976	.981	.980	.998	.994	.996	.992	28.7	32.0	27.6
image	.971	.939	.966	.960	.999	.989	.995	.992	45.1	69.5	43.2
iris	.906	.626	.908	.910	.985	.782	.945	.943	3.2	6.0	5.7
steel	.757	.692	.733	.727	.928	.870	.882	.866	93.2	142.4	95.9
tae	.397	.358	.397	.396	.576	.523	.543	.548	2.2	17.3	13.4
user	.897	.864	.888	.885	.984	.954	.957	.949	18.6	21.2	15.6
wave	.853	.757	.811	.816	.967	.904	.923	.910	126.2	165.0	96.7
vehicle	.734	.661	.717	.710	.920	.854	.889	.875	43.1	74.9	49.2
whole	.693	.685	.709	.708	.496	.491	.492	.495	6.2	3.9	2.2
wine	.942	.825	.908	.915	.995	.875	.946	.950	5.3	8.7	7.8
wineR	.632	.553	.608	.599	.802	.711	.730	.717	45.2	153.5	78.9
wineW	.663	.535	.634	.563	.844	.709	.748	.709	523.1	850.5	103.0
vowel	.891	.641	.839	.798	.569	.886	.949	.944	89.6	150.0	98.5
yeast	.591	.538	.581	.583	.571	.772	.788	.777	47.1	126.9	78.7
Mean	**.783**	**.706**	**.767**	**.759**	**.851**	**.821**	**.850**	**.842**	**63.9**	**114.5**	**49.6**
Mean Rank	**1.37**	**3.89**	**2.16**	**2.58**	**1.32**	**3.53**	**2.11**	**3.05**	**1.42**	**2.95**	**1.63**

Investigating the error rates of all setups in the top part of Table 7, presented at the end of the paper, shows that all setups are empirically valid. In the bottom part of the same table, OneAcc results are presented. Again it must be noted that OneAccs generally are higher than the corresponding accuracies from the underlying trees. Comparing the setups, two patterns emerge; first of all, setups utilizing hinge as the nonconformity generally have higher OneAccs. Second, the oracle-coached setups, in particular IX, again clearly outperforms I.

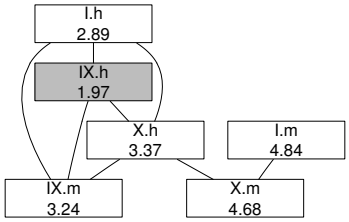
The detailed efficiency results are presented in Table 8, at the end of the paper, while Figures 3 and 4 below, show the results from statistical testing of AvgC and OneC results, respectively. In the graphs, setups not obtaining significantly different efficiencies are connected. Here it must be noted that the tests will be rather weak since we evaluate all-in-all six setups using only 19 data sets. With this in mind, we use $\alpha = 0.1$. While there are some minor variations between different ϵ-values, there are two important results: (i) The oracle-coached setups, in particular IX, are most often significantly more efficient than I and (ii) for AvgC, the nonconformity function based on the hinge error clearly outperforms the margin-based, while the opposite is true for OneC.



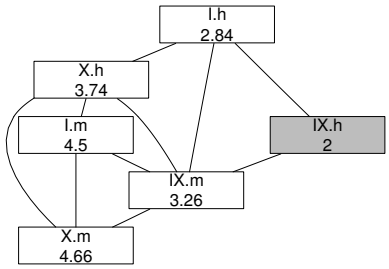
(a) $\epsilon = 0.2$



(b) $\epsilon = 0.1$

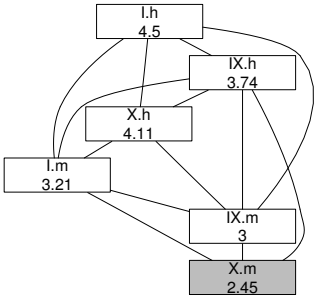


(c) $\epsilon = 0.05$

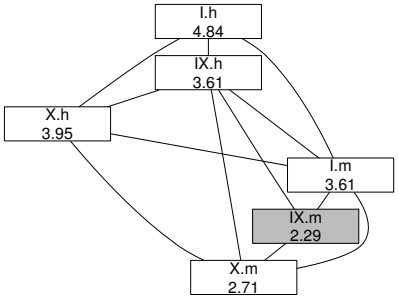


(d) $\epsilon = 0.01$

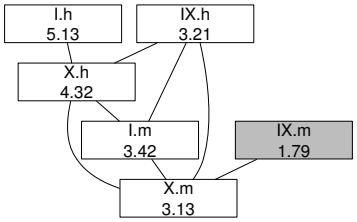
Figure 3: Efficiency (AvgC): Significance tests with $\alpha = 0.1$.



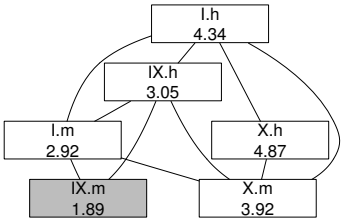
(a) $\epsilon = 0.2$



(b) $\epsilon = 0.1$



(c) $\epsilon = 0.05$



(d) $\epsilon = 0.01$

Figure 4: Efficiency (OneC): Significance tests with $\alpha = 0.1$.

5. Concluding remarks

In this paper, it has been shown how oracle coaching can be used for producing interpretable and accurate underlying models for conformal predictors, tailored for a specific test set. Specifically, two different setups making it possible to utilize knowledge about test set input vectors, without sacrificing the validity guarantees provided by conformal prediction, were suggested and evaluated.

The results from an extensive investigation using both two-class and multi-class data sets clearly show that the utilization of oracle-coached underlying models will produce significantly more efficient conformal predictors, compared to standard induction. In addition, it was demonstrated that for multi-class problems, a nonconformity function based on the hinge error lead to the rejection of the most labels, while a margin-based nonconformity function produced the most singleton predictions. This confirmed the results from a previous study with neural networks as underlying models.

The overall result of combining oracle coaching with conformal prediction is a very informative model; a standard decision tree, optimized for the test set at hand, and with valid label sets instead of single classes in the leaves.

Acknowledgments

This work was supported by the Swedish Knowledge Foundation (DATAKIND 20190194), by the Swedish Governmental Agency for Innovation Systems (Airflow 2018-03581) and by Region Jönköping (DATAMINE HJ 2016/874-51).

References

ACM. New statement on algorithmic transparency and accountability by ACM U.S. Public Policy Council, 2017. URL https://techpolicy.acm.org/?p=6156.

Kevin Bache and Moshe Lichman. UCI machine learning repository, 2013.

Beate Bergmann and Gerhard Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. In *Multiple Hypotheses Testing*, pages 100–115. Springer, 1988.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984. ISBN 0412048418.

L. Crielaard and P. Papapetrou. Explainable predictions of adverse drug events from electronic health records via oracle coaching. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 707–714, Nov 2018. doi: 10.1109/ICDMW.2018.00108.

Darpa. Explainable Artificial Intelligence (XAI), 2016. URL https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf.

N. Diakopoulos, S. Friedler, M. Arenas, S. Barocas, M. Hay, B. Howe, H. V. Jagadish, K. Unsworth, A. Sahuguet, S. Venkatasubramanian, C. Wilson, and B. Zevenbergen C. Yu. *Principles for Accountable Algorithms and a Social Impact Statement for Algorithms*. FAT/ML, 2017. URL http://www.fatml.org/resources/principles-for-accountable-algorithms.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of American Statistical Association*, 32:675–701, 1937.

IEEE. The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. Ethically aligned design: A vision for prioritizing human well-being with autonomous and intelligent systems, 2017. URL http://standards.ieee.org/develop/indconn/ec/autonomous_systems.html.

Ulf Johansson and Lars Niklasson. Evolving decision trees using oracle guides. In *CIDM*, pages 238–244. IEEE, 2009.

Ulf Johansson, Cecilia Sönströd, Tuve Löfström, and Henrik Boström. Obtaining accurate and comprehensible classifiers using oracle coaching. *Intell. Data Anal.*, 16(2):247–263, 2012.

Ulf. Johansson, Henrik. Linusson, Tuwe. Lfstrm, and Henrik. Bostrm. Model-agnostic nonconformity functions for conformal classification. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2072–2079, 2017.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Machine Learning: ECML 2002*, pages 345–356. Springer, 2002.

Foster Provost and Pedro Domingos. Tree induction for probability-based ranking. *Mach. Learn.*, 52(3):199–215, 2003.

J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993. ISBN 1558602380.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2.

J. Sayyad Shirabad and T.J. Menzies. The PROMISE Repository of Software Engineering Databases. School of IT and Engineering, Univ. of Ottawa, Canada, 2005.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag New York, Inc., 2005.

Table 7: Empirical error rates and OneAcc - multiclass data sets

	$\epsilon = 0.2$						$\epsilon = 0.1$						$\epsilon = 0.05$						$\epsilon = 0.01$					
Error rate	Hinge			Margin			Hinge			Margin			Hinge			Margin			Hinge			Margin		
	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X
balance	.201	.195	.190	.202	.196	.192	.099	.099	.096	.098	.101	.094	.050	.046	.049	.050	.050	.048	.008	.008	.008	.007	.009	.009
cars	.195	.197	.202	.195	.197	.201	.094	.095	.099	.093	.095	.099	.045	.049	.048	.045	.048	.048	.009	.009	.008	.010	.009	.008
cmc	.201	.195	.200	.199	.197	.201	.095	.100	.102	.100	.099	.103	.049	.050	.051	.049	.049	.052	.010	.011	.009	.010	.010	.010
cool	.192	.197	.197	.189	.193	.196	.097	.100	.096	.099	.099	.097	.050	.051	.051	.048	.051	.049	.010	.012	.009	.010	.012	.010
glass	.190	.199	.200	.186	.198	.199	.094	.093	.098	.086	.101	.101	.050	.048	.046	.042	.047	.050	.008	.014	.010	.007	.011	.011
heat	.203	.208	.208	.206	.208	.205	.098	.102	.105	.102	.103	.105	.051	.049	.049	.052	.049	.049	.009	.011	.010	.010	.010	.011
image	.201	.199	.202	.201	.201	.202	.102	.100	.101	.102	.099	.101	.052	.050	.050	.051	.050	.049	.009	.010	.010	.009	.010	.011
iris	.197	.198	.209	.193	.183	.191	.097	.105	.107	.104	.099	.094	.055	.052	.057	.057	.057	.051	.009	.011	.007	.010	.013	.011
steel	.192	.199	.201	.197	.198	.200	.097	.104	.103	.097	.102	.103	.048	.051	.049	.049	.054	.052	.011	.010	.010	.010	.011	.011
tae	.182	.202	.193	.179	.195	.189	.089	.108	.091	.086	.103	.095	.049	.048	.040	.037	.058	.046	.011	.007	.011	.007	.007	.007
user	.197	.192	.192	.192	.192	.186	.102	.095	.096	.098	.095	.098	.056	.053	.047	.057	.053	.044	.013	.010	.009	.015	.010	.011
wave	.199	.198	.194	.198	.197	.195	.097	.099	.097	.098	.097	.097	.051	.049	.050	.051	.051	.050	.010	.010	.009	.010	.010	.009
vehicle	.203	.201	.202	.203	.204	.203	.102	.096	.101	.100	.095	.098	.048	.046	.048	.048	.044	.048	.008	.009	.009	.009	.008	.009
whole	.191	.199	.192	.196	.190	.189	.095	.101	.090	.098	.098	.094	.051	.051	.042	.052	.049	.050	.010	.012	.009	.012	.009	.010
wine	.201	.217	.206	.199	.220	.208	.101	.107	.099	.102	.108	.104	.057	.052	.048	.052	.058	.048	.014	.013	.010	.010	.013	.008
wineR	.203	.199	.204	.202	.201	.202	.099	.096	.101	.100	.098	.100	.048	.049	.049	.051	.049	.051	.009	.009	.009	.010	.011	.010
wineW	.200	.196	.196	.197	.196	.197	.101	.097	.097	.100	.098	.096	.051	.050	.050	.051	.047	.047	.009	.010	.010	.009	.008	.008
vowel	.204	.191	.202	.199	.192	.198	.096	.096	.097	.098	.095	.097	.048	.048	.047	.049	.045	.046	.010	.011	.011	.011	.009	.009
yeast	.202	.198	.200	.200	.203	.201	.100	.101	.098	.098	.100	.099	.052	.049	.050	.048	.051	.050	.009	.010	.009	.011	.010	.009
Mean	**.197**	**.199**	**.199**	**.196**	**.198**	**.198**	**.098**	**.100**	**.099**	**.098**	**.099**	**.099**	**.051**	**.050**	**.048**	**.049**	**.051**	**.049**	**.010**	**.010**	**.009**	**.010**	**.010**	**.010**
OneAcc	Hinge			Margin			Hinge			Margin			Hinge			Margin			Hinge			Margin		
	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X
balance	.801	.830	.830	.796	.829	.830	.867	.891	.898	.858	.878	.884	.896	.922	.919	.888	.913	.915	.957	.943	.947	.958	.954	.929
cars	.976	.983	.979	.977	.985	.980	.955	.974	.967	.954	.974	.967	.953	.963	.956	.953	.963	.956	.991	.990	.992	.989	.990	.991
cmc	.689	.672	.657	.643	.644	.636	.828	.768	.720	.734	.716	.699	.905	.835	.779	.819	.784	.743	.972	.931	.906	.946	.914	.899
cool	.973	.980	.968	.974	.978	.968	.961	.966	.956	.960	.967	.955	.950	.949	.949	.950	.949	.952	.989	.986	.992	.991	.987	.992
glass	.695	.730	.714	.680	.715	.691	.743	.770	.728	.732	.724	.694	.811	.727	.630	.847	.814	.754						
heat	.990	.994	.986	.991	.993	.986	.987	.990	.984	.988	.991	.984	.984	.988	.983	.984	.988	.982	.990	.989	.989	.989	.990	.987
image	.977	.990	.983	.977	.990	.983	.961	.982	.975	.962	.982	.975	.953	.974	.967	.952	.974	.967	.990	.990	.988	.988	.988	.987
iris	.727	.924	.921	.728	.925	.926	.821	.927	.929	.803	.927	.931	.887	.935	.933	.851	.932	.936	.976	.973	.975	.938	.964	.968
steel	.787	.784	.776	.764	.775	.765	.875	.856	.837	.835	.836	.815	.925	.925	.920	.885	.879	.873	.959	.979	.980	.960	.966	.975
tae	.476	.345	.419	.534	.390	.395	.500	.344	.403	.563	.340	.429	.500	.313	.467	.667	.333	.500	1.000	1.000			.000	
user	.901	.915	.908	.900	.917	.910	.899	.909	.906	.899	.909	.904	.929	.932	.931	.930	.934	.929	.945	.975	.931	.956	.964	.964
wave	.783	.819	.827	.784	.819	.827	.854	.868	.865	.851	.868	.865	.888	.904	.889	.886	.901	.889	.922	.930	.927	.919	.935	.921
vehicle	.758	.775	.770	.741	.765	.759	.835	.854	.838	.819	.845	.829	.884	.903	.887	.872	.895	.880	.959	.945	.977	.933	.946	.943
whole	.719	.713	.720	.710	.715	.726	.711	.671	.694	.675	.690	.702	.722	.660	.810	.735	.695	.692	.769	.667	1.000	.727	.000	
wine	.827	.919	.930	.829	.922	.925	.848	.921	.925	.841	.919	.924	.847	.933	.945	.865	.926	.938	.886	.928	.947	.895	.933	.953
wineR	.650	.665	.659	.643	.661	.658	.719	.714	.687	.699	.696	.697	.799	.729	.737	.735	.717	.725	.667	.875		.777	.788	.681
wineW	.634	.682	.607	.622	.668	.605	.674	.711	.650	.657	.689	.631	.702	.724	.668	.683	.712	.658	.792	.707		.770	.748	.712
vowel	.760	.859	.820	.726	.862	.819	.820	.888	.877	.774	.883	.858	.851	.897	.913	.806	.891	.886	.955	.934	.896	.862	.928	.926
yeast	.684	.685	.672	.648	.664	.657	.727	.717	.705	.703	.710	.682	.750	.748	.766	.750	.737	.684			1.000	.855	.857	.769
Mean	**.779**	**.803**	**.797**	**.772**	**.801**	**.792**	**.820**	**.827**	**.818**	**.806**	**.818**	**.812**	**.849**	**.840**	**.845**	**.845**	**.839**	**.835**	**.925**	**.926**	**.963**	**.909**	**.825**	**.912**
Mean Rank	**4.11**	**2.21**	**3.32**	**5.05**	**2.68**	**3.63**	**3.42**	**2.11**	**3.53**	**4.63**	**3.11**	**4.21**	**3.63**	**2.63**	**3.21**	**4.08**	**3.32**	**4.13**	**2.87**	**3.24**	**3.37**	**4.00**	**3.45**	**4.08**

Table 8: Efficiency - multiclass data sets

AvgC	$\epsilon = 0.2$						$\epsilon = 0.1$						$\epsilon = 0.05$						$\epsilon = 0.01$					
	Hinge			Margin			Hinge			Margin			Hinge			Margin			Hinge			Margin		
	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X
balance	1.12	1.00	1.00	1.10	1.00	1.00	1.75	1.50	1.52	1.72	1.45	1.48	2.23	1.97	1.97	2.20	1.91	1.94	2.84	2.77	2.77	2.84	2.72	2.73
cars	0.82	0.82	0.81	0.82	0.82	0.82	0.95	0.93	0.93	0.95	0.93	0.93	1.07	0.99	1.01	1.12	1.00	1.02	1.45	1.45	1.67	1.65	1.60	1.76
cmc	1.83	1.88	1.87	1.86	1.91	1.91	2.28	2.30	2.32	2.29	2.32	2.34	2.55	2.56	2.60	2.56	2.57	2.60	2.81	2.84	2.91	2.82	2.83	2.89
cool	0.83	0.82	0.83	0.83	0.82	0.83	0.94	0.93	0.95	0.94	0.93	0.95	1.05	1.03	1.07	1.06	1.04	1.07	1.55	1.46	1.68	1.55	1.49	1.68
glass	2.29	2.08	2.24	2.75	2.43	2.62	3.44	3.50	3.66	4.08	3.83	4.09	4.29	4.29	4.52	4.68	4.65	4.79	5.39	5.31	5.43	5.56	5.49	5.56
heat	0.81	0.80	0.80	0.80	0.80	0.81	0.91	0.91	0.91	0.91	0.91	0.91	0.97	0.96	0.97	0.96	0.96	0.97	1.27	1.20	1.40	1.33	1.22	1.43
image	0.82	0.81	0.81	0.82	0.81	0.81	0.94	0.92	0.92	0.93	0.92	0.92	1.05	0.98	0.99	1.09	0.98	0.99	2.41	1.78	2.22	2.78	1.95	2.48
iris	1.39	0.91	0.90	1.40	0.94	0.92	1.75	1.14	1.13	1.81	1.16	1.15	2.09	1.46	1.44	2.14	1.47	1.45	2.52	2.18	2.17	2.59	2.27	2.29
steel	1.45	1.25	1.35	1.86	1.65	1.83	2.47	2.23	2.92	3.37	3.19	3.55	3.69	3.45	4.18	4.41	4.14	4.64	5.44	5.35	5.89	5.67	5.58	5.74
tae	2.41	2.38	2.36	2.41	2.38	2.38	2.70	2.69	2.69	2.71	2.69	2.69	2.83	2.86	2.85	2.86	2.85	2.85	2.97	2.97	2.96	2.97	2.97	2.97
user	0.89	0.88	0.89	0.90	0.88	0.89	1.16	1.11	1.11	1.23	1.16	1.17	1.69	1.75	2.27	1.79	1.86	2.48	3.62	3.73	4.52	3.70	3.70	4.31
wave	1.09	0.98	0.98	1.09	0.98	0.98	1.38	1.28	1.39	1.54	1.40	1.51	1.69	1.61	1.92	2.01	1.85	2.06	2.55	2.45	2.70	2.74	2.66	2.78
vehicle	1.40	1.25	1.33	1.53	1.36	1.44	1.98	1.80	1.92	2.23	2.06	2.14	2.62	2.30	2.49	2.84	2.60	2.73	3.50	3.39	3.53	3.62	3.53	3.65
whole	1.55	1.52	1.62	1.56	1.54	1.63	2.16	2.16	2.27	2.20	2.16	2.26	2.56	2.56	2.63	2.58	2.57	2.63	2.92	2.91	2.92	2.91	2.92	2.92
wine	0.99	0.85	0.85	1.00	0.85	0.86	1.68	1.01	1.04	1.72	1.02	1.03	2.25	1.43	1.41	2.29	1.45	1.46	2.81	2.39	2.34	2.83	2.43	2.45
wineR	1.74	2.10	2.13	2.36	2.76	2.86	2.40	3.14	3.12	3.57	4.04	3.93	3.19	3.91	3.86	4.46	4.71	4.62	4.88	5.06	5.40	5.38	5.47	5.46
wineW	2.18	2.25	1.95	3.42	3.26	3.40	3.54	3.98	2.88	4.81	4.88	4.70	4.49	5.00	3.96	5.65	5.83	5.38	6.08	6.28	5.67	6.61	6.72	6.40
vowel	2.21	0.95	1.01	3.68	0.94	1.03	5.30	2.03	2.10	6.56	2.91	4.03	7.34	5.10	5.20	8.26	6.60	6.88	9.95	9.32	9.23	10.17	9.68	9.61
yeast	2.06	2.41	2.98	3.86	4.25	4.48	3.39	4.24	5.01	5.83	6.10	6.30	5.38	5.96	6.54	7.15	7.28	7.58	8.39	8.27	9.07	9.05	9.04	9.32
Mean	**1.47**	**1.37**	**1.41**	**1.79**	**1.60**	**1.66**	**2.17**	**1.99**	**2.04**	**2.60**	**2.32**	**2.43**	**2.79**	**2.64**	**2.73**	**3.17**	**2.96**	**3.06**	**3.86**	**3.74**	**3.92**	**4.04**	**3.91**	**4.02**
Mean Rank	**3.84**	**2.03**	**2.63**	**5.21**	**3.08**	**4.21**	**3.47**	**1.82**	**3.18**	**4.89**	**3.11**	**4.53**	**2.89**	**2.00**	**3.34**	**4.84**	**3.21**	**4.71**	**2.84**	**2.00**	**3.74**	**4.50**	**3.26**	**4.66**
OneC	Hinge			Margin			Hinge			Margin			Hinge			Margin			Hinge			Margin		
	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X	I	IX	X
balance	.891	.949	.952	.915	.948	.957	.512	.670	.637	.587	.749	.724	.275	.425	.407	.346	.503	.488	.026	.039	.040	.034	.084	.081
cars	.824	.817	.815	.824	.816	.815	.949	.929	.933	.951	.929	.932	.937	.983	.981	.939	.983	.982	.750	.773	.704	.752	.772	.719
cmc	.320	.382	.417	.432	.485	.496	.142	.195	.199	.244	.292	.273	.086	.116	.094	.145	.173	.156	.056	.042	.015	.070	.066	.032
cool	.831	.819	.830	.833	.825	.831	.940	.932	.945	.938	.932	.945	.952	.968	.947	.947	.967	.947	.626	.691	.583	.647	.691	.585
glass	.285	.508	.418	.457	.614	.505	.078	.132	.089	.136	.207	.143	.025	.021	.013	.058	.053	.032	.000	.000	.000	.000	.000	.000
heat	.805	.798	.804	.802	.798	.806	.914	.907	.910	.909	.905	.909	.963	.962	.968	.963	.962	.968	.825	.876	.768	.813	.872	.764
image	.818	.810	.812	.819	.807	.812	.935	.917	.922	.934	.917	.922	.966	.975	.980	.972	.976	.981	.595	.804	.657	.640	.816	.695
iris	.599	.837	.830	.600	.839	.840	.354	.833	.839	.397	.833	.843	.183	.689	.697	.259	.685	.703	.082	.249	.236	.097	.239	.232
steel	.674	.800	.772	.776	.850	.829	.369	.464	.445	.516	.562	.519	.233	.262	.214	.346	.380	.285	.094	.055	.058	.167	.184	.157
tae	.028	.175	.139	.038	.223	.174	.012	.042	.041	.021	.064	.046	.005	.011	.010	.008	.014	.008	.001	.001	.000	.000	.001	.000
user	.891	.883	.890	.896	.881	.894	.898	.925	.923	.899	.925	.924	.658	.668	.527	.712	.697	.527	.122	.091	.025	.187	.164	.075
wave	.914	.973	.961	.914	.974	.963	.632	.731	.699	.645	.729	.706	.401	.478	.377	.428	.501	.405	.027	.060	.021	.064	.091	.052
vehicle	.679	.807	.769	.746	.838	.811	.421	.554	.485	.495	.578	.530	.200	.346	.234	.292	.379	.305	.026	.052	.015	.060	.099	.040
whole	.512	.515	.418	.552	.507	.413	.078	.054	.028	.131	.084	.030	.020	.011	.005	.039	.013	.003	.003	.001	.000	.005	.000	.000
wine	.944	.852	.854	.928	.846	.856	.542	.927	.907	.534	.924	.906	.231	.685	.670	.249	.688	.669	.020	.124	.147	.032	.152	.169
wineR	.349	.450	.446	.468	.554	.541	.092	.166	.152	.241	.276	.262	.021	.054	.040	.127	.131	.129	.000	.002	.000	.024	.019	.005
wineW	.281	.431	.270	.464	.559	.438	.106	.167	.039	.239	.277	.168	.037	.062	.009	.125	.136	.059	.003	.006	.000	.021	.022	.001
vowel	.489	.935	.936	.669	.935	.944	.197	.720	.575	.359	.777	.646	.086	.308	.226	.181	.381	.330	.004	.030	.023	.028	.069	.055
yeast	.223	.321	.377	.455	.554	.556	.017	.085	.122	.209	.276	.213	.004	.015	.020	.078	.128	.073	.000	.000	.000	.004	.002	.001
Mean	**.598**	**.687**	**.669**	**.663**	**.729**	**.709**	**.431**	**.545**	**.521**	**.494**	**.591**	**.560**	**.331**	**.423**	**.390**	**.380**	**.460**	**.424**	**.172**	**.205**	**.173**	**.192**	**.228**	**.193**
Mean Rank	**4.53**	**3.71**	**4.11**	**3.21**	**3.03**	**2.42**	**4.84**	**3.53**	**4.00**	**3.63**	**2.37**	**2.63**	**5.16**	**3.21**	**4.39**	**3.34**	**1.79**	**3.11**	**4.34**	**3.05**	**4.87**	**2.92**	**1.89**	**3.92**