Regression Trees for Streaming Data with Local Performance Guarantees

Ulf Johansson, Cecilia Sönströd, Henrik Linusson
School of Business and IT
University of Borås, Sweden
Email: {ulf.johansson, cecilia.sonstrod, henrik.linusson}@hb.se
Henrik Boström
Dept. of Computer and Systems Sciences
Stockholm University, Sweden
Email: henrik.bostrom@dsv.su.se

***Abstract*—Online predictive modeling of streaming data is a key task for big data analytics. In this paper, a novel approach for efficient online learning of regression trees is proposed, which continuously updates, rather than retrains, the tree as more labeled data become available. A conformal predictor outputs prediction sets instead of point predictions; which for regression translates into prediction intervals. The key property of a conformal predictor is that it is always valid, i.e., the error rate, on novel data, is bounded by a preset significance level. Here, we suggest applying Mondrian conformal prediction on top of the resulting models, in order to obtain regression trees where not only the tree, but also each and every rule, corresponding to a path from the root node to a leaf, is valid. Using Mondrian conformal prediction, it becomes possible to analyze and explore the different rules separately, knowing that their accuracy, in the long run, will not be below the preset significance level. An empirical investigation, using 17 publicly available data sets, confirms that the resulting rules are independently valid, but also shows that the prediction intervals are smaller, on average, than when only the global model is required to be valid. All-in-all, the suggested method provides a data miner or a decision maker with highly informative predictive models of streaming data.**

Keywords*—Conformal prediction, Streaming data, Regression trees, Interpretable models

I. Introduction

In a data stream, the instances represent an ordered sequence and typically arrive so rapidly that it is not possible to store them for future inspection and analysis. With this in mind, the modeling and analysis of streaming data is said to be performed *online*. Often, the task is to predict target values for new instances in the data stream, based on previous labeled instances in the stream. Such online predictive modeling of streaming data is an important tool for efficient mining of big data.

This paper focuses on the very common situation where an initial predictive model is built offline from existing labeled data, but the predictions must be performed online as new instances arrive, i.e., it is only the test set that is actually streaming. In this scenario, since retraining models is a very expensive operation, especially for techniques like Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs), it is often crucial to be able to continuously update a predictive model as more labeled instances become available.

In many real-world scenarios, predictive models need to be interpretable, i.e., it should be possible for humans to, at the very least, understand the logic behind individual predictions. One example is when it must be possible to verify the predictions for legal or safety reasons [1]. But it has also been argued that interpretable models increase user acceptance, see e.g. [2] and [3]. To obtain interpretable models, most data miners will use either tree models or rule sets. Naturally, a tree model can be trivially converted into a rule set; each path from the root node to a leaf becomes a separate rule. Consequently, a data miner or decision maker can inspect the entire model to discover and analyze the overall relationships present in the data, but also use individual rules to explain the reasoning behind specific predictions. For online modeling, it should be noted that when interpretable models are required, we would typically also like the models to be at least fairly stable. It would be very hard to inspect and analyze constantly shifting models, even if they are all transparent. A stable, but still accurate, model for online prediction is, on the other hand, a very good tool for both explaining predictions and discovering relationships in the streaming data.

In predictive modeling, it is often vital to be able to estimate how accurate a model will be before applying it to test data. This becomes even more important when an interpretable model is used not only for prediction but also as a tool for exploration and explanation. Simply put, we would like to be able to quantify the confidence we have in the model, or ultimately, in individual predictions. In this study, we will use the conformal prediction (CP) framework [4] for this purpose. CP produces prediction sets instead of point predictions. For regression, which is the focus of this study, a prediction set is simply a prediction interval. All conformal predictors are *valid*, i.e., given a significance level $\epsilon \in (0, 1)$, the prediction regions will contain the true target value with a probability of at least ϵ. Until now, CP has mainly been applied on a global (model) level. In the studied scenario, however, it would be valuable to obtain performance guarantees for each and every

This work was supported by the Swedish Foundation for Strategic Research through the project High-Performance Data Mining for Drug Effect Detection (IIS11-0053), the Swedish Retail and Wholesale Development Council through the project Innovative Business Intelligence Tools (2013:5) and the Knowledge Foundation through the project Big Data Analytics by Online Ensemble Learning (20120192).

rule, not just for the overall model. In that case, a decision maker would have access to a number of rules for inspection and analysis, where each rule has a guaranteed error bound. In this paper, we suggest and evaluate an approach, based on CP in the online scenario, making this possible.

The main contributions of the paper are:

1) A straightforward but very efficient way of updating, instead of retraining, regression tree models as more labeled data become available.
2) A novel way of using the conformal prediction framework, resulting in guaranteed error bounds for individual rules in regression trees.
3) An extensive empirical investigation, comparing online updating of regression trees, with and without guarantees on each individual leaf (rule), to retraining trees from scratch.

In the next section, we provide a brief introduction to the conformal prediction framework, introducing the basic concepts and definitions. In Section III, the proposed approaches to online updating of regression trees with local and global performance guarantees are presented, together with the setup of the empirical investigation. Following that, the actual results are presented in Section IV. The relation to previous work is discussed in Section V, and finally, in Section VI, we summarize the main findings and point out directions for future research.

II. Conformal prediction

The CP framework [4] is a tool for associating the predictions of a classification or regression model with a measure of their confidence. Whereas typical predictive models output point predictions, a conformal predictor outputs a *prediction region*, i.e., a set of class labels or a real-valued interval, for each test pattern. Given a predefined significance level ϵ, each such prediction region contains the true target of the corresponding test pattern with probability $1-\epsilon$.

To produce such confidence predictions, a conformal predictor relies on a *nonconformity function* — a function $A(\boldsymbol{x}, y, \mathbb{Z}) \rightarrow \mathbb{R}$ that measures the strangeness (the nonconformity) of an example $(\boldsymbol{x}, y)$ compared to a multiset of examples Z. The nonconformity function A is applied to a set of training examples $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_k, y_k)$, to obtain a set of nonconformity scores $\alpha_1, ..., \alpha_k$, as well as to a tentatively labeled test example $(\boldsymbol{x}_{k+1}, \tilde{y})$ to obtain a nonconformity score $\alpha_{k+1}^{\tilde{y}}$. Using a form of hypothesis testing, $\alpha_{k+1}^{\tilde{y}}$ is compared to $\alpha_1, ..., \alpha_k$ to discern whether or not $\tilde{y}$ is likely to be the correct label for $\boldsymbol{x}$, and thus whether or not it should be included in the prediction region. This process is repeated for each possible output $\tilde{y} \in \mathbb{Y}$, and the final prediction region contains the true target y_{k+1} with probability $1-\epsilon$.

Although A can be any real-valued function [4], it is typically based on the error (or some other property) of a traditional machine learning model, called the *underlying model*, h, of the conformal predictor. Nonconformity is thus measured as $A(\boldsymbol{x}, y, h)$, where h represents a generalization of $\boldsymbol{Z}$ (the training data). For a regression problem, nonconformity can, for instance, be defined as the absolute error [5] or the signed error [6] of a regression model. The motivation behind this kind of nonconformity function is that the stranger an example is (i.e., the more atypical it is with regard to the training data of h), the more likely it is that h will make a larger error in its prediction of the example's output.

There are two major categories of conformal predictors: *transductive* conformal predictors (TCP) [4] and *inductive* conformal predictors (ICP) [7]. In TCP, the nonconformity of an example $(\boldsymbol{x}_i, y_i)$ is measured in relation to the multiset $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_{k+1}, \tilde{y})$, and predictions are made according to the following scheme:

1) Train h on $Z = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_{k+1}, \tilde{y})$
2) Measure the nonconformity α_i of $(\boldsymbol{x}_i, y_i) \in \boldsymbol{Z}$ as $A(\boldsymbol{x}_i, y_i, h)$
3) Calculate a p-value for $(\boldsymbol{x}_{k+1}, \tilde{y})$ as

$$p_{k+1}^{\tilde{y}} = \frac{\#\left\{\boldsymbol{z}_i \in \boldsymbol{Z} \mid \alpha_i \geq \alpha_{k+1}^{\tilde{y}}\right\}}{\#\boldsymbol{Z}} \tag{1}$$

4) If $p_{k+1}^{\tilde{y}} \leq \epsilon$, reject $\tilde{y}$ from the prediction region of $\boldsymbol{x}_{k+1}$, otherwise include it
5) Repeat steps 1-4 for each $\tilde{y} \in \boldsymbol{Y}$

Hence, in TCP, the underlying model h needs to be retrained once for every new test pattern and every possible output value, rendering it computationally expensive. In ICP, only part of the training data is used to train h, while the remainder of the training data is used to *calibrate* the conformal predictor. This effectively leads to an inductive confidence predictor that only needs to be trained once:

1) Divide the training set $\boldsymbol{Z}$ into two subsets $\boldsymbol{Z}^t$ (the proper training set) and $\boldsymbol{Z}^c$ (the calibration set)
2) Train h on $\boldsymbol{Z}^t$
3) Measure the nonconformity α_i of $(\boldsymbol{x}_i, y_i) \in \boldsymbol{Z}^c$ as $A(\boldsymbol{x_i}, y_i, h)$

When making confidence predictions with an ICP, the underlying model does not need to be retrained; instead $\alpha_{k+1}^{\tilde{y}}$ is computed from the already-trained model h, and $p_{k+1}^{\tilde{y}}$ is calculated using the precomputed nonconformity scores $A(\boldsymbol{x}_i, y_i, h), \boldsymbol{x}_i \in \boldsymbol{Z}^c$:

1) Measure the nonconformity of $(\boldsymbol{x}_{k+1}, \tilde{y})$ as $A(\boldsymbol{x}_{k+1}, \tilde{y}, h)$
2) Calculate the p-value for $(\boldsymbol{x}_{k+1}, \tilde{y})$ as

$$p_{k+1}^{\tilde{y}} = \frac{\#\left\{\boldsymbol{z}_i \in \boldsymbol{Z}^c \mid \alpha_i \geq \alpha_{k+1}^{\tilde{y}}\right\} + 1}{\#\boldsymbol{Z}^c + 1} \tag{2}$$

3) If $p_{k+1}^{\tilde{y}} \leq \epsilon$, reject $\tilde{y}$ from the prediction region of $\boldsymbol{x}_{k+1}$, otherwise include it
4) Repeat steps 1-3 for each $\tilde{y} \in \boldsymbol{Y}$

In regression, since not all $\tilde{y} \in \boldsymbol{Y}$ can be explicitly tested, $\alpha_{k+1}^{\tilde{y}}$ is implicitly measured based on the minimal non-rejectable p-value; that is, if nonconformity is defined as the absolute error of h,

$$A(\boldsymbol{x}_i, y_i, h) = |y_i - h(\boldsymbol{x}_i)|, \tag{3}$$

and predictions are to be made with confidence ϵ, then exactly a fraction ϵ of the calibration examples should have an absolute

error nonconformity score larger than $\alpha_{k+1}^{\tilde{y}}$. Hence, $\alpha_{k+1}^{\tilde{y}}$ is assumed to be equal to α_ϵ, where α_ϵ is the $(1-\epsilon)$-percentile absolute error of h on the calibration set. The prediction for $\boldsymbol{x}_{k+1}$ is then formulated as

$$\hat{Y}_{k+1}^{\epsilon} = h(\boldsymbol{x}_{k+1}) \pm \alpha_\epsilon \,. \tag{4}$$

It should be noted that for inductive conformal regression the calibration set must be sufficiently sized for the chosen significance level. More specifically, since one particular calibration instance is used to determine the interval width, there must be enough calibration instances to allow a partition into an appropriate number of subsets. Thus, $\epsilon = 0.9$ requires ten calibration instances, $\epsilon = 0.95$ requires 20 calibration instances and $\epsilon = 0.99$ requires 100 calibration instances.

Although the confidence measures provided by a conformal predictor are valid in the long run, i.e., given a set of n confidence predictions, approximately $n(1-\epsilon)$ will be correct, a basic conformal predictor does not guarantee that its errors are evenly distributed. Some patterns exhibiting a particular characteristic might be more or less difficult to predict, thus making it more or less likely for the conformal predictor to make an erroneous prediction. Mondrian conformal predictors (MCP) [4] are able to provide a stronger guarantee of validity than such basic conformal predictors. By conditioning the confidence predictions on some characteristic of a test pattern, an MCP can provide confidence measures in a category-wise manner. If, for instance, an MCP classifier is conditioned on a per-class basis, its validity will apply not only for the full test set, but also for each of the classes separately; that is, a class-conditioned MCP will not make a disproportionately large (or small) amount of errors on examples belonging to a particular class. Similarly, an MCP can be conditioned on some input characteristic(s), so that errors are distributed proportionately amongst some set of subcategories within the input space.

To convert a TCP model to a MTCP model, (1) simply needs to be redefined as

$$p_{k+1}^{\tilde{y},\kappa} = \frac{\#\left\{\boldsymbol{z}_i \in \boldsymbol{Z}_\kappa \mid \alpha_i \geq \alpha_{k+1}^{\tilde{y}}\right\}}{\#\boldsymbol{Z}_\kappa}, \tag{5}$$

where $\boldsymbol{Z}_\kappa \subseteq \boldsymbol{Z}$, given by some condition(s) κ. The conversion of an ICP into an MICP is analogous in (2).

III. Method

In this section, the suggested algorithms are first presented, with emphasis on explaining how the different guarantees are provided by the conformal prediction framework. After that, the experimentation is described in detail.

A. Online updating of conformal regression trees

Two novel interpretable online regression models, based on CART [8] and ICP are evaluated in this paper. The first, *Global Online Inductive Conformal Predictor* (G-OICP), is an online-updated regression tree that leverages the conformal prediction framework to provide confidence predictions. In G-OICP, a standard regression tree ICP is first constructed, i.e., only part of the initially available training data is used to train a CART model, while the remainder of the training data is used to calibrate the ICP. As G-OICP operates in the online setting, the true labels of already predicted test examples are revealed and incorporated into the model. However, rather than retraining the entire regression tree, only the leaves of the tree are updated. That is, once the true target of a test instance $\boldsymbol{x}_{k+n}$ is revealed, the leaf, l_{k+n}, of the decision tree that predicted the output of $\boldsymbol{x}_{k+n}$ is updated with y_{k+n}. After updating l_{k+n}, the nonconformity scores of the calibration set are recomputed before predicting the output of the next test pattern.

The second model, *Local Online Inductive Conformal Predictor* (L-OICP), is also incremented online after each $\boldsymbol{x}_{k+n}$ by updating l_{k+n} with y_{k+n}. However, L-OICP is defined as a rule-conditioned Mondrian conformal predictor. This means that each leaf l in the decision tree represents a separate Mondrian category κ_l, given by the chain of conditions on the path from the root of the tree to l. As such, L-OICP is able to guarantee the validity of each separate leaf (derived rule) in the tree. Both OICP variants use the absolute error nonconformity function (3).

B. Experimental setup

All experiments were performed in MatLab, in particular using the Statistics toolbox. The regression trees used are built using the MatLab version of CART, called *rtree*. All parameter values, with the exception of MinLeaf, were left at the default vales. MinLeaf is the minimum number of leaf node observations from the training set, ensuring that in a trained regression tree, each leaf covers at least MinLeaf instances. Thus, MinLeaf will have a direct impact on the overall sizes of the resulting trees. In addition, it should be noted that the parameter MinLeaf will affect L-OICP and G-OICP differently. Since L-OICP uses a separate calibration set for each leaf, it is important that these sets are large enough to provide well-calibrated non-conformity scores. When the calibration set is not sufficiently large to support the significance level, which in this study may occur for L-OICP using the lower MinLeaf values and a very high significance level, this was handled by setting that particular prediction interval to the entire range. G-OICP, on the other hand, uses a global set of non-conformity scores, so there is no need to reduce the number of leaves just to obtain accurate calibrations. Still, it is important to keep in mind that smaller models are inherently more comprehensible, which is an important consideration in most projects where interpretable models were chosen in the first place. In the experimentation, reasonable but not optimized values for MinLeaf were used for the different approaches.

For the actual evaluation, each data set was randomly divided into three parts; the training set (30%), the initial calibration set (20%) and the streaming test set (50%). When multiple runs were used, the partition was performed separately for each run, and the tabulated results were averaged over all runs. Before the experimentation, all target values were normalized to $[0, 1]$, in order to obtain more readable comparisons across data sets. With this scaling, the size of a prediction interval, of course, expresses the fraction of the target range covered by the interval.

In the experiments, 17 publicly available medium-sized data sets are used, ranging from approximately 4000 to 9500

instances. All data sets are from the UCI [9], Delve [10] or KEEL [11] repositories. The data sets are described in Table I below, where *#inst.* is the number of instances and *#attrib*. is the number of input attributes.

TABLE I. DATA SETS

Name	#inst.	#attrib.	Origin
abalone	4177	8	UCI
bank8fh	8192	8	Delve
bank8fm	8192	8	Delve
bank8nh	8192	8	Delve
bank8nm	8192	8	Delve
comp	8192	12	Delve
deltaA	7129	5	KEEL
deltaE	9517	6	KEEL
kin8fh	8192	8	Delve
kin8fm	8192	8	Delve
kin8nh	8192	8	Delve
kin8nm	8192	8	Delve
puma8fh	8192	8	Delve
puma8fm	8192	8	Delve
puma8nh	8192	8	Delve
puma8nm	8192	8	Delve
wineWhite	3961	11	UCI

The three different experiments are described below:

Experiment 1: The purpose of this experiment is to further explain the suggested approach and to demonstrate how the different setups operate. Here, we use the Puma8fm data set, and 2457 of the 8192 instances were used to build the regression tree. Another 1638 were used for the initial calibration and the final 4097 were used as a streaming test set. As mentioned above, a key parameter is MinLeaf, which determines the number of training instances that must fall into each leaf. Naturally, the larger MinLeaf is, the smaller the resulting tree. For this demonstration, MinLeaf was set to 299, i.e., all leaves must include at least 299 training instances. The reason for this very high value is that it will result in quite compact trees that can be easily inspected and analyzed. The downside is, of course, that the predictive performance may suffer.

Experiment 2: Here, we evaluate the suggested L-OICP and G-OICP approaches on a number of data sets and using different significance levels. In this experiment, we first show that both approaches produce empirically valid models. In particular, validity must apply for individual rules in conformal regressors generated using L-OICP. The two approaches are contrasted by looking at how the errors are distributed over the different leaves (rules) in the trees. For this analysis, we use the standard deviation of the individual error rates over all leaves. In addition, the informativeness of the different approaches is evaluated by comparing (average) interval sizes. Finally, the effect of different MinLeaf parameter values is investigated.

Experiment 3: In this final experiment, the suggested approaches are compared to a batch-incremental scheme where the predictive model is frequently retrained. This setup starts with the same induced tree as the other two, but after every five test instances, a new model is built using all available data. Of the five new training instances, three are added to the training set, and the remaining two to the calibration set. Here, the predictive performance and the informativeness of the resulting models are compared. The overall purpose of this experiment is to investigate how much, if any, accuracy that is lost by updating and recalibrating instead of retraining the model.

IV. RESULTS

First, we take a very detailed look at the results from Experiment 1, where L-OICP and G-OICP were applied to just one fold of the Puma8fm data set. Fig. 1 below shows the induced regression tree for this specific fold. In this tree, there are only five splits altogether, and only two (out of eight possible) input variables are actually used. There are six leaves, with output values ranging from 0.146 to 0.883. As described in the method section, all output values were normalized to the interval $[0.0 - 1.0]$, so the possible outputs cover most of the target range.

```
x2<-0.147
|   x3<0.153
|   |   x2<-0.561:
|   |   |   y=0.146
|   |   x2>=-0.561
|   |   |   y=0.293
|   x3>=0.153
|   |   y=0.475
x2>=-0.147
|   x3<-0.237:
|   |   y=0.541
|   x3>=-0.237
|   |   x2<0.299
|   |   |   y=0.711
|   |   x2>=0.299
|   |   |   y=0.883
```

Fig. 1. Regression tree for Puma8fm

Interestingly enough, even though the tree is very small, it is still fairly accurate. In fact, if this regression tree is applied to the test set (in batch mode), the root-mean-square-error (RMSE) is as low as 0.1248 and the correlation coefficient is as high as r=0.89. In Fig. 2 below, we see the error distribution which shows that a large majority of all the errors are actually rather small.

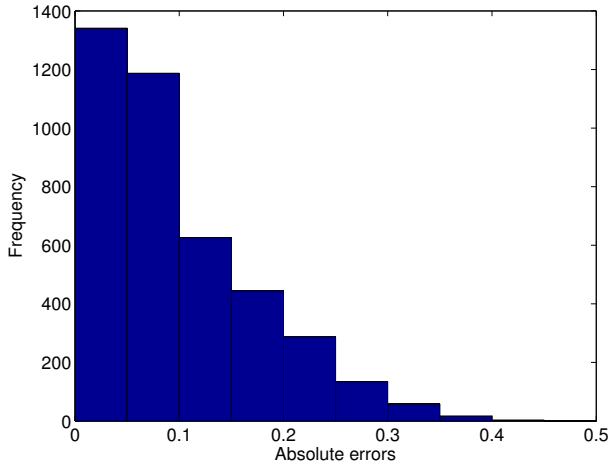

Fig. 2. Error distribution

When generating a conformal tree regressor using separate calibrations sets for each leaf, the error rate is bounded not on the model level, but for each leaf. This will result in intervals of different sizes, i.e., more narrow intervals for easier parts of the feature space and wider intervals for harder parts. Obviously, the higher the significance level, the larger the intervals. Fig. 3 below shows the final L-OICP regressor for $\epsilon = 0.90$.

```
x2<-0.147
|   x3<0.153
|   |   x2<-0.561:
|   |   |   y=[0.037, 0.263]
|   |   x2>=-0.561
|   |   |   y=[0.137, 0.451]
|   x3>=0.153
|   |   y=[0.204, 0.739]
x2>=-0.147
|   x3<-0.237:
|   |   y=[0.298, 0.778]
|   x3>=-0.237
|   |   x2<0.299
|   |   |   y=[0.496, 0.929]
|   |   x2>=0.299
|   |   |   y=[0.769, 1.000]
```

Fig. 3. L-OICP for Puma8fm. $\epsilon = 0.9$

Comparing the corresponding G-OICP regressor in Fig. 4 below to the L-OICP regressor in Fig. 3 above, we immediately see that for the G-OICP, all intervals have the same width.

```
x2<-0.147
|   x3<0.153
|   |   x2<-0.561:
|   |   |   y=[-0.062, 0.362]
|   |   x2>=-0.561
|   |   |   y=[0.081, 0.506]
|   x3>=0.153
|   |   y=[0.259, 0.684]
x2>=-0.147
|   x3<-0.237:
|   |   y=[0.326, 0.750]
|   x3>=-0.237
|   |   x2<0.299
|   |   |   y=[0.500, 0.925]
|   |   x2>=0.299
|   |   |   y=[0.676, 1.100]
```

Fig. 4. G-OICP for Puma8fm. $\epsilon = 0.9$

The reason is that for G-OICP, the error bound is on the model level, i.e., for some leaves the error rate will be above the significance level, and for some leaves it will be below. Looking at Fig. 5 below, which shows the cumulative error rates over the streaming test set for the different leaves in the G-OICP model, this is confirmed. For one leaf, the cumulative error rate is over 0.2, but for others it is below 0.05. So, with the G-OICP setup, all guarantees are on the model level, i.e., we know that the probability for a test set error is always $1-\epsilon$, but we can not say anything about individual leaves (rules) in the tree.

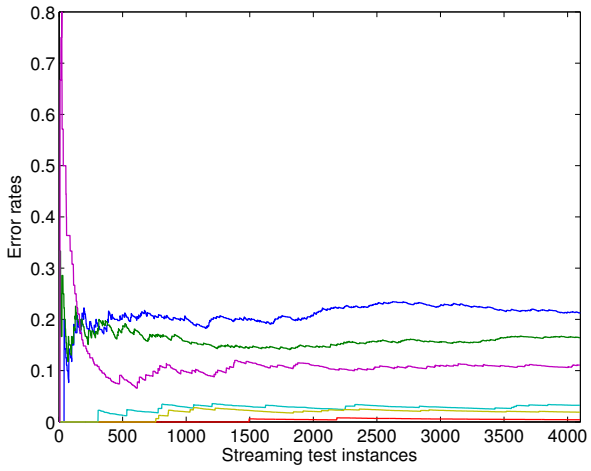

Fig. 5. G-OICP: error rates for the different leaves. $\epsilon = 0.9$

For L-OICP, however, the error rate for each leaf is, in the long run, i.e., disregarding statistical fluctuations, bounded by $1-\epsilon$. This is clearly demonstrated in Fig. 6 below, where we see that the individual error rates for all the leaves are quite similar, and indeed appear to converge towards the significance level $1-\epsilon = 0.1$. This is an important result, since it means that each leaf (rule) now can be inspected and analysed in isolation. Using L-OICP, we know that the probability for a test set error is $1-\epsilon$, for each leaf.

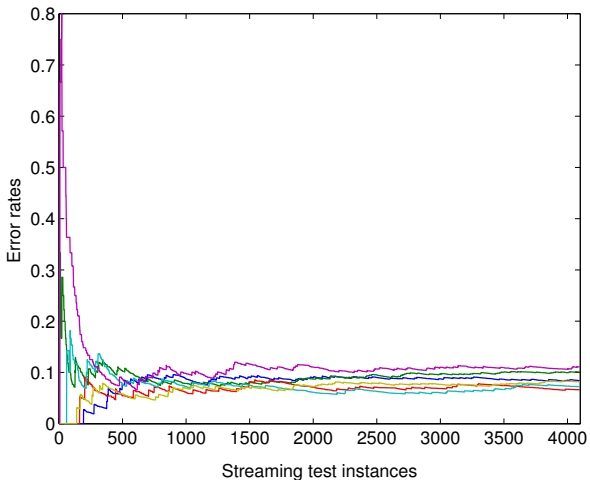

Fig. 6. L-OICP: error rates for the different leaves. $\epsilon = 0.9$

Retraining models during online learning is of course often too expensive. When using a (local) inductive conformal prediction on top of a regression tree in an online setting, we do not rebuild the model but update the predictions and recalibrate when correct test targets become available. Using this very efficient method, we expect to lose some predictive performance compared to retraining the model. This is further analyzed in Experiment 2, below. If there is no concept drift, which is the case in this study since we use artificial streaming data based on folding schemes, the models should improve, i.e., the intervals should become smaller, over time, as we see more instances. Fig. 7 below shows the interval sizes, as a function of the number of streaming test instances processed. Here, the overall picture is that most intervals become slightly smaller, even if they appear to be rather well-calibrated to start with.

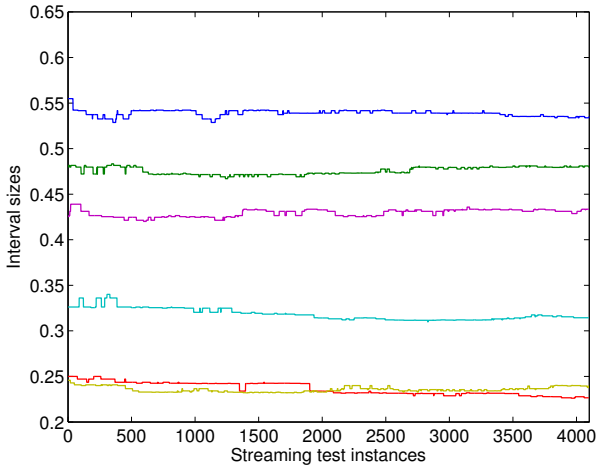

Fig. 7. L-OICP: interval sizes for the different leaves. $\epsilon = 0.9$

Table II below shows detailed results for every leaf and the different significance levels. In the table, the leaves from Fig. 3 and Fig. 4 are numbered in a breadth-first fashion. We see that for G-OICP, all leaves have the same interval size, but for L-OICP there are both narrow and wide intervals. Overall, the error rates for L-OICP are quite close to the different significance levels, for each and every leaf. Finally, we see that most intervals become slightly smaller as the model encounters more streaming test instances, as seen by the differences in interval sizes at Start and End.

TABLE II. RESULTS FOR INDIVIDUAL LEAVES

	$\epsilon = 0.8$			$\epsilon = 0.9$			$\epsilon = 0.95$			$\epsilon = 0.99$		
		Int. size			Int. size			Int. size			Int. size	
G-OICP	Err.	Start	End	Err.	Start	End	Err.	Start	End	Err.	Start	End
L1	.401	.322	.317	.214	.425	.425	.117	.509	.509	.025	.658	.653
L2	.347	.322	.317	.164	.425	.425	.080	.509	.509	.006	.658	.653
L3	.028	.322	.317	.004	.425	.425	.002	.509	.509	.000	.658	.653
L4	.084	.322	.317	.032	.425	.425	.019	.509	.509	.002	.658	.653
L5	.267	.322	.317	.111	.425	.425	.045	.509	.509	.012	.658	.653
L6	.046	.322	.317	.019	.425	.425	.006	.509	.509	.002	.658	.653
		Int. size			Int. size			Int. size			Int. size	
L-OICP	Err.	Start	End	Err.	Start	End	Err.	Start	End	Err.	Start	End
L1	.203	.433	.436	.085	.555	.535	.047	.617	.614	.003	.793	.734
L2	.244	.382	.401	.101	.480	.480	.046	.544	.541	.007	.646	.634
L3	.143	.184	.158	.066	.250	.227	.045	.280	.273	.000	.522	.404
L4	.183	.252	.244	.072	.326	.314	.032	.411	.382	.017	.526	.552
L5	.217	.342	.349	.111	.425	.434	.062	.484	.503	.010	.665	.666
L6	.174	.185	.176	.083	.246	.240	.045	.321	.319	.008	.492	.491

Table III below, finally, shows results for the models. Here, *Err.* is the overall error rate, *Dev.* is the standard deviation of the individual error rates over all leaves and *Size* is the average interval size over all streaming test instances. Since empirical error rates are very close to the significance levels, it is obvious that both setups produce valid conformal predictors. The main result is, however, the difference in *Dev.* between the two setups. Using L-OICP, we expect each leaf (rule) to be valid in itself, so it is reassuring to see that the error rate is indeed consistent over the different leaves. In addition, for this particular data set, setting and fold, L-OICP also produced tighter, i.e., more informative intervals.

TABLE III. MODEL RESULTS

	$\epsilon = 0.8$			$\epsilon = 0.9$			$\epsilon = 0.95$			$\epsilon = 0.99$		
	Err.	Dev.	Size	Err.	Dev.	Size	Err.	Dev.	Size	Err.	Dev.	Size
G-OICP	.209	.163	.320	.099	.086	.425	.049	.046	.509	.008	.009	.655
L-OICP	.198	.035	.294	.088	.017	.389	.046	.010	.453	.007	.006	.613

In summary, Experiment 1 has showed how regression trees and inductive conformal prediction can be used for predictive modeling of streaming data. The main purpose was to introduce, demonstrate and reason about the novel setup L-OICP. The key property of L-OICP is the ability to provide guarantees (error bounds) for parts of the model (here leaves/rules), making it much easier to inspect the regression tree and further analyse the underlying relationships. This should be compared to standard inductive conformal prediction (here called G-OICP) where the guarantee is only global, i.e., for the entire model.

Turning to Experiment 2, where we evaluate the suggested approaches on a number of data sets and using different significance levels, we start by fixing the significance level to $\epsilon = 0.9$ and investigate the effect of the MinLeaf parameter. Table IV below shows error rates and error deviations for

both L-OICP and G-OICP, using different MinLeaf values. First of all, it should be noted that both approaches are empirically valid for all MinLeaf parameter values. They are also generally well-calibrated, even if there is a tendency to be overly conservative for smaller MinLeaf values. Looking at the error deviations, we can confirm that L-OICP operates as expected, i.e., the error rate is very consistent over all leaves. For G-OICP, on the other hand, the deviations are high, indicating that while some leaves are more accurate than the significance level, other leaves are less accurate. As expected, the larger the parameter MinLeaf is, the smaller the deviations.

TABLE IV. ERROR RATES AND ERROR DEVIATIONS. $\epsilon = 0.9$

	Error rates						Error deviations					
	G-OICP			L-OICP			G-OICP			L-OICP		
MinLeaf	9	29	59	59	99	199	9	29	59	59	99	199
abalone	.092	.096	.097	.091	.095	.098	.119	.102	.096	.027	.021	.016
bank8fh	.088	.097	.098	.092	.096	.097	.092	.075	.073	.027	.020	.014
bank8fm	.097	.099	.100	.094	.096	.097	.118	.105	.115	.026	.020	.014
bank8nh	.087	.096	.098	.092	.095	.097	.099	.084	.083	.025	.020	.014
bank8nm	.097	.097	.098	.092	.096	.098	.165	.154	.151	.025	.020	.014
comp	.093	.097	.099	.094	.096	.097	.113	.096	.095	.032	.023	.023
deltaA	.092	.097	.099	.094	.099	.103	.125	.105	.094	.030	.026	.022
deltaE	.091	.098	.099	.095	.098	.100	.089	.061	.051	.032	.028	.024
kin8fh	.095	.098	.099	.094	.097	.099	.091	.064	.057	.027	.021	.014
kin8fm	.097	.098	.098	.094	.097	.099	.096	.068	.061	.026	.020	.014
kin8nh	.087	.095	.098	.092	.096	.098	.084	.059	.052	.026	.020	.013
kin8nm	.090	.096	.098	.093	.096	.098	.095	.073	.065	.026	.021	.014
puma8fh	.083	.094	.098	.092	.095	.097	.088	.072	.070	.026	.020	.014
puma8fm	.091	.099	.099	.094	.096	.098	.097	.080	.075	.026	.020	.014
puma8nh	.089	.098	.099	.093	.096	.097	.104	.091	.089	.027	.020	.015
puma8nm	.096	.098	.099	.093	.095	.097	.114	.106	.109	.026	.021	.015
wineWhite	.087	.097	.098	.098	.103	.105	.090	.072	.065	.045	.038	.033
Mean	**.091**	**.097**	**.099**	**.093**	**.096**	**.099**	**.105**	**.086**	**.082**	**.028**	**.022**	**.017**

In order to compare how informative and comprehensible the different models are, Table V below shows interval widths and model sizes for the two approaches, using different MinLeaf values. Starting with model sizes, the results are as expected; the smaller the MinLeaf value, the larger the trees. Still, it must be noted that there is a huge difference in comprehensibility between the different approaches. It would be very hard for a human to inspect and analyze regression trees with hundreds of nodes. Looking at the interval widths, remembering that this is the most important criterion when comparing approaches guaranteeing bounded error rates, we see that L-OICP often produces comparable, or even smaller, intervals than G-OICP. As a matter of fact, a ranking of the different approaches shows that L-OICP, using a MinLeaf value of 59, produced the most informative models. This is a very important result, since it shows that we do not need to sacrifice model performance in order to obtain the local guarantees provided by L-OICP. Actually, even the very compact trees produced by L-OICP using a MinLeaf value of 199 obtained prediction intervals only slightly larger than the competing setups. Finally, an outright comparison between L-OICP and G-OICP when using the same parameter value for MinLeaf (59), i.e., starting from the same induced tree, confirms that the models with the local guarantee also are clearly more informative.

TABLE V. INTERVAL WIDTHS AND MODEL SIZES. $\epsilon = 0.9$

	Interval widths						Model sizes					
	G-OICP			L-OICP			G-OICP			L-OICP		
MinLeaf	9	29	59	59	99	199	9	29	59	59	99	199
abalone	.276	.264	.267	.251	.254	.269	217	65	31	31	19	9
bank8fh	.343	.323	.329	.292	.297	.311	427	129	63	63	37	18
bank8fm	.173	.186	.212	.180	.200	.247	422	129	62	62	37	18
bank8nh	.401	.372	.371	.289	.286	.291	429	130	62	62	38	18
bank8nm	.220	.233	.244	.169	.174	.185	425	129	63	63	37	18
comp	.122	.123	.129	.118	.140	.206	417	128	62	62	37	18
deltaA	.133	.128	.129	.125	.127	.136	370	112	55	55	32	16
deltaE	.191	.183	.182	.183	.182	.191	495	149	72	72	42	22
kin8fh	.318	.322	.334	.337	.345	.361	424	127	62	62	36	17
kin8fm	.262	.284	.307	.310	.327	.352	423	127	61	61	36	17
kin8nh	.508	.489	.489	.497	.495	.497	427	129	63	63	37	17
kin8nm	.469	.471	.480	.484	.489	.497	426	129	63	63	37	18
puma8fh	.538	.494	.485	.461	.459	.477	427	128	61	61	36	17
puma8fm	.238	.237	.260	.247	.271	.324	426	127	62	62	36	17
puma8nh	.498	.468	.474	.446	.467	.489	427	129	63	63	37	20
puma8nm	.210	.245	.302	.279	.346	.401	425	129	63	63	37	20
wineWhite	.442	.417	.415	.407	.409	.417	205	62	30	30	17	8
Mean	**.314**	**.308**	**.318**	**.299**	**.310**	**.332**	**401**	**121**	**59**	**59**	**35**	**17**
Mean Rank	**3.83**	**3.00**	**3.78**	**2.17**	**3.11**	**5.11**						

Table VI below shows aggregated results over all data sets and at different significance levels for the two approaches using different values for the parameter MinLeaf. Again, we see that the results indicate that all setups are empirically valid, and rather well calibrated. Most importantly, as seen from the error deviations, L-OICP provides error bounds for each leaf/rule. Looking at how informative the approaches are, we see that for $\epsilon = 0.8$, $\epsilon = 0.9$, and $\epsilon = 0.95$, the resulting intervals cover approximately 20%, 30% and 40% respectively, of the range. When the significance level is very high ($\epsilon = 0.99$), the intervals become quite large, especially for L-OICP. This is partly due to an insufficient number of calibration instances, which was, as described in the Method section, handled by setting the corresponding interval sizes to a maximum value. An outright comparison of how informative the different approaches are, based on interval widths, shows that the local approach is as efficient as the global approach. In fact, for all significance levels except $\epsilon = 0.99$, the smallest intervals are obtained using L-OICP with MinLeaf=59.

TABLE VI. EXP2: AGGREGATED RESULTS OVER 17 DATA SETS

$\epsilon = 0.8$	G-OICP			L-OICP		
MinLeaf	9	29	59	59	99	199
Error rate	.188	.196	.198	.191	.196	.199
Error deviations	.155	.135	.130	.044	.037	.029
Interval widths	.231	.226	.234	.228	.238	.256
Mean rank	3.50	2.78	3.83	2.44	3.56	4.89
$\epsilon = 0.9$	G-OICP			L-OICP		
MinLeaf	9	29	59	59	99	199
Error rate	.091	.097	.099	.093	.096	.099
Error deviations	.105	.086	.082	.028	.022	.017
Interval widths	.314	.308	.318	.299	.310	.332
Mean rank	3.83	3.00	3.78	2.17	3.11	5.11
$\epsilon = 0.95$	G-OICP			L-OICP		
MinLeaf	9	29	59	59	99	199
Error rate	.044	.048	.049	.044	.046	.048
Error deviations	.068	.052	.048	.019	.015	.011
Interval widths	.394	.387	.400	.372	.382	.404
Mean rank	3.94	2.89	3.94	2.78	3.17	4.28
$\epsilon = 0.99$	G-OICP			L-OICP		
MinLeaf	9	29	59	59	99	199
Error rate	.008	.009	.010	.005	.007	.008
Error deviations	.026	.016	.014	.006	.006	.004
Interval widths	.568	.563	.582	.709	.584	.563
Mean rank	3.22	2.33	3.33	5.67	3.61	2.83

Summarizing Experiment 2, the main result is that the suggested L-OICP approach is able to provide guarantees (error bounds) for individual leaves (rules) without sacrificing global efficiency. The parameter MinLeaf is very important, but the effect of changes in the parameter value is obvious; lowering MinLeaf will result in larger trees. L-OICP benefits from using (much) higher MinLeaf values than what is normally used when inducing regression trees. The reason is the need for a sufficiently sized local calibration set. But, obtaining more compact trees is also an important goal in itself, since the overall purpose of choosing a modeling technique producing interpretable models is normally to provide the opportunity for human inspection and analysis. With this in mind, we think that the parameter values used here for MinLeaf, i.e., between 9 and 59 for G-OICP and between 59 and 199 for G-OICP, are reasonable choices for moderately sized data sets, and the particular scenario studied here, i.e., modeling streaming data using interpretable models. For truly large data sets, MinLeaf could be set (substantially) higher. In practice, MinLeaf would probably be optimized for individual data sets by using some kind of cross-validation scheme, or the trees would be built using pruning.

The aggregated results from Experiment 3 are presented in Table VII below. Here, the two approaches updating the regression tree are compared to an approach training a new regression tree after every five streaming test instances. Interestingly enough, as can be seen from the RMSE and correlation coefficient results, the very efficient method of updating the predictions but leaving the splits intact, does not reduce the accuracy, compared to retraining the entire tree. In fact, G-OICP with a MinLeaf setting of 9 is actually the most accurate setup on each and every data set. Looking at the error rates, we see that using retraining will lead to the most well-calibrated models. Despite this, using model retraining does not produce significantly more informative models i.e., the prediction intervals are generally not smaller than when using model updating. Finally, it must be noted that when using model retraining, there is of course not one but several models used for the actual prediction, making it very hard to use the predictive modeling to obtain insights about the underlying relationship.

TABLE VII. EXP3: AGGREGATED RESULTS OVER 17 DATA SETS

Model stats	G-OICP		G-OICP-retrain		L-OICP	
MinLeaf	9	29	9	29	59	99
RMSE	.087	.092	.096	.093	.097	.103
Mean rank	1.00	2.65	4.35	3.47	4.18	5.35
Correlation coefficient (r)	.811	.788	.770	.781	.768	.745
Mean rank	1.00	2.82	4.00	3.29	4.24	5.65
$\epsilon = 0.8$	G-OICP		G-OICP-retrain		L-OICP	
MinLeaf	9	29	9	29	59	99
Error rate	.188	.196	.199	.199	.191	.196
Interval widths	.231	.226	.232	.223	.228	.238
Mean rank	3.88	3.59	3.71	2.71	3.12	4.00
$\epsilon = 0.9$	G-OICP		G-OICP-retrain		L-OICP	
MinLeaf	9	29	9	29	59	99
Error rate	.091	.097	.010	.010	.093	.096
Interval widths	.314	.308	.316	.303	.299	.310
Mean rank	3.94	3.76	4.06	2.94	2.71	3.59
$\epsilon = 0.95$	G-OICP		G-OICP-retrain		L-OICP	
MinLeaf	9	29	9	29	59	99
Error rate	.044	.048	.049	.050	.044	.046
Interval widths	.394	.387	.396	.381	.372	.382
Mean rank	4.06	3.29	4.24	2.59	3.12	3.71
$\epsilon = 0.99$	G-OICP		G-OICP-retrain		L-OICP	
MinLeaf	9	29	9	29	59	99
Error rate	.008	.009	.010	.010	.005	.007
Interval widths	.568	.563	.573	.558	.709	.584
Mean rank	3.18	2.82	3.53	2.59	5.53	3.35

One major difference between the local and global calibration approaches is that the former produces trees where the leaves represent prediction intervals of different sizes while the latter leads to uniformly sized prediction intervals. The most important difference between the approaches, however, is that the validity guarantee applies to each and every leaf (rule) in the tree when performing the calibration locally (as done by L-OICP), while this is not true when performing the calibration globally (as done by G-OICP), which also was illustrated by the experiments.

In addition to the above theoretical properties, the main empirical findings are:

1) When using regression trees for streaming predictive modeling, the very efficient procedure of updating the predictions but leaving the tree structure (the splits) intact, did not reduce the accuracy or the informativeness, compared to frequently retraining the tree from scratch.
2) The prediction intervals from local calibration were generally smaller than the corresponding intervals associated with globally calibrated models.

V. RELATED WORK

In analysis and modeling of streaming data using regression trees, the two fundamental approaches are *batch* model building, which is essentially normal regression tree induction, using a set training data to build a static model, and *incremental* model building, where a dynamic model is maintained by incorporating new instances as they arrive in the data stream. Since the batch learning scheme requires that the entire set of training instances is available, this approach

is limited by memory and time constraints [12]. Incremental approaches, on the other hand, face the problem of dealing with continuous, and potentially computationally expensive, updates to the regression tree model. Recently, the hybrid approach of *batch-incremental model building*, where the model is updated using batches of new instances, typically using a sliding window, has become the most common approach [13].

Several algorithms exist for producing regression trees in an incremental fashion. Existing strategies typically address the issues of limiting computational cost and memory consumption required for producing and maintaining the model. This is achieved by considering streaming examples only once, while detecting and adapting to changes in the data stream.

The Hoeffding bound, as utilized in the VFDT algorithm [14], is a common component in tree induction for streaming classification and regression data [12], [14]–[16], which provides bounds on the probability of selecting an inappropriate split criterion in an internal node after observing only a subset of all examples. It has recently been pointed out that McDiarmid's inequality is better suited than the Hoeffding inequality for bounding the error of early splitting in the tree induction algorithm [17], but the consequences for streaming learning algorithms remain the same. It should be noted that in contrast to the approach proposed in this paper, the previous approaches may revise the structure of the generated tree as novel data is observed. In principle, this means that these methods may be more effective than the proposed method, if an appropriate structure cannot be identified using the initial batch of data that is utilized by the latter. On the other hand, the previous approaches do not provide any performance guarantees similar to the ones that follow from the adoption of the conformal prediction framework.

One specific property of data streams, which has attracted a lot of research, is detecting and handling *concept drift* [18]. For handling data stream changes, the CVFDT algorithm [15] introduced the notion of growing alternate subtrees at internal splits found to be inconsistent due to no longer passing the Hoeffding test. Once such an alternate subtree is found to be more accurate than the original subtree, the latter is replaced with the more recent one. The FIRT and FIMT algorithms [12] for incremental regression and model trees adopt the same strategy, while the SAIRT algorithm [19] utilizes a scheme that, unlike the Hoeffding test, does not require any *a priori* parameters. Option Hoeffding trees [20] and the ORTO algorithm [16] expand on the concept of growing subtrees, by letting multiple alternative subtrees exist simultaneously in the decision tree. These so-called *option nodes* reduce the number of examples needed to make a split in a node where the Hoeffding test is inconclusive (multiple split criteria appear appropriate), and increase the overall generalization performance of the full tree by averaging the prediction for an example over multiple subtrees. It should be noted that the conformal prediction framework requires that the data is *exchangeable*, i.e., if a concept drift is present, the conformal predictors will no longer be valid. The proposed approach is hence not suited for directly handling such situations. However, the framework can very well be used for detecting that a concept drift has occurred, e.g., if the predictions are found to be invalid, suggesting that retraining should take place.

When using tree models for regression, leaf node representation can either be very simple *point predictions*, i.e. single values, or consist of an entire (linear) regression model, with the resulting representation being called a *model tree*. Point prediction regression tree models have the intrinsic property of producing easily interpretable leaf nodes, but depend heavily on the exact formulation of the split criterion to obtain high accuracy. Predictive performance can be enhanced by using model trees, but these are, naturally, harder to interpret. A further option, aimed at providing improved predictive performance, but retaining interpretability [21], is to use prediction intervals in the leaf nodes of the regression tree.

To the best of our knowledge, no algorithm employing interval predictions use incoming test instances from the stream to improve prediction quality by dynamically adjusting the intervals. The idea of using new samples based on the same underlying distribution to improve prediction accuracy is, however, a component of the Adaptive Model Tree (AMT) algorithm by Zimmer et al. [22]. In AMT, the updates can be quite extensive, affecting not only the leaf node model, but all splits along the path to that leaf, thus sacrificing tree stability in order to utilize the extra information from new examples.

In addition, there are a number of studies on conformal prediction for regression, using, for instance, ridge regression [23], ANNs [24] and random forests [25]. Conformal prediction has also been successfully used in a number of applications where confidence in the predictions is of concern, including prediction of space weather parameters [26], estimation of software project effort [27], early diagnostics of ovarian and breast cancers [28] and diagnosis of acute abdominal pain [29].

VI. Concluding remarks

We have in this paper introduced and evaluated a novel method for online prediction where models need to be interpretable. A key component of the suggested approach is the continual refinement of predicted values from models. This updating, as performed here, is a very efficient operation, compared to model retraining, which is often too computationally costly for online scenarios. In the empirical investigation, it could be observed that the results obtained by updating indeed were comparable to those obtained for retraining. Another advantage is that the resulting models are fairly stable since only the predictions (or the prediction intervals), but not the structure, are modified. This is important when models are used for exploration or explanation, which is often the case when interpretable models are chosen.

In addition, we have suggested using Mondrian conformal prediction, where each leaf in the tree is considered to be a separate category, on top of regression trees. The result is that the prediction intervals of a regression tree will have different sizes, depending on how hard or easy prediction is in that part of the feature space. Most importantly, each and every rule will be independently valid, making it possible to analyse and explore rules separately. In practice, the Mondrian conformal predictor is accomplished by using different calibration sets for each leaf. The empirical results support the claim about valid rules, and also show that the conformal predictors are rather well-calibrated, especially for more compact trees, with relatively few leaves. Finally, the results also show that the prediction intervals produced using the local (Mondrian) setting are smaller, on average, than the corresponding intervals

of a regression tree using just one, global, calibration set. The added local guarantees are therefore not gained from a decrease in global informativeness.

Regarding future work, it should be noted that the suggested use of Mondrian conformal prediction, where the categories consist of parts of a predictive model and the purpose is to produce independently valid sub-models, is not restricted to regression trees and online learning. As a matter of fact, it could readily be applied to batch learning and, for instance, rule set learning; thus producing ordered or unordered rule sets where each rule is independently valid. Additional directions for future work include investigating extensions of the approach to allow for efficient restructuring of the trees and handling of concept drift.

REFERENCES

[1] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowl.-Based Syst.*, vol. 8, no. 6, pp. 373–389, 1995.

[2] M. W. Craven and J. W. Shavlik, "Extracting tree-structured representations of trained networks," in *Advances in Neural Information Processing Systems*. MIT Press, 1996, pp. 24–30.

[3] A. A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," in *In: A. Ghosh, and S. Tsutsui (Eds.) Advances in Evolutionary Computation*. Springer-Verlag, 2002.

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

[5] A. Gammerman and V. Vovk, "Hedging predictions in machine learning the second computer journal lecture," *The Computer Journal*, vol. 50, no. 2, pp. 151–163, 2007.

[6] H. Linusson, U. Johansson, and T. Löfström, "Signed-error conformal regression," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2014, pp. 224–236.

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

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

[9] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[10] C. E. Rasmussen, R. M. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani, "Delve data for evaluating learning in valid experiments," *www. cs. toronto. edu/delve*, 1996.

[11] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[12] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data Min. Knowl. Discov.*, vol. 23, no. 1, pp. 128–168, Jul. 2011.

[13] D. Alberg, M. Last, and A. Kandel, "Knowledge discovery in data streams with regression tree methods." *Wiley Interdisc. Rev.: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 69–78, 2012.

[14] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 71–80.

[15] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 97–106.

[16] E. Ikonomovska, J. Gama, B. Zenko, and S. Dzeroski, "Speeding-up hoeffding-based regression trees with options," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 537–544.

[17] L. Rutkowski, L. Pietruczuk, P. Duda, and M. Jaworski, "Decision trees for mining data streams based on the mcdiarmid's bound," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 6, pp. 1272–1279, 2013.

[18] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014.

[19] R. Fidalgo-Merino and M. Nunez, "Self-adaptive induction of regression trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1659–1672, 2011.

[20] B. Pfahringer, G. Holmes, and R. Kirkby, "New options for hoeffding trees," in *AI 2007: Advances in Artificial Intelligence*. Springer, 2007, pp. 90–99.

[21] D. Alberg, M. Last, R. Neuman, and A. Sharon, "Induction of mean output prediction trees from continuous temporal meteorological data," *2013 IEEE 13th International Conference on Data Mining Workshops*, vol. 0, pp. 208–213, 2009.

[22] A. M. Zimmer, M. Kurze, and T. Seidl, "Adaptive model tree for streaming data," *2013 IEEE 13th International Conference on Data Mining*, vol. 0, pp. 1319–1324, 2013.

[23] H. Papadopoulos, K. Proedrou, V. Vovk, and A. Gammerman, "Inductive confidence machines for regression," in *Machine Learning: ECML 2002*. Springer, 2002, pp. 345–356.

[24] H. Papadopoulos and H. Haralambous, "Neural networks regression inductive conformal predictor and its application to total electron content prediction," in *Artificial Neural Networks – ICANN 2010*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6352, pp. 32–41.

[25] U. Johansson, H. Boström, T. Löfström, and H. Linusson, "Regression conformal prediction with random forests," *Machine Learning*, vol. 97, no. 1-2, pp. 155–176, 2014.

[26] H. Papadopoulos and H. Haralambous, "Reliable prediction intervals with regression neural networks," *Neural Networks*, vol. 24, no. 8, pp. 842–851, 2011.

[27] H. Papadopoulos, E. Papatheocharous, and A. S. Andreou, "Reliable confidence intervals for software effort estimation." in *AIAI Workshops*. Citeseer, 2009, pp. 211–220.

[28] D. Devetyarov, I. Nouretdinov, B. Burford, S. Camuzeaux, A. Gentry-Maharaj, A. Tiss, C. Smith, Z. Luo, A. Chervonenkis, R. Hallett *et al.*, "Conformal predictors in early diagnostics of ovarian and breast cancers," *Progress in Artificial Intelligence*, vol. 1, no. 3, pp. 245–257, 2012.

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