Journal of Machine Learning Research X (2018) X-X Submitted X/X; Published X/X

Quantifying Uncertainty in Online Regression Forests

Theodore Vasiloudis TVAS@SICS.SE
RISE SICS
Stockholm, Sweden

Gianmarco De Francisci Morales GDFM@ACM.ORG
ISI Foundation
Torino, Italy

Henrik Boström BOSTROMH@KTH.SE
KTH Royal Institute of Technology
Stockholm, Sweden

Editor: Alexander Rakhlin

Abstract

Accurately quantifying uncertainty in predictions is essential for the deployment of machine learning algorithms in critical applications where mistakes are costly. Most approaches to quantifying prediction uncertainty have focused on settings where the data is static, or bounded. In this paper, we investigate methods that quantify the prediction uncertainty in a streaming setting, where the data is potentially unbounded.

We propose two meta-algorithms that produce prediction intervals for online regression forests of arbitrary tree models; one based on conformal prediction, and the other based on quantile regression. We show that the approaches are able to maintain specified error rates, with constant computational cost per example and bounded memory usage. We provide empirical evidence that the methods outperform the state-of-the-art in terms of maintaining error guarantees, while being an order of magnitude faster. We also investigate how the algorithms are able to recover from concept drift.

Keywords: Online learning, Uncertainty, Decision Trees, Regression

1. Introduction

Machine learning algorithms are increasingly being deployed in critical settings, such as the financial and medical domains. In such domains, the ability to quantify the uncertainty in the algorithms' predictions is crucial to guide decision making. In addition, in time-critical applications, such as autonomous driving, algorithms need to produce predictions and uncertainty estimates under tight computational and time budgets. Online learning methods are often used in such settings, alas, without providing uncertainty estimates, which makes them unsuitable for scenarios where errors are costly.

There exists a variety of applications where such critical decisions need to be made in real time, while using limited computational resources. In the financial sector, time-critical decisions need to be made based on noisy signals, and the models must be kept continuously up to date with the latest market data (Gama, 2017). In autonomous vehicles (AV), an embedded system needs to continuously translate noisy input from multiple sensors into driving commands, with the safety of passengers and pedestrians at stake. Quantification of

the uncertainty in the predictions of the models is paramount to avoid potentially catastrophic mishappenings (McAllister et al., 2017).

For example, take the problem of estimating the distance from the vehicle ahead for an AV. A simple point estimate could be dangerous in the presence of uncertainty: assume it takes the AV 100 meters to stop, and the distance point prediction is 110m but the true distance is 90m. In an emergency situation, the AV would not be able to stop before crashing into the vehicle ahead. If instead the model produces a 99.9%-confidence interval of the distance, we could use its lower end to ensure with high probability that there is always enough braking distance from the vehicle ahead.

In this work, we focus on ensembles of decision trees as predictors, which have been shown to be highly competitive in a variety of settings (Fernández-Delgado et al., 2014), including their online variants (Oza, 2005). Most previous work on decision trees has focused on one of the aspects of the problem: either providing online algorithms, or providing uncertainty estimates for batch algorithms. A number of approaches for quantifying the uncertainty of predictions in random forests (RF) have been proposed in the past, but they all assume a bounded, static dataset (Mentch and Hooker, 2016; Wager et al., 2014; Chipman et al., 2010; Meinshausen, 2006; Johansson et al., 2014b; Boström et al., 2017). On the other hand, most approximate tree algorithms that fit the computational and time limitations of the domain (Domingos and Hulten, 2000; Ben-Haim and Tom-Tov, 2010; Ikonomovska et al., 2011) do not provide uncertainty estimates. One notable exception are Mondrian Forests (Lakshminarayanan et al., 2016), whose runtime cost, however, is still prohibitive for limited-memory, real-time systems, as both its computational and memory costs increase with each incoming example.

In this study, we investigate approaches to quantify model uncertainty for regression tasks, when using ensembles of online decision tree learners with bounded computational and memory cost. We propose two online algorithms, one based on conformal prediction (Vovk et al., 2005), and another based on quantile regression (Meinshausen, 2006), and perform a large-scale empirical evaluation.

The proposed algorithms are meta-algorithms, in the sense that they allow to use different underlying learning algorithms. However, optimizing the selection of the underlying learner and tuning the approximate data structures used are out of the scope of the current study. We instead focus on the performance of the meta-algorithms themselves, while keeping all other parts constant.

In summary, our contributions are the following:

- We describe how to adapt inductive conformal prediction and quantile regression forests to the online setting by bounding their computational and memory use (Section 3).
- We report on an extensive experimental evaluation with thirty datasets of various sizes, including datasets that exhibit concept drift, and compare the proposed approaches against both a simple baseline and a state-of-the-art algorithm (Section 4).
- We empirically show that the proposed algorithms are able to maintain prediction error bounds at a rate better than the state-of-the-art, while being bounded in memory and computation, regardless of the number of data points in the input.[1]

1. All algorithm implementations, data, and experiment automation scripts used in this study are available as open source to ensure reproducibility at `https://github.com/thvasilo/uncertain-trees-reproducible`

2. Background

In this section, we formally define the problem of online interval prediction for regression, and provide descriptions of the algorithms that we later adapt to the online scenario.

2.1 Problem definition

We assume a sequentially arriving data stream $(\mathbf{X}_i, y_i)$, sampled from some fixed underlying distribution, where $\mathbf{X}_i \in \mathbb{R}^p$ are the feature vectors and $y_i \in \mathbb{R}$ the labels. Additionally $i \in \mathbb{N}^+$ indicates the index of a potentially unbounded dataset. Our goal is to learn a function $\Gamma(\mathbf{X}_i, \alpha) : \mathbb{R}^p \times (0, 1) \to [l, u]$, where $l, u \in \mathbb{R}$ and $l \leq u$, such that the probability of drawing an example $(\mathbf{X}_i, y_i)$ from the fixed underlying distribution where $y_i \notin \Gamma(\mathbf{X}_i, \alpha)$ is *less than or equal* to α, which is referred to as the *significance* level. We refer to $1 - \alpha$ as the *confidence* level.

In the conformal prediction literature, the property described above is termed *conservative* validity, as opposed to *exact* validity, for which the above probability is exactly α. This distinction makes our problem different from quantile regression (Koenker, 1996) which aims for exact validity as well.

Henceforth, for the description of the algorithms, we assume an ensemble that consists of a random forest ℓ of T decision trees, each denoted by $\ell_t, t \in [1, T]$. We consider an interval predictor to be *valid* if it makes errors, when $y_i \notin \Gamma(\mathbf{X}_i, \alpha)$, at a rate at most α.

2.2 Inductive Conformal Prediction

Conformal predictors (CP) output *prediction regions*, that is, sets of labels for classification and intervals for regression, instead of traditional point predictions. The main property of a conformal predictor is that it is *valid*: the expected error rate is upper-bounded by a predetermined significance level. By leveraging past experiences, the conformal predictor can guarantee that, as long as the examples are drawn from the same distribution, the probability of excluding the true label from the prediction region is less than or equal to the specified significance. In addition, it is a meta-algorithm, which can use any base point predictor for classification or regression. For a comprehensive review of conformal prediction see (Vovk et al., 2005).

The original framework for conformal prediction was designed for an online scenario, in which observations are received one-by-one. Each observation triggers a prediction, after which the true label is revealed, which in turn allows the predictor to be updated. However, this framework, which is called *transductive conformal prediction*, requires retraining the underlying model with the entire dataset for each new observation to obtain the so-called *non-conformity scores*. These scores measure how "out of the ordinary" examples are compared to the already observed data, and are used to produce the prediction intervals. The retraining requirement makes the method very costly computationally, and prevents it from being used straightforwardly for large datasets.

To overcome the computational cost of the transductive framework, another instantiation of the conformal prediction framework was proposed by Papadopoulos et al. (2002), named *inductive conformal prediction* (ICP). This approach is aimed at the batch setting, and does not require continuous re-training of the underlying model. Instead, it sets aside a

subset of the training examples, referred to as the *calibration set*, $\mathbf{C}$. The rest of the training data are used to train a batch predictor. The calibration set is used to produce sorted *non-conformity* scores, $\mathbf{S}$, which are again used to produce the prediction intervals. In regression, we commonly measure the non-conformity of an example by the absolute error of the model's prediction for that example. To determine the interval, we scan the (ascending) sorted list of non-conformity scores, $\mathbf{S}$, until we pass $\lfloor (1-\alpha) \cdot |\mathbf{S}| \rfloor$ values. The non-conformity score at this point gives us the prediction interval. Specifically, let $\phi = \mathbf{S}[\lfloor (1-\alpha) \cdot |\mathbf{S}| \rfloor]$ be the value selected from $\mathbf{S}$. Then the produced interval will be $\Gamma(\mathbf{X}_i, \alpha) = [\hat{y} - \phi, \hat{y} + \phi]$, where $\hat{y}$ is the point prediction of the model for $\mathbf{X}_i$.

Vovk (2002) proposed a way to adapt ICP for sequentially arriving datasets. The proposed method re-trains a new model from scratch after a fixed number of data points have been observed. This choice creates a trade-off between the computational efficiency of the algorithm and the predictive effectiveness in terms of the interval size. However, due to the need to store the complete training set, and completely retrain a new model at pre-set intervals, this method is still not a viable option for streaming datasets.

2.3 Quantile Regression Forests

Quantile Regression Forests (QRF) were introduced by Meinshausen (2006) with the purpose of extending random forests from computing the conditional mean to computing the full conditional cumulative distribution function (CDF). Following the notation by Meinshausen, we define the conditional CDF as

$$F(y \mid X = x) = P(Y \leq y \mid X = x). \tag{1}$$

Given Equation 1 we can define the β-quantile $Q_\beta(x)$ as the value for which the probability of Y being smaller than $Q_\beta(x)$ for a given x, is exactly β:

$$Q_\beta(x) = \inf\{y : F(y \mid X = x) \geq \beta\}.$$

Having access to the conditional CDF, we are able to produce prediction intervals by calculating the quantiles at the endpoints of the desired interval. For example, a prediction interval for significance level $\alpha = 0.1$ can be obtained by

$$\Gamma(x, \alpha) = [Q_{0.05}(x), Q_{0.95}(x)].$$

QRF, unlike other quantile regression methods, does not change the loss function of the underlying random forest algorithm. It instead calculates the conditional distribution as the *weighted distribution* of the observed labels across the forest. Therefore, only minimal changes to the underlying learning algorithm are needed in order to provide prediction intervals. Meinshausen shows that, under some mild assumptions, QRF is a consistent estimator of the conditional distribution.

Let us describe batch QRF briefly. It grows trees in the random forest as a regular RF algorithm. However, at every leaf, it stores all the label values, rather than only their average. Each new observation x is routed down all trees of the forest until it reaches a leaf. Denote $\texttt{leaf}(x, \ell_t)$ the leaf x reaches for tree $\ell_t \in \ell$. The weight of the example at each tree is given by $w_i(x, \ell_t) = \mathbb{1}_{\mathbf{X}_i \in \texttt{leaf}(x, \ell_t)} / |j : \mathbf{X}_j \in \texttt{leaf}(x, \ell_t)|$, i.e., the reciprocal of the number

of observations already in that leaf for tree ℓ_t. To get the observation's weight over the whole forest, we average over the individual trees: $w_i(x) = T^{-1} \sum_{t=1}^{T} w_i(x, \ell_t)$. The estimate for the conditional distribution is then given by the weighted sum over all the observations in the corresponding leaves for which the label is less than or equal to the requested y:

$$\hat{F}(y \mid X = x) = \sum_{i}^{N} w_i(x) \mathbb{1}_{\{Y_i \leq y\}} \tag{2}$$

where N is the number of data points in the forest.

QRF is a batch method, which requires access to the complete dataset beforehand, as we need to ensure the weights of all examples sum to one. In addition, in order to provide access to the full conditional distribution, the labels from all N examples must be stored in the leaves of the trees, in a mapping from the example weight to the corresponding label. This requirement imposes a prohibitive memory cost when datasets are massive or unbounded, which makes the algorithm unsuitable for the online setting.

3. Methods

This section describes the algorithms developed for this work, one based on inductive conformal prediction, and one on quantile regression forests.

3.1 Conformal Prediction with Online Regression Forests

As mentioned in Section 2, inductive conformal prediction was proposed by Papadopoulos et al. (2002) as an offline method, while the modification proposed by Vovk (2002) requires constant retraining of a model at preset intervals, and maintaining the complete training set. Both methods are thus unsuitable for settings where storing the complete dataset is impossible, e.g., when streaming massive amounts of data or when the dataset is unbounded.

Our proposed algorithm overcomes these issues, and removes the need to store all data points. Instead of retraining a new model with an increasing subset of the observed data, we use a single-pass, online model which continuously incorporates information as new examples arrive. In addition, by using an online random forest, we are able to maintain an up-to-date calibration set without setting aside examples exclusively for it. We keep the size of the calibration set bounded, thus making the memory requirements of the algorithm constant. Algorithm 1 shows the training function, and Algorithm 2 the corresponding prediction one.

3.1.1 Algorithm Description

We start by describing the training algorithm shown in Algorithm 1. It uses a combination of the online bagging algorithm by Oza (2005) and the out-of-bag conformal regression algorithm by Johansson et al. (2014a). Oza made the observation that for a large number of samples, the binomial distribution used to select samples in bagging tends to a Poisson(1) distribution. Therefore, this distribution can be used to approximate the bagging process online when we do not have access to the number of samples beforehand. Given an example to train on, for each member of the ensemble we make a draw k from a Poisson distribution with rate parameter $\lambda = 1$. If the drawn sample k is larger than zero, we use the example to train the member of the ensemble (with weight k). Oza showed that this process converges

Algorithm 1: OnlineCP Training (ℓ, $(\mathbf{X}_i, y_i)$), λ

input : ℓ: the current decision tree ensemble; $\mathbf{C}$: the set of calibration examples; $(\mathbf{X}_i, y_i)$: a labeled training example, c: max calibration set size.
output : $\mathbf{C}$: Updated set of calibration examples; ℓ_{oob}: a mapping from calibration examples to learners.

Set *isOutOfBag* to False
foreach $\ell_t \in \ell$ **do** `// in order` $t \in [1, T]$
  k = Poisson(1)
  if $k > 0$ **then** `// example in-bag for` ℓ_t
    Train ℓ_t with $(\mathbf{X}, y)$, weighted by k
  else `// example out-of-bag for` ℓ_t
    Set *isOutOfBag* to True
    Add ℓ_t to ℓ_{oob} for $\mathbf{X}_i$
if *isOutOfBag* **then** `//` $\exists\ \ell_t : (\mathbf{X}_i, y_i)$ `was out-of-bag`
  if $|\mathbf{C}| < c$ **then**
    Add $(\mathbf{X}_i, y_i)$ to bounded set of calibration examples, $\mathbf{C}$
  else
    Remove oldest example from set $\mathbf{C}$
    Add $(\mathbf{X}_i, y_i)$ to set $\mathbf{C}$

Algorithm 2: OnlineCP Interval Prediction (ℓ, $\mathbf{X}_i$, $\mathbf{C}$, α)

input : ℓ: the current decision tree ensemble; $\mathbf{X}_i$: an unlabeled example; $\mathbf{C}$: the calibration set; α: the desired significance level.
output : $\Gamma(\mathbf{X}_i, \alpha)$: the prediction interval for $\mathbf{X}_i$ at the requested significance α

foreach $\ell_t \in \ell$ **do** `// in order` $t \in [1, T]$
  Compute the prediction $p_t = \ell_t(\mathbf{X})$
$\hat{y} = \frac{1}{T} \sum_t p_t$ `// Compute ensemble prediction`
Compute the sorted non-conformity scores $\mathbf{S}$ using Algorithm 3
$\phi = \mathbf{S}[\lfloor (1 - \alpha) \cdot |\mathbf{C}| \rfloor]$
$\Gamma(\mathbf{X}_i, \alpha) = [\hat{y} - \phi, \hat{y} + \phi]$
return $\Gamma(\mathbf{X}_i, \alpha)$ `// Return the prediction interval`

Algorithm 3: Update Non-conformity Scores (ℓ, $\mathbf{C}$)

input : ℓ: the current decision tree ensemble; $\mathbf{C}$: the calibration set, $(\mathbf{X}_i, y_i)$, $i \in [1, c]$; ℓ_{oob}: mapping from calibration examples to learners.
output : $\mathbf{S}$: the sorted set of non-conformity scores

foreach $(\mathbf{X}_i, y_i) \in \mathbf{C}$ **do**
  `// Only use learners for which` $(\mathbf{X}_i, y_i)$ `was oob`
  foreach $\ell_t \in \ell_{oob}(\mathbf{X}_i)$ **do**
    $\hat{y}_i^t = \ell_t(\mathbf{X}_i)$ `// Compute oob prediction`
  $\hat{y}_i = \frac{1}{|\ell_{\text{oob}}(\mathbf{X}_i)|} \sum_t \hat{y}_i^t$ `// Aggregate oob predictions`
`// Compute non-conformity scores.`
$\mathbf{S} = |\mathbf{y} - \hat{\mathbf{y}}|$ `//` $\mathbf{y}$ `true labels,` $\hat{\mathbf{y}}$ `oob predictions`
Sort and return the non-conformity scores $\mathbf{S}$

to the equivalent batch bagging model. We note that λ can be set to larger values as done by Bifet et al. (2010b), however we follow the original work by Oza and set $\lambda = 1$.

If k is zero for some tree in the forest, we add the example to a bounded set, $\mathbf{C}$, of calibration examples. This set is the equivalent of the *calibration set* in batch inductive conformal prediction. The set is kept to a fixed size c by using a FIFO policy. When it is time to make a prediction, these examples are used to compute the *non-conformity scores* by using Algorithm 3. To ensure we compute non-conformity scores only on out-of-bag examples, we keep track of the learners for which the current example is out-of-bag, by using a mapping from example identifier to learner index.

The first part of the prediction step in Algorithm 2 works similarly to any bagging algorithm. It computes and averages the predictions from each member of the ensemble (Algorithm 2, lines 1-3). However, in order to compute the interval size, it also needs to have up-to-date non-conformity scores, which Algorithm 3 is responsible for. This algorithm takes the calibration set $\mathbf{C}$, current ensemble ℓ, and the ℓ_{oob} mapping as input, and produces a sorted (ascending) list $\mathbf{S}$ of non-conformity scores. As in batch ICP for random forests (Johansson et al., 2014a), our algorithm makes a prediction for each example in the calibration set by using only the trees for which it was an out-of-bag sample. Then, it uses a *non-conformity function* to compute the non-conformity scores. For this work, we choose the absolute error as our non-conformity function, which is a standard choice for conformal regression models.

Once we have the non-conformity scores $\mathbf{S}$, we can calculate the interval for the prediction. Given the desired significance level α and a sorted list $\mathbf{S}$ with c elements, the algorithm picks the element in the list whose index is $\lfloor (1-\alpha) \cdot c \rfloor$. The value $\phi = \mathbf{S}[\lfloor (1-\alpha) \cdot c \rfloor]$ represents half the interval width for the current prediction at the given significance level α. Finally, we use this interval width in line 6 of Algorithm 2 to produce a prediction interval for the example.

This version of ICP produces the same interval size for all examples, given a fixed calibration set $\mathbf{C}$. While techniques that adapt the interval size based on an approximation of the difficulty to predict each example exist (Papadopoulos et al., 2011; Papadopoulos and Haralambous, 2011), we elect to use the simpler version of ICP. Since the calibration set is continuously updated, the interval sizes will also be updated based on the performance of the underlying learner.

Our proposed method is designed as a meta-algorithm, therefore any online tree-based regression algorithm can be used as the underlying learner. For this paper, we use the Fast and Incremental Model Tree (FIMT) algorithm (Ikonomovska et al., 2011). FIMT is an extension to the Very Fast Decision Tree (Domingos and Hulten, 2000) and it inherits its memory management features, such as deactivating the least promising leaves or dropping unpromising attributes in order to keep the memory requirements of the algorithm bounded. It also provides a concept drift adaptation mechanism that allows our methods to adapt to changes in the underlying data distribution. We call this meta-algorithm CPExact.

3.1.2 Optimizations

In the previous section, we mention that the set of calibration examples is bounded, that is, we ensure that we never have to keep more than $|\mathbf{C}| = c$ examples in memory. In our case,

the size of the calibration set affects the computational cost significantly, as in the worst case we need to make an ensemble prediction for each calibration example before making an interval prediction.

The naïve way to implement Algorithm 3 would require each member of the ensemble to make a prediction for each (out-of-bag) example in the calibration set. However, this step is unnecessary; by storing the predictions of each learner for which a calibration example was out-of-bag, we only need to update that prediction when the learner itself gets updated, otherwise we can use its previous prediction. We call this technique prediction memoization.

While this small optimization can reduce the number of required predictions somewhat, the factor of c still remains, which implies that for all ensemble members that have been updated since the last prediction, up to c new predictions are needed to compute the updated calibration scores. In our implementation, we mitigate this issue by using parallelism across the members of the ensemble and updating the calibration scores asynchronously as soon as the training step has finished, thus allowing for overlapping computation until the next call to the prediction function.

3.1.3 Approximate algorithm

In the common prequential evaluation scenario of online learning (Gama et al., 2009), the model makes a prediction for an example before being shown its label, which is then used (together with the example) to update the model. Given that the model is updated for each new example, in the worst case, we need to calculate the non-conformity scores for the complete calibration set for each example, thus incurring a high computational cost. While prediction memoization and asynchronous execution can reduce the runtime of CPExact to some extent, the core computational constraint of updating the predictions for the complete calibration set remains.

To address this computational issue, we propose to use an approximate solution to the computation of the non-conformity scores. In particular, we only use the prediction from the model obtained when the new example enters the calibration set. In other words, regardless of whether a tree has been updated, we never re-evaluate its predictions for all of its calibration examples. Rather, we only make predictions for the newly introduced examples, i.e., those examples that have been added to the calibration set since the last time we made a prediction. Note that, as more and more examples accumulate, the predictions of the underlying models for a given example will change progressively less. By not making predictions on older examples, we achieve significant computational savings and remain true to the online spirit of the algorithm. Moreover, by constantly introducing new examples in the calibration set, we ensure that it remains an accurate representation of the non-conformity of the examples given the current model. We call this meta-algorithm CPApproximate.

CPApproximate has an inherent tradeoff: when many samples are used in its calibration set, it might end up using several stale predictions to calculate the intervals. As we show in Section 4.4, this tradeoff has an effect on the ability of the algorithm to adapt to concept drift. By maintaining predictions that were made using a distribution different from the current concept, the algorithm is slow to adapt its intervals to the newly introduced concept. This delay can lead to intervals that do not maintain the requested error rate.

3.2 Online Quantile Regression Forests

In Section 2.3, we mention that batch QRF requires the complete dataset to be available in order to assign a weight to each example, and has large memory requirements due to the need to store a mapping from the weight of each example to its label, for every data point.

The main idea for our adaptation of the algorithm to the online setting is to use an approximate representation of the label values to bound the memory requirements of the algorithm and doing away with the need to maintain a weight for each example. In particular, we employ the *KLL* streaming quantile estimates proposed by Karnin et al. (2016) to store the approximate cumulative distribution of label values in the leaves. By using the online bagging algorithm of Oza (2005) in combination with a feature subset selection scheme (Gomes et al., 2017), we propose OnlineQRF (oQRF): online regression forests that provide quantile estimates.

We now briefly explain the core concepts of the KLL sketch, and refer the reader to Karnin et al. (2016) for a detailed description. Estimating streaming quantiles is an established problem in data management. The task is to estimate quantile values from an incoming stream of potentially infinite real values, by utilizing minimal memory and computational cost. See the work by Greenwald and Khanna (2016); Wang et al. (2013) for surveys on the subject. The problem can be formulated as finding the approximate rank of a given value x in a sorted set of n items. An ϵ approximate quantile sketch returns the approximate rank up to an additive error of ϵn. As customary, the KLL algorithm gives a probabilistic guarantee that the resulting sketch is an ϵ-approximation with probability $1 - \delta$ (over its executions). Another important property is the *mergeability* of a sketch; this allows us to sketch different parts of a stream in parallel, and merge the partial sketches to a final sketch that has the same accuracy of a single sketch over the complete stream. This scenario is common in distributed settings (De Francisci Morales and Bifet, 2015), and we make use of this property to merge the quantile sketches of each tree in the ensemble into a single quantile prediction.

KLL belongs in the family of methods for quantile estimation that make use of *compactors*, a data structure that can store c items, with the same weight w. Compactors can compact their c elements into $c/2$ by first sorting them, and then discarding either the odd or even items, while doubling the weight of the remaining items. Multiple compactors can be chained together to create a rank estimator; each compactor takes a stream as input and outputs a halved stream to be used as input for the next compactor, until a sequence of length c is generated, which can be simply stored in memory. For a new value x, we can estimate its approximate rank in the original stream from its rank the final sequence of c items. The KLL sketch uses a cascade of such compactors and samplers to process the stream and selectively store a small number of elements, which grows logarithmically with the number of elements in the stream. The only parameter of the algorithm is K, which determines a tradeoff between the memory cost and accuracy of the algorithm. Higher K means more elements will be stored, but also higher accuracy. Note that the error ϵ is not set explicitly by the user, but depends on K.

Specifically, the KLL sketch is an improvement upon the quantile sketch of Agarwal et al. (2012), which chooses whether to keep the even or odd positions during compaction by flipping a fair coin, and subsamples the stream before feeding it to the first compactor. KLL

improves the memory footprint of the algorithm by Agarwal et al. by using compactors of exponentially decreasing size and replacing some of the compactors with a sampler, thus resulting in a mergeable sketch with a space complexity of $O((1/\epsilon)\log^2\log(1/\delta\epsilon))$, which is currently the state of the art. Their limited memory footprint and mergeability make them perfect candidates to be used as quantile estimators for OnlineQRF.

Similarly to online conformal prediction, OnlineQRF is meta-algorithm for which any tree-learning algorithm can be used. For consistency, we again choose to use a variation of the FIMT algorithm, but this time we maintain an efficient quantile sketch at each leaf. We use these sketches at prediction time to provide the prediction intervals of the forest. The parameter K of the KLL sketches constitutes the only parameter of the algorithm.

While QRF is a consistent estimator of the conditional distribution, i.e., the empirical distribution used at the leaves of QRF converges to the real conditional distribution when the number of observations tends to infinity, OnlineQRF lacks this property. Indeed, the use of sketching introduces a fixed error in the approximation of the distribution, which is guaranteed to be smaller than ϵ, but is nevertheless constant. Therefore, adding observations to the sketch does not improve the approximation of the real conditional distribution beyond a given point. Looking at the issue from a different angle; we cannot hope to approximate an arbitrary distribution with infinitesimal error by using a fixed-size sketch. The loss of consistency of the estimation is thus a necessary sacrifice to make the algorithm viable in an online setting.

Algorithm 4: OnlineQRF Training (ℓ, $(\mathbf{X}_i, y_i)$), λ

input : ℓ: the current decision tree ensemble; $(\mathbf{X}, y)$: a labeled training example.

foreach $\ell_t \in \ell$ **do** `// in order` $t \in [1, T]$
  k = Poisson(1)
  if $k > 0$ **then**
    Train ℓ_t with $(\mathbf{X}, y)$, weighted by k
    Add y_i to H_t, the sketch at the leaf where the example is sorted into.

Algorithm 4 describes the training step of OnlineQRF, which is similar to the original training for bagged learners by Oza (2005). However, we modify the underlying learners to maintain an online quantile sketch of the labels. After updating the learner with the example $(\mathbf{X}_i, y_i)$, we add the label y_i to the corresponding sketch at the leaf, which is a constant time operation.

We describe the prediction process of OnlineQRF in Algorithm 5. As opposed to the original QRF algorithm, which requires summing over all the data points (Eq. 2), our algorithm only requires $T-1$ merge operations, and one operation to calculate the quantiles, both of which take constant time.

Because of the online nature of the sketches, we are able to merge them incrementally, and so maintain the constant memory requirements of the algorithm. Additionally, when executing the algorithm in parallel, we are able to use a parallel reduce operation, therefore further reducing the runtime of the predict operation, in which the sketch merge is the most expensive operation. Given that the merge operation is the most time consuming operation for the algorithm, we also experimented with approximate alternatives, such as computing the quantiles for each sketch separately and averaging the results, but the

Algorithm 5: OnlineQRF Interval Prediction (ℓ, $\mathbf{X}_i$, α)

input : ℓ: the current decision tree ensemble; $\mathbf{X}_i$: an unlabeled example; α: the desired significance level.
output : $\Gamma(\mathbf{X}_i, \alpha)$: the prediction interval for $\mathbf{X}_i$ at the requested significance α

$\hat{H} = \emptyset$ `// Initialize empty sketch`
foreach $\ell_t \in \ell$ **do** `// in order` $t \in [1, T]$
  Drop $\mathbf{X}_i$ down the tree ℓ_t until we arrive at a leaf
  Retrieve the sketch of the leaf, H_t
  Merge H_t into $\hat{H}$
$\Gamma(\mathbf{X}_i, \alpha) = [Q_{\alpha/2}(\hat{H}), Q_{1-\alpha/2}(\hat{H})]$
return $\Gamma(\mathbf{X}_i, \alpha)$ `// Return the calculated interval`

intervals produced were not valid. This result indicates that information from all the trees needs to be aggregated in order for the algorithm to work as expected, which is also the case for batch QRFs.

One positive characteristic that OnlineQRF shares with the online conformal prediction methods developed for this work is that the significance level α does not need to be set from the start. Rather, it is a runtime parameter that the user can set when making a prediction. Therefore, the user may also request multiple significance levels without the need to train separate models, which is necessary in QR algorithms that modify the loss function, such as linear QR (Koenker, 1996).

4. Empirical evaluation

In this section, we present results from an extensive empirical investigation, concerning 20 small-scale datasets, drawn from a diverse set of domains, and another 10 datasets for studying the effect of concept drift, with millions of data points.

4.1 Experiment Design

The evaluation follows the prequential evaluation design that is commonly employed in online learning settings (Gama et al., 2009). For each example in the dataset, we first make a prediction, then update our metrics based on the predicted and true labels, and finally, reveal the true label to the algorithm. When reporting results over time for the concept drift datasets, we use tumbling windows of size 10 000, that is, we measure the mean performance of each algorithm for every 10 000 samples. Because random forests are non-deterministic, we repeat each experiment 10 times. We report the mean of the metrics across the repetitions and we show the standard deviation as a shaded area in Figures 5 and 6.

4.1.1 Metrics

We report the Mean Error Rate (MER), Relative Interval Size (RIS), and use Quantile Loss and Utility as combined metrics of the MER and RIS. The Mean Error Rate measures the percentage of errors that a method incurs, essentially the average of the 0-1 loss over the complete dataset:

$$\text{MER} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\{y_i \notin [l_i, u_i]\}},$$

where N is the size of the dataset and $[l_i, u_i]$ the predicted interval for $\mathbf{X}_i$. This metric quantifies the *validity* of the method; for a significance level α, a method should have an MER of at most α in order to be considered valid. For example, in our validity experiments, we set a significance level of 0.1, so we should observe an MER of at most 0.1.

However, the MER metric offers only a partial view of the performance of a method; it is easy to produce a naïve algorithm with a valid MER by always predicting very large intervals, which however would not be informative. To measure how informative the produced intervals are, a property sometimes referred to as the *efficiency* of the intervals (Vovk et al., 2005), we employ the Relative Interval Size (RIS) metric. RIS measures the average size of the intervals over the data, normalized by the range of values in the data to enable comparisons across datasets:

$$\text{RIS} = \frac{1}{N}\sum_{i=1}^{N} \frac{u_i - l_i}{\rho},$$

where ρ is the range of the dependent variable (label) of the dataset, $\rho = max(\mathbf{y}) - min(\mathbf{y})$.

Lower values are better for the RIS, with a value of one indicating a non-informative interval that covers the whole range of observed values for the target. Values larger than one are possible for methods that can produce interval predictions outside the range of observed values, such as the conformal prediction methods. RIS needs to be viewed together with the MER to get a complete picture of the performance of a method.

To ease presentation, we also employ two combined metrics, Quantile Loss and Utility. Quantile Loss is based on the established quantile error objective function used in quantile regression (Koenker, 1996). Since in our case we are dealing with intervals, we can take the quantile loss for each quantile, $Q_{\alpha/2}$ for the lower and $Q_{1-\alpha/2}$ for the upper, and sum their losses. So for the case of a correct interval prediction, when $y \in [l, u]$ we have:

$$\begin{aligned}\text{Quantile Loss}_{y\in[l,u]} &= \alpha(y - l) + (1 - (1 - \alpha))(u - y) \\ &= \alpha(u - l).\end{aligned}$$

While in the case where $y \notin [l, u]$ and, for example, $y < l$:

$$\begin{aligned}\text{Quantile Loss}_{y<l} &= (1 - \alpha)(l - y) + (1 - (1 - \alpha))(u - y) \\ &= \alpha(u - l) + (l - y).\end{aligned}$$

The loss is symmetric for the case where $y > u$. The purpose of this metric is to penalize methods that deviate from the requested error rate, according to the distance from the error rate, scaled by the size of the interval. Combining the above we can define Quantile Loss as follows:

$$\text{Quantile Loss} = \text{RIS} \cdot \alpha + \beta.$$

We use RIS here to normalize the interval widths and allow comparison across datasets. The factor β is the distance of the true y value from the closest interval boundary:

$$\beta = \frac{\mathbb{1}_{(y<l)}|y-l| + \mathbb{1}_{(y>u)}|y-u|}{\rho}.$$

In other words, β tells us how far outside the predicted interval the true value lies. Like the RIS, β is normalized by the range of the dependent. While the quantile loss is an established metric, it does not match exactly our problem definition, as it penalizes conservative methods whose MER is lower than the requested significance. For that purpose we also present a metric that does not penalize conservative methods, called Utility.

Utility is a single metric based on the time-utility functions commonly used in real-time systems (Ravindran et al., 2005). These functions combine two measurements: the *utility* of a result as a factor of the *time* it takes to obtain it. We set a time deadline after which the method is penalized, while before the deadline the method incurs no penalty. We use $(1 - \text{RIS})$ as the utility, and set the requested significance as the "time" deadline, with exponential decay. Formally, we define utility as:

$$\text{Utility} = \begin{cases} 1 - \text{RIS}, & \text{MER} \leq \alpha \\ (1 - \text{RIS}) \cdot \exp\left(-\gamma(\text{MER} - \alpha)\right), & \text{otherwise,} \end{cases}$$

where α denotes the requested significance level, and γ is set such that the half-life of the utility is at $1.5 \cdot \alpha$. In other words, once a method's MER passes the requested significance, its utility drops off exponentially, and reaches half its original value when the MER is 50% larger than the requested significance. This metric takes values from 0 (worst) to 1 (best) and indicates how useful the produced intervals are. If the algorithm is able to maintain the requested significance level, $1 - RIS$ indicates how tight the intervals are. Otherwise the intervals are invalid and we discount the utility accordingly. Finally, if the RIS is larger than one we set the utility to zero.

4.1.2 Baselines

We compare our algorithms with the state-of-the-art Mondrian Forest (MF) algorithm (Lakshminarayanan et al., 2016). MF makes use of Mondrian processes (Roy and Teh, 2009) to create tree-like online prediction models that use a hierarchy of Gaussians to model the dependent variable and can produce the full predictive posterior distribution. Mondrian Forest maintains a Gaussian at each *node* in the forest, consequently it has an unbounded computational cost that increases with each incoming example. In comparison, our methods perform learning only at the leaves, and are thus more efficient computationally. We provide more details for the method in Section 5. As done in the original work, we use the predicted mean and variance to extract Gaussian quantiles for the requested significance level.

We also employ a simple baseline as an illustration that the problem is non-trivial, similar to the one used by Lakshminarayanan et al. (2016). We compute the predicted mean and variance among the trees in an online forest that uses FIMT as the underlying learner, and

Dataset	Examples	Features	Domain
2dplanes	40768	10	Artificial
abalone	4177	8	Biology
ailerons	13750	40	Control
bank32	8192	32	Simulation
calHouse	20640	8	Housing
cpuAct	8192	21	Computing
elevators	16599	18	Control
energy	19735	27	Energy Use
friedman	40768	10	Artificial
house8	22784	8	Housing
house16	22784	16	Housing
kin8nm	8192	8	Robotics
mv	40768	10	Artificial
newsPop	39644	60	Web
puma8	8192	8	Robotics
puma32	8192	32	Robotics
qs240	6003	43	Bioinformatics
qs253	4332	26	Bioinformatics
sulfur	10081	6	Monitoring
yprop	8885	251	Drug Design

Table 1: The small-scale datasets used in this study.

take Gaussian quantiles to create the prediction intervals. We report the results of this method in Section 4.2.

4.1.3 Data

We use 20 small-scale datasets from a variety of domains to test the validity and efficiency of the algorithms, and 10 data sets to test their ability to deal with concept drift. The small-scale datasets are gathered from the OpenML repository (Vanschoren et al., 2013). Table 1 provides summary information about these datasets. We provide more detailed information about the small-scale data in Appendix A.1.

For the concept drift experiments, we follow Ikonomovska et al. (2011) and generate datasets using the three "Friedman" functions first introduced in Friedman's MARS paper (Friedman, 1991). The first Friedman function includes non-linear dependencies between five relevant features and the dependent variable, and in addition includes five irrelevant features that do not affect the dependent variable. All features are independent and uniformly distributed over $[0, 1]$. The second and third Friedman functions include four uniformly distributed features and a noise factor. The dependent variable simulates the impedance and phase shift in an alternating current circuit. We refer the reader to Friedman (1991) for more details on these functions.

As done by Ikonomovska et al. (2011), we introduce three different kinds of concept drift for each original function, and produce 1M data points for each dataset, for a total of nine artificial concept drift datasets. We provide more information about the exact function definitions in Appendix A.2, and give a brief overview of each type of concept drift below.

- *Local expanding*: The first type of drift appears only in two distinct regions of the input space. There are three change points, one every 1/4 of the data, which modify the generating function if the point lies in one of two designated regions. These regions expand with each consecutive change point.
- *Global reoccurring abrupt drift:* This type of drift is global and abrupt. After 1/2 of the examples have been observed, we abruptly change the generating function. After 3/4 of the examples have been observed, we change back to the original function.
- *Global slow:* The third type of drift is also global but gradual. Starting after 1/2 of the examples have been observed, we gradually introduce data points from a different concept, with a probability that linearly increases to one after 100k additional points have been observed. At the 3/4 point, we introduce a new concept in a similar manner, which completely replaces the previous concept after 100k points.

We also use a real-world dataset of flight delays. [2] This dataset contains flight arrival and departure details for all commercial flights in the US during 2008, with the task being to predict the delay of flights based on attributes such as the age of the airplane or the day of the year. This dataset was already used in the evaluation of Mondrian Forests (Lakshminarayanan et al., 2016), and we have re-created the same splits of the data in samples with 700K, 2M, and 5M data points. [3] We perform a standard pre-processing step to transform some categorical attributes, such as day of week or month, to a one-hot (binary) representation, resulting in a dataset with fifty-five features.

4.1.4 Parameter Settings

For all the experiments, we use an ensemble size of 10. We use $|C| = 1000$ calibration examples for the conformal prediction algorithms, while for OnlineQRF we use quantile sketches at the leaves with the default accuracy parameter $K = 200$, as recommended by Karnin et al., which yields a normalized rank error of 1.65%.

We experimented with different numbers of K and calibration examples, and these parameters did not significantly affect the results. We have implemented these algorithms in the MOA online learning framework (Bifet et al., 2010a). For the Mondrian Forest, we use the optimized implementation available in the `scikit-garden` [4] Python library. We have verified with the author of the original work that this implementation produces correct results in online learning mode for the parameter settings we use. Specifically, we set `min_samples_split` = 2, which determines the minimum number of samples necessary for a leaf to be considered for a split. We note that for values larger than this, MF requires that the entire dataset be stored in memory at the leaves, making it infeasible for the large datasets and limited resources common in online learning.

2. `http://stat-computing.org/dataexpo/2009`
3. We thank M. Deisenroth for providing the pre-processing scripts.
4. `https://github.com/scikit-garden/scikit-garden`

Dataset	MER	RIS	Dataset	MER	RIS
2dplanes	0.394	0.10	house16	0.367	0.11
abalone	0.727	0.05	kin8nm	0.659	0.14
ailerons	0.271	2.89	mv	0.147	0.16
bank32	0.413	0.20	newsPop	0.053	0.06
calHouse	0.524	0.15	puma8	0.581	0.22
cpuAct	0.525	0.15	puma32	0.502	0.24
elevators	0.392	560.95	qs240	0.448	0.28
energy	0.137	111.9	qs253	0.510	0.48
friedman	0.379	0.12	sulfur	0.462	0.04
house8	0.405	0.09	yprop	0.073	8.32

Table 2: Mean Error Rate and Relative Interval Size for the Gaussian quantiles baseline. Desired MER is 0.1.

We set the requested significance level α to 0.1 for the validity experiments, but explore the effect of different significance levels in Section 4.5.

4.2 Baseline

We start by reporting the validity and interval sizes of the Gaussian quantiles baseline method, as a way to illustrate the difficulty of the problem. Table 2 shows the MER and RIS for the Gaussian quantiles baseline, with a desired MER of 0.1. As can be seen, the observed mean error rate is several times larger than this; the average MER over all datasets is 0.4. This indicates that merely assuming a Gaussian distribution over the trees' predictions is not an appropriate way to quantify the uncertainty.

4.3 Small-scale data

Figure 1 breaks down the MER and RIS for each method, while Figure 2 reports the utility for the small-scate data. Each point in Figure 1 represents the average MER and RIS of a method for a single dataset, over the ten repeats of each experiment.

The ability to maintain the desired error rate, and the required interval size to do so, differ significantly between methods. OnlineQRF is the overall best performing method for these datasets, managing to strike a balance between maintaining the requested error rate and having small prediction intervals. On the other hand, CPExact is the only method that is consistently valid for all the datasets, i.e., stays below the requested 0.1 error level on average. However, it will sometimes produce intervals with an RIS close or larger than one, leading to a utility near zero for those datasets. As a result, its mean utility is lower than OnlineQRF, with CPApproximate following closely. Mondrian Forests, while generally producing tight intervals, manage to maintain the requested error rate on average for only nine out of the twenty datasets, leading to the lowest utility among all methods.

In Figure 3 we can see that, modulo one outlier for the conformal prediction methods, all methods perform roughly equally in terms of quantile loss. A slight advantage for the

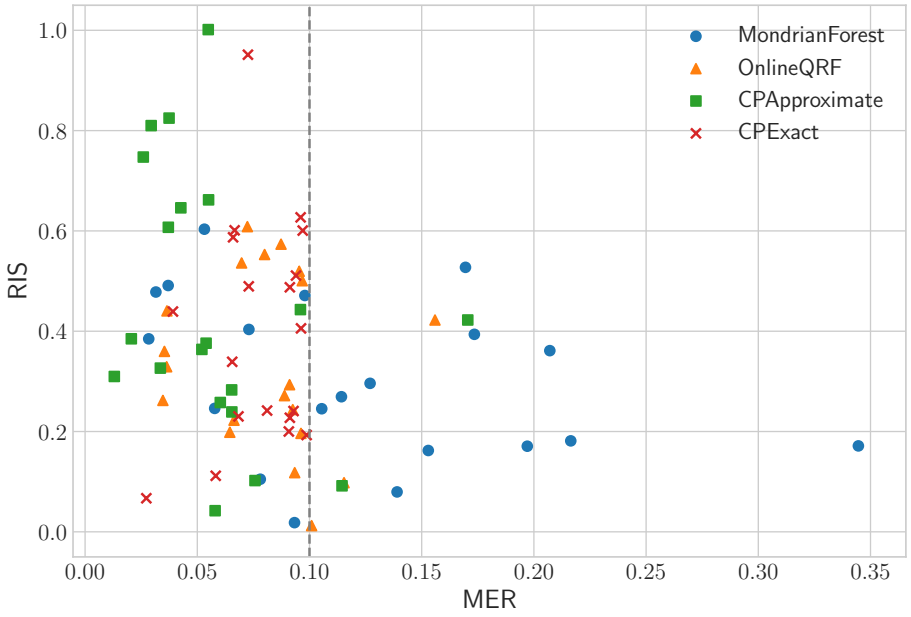


Figure 1: MER and RIS for small-scale data. Each point corresponds to one dataset. The dashed vertical line indicates the desired error rate (significance) at 0.1.

methods proposed in this paper can be seen in Figure 2 in terms of utility, which unlike quantile loss does not penalize methods with conservative predictions.

These results can, to some degree, be explained by the nature of each algorithm. It is likely that aggregating the mixture of Gaussians produced by the Mondrian Forest to a single Gaussian distribution to be used for quantile estimation is the main source of errors for the method. In MF, each node in the path from the root to the leaf for an example produces a mean and a variance. These are averaged to produce one mean and variance per tree. Then the means and variances of all trees in the ensemble are averaged again to produce a final Gaussian distribution that is used to produce the intervals. These two levels of aggregation are a potential source of error for MF.

Unlike MF and the CP methods, the intervals of OnlineQRF never lie outside the range of observed labels, thus making the method less likely to produce outliers which the conformal prediction methods suffer from, as exemplified in Figure 1. Finally, we note the ability of CPApproximate and particularly CPExact to maintain the expected error rate. However, this result often comes at the cost of larger interval sizes and, as shown later in Section 4.6, significantly increased runtime for CPExact.

4.4 Concept Drift

In this section, we examine the validity of the algorithms on datasets with concept drift.

We illustrate the performance of the algorithms on the artificial Friedman #1 data in Figure 5. The results are consistent with the other Friedman datasets, which we summarize in Figures 7 and 8 for utility and quantile loss respectively, and Figure 4 for a combined view into MER and RIS.

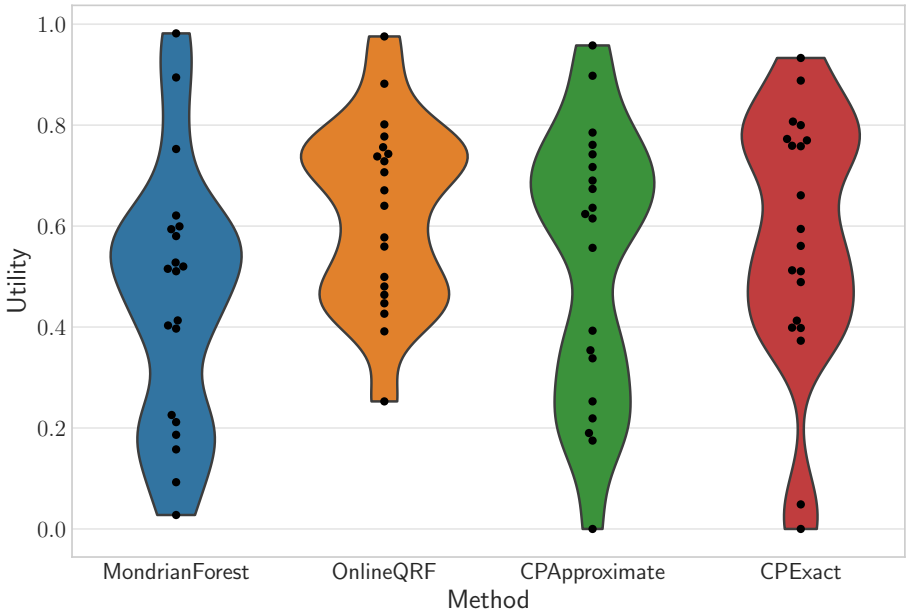


Figure 2: Utility for the small-scale data. Higher is better.

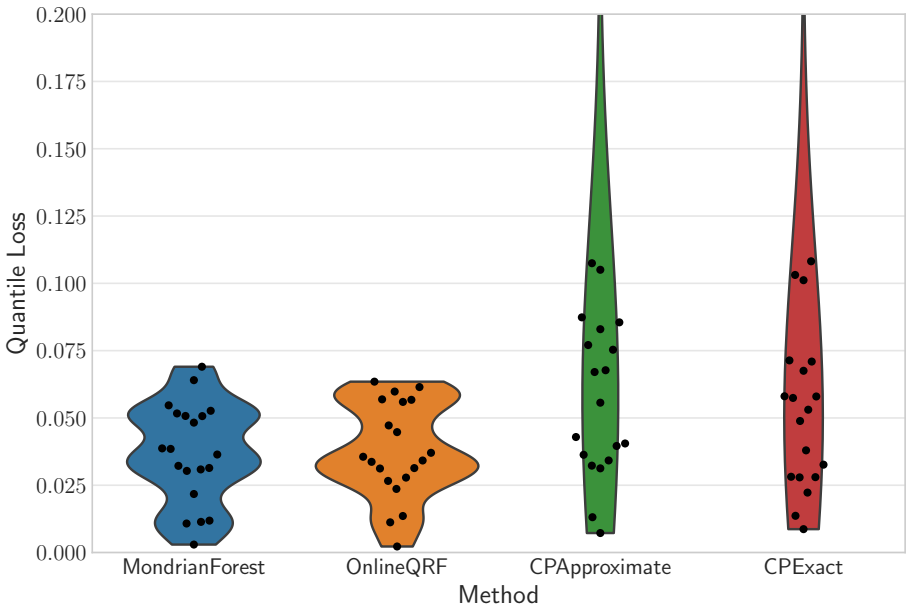


Figure 3: Quantile loss for the small-scale data. Lower is better. The conformal prediction methods have an outlier for the `yprop` dataset which is not shown here to ease presentation.

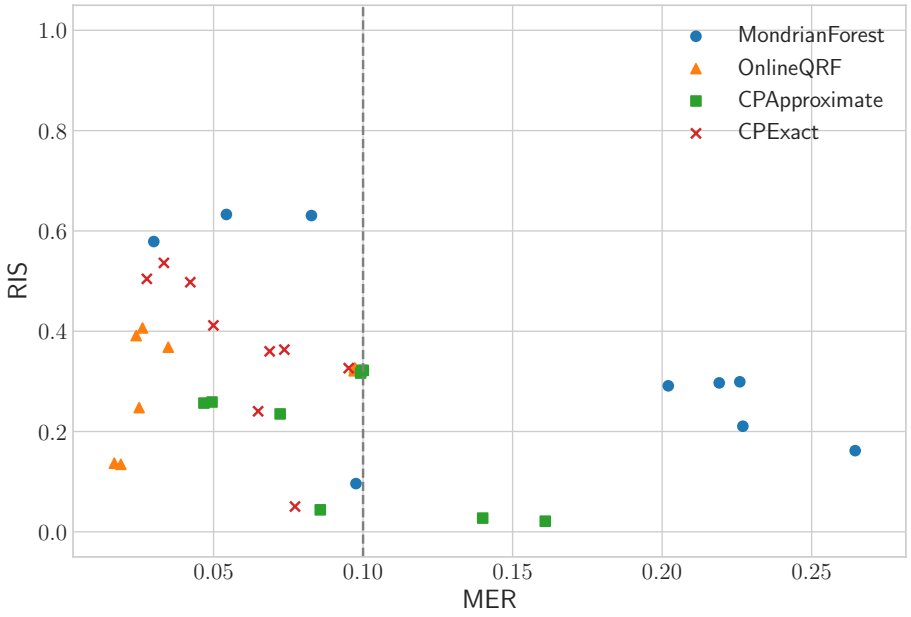


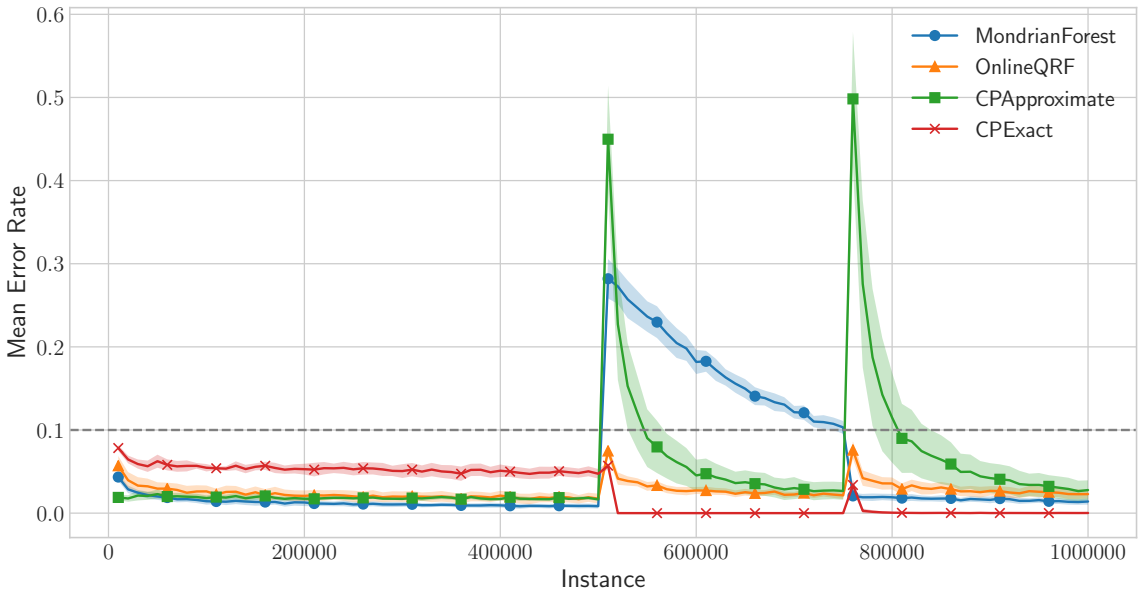
Figure 4: MER and RIS for Friedman data. Each point corresponds to one dataset. The dashed vertical line indicates the desired error rate (significance) at 0.1.

Starting from Figure 5, the first concept drift occurs at 500k examples, and the second at 750k. We can observe different behaviors of the algorithms. Starting with CPExact, after the first concept drift, its RIS rises rapidly to account for the additional uncertainty in the model. As a result, it manages to maintain the requested error rate throughout the experiment, but its intervals remain large after the concept drift. In contrast, CPApproximate maintains a stable RIS throughout the experiment, with a MER that after an initial spike drops off exponentially to return to the requested error rate. This behavior follows from the fact that the method does not update the old non-conformity scores.

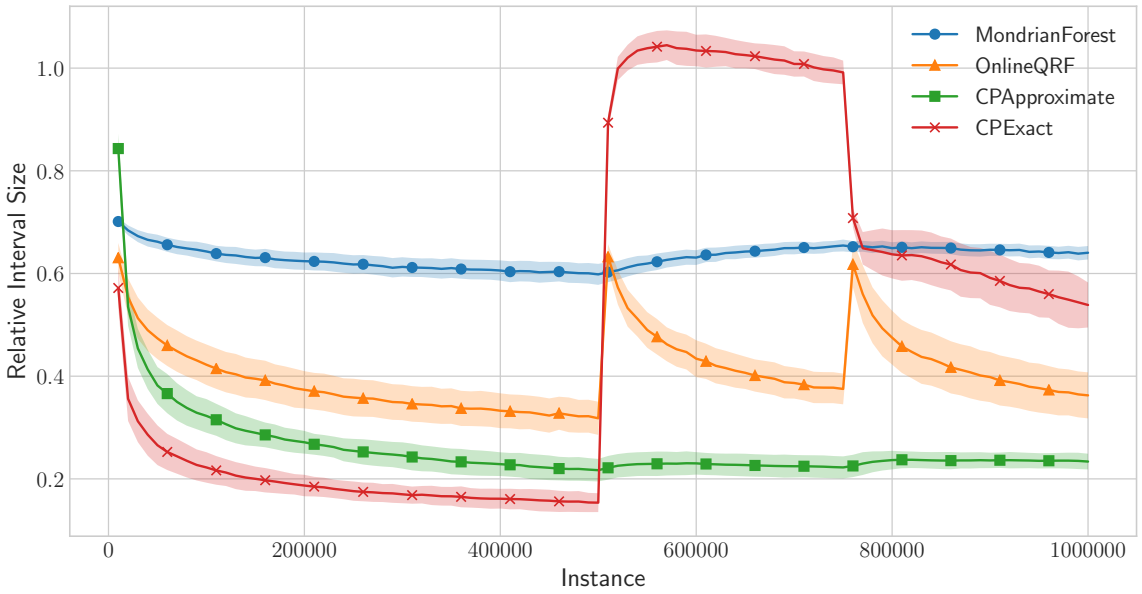
OnlineQRF lies in the middle between the two; it also exhibits a spike in RIS at the concept drift points, but it is quick to recover and, importantly, it remains valid throughout the experiment. Finally, Mondrian Forest has consistently wide intervals, and its error is slow to recover after the first concept drift.

As with the small-scale datasets, OnlineQRF is a strong performer, providing consistent results. However, in this set of experiments, CPApproximate performs better than CPExact due to its tight intervals, and actually has the best average utility among all methods, with OnlineQRF nearly matching it. Mondrian Forest is only able to maintain the requested error rate for half of the datasets (Figure 4), and even in those it mostly has the lowest utility.

The performance of the algorithms in the real-world flight delay data is shown in Figure 6 for the two million examples dataset. The concept drift occurs after having observed around 1.1 million examples, which causes the MER of all the algorithms to spike considerably. Mondrian Forest and OnlineQRF are able to return to the desired error rate before observing 1.2 million examples. On the other hand, CPApproximate requires close to 900 000 additional

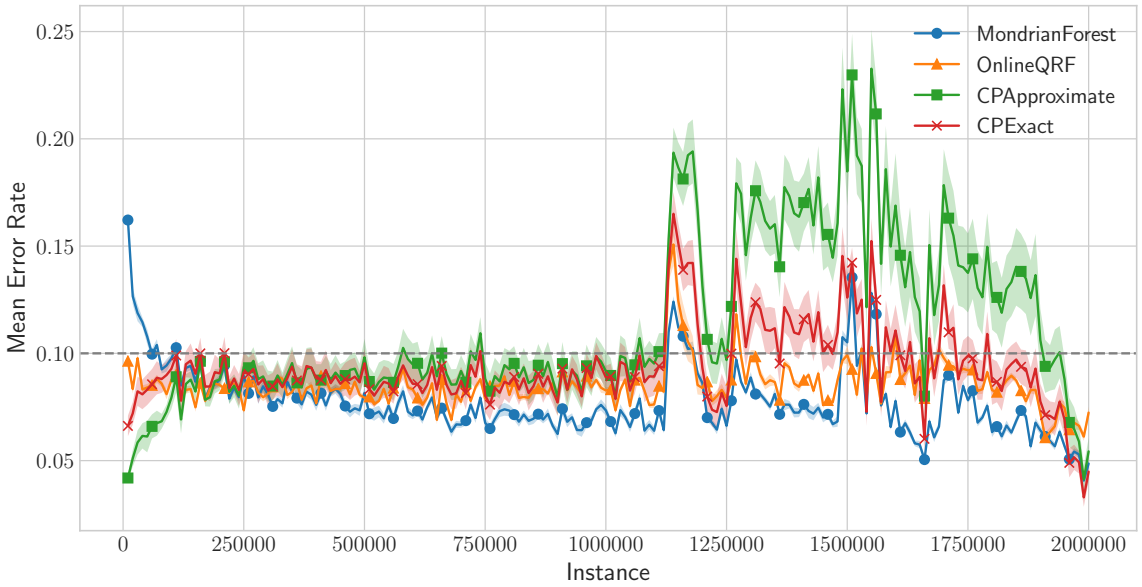


(a) Friedman #1 Mean Error Rate. The dashed horizontal line indicates the desired error rate at 0.1.

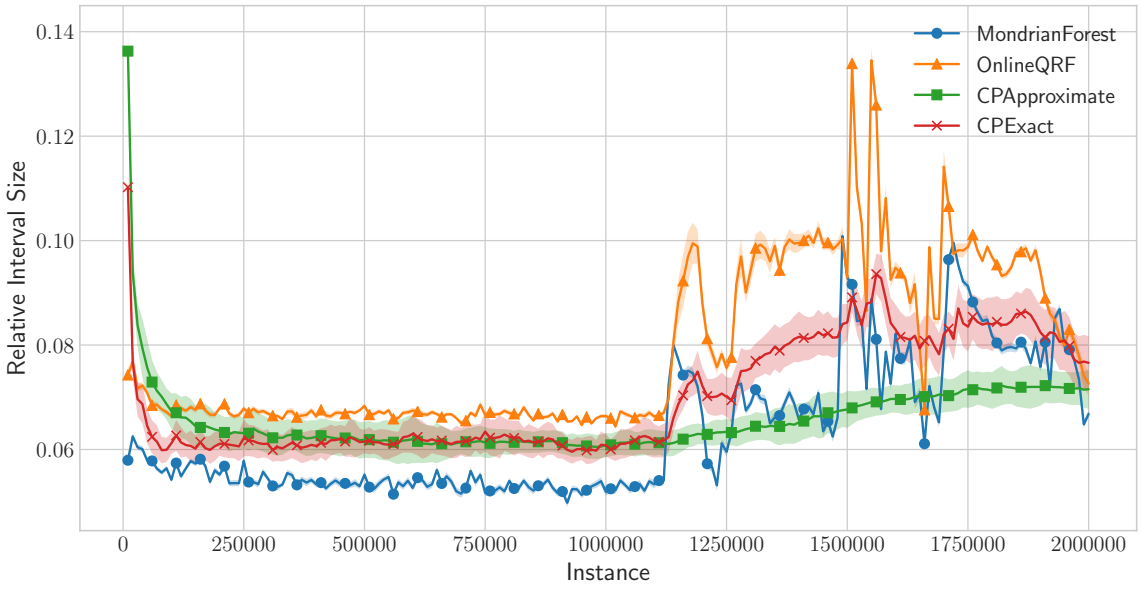


(b) Friedman #1 Relative Interval Size.

Figure 5: MER and RIS evolution over time for Friedman #1 with global abrupt drift.



(a) Flight delay Mean Error Rate.



(b) Flight delay Relative Interval Size.

Figure 6: MER and RIS evolution over time for the flight delay dataset.

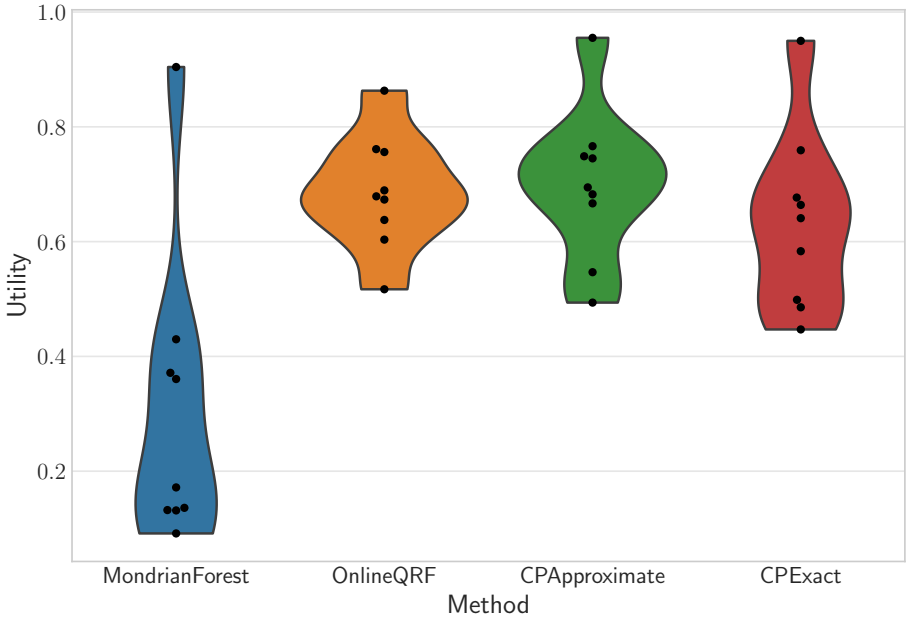


Figure 7: Utility for the concept drift data. Higher is better.

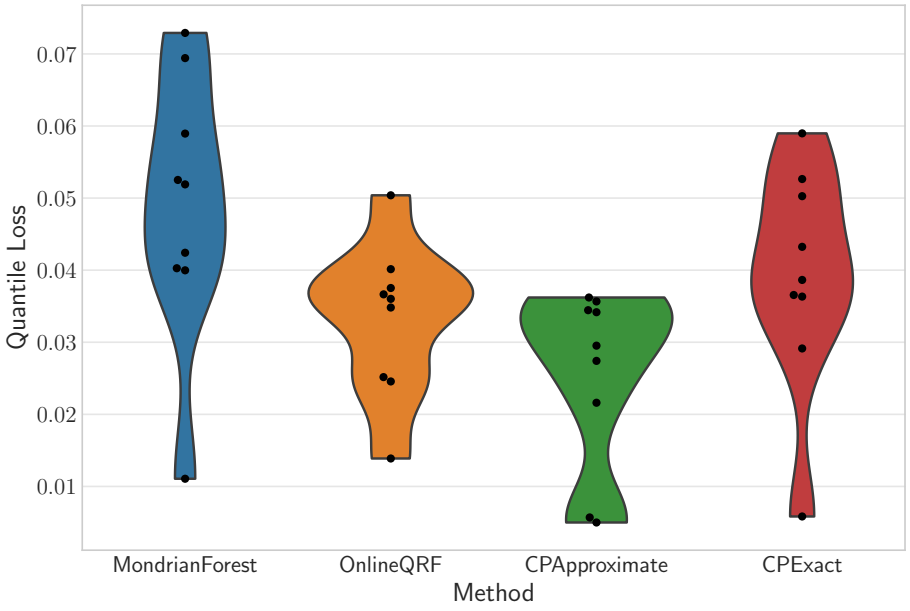


Figure 8: Quantile loss for the concept drift data. Lower is better.

Method	Metric	α				
		0.3	0.2	0.1	0.05	0.01
MF	MER	0.28	0.20	0.13	0.094	0.062
	RIS	0.19	0.23	0.29	0.349	0.460
oQRF	MER	0.23	0.14	0.07	0.036	0.015
	RIS	0.20	0.23	0.31	0.345	0.510
CP_{approx}	MER	0.22	0.13	0.06	0.032	0.006
	RIS	0.21	0.31	0.48	0.785	1.130
CP_{exact}	MER	0.25	0.16	0.08	0.039	0.009
	RIS	0.28	0.37	0.57	0.622	0.944

Table 3: MER and RIS for different significance levels. The target MER is α where α the significance level. See Appendix B for detailed results.

examples before returning to under the requested error rate. CPExact is able to adapt faster than CPApproximate, and exhibits a lower MER during the concept drift.

Overall, CPApproximate is slow to adapt its intervals, which can be explained by the slow update of its nonconformity scores, but as a result maintains tighter intervals. Conversely, CPExact is able to adapt faster, since it updates the predictions for all elements in the calibration set to derive the non-conformity scores. This means that as the learners shift to the new distribution, so will all the non-conformity scores, as opposed to one-at-a-time with CPApproximate. However, this adaptivity comes at the cost of increased interval sizes, and much slower computation (Section 4.6). Mondrian Forest is also slow to adapt to changes in the underlying distribution, thus leading to large areas where its intervals are wrong, the result of which can be seen in Figure 7, where its utility is the lowest among all methods.

OnlineQRF provides a good compromise, both for the artificial and the real-world datasets. Its intervals are quickly adapted to distribution changes, which makes it recover the requested error rates, and also produce tight intervals in a timely manner, therefore making its predictions both valid and efficient.

The situation is similar in terms of quantile loss, shown in Figure 8, with CPApproximate having the lowest median loss, and Mondrian Forest performing the worst among all methods.

4.5 Significance level

Depending on how critical an application is, a user might require different error guarantees. A natural question that arises is how different settings of the significance level affect the methods. We investigate this question by running all the algorithms on the small-scale data with a significance level varying from 0.3 to 0.01, and report the aggregated results, averaged over all datasets, for each method and significance level in Table 3. We provide a more in-depth look with MER/RIS figures similar to Figure 1 in Appendix B. Ideally, we would like the MER to never cross the value α, where α is the significance level, while keeping the RIS low.

Dataset	MF	oQRF	CP_{approx}	CP_{exact}
Small-scale data	41.6	5.4	5.7	102.3
Flight delays	6340	836	899	27863
Friedman	2380	114	213	2011

Table 4: Average running times for all algorithms (seconds).

Table 3 shows that, for larger significance values, Mondrian Forest maintains a valid MER. However, Mondrian Forest fails to meet the MER requirements for significance values below 0.1, which represent critical use cases with little room for errors. OnlineQRF shows a very robust tradeoff between MER and RIS for a range of values of α. It fails to meet the MER requirements only for $\alpha = 0.01$, while maintaining an RIS competitive with MF for all significance levels. Finally, the conformal prediction methods are able to maintain the error guarantee across the range of significance values. The price to pay is increased interval sizes, which with lower significance levels approach, or even exceed, the range of observed label values.

4.6 Computational cost

This section reports the computational cost of each method, which is one of the main considerations when selecting an online learning method. For the running time comparison, note that Mondrian Forest is implemented in Cython using Numpy primitives. Therefore, a perfectly fair comparison with the Java implementations of the algorithms developed for this study is not possible without a full porting, which is outside of the scope of the current work.

However, we mention that for each tree in a Mondrian Forest, the cost to process the n^{th} data point is $\mathcal{O}(\log n)$, and hence growing for each additional data point, so the total cost to process N data points is $\mathcal{O}(\sum_{n=1}^{N} \log n) = \mathcal{O}(\log N!)$. In contrast, the computational cost of the methods developed in this paper, and the underlying FIMT learner, are constant, regardless of the number of data points. They depend only on constant values, such as the number of features for FIMT, sketch accuracy for OnlineQRF, and the size of the calibration set for the conformal prediction methods, and are generally bounded by the number of features p, so the cost to train the n^{th} data point is always $\mathcal{O}(p)$.

Table 4 lists the results; to ease presentation, we average the runtimes of all the different datasets in each group. OnlineQRF is the fastest algorithm for all the data groups, with CPApproximate a close second. We note that both approaches are an order of magnitude faster than Mondrian Forest, confirming the theoretical computational costs of the algorithms. Finally, we can clearly see the drawbacks of CPExact: for every data point, it requires a pass through the entire calibration set to get updated predictions. This incurs a significant computational cost, making it more than an order of magnitude slower than the approximate version of the algorithm.

5. Related Work

Decision trees have attracted a large body of research, both in order to extend them to the online learning setting, and to quantify the uncertainty of their predictions. In this section, we describe work in these two areas that is directly related to this study and refer the interested reader to Kotsiantis (2013) for a general survey of decision tree learning.

Online decision trees and forests

One of the first adaptations of decision tress to the online setting is the Very Fast Decision Tree (VFDT), or Hoeffding Tree (Domingos and Hulten, 2000). Unlike batch decision trees, which recursively partition all examples when growing the tree, VFDT is a single-pass algorithm, where new examples can only affect the leafs of the tree. The statistics of each feature are accumulated at leafs denoted as "learning leafs" which are tested periodically to determine whether they should be split. The Hoeffding bound provides a δ-guarantee on the optimality of the selected split.

While single trees can deal with simpler problems and provide better interpretability, ensemble techniques such as random forests (RF) have been shown to achieve better accuracy for a variety of tasks (Fernández-Delgado et al., 2014). As a result, there have been multiple studies to adapt them to the online learning setting. The bagging part of the RF algorithm was originally adapted to online learning by Oza (2005). Saffari et al. (2009) use a combination of the online bagging algorithm by Oza and propose a tree randomization scheme to adapt random forests to the online setting. Adaptation to concept drift was proposed to be handled by discarding trees that exhibit a high out-of-bag error. Gomes et al. (2017) also leverage the online bagging algorithm by Oza (2005). The authors restrict the number of features that leaves can consider for splitting to create an algorithm that is similar the original random forests by Breiman, but adapted to the online setting through the use of Hoeffding trees as the base learner. The algorithm includes a warning mechanism to deal with concept drift that learns "background" trees, which eventually replace original trees if drift is confirmed for a tree.

Confidence and prediction intervals in tree models

Decision and regression trees can be modified to output not only a single label, i.e., class or regression value, but also some estimate of the *confidence* that the model has in its prediction. Many alternative approaches have been proposed in the batch setting: Mentch and Hooker (2016) use formal statistical inference to produce prediction intervals, Wager et al. (2014) make use of the jackknife to estimate standard errors for random forests, and Chipman et al. (2010) propose Bayesian additive regression trees that estimate a distribution over decision trees which can be used to produce prediction intervals. Johansson et al. (2014b) propose a model for conformal prediction that uses a batch trained decision tree that updates its predictions in an online manner. The structure of the tree however remains static, i.e. no online training is performed.

Lakshminarayanan et al. (2016) use Mondrian processes (Roy and Teh, 2009) to create tree-like online prediction models which have the property of being "extensible", i.e., the distribution of trees in the online version is the same as the batch one. This property can be leveraged to create online random forests that can be updated efficiently and produce

models comparable to their batch version. Mondrian forests are able to predict the full conditional distribution for the predicted variable, and therefore can produce prediction intervals. However, the prediction model is parametric: it models the variable as a hierarchy of Gaussians, one for each node in the tree. The benefit of having access to the complete predictive posterior comes at cost as well: the memory and computational cost for each tree grows for each data point processed, making them unsuitable for unbounded streaming data.

6. Conclusions

In this paper, we proposed two algorithms that quantify the prediction uncertainty of regression forests by producing prediction intervals for unbounded data streams. We have presented an extensive empirical investigation, which shows that our approaches outperform naïve and state-of-the-art methods for a variety of datasets in terms of error rate and interval size, while being an order of magnitude faster to execute.

Comparing the proposed algorithms in terms of practicality, OnlineQRF appears to offer the best trade-off between maintaining the requested error rate and mean interval size, while being computationally efficient. Although the conformal prediction methods are able to better maintain the requested error rate at a confidence level above 95%, they do so at a cost of significantly increased interval size and, in the case of CPExact, an order of magnitude longer running time.

As evidenced by our experiments on the datasets exhibiting concept drift, when the theoretical assumptions of the algorithms are violated, OnlineQRF is able to adapt relatively quickly to the concept drift. CPApproximate is slower to adapt but maintains tight intervals, and while CPExact can maintain the requested error rate, it does so at the cost of the interval's utility. Moreover, its computational cost makes it unsuitable for large scale data. That said, CPExact is the only method that consistently maintains validity in both stationary and concept drift settings, so it is our recommendation in cases where maintaining the error rate is of utmost importance.

This work opens several important research avenues. First, the empirical results presented here can be reinforced by providing validity guarantees for the conformal prediction methods. They can also be extended to incorporate normalization to possibly produce more informative intervals, see e.g., (Boström et al., 2017). Second, we aim to properly address the issues of concept drift at the meta-algorithm level, by introducing mechanisms that inject more uncertainty in the predictions when a concept drift is detected. Finally, we would like to provide more scalable implementations of the methods, for example by developing efficient distributed algorithms for OnlineQRF and the conformal prediction methods.

Appendix A. Data Descriptions

A.1 Small-scale data

In this appendix we provide a brief description of each small-scale dataset used in this study. When possible we have used the description of the data provided by the original authors.

2dplanes: This is an artificial dataset described in Breiman et al. (1984), with variance 1 instead of 2. The 10 attributes are generated independently using:

$$P(X_1 = -1) = P(X_1 = 1) = 1/2$$
$$P(X_m = -1) = P(X_m = 0) = P(X_m = 1) = 1/3, m = 2, ..., 10.$$

We obtain the value of the target variable Y using the rule:

$$\text{if } X_1 = 1 \text{ set } Y = 3 + 3X_2 + 2X_3 + X_4 + \sigma(0, 1)$$
$$\text{if } X_1 = -1 \text{ set } Y = -3 + 3X_5 + 2X_6 + X_7 + \sigma(0, 1)$$

abalone: The task is to predict the age of abalone from physical measurements. Features include sex, dimensions, and weight.

ailerons: This dataset addresses a control problem, namely flying a F16 aircraft. The attributes describe the status of the aeroplane, while the goal is to predict the control action on the ailerons of the aircraft.

bank32nh: A synthetically generated dataset from a simulation of how bank-customers choose their banks. Tasks are based on predicting the fraction of bank customers who leave the bank because of full queues.

calHousing: The task here is to predict the median house value based on features like median income and median age, collected from the 1990 California census.

cpu_act: A collection of computer systems activity measures. The task is to predict the portion of time that CPUs run in user mode, based on attributes such as number of reads/writes to the system, system calls, and page requests.

energy: Appliances energy prediction dataset. The task is to predict the energy consumption of appliances in a home. From the UCI data repository: `https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction`

elevators: This dataset also concerns the task of controlling a F16 aircraft, although the target variable and attributes are different from the ailerons domain. In this case the goal variable is related to an action taken on the elevators of the aircraft.

friedman: This is an artificial dataset used in Friedman (1991). The examples are generated using the following method: Generate the values of 10 attributes, $X_1, ..., X_{10}$ independently, each of which is uniformly distributed over $[0, 1]$. Obtain the value of the target variable Y using the equation:

$$Y = 10 * sin(\pi * X_1 * X_2) + 20 * (X_3 - 0.5)^2 + 10 * X_4 + 5 * X_5 + \sigma(0, 1),$$

where $\sigma(0, 1)$ is a random number generated from a normal distribution with mean 0 and variance 1.

house: Both the `house_8L` and `house_16H` datasets are concerned with predicting the median price of a house in a region based on demographic composition and a state of housing market. The number signifies the approximate difficulty of the task, L for low difficulty, H for high.

kin8nm: This dataset is concerned with the forward kinematics of an 8 link robot arm. Among the existing variants of this dataset we have used the variant 8nm, which is known to be highly non-linear and medium noisy.

mv: This is an artificial dataset with dependencies between the attribute values. It contains three nominal and seven continuous features, which contain dependencies of the form `IF` $(X_5 < 0.5)$ `THEN` $X_8 =$ `normal` `ELSE` $X_8 =$ `large`, where $X_5 \sim U(-1, 1)$. The full interactions can be found on the dataset's website [5].

newsPopularity: This dataset summarizes a heterogeneous set of features about articles published on Mashable[6] in a period of two years. The goal is to predict the number of shares in social networks (popularity) based on attributes such as the number of words, images, and videos in the article, publication time and other word-based derived features.

puma: This is a family of datasets synthetically generated from a realistic simulation of the dynamics of a Unimation Puma 560 robot arm. There are eight datasets in this family. The task is to predict the angular acceleration of one of the robot arm's links. The inputs include angular positions, velocities and torques of the robot arm. The number in the name of the dataset indicates the number of features, 8 or 32.

qsar-chembl: This dataset contains QSAR data (from ChEMBL[7] version 17) showing activity values (unit is pseudo-pCI50) of several compounds on drug target ChEMBL_ID.

sulfur: These are measurements for the amount of sulfur recovered by recovery units that remove environmental pollutants from acid gas streams in industrial settings before they are released into the atmosphere. The features are gas and air flows.

yprop_4_1: A drug design dataset, used in Feng et al. (2003). The task is to predict the toxic effects of a substance based on a number of chemical descriptors.

A.2 Friedman Functions

In this section we provide a description for all the Friedman function concept drift datasets used in this study.

We follow the concept drift types and equations used by Ikonomovska et al. (2011) for the first Friedman function, and apply similar transformations for the second and third functions.

Friedman #1

The original Friedman #1 function is the one used for our small-scale data: it generates the values of 10 attributes, $X_1, ..., X_{10}$ independently, each of which is uniformly distributed over $[0, 1]$, out of which only the five first affect the dependent. We obtain the value of the target variable Y using the equation:

$$Y = 10 * sin(\pi * X_1 * X_2) + 20 * (X_3 - 0.5)^2 + 10 * X_4 + 5 * X_5 + \sigma(0, 1),$$

where $\sigma(0, 1)$ is again drawn from a standard normal distribution.

The concept drift functions are the same as in (Ikonomovska et al., 2011):

Local expanding abrupt drift: At the first change point (250k points) two regions of the input space are introduced, each of which has a different generating function than the original. The first region is $R1 \equiv X_2 < 0.3$ & $X_4 > 0.7$ & $X_5 < 0.3$. The new generating function for

5. http://www.dcc.fc.up.pt/ ltorgo/Regression/DataSets.html (MV Artificial Domain)
6. https://www.mashable.com
7. https://www.ebi.ac.uk/chembl/index.php/

this region is:

$$y_{1,R1} = 10 * X_1 * X_2 + 20 * (X_3 - 0.5) + 10 * X_4 + 5 * X_5 + \sigma(0, 1)$$

The second region is $R2 \equiv X_2 > 0.7$ & $X_3 > 0.7$ & $X_4 < 0.3$ & $X_5 > 0.7$. The new generating function for this region is:

$$y_{1,R2} = 10 * cos(X_1 * X_2) + 20 * (X_3 - 0.5) + e^{X_4} + 5 * {X_5}^2 + \sigma(0, 1).$$

At the second point of change at 500k points the regions are expanded by removing the last inequality ($x_5 < 0.3$ and $x_5 > 0.7$, respectively). At the third change point (750k instances) we again remove the last inequality ($x_4 > 0.7$ and $x_4 < 0.3$, respectively).

Global reoccurring concept drift: At the first change (500k) point we permute the positions of independent variables so the generating function becomes:

$$y_2 = 10 * sin(\pi * X_4 * X_5) + 20 * (X_2 - 0.5)^2 + 10 * X_1 + 5 * X_3 + \sigma(0, 1).$$

At the second change point (750k) the original function reoccurs.

Global slow gradual drift: In this drift we gradually introduce instances generated from a different concept after the change point, with a linearly increasing probability, until 100k instances later the new concept takes over. At the first change point (500k) we introduce points from the function:

$$y_3 = 10 * sin(\pi * X_4 * X_5) + 20 * (X_2 - 0.5)^2 + 10 * X_1 + 5 * X_3 + \sigma(0, 1).$$

At the second point of change (750k) we gradually introduce instances from the second concept, which is a permutation of the previous concept:

$$y_4 = 10 * sin(\pi * X_2 * X_5) + 20 * (X_4 - 0.5)^2 + 10 * X_3 + 5 * X_1 + \sigma(0, 1).$$

Friedman #2 and #3

The second and third Friedman functions model an alternating current series circuit involving a resistor R, inductor L and capacitor C. A generator places voltage on the circuit with an angular frequency $\omega = 2\pi f$, where f the cyclic frequency.

Then the impedance, Z, and phase shift, ϕ, both depend on the components of the circuit:

$$Z(R, \omega, L, C) = [R^2 + (\omega L - 1/\omega C)^2]^{1/2}$$
$$\phi(R, \omega, L, C) = \tan^{-1}\left[\frac{\omega L - 1/\omega C}{R}\right].$$

We add an error factor to the values, drawn from a standard normal.

The independent variables are uniformly distributed and lie in the range:

$$0 \leq R \leq 100 \text{ ohms},$$
$$20 \leq f \leq 280 \text{ hertz},$$
$$0 \leq L \leq 1 \text{ henries},$$
$$1 \leq C \leq 11 \text{ microfarads}.$$

Similarly to the above we introduce artificial concept drift to each function:

Local expanding abrupt drift: For the drift regions we use the same criteria as for the first Friedman function, only on normalized values of the independent variables, so that they again belong in the $[0, 1]$ range. We again permute the positions of the independent variables when the concept changes. For the impedance Z we use the functions:

$$Z_{1,R1}(R, \omega, L, C) = [C^2 + (\omega R - 1/\omega L)^2]^{1/2}$$
$$Z_{1,R2}(R, \omega, L, C) = [R^2 + (RC - 1/RL)^2]^{1/2},$$

for regions $R1$ and $R2$.

For the phase shift we use the functions:

$$\phi_{1,R1}(R, \omega, L, C) = \tan^{-1}\left[\frac{RC - 1/\omega L}{\omega}\right]$$
$$\phi_{1,R2}(R, \omega, L, C) = \tan^{-1}\left[\frac{\omega C - 1/\omega R}{R}\right],$$

for regions $R1$ and $R2$.

Global abrupt reoccurring: The new concepts that are introduced between 500k and 750k instances for Z and ϕ are:

$$Z_2(R, \omega, L, C) = [C^2 + (\omega E - 1/\omega L)^2]^{1/2} \tag{3}$$
$$\phi_2(R, \omega, L, C) = \tan^{-1}\left[\frac{RC - 1/\omega L}{R}\right]. \tag{4}$$

As before, we use the original functions for instances 0-250k and 750k-1M.

Global slow gradual drift: The concepts introduced gradually at the first change point are the same used above for the global abrupt reoccurring, i.e. Equation 3 for Z and Equation 4 for ϕ.

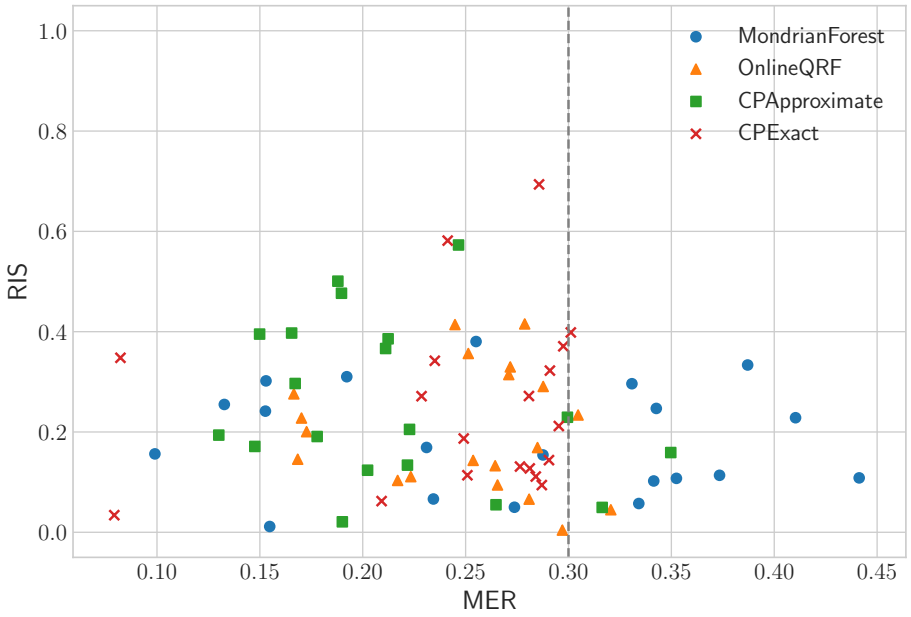
At the second change point we introduce:

$$Z_4(R, \omega, L, C) = [L^2 + (\omega C - 1/\omega R)^2]^{1/2}$$
$$\phi_4(R, \omega, L, C) = \tan^{-1}\left[\frac{\omega C - 1/\omega R}{R}\right].$$

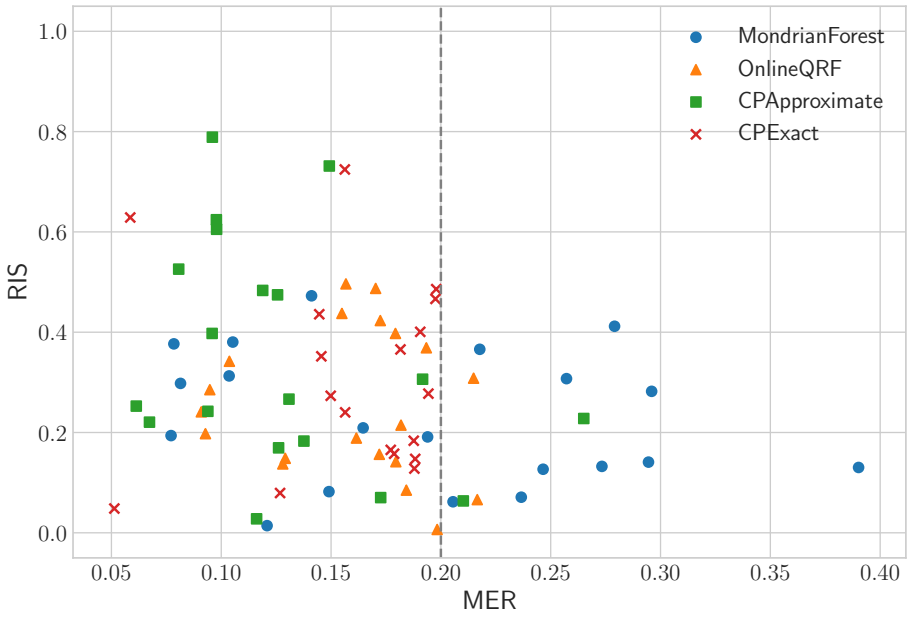
Appendix B. Significance Level Figures

In this appendix we provide combined MER-RIS figures for the confidence experiments of Section 4.5, which are shown aggregated in Table 3. The results for $\alpha = 0.1$ (90% confidence) can be seen in Figure 1 so we omit it here.

These figures clearly demonstrate the inability of Mondrian Forest to maintain the requested error, especially for a confidence of 95% and 99%, as shown in Figure 10. We also note the ability of CPExact to maintain the requested error rate across all experiments.

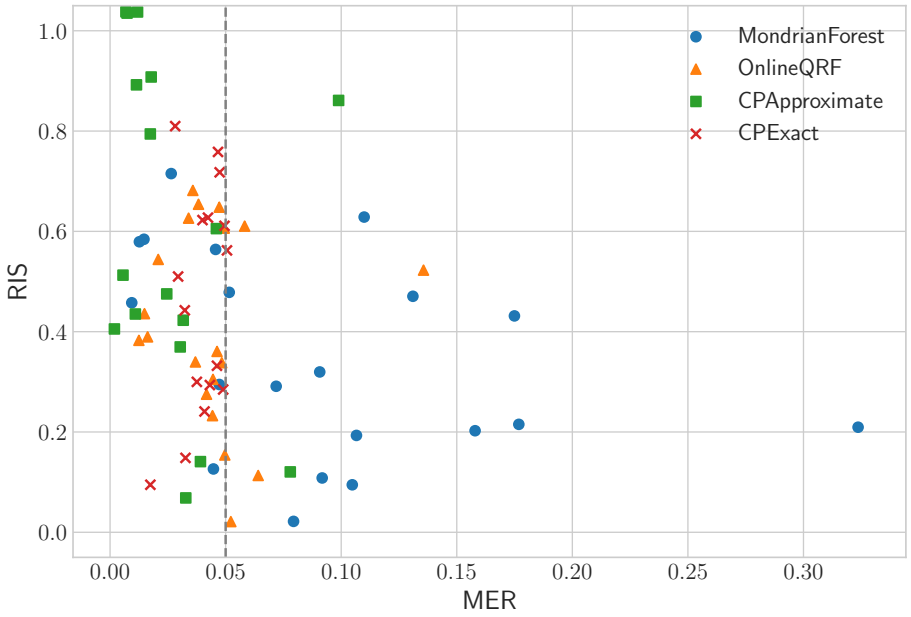


(a) Significance 0.3 (70% confidence).

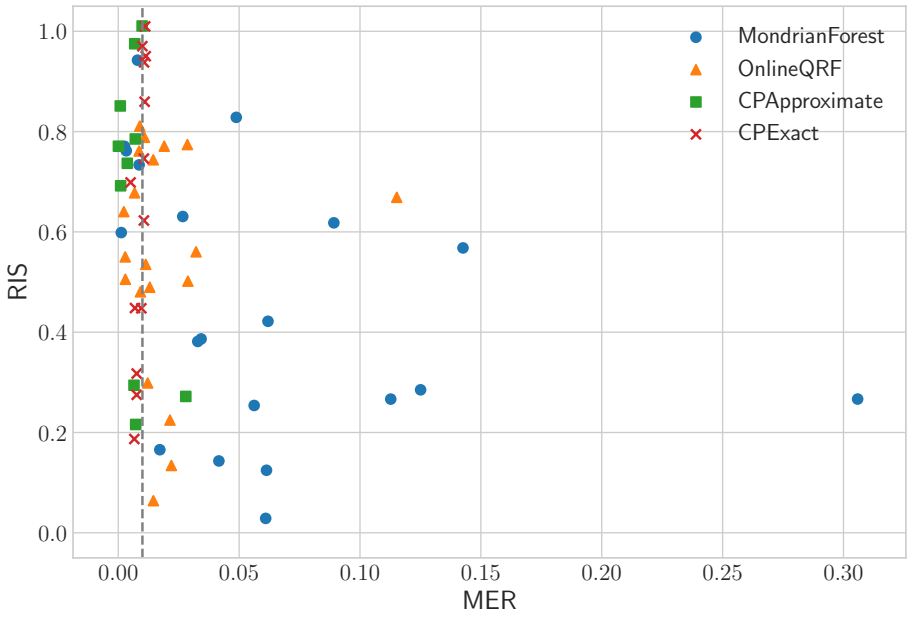


(b) Significance 0.2 (80% confidence).

Figure 9: MER - RIS plots for significance 0.3 (top) and 0.2 (bottom).



(a) Significance 0.05 (95% confidence).



(b) Significance 0.01 (99% confidence).

Figure 10: MER - RIS plots for significance 0.05 (top) and 0.01 (bottom).

References

Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '12, pages 23–34, 2012.

Yael Ben-Haim and Elad Tom-Tov. A streaming parallel decision tree algorithm. *Journal of Machine Learning Research*, 11:849–872, 2010.

Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: Massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010a.

Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 135–150, 2010b.

Henrik Boström, Henrik Linusson, Tuve Löfström, and Ulf Johansson. Accelerating difficulty estimation for conformal regression forests. *Annals of Mathematics and Artificial Intelligence*, 81(1-2):125–144, 2017.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and Regression Trees*. CRC press, 1984.

Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. BART: Bayesian Additive Regression Trees. *The Annals of Applied Statistics*, 4:266–298, 2010.

Gianmarco De Francisci Morales and Albert Bifet. SAMOA: Scalable Advanced Massive Online Analysis. *Journal of Machine Learning Research*, 16:149–153, 2015.

Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000.

Jun Feng, Laura Lurati, Haojun Ouyang, Tracy Robinson, Yuanyuan Wang, Shenglan Yuan, and S. Stanley Young. Predictive Toxicology: Benchmarking Molecular Descriptors and Statistical Methods. *Journal of Chemical Information and Computer Sciences*, 43: 1463–1470, 2003.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014.

Jerome H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19: 1–67, 1991.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 329–338, 2009.

Joao Gama. Evolving Data, Evolving Models in Economy and Finance. In *MIDAS '17: Second Workshop on Mining Data for Financial Applications*, 2017. Keynote.

Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106:1469–1495, 2017.

Michael B. Greenwald and Sanjeev Khanna. *Quantiles and Equi-depth Histograms over Streams*, pages 45–86. Springer, 2016.

Elena Ikonomovska, João Gama, and Sašo Džeroski. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, 23:128–168, 2011.

Ulf Johansson, Henrik Boström, Tuve Löfström, and Henrik Linusson. Regression conformal prediction with random forests. *Machine Learning*, 97:155–176, 2014a.

Ulf Johansson, Cecilia Sönströd, Henrik Linusson, and Henrik Boström. Regression trees for streaming data with local performance guarantees. In *2014 IEEE International Conference on Big Data*, pages 461–470, 2014b.

Zohar Karnin, Kevin Lang, and Edo Liberty. Optimal quantile approximation in streams. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 71–78, Oct 2016.

Roger Koenker. *Quantile Regression*. Cambridge University Press, 1996.

Sotiris Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4): 261–283, 2013.

Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Mondrian Forests for Large-Scale Regression when Uncertainty Matters. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pages 1478–1487, 2016.

Rowan McAllister, Yarin Gal, Alex Kendall, Mark van der Wilk, Amar Shah, Roberto Cipolla, and Adrian Weller. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4745–4753, 2017.

Nicolai Meinshausen. Quantile Regression Forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

Lucas Mentch and Giles Hooker. Quantifying Uncertainty in Random Forests via Confidence Intervals and Hypothesis Tests. *Journal of Machine Learning Research*, 17:1–41, 2016.

Nikunj C. Oza. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345, 2005.

Harris Papadopoulos and Haris Haralambous. Reliable prediction intervals with regression neural networks. *Neural Networks*, 24(8):842 – 851, 2011.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Proceedings of the 13th European Conference in Machine Learning*, pages 345–356, 2002.

Harris Papadopoulos, Vladimir Vovk, and Alex Gammerman. Regression conformal prediction with nearest neighbours. *Journal of Artificial Intelligence Research*, 40(1):815–840, 2011.

Binoy Ravindran, E. Douglas Jensen, and Peng Li. On recent advances in time/utility function real-time scheduling and resource management. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, pages 55–60, 2005.

Daniel M Roy and Yee W. Teh. The Mondrian Process. In *Advances in Neural Information Processing Systems 21*, pages 1377–1384, 2009.

Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1393–1400, 2009.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15:49–60, 2013.

Vladimir Vovk. On-line confidence machines are well-calibrated. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 187–196, 2002.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

Stefan Wager, Trevor Hastie, and Bradley Efron. Confidence Intervals for Random Forests: The Jackknife and the Infinitesimal Jackknife. *Journal of Machine Learning Research*, 15: 1625–1651, 2014.

Lu Wang, Ge Luo, Ke Yi, and Graham Cormode. Quantiles over data streams: An experimental study. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 737–748, 2013.