Table 5: MER and RIS for different significance levels. The target MER is α where α the significance level.

Method	Metric	α				
		0.3	0.2	0.1	0.05	0.01
MF	MER	0.28	0.20	0.13	0.09	0.062
	RIS	0.19	0.23	0.29	0.35	0.460
oQRF	MER	0.23	0.14	0.07	0.04	0.015
	RIS	0.20	0.23	0.31	0.35	0.510
CP_{approx}	MER	0.22	0.13	0.06	0.03	0.006
	RIS	0.21	0.31	0.48	0.79	1.130
CP_{exact}	MER	0.25	0.16	0.08	0.04	0.009
	RIS	0.28	0.37	0.50	0.62	0.940

4.6 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 experiment by running all the algorithms on the small-scale data with a significance level varying from 0.3 to 0.01, and report the results for each method and significance level in Table 5. Ideally, we would like the MER to match the value α, where α is the significance level, while keeping the RIS low.

We report the mean MER over the datasets, while for the RIS we report the median, to filter out the outliers due to the numerical errors in the `electricity_prices`, `elevators`, and `newsPopularity` datasets, as evident in Table 4. These outliers are caused by single windows, and disproportionately affect the mean measurements.

Table 5 shows that, for higher significance values, Mondrian Forest tends to be closer to the desired MER. However, Mondrian Forest fails to meet the MER requirements for significance values below 0.2 when we allow little room for errors. The conformal prediction methods are able to maintain the error guarantee across the range of significance values. The cost to pay are increased interval sizes, which approach or even exceed the range of observed label values for $\alpha = 0.01$ (99% confidence). Finally, 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 $\alpha = 0.01$, although not by a large margin, while maintaining an RIS competitive with MF for all significance levels.

4.7 Computational cost

This section reports the computational cost of each method, which is one of the main considerations in 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 paper is not possible without a full porting, which is outside of the scope of the current work.

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

Table 6: Running times for all the algorithms, in seconds.

Dataset	MF	oQRF	CP_{approx}	CP_{exact}
Small scale data	41.6	11.5	13.2	117.1
Airlines 700k	1773	744	463	7925
Airlines 2M	5050	2813	2461	24879
Airlines 5M	12197	7724	6851	110842

Table 6 clearly shows the main drawback of CPExact: the need to constantly update the complete calibration set before each prediction makes the algorithm an order of magnitude slower to execute than CPApproximate and OnlineQRF. OnlineQRF is the fastest algorithm for the small-scale data, but as we increase the size of the dataset CPApproximate becomes the fastest algorithm. As mentioned in Section 3.2, there is a need to update up to T histograms for each training example, and the prediction step requires merging T histograms, where T is the number of learners. Although both these operations are constant time, they introduce additional overhead on top of the underlying learner, which CPApproximate does not require. Hence their runtimes are similar, but with CPApproximate edging out OnlineQRF.

Although our implementations use parallelism for training and prediction, the benefit is somewhat limited by the single-threaded part of MOA which is reading in the data. As an example of the speed gain, we observed only a 30% reduction in total runtime using four threads instead of one for learning and prediction.

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.

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 [5]. 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 [6]. As a result, there have been multiple studies to adapt them to the online learning setting. The bagging part of the RF algorithm was adapted to online learning by Oza [15]. Saffari et al. [17] use a combination of the online bagging algorithm by Oza and propose a tree randomization scheme to adapt random forests to