Conformal Regression with Reject Option

Ulf Johansson ULF.JOHANSSON@JU.SE
Cecilia Sönströd CECILIA.SONSTROD@JU.SE
Dept. of Computing, Jönköping University, Sweden

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

Editor: Simone Vantini, Matteo Fontana, Aldo Solari, Henrik Boström and Lars Carlsson

Abstract

A regressor with reject option may refrain from making predictions expected to be inaccurate. In this paper, we introduce and evaluate conformal regression with reject option. Consistent with standard conformal regression, non-rejected predictions are valid prediction intervals. The suggested approach utilizes Mondrian conformal regression, where the categories are dynamically created from difficulty estimations of individual instances and requested rejection levels. As shown in the experiments, using 16 publicly available data sets and random forests as underlying models, the conformal regressors produced progressively tighter intervals for higher rejection levels, thus demonstrating the trade-off between coverage and informativeness targeted when adding a reject option. A key property of the novel method is the fact that the informativeness, i.e., the interval sizes, resulting from any combination of significance and rejection levels is known to the user before making any test predictions. While all four different difficulty estimators evaluated led to consistently tighter intervals for higher rejection levels, the one producing the most efficient conformal regressors utilized the disagreement between the trees in the Random forest.

Keywords: Conformal prediction, Regression, Regression with reject option

1. Introduction.

Many AI systems for decision support in high-stakes domains, such as medicine or finance, include a human-in-the-loop, where a predictive model is used for easy instances, whilst more difficult instances are referred to a human expert. To achieve a trustworthy and efficient AI solution in these situations, it is essential to accurately determine which instances to predict using the model, and which to refer to the human. In addition, it is desirable to know the human workload in advance. The *prediction with reject option* framework, developed by (Chow, 1957, 1970), introduces the option for a model to refrain from making predictions for instances where uncertainty is high. The underlying assumption is that the error on predictions made should decrease as the proportion of rejected instances increases, so that predictive quality can be increased by decreasing coverage, i.e. the proportion of instances for which predictions are delivered. The prediction with reject option framework thus offers a methodological framework for formalizing the trade-off between predictive performance and coverage, allowing the design of human-in-the-loop systems with shared workload.

In this paper we introduce and present conformal regression with reject option. Since the models are conformal regression models, the non-rejected predictions come in the form of intervals with validity guarantees. With the suggested approach, it becomes possible for a user to, directly after the calibration step, assess and compare the resulting test prediction interval sizes from any combination of rejection and significance levels, thus controlling the trade-off between human workload, model informativeness and error rate.

2. Background.

2.1. Regression with reject option

In predictive regression, the task is to estimate a function $f(\boldsymbol{X}) \to \mathbb{R}$, mapping each input vector $\boldsymbol{x_i}$ to a real-valued target value y_i. Many different machine learning techniques, such as decision trees, random forests, support vector machines, and neural networks, handle regression tasks by building a model, hereafter called a regressor, to represent the function f, capable of producing a predicted value $\hat{y}$ when provided with an input vector $\boldsymbol{x_i}$.

Regressors with a reject option may refrain from making predictions that are likely to be inaccurate, thus increasing the trustworthiness of the model. The choice of whether to accept or reject a prediction is made on the basis of the estimated uncertainty of the prediction, defined by an uncertainty function $g(\boldsymbol{x_i})$ and a threshold τ (Shah et al., 2022). Formally, a regressor with reject option includes an additional output:

$$m(\boldsymbol{x_i}) = \begin{cases} \circledR & \text{if } g(\boldsymbol{x_i}) \geq \tau \\ h(\boldsymbol{x_i}) & \text{if } g(\boldsymbol{x_i}) < \tau. \end{cases} \tag{1}$$

where $m(\boldsymbol{x_i})$ is the output from the regressor with reject option and $h(\boldsymbol{x_i})$ is the prediction from the underlying model. The threshold τ is used to control the trade-off between error, e.g. measured as MAE, and prediction coverage.

2.2. Conformal regression

Conformal prediction (Vovk et al., 2005) produces prediction regions and all conformal predictors are *valid*, i.e., given a significance level $\epsilon \in [0, 1]$, the error rate of a conformal predictor will, in the long run, be exactly ϵ. Conformal regressors output prediction intervals, and an error is committed when the target value is outside the interval.

Conformal prediction was introduced in a transductive setting, but this and most other recent studies use *inductive* or *split* conformal prediction. Inductive conformal prediction (ICP) can be applied on top of any predictive model (called the *underlying model*), to create a conformal predictor. ICP requires a labeled data set (the *calibration set*) that was not used for training the underlying model. Formally, the validity guarantees of ICP rely on only one assumption; that the calibration and test sets are *exchangeable*, which is slightly weaker than i.i.d.

While all conformal regressors are guaranteed to be valid, the informativeness varies. Specifically, we want the prediction intervals to be as tight as possible, a property which in conformal prediction is referred to as *efficiency*. But, in most applications, we also want the predictions to be *specific* (or *sharp*), i.e., for conformal regressors, the interval sizes should differ between instances. Using basic ICP, however, all prediction intervals produced by a conformal regressor will have the same size.

The standard way of making conformal regressors sharp is to apply a procedure called *normalization*. Here, the key idea is to estimate the difficulty of every instance, and adjust the interval sizes accordingly. As confirmed in many previous studies, e.g., (Papadopoulos et al., 2002; Papadopoulos and Haralambous, 2010, 2011; Johansson et al., 2014; Boström et al., 2017) normalization will lead to tighter prediction intervals on average, i.e., normalized conformal regressors are generally more efficient. In addition, since instances deemed to be easier will have smaller intervals than harder instances, the model also provides additional information on a per-instance basis, thus making it more informative.

It should be noted that individual intervals from normalized conformal regression can become very large. This is in sharp contrast to the basic version, where the intervals cannot be larger than twice the biggest absolute error in the calibration set, see (Boström and Johansson, 2020).

An alternative to using normalization is to instead employ *Mondrian* conformal regression. In the Mondrian setup, instances are partitioned into categories, and then ICP is applied to each category separately. This means that every category requires its own calibration set, but also that the validity guarantees apply to each category independently. In principle, normalization could be applied to each category in a Mondrian setup, but this is normally not the case. However, even without normalization, the Mondrian approach makes the conformal regressor more sharp since the interval sizes are identical only within each category, but differ between categories.

A key part of conformal prediction is the so-called *nonconformity functions* which are real-valued functions measuring the strangeness of instances in the form of attribute-target pairs ($z_i = \boldsymbol{x_i}, y_i$). While all nonconformity functions will produce valid conformal predictors, efficiency is highly affected by both the underlying model and the nonconformity function. For basic conformal regressors, the nonconformity of an instance $(\boldsymbol{x}_i, y_i)$ is the absolute error

$$A\left(\boldsymbol{x}_i, y_i, h\right) = \left|y_i - h\left(\boldsymbol{x}_i\right)\right|, \tag{2}$$

where h is the underlying predictive model providing real-valued point predictions. Formally, an inductive conformal regressor is generated as follows:

1. Divide the training data Z_{tr} into a proper training set Z_t and a calibration set Z_c.
2. Train the underlying model h on Z_t.
3. Apply the nonconformity function, e.g., Eq. 2, to the calibration instances in Z_c, obtaining a list of calibration scores $S = \alpha_1, ..., \alpha_q$ where $q = |Z_c|$ and S is sorted in descending order.

A prediction interval containing the true target value for a test instance $\boldsymbol{x}_{l+1}$, with the probability $1 - \epsilon$ and using the absolute error in Eq. 2 as the nonconformity function, is constructed as follows:

1. Obtain a point prediction $h(\boldsymbol{x}_{l+1})$.
2. Find the calibration score α_p where $p = \lfloor \epsilon(q + 1) \rfloor$.
3. Let $s = \lfloor \epsilon(q + 1) \rfloor$. This is the index of the $(1 - \epsilon)$-percentile nonconformity score, α_s.

4. Let the prediction interval for $\boldsymbol{x}_{l+1}$ be $\hat{y}_{l+1}^{\epsilon} = h\left(\boldsymbol{x}_{l+1}\right) \pm \alpha_p$

Using this procedure, prediction interval sizes will be identical, i.e., $2\alpha_p$, for all test instances. In practice, every instance is assumed to be equally hard to predict. When adding normalization, the nonconformity of an instance is instead defined as

$$A\left(\boldsymbol{x}_i, y_i, h\right) = \frac{\left|y_i - h\left(\boldsymbol{x}_i\right)\right|}{\sigma_i + \beta}, \tag{3}$$

where σ_i is a difficulty estimation of the instance $\boldsymbol{x}_i$, and β is a sensitivity parameter. Using a normalized nonconformity function the prediction intervals become:

$$\hat{y}_{l+1}^{\epsilon} = h\left(\boldsymbol{x}_{l+1}\right) \pm \alpha_p\left(\sigma_{l+1} + \beta\right) \tag{4}$$

A Mondrian inductive conformal regressor without normalization and using the absolute error as nonconformity function is constructed as follows:

1. Divide the training data Z_{tr} into two disjoint subsets: the proper training set Z_t and the calibration set $Z_c = \{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_q}, y_q))\}$.
2. Train the underlying model h using Z_t.
3. Partition Z_c into k subsets $Z_{c_1}, \ldots, Z_{c_k}$, following a Mondrian taxonomy κ with categories $\kappa_1, \ldots, \kappa_k$
4. Apply the nonconformity function Eq. 2 to the instances in Z_{c_i}, for each $i = 1, \ldots, k$, producing a list of calibration scores $S_i = \alpha_1, ..., \alpha_{q_i}$ where $q_i = |Z_{c_i}|$ and S_i is sorted in descending order.

A valid prediction interval for a test instance $\boldsymbol{x}_{l+1}$ at the significance level ϵ is obtained from a Mondrian conformal regressor by following these steps:

1. Make a point prediction $h(\boldsymbol{x}_{l+1})$.
2. Find the category $\kappa_i \in \{\kappa_1, \ldots, \kappa_k\}$ for $\boldsymbol{x}_{l+1}$
3. Find the calibration score α_{i_p} where $p = \lfloor \epsilon(q_i + 1) \rfloor$.
4. Let the prediction interval for $\boldsymbol{x}_{l+1}$ be $\hat{y}_{l+1}^{\epsilon} = h\left(\boldsymbol{x}_{l+1}\right) \pm \alpha_{i_p}$

When using a Mondrian approach, each category, as described above, needs its own calibration set. For smaller data sets, this can be a problem, leading to fewer training instances or tiny calibration sets. One solution, applicable to Random forests and originally suggested by Johansson et al. (2014), is to perform the calibration on the out-of-bag instances. This approach, which is also employed in this study, makes it possible to use all instances for both model training and calibration, i.e., $q = |Z_c| = |Z_{tr}|$. It should be noted that this procedure does not come with the theoretical validity guarantees of inductive conformal regression, simply because calibration and test instances are not treated in an identical way, see (Boström et al., 2017). Still, many previous studies, e.g., (Johansson et al., 2014, 2015; Boström et al., 2017; Linusson et al., 2020) have shown excellent empirical validity. If anything, the conformal regressors were slightly conservative.

2.3. Related Work

Classification with reject option has been studied extensively, since being introduced by (Chow, 1957, 1970), see e.g. (Hanczar and Dougherty, 2008; Li and Sethi, 2006; Geifman and El-Yaniv, 2017). The use of a reject option in combination with conformal prediction has previously been studied for classification, see e.g. (Johansson et al., 2023a,b).

Regression with reject option has only been studied in recent years, and in a few papers. Several authors (Zaoui et al., 2020; Shah et al., 2022) note that adapting classification with reject option to a regression setting relies on formulating suitable uncertainty measures for regressors, to define the rejection function. Two notable approaches using deep neural networks are (Geifman and El-Yaniv, 2019) and (Jiang et al., 2020). Geifman and El-Yaniv (2019) proposed a neural network architecture, optimizing both the prediction and rejection functions in one deep neural network, whereas Jiang et al. (2020) used deep neural networks as regressors and uncertainty estimation functions that depend on this architecture on two regression problems with image data as inputs.

In (Zaoui et al., 2020), a model-agnostic framework for regression with reject option, using a conditional variance function to control rejection rate, was introduced. In (Shah et al., 2022), regression with reject option was employed with subgroup fairness criteria, which requires that error not only decreases overall when decreasing coverage, but does so monotonically for a set of defined subgroups of instances.

In a recent preprint, Sokol et al. (2024) studied regression with reject option, using conformalized quantile regression with interval size as the difficulty estimation of an instance. Here, the conformal prediction was used as a tool for rejecting instances with high uncertainty, in contrast to the suggested approach in our paper, which equips a standard conformal regressor with a reject option.

3. Method.

As described in the Introduction, the overall purpose of this paper is to introduce and evaluate conformal regression with reject option. Specifically, the method should have the following properties:

1. A test prediction should be either a prediction interval or ® (reject).
2. The non-rejected predictions should be valid in the standard conformal sense, i.e., the error rate of the predicted intervals should be ϵ.
3. The efficiency of the non-rejected intervals should increase, i.e., the intervals should be tighter, when the regressor is allowed to reject more instances.
4. It should be possible for a user to know the interval sizes for any combination of significance and rejection levels after the calibration step, i.e., before making the first test prediction.

The way to achieve this, which is a key contribution of this paper, is to employ Mondrian conformal regression, where the categories are determined from a difficulty estimator. For a chosen rejection level ρ, e.g., 0.1, the Mondrian taxonomy dictates that all test instances with a higher difficulty than the calibration instance corresponding to the $(1-\rho)$-percentile difficulty estimate in the calibration set should belong to Category Ⓡ and the remaining to Category P. Obviously, all test instances belonging to Category Ⓡ are rejected, while the ones belonging to Category P are predicted. The prediction intervals for the predicted instances are generated using standard ICP, i.e., without normalization. Using this setup, the interval sizes resulting from any combination of significance (ϵ) and rejection (ρ) levels can be found from absolute errors and difficulty estimates of the calibration instances, see Algorithm 1.

Algorithm 1 Conformal Regression with Reject Option - calibration step

```
Input: Underlying regressor $h$, Difficulty estimations of calibration instances $\sigma_1, \cdots, \sigma_q$,
       Significance level $\epsilon$, Rejection level $\rho$
Output: Difficulty threshold for rejection: $\sigma_\rho$, Interval size for non-rejected instances: $\alpha_P$
$\sigma_\rho \leftarrow \lfloor \rho(q+1) \rfloor$
$Z_P \leftarrow \{z_j \in Z_c : \sigma_j \leq \sigma_\rho\}$
$q_P \leftarrow |Z_P|$
$S \leftarrow \{|y_i - h(\boldsymbol{x}_i)| : (x_i, y_i) \in Z_P\}$
Sort $S$ in descending order
$p \leftarrow \lfloor \epsilon(q_P+1) \rfloor$
$\alpha_P \leftarrow S_p$
```

When performing the actual prediction, see Algorithm 2, the difficulty estimate of the test instance σ_{k+1} is compared to the difficulty threshold for rejection. If the difficulty is higher, the prediction is rejected; if not, the prediction interval is $h(x_{l+1}) \pm \alpha_P$.

Algorithm 2 Conformal Regression with Reject Option - prediction step

```
Input: Test instance $x_{l+1}$, Underlying regressor $h$, Difficulty estimator $\delta$, Difficulty thresh-
       old for rejection $\sigma_\rho$, Interval size for non-rejected instances $\alpha_P$
Output: Prediction: either a prediction interval or Ⓡ (reject)
$\sigma_{l+1} \leftarrow \delta(x_{l+1})$
if $\sigma_{l+1} > \sigma_\rho$ then
    return Ⓡ
else
    return $h(x_{l+1}) \pm \alpha_P$
end
```

The regressors used were random forests (Breiman, 2001), as implemented in scikit learn. Parameters were left at default values, except using 300 trees in the forests. For the conformal regressors, the Crepes package (Boström, 2022) was used, and out-of-bag calibration was employed. Regarding difficulty estimators, four options, all readily available in Crepes, were evaluated:

- **kNN distance**: The average distance to the 25 nearest neighbors. The motivation is that less populated parts of feature space should be harder.

- **kNN variance**: The standard deviation of the target values of the 25 nearest neighbors. We expect instances where the target values of the neighbors vary more to be harder.
- **kNN residuals**: The mean of the absolute residuals of the 25 nearest neighbors. Instances where the model is less accurate on their neighbors are expected to be harder.
- **Tree variance**: The variance of the individual tree predictions. Instances are estimated to be harder the more the trees disagree.

The testing protocol employed was standard 10-fold cross-validation. In the experiments, 16 publicly available medium-sized data sets ranging from approximately 4200 to 9500 instances were used. All data sets are from the UCI (Bache and Lichman, 2013), Delve (Rasmussen et al., 1996) or KEEL (Alcalá-Fdez et al., 2011) repositories. The data sets are listed in Table 1 below, where *#inst.* is the number of instances and *#attrib.* is the number of features.

Table 1: Data sets

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

4. Results.

We first present detailed results for the four kin data sets. These data sets are all variations on the same model; a realistic simulation of the forward dynamics of an eight link all-revolute robot arm. The task is to predict the distance of the end-effector from a target. The inputs include joint positions, twist angles, etc. The four data sets differ in two ways: (i) they are either "fairly linear" (identified by the letter f in the data set name) or "non-linear" (letter n in data set name) and (ii) the noise level is either "medium unpredictability/noise" (letter m in data set name) or "high unpredictability/noise" (letter h in data set name).

We start with the kin8fh data set, i.e., fairly linear but high noise level. Here, each part of Figure 1 shows the results for one difficulty estimator, with the top part giving mean interval sizes and the bottom part presenting empirical error rates and rejection rates for the different rejection and significance levels. Looking first at the interval sizes, we want these to decrease for higher rejection rates. For this data set, we do indeed observe this pattern for all difficulty estimators and significance levels, possibly with the exception of kNN distance. When it comes to the empirical error rates, these are most often very close to the significance levels. As expected, there are some fluctuations, especially for the higher rejection rates. The reason is of course the smaller calibration sets, that should result in slightly conservative conformal regressors, but also the fact that relatively few instances are predicted. Comparing the empirical rejection rates to the rejection levels, most differences are very small, often not even noticeable in these plots. This is of course reassuring, telling us that the out-of-bag calibration procedure, using these difficulty estimators, works very well in practice.

Figure 2 presents the same results, but focusing on providing an outright comparison between the four estimators. Here, it is obvious that kNN distance is the worst option, leading to not only larger intervals, but also smaller differences between the rejection levels. The other three options, on the other hand, produce tighter and tighter intervals as more instances are rejected. For this particular data set, there are only small differences in interval sizes between these three difficulty estimators for all rejection and significance levels. The lower plots again show that all conformal regressors are well-calibrated, and that they obtain an almost perfect match between the required rejection level and the actual rejection rate.

Next, we look at the results for the kin8fm (fairly linear medium noise) data set. Comparing Figure 3 to Figure 1, the interval sizes are, as expected since the data set is easier, tighter for all difficulty estimators and significance levels. In addition, all difficulty estimators are able to produce smaller intervals when the rejection level goes up. Regarding empirical error and rejection rates, they are almost perfect, again with the exception of slightly conservative results for the highest rejection levels and $\epsilon = 0.01$. Interestingly enough, as seen in Figure 4, the best estimator for this data set is clearly kNN residuals, despite also being the most conservative, at all but the highest rejection levels for $\epsilon = 0.01$. Again, kNN distance is the worst, while Tree variance and kNN variance obtain very similar results.

The third data set is kin8nm, i.e., non linear but moderate noise. The overview in Figure 5 shows that all difficulty estimators produce tighter intervals for higher rejection levels. As for the other data sets, both error rates and rejection rates are close to the required. The larger intervals compared to kin8fh indicate that the nonlinearity affects the difficulty of the problem more than the added noise. Figure 6 shows the comparison between the estimators. For this data set, there is actually a clear ordering; kNN residuals is the most efficient, closely followed by Tree variance. The third best is kNN variance, which is clearly worse than the two best, but at the same time substantially better than kNN distance. Again, there are small differences regarding error and rejection rates, that for all setups are very close to the required.

Looking finally at the hardest version of the data set, i.e., kin8nh, the overview in Figure 7 shows, as expected, generally larger intervals. Still, all difficulty estimators are able to produce tighter intervals for higher rejection levels. When comparing the setups in Figure 8, kNN distance is again the least efficient for all rejection levels. The other three difficulty estimators produce very similar intervals for the lower rejection levels, but for the higher, i.e., rejecting 80% or 90%, Tree variance is the best, followed by kNN residuals.

Summarizing the results for the four kin data sets, the main observation is that, for these data sets, all setups consistently deliver the required rejection and error rates, whilst also achieving increased efficiency for higher rejection levels. Thus, these detailed results demonstrate that the method works, with respect to properties 1-4 stated in section 3 above. Overall comparison of difficulty estimators shows that kNN distance clearly performs the worst on all four data sets, with kNN residuals and Tree variance performing best overall.

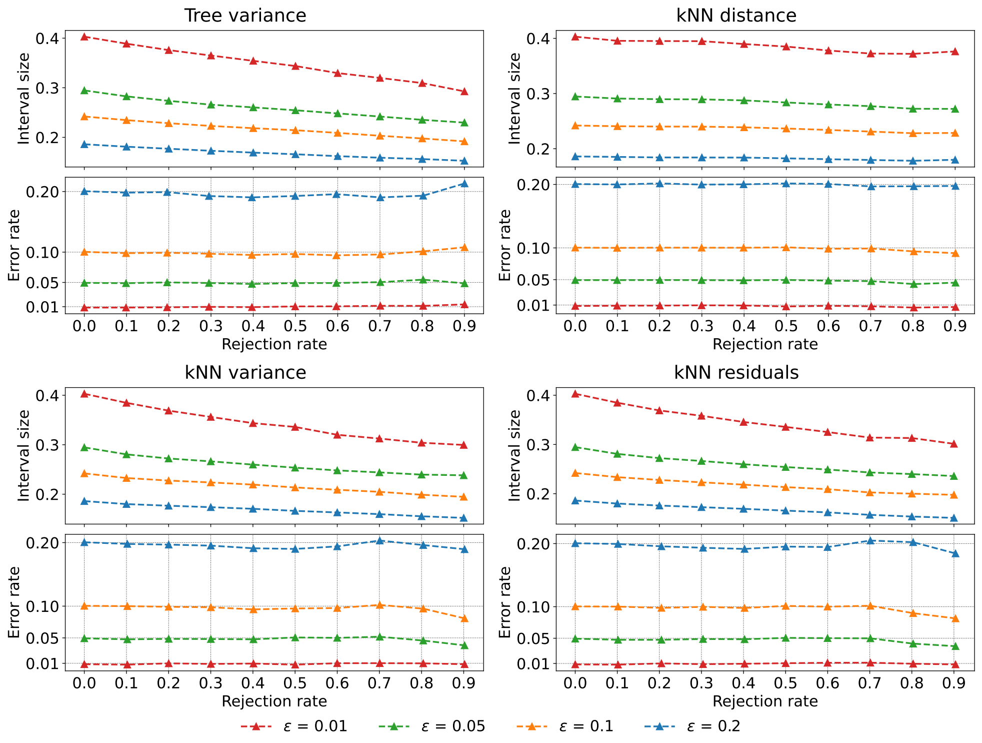

Figure 1: Kin8fh - overview

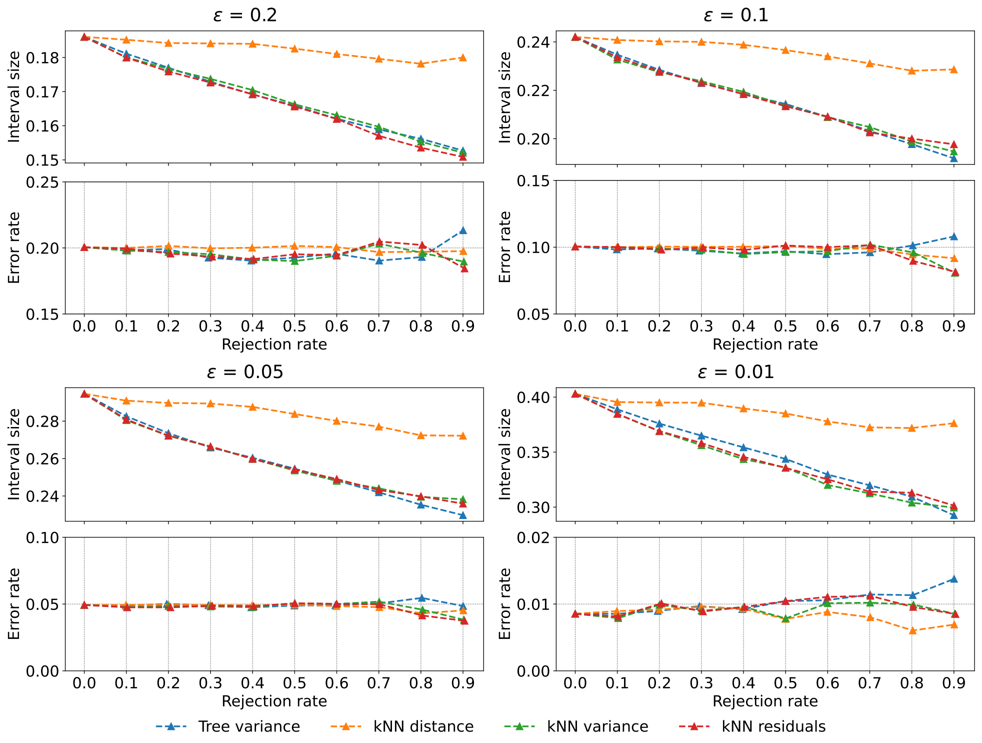

Figure 2: Kin8fh - estimator comparison

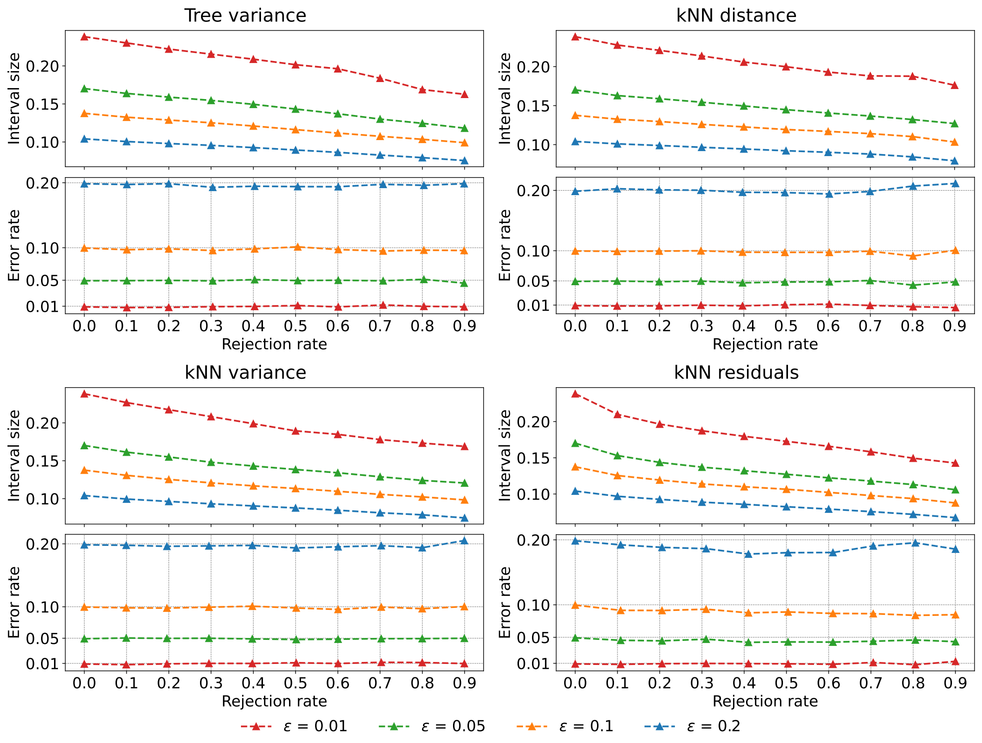

Figure 3: Kin8fm - overview

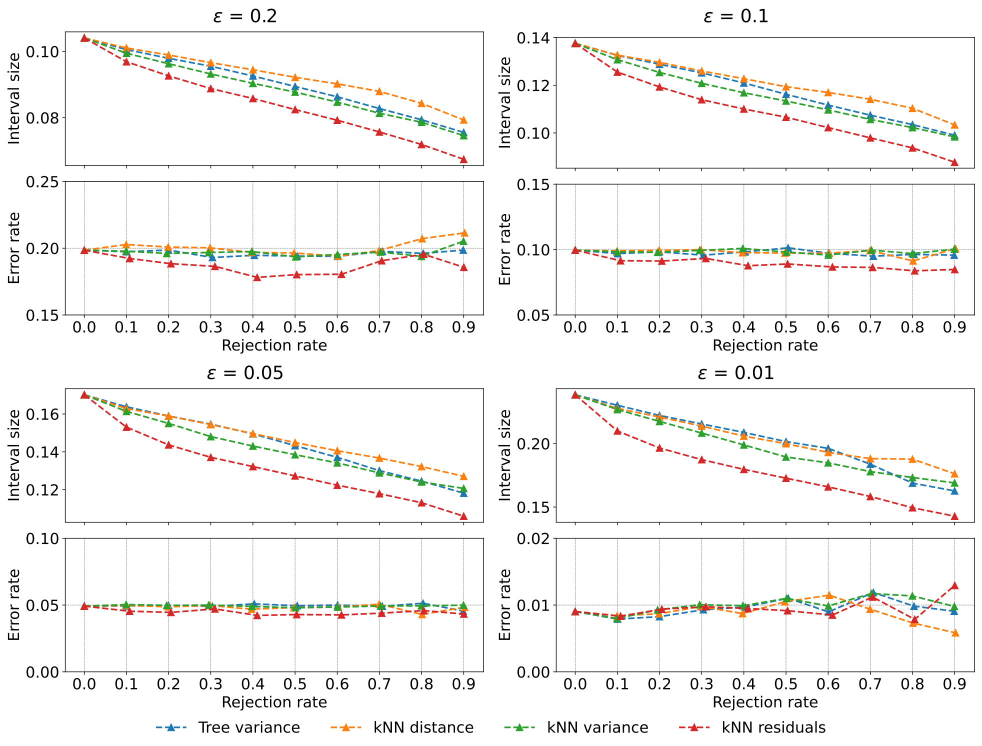

Figure 4: Kin8fm - estimator comparison

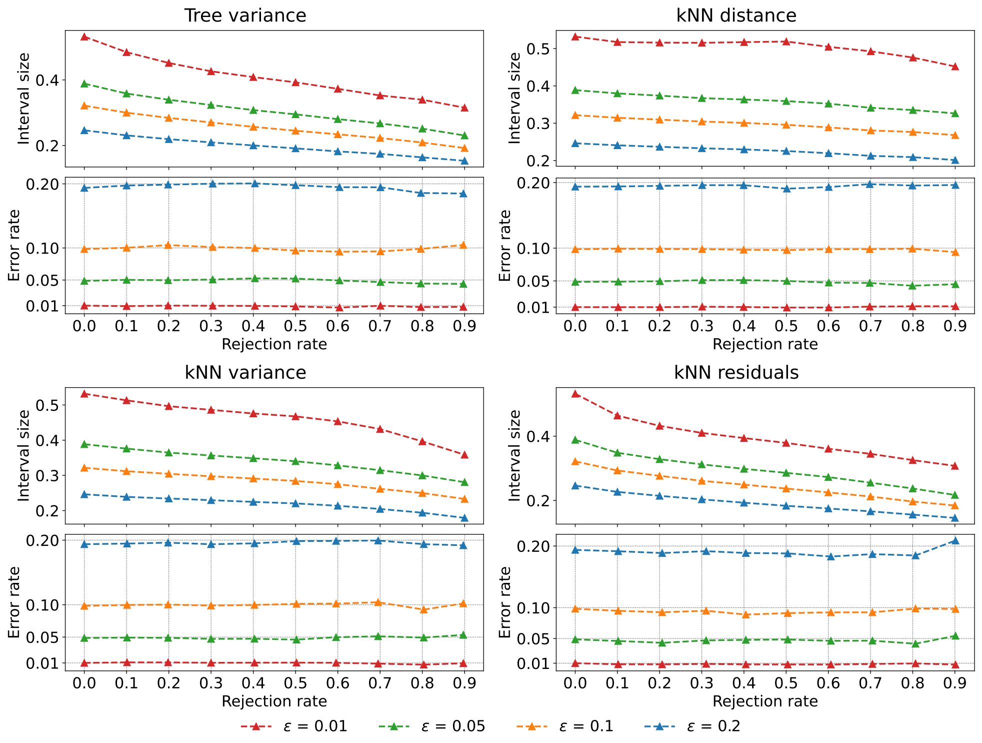

Figure 5: Kin8nm - overview

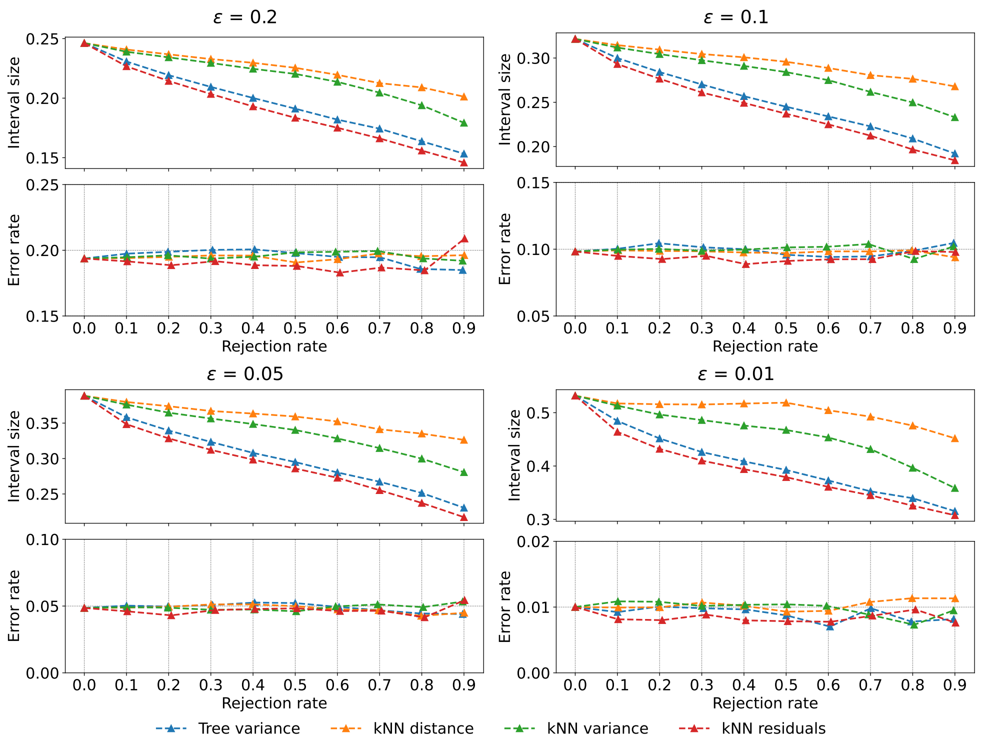

Figure 6: Kin8nm - estimator comparison

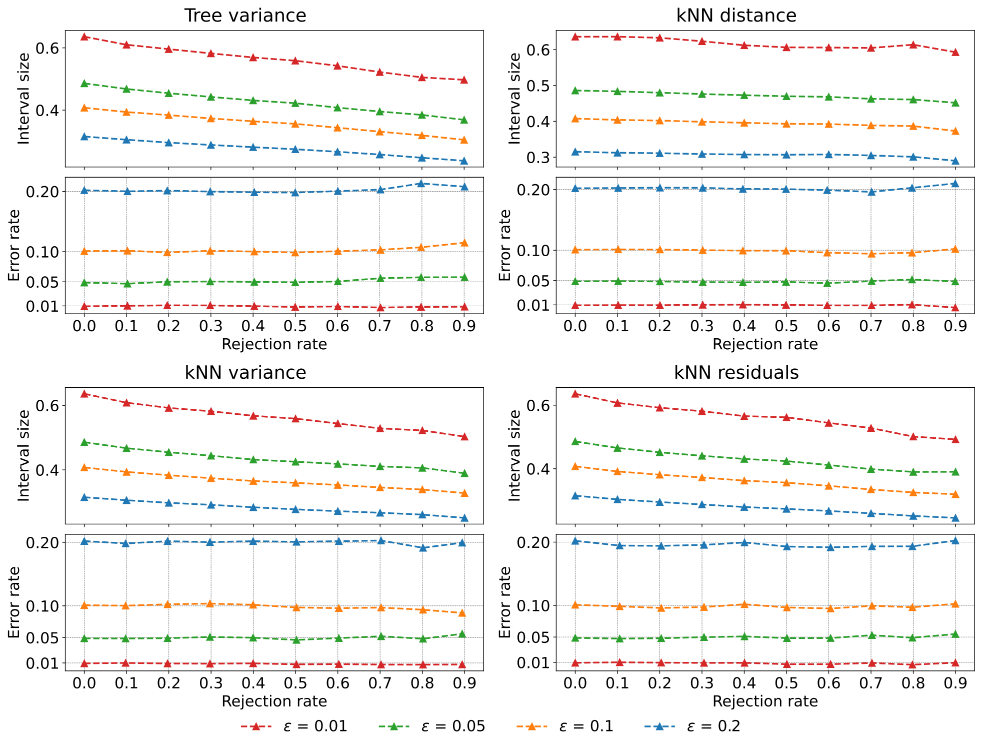

Figure 7: Kin8nh - overview

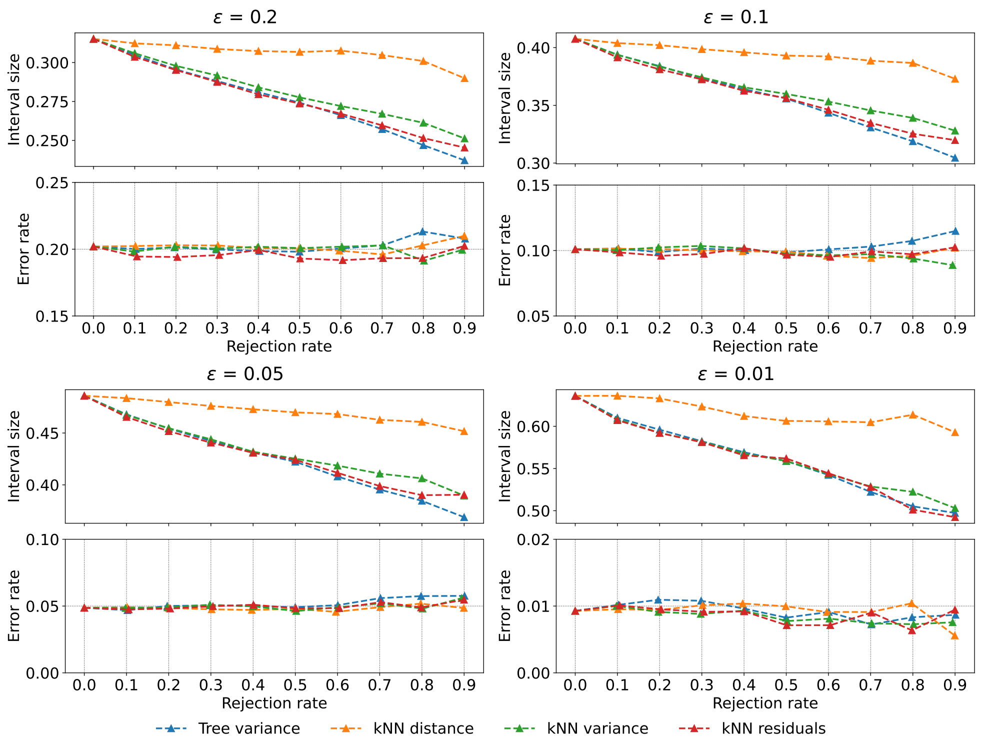

Figure 8: Kin8nh - estimator comparison

Next, we turn to aggregated results over all data sets. As described above, each test instance is compared to the difficulty threshold found from the calibration (out-of-bag) instances in order to determine if it should be rejected or not. With this in mind, it becomes important to ensure that the actual rejection level is close to the requested. As seen in Table 2 below, the empirical rejection levels are, for all four difficulty estimation functions, almost identical to the requested.

Table 2: Empirical rejection rates per rejection level

	Rejection level								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
Tree variance	.100	.201	.300	.400	.499	.600	.700	.800	.900
kNN distance	.102	.202	.303	.401	.501	.602	.701	.801	.900
kNN variance	.100	.199	.300	.400	.500	.600	.700	.800	.900
kNN residuals	.100	.199	.300	.400	.499	.600	.700	.800	.899

While the purpose of the suggested method is to produce prediction intervals, successful ordering functions should also lead to lower point prediction errors for higher rejection levels. Looking at the mean absolute errors in Table 3 below, all four difficulty estimators exhibit this pattern. Comparing the setups, Tree variance is the best, followed by kNN residuals and kNN variance.

Table 3: Mean absolute errors per rejection level

	Rejection level									
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Tree variance	.057	.054	.052	.049	.047	.044	.042	.039	.036	.033
kNN distance	.057	.056	.055	.054	.053	.052	.052	.051	.050	.049
kNN variance	.057	.055	.052	.051	.049	.047	.045	.044	.041	.038
kNN residuals	.057	.054	.052	.050	.048	.047	.045	.043	.041	.038

Table 4 shows the empirical error rates. As expected, these are very close to the significance levels. If anything, there is a small tendency that the conformal regressors are conservative. This is, as described above, to be expected when using out-of-bag calibration.

Table 4: Conformal error rates per rejection level

		Rejection level									
		0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
$\epsilon = 0.2$	Tree variance	.199	.199	.200	.198	.198	.199	.198	.200	.200	.203
	kNN distance	.199	.199	.199	.199	.198	.198	.197	.198	.199	.198
	kNN variance	.199	.199	.198	.198	.198	.197	.196	.199	.198	.201
	kNN residuals	.199	.198	.198	.197	.196	.197	.197	.198	.203	.204
$\epsilon = 0.1$	Tree variance	.099	.099	.100	.099	.099	.099	.098	.098	.100	.100
	kNN distance	.099	.099	.099	.099	.099	.100	.098	.099	.098	.096
	kNN variance	.099	.099	.099	.099	.099	.098	.098	.100	.100	.100
	kNN residuals	.099	.098	.098	.099	.098	.099	.099	.100	.101	.100
$\epsilon = 0.05$	Tree variance	.049	.049	.050	.049	.049	.049	.049	.049	.050	.051
	kNN distance	.049	.049	.049	.049	.049	.049	.049	.050	.049	.050
	kNN variance	.049	.049	.049	.050	.049	.048	.048	.050	.050	.051
	kNN residuals	.049	.049	.049	.050	.049	.049	.049	.050	.050	.050
$\epsilon = 0.01$	Tree variance	.010	.010	.010	.010	.010	.009	.009	.010	.009	.010
	kNN distance	.010	.010	.010	.010	.010	.010	.010	.010	.009	.009
	kNN variance	.010	.010	.010	.010	.010	.009	.010	.010	.009	.009
	kNN residuals	.010	.010	.010	.010	.010	.010	.010	.010	.009	.009

A key component of the suggested method is the possibility for a user to know the interval sizes of future test set predictions for different significance and rejection levels. Table 5 below shows these numbers. First of all, it should be noted that all setups do indeed lead to tighter intervals for higher rejection levels. Starting with no rejected instances, and remembering that all target values were scaled to the interval $[0, 1]$, we see that the mean interval size varies from approximately 0.41 for $\epsilon = 0.01$ to 0.18 for $\epsilon = 0.2$. If the procedure is allowed to reject 50% of all instances, the interval sizes are reduced by approximately 20% for these two significance levels. Rejecting 90%, the corresponding reduction in interval size is 44% for $\epsilon = 0.2$ and 36% for $\epsilon = 0.01$. Ranking the four difficulty estimators, Tree variance produces the tightest intervals for almost all significance and rejection levels. The results for the two setups kNN variance and kNN residuals are quite similar, although kNN residuals perform slightly better overall. Both kNN variance and kNN residuals are clearly worse than Tree variance, but substantially better than kNN distance.

As a complement to the results in Table 5, we also look at the corresponding interval sizes for the rejected instances in Table 6 below. Again, the results follow the expected behavior. Specifically, as seen by the very large intervals when only rejecting rather few instances, i.e., rejection levels 10% - 30%, many data sets contain some instances that are found to be very difficult, and thus rejected, by the difficulty estimators.

Summarizing the aggregated results, it is seen that also over all data sets the method works well, with empirical error and rejection rates matching requested levels. Furthermore, interval sizes invariably decrease overall, as the rejection level is increased, empirically demonstrating that this desired efficiency property holds.

Table 5: Interval sizes of non-rejected instances per rejection level

		Rejection level									
		0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
$\epsilon = 0.2$	Tree variance	.178	.167	.158	.150	.142	.134	.125	.116	.106	.095
	kNN distance	.178	.173	.170	.167	.164	.162	.159	.156	.153	.150
	kNN variance	.178	.169	.162	.156	.150	.144	.138	.131	.124	.113
	kNN residuals	.178	.168	.161	.155	.149	.143	.137	.131	.123	.112
$\epsilon = 0.1$	Tree variance	.238	.223	.211	.200	.189	.179	.168	.157	.144	.131
	kNN distance	.238	.231	.227	.222	.218	.214	.211	.207	.203	.197
	kNN variance	.238	.225	.215	.207	.199	.192	.185	.176	.166	.153
	kNN residuals	.238	.224	.214	.206	.198	.190	.183	.174	.164	.152
$\epsilon = 0.05$	Tree variance	.292	.274	.260	.248	.236	.224	.211	.198	.184	.169
	kNN distance	.292	.284	.278	.273	.268	.264	.259	.254	.249	.243
	kNN variance	.292	.276	.264	.254	.245	.237	.229	.219	.207	.191
	kNN residuals	.292	.274	.262	.252	.243	.234	.226	.215	.204	.192
$\epsilon = 0.01$	Tree variance	.407	.384	.368	.352	.339	.326	.309	.291	.273	.252
	kNN distance	.407	.394	.388	.382	.376	.372	.363	.356	.357	.347
	kNN variance	.407	.385	.370	.359	.345	.336	.324	.313	.299	.278
	kNN residuals	.407	.380	.365	.351	.338	.328	.317	.305	.292	.277

Table 6: Interval sizes of rejected instances per rejection level

		Rejection level									
		0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
$\epsilon = 0.2$	Tree variance	.271	.254	.241	.230	.221	.212	.203	.195	.187	.095
	kNN distance	.224	.212	.206	.201	.196	.191	.188	.185	.181	.150
	kNN variance	.262	.244	.231	.221	.213	.205	.198	.192	.185	.113
	kNN residuals	.269	.247	.233	.223	.214	.206	.199	.192	.185	.112
$\epsilon = 0.1$	Tree variance	.342	.322	.307	.294	.284	.274	.265	.256	.247	.131
	kNN distance	.296	.279	.270	.264	.259	.254	.250	.246	.242	.197
	kNN variance	.334	.314	.298	.287	.277	.268	.260	.253	.246	.153
	kNN residuals	.341	.316	.300	.288	.277	.268	.261	.253	.246	.152
$\epsilon = 0.05$	Tree variance	.407	.382	.366	.353	.341	.331	.321	.312	.303	.169
	kNN distance	.364	.343	.331	.323	.316	.311	.306	.301	.297	.243
	kNN variance	.402	.378	.359	.346	.335	.325	.317	.309	.301	.191
	kNN residuals	.406	.379	.360	.347	.335	.325	.317	.309	.301	.192
$\epsilon = 0.01$	Tree variance	.542	.512	.489	.473	.461	.449	.439	.429	.418	.252
	kNN distance	.501	.469	.451	.442	.436	.428	.423	.418	.412	.347
	kNN variance	.527	.504	.482	.468	.456	.445	.435	.426	.417	.278
	kNN residuals	.538	.507	.481	.468	.456	.443	.434	.425	.416	.277

Finally, Table 7 below shows the Spearman correlations between test set difficulty estimations and test set errors. Despite the success of the proposed method, these correlations are generally rather low and also vary substantially between techniques on some data sets.

As expected from the other results, Tree variance has the highest mean correlation, but it could be noted that other options are actually better on some individual data sets. kNN distance has the lowest correlation of the four setups on a large majority of all data sets.

Table 7: Correlations between difficulty estimates and errors

	Tree variance	kNN distance	kNN variance	kNN residuals
abalone	.380	.214	.350	.365
bank8fh	.279	.122	.188	.141
bank8fm	.461	.303	.360	.320
bank8nh	.264	.209	.181	.183
bank8nm	.599	.478	.553	.574
comp	.399	.274	.326	.433
deltaA	.413	.324	.409	.425
deltaE	.193	.108	.166	.155
kin8fh	.177	.035	.192	.194
kin8fm	.206	.180	.248	.401
kin8nh	.207	.058	.198	.235
kin8nm	.333	.115	.165	.431
puma8fh	.261	-.035	.266	.249
puma8fm	.301	-.045	.297	.276
puma8nh	.346	.072	.262	.228
puma8nm	.322	.040	.281	.213
Mean	**.321**	**.153**	**.278**	**.301**

5. Concluding remarks.

We have in this paper introduced and evaluated conformal regression with reject option. The key ideas, consistent with the conformal approach, are that a test prediction should either be a prediction interval or a reject, and that the non-rejected predictions are valid. The empirical evaluation demonstrated these properties, as well as the fact that all evaluated difficulty estimators produced tighter intervals the more instances the regressor was allowed to reject. From a practitioner's perspective, the suggested setup makes it possible to compare the interval sizes resulting from combinations of significance and rejection levels before making any predictions.

In the experimentation, four different difficulty estimators were evaluated; three looking at different properties of neighboring instances, and one, which turned out to be the most successful, measuring the disagreement between the trees in the Random forest. While all difficulty estimators worked as intended, i.e., produced orderings that were good enough to make the intervals tighter for higher rejection levels, it should be noted that the correlation between the difficulty estimates and prediction errors was not very high. So, one obvious avenue for future work is exploring new and potentially stronger difficulty estimators. Specifically, investigating different, and potentially varying, number of neighbors to consider would be a straightforward extension.

Acknowledgements

The authors acknowledge the Swedish Knowledge Foundation, Jönköping University, and the industrial partners for financially supporting the research through the following projects: AFAIR (grant 20200223), PREMACOP (grant 20220187) and ETIAI (grant 20230036), as part of the research and education environment SPARK at Jönköping University, Sweden.

References

Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, and Salvador García. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

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

Henrik Boström. Crepes: a Python package for generating conformal regressors and predictive systems. In *COPA*, volume 179 of *Proceedings of Machine Learning Research*. PMLR, 2022.

Henrik Boström and Ulf Johansson. Mondrian conformal regressors. In *COPA 2020*, volume 128 of *Proceedings of Machine Learning Research*, pages 114–133. PMLR, 2020.

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

Chi-Keung Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, EC-6(4):247–254, 1957.

Chi-Keung Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46, 1970.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International conference on machine learning*, pages 2151–2159. PMLR, 2019.

Blaise Hanczar and Edward R. Dougherty. Classification with reject option in gene expression data. *Bioinform.*, 24(17):1889–1895, 2008.

Wenming Jiang, Ying Zhao, and Zehan Wang. Risk-controlled selective prediction for regression deep neural network models. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. doi: 10.1109/IJCNN48605.2020.9207676.

Ulf Johansson, Henrik Boström, Tuve Löfström, and Henrik Linusson. Regression conformal prediction with random forests. *Machine Learning*, 97(1-2):155–176, 2014.

Ulf Johansson, Cecilia Sönströd, and Henrik Linusson. Efficient conformal regressors using bagged neural nets. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.

Ulf Johansson, Tuwe Löfström, Cecilia Sönströd, and Helena Löfström. Conformal prediction for accuracy guarantees in classification with reject option. In *Modeling Decisions for Artificial Intelligence*, page 133–145. Springer-Verlag, 2023a.

Ulf Johansson, Cecilia Sönströd, Tuwe Löfström, and Henrik Boström. Confidence classifiers with guaranteed accuracy or precision. In *International Symposium on Conformal and Probabilistic Prediction with Applications*, 2023b.

Mingkun Li and Ishwar K. Sethi. Confidence-based classifier design. *Pattern Recognition*, 39(7):1230–1240, 2006. ISSN 0031-3203.

Henrik Linusson, Ulf Johansson, and Henrik Boström. Efficient conformal predictor ensembles. *Neurocomputing*, 397:266–278, 2020. ISSN 0925-2312.

Harris Papadopoulos and Haris Haralambous. Neural networks regression inductive conformal predictor and its application to total electron content prediction. In *Artificial Neural Networks – ICANN 2010*, volume 6352 of *Lecture Notes in Computer Science*, pages 32–41. Springer Berlin Heidelberg, 2010.

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 *Machine Learning: ECML 2002*, pages 345–356. Springer, 2002.

Carl Edward Rasmussen, Radford M Neal, GE Hinton, Drew van Camp, Michael Revow, Zoubin Ghahramani, Rafal Kustra, and Rob Tibshirani. Delve data for evaluating learning in valid experiments. *www. cs. toronto. edu/delve*, 1996.

Abhin Shah, Yuheng Bu, Joshua K Lee, Subhro Das, Rameswar Panda, Prasanna Sattigeri, and Gregory W Wornell. Selective regression under fairness criteria. In *International Conference on Machine Learning*, pages 19598–19615. PMLR, 2022.

Anna Sokol, Nuno Moniz, and Nitesh Chawla. Conformalized selective regression. *arXiv*, 2402.16300, 2024.

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

Ahmed Zaoui, Christophe Denis, and Mohamed Hebiri. Regression with reject option and application to knn. *Advances in Neural Information Processing Systems*, 33:20073–20082, 2020.