Symbolic Regression of Confidence Intervals for Conformal Prediction

Alberto Tonda[1,2][0000−0001−5895−4809],
Alejandro Lopez-Rincon[3][0000−0003−4491−5889],
David Rojas-Velazquez[3,4][0000−0003−4402−4736], and
Evelyne Lutton[1,2][0000−0003−0889−4427]

[1] UMR 518 MIA-PS, INRAE, Université Paris-Saclay, Palaiseau France
[2] UAR 3611 ISC-PIF, CNRS, Paris, France
{alberto.tonda,evelyne.lutton}@inrae.fr
[3] Division of Pharmacology, University of Utrecht, The Netherlands
[4] Julius Center for Health Sciences and Primary Care, University Medical Center Utrecht, The Netherlands
{a.lopezrincon,e.d.rojasvelazquez}@uu.nl

Abstract. Conformal prediction is a class of algorithms designed to deliver confidence intervals around point predictions of models, with robust theoretical guarantees. Nevertheless, when dealing with regression problems, the original methodology always computes confidence intervals of the same size, independently of the magnitude of the predicted value $\hat{y}$, impairing the potential usefulness of the information. Several alternatives to properly scale the confidence intervals have been proposed in specialized literature, using assessments of the difficulty of predictions to produce wider intervals for more difficult points and tighter ones for easier predictions. However, each conformal prediction algorithm only exploits one specific type of information to evaluate the difficulty of a point prediction, such as Euclidean distance from points observed during training, or variance in the values of predictions for neighboring points. In this work, we introduce a novel symbolic regression approach to computation of confidence intervals. The algorithm can take into account all types of information considered by other conformal predictors at the same time, delivering human-interpretable equations that describe the amplitude of the intervals, tailored to the specific regression problem under evaluation. Experimental results show that the proposed approach outperforms other conformal predictors on an established benchmark suite of regression problems.

Keywords: Confidence intervals · Conformal prediction · Conformal regression · Machine learning · Regression · Symbolic regression.

1 Introduction

The last decades have seen a considerable stream of success stories for predictive models able to exploit available data, from physics-based simulation to empirical models, to machine learning (ML) models. Most models, however, are designed to deliver a single answer, also called *point prediction*, be it a class in the case of classification, or a continuous value for regression problems. For the end user, it would be more informative if the models also delivered a *predictive set*, a range of values within which the true value has a high probability of being found. For classification problems, the predictive set can take the form of a list of classes; for regression problems, it can be a confidence interval around a point prediction. Predictive sets can provide more insight into the prediction: for example, a wider confidence interval indicates greater model uncertainty, while a tighter interval suggests higher confidence in the prediction.

Still, predictive models can widely differ in design and structure, so obtaining predictive sets might require ad-hoc techniques for each specific model. Conformal prediction (CP) [11,17] is a class of model-agnostic algorithms designed to provide predictive sets for any kind of predictive model, just by testing their behavior on a data set unseen during training, called *calibration set*. After evaluating the distribution of a *non-conformity score* on the samples in the calibration set, CP algorithms can then compute predictive sets for new samples, with robust theoretical guarantees that the true value will be found within the predictive set for new samples with a user-defined frequency.

The standard CP algorithm for regression is simple and effective, but delivers confidence intervals of the same width for all point prediction, a characteristic that impairs its usefulness, as the users in general would rather have tighter confidence intervals for small predicted values, and wider confidence intervals for larger values. The CP community delivered more sophisticated approaches to solve the issue, like normalized CP algorithms [13] and Mondrian CP [2], that attempt to estimate the difficulty of a point prediction to then resize the intervals accordingly. However, each approach relies upon a single difficulty estimation metric, which could be more or less effective depending on the target problem.

In this work, we propose a novel approach to computing confidence intervals for regression problems, based on symbolic regression (SR) [14,20], which we call SRCP. Using the same approach as other CP methods, we train a SR model to predict the non-conformity scores of samples in the calibration set, based on the information from all difficulty estimation metrics commonly employed by all normalized CP algorithms. Being able to pick and choose the difficulty estimation metrics most appropriate for the problem at hand, and include them in potentially complex equations, SRCP should be able to automatically discover the best solution for each data set.

SRCP is compared to other state-of-the-art CP approaches for confidence intervals in regression, on two quality metrics: *coverage*, the frequency by which the true value is found within the predictive set; and the median size of the confidence intervals produced. Since this is a bi-objective problem, we consider the Pareto-optimality of the different approaches. Experimental results carried out on an established benchmark suite for regression, OpenML-CTR23 [10], show that SRCP is Pareto-dominant more often than all competitors, opening a promising research line for the use of SR in the prediction of confidence intervals.

2 Background

2.1 Conformal prediction

In the context of this work, a *predictor* p is defined as any kind of function that, given in input a sample vector $X_i = \{x_1, x_2, ..., x_N\}$ containing values for N features, returns a predicted value $\hat{y}_i$ for a target feature:

$$\hat{y}_i = p(X_i) \tag{1}$$

Examples of predictors include machine learning classifiers and regressors, but the definition is intentionally broad, so that it includes any kind of mathematical or simulation-based model. The performance of a predictor can be evaluated using a problem-specific *quality metric* that takes into account the known, measured value of the target y. For instance, mean squared error and the coefficient of correlation $R2$ are popular metric for regression; while precision, recall, and the $F1$ score are commonly adopted for classification.

Still, while obtaining a single predicted value $\hat{y}$ for a sample (a so-called *point prediction*) can be enough for some applications, for the end user is typically more informative to have access to a range of values, within which the predictor has high confidence that the measured value will be found. In more formal terms, given $\alpha \in [0, 1]$ as an acceptable value of miscoverage (as in, the frequency for which the measured value will *not* be found in the range), it is now possible to define a *predictive set* $\mathcal{C}_\alpha$ for any dataset $Z = \{X, y\}$ as:

$$\mathbb{P}\{y \in \mathcal{C}_\alpha(X)\} \geq 1 - \alpha \tag{2}$$

For example, considering a point prediction of a continuous value $\hat{y}_i \in \mathbb{R}$ for sample X_i, the corresponding predictive set $\mathcal{C}_\alpha(X_i)$ could be an interval:

$$\mathcal{C}_\alpha(X_i) = [\hat{y}_i - q_i^-; \hat{y}_i + q_i^+] \tag{3}$$

with q_i^-, q_i^+ being the lower and upper part of the confidence interval computed for prediction $\hat{y}_i = p(X_i)$. In other words, if $\alpha = 0.05$, $\mathcal{C}_{0.05}$ will contain confidence intervals for each possible prediction $\hat{y_i}$, with q_i^- and q_i^+ sized so that the frequency of the true value y_i falling within $\mathcal{C}_\alpha(X_i)$ will be $1 - \alpha = 0.95$. Not only this could deliver more reliable predictions for the end user, but the amplitude of the confidence intervals could also be informative: wider confidence intervals on a prediction will indicate samples for which the predictor is more *uncertain.*

The core idea of CP [11] is to evaluate the behavior of a predictor, trained on a training set Z^t, on unseen data from the same problem, the calibration set Z^c; the information of a user-defined *non-conformity score* δ, computed on the samples in Z^c, is then used to generate predictive sets that guarantee the $(1-\alpha)$ desired coverage. In the following, we call the original algorithm Standard Conformal Prediction (SCP) to differentiate it from other CP approaches.

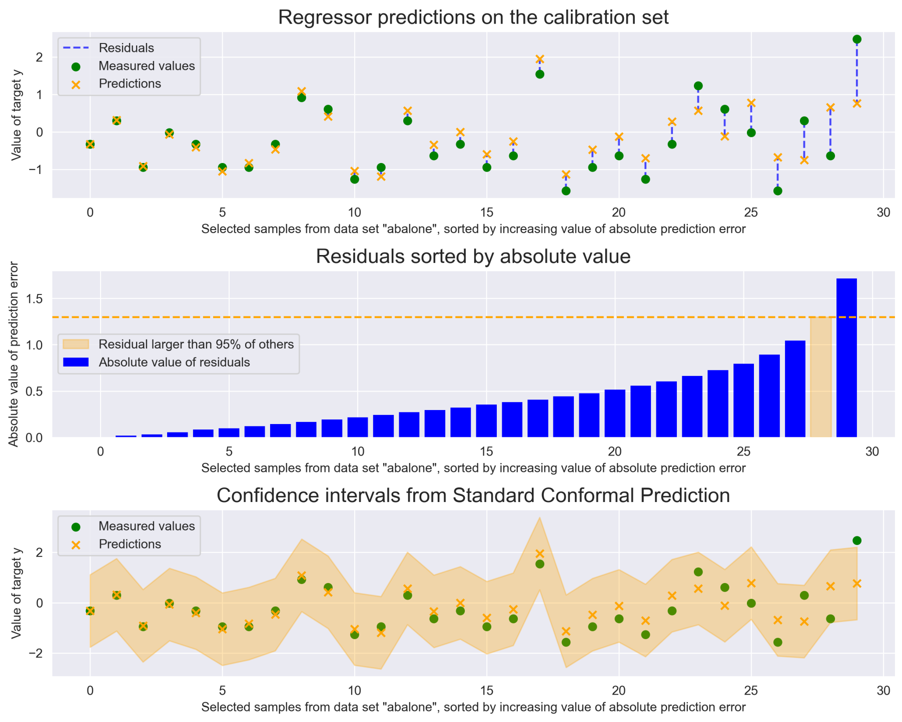

Fig. 1. Example of Standard Conformal Prediction for regression. (**top**) After being trained on a training set Z^t, a regressor is tested on an unseen data set from the same problem, the calibration set Z^c. The point predictions $\hat{y}_i$ will have different errors with respect to the measured values y_i, depending on each sample. (**middle**) The residual errors for each sample are sorted by absolute value, and the error in correspondence of a user-defined threshold (for example, 0.95) is identified. (**bottom**) The identified value is applied to each point prediction as a confidence interval, and now the measured value is found within the confidence intervals with the user-defined frequency of 0.95

Standard conformal predictors (SCPs) [24] for regression problems use absolute prediction errors, also called *residuals* or *pseudo-residuals*, as the non-conformity score δ:

$$\delta_i = |y_i - \hat{y}_i| = |y_i - p(X_i)| \tag{4}$$

Once all δ_i have been computed for samples in the calibration set Z^c, the SCP algorithm selects among them the value δ_{i_α}, defined as the smallest value that is larger than or equal to $(1-\alpha) * |Z^c|$ others (for example, with $\alpha = 0.05$, it would be the value larger than or equal to 95% of the other $\delta_{i \neq i_\alpha}$). Finally, the value for the confidence intervals is set to $q = \delta_{i_\alpha}$. An example of the application of the SCP algorithm is reported in Figure 1.

SCPs have robust theoretical guarantees and work well in practice, but they have a considerable drawback for regression problems: by definition, all confidence intervals computed by SCP will have the same size, independently from the predicted value $\hat{y}$. The size is computed on the empirical $(1-\alpha)$ quantile of the non-conformity scores, which is likely to contain large values, in correspondence to large values of $\hat{y}$; but this means that even small values of $\hat{y}$ will present relatively wide confidence intervals, which might make the confidence intervals less useful in practice (for instance, imagine a $\hat{y}_i = 0.05$ with $\mathcal{C}_\alpha(X_i) = [\hat{y}_i - 0.20; \hat{y}_i + 0.20]$!). It would be highly desirable instead to have tighter confidence intervals for smaller values and wider confidence intervals for larger values. The CP community tackled the problem by proposing different ways of resizing confidence intervals, on the basis of problem characteristics. It is interesting to notice that, while the SCP algorithm does not have any hyper-parameter to tune, the more sophisticated algorithms described in the following require choices by the user.

Normalized conformal predictors (NCPs) [16] relate the non-conformity function to an estimation of the accuracy of the underlying predictor p on a sample $Z_i = \{X_i, y_i\}$, called σ_i:

$$\delta_i = \frac{|y_i - \hat{y}_i|}{\sigma_i + \beta} = \frac{|y_i - p(X_i)|}{\sigma_i + \beta} \tag{5}$$

where β is a small constant, to prevent the denominator from being zero. Just as in SCP, the non-conformity scores δ_i are sorted, and the one that satisfies the miscoverage α (e.g. if $\alpha = 0.05$, it will be the δ_{i_α} larger than 95% of the other values) is selected. Differently from SCP, when NCP algorithms need to compute the confidence interval for a new unseen sample $Z_T = \{X_T, y_T\}$ they will employ the σ_T value computed for the sample in order to rescale the confidence intervals as follows:

$$q_T = \delta_{i_\alpha} * (\sigma_T + \beta) \tag{6}$$

$$\mathcal{C}_\alpha(X_T) = [\hat{y}_T - q_T; \hat{y}_T + q_T] \tag{7}$$

In this way, intervals become larger for instances that are considered more difficult to estimate, and vice versa. While several metrics to compute values of σ have been proposed in CP literature, all NCP algorithms employ the same strategy to estimate the value of σ for unseen samples, namely a K-Nearest Neighbors (KNN) function trained on the training set Z^t. Here below, four examples of σ functions used in practice, which will also be used later in the experimental evaluation:

$$NCP_d: \qquad \sigma_i^d = \sum_{k=k_0}^{K} \left(\sqrt{\sum_{n=0}^{N} (x_{ni} - x_{nk})^2} \right) \tag{8}$$

where $K = \{k_0, ..., k_K\}$ are the indices of the nearest neighbors in Z^t of sample $X_i \in Z^c$. Eq. 8 amounts to computing σ^d as the sum of the Euclidean distances from the current sample to the K nearest neighbors. In other words, if the sample is close to points already seen by the predictor during training, the value of q computed by Eq. 6 will be smaller, and the resulting confidence interval will be tighter. Vice versa, points far away from the samples seen during training will have wider confidence intervals. Another possible approach is to use the standard deviation of the measured values of y:

$$NCP_{std}: \qquad \sigma_i^{std} = \frac{\sum\limits_{k=k_0}^{K} (y_k - \bar{y})^2}{K} \tag{9}$$

where $\bar{y} = \frac{\sum^K y_k}{K}$ is the mean value of the y over the K nearest neighbors of $X_i \in Z^c$ found in Z^t, again identified using the Euclidean distance in the space of features. The idea behind this approach is that larger standard deviations of y might indicate areas of the function with a highly irregular behavior, and thus predictions should be linked to a larger incertitude. Another way of computing σ is to take into account the residuals of the K nearest neighbors:

$$NCP_{res}: \qquad \sigma_i^{res} = \sum_{k=k_0}^{K} |y_k - \hat{y}_k| \tag{10}$$

with y_k and $\hat{y}_k$ being the measured value and the predicted value of the target for sample k, respectively. A relatively small sum of residuals might indicate a part of the feature space where the predictor is performing well; on the other hand, a large value of σ^{res} might indicate an area where the predictor is struggling, and thus the predicted value should be associated with a larger incertitude. It is interesting to notice that, while this approach might intuitively look more reliable, several state-of-the-art predictors like Random Forest overfit the training set [4], leading to an underestimation of the residuals for training samples. For these kinds of predictors, the approach in Eq 10 is redefined to take into account Out-Of-Bag (OOB) predictions [13]. In RF's algorithm, each tree in the ensemble only sees a random subset of the training samples during training; samples unobserved by a tree are called OOB. When CP uses OOB evaluations, $\hat{y}_k$ is estimated using only the trees for which sample X_k is considered OOB. We denote this strategy as NCP_{oob} and the difficulty estimation metric as σ^{oob}, respectively.

A last type of NCP, which can only work on ensemble predictors like Random Forest or XGBoost, computes σ based on the variance in the predictions of each element in the ensemble (usually a weak predictor, like a Decision Tree):

$$NCP_{var}: \qquad \sigma_i^{var} = \sum_{j=0}^{P} (\hat{y}_{ij} - \bar{\hat{y}})^2 \tag{11}$$

where $\hat{y}_{ij}$ is the prediction of element j of the ensemble for the target y_i associated to sample X_i, and $\bar{\hat{y}}$ is the mean value of the predictions over all P

elements in the ensemble. NCP_{var} follows the intuition that the predictions of the elements of an ensemble will differ more for samples that are more difficult to predict. In most practical applications, NCPs perform well; in some cases, however, it has been experimentally noticed that NCP intervals may become too large to be informative [2], an observation that led to the development of Mondrian conformal predictors.

Mondrian conformal predictors (MCPs) [2] split samples into subsets. MCPs have one extra user-defined hyperparameter n_{bins}, that is the number of non-overlapping Mondrian categories, which can also be seen as the number of bins to use for a binning procedure. Then one prediction interval is estimated for each bin, using the SCP algorithm. Categories can be built according to the features X of the samples, but another alternative is to build an equal-sized binning based on difficulty scores $\{\sigma_1, ..., \sigma_q\}$ computed by NCPs.

2.2 Symbolic regression

The term symbolic regression (SR) defines a class of algorithms that search the space of mathematical expressions to find the equation model that best fits a given data set. Unlike traditional regression methods that specify a predefined model structure, symbolic regression algorithms dynamically evolve the structure of the model along with its parameters. This process typically involves evolutionary algorithms, more precisely, genetic programming (GP), to explore and optimize the space of possible models [14], but there exist alternative approaches based on neural networks [15,19]. The resulting models are in the form of mathematical expressions, which can be more interpretable than complex black-box models [20]. As equations that become too complex lose their interpretability, several SR approaches found ways to apply a pressure for simpler candidate models, for example using multi-objective optimization to balance the complexity-accuracy trade-off [7,23].

3 Proposed approach

The main idea of the presented approach is to use Symbolic Regression to predict the amplitude of the confidence intervals for each given predicted value of the target feature $\hat{y}_i$, exploiting the information contained in the value of $\hat{y}_i$, in the features X_i, and all other possible σ_i assessing the difficulty of predicting sample X_i. Both NCPs and MCPs share this same intuition, but each different type only exploits one specific type of information, which might be more or less effective, depending on the problem.

Following the steps of CP, a predictor p will first be trained on a training set Z^t. Given a calibration set Z^c, the proposed approach will train a symbolic regressor s attempting to predict the values δ_i:

$$\delta_i = |y_i - \hat{y}_i| = |y_i - p(X_i)| \qquad \forall X_i, y_i \in Z_c \tag{12}$$

with y_i the true value for the target of sample i and $\hat{y}_i = p(X_i)$ the value predicted for the target by predictor p, starting from the features X_i of sample i. In other words, here the non-conformity score δ_i represents the smallest amplitude from the predicted value $\hat{y}_i$ that is necessary to cover the true value y_i.

It is important to notice that the errors of a regressor attempting to approximate δ_i need to be weighted differently; an approximation $\hat{\delta}_i > \delta_i$ is wide enough to cover the true value, but a $\hat{\delta}_i < \delta_i$ is not. For this reason, the error function of the SR approach is defined as:

$$E(s, X, y, \hat{y}) = \frac{\sum_{i=0}^{N} e(\hat{\delta}_i, \delta_i)}{N} \qquad e(\hat{\delta}_i, \delta_i) = \begin{cases} w_e * (\hat{\delta}_i - \delta_i)^2 & \text{if } \hat{\delta}_i < \delta_i \\ (\hat{\delta}_i - \delta_i)^2 & \text{otherwise} \end{cases} \tag{13}$$

with s a candidate regressor for confidence intervals, $\hat{\delta}_i = s(\hat{y}_i, X_i, \sigma_i)$ a prediction for the confidence intervals of sample i, N number of samples, and $w_e >> 1.0$ a user-defined hyperparameter, tuned to give more importance to errors that underestimate δ_i. Furthermore, when computing confidence intervals, it is also necessary to define the acceptable miscoverage α. For this reason, an additional penalty is introduced to penalize candidate solutions that exceed the miscoverage:

$$P_\alpha(s, X, y, \hat{y}, \alpha) = \begin{cases} (1-\alpha) - \mathbb{P}\{y \in \mathcal{C}_\alpha(X)\} & \text{if } \mathbb{P}\{y \in \mathcal{C}_\alpha(X)\} < 1 - \alpha \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where $\mathcal{C}_\alpha$ is the predictive set generated by the candidate s on data set X (for instance a set of intervals such as in Eq. 3), and $\mathbb{P}\{y \in \mathcal{C}_\alpha(X)\}$ the frequency of the elements of the predictive set covering the true value y. If this frequency is below the desired threshold $(1 - \alpha)$, the candidate solution will be penalized proportionally to the difference. This penalization term is weighted by a user-defined hyperparameter w_α in the final fitness function, as follows:

$$F(s, X, y, \hat{y}, w_e, w_\alpha, \alpha) = E(C, X, \hat{y}, w_e) + w_\alpha * P_\alpha(s, X, \hat{y}, \alpha) \tag{15}$$

4 Experimental evaluation

All the necessary code to reproduce the experiments is implemented in Python 3, and freely available on a GitHub repository[5].

4.1 Quality metrics

A classic metric to evaluate the performance of a regression algorithm on a given data set $Z = \{X, y\}$ of size N is the mean squared error (MSE), defined as $MSE = \sum_{i=0}^{N} (y_i - \hat{y}_i)^2 / N$ where y_i is the measured value of the target for the

[5] `https://github.com/albertotonda/symbolic-regression-conformal-prediction`

i-th sample, $\hat{y}_i$ is the value predicted by the regression model for the i-th sample, and N is the number of samples in the data set. Still, MSE is often hard to interpret, because if the data has not been normalized, the magnitude of the result is not very informative (e.g. an extremely high value might still indicate a good performance).

For this reason, the correlation coefficient $R2$ is usually preferred, $R2 = 1 - \frac{\sum_{i=0}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{N}(y_i - \bar{y})^2}$ with $\bar{y} = \frac{1}{N}\sum_{i=0}^{N} y_i$. $R2$ is easier to interpret, because values close to 1.00 indicate good performance, while results close to 0.00 (or negative) imply poor performance.

Once it comes to confidence intervals, the ideal confidence interval around a point prediction would be the smallest that also contains the true, measured value. Thus, it is possible to evaluate confidence intervals on these two metrics: frequency of presence of the true value within the confidence interval, called *coverage* (the higher, the better), and mean or median amplitude of the intervals (the lower, the better). Intuitively, obtaining satisfying values for both metrics is a conflicting problem: it is easy to imagine a trivial solution where the amplitude of the confidence intervals is infinite, and coverage is a perfect 1.00; or the opposite case, where the confidence intervals have amplitude 0.0, but the coverage would also likely be close to 0.0. For this reason, in our opinion it only makes sense to compare different CP algorithms using the concept of Pareto dominance [8], framing the problem as multi-objective: if a given approach has both higher coverage and smaller intervals than another, we say that the first *dominates* the second; if no other algorithm is better than a given one for both metrics, we say that that algorithm is *non-dominated*; otherwise, the algorithm is *dominated* by others. The perfect CP algorithm would ideally dominate all others, or at least be non-dominated in as many cases as possible. An example of the evaluation is presented in Figure 2.

4.2 Regressor

Being model-agnostic, in principle conformal prediction can be applied to any regressor: for the following experiments we selected Random Forest (RF) [4]. RF has several desirable characteristics for our proof of concept: being an ensemble approach, it is possible to get separate predictions for each predictor in the ensemble, which makes it possible to compute statistics such as the inter-predictor variance for each sample, which is used by the NCP_{var} approach, see Eq. 11; and since creating each predictor does not use all available samples in the training set, it is also possible to obtain OOB predictions for use in NCP_{oob}, see Eq. 10. The RF implementation is from the `scikit-learn` Python package [18], with its default hyperparameters, except for the number of estimators set to $n_P = 500$, following [2].

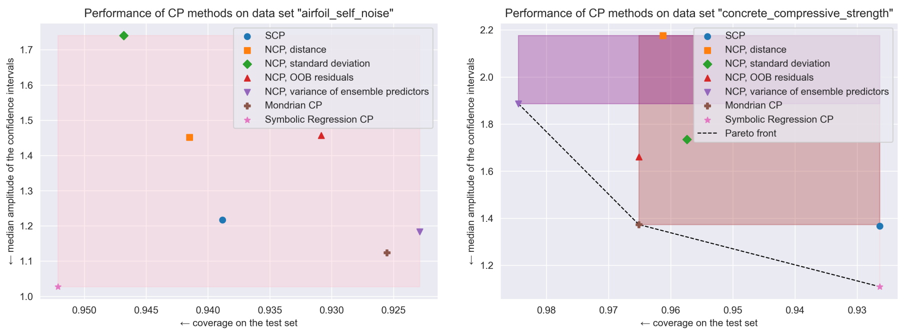

Fig. 2. Two extracts from the experiments, on data set `airfoil_self_noise` (left) and `concrete_compressive_strength` (right). The x-axis is inverted to improve readibility, so that the best values are to the left and to the bottom. Each point represents the performance of a CP method, and squares of the same color cover points dominated by that approach. For the dataset on the left, the result obtained by SRCP dominates all other approaches. On the right, SRCP is Pareto-optimal (non-dominated), together with two other methods.

4.3 Data set selection

OpenML-CTR23 [10] is a benchmark suite containing 36 curated data sets for regression, hand picked from the larger OpenML repository [9,22]. In an initial evaluation, we compare the performance of RF and eXtreme Gradient Boosting (XGBoost) [6], both set with $n_P = 500$ estimators. RF is our choice for the experiments, due to its desirable properties previously described, but since XGBoost is widely considered to be the state of the art for regression of tabular data sets, we are interested in assessing how the two algorithms compare. Furthermore, we are also interested in knowing whether a typical data split used for CP (50% training, 25% calibration, 25% test) is enough to obtain a good performance. So, while RF is run on each data set 10 times, using 50% of the data as training and the rest as test, XGBoost is put in a more favorable situation, performing a classic 10-fold cross-validation (using 90% of the data for training and 10% for test at each iteration). The two algorithms are compared on their mean R2 values over the test set, and from the results (reported in the additional materials[6], Table 1), the two performances seem comparable: this evidence corroborates our choices for the experimental setup. After this first experiment, all data sets for which the performance of RF is considered poor ($R2 < 0.40$) or excellent ($R2 > 0.95$), using two arbitrary thresholds, are discarded, as there is low interest in computing confidence intervals for such extreme cases. After this operation, 22 data sets are left to be considered for the rest of the experimental evaluation.

[6] `https://github.com/albertotonda/symbolic-regression-conformal-prediction/blob/main/EA_2024__Symbolic_Regression_for_Conformal_Prediction_Additional_Materials.pdf`

4.4 Data pre-processing

For the experimental evaluation proper, for each data set columns with categorical data are converted to integer values, while columns with missing data are discarded. The data is then randomly split into a training set Z^t (50% of the samples), a calibration set Z^c (25% of the samples) and a test set Z^T (the remaining 25%). Data is normalized to zero mean and unit variance, with a normalization learned on the training set and applied to conformal and test sets. Being an ensemble of decision trees, the regressor used in the following does not strictly need data normalization for optimal performance, but we decided to apply it anyways to later ease the comparison of the amplitude of the confidence intervals obtained by the different methods used as a comparison. All data pre-processing is carried out using the `pandas` [21] and `scikit-learn` Python packages [18].

4.5 Conformal predictors

All implementations used for SCP, NCP, and MCP come from the `crepes` Python package [1]. For NCPs using KNN, the hyperparameter K defining the number of neighbors to consider for computing the values of σ is set to its default value of $K = 25$; the hyperparameter characterizing the value added to the difficulty estimates after normalization is also left to its default value $\beta = 0.01$, a value commonly found in literature for NCPs [3]. For MCPs, the binning is performed on the difficulty estimate σ^{var}; the hyperparameter representing the number of bins is initially set to $n_{bins} = 99$ and iteratively lowered until the algorithm returns non-infinite confidence intervals, or until all data is clustered together in a single bin. In practice, this last case never happens, and MCP is always able to find a value of $n_{bins} > 1$ on each data set.

The proposed methodology, employing symbolic regression to obtain confidence intervals, labeled as SRCP in the following, is implemented resorting to the `PySR` Python package [7]. `PySR` employs a GP-based multi-objective approach to SR, generating a Pareto front of equations representing trade-offs between complexity (GP tree size) and value of the fitness function. It then selects one equation on the Pareto front as the final solution, using a heuristic that takes into account the values of the other Pareto-optimal candidate solutions. The tool manages n_I parallel islands, each with a population of μ individuals, running for n_G generations. After a set of trial runs, the hyperparameters of `PySR` are set to $n_I = 15, \mu = 100, \lambda = 100, n_G = 2,000$; the operator set is $O = \{+, -, *, /, sin, cos, log, exp\}$; terminals include constants and variables describing: the value predicted by the regressor $\hat{y}$, the features X, and all the different σ values computed for the estimation of the accuracy of predictor, namely σ^d , σ^{std}, σ^{oob}, and σ^{var} (see Eqs. 8-11); all other hyperparameters are left at their default values. The weight values used in Eq. 15 are $w_e = 10$ and $w_\alpha = 100 * |Z^c|$.

4.6 Results

After testing the proposed approach and all competing algorithms on the selected data sets, the results are summarized in Table 1. The proposed approach is Pareto-dominant for all but 2 data sets, and in 3 cases it dominates all other approaches.

Table 1. Experimental results. The value in each cell illustrates the number of times that an approach dominates all others, is non-dominated (Pareto-optimal, but with other approaches on the Pareto front), or is dominated by at least another algorithm. Numbers in **bold** indicate the best algorithm in the row.

Number of times the approach	**SCP**	**NCP$_\mathbf{d}$**	**NCP$_\mathbf{std}$**	**NCP$_\mathbf{oob}$**	**NCP$_\mathbf{var}$**	**MCP**	**SRCP**
Dominates all others	0	0	0	0	0	0	**3**
Is non-dominated	6	5	6	10	11	14	**20**
Is dominated	16	17	16	12	11	8	**2**

The best equations found for each data set are reported in Table 2. It is interesting to notice that some of the difficulty estimation metrics σ appear more often than others: in particular, σ^{oob} and σ^{var} are included in 9/22 equations each, while σ^{d} only appears once, and σ^{std} is never used. The prediction $\hat{y}$ is used 4/22 times, and problem features are employed 5/22 times. These results seem to suggest that σ^{oob} and σ^{var} are generally the most informative ways of estimating difficulty, but that no single metric is the best for all data sets, which supports the use of methods like SRCP, able to exploit different types of information.

Table 2. Best equation found for each data set. $x_0, ..., x_n$ are values for the corresponding features in X for that particular data set. $\hat{y} = p(X)$ is the prediction for the target given by the regressor. $\sigma^d, \sigma^{std}, \sigma^{oob}, \sigma^{var}$ are the values computed by the different NCP approaches, see Eqs. 8-11. Cells in dark gray mark the two case studies where SRCP is not Pareto-optimal; cells in light gray highlight the case studies where SRCP dominates all other approaches.

Data set name	**Best equation**	**Data set name**	**Best equation**
`abalone`	$e^{\sigma^{oob}}$	`miami_housing`	$\sigma^d + 5.22\sigma^{var}$
`airfoil_self_noise`	$\sigma^{var} + 0.364$	`Moneyball`	$\sin(\sin(\sin(\sin(\cos(\cos(x_0(x_7 - 0.861)))))))$
`brazilian_houses`	$\log(\hat{y} + 1.23)$	`physiochemical_protein`	$\sigma^{var} + 0.923$
`california_housing`	$3.00\sigma^{var} + 0.451$	`pumadyn32nh`	$-0.0338x_{12} + 0.0338x_1 + 1.16$
`cars`	$\log\left(\sigma^{oob} + 1.11\right)$	`QSAR_fish_toxicity`	$e^{\sin\left(\frac{-x_5-0.901}{\cos\left(0.592\sigma^d+0.592x_4\right)}\right)\cos(\hat{y})} + 0.386$
`concrete_compressive_strength`	$\cos\left(\cos\left(\frac{\sigma^{var}e^{\hat{y}-\sin(\sin(x_6))}}{x_2}\right) + 0.0763\right)$	`red_wine`	$\sigma^{var} + 1.33$
`fifa`	$4.98\sigma^{oob}$	`socmob`	$\log(\hat{y} + 1.51) - \sin\left(\frac{\sigma^{var}}{\log(\sigma^{var})}\right)$
`grid_stability`	$\sigma^{oob} + 0.360$	`space_ga`	$0.706e^{\sigma^{oob}+(x_0-x_2)e^{-\sin(x_5x_0)}}$
`health_insurance`	$\sigma^{oob} + 1.09$	`superconductivity`	$\sigma^{oob} + 0.297$
`kin8nm`	$\sigma^{oob} + 0.741$	`wave_energy`	$\sigma^{oob} + 0.435$
`kings_county`	$7.96\sigma^{var} + 0.371$	`white_wine`	$1.17e^{\sigma^{var}}$

5 Discussion and conclusions

While the proposed approach seems to be effective with respect to the stated objective, there are a few connected topics that we deem worth discussing: the motivation for using SR over other possible solutions; and the evaluation of coverage guarantees. At its core, the proposed approach transforms the problem of finding appropriate confidence intervals into a regression problem, for which several other possible regressors could be employed. The motivation for the use of SR over other available candidates, such as XGBoost, RF, and similar algorithms, is twofold. First of all, modifying the objective function to be optimized to take into account coverage, as in Eq. 15, is much more straightforward in SR than in other regression algorithms. The most efficient regressors for tabular data are typically ensembles of decision trees [12] that are locally created using greedy CART algorithms [5], and changing the objective metrics either in the ensemble creation algorithm (boosting or bagging) or in the greedy CART is far from trivial. Secondly, one of the big advantages of SR is that the final model is completely human-readable, in the form of an equation. This observation leads to a second topic, concerning the validity of the coverage guarantees. All CP approaches, from SCPs to the different NCPs, to MCPs, have robust coverage guarantees, supported by mathematical demonstrations. While our approach can be framed as a NCP, since it is generally using non-linear regression, the coverage guarantees might not hold. Having a human-interpretable equation available, however, might make it possible for mathematicians to derive rigorous proofs of coverage for specific applications. This possibility will be explored in future works.

Acknowledgments. This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. EINF-6769.

Disclosure of Interests. The authors declare no competing interests.

References

1. Boström, H.: crepes: a python package for generating conformal regressors and predictive systems. In: Johansson, U., Boström, H., An Nguyen, K., Luo, Z., Carlsson, L. (eds.) Proceedings of the Eleventh Symposium on Conformal and Probabilistic Prediction and Applications. Proceedings of Machine Learning Research, vol. 179. PMLR (2022)
2. Boström, H., Johansson, U.: Mondrian conformal regressors. In: Gammerman, A., Vovk, V., Luo, Z., Smirnov, E., Cherubin, G. (eds.) Proceedings of the Ninth Symposium on Conformal and Probabilistic Prediction and Applications. Proceedings of Machine Learning Research, vol. 128, pp. 114–133. PMLR (09–11 Sep 2020)
3. Boström, H., Linusson, H., Löfström, T., Johansson, U.: Accelerating difficulty estimation for conformal regression forests. Annals of Mathematics and Artificial Intelligence **81**(1–2), 125–144 (Mar 2017)
4. Breiman, L.: Random forests. Machine learning **45**, 5–32 (2001)

5. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification And Regression Trees. Chapman and Hall/CRC (10 1984)
6. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. ACM (8 2016)
7. Cranmer, M.: Interpretable machine learning for science with PySR and SymbolicRegression.jl (2023)
8. Deb, K.: Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons (2001)
9. Feurer, M., et al.: OpenML-Python: an extensible python api for openml. arXiv **1911.02490** (2019)
10. Fischer, S.F., Feurer, M., Bischl, B.: OpenML-CTR23–a curated tabular regression benchmarking suite. In: AutoML Conference 2023 (Workshop) (2023)
11. Gammerman, A., Vovk, V., Vapnik, V.: Learning by transduction. In: Cooper, G.F., Moral, S. (eds.) UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998. pp. 148–155. Morgan Kaufmann (1998)
12. Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 507–520. Curran Associates, Inc. (2022)
13. Johansson, U., Boström, H., Löfström, T., Linusson, H.: Regression conformal prediction with random forests. Machine Learning **97**(1–2), 155–176 (Jul 2014)
14. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992)
15. Martius, G., Lampert, C.H.: Extrapolation and learning equations. CoRR **abs/1610.02995** (2016)
16. Papadopoulos, H., Haralambous, H.: Reliable prediction intervals with regression neural networks. Neural Networks **24**(8), 842–851 (Oct 2011)
17. Papadopoulos, H., Proedrou, K., Vovk, V., Gammerman, A.: Inductive confidence machines for regression. In: International Symposium on Artificial Intelligence and Mathematics, AI&M 2002, Fort Lauderdale, Florida, USA, January 2-4, 2002 (2002)
18. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
19. Petersen, B.K., Landajuela, M., Mundhenk, T.N., Santiago, C.P., Kim, S., Kim, J.T.: Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021)
20. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science **324**(5923), 81–85 (2009)
21. The pandas development team: pandas-dev/pandas: Pandas (Feb 2020)
22. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: Openml: Networked science in machine learning. SIGKDD Explorations **15**(2), 49–60 (2013)
23. Vladislavleva, E.Y., Smits, G.F., Den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation **13**(2), 333–349 (2010)
24. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic learning in a random world, vol. 29. Springer (2005)

A Additional materials

A.1 Results of the experimental run on OpenML-CTR23

OpenML-CTR23 [10] is a benchmark suite of hand-picked data sets from OpenML, chosen for several desirable characteristics. All data sets have been selected according to desirable properties, such as: containing between 500 and 100,000 samples and less than 5,000 features; having a referenced source; containing no artificial data; presenting a regression task is not trivially solvable by a linear model, or in other words, a linear regression algorithm cannot attain a $R2 = 1.00$ in a cross-validation; and so on.

We performed a trial run on the data sets included in OpenML-CTR23, to (i) compare the performance of RF [4] against XGBoost [6], which is commonly considered the state of the art for regression on tabular data; (ii) assess whether the proposed split (50% training, 25% calibration, 25% test) can still deliver good predictions; (iii) identify data sets for which the computation of confidence intervals does not make sense, as the performance of RF is nearly perfect ($R2 > 0.95$) or extremely poor ($R2 < 0.40$), using two arbitrary thresholds. The complete results are reported in Table 3.

A.2 Full experimental results

Table 4 shows the detailed performance of each method on each data set, for coverage and median width of the confidence intervals. Interestingly, $SRCP$ often delivers the tightest confidence intervals, with the lowest empirical coverage on the test set.

A.3 Pareto fronts

As we are employing a multi-objective approach (evaluating coverage and median width of the confidence intervals) to compare the different CP methods, each run on a different data set produces a Pareto front. The Pareto fronts are reported in Figures 3-5.

Table 3. Data sets in OpenML-CTR23, with mean performance of Random Forest and XGBoost regressors in a 10-fold cross-validation. Rows highlighted in grey represent data sets that have been discarded from the experimental evaluation, due to poor ($R2 < 0.40$) or exceptional ($R2 > 0.95$) performance.

Task ID	**Data set name**	**Samples**	**Features**	**Missing values**	**Categorical features**	**RF R2 (test) mean +/- std**	**XGB R2 (test) mean +/- std**
361234	abalone	4,177	8	0	1	0.53 +/- 0.02	0.47 +/- 0.02
361235	airfoil_self_noise	1,503	5	0	0	0.90 +/- 0.01	0.96 +/- 0.01
361236	auction_verification	2,043	7	0	2	0.99 +/- 0.00	1.00 +/- 0.00
361267	brazilian_houses	10,692	9	0	4	0.52 +/- 0.24	0.59 +/- 0.36
361255	california_housing	20,640	8	0	0	0.81 +/- 0.00	0.83 +/- 0.01
361622	cars	804	17	0	0	0.94 +/- 0.01	0.94 +/- 0.02
361237	concrete_compressive_strength	1,030	8	0	0	0.88 +/- 0.01	0.94 +/- 0.02
361261	cps88wages	28,155	6	0	4	0.15 +/- 0.02	0.21 +/- 0.04
361256	cpu_activity	8,192	21	0	0	0.98 +/- 0.00	0.98 +/- 0.01
361257	diamonds	53,940	9	0	3	0.98 +/- 0.00	0.98 +/- 0.00
361617	energy_efficiency	768	8	0	0	1.00 +/- 0.00	1.00 +/- 0.00
361272	fifa	19,178	28	0	1	0.77 +/- 0.01	0.75 +/- 0.04
361618	forest_fires	517	12	0	2	-0.95 +/- 1.26	-10.35 +/- 17.06
361268	fps_benchmark	24,624	39	69,696	13	0.99 +/- 0.00	1.00 +/- 0.00
361243	geographical_origin_of_music	1,059	116	0	0	0.23 +/- 0.02	0.22 +/- 0.09
361251	grid_stability	10,000	12	0	0	0.89 +/- 0.00	0.93 +/- 0.01
361269	health_insurance	22,272	11	0	7	0.32 +/- 0.01	0.31 +/- 0.02
361258	kin8nm	8,192	8	0	0	0.68 +/- 0.01	0.78 +/- 0.01
361266	kings_county	21,613	21	0	4	0.87 +/- 0.01	0.88 +/- 0.04
361260	miami_housing	13,932	15	0	0	0.90 +/- 0.00	0.91 +/- 0.01
361616	Moneyball	1,232	10	3,600	4	0.92 +/- 0.00	0.93 +/- 0.00
361247	naval_propulsion_plant	11,934	14	0	0	0.99 +/- 0.00	1.00 +/- 0.00
361241	physiochemical_protein	45,730	9	0	0	0.64 +/- 0.00	0.63 +/- 0.01
361259	pumadyn32nh	8,192	32	0	0	0.64 +/- 0.00	0.58 +/- 0.03
361621	QSAR_fish_toxicity	908	6	0	0	0.61 +/- 0.03	0.58 +/- 0.08
361250	red_wine	1,599	11	0	0	0.44 +/- 0.02	0.44 +/- 0.09
361254	sarcos	48,933	21	0	0	0.97 +/- 0.00	0.98 +/- 0.00
361264	socmob	1,156	5	0	4	0.75 +/- 0.03	0.78 +/- 0.21
361244	solar_flare	1,066	10	0	8	-0.09 +/- 0.12	-0.42 +/- 0.38
361623	space_ga	3,107	6	0	0	0.63 +/- 0.01	0.69 +/- 0.04
361619	student_performance_por	649	30	0	17	0.23 +/- 0.08	0.17 +/- 0.19
361242	superconductivity	21,263	81	0	0	0.91 +/- 0.00	0.92 +/- 0.01
361252	video_transcoding	68,784	18	0	2	0.98 +/- 0.00	0.99 +/- 0.00
361253	wave_energy	72,000	48	0	0	0.83 +/- 0.00	0.97 +/- 0.00
361249	white_wine	4,898	11	0	0	0.46 +/- 0.01	0.50 +/- 0.05

Table 4. Detailed results of the experimental run for each data set, reporting values of coverage and median width of the confidence intervals for each method. Cells in green show the best value obtained in the row, cells in magenta show the worst value.

Data set name	**Metric**	SCP	NCP_d	NCP_{std}	NCP_{oob}	NCP_{var}	MCP	$SRCP$
`abalone`	Coverage	0.9474	0.9502	0.9512	0.9455	0.9311	0.9483	0.9522
	Median	2.9545	3.2021	3.6647	3.3293	2.7972	2.6755	2.6284
`airfoil_self_noise`	Coverage	0.9362	0.9388	0.9468	0.9441	0.8989	0.9282	0.9441
	Median	1.1607	1.4426	1.6024	1.5573	1.0324	1.1157	0.9472
`brazilian_houses`	Coverage	0.9577	0.9562	0.9581	0.9540	0.9600	0.9727	0.9633
	Median	0.4947	0.4813	0.3420	0.2937	0.4858	0.3453	0.2594
`california_housing`	Coverage	0.9531	0.9547	0.9521	0.9514	0.9554	0.9663	0.9525
	Median	1.9542	2.3276	1.8447	1.6705	1.4844	1.7037	1.3536
`cars`	Coverage	0.9303	0.9751	0.9502	0.9453	0.9701	0.9602	0.9254
	Median	1.0866	5.2368	1.2290	0.7567	0.9936	1.0866	0.6614
`concrete_compressive_strength`	Coverage	0.9264	0.9574	0.9535	0.9651	0.9806	0.9302	0.9147
	Median	1.2917	2.0088	1.7034	1.6180	1.6667	1.2881	0.9751
`fifa`	Coverage	0.9545	0.9495	0.9395	0.9395	0.9452	0.9568	0.9395
	Median	1.8810	1.6280	0.5688	0.5400	0.5754	0.5692	0.5399
`grid_stability`	Coverage	0.9552	0.9424	0.9432	0.9560	0.9360	0.9520	0.9540
	Median	1.3653	1.5484	1.4467	1.3390	1.4780	1.3759	1.2266
`health_insurance`	Coverage	0.9481	0.9582	0.9510	0.9515	0.9542	0.9623	0.9481
	Median	3.3151	5.7595	3.6869	4.2658	5.0377	3.6209	3.2132
`kin8nm`	Coverage	0.9482	0.9526	0.9424	0.9507	0.9458	0.9463	0.9551
	Median	2.1733	2.7016	2.3705	2.0131	2.3174	2.1190	2.0097
`kings_county`	Coverage	0.9524	0.9556	0.9617	0.9563	0.9574	0.9684	0.9552
	Median	1.4712	1.1274	1.0826	0.9722	0.7327	0.9059	0.9615
`miami_housing`	Coverage	0.9526	0.9518	0.9475	0.9500	0.9388	0.9701	0.9440
	Median	1.1928	0.8502	0.5071	0.5355	0.3968	0.5779	0.5633
`Moneyball`	Coverage	0.9253	0.9805	0.9351	0.9448	0.9448	0.9448	0.9156
	Median	1.0988	3.3098	1.4842	1.4414	1.7583	1.1847	1.0084
`physiochemical_protein`	Coverage	0.9484	0.9480	0.9508	0.9496	0.9493	0.9580	0.9514
	Median	2.6304	2.4417	2.9550	2.3881	1.9352	2.3144	2.2510
`pumadyn32nh`	Coverage	0.9468	0.9458	0.9580	0.9639	0.9526	0.9526	0.9448
	Median	2.3860	2.7065	2.8763	3.0634	2.9467	2.5839	2.3213
`QSAR_fish_toxicity`	Coverage	0.9868	0.9604	0.9824	0.9648	0.9559	0.9780	0.9207
	Median	3.4230	3.2027	3.4439	4.5222	3.5589	3.7275	1.9548
`red_wine`	Coverage	0.9550	0.9625	0.9550	0.9525	0.9575	0.9600	0.9500
	Median	3.4234	4.5759	3.2935	3.4862	3.5041	3.7128	3.0421
`socmob`	Coverage	0.9412	0.9585	0.9550	0.9377	0.9550	0.9550	0.9550
	Median	1.5951	1.5285	0.5835	0.5806	0.3293	0.7478	0.3265
`space_ga`	Coverage	0.9614	0.9575	0.9601	0.9678	0.9511	0.9665	0.9562
	Median	2.4853	3.5980	2.9231	3.0588	2.8941	2.4853	2.0549
`superconductivity`	Coverage	0.9528	0.9492	0.9464	0.9481	0.9466	0.9626	0.9518
	Median	1.3232	4.9142	0.7834	0.7384	0.5328	0.9699	0.9111
`wave_energy`	Coverage	0.9529	0.9479	0.9489	0.9519	0.9516	0.9531	0.9535
	Median	1.7184	2.9430	1.9980	1.6872	1.8567	1.6691	1.5973
`white_wine`	Coverage	0.9592	0.9584	0.9461	0.9502	0.9665	0.9608	0.9600
	Median	2.9805	3.2859	3.6857	3.2506	2.9687	3.1553	2.7576

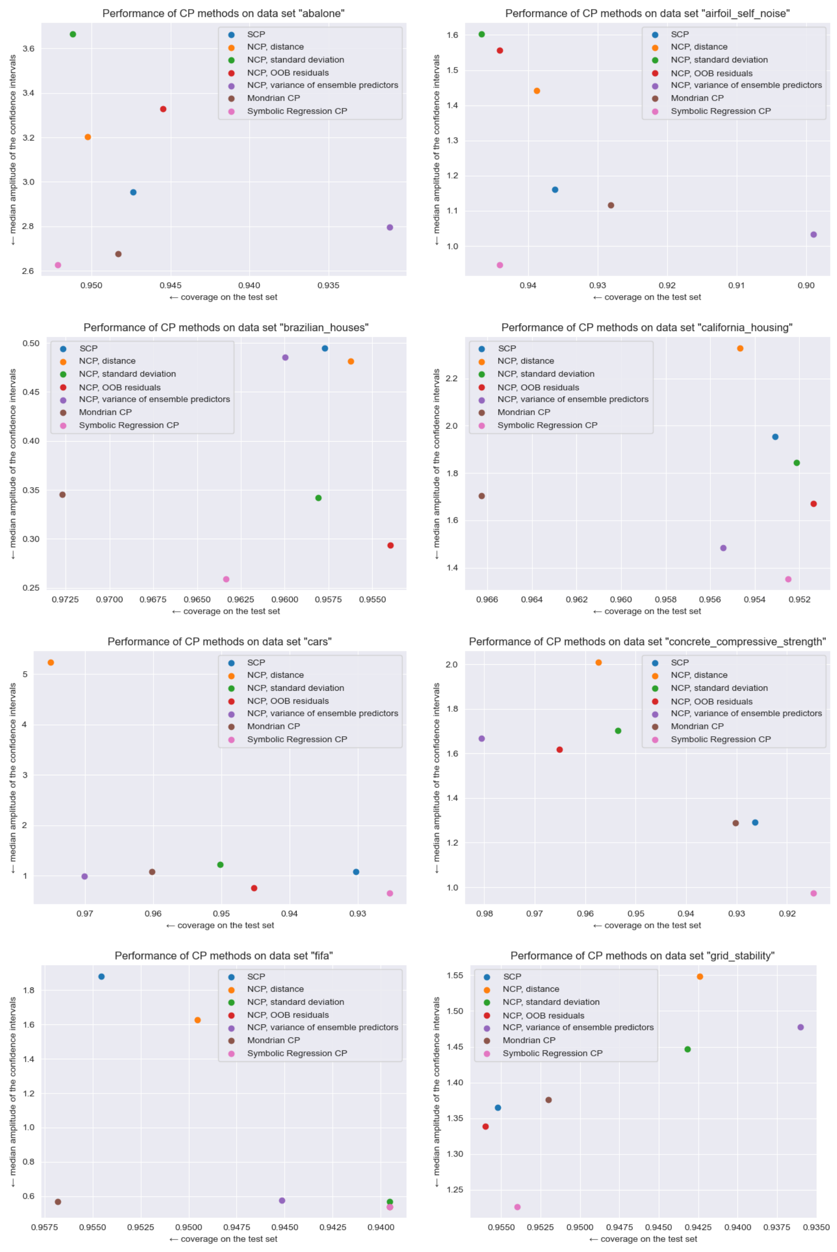

Fig. 3. Pareto fronts obtained for the different data sets (1).

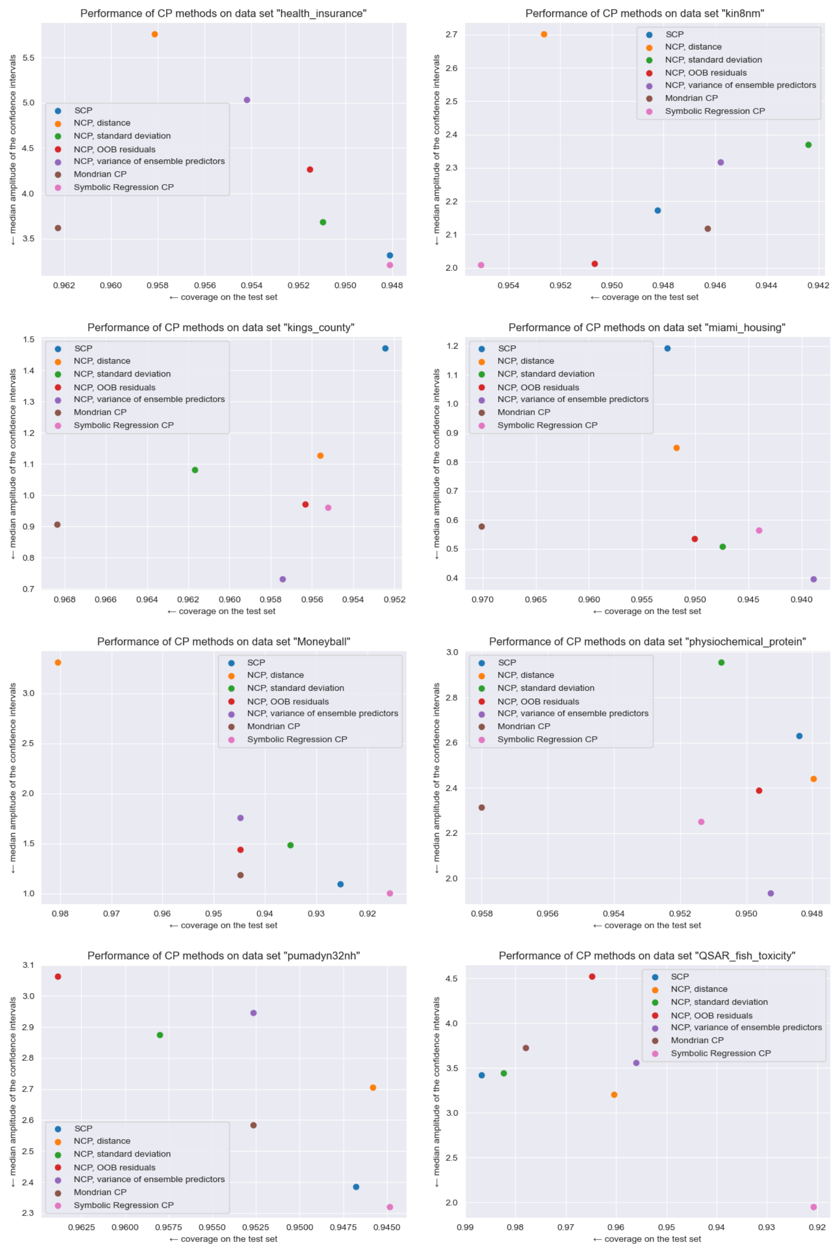

Fig. 4. Pareto fronts obtained for the different data sets (2).

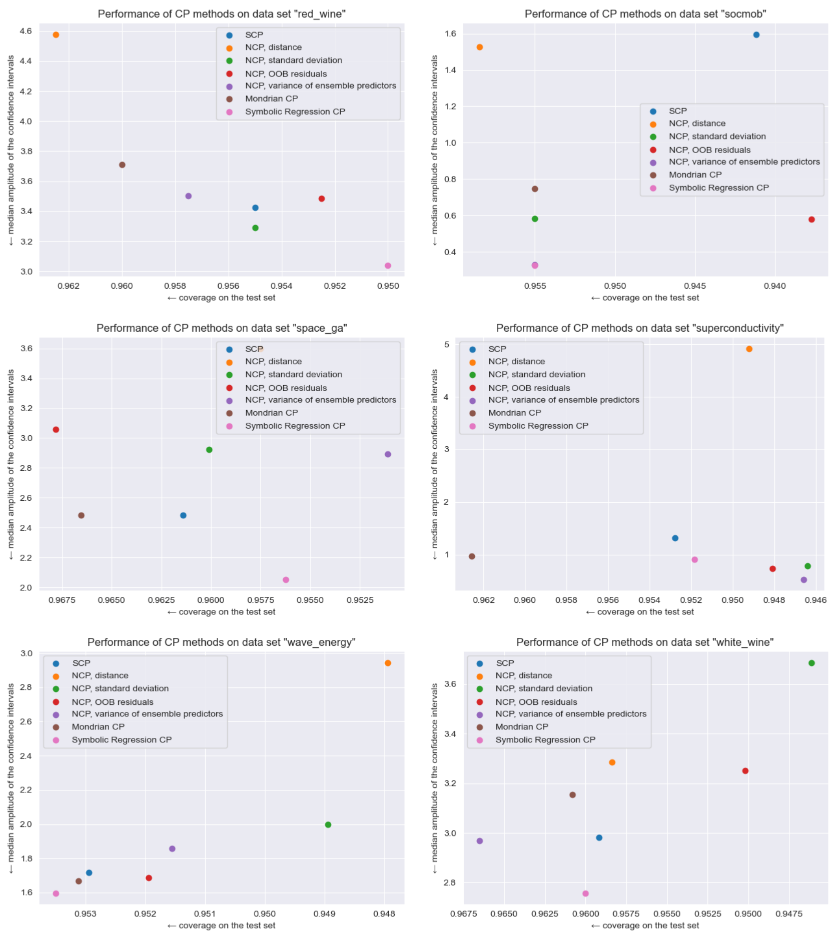

Fig. 5. Pareto fronts obtained for the different data sets (3).