Tutorial on using Conformal Predictive Systems in KNIME

Tuwe Löfström* TUWE.LOFSTROM@JU.SE
Jönköping AI Lab, Department of Computing, Jönköping University, Sweden
Alexander Bondaletov ALEXANDER.BONDALETOV@REDFIELD.SE
Redfield AB, Sweden
Artem Ryasik ARTEM.RYASIK@REDFIELD.SE
Redfield AB, Sweden
Henrik Boström BOSTROMH@KTH.SE
School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden
Ulf Johansson ULF.JOHANSSON@JU.SE
Jönköping AI Lab, Department of Computing, Jönköping University, Sweden

Editor: Harris Papadopoulos, Khuong An Nguyen, Henrik Boström and Lars Carlsson

Abstract

KNIME is an end-to-end software platform for data science with an open-source analytics platform for creating solutions and a commercial server solution for productionization. Conformal classification and regression have previously been implemented in KNIME. We extend the conformal prediction package with added support for conformal predictive systems, taking inspiration from the interface of the Crepes package in Python. The paper demonstrates some typical use cases for conformal predictive systems. Furthermore, the paper also illustrates how to create Mondrian conformal predictors using the KNIME implementation. All examples are publicly available, and the package is available through KNIME's official software repositories.

Keywords: Conformal predictive systems, software implementation, KNIME Analytics Platform, Mondrian conformal predictive systems

1. Introduction

Machine learning and data science have become increasingly popular in recent years, with techniques such as random forests and deep learning now widely available through libraries and tools. This trend has led to a proliferation of machine learning applications across various industries, influencing decision-making and work processes wherever it is applied. One approach to building data science workflows is using tools that allow users to construct workflows using nodes. Each node represents a specific behavior, such as data manipulation, modeling, or visualization. These workflows can be created using visual programming, enabling the creation of no-code data science pipelines without any coding knowledge. Popular tools for creating these workflows include RapidMiner, WEKA, Orange, and KNIME. Although these tools vary in interactivity, openness, and ease of use, their goal is to make data analysis more accessible to novice users.

In this paper, we introduce an extension of the Conformal Prediction package (Löfström et al., 2022) for KNIME[1] (Berthold et al., 2009), implementing conformal predictive systems.

* Corresponding author.
1. knime.com

Furthermore, the paper also illustrates how Mondrian conformal prediction can be achieved using the current implementation.

Conformal prediction (Vovk et al., 2005a) is a powerful framework that provides accurate levels of confidence in predictions. Despite attracting growing scientific interest in recent years, it has not yet gained widespread recognition outside of academia as a valuable tool for improving the quality of decision-support systems based on predictions. Therefore, it is crucial to make this framework more easily accessible to a broader audience, regardless of their programming knowledge, to expand its reach and impact.

A conformal prediction toolbox for KNIME is available through KNIME's official software channels as a community package. It was originally developed for conformal classification by Redfield AB (Ryasik and Landrum, 2020; Ryasik, 2023b) and later extended to include support for conformal regression (Löfström et al., 2022; Ryasik, 2023a). In this paper, we introduce an update and extension of the package to include conformal predictive systems.

2. KNIME Analytics Platform

KNIME is a data science platform developed by a university group at the University of Konstanz in 2004 and released as an open source tool in 2006[2]. Since then, it has grown into a comprehensive and open platform supported by an active user community. The platform provides extensive support for data blending, preprocessing, integration, and deployment, and it also includes built-in support for most standard machine-learning techniques. One of the key strengths of KNIME is its vast library of community content, which covers a wide range of application areas, with a particular emphasis on life science (Fillbrunn et al., 2017; Afantitis et al., 2020). This content includes advanced techniques like deep learning and support for various data sources like images, text, and network mining.

3. Conformal Prediction

Conformal predictors are models that provide a level of confidence for each prediction they make. Specifically, given a test pattern x_i and a significance level ϵ, a conformal predictor generates a prediction region Γ_i^ϵ, which is a set of values that includes the true target y_i with probability $1 - \epsilon$. In classification problems, the prediction regions are sets of class labels, while in regression problems, they are prediction intervals.

Errors occur in conformal prediction when the true target falls outside of the prediction region. However, conformal predictors are automatically valid, meaning that under the exchangeability assumption, the error rate will be exactly ϵ over the long run. Therefore, the key criterion for evaluating conformal predictors is their efficiency, which refers to the size of the prediction regions. In addition to being small, prediction regions should also be individualized or sharp to increase their informativeness. For regression problems, this means that the prediction intervals should be as tight as possible for each test instance.

2. KNIME: How it all began.

3.1. Conformal Regression

Constructing a conformal predictor for regression, known as an *inductive (split) conformal regressor*, involves the following steps:

1. Divide the training set Z into two disjoint subsets, resulting in a *proper training set* Z_t and a *calibration set* Z_c, where $|Z_c| = q$.
2. Using Z_t, fit a regression model h to be used as the *underlying model*.
3. Let the absolute error be defined as the nonconformity function:
$$f(z_i) = |y_i - h(x_i)|\,. \tag{1}$$
4. Apply the nonconformity function on all calibration instances to get nonconformity scores $f(z)$ for all $z_i \in Z_c$. Sort them in descending order to obtain $\alpha_1 \leq \alpha_2 \leq ... \leq \alpha_q$.
5. Assign a significance level $\epsilon \in (0, 1)$, where typical choices include $\epsilon = 0.01$, $\epsilon = 0.05$, or $\epsilon = 0.1$.
6. The index of the $(1-\epsilon)$-percentile nonconformity score, α_s, is defined as $s = \lfloor \epsilon(q + 1) \rfloor$.
7. To determine the prediction interval for a new instance x_i the following equation is used:
$$\Gamma_i^\epsilon = h(x_i) \pm \alpha_s, \tag{2}$$
where the interval is centered around $h(x_i)$ and has a width of $2\alpha_s$. The size of this interval is based on the observation that the probability of a test instance having a larger absolute error than α_s, under the exchangeability assumption, must be ϵ. Therefore, the interval contains y_i with confidence $1 - \epsilon$.

Other nonconformity functions for a specific underlying regression model could be used to define a different conformal regressor. However, even if all of these conformal regressors will be valid, they may result in significant differences in terms of efficiency. In the procedure described above, using $f(z_i) = |y_i - h(x_i)|$ and $\Gamma_i^\epsilon = h(x_i) \pm \alpha_s$, the size of all prediction interval will be $2\alpha_s$.

To obtain individual bounds for each x_i, *normalized nonconformity function* can be used. By incorporating the additional terms σ_i and β into the basic nonconformity function, normalized conformal regression achieve increased informativeness. These terms adjust the prediction interval based on the estimated difficulty (represented by σ_i) of predicting the target variable y_i. Intuitively, the quality (or difficulty) estimate σ_i represents our confidence in the prediction for y_i given x_i. The normalized nonconformity function is defined as:

$$f(z_i) = \frac{|y_i - h(x_i)|}{\sigma_i + \beta} \tag{3}$$

where β controls the sensitivity of the normalization term, and determines the relative importance of σ_i in determining the size of the prediction interval. The normalized prediction interval for a new example is given by:

$$\Gamma_i^\epsilon = h(x_i) \pm \alpha_s(\sigma_i + \beta). \tag{4}$$

By using a normalized nonconformity function, we can obtain individualized prediction intervals that adjust to the specific difficulty of predicting y_i for each input x_i, potentially leading to smaller and more informative prediction regions.

3.2. Conformal Predictive Systems

Conformal predictive systems (Vovk et al., 2019) are an extension of conformal regressors that produce *conformal predictive distributions*, which are cumulative distribution functions. These distributions can be used for various purposes, such as deriving prediction intervals for specified confidence levels or obtaining threshold values based on the probability of the true target falling below or above a certain level. For example, the probability that an ordered spare part is delayed beyond the safety buffer. Since prediction intervals in conformal regressors do not provide any information about how values within and outside the intervals are distributed, these capabilities are important additions.

Efficiently generating (normalized) inductive conformal predictive systems is similar to how inductive conformal regressors are formed. The most important difference is that the nonconformity scores are calculated by considering actual and not absolute errors:

$$f\left(z_i\right) = y_i - h\left(x_i\right), \tag{5}$$

or, when using normalization:

$$f\left(z_i\right) = \frac{y_i - h\left(x_i\right)}{\sigma_i + \beta}, \tag{6}$$

where σ_i, x_i, and β are defined as before. The prediction for a test instance x_i (potentially with an estimated difficulty σ_i) then becomes the following cumulative distribution function (conformal predictive distribution):

$$Q(y) = \begin{cases} \frac{n+\tau}{q+1}, \text{ if } y \in \left(C_{(n)}, C_{(n+1)}\right), & \text{for } n \in \{0, ..., q\} \\ \frac{n'-1+(n''-n'+2)\tau}{q+1}, \text{if } y = C_{(n)}, & \text{for } n \in \{1, ..., q\} \end{cases} \tag{7}$$

where $C_{(1)}, \ldots, C_{(q)}$ are obtained from the calibration scores $\alpha_1, \ldots, \alpha_q$, sorted in increasing order:

$$C_{(i)} = h\left(x\right) + \alpha_i$$

or, when using normalization:

$$C_{(i)} = h\left(x\right) + \sigma\alpha_i$$

with $C_{(0)} = -\infty$ and $C_{(q+1)} = \infty$. τ is sampled from the uniform distribution $\mathcal{U}(0, 1)$ and its role is to allow the p values of target values to be uniformly distributed. n'' is the highest index such that $y = C_{(n'')}$, while n' is the lowest index such that $y = C_{(n')}$ (in case of ties). For a specific value y, the function returns the estimated probability $\mathcal{P}(Y \leq y)$, where Y is a random variable corresponding to the true target.

Given a conformal predictive distribution, a two-sided prediction interval for a chosen significance level ϵ can be obtained by $[C_{\lfloor(\epsilon/2)(q+1)\rfloor}, C_{\lceil(1-\epsilon/2)(q+1)\rceil}]$. One-sided prediction intervals can be obtained by $[C_{\lfloor\epsilon(q+1)\rfloor}, \infty]$ for a lower-bounded interval, and by $[-\infty, C_{\lceil(1-\epsilon)(q+1)\rceil}]$ for an upper-bounded interval. Similarly, a point prediction corresponding to the median of the distribution can be obtained by $C_{\lceil 0.5(q+1)\rceil}$.

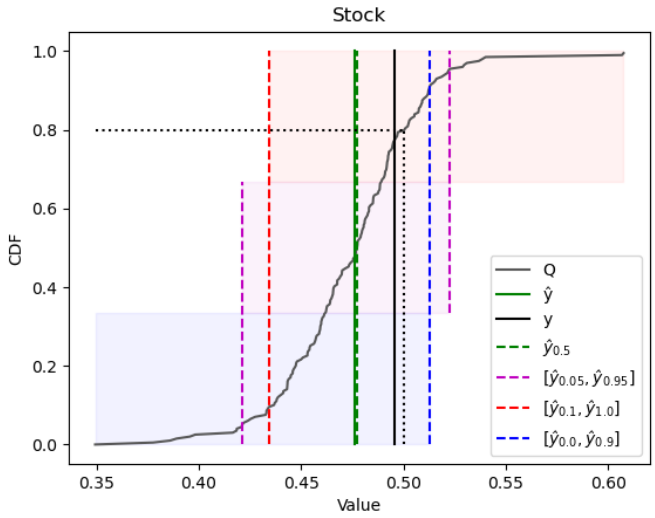


Figure 1: A conformal predictive distribution with three different intervals representing 90% confidence are defined: Lower-bounded interval: more than the 10^{th} percentile; Two-sided interval: between the 5^{th} and the 95^{th} percentiles; Upper-bounded interval: less than the 90^{th} percentile. The black dotted lines indicate how to determine the probability of the true target being smaller than 0.5, which in this case would be approximately 80%.

Figure 1 illustrates how the conformal predictive distribution can form one-sided and two-sided confidence intervals. It also illustrates how the probability of the true target falling behind a given threshold can be determined. Or, conversely, it illustrates what threshold a specific probability does correspond to.

3.3. Mondrian Conformal Predictive Systems

As pointed out by Boström et al. (2021), neither standard nor normalized predictive systems may be very effective tools for calibrating the predictions when the residuals (differences between actual and predicted values) are heteroscedastic, e.g., their distribution is not independent of the actual predictions. An example of such a model is an ensemble averaging the predictions from the base regressors, resulting in a tendency to overestimate low actual values and underestimate high actual values. The result will be that the residuals for low predicted values will be negative on average and positive on average for high predicted values. Since the distribution's median will always be on the same side relative to the underlying prediction, the distribution can suggest moving the predictions at most in one direction. As pointed out above, neither standard nor normalized conformal predictive systems can correct this, as the shape of standard predictive distributions will be the same for all predictions and σ always being positive, resulting in the direction of the normalization correction being the same as for the standard distribution.

Mondrian conformal predictive systems were proposed in (Boström et al., 2021). The idea was borrowed from Mondrian conformal prediction (Vovk et al., 2005b), originally proposed for allowing control of error levels of objects falling into *a priori* defined categories. The most typical use is to define the categories as class labels, ensuring the same error level for all classes. Mondrian conformal regressors were introduced in (Boström and Johansson, 2020) to handle two problems of normalized conformal regressors; 1) prediction intervals may be several times larger (or smaller) than the largest (or smallest) previously observed error, and 2) the sizes of the intervals become less uniform with less informative quality (difficulty) estimates. It was shown that categories formed by binning the quality estimate σ resulted in 1) intervals bounded by the size of the largest observed error and 2) non-informative quality estimates resulting in more uniformly sized intervals.

A solution for the ensemble example above would be to use two categories defined using the predictions of the underlying model, where one corresponds to low-valued predictions and the other corresponds to high-valued predictions. These two categories would enable the conformal predictive distribution to correct for systematic overestimations and underestimations. As the predictions from conformal regressors are centered around the point predictions and provide no information on whether the true target can be expected to be higher or lower than the point prediction, the same kind of correction can not be done for them. Still, defining Mondrian categories using the quality estimate σ may be used to form normalized conformal predictive distributions.

Figure 2 illustrates Mondrian conformal predictive systems by showing four instances from the four different bins defined using equal-frequency binning of the predictions from the underlying model. The bins are defined based on the predictions, with Bin 1 representing the lowest predictions and Bin 4 representing the highest predictions. As can be seen, in the (two) lowest bin(s), the median ($\hat{y}_{0.5}$) is clearly below the prediction, correcting for the bias of the random forest. Similarly, the median ($\hat{y}_{0.5}$) is above the prediction in the highest bin, again, making a correction of the bias of the random forest.

3.4. Adding Conformal Predictive Systems in KNIME

Redfield AB deployed the Conformal Prediction package in KNIME for class conditional conformal classification in 2020. In 2022, the package was extended to include conformal regression.

For the release of the updated Conformal Prediction package, functionality for conformal predictive systems has been added. The aim was to follow the same logic as used in the original implementation. Consequently, a calibrator, a predictor, and a classifier for conformal predictive systems have been added. Furthermore, a Predictive Systems Regression node has also been added, combining the functionality of the calibrator, predictor, and classifier nodes. Finally, the Conformal Scorer (Regression) was updated to allow more alternatives when evaluating regression results.

- Predictive Systems Calibrator (Regression) - Calculates the α-values for all calibration instances using Equation (5) for standard conformal predictive systems. If normalization is used, the difficulty, or σ, and β are also used in calculating α, according to Equation (6).

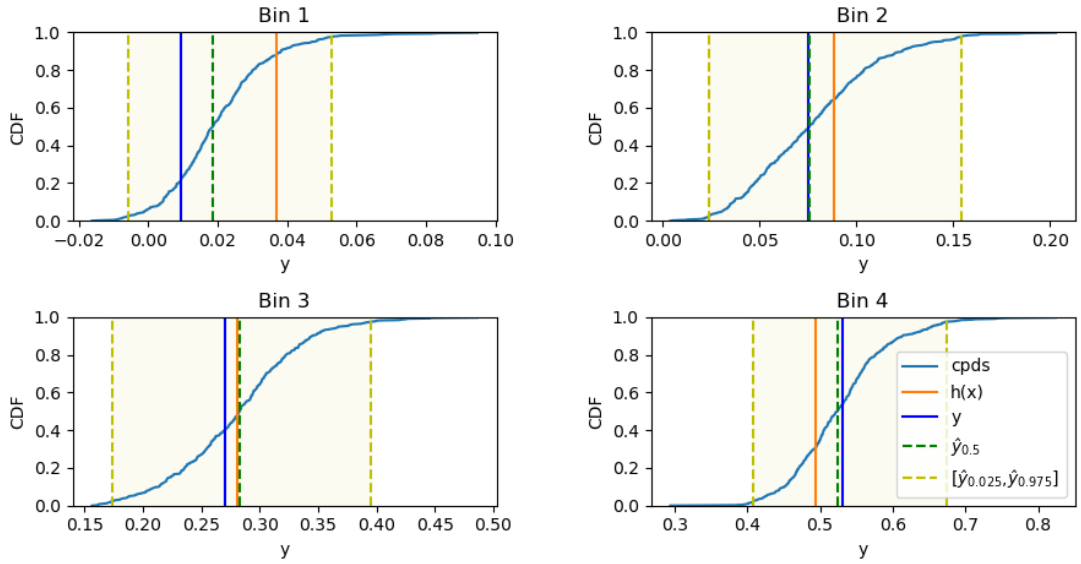


Figure 2: Four Mondrian conformal predictive distributions from four different bins.

- Predictive Systems Predictor (Regression) - The node takes the calibration table and the test data and defines the conformal predictive distribution using Equation (7). If normalization was used for calibration, the same setting as in the Predictive Systems Calibrator (Regression) must also be used in this node.

- Predictive Systems Classifier (Regression) - This node is used to define how to query the conformal predictive distribution. Following the logic used in the crepes[3] package in Python (Boström, 2022), the node takes four types of input parameters:

  - Target value: A fixed threshold for the target value v. When this parameter is used, the node will output a column containing $\mathcal{P}(Y \leq v)$, i.e., the estimated probability that the predicted value is below v.
  - Target Column: A column containing individual thresholds v_i for each instance. When this parameter is used, the node will output a column containing $\mathcal{P}(Y \leq v_i)$, i.e., the estimated probability that the predicted value is below v_i.
  - Lower Percentiles (%): A dynamic number of user-defined lower percentiles. For each lower percentile defined, the node will output a column with the values corresponding to that percentile in the conformal predictive distribution.
  - Upper Percentiles (%): A dynamic number of user-defined upper percentiles. For each upper percentile defined, the node will output a column with the values corresponding to that percentile in the conformal predictive distribution.

3. https://github.com/henrikbostrom/crepes

- Predictive Systems Regression - This node combines the functionality in the three nodes above with the sum of the parameters available in the individual nodes. The purpose of this node is to create an ease-of-use node for regular use cases.

- Conformal Scorer (Regression) - This node has been updated to allow evaluation of both two-sided and one-sided intervals. It compares prediction intervals with actual values and calculates metrics for estimating conformal predictions.

As explained in (Löfström et al., 2022), the first three nodes are intended to be used within conformal loops, enabling multiple training models and calculating a set of CDFs. Afterwards, the CDFs are aggregated with the median function. This approach helps to increase the robustness of the predictions, however, it requires more effort in building the workflow and execution time due to repetitive training and calculating CDFs.

4. Use cases

To illustrate how conformal predictive systems can be applied in KNIME, a number of use cases, from simple to more complex, will be presented here. All examples and experiments presented below are available for download at KNIME Hub[4].

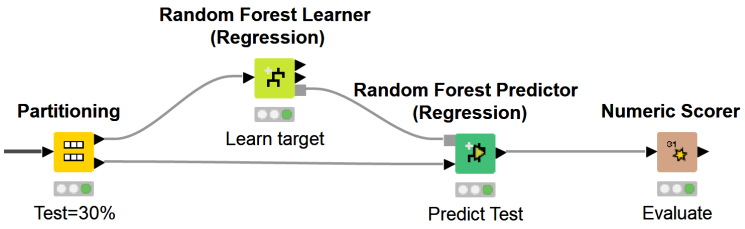


Figure 3: Regression without conformal prediction

Figure 3 illustrate how to train and evaluate a machine learning model in KNIME. The data set is divided into a training and a test set using a Partitioning node. A model is trained using a Learner node applied to the training set and the prediction is achieved by applying a Predictor node to the test set. Finally, evaluation is done using a Scorer node.

All of the steps described above are included when applying conformal predictive systems, but some additional steps also need to be done. As before, the data must first be divided into training and test sets. Since inductive conformal predictive systems also require a calibration set, the training set is divided into a proper training set used to train the model and a calibration set used for calibration. As before, the model is trained using a Learner node applied to the proper training set. Since calibration and test sets must always be exchangeable, they must always be handled similarly, so both sets are predicted using Predictor nodes. The predictions from both sets are fed into the Predictive Systems Regression node to get the outputs based on the user-defined parameters for Target value(s) and

4. KNIME Community Hub > tuwelofstrom > Spaces > Predicting with Confidence > COPA 2023 - Conformal Predictive Systems

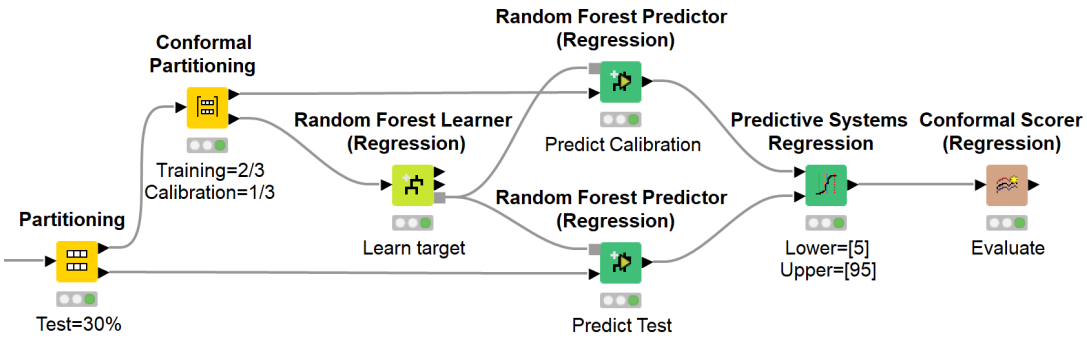


Figure 4: Conformal predictive systems with scoring using partitioning

lower and upper percentiles. Finally, if lower and/or upper percentiles are defined, the intervals they form can be evaluated using a Conformal Scorer (Regression) node. Regardless of whether the data set is simply divided into a training and test set using a Partitioning node, see Figure 4, or whether Cross-Validation is used, see Figure 5, the evaluation is done once at the end, on the entire evaluated data. In order to use normalized conformal predictive

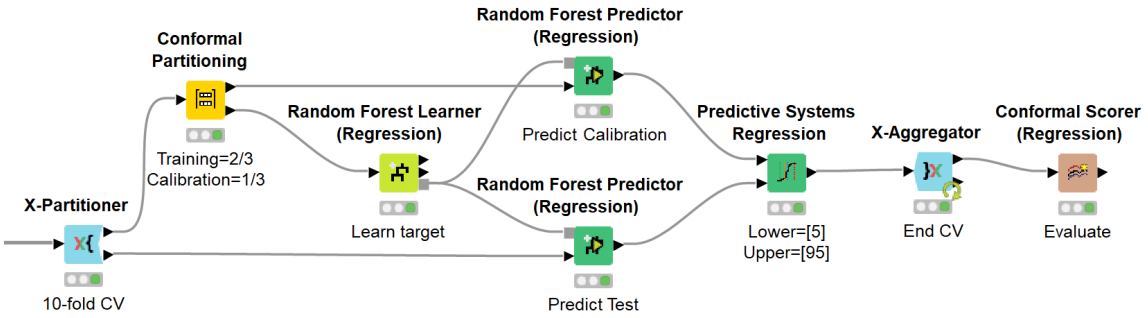
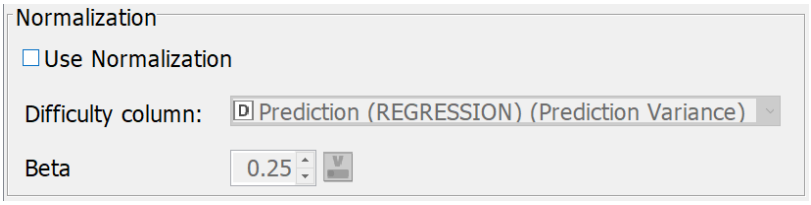


Figure 5: Conformal predictive systems using cross validation

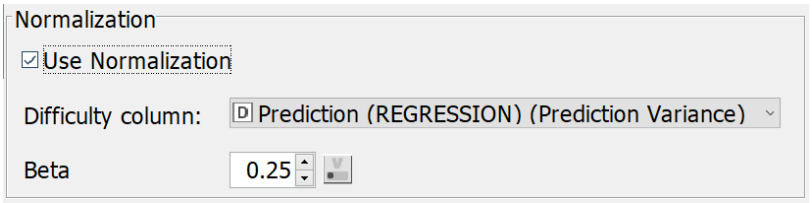
systems, an additional column of data representing σ, i.e., how difficult an instance is, must exist in the data. If such a column exists, changing from standard conformal predictive systems to normalized conformal predictive systems requires only opting for normalization, selecting the σ-column, and assigning a β-value. If using a random forest as the underlying model, the prediction variance, suitable to use as σ, will automatically be added as a separate column along with the prediction.

Consequently, by only changing the Normalization options, without altering the workflow, standard conformal predictive systems (using the configuration in Figure 6(a)) will change into normalized conformal predictive systems using prediction variance from the random forest (using the configuration in Figure 6(b)).

However, if your underlying model does not automatically provide you with a difficulty estimate, as is the case for most techniques, a difficulty estimation must be defined separately somehow. One way of doing that is to train a separate model on the absolute error



(a) Standard conformal predictive systems without Normalization



(b) Normalized conformal predictive systems with prediction variance as σ

Figure 6: Setting up conformal predictive systems

of the first model. An example of how this is done for conformal regression was given in (Löfström et al., 2022) with a workflow available at KNIME Hub[5]. The exact same procedure works for conformal predictive systems as well.

Looking at the conformal predictive systems on an individual data set, the Stock data set in this case, Figure 7 shows the results using four different line plots for a standard conformal predictive system with two intervals representing significance levels $\epsilon \in \{0.01, 0.2\}$. These two intervals are formed using the 0.5^{th} lower and 99.5^{th} upper percentiles for $\epsilon = 0.01$ and the 10^{th} lower and 90^{th} upper percentiles for $\epsilon = 0.2$. The two subplots to the left are sorted on target values, and the two subplots to the right are sorted on prediction values, here represented by the 50^{th} percentile, i.e., the median. The plot shows the result for standard conformal predictive systems, where the prediction intervals are equally wide for all instances. That the intervals are equally wide is most clearly seen in the two subplots to the right, since they are sorted on the prediction values. The two top plots compare the median and the two intervals defined using the conformal predictive systems to the true target (in black).

The same set of plots for normalized conformal predictive systems using prediction variance as difficulty estimate can be seen in Figure 8. As can be seen, the prediction intervals are clearly affected by the difficulty estimate used, with some intervals being narrow and others being wide. The interval's width can be used to estimate an instance's difficulty. β was set to 0.005, to create the visual effect of intervals with a great difference. Smaller β-values increase the impact of σ. Generally, to get a similar impact of σ in conformal predictive systems compared to conformal regression, smaller β-values will be needed.

To illustrate how the target value and target column parameters can be used, two additional examples are introduced. In the first example, the target value parameter is set to 0.5, resulting in an output column named $P(cpds < 0.5)$. Figure 9 illustrates how $P(cpds < 0.5)$ is related to the median.

5. See the COPA 2022 - Redfield & JU - Conformal Regression Experiment workflow.

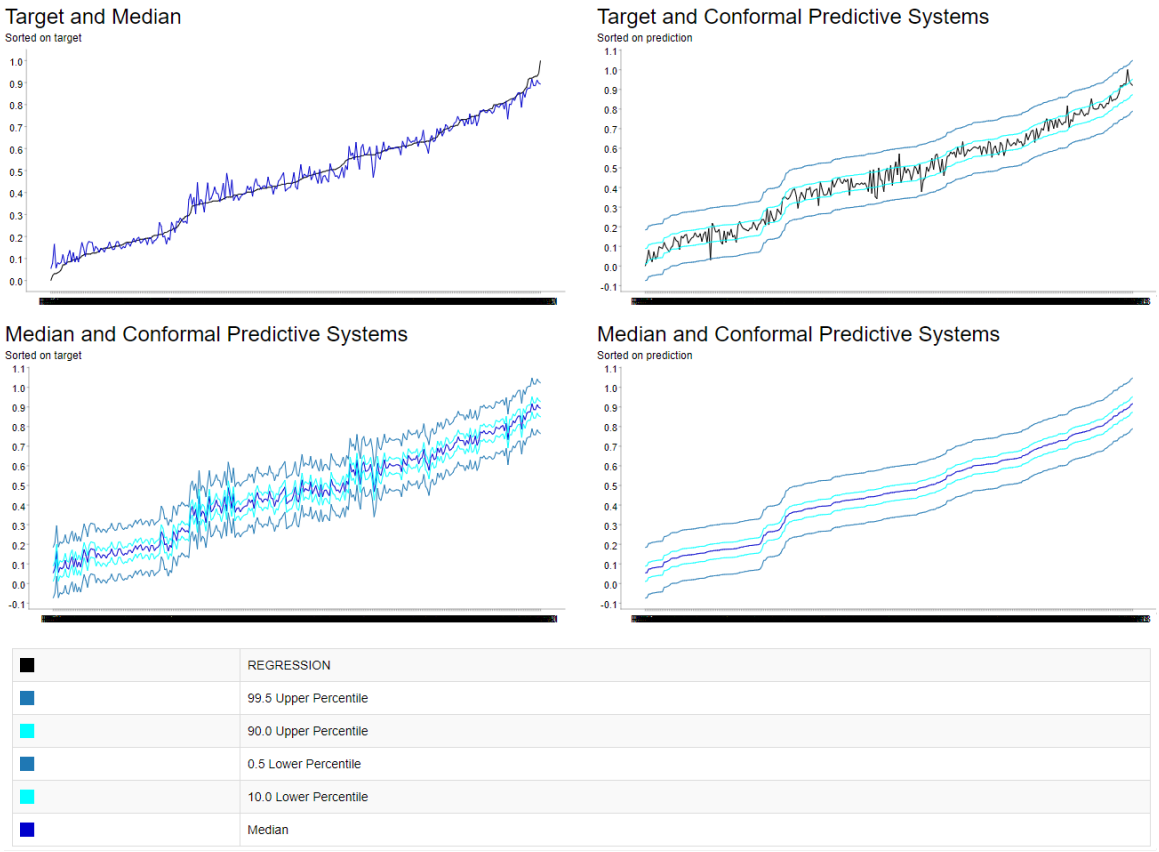


Figure 7: Standard conformal predictive systems without normalization

In practice, the $P(cpds < 0.5)$ represents the proportion of the conformal predictive distribution below 0.5 for a particular instance. For this particular data set, with target values normalized to the interval $[0, 1]$, the following can be seen:

- For instances with median values ≤ 0.4, the probabilities are $P(cpds < 0.5) \approx 1.0$.
- For instances with median values ≥ 0.6, the probabilities are $P(cpds < 0.5) \approx 0.0$;
- For instances with median values ≈ 0.5, $P(cpds < 0.5) \approx 0.5$.

Figure 10 illustrate the result of assigning the column containing target values (REGRESSION) in the target column parameter. In the figure, the data is sorted on the $P(cpds <$ [REGRESSION]) column, i.e., on the probability of the true target being below the prediction. In practice, this corresponds to the percentiles corresponding to the position of the target value in the conformal predictive distribution. Naturally, sorting based on $P(cpds <$ [REGRESSION]) will produce a similar ordering as if sorting on the residuals.

Obviously, using the target column in that way will not be possible when evaluating new data, since the target values will not be known. However, these two examples serve to illustrate how the target value parameter works. The target column parameter is identical to

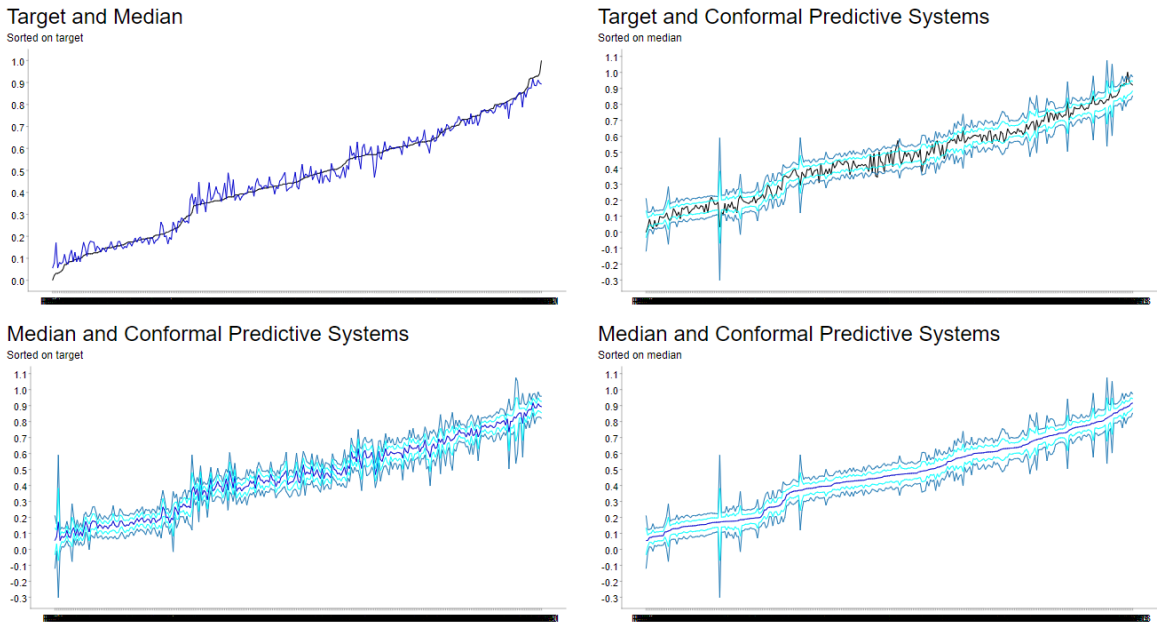


Figure 8: Normalized conformal predictive systems using the prediction variance as difficulty estimate

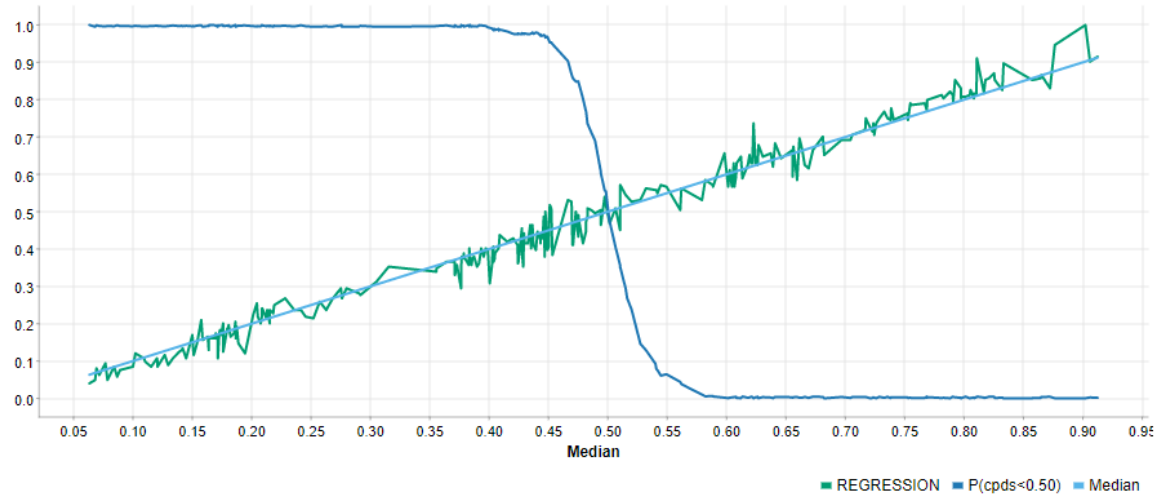


Figure 9: The target (REGRESSION), median and $P(cpds < 0.5)$, sorted on median

the target value parameter except that each instance gets its (potentially unique) threshold from the column assigned. A typical use case for the target value parameter is when a fixed threshold exists for which it is important to identify if a prediction is at risk of falling below or above. Analogously, if the threshold depends on some outer factor, e.g.,

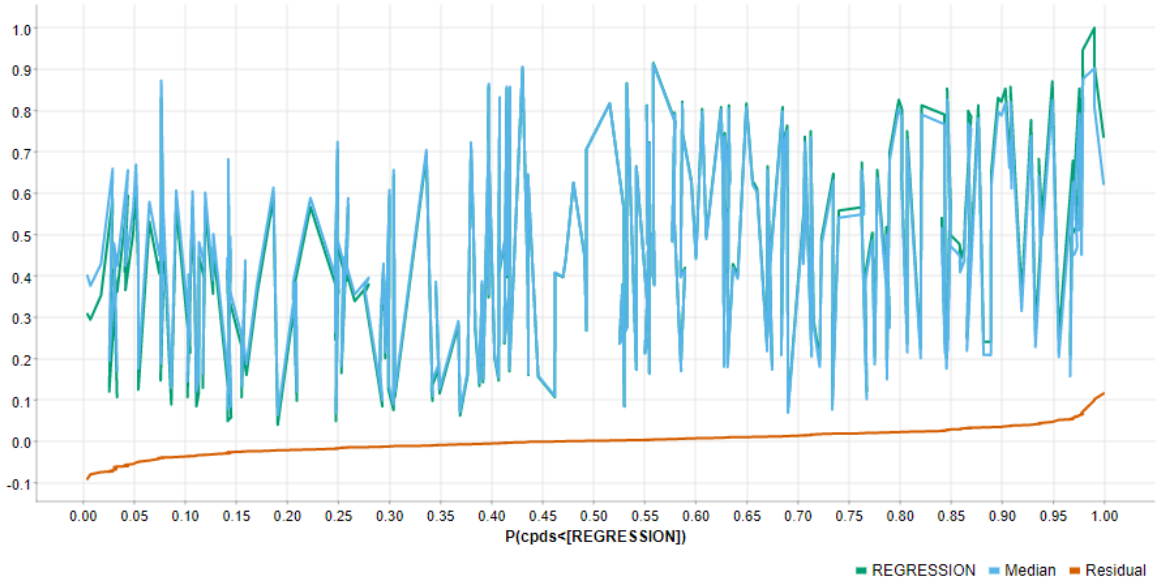


Figure 10: The target (REGRESSION), median and residual, sorted on the probability of the true target being below the prediction.

time, temperature, or speed, the target column parameter can be used to define dynamic thresholds instead.

Figure 11 introduces how Mondrian conformal prediction can be achieved in KNIME. The functionality to support Mondrian solutions have not been implemented into the nodes, but can easily be achieved by using a few extra nodes. Since all Mondrian solutions are based on dividing the calibration sets into subgroups based on some category, the first step is to decide the categories. When the categories are based on a numerical feature, the values must be binned somehow. In KNIME, the Auto-Binner node can automatically bin data based on parameters like deciding between percentile-based binning or by a given number of bins combined with a choice between equal-width or equal-frequency binning. In this case, the binning is performed on the prediction from the underlying model using four bins and equal-frequency binning. Once the categories (the result from the binning) are defined, one conformal predictive system is fitted and applied for each set of instances covered by the category. In KNIME, this can be done using a Group Loop Start node, looping over groups of calibration instances, conditioned on the categories. The current category from the loop is used in a Reference Row Filter node to include test instances covered by the same category. The Predictive Systems Regression node is used just like before. Finally, a Loop End node wraps up the procedure. This example shows that applying Mondrian categories to conformal predictive systems (or conformal prediction) is a matter of dividing instances based on the categories and applying conformal predictive systems (or conformal prediction) to one category at a time. Consequently, if someone wants to use another categorization (like defining the categories based on the quality estimate σ), it is just as easy.

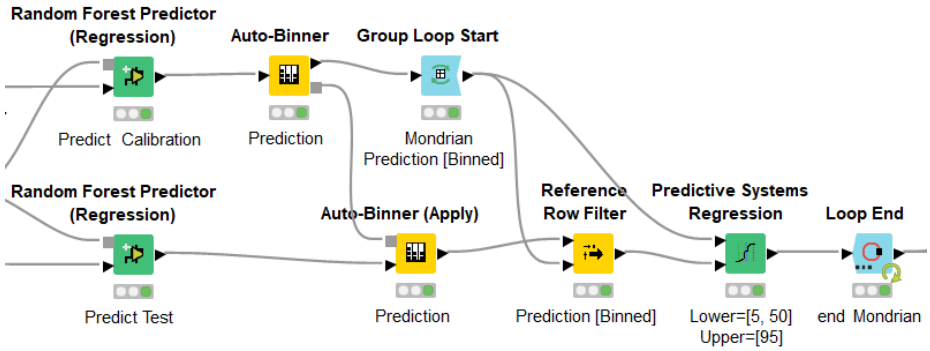


Figure 11: Solution to achieve Mondrian conformal predictive systems.

The solutions presented above and additional example workflows can be downloaded and used as templates[6]. No coding knowledge is required to use the Conformal Predictive Systems nodes introduced above (even if some supplementary material includes nodes with short code snippets).

5. Experimental setup

The Conformal Predictive Systems nodes have also been evaluated in experiments using 23 benchmarking data sets. The selection criteria were based on convenience as these data sets had been used in previous studies. Furthermore, since the Mondrian setups divide instances into five equal-sized subsets, only fairly large data sets were used. The target variables for all regression data sets were Min-Max Normalized, i.e., scaled to $[0, 1]$. By doing that, all interval sizes will be comparable and represent the proportion of the range covered by the prediction intervals in the same scale.

Percentiles are extracted so that $\epsilon \in (0, 1) = \{0.025, 0.050, \ldots, 0.975\}$ to allow plotting significance levels vs error rates. The tabulated results only include $\epsilon \in \{0.05, 0.10, 0.20\}$. The experiments use 10x10-fold cross-validation. Four setups are evaluated: standard conformal predictive systems, normalized conformal predictive systems using prediction variance from the random forest to estimate difficulty (σ), Mondrian conformal predictive systems using five equal-frequency bins defined from the prediction of the underlying model, and normalized Mondrian conformal predictive systems (combining the setups of normalized and Mondrian conformal predictive systems). The workflow used for the experiments can be found on KNIME Hub[7].

6. The COPA 2023 - Conformal predictive systems simple use cases workflow includes the examples given above.
7. The COPA 2023 - Mondrian conformal predictive systems experiment workflow includes the experiment plus some additional results.

6. Results

In the sections below, the results from our experiments are presented. As four setups have been evaluated for regression, error rate and efficiency results are separated into Tables 1 and 2. For the error rates, only results corresponding to $\epsilon = 0.05$ are shown, to allow evaluation of both two-sided and one-sided intervals. Error rate results for $\epsilon \in (0, 1) = \{0.025, 0.050, \ldots, 0.975\}$ are shown in Figure 12, below. Std stands for **st**andar**d** conformal predictive systems, N stands for **N**ormalized conformal predictive systems using prediction variance from the random forest to estimate difficulty, M stands for (standard) **M**ondrian conformal predictive systems, and NM stands for **N**ormalized **M**ondrian conformal predictive systems (also using prediction variance as σ). The error rate is the fraction of instances where the true target value is outside the prediction interval from the conformal predictive distribution. As seen in Table 1, almost all results are the same as the significance levels, regardless of whether two-sided or one-sided intervals are used. However, there is a slight tendency for the errors to be conservative on some data sets, more clearly manifested for normalized conformal predictive systems.

Table 1: Error rates for $\epsilon = 0.05$ from two-sided and one-sided intervals formed using conformal predictive systems.

	Two-sided interval				**Lower-bounded interval**				**Upper-bounded interval**			
Data sets	**Std**	**N**	**M**	**NM**	**Std**	**N**	**M**	**NM**	**Std**	**N**	**M**	**NM**
abalone	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
bank8fh	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
bank8fm	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
bank8nh	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
bank8nm	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
concrete	.05	.03	.04	.03	.05	.05	.05	.05	.05	.04	.05	.05
cooling	.04	.04	.04	.05	.05	.05	.05	.06	.04	.04	.04	.04
deltaA	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
deltaE	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
friedm	.04	.03	.04	.03	.05	.05	.05	.05	.05	.04	.05	.05
heating	.04	.04	.04	.04	.05	.06	.05	.05	.04	.04	.04	.04
kin8fh	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
kin8fm	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
kin8nh	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
kin8nm	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
laser	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.06	.05
mg	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
plastic	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
puma8fh	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
puma8fm	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
puma8nh	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
puma8nm	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
wizmir	.05	.05	.05	.05	.05	.05	.05	.05	.05	.04	.05	.05
Mean	**.05**	**.05**	**.05**	**.05**	**.05**	**.05**	**.05**	**.05**	**.05**	**.05**	**.05**	**.05**

Figure 12 shows the error rate vs the significance levels for two-sided, lower-bounded, and upper-bounded intervals. As can be seen, the error rate is almost perfectly aligned

along the diagonal in black representing the significance level, indicating valid results for all significance levels.

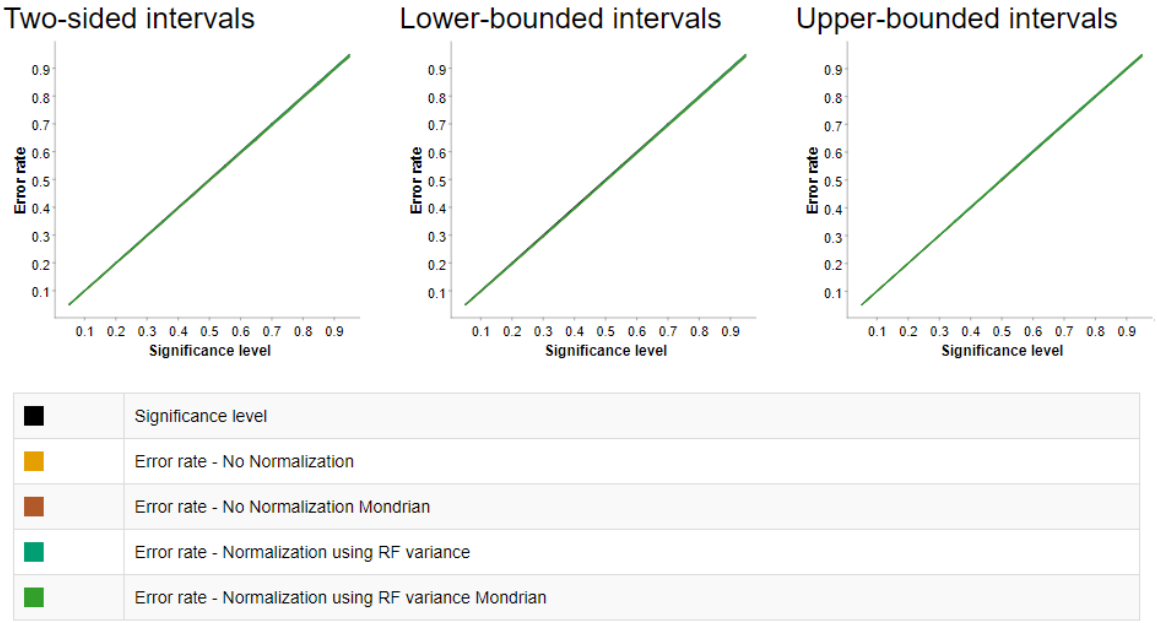


Figure 12: Error rates vs significance levels from conformal predictive systems for two-sided, lower-bounded upper-bounded intervals.

Looking at the efficiency in Table 2, the results are shown for $\epsilon \in \{0.05, 0.10, 0.20\}$. Efficiency is defined as the Median Interval Size. As all targets are in the $[0, 1]$ range, the interval sizes represent the percent of the entire output range covered by an interval (on average). As expected, smaller significance levels result in wider prediction intervals. More importantly, there is a clear difference in efficiency between standard, normalized, Mondrian, and normalized Mondrian conformal predictive systems, with increased efficiency (i.e., smaller intervals) in that order. These results are in line with previous research and should be expected. When being 95% confident in the prediction (i.e., when $\epsilon = 0.05$), the interval will, on average, cover between $\frac{1}{3}^{\text{rd}} - \frac{1}{4}^{\text{th}}$ of the target range (depending on the setup used). Reducing confidence to 90% or 80% will reduce the interval sizes towards $\frac{1}{5}^{\text{th}}$ or $\frac{1}{6}^{\text{th}}$ of the target range, respectively (for the most efficient setups).

7. Conclusions

In this paper, we introduce conformal predictive systems as an extension of the Conformal Prediction package in KNIME through a number of straightforward use cases and experiments. The package has already offered the benefits of conformal prediction, which has now been extended to include the powerful tools offered by conformal predictive systems. Furthermore, the paper also highlights how Mondrian conformal predictive systems can easily

Table 2: Efficiency from for two-sided intervals using conformal predictive systems.

	ϵ =**0.05**				ϵ =**0.10**				ϵ =**0.20**			
Data sets	**Std**	**N**	**M**	**NM**	**Std**	**N**	**M**	**NM**	**Std**	**N**	**M**	**NM**
abalone	.329	.286	.255	.266	.251	.224	.201	.207	.168	.154	.148	.149
bank8fh	.382	.348	.355	.361	.307	.287	.277	.290	.229	.216	.207	.215
bank8fm	.201	.165	.148	.152	.155	.136	.123	.128	.113	.105	.096	.099
bank8nh	.450	.395	.412	.416	.343	.312	.304	.313	.244	.220	.216	.220
bank8nm	.236	.102	.106	.093	.167	.079	.084	.075	.099	.058	.061	.055
concrete	.308	.340	.264	.304	.235	.232	.208	.195	.173	.162	.154	.143
cooling	.245	.191	.193	.170	.194	.172	.152	.153	.130	.140	.106	.123
deltaA	.155	.130	.132	.129	.117	.099	.103	.095	.081	.072	.073	.068
deltaE	.216	.213	.209	.206	.174	.172	.165	.165	.128	.131	.120	.120
friedm	.283	.306	.293	.315	.227	.217	.231	.221	.169	.163	.170	.164
heating	.168	.117	.104	.096	.106	.098	.083	.082	.080	.073	.066	.065
kin8fh	.301	.281	.289	.282	.248	.233	.241	.235	.189	.182	.186	.182
kin8fm	.188	.146	.178	.145	.153	.121	.147	.121	.116	.092	.113	.094
kin8nh	.495	.501	.490	.489	.421	.420	.416	.411	.330	.326	.328	.320
kin8nm	.420	.401	.386	.364	.351	.329	.331	.306	.276	.254	.265	.241
laser	.100	.129	.054	.061	.057	.063	.038	.039	.033	.031	.026	.026
mg	.403	.403	.186	.205	.284	.295	.145	.154	.177	.180	.110	.112
plastic	.654	.643	.723	.700	.549	.577	.590	.577	.421	.437	.443	.433
puma8fh	.572	.567	.547	.557	.476	.481	.452	.464	.368	.371	.358	.363
puma8fm	.271	.281	.263	.264	.224	.233	.217	.215	.172	.183	.169	.164
puma8nh	.543	.548	.503	.512	.445	.453	.408	.415	.339	.332	.319	.315
puma8nm	.282	.252	.256	.243	.235	.205	.219	.199	.185	.153	.179	.151
wizmir	.085	.089	.081	.086	.068	.068	.064	.066	.051	.051	.049	.049
Mean	**.317**	**.297**	**.279**	**.279**	**.252**	**.239**	**.226**	**.223**	**.186**	**.178**	**.172**	**.168**

be achieved through a few additional nodes. Having these strong tools easily accessible in a user-friendly platform such as KNIME increase the possibility of reaching user groups otherwise without access to the conformal framework.

Future planned expansions of the package include supporting Venn and Venn-Abers predictors for classification. Adding internal support for Mondrian categories is also a strong candidate for future improvements. Smoothed p-values for classification and interpolation are two other updates on the future work list.

Combining conformal prediction with interpretable models like decision trees and rule sets has previously been proposed in many papers. Adding nodes that enrich interpretable models with conformal and Venn predictions is also an interesting option for future work.

Acknowledgments

The authors acknowledge the Knowledge Foundation and the industrial partners for financially supporting the research and education environment on Knowledge Intensive Product Realization SPARK at Jönköping University, Sweden. Projects: AFAIR (20200223) and PREMACOP (20220187). Redfield has also contributed substantially with time and experience in discussions and development.

References

Antreas Afantitis, Andreas Tsoumanis, and Georgia Melagraki. Enalos suite of tools: Enhancing cheminformatics and nanoinformatics through knime. *Current Medicinal Chemistry*, 27:6523–6535, 7 2020. ISSN 09298673. doi: 10.2174/0929867327666200727114410.

Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime - the konstanz information miner: Version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11(1): 26–31, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656280. URL http://doi.acm.org/10.1145/1656274.1656280.

Henrik Boström. crepes: a python package for generating conformal regressors and predictive systems. In Ulf Johansson, Henrik Boström, Khuong An Nguyen, Zhiyuan Luo, and Lars Carlsson, editors, *Proceedings of the Eleventh Symposium on Conformal and Probabilistic Prediction and Applications*, volume 179 of *Proceedings of Machine Learning Research*. PMLR, 2022.

Henrik Boström and Ulf Johansson. Mondrian conformal regressors. In Alexander Gammerman, Vladimir Vovk, Zhiyuan Luo, Evgueni Smirnov, and Giovanni Cherubin, editors, *Proceedings of the Ninth Symposium on Conformal and Probabilistic Prediction and Applications*, volume 128 of *Proceedings of Machine Learning Research*, pages 114–133. PMLR, 09–11 Sep 2020. URL https://proceedings.mlr.press/v128/bostrom20a.html.

Henrik Boström, Ulf Johansson, and Tuwe Löfström. Mondrian conformal predictive distributions. In Lars Carlsson, Zhiyuan Luo, Giovanni Cherubin, and Khuong An Nguyen, editors, *Proceedings of the Tenth Symposium on Conformal and Probabilistic Prediction and Applications*, volume 152 of *Proceedings of Machine Learning Research*, pages 24–38. PMLR, 08–10 Sep 2021. URL https://proceedings.mlr.press/v152/bostrom21a.html.

Alexander Fillbrunn, Christian Dietz, Julianus Pfeuffer, René Rahn, Gregory A. Landrum, and Michael R. Berthold. Knime for reproducible cross-domain analysis of life science data. *Journal of Biotechnology*, 261:149–156, 11 2017. ISSN 0168-1656. doi: 10.1016/J.JBIOTEC.2017.07.028.

Tuwe Löfström, Artem Ryasik, and Ulf Johansson. Tutorial for using conformal prediction in knime. In *Conformal and Probabilistic Prediction with Applications*, pages 4–23. PMLR, 2022.

Artem Ryasik. Conformal prediction for regression. Medium, March 29 2023a. https://medium.com/@artem_ryasik/conformal-prediction-for-regression-1ba24f1442df.

Artem Ryasik. Conformal prediction for classification. Medium, 2023b. https://medium.com/low-code-for-advanced-data-science/conformal-prediction-for-classification-6bf152abb491.

Artem Ryasik and Greg Landrum. Conformal prediction: Enhanced method for understanding prediction quality. YouTube video, May 13 2020. https://www.youtube.com/watch?v=_ZVuEWEfwuw&t=10s.

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

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

Vladimir Vovk, Jieli Shen, Valery Manokhin, and Min-ge Xie. Nonparametric predictive distributions based on conformal prediction. *Mach. Learn.*, 108(3):445–474, 2019.