Tutorial for using conformal prediction in KNIME

Tuwe Löfström[*] TUWE.LOFSTROM@JU.SE
Jönköping AI Lab, Department of Computing, Jönköping University, Sweden
Artem Ryasik ARTEM.RYASIK@REDFIELD.SE
Redfield AB, Sweden
Ulf Johansson ULF.JOHANSSON@JU.SE
Jönköping AI Lab, Department of Computing, Jönköping University, Sweden

Editor: Ulf Johansson, Henrik Boström, Khuong An Nguyen, Zhiyuan Luo 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. Redfield have previously developed nodes for conformal classification in KNIME. We introduce an extended conformal prediction package with added support for conformal regression. The conformal prediction package include class-conditional conformal classification, conformal regression and normalized conformal regression. The updated package also includes several new and updated nodes that focus on ease-of-use. This paper provide an introduction to various use cases for both simplified and advanced use as well as experiments to prove validity and showcase functionality. All examples are publicly available and the package is available through KNIME's official software channels.

Keywords: Conformal prediction, software implementations, KNIME Analytics Platform

1. Introduction

Machine learning and data science are gradually becoming more and more mainstream. Standard techniques, like random forests and deep learning, are becoming available through libraries and tools. This makes usage of machine learning more and more widespread in a wide range of industries and applications, influencing decision making and work processes wherever it is applied. A particular set of tools offer users with the possibility to build up data science workflows using components. Each component represent a specific behaviour, such as data manipulation, modeling or visualization. The workflows are often constructed using visual programming, enabling the creation of no-code data science pipelines without any knowledge of coding. There are many tools that offer the possibility to build workflows using visual programming, such as RapidMiner, WEKA, Orange and KNIME. Even if these tools vary in interactivity, openness and ease-of-use, they all aim to make data analysis more easily available to novice users.

In this paper, we will describe the Conformal Prediction package for KNIME and provide a brief introduction to KNIME[1] (Berthold et al., 2009). KNIME consists of two parts, the KNIME Analytics Platform, which is an open source software for end-to-end data analytics using visual programming, and the KNIME Server, which is an enterprise software for team-based collaboration, automation, management, and deployment of data science workflows.

[*] Corresponding author.
1. knime.com

Conformal prediction (Vovk et al., 2005) is a framework providing precise levels of confidence in predictions. Conformal prediction has attracted a growing amount of scientific interest in recent years, with a predicted increase in attention onward, as illustrated by Figure 1 (Manokhin and Vishwakarma). However, it is not yet widely acknowledge outside academia as a natural and important tool to improve the quality of decision support systems relying on predictions. Making it more easily accessible to a wider public, not necessarily knowledgeable in programming, is important for increased outreach.

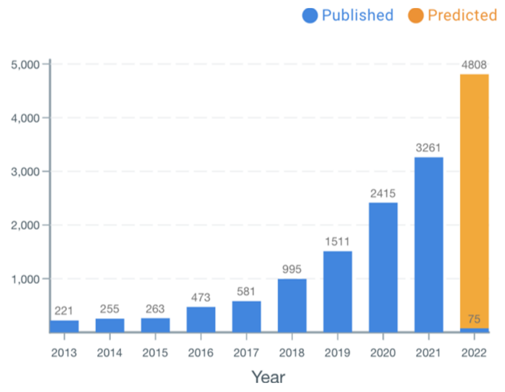


Figure 1: Past and predicted number of publications on conformal prediction

The Swedish company Redfield AB has previously developed a toolbox for conformal classification for KNIME which has been available through KNIME's official software channels as a community package. The original package was developed based on the work by Norinder et al. (2015). In this paper, we introduce an update and extension of the package to include conformal regression as well as a number of additions to improve ease-of-use and accessibility to novice users. We illustrate the functionality using a number of typical use cases.

2. KNIME Analytics Platform

The development of KNIME began in 2004 within a university group and had its first software release in 2006. In collaboration with an active user community, KNIME has been developed into a full-fledged and open platform for data science. KNIME has support for most standard machine learning techniques of-the-shelf and has extensive support for data blending, preprocessing, integration and deployment. On top of that, a lot of additional functionality is available through community content, providing support for a very wide range of application areas (with a focus on life science (Fillbrunn et al., 2017; Afantitis et al., 2020)), techniques (such as deep learning), and types of data sources (including image, text and network mining).

3. Conformal Prediction

Conformal predictors are predictive models that associate each of their predictions with a measure of confidence. Given a test pattern x_i, and a significance level ϵ, a conformal predictor outputs a multi-valued *prediction region* Γ_i^ϵ containing the correct target y_i with probability $1 - \epsilon$. For classification, the prediction regions are sets of class labels, and for regression they are prediction intervals.

An *error* in conformal prediction is when the true target is not included in the prediction region. Conformal predictors are *automatically valid*, in the sense that the error rate will be exactly ϵ, in the long run, and under the common i.i.d. assumption. Since all conformal predictors are valid, the key criterion when evaluating conformal predictors is *efficiency*, i.e., the size of the prediction regions. Naturally, to increase informativeness, prediction regions should also be individualized (or sharp) and small, which for classification means fewer predicted classes and for regression means that the prediction intervals optimally should be as tight as possible for every individual test instance.

3.1. Conformal Classification

A central component of the conformal prediction framework is the nonconformity function. The nonconformity function assigns a nonconformity score to each instance-label pair in classification. When predicting a test instance, a nonconformity score is assigned to each possible class label, and the scores are compared to the scores obtained from calibration instances with known class labels. The calibration instances are only assigned a nonconformity score for the true class. The labels assigned to a test instance that is found to be nonconforming compared to the scores of the instances with known labels are excluded. A label is considered nonconforming if the nonconformity score for that label is higher than a predefined fraction (i.e., the significance level ϵ) of the scores assigned to the calibration instances. Therefore, clearly, conformal prediction is a form of hypothesis testing. Each possible class could be considered a null hypothesis that may be disproved if it is significantly different from labelled data. The prediction for the test instance is the set of class labels that were not excluded.

A conformal predictor for classification, called an *inductive conformal classifier*, is constructed using the following general procedure:

1. Divide the training set Z into two disjoint subsets: A *proper training set* Z_T and a *calibration set* Z_C where $|Z_C| = q$. Each instance is composed of two parts, $z = (x, y)$, where x is the input pattern and y is the target.

2. Fit a classification model h using Z_T. This is the *underlying model.*

3. Let
$$f(z_i) = \Delta\left(y_i, h(x_i)\right),$$
i.e., the distance between the true target and the prediction. This is the so-called *nonconformity function.* The probability estimate from the underlying model is often utilized as nonconformity for classification.

4. Apply $f(z)$ to $\forall z_i \in Z_C$ and save these *calibration scores*, sorted in descending order, as $\alpha_1, ..., \alpha_q$.

5. Calculate the p-value for each possible class label y^c for a test example in the following way:
$$p^{y^c} = \frac{\left(|\{i = 1, ..., q : y_i = y^c \wedge \alpha_i \geq \alpha^{y^c}\}| + 1\right)}{\left(|\{i = 1, ..., q : y_i = y^c\}| + 1\right)},$$
where $\alpha_i = f(z_i), i = 1, ..., q$ and $\alpha^{y^c} = \Delta\left(y^c, h(x_i)\right)))$ are the nonconformity scores for the calibration set and for class y^c on the test example, respectively. This form of conformal classification is called class-conditional, since only calibration instances with the class y^c are included in the calculation.

6. Fix a significance level $\epsilon \in (0, 1)$, where typical values are $\epsilon = 0.01$, $\epsilon = 0.05$, $\epsilon = 0.1$.

7. The prediction region for a new example is $\Gamma^\epsilon := \{y^c | p^{y^c} > \epsilon\}$.

Class-conditional conformal classification provide guarantees for each class label individually as well as for the entire prediction region. When not using class-conditional conformal classification, the guarantees only apply for the entire prediction region but not for each class individually. This means that for each individual class, nothing can be guaranteed since all errors may be made on only one of the classes (Löfström et al., 2015). Using class-conditional conformal prediction may result in larger prediction regions compared to standard conformal classification, but provide results that are more intuitive and useful.

3.2. Conformal Regression

Constructing a conformal predictor for regression, called an *inductive conformal regressor*, is in many ways analogous to classification but differ in some important ways. It is constructed using the following general procedure:

1. Divide the training set Z into two disjoint subsets: A *proper training set* Z_t and a *calibration set* Z_c where $|Z_c| = q$.

2. Fit a regression model h using Z_t. This is the *underlying model*.

3. Let
$$f(z_i) = |y_i - h(x_i)| \tag{1}$$
i.e., the absolute error. This is the standard *nonconformity function* for regression.

4. Apply $f(z)$ to $\forall z_i \in Z_c$ and save these *calibration scores*, sorted in descending order, as $\alpha_1, ..., \alpha_q$.

5. Fix a significance level $\epsilon \in (0, 1)$, where typical values are $\epsilon = 0.01$, $\epsilon = 0.05$, $\epsilon = 0.1$.

6. Let $s = \lfloor \epsilon(q + 1) \rfloor$, i.e., the index of the $(1 - \epsilon)$-percentile nonconformity score, α_s.

7. The prediction interval for a new example is
$$\Gamma_i^\epsilon = h(x_i) \pm \alpha_s, \tag{2}$$
based on the reasoning that the probability for a test instance to obtain a larger absolute error than α_s, under the i.i.d. assumption, must be ϵ. So, the interval contains y_i with confidence $1 - \epsilon$.

It is possible to design many different nonconformity functions for a specific underlying regression model, and each of them will define a different conformal regressor. It must be noted, that *all* of these conformal regressors will be valid, but there may be significant differences in terms of efficiency. Specifically, in the procedure described above, i.e., using $f(z_i) = |y_i - h(x_i)|$ and $\Gamma_i^\epsilon = h(x_i) \pm \alpha_s$, each prediction interval will have the same size, which is $2\alpha_s$. However, in order to increase the informativeness, and to potentially minimize prediction regions, we would like to obtain individual bounds for each x_i, which is achieved using a *normalized nonconformity function*.

Normalized nonconformity functions utilize additional terms σ_i and β.

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

where σ_i is an estimate of the difficulty of predicting y_i. The normalized prediction for a new example is

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

where β is a sensitivity parameter determining the relative importance of the normalization term.

3.3. Conformal Prediction in KNIME

Redfield AB deployed the Conformal Prediction package in KNIME for class conditional conformal classification in 2020. The original Conformal Prediction package contains several advanced nodes for creating efficient conformal classifiers. The original nodes are:

- Conformal Calibrator - Creates a calibration table with ranking for further calibration of the test data sets by the Conformal Predictor.

- Conformal Predictor - Calculates the table with p-values for test or unlabeled data based on the provided calibration table. The p-value for a class is the fraction of entries from that class in the calibration set that have a model probability less than or equal to the model probability for the record under consideration. Small p-values indicate records that are nonconforming, larger p-values indicate records that are conforming.

- Conformal Classifier - Produces predictions based on the calibration table and the significance level provided by the user. The significance level defines the tolerable percentage of the prediction errors. The node expects p-value columns (produced by the Conformal Predictor) for all the classes in the target domain.

- Conformal Scorer - Compares predictions made by Conformal Classifier with actual values and calculates metrics for estimating conformal predictions.

- Conformal Calibration Loop Start/End - A pair of loop nodes that divides the input table into a training and calibration set. The nodes iterate over the training and calibration data.

- Conformal Prediction Loop Start/End - A pair of nodes that iterate over synchronized model and calibration data from each iteration run using the calibration loop nodes. The Conformal Prediction Loop End node aggregates over the calibration iterations by taking the median of all the iterations.

These nodes are left as is for compatibility reasons. The Conformal Scorer has been updated to output total aggregated evaluation metrics along with the per-class metrics originally included. The metrics presented in Vovk et al. (2017) have been added as optional aggregated metrics.

3.4. Adding Conformal Regression

For the release of the updated Conformal Prediction package, functionality for conformal regression has been added. The aim was to follow the same logic as used in the original implementation. Consequently, the intention was to create a calibrator, a predictor, a classifier and a scorer node for conformal regression. However, as opposed to conformal classification where the p-value can be calculated without knowledge of the significance level, the significance level is explicitly used to select the α_s which is defining the lower and upper bounds of the conformal regression. Consequently, it is not possible to separate the predictor and classifier for regression, resulting in the following nodes being added to facilitate functionality for conformal regression:

- Conformal Calibrator (Regression) - Calculates the α-values for all calibration instances using Equation (1) for standard conformal regression. If normalization is used, the difficulty, or σ, and β are also used in the calculation of α, according to Equation (3).

- Conformal Predictor and Classifier (Regression) - Combines the functionality of the predictor and classifier. The node takes the calibration table and the test data and defines the prediction interval based on the user defined the significance level using Equation (2). If normalization was used for calibration, the exact same setting as in the Conformal Calibrator (Regression) must be used in this node as well, defining the prediction interval using Equation (4).

- Conformal Scorer (Regression) - Compares prediction intervals with actual values and calculates metrics for estimating conformal predictions.

The nodes are named following the naming convention for regression used in KNIME, using the same names as for classification with "(Regression)" added at the end.

3.5. Addressing Usability Aspects

When using conformal prediction, the input is the calibration data, the test data and (at least) the significance level as a parameter, producing conformal predictions as the output. Even if the calculations of α-values, p-values and prediction regions for classification or α-values and prediction intervals for regression can be separated into multiple atomic steps, most use cases don't require such an explicit separation. Instead, the most intuitive way to use conformal prediction is to see conformal classification or conformal regression as a single step, taking the calibration and test data together with necessary parameters as input, producing conformal predictions as output. Consequently, in order to address these situations and to make straightforward use of conformal prediction more accessible, we also added two additional nodes:

- Conformal Classification - The node merges the functionality of the Conformal Calibrator, the Conformal Predictor and the Conformal Classifier into a single node with the sum of the parameters available in the individual nodes.

- Conformal Regression - The node merges the functionality of the Conformal Calibrator (Regression) and the Conformal Predictor and Classifier (Regression) into a single node with the sum of the parameters available in the individual nodes. Using this node alleviates the need for exact matching of normalization settings across nodes.

Finally, the error rates of the conformal predictors will be exactly the same as the significance levels in the long run, meaning that this will eventually average to the same values if the entire process of generating a conformal prediction is repeated a sufficient number of times. However, the error rate for a single run may deviate from the significance level and in order to get error rates closer to the significance levels used, there are some tricks that have been proposed. Interpolation was proposed by Carlsson et al. (2015). A simpler approach is to let the number of calibration instances be easily dividable by typical significance levels, making it possible to identify a particular instance as being the $(1-\epsilon)^{\text{th}}$ percentile. Defining the calibration size in this way will reduce the variance in error rates, making each repetition having an error rate closer to the significance level.

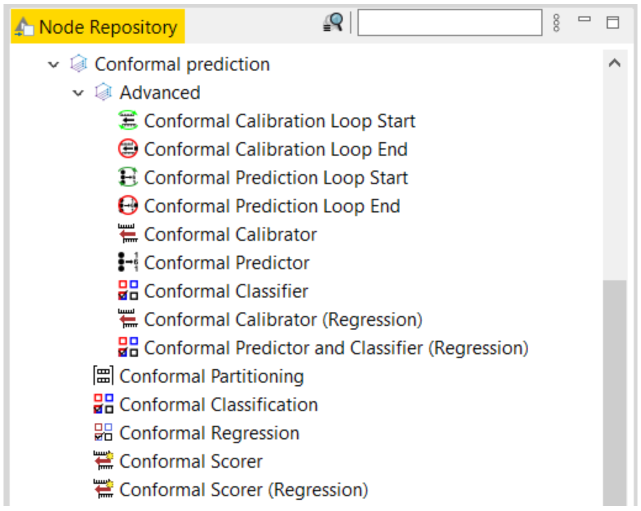

Node Repository
- Conformal prediction
  - Advanced
    - Conformal Calibration Loop Start
    - Conformal Calibration Loop End
    - Conformal Prediction Loop Start
    - Conformal Prediction Loop End
    - Conformal Calibrator
    - Conformal Predictor
    - Conformal Classifier
    - Conformal Calibrator (Regression)
    - Conformal Predictor and Classifier (Regression)
  - Conformal Partitioning
  - Conformal Classification
  - Conformal Regression
  - Conformal Scorer
  - Conformal Scorer (Regression)

Figure 2: The structure of the Conformal Prediction package in KNIME after update

In order to make this approach available, an additional node have been included:

- Conformal Partitioning - A partitioning node with exactly the same settings as an ordinary partitioning node, which adjust the calibration size to be $k*100-1$, where k is the largest number making the calibration set less than the original size n it would have got from an ordinary partitioning node. If the original size is $100 > n \geq 10$, the calibration size is $k * 10 - 1$. If too few instances ($n < 10$) are assigned as calibration set, execution is halted showing an error indicating that the calibration set is too small.

The set of nodes are organized in the package as seen in Figure 2.

4. Use cases

To illustrate how conformal prediction can be applied in KNIME, a number of use cases, from simple to more complex, will be presented here. All workflows presented below plus some additional workflows are available for download at KNIME Hub[2].

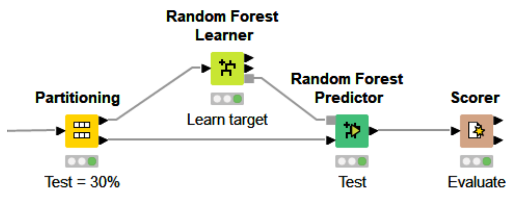


Figure 3: Classification without conformal prediction

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.

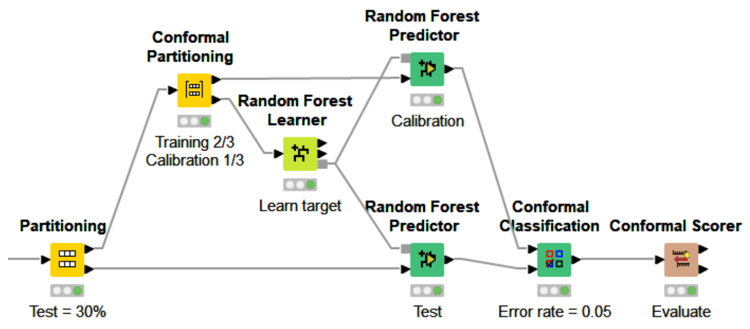


Figure 4: Conformal classification with scoring using partitioning

When applying conformal prediction, all of the steps described above are included but some additional steps need to be done as well. As before, the data must first be divided into training and test sets. Since conformal prediction requires a calibration set as well, 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

2. URL: https://kni.me/s/BuIvXUolHRH3Cy5o

the proper training set. Since calibration and test sets must always be exchangeable, they must always be handled in the same way, so both sets are predicted using Predictor nodes. It is important to select the option to *append individual class probabilities* in both Predictor nodes. The predictions from both sets are fed into the Conformal Classification node to get the conformal prediction based on the user defined significance level. Finally, the prediction is evaluated using a Conformal Scorer 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 always done once at the end, on the entire evaluated data.

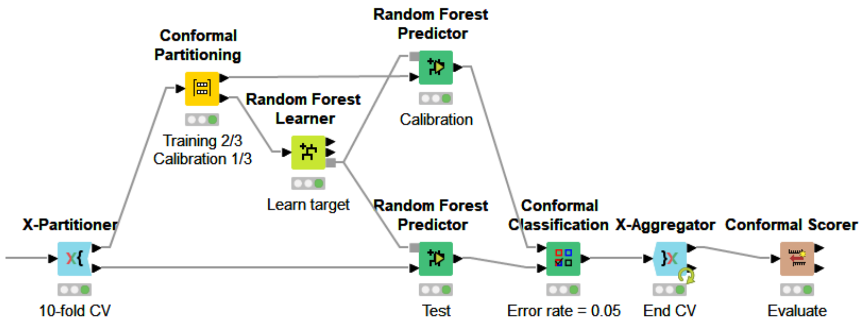


Figure 5: Conformal classification with scoring using cross validation

Creating a solution for standard conformal regression is analogous to classification, Cf. Figures 4 and 6.

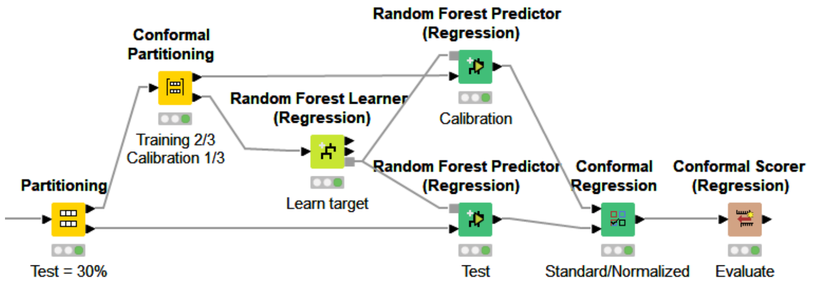
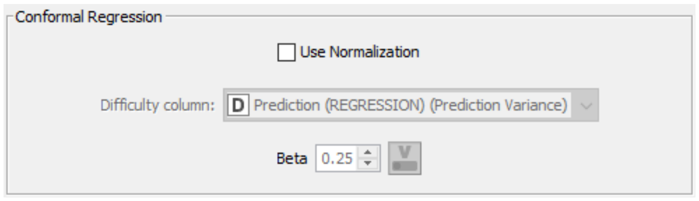


Figure 6: Standard (or Normalized) Conformal Regression with scoring

In order to use normalized conformal regression, an additional column of data representing σ, i.e., how difficult an instance is, must exist in the data. If such a column exist, changing from standard conformal regression to normalized conformal regression requires only opting for using normalization and selecting the σ-column. If using a random forest as underlying model, the prediction variance will automatically be added as a separate column along with the prediction. Consequently, changing from standard conformal regression, see configuration in Figure 7(*a*), into normalized conformal regression using prediction variance from the random forest, see configuration in Figure 7(*b*), will still result in a workflow that looks like Figure 6, only changing the options in the Conformal Regression node.



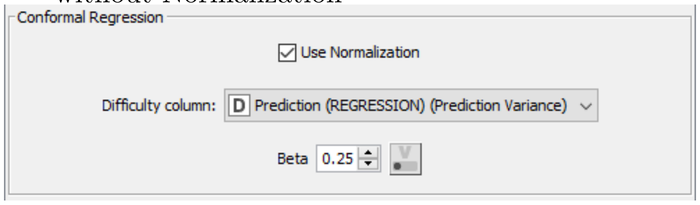
Conformal Regression

☐ Use Normalization

Difficulty column: D Prediction (REGRESSION) (Prediction Variance)

Beta 0.25

(*a*) Standard Conformal Regression without Normalization



Conformal Regression

☑ Use Normalization

Difficulty column: D Prediction (REGRESSION) (Prediction Variance)

Beta 0.25

(*b*) Normalized Conformal Regression using the prediction variance as difficulty estimate

Figure 7: Setting up conformal regression

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 to do it is to train a separate model on the absolute error of the first model. A workflow doing that using the training data is shown in Figure 8.

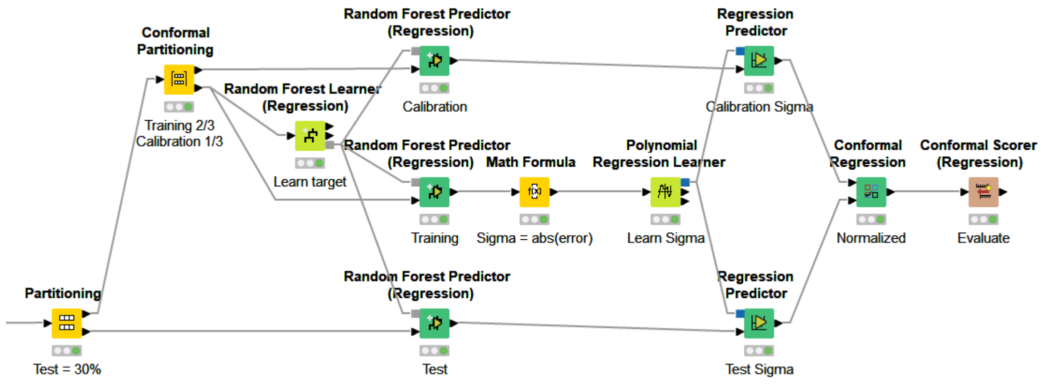
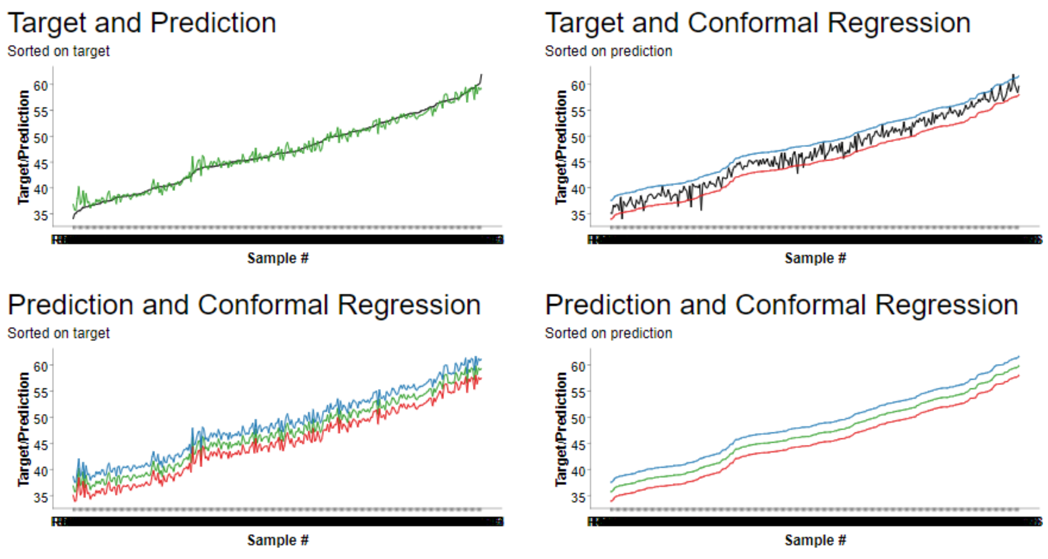

Random Forest Predictor (Regression)
Regression Predictor
Conformal Partitioning
Random Forest Learner (Regression)
Calibration
Calibration Sigma
Training 2/3 Calibration 1/3
Learn target
Random Forest Predictor (Regression)
Math Formula
Polynomial Regression Learner
Conformal Regression
Conformal Scorer (Regression)
Training
Sigma = abs(error)
Learn Sigma
Normalized
Evaluate
Random Forest Predictor (Regression)
Regression Predictor
Partitioning
Test = 30%
Test
Test Sigma

Figure 8: Normalized Conformal Regression with scoring

The underlying model is trained in the same way and the calibration and test data is predicted as before. In order to train the difficulty estimator on the absolute errors, a prediction is needed on the data set used to train the difficulty estimator (in this workflow we use the proper training set, but an additional validation set can be used to get unbiased errors if there is an abundance of data). The absolute error can be calculated using a Math

Formula node, which in our case creates a new difficulty column called *Sigma*. The difficulty estimator learns to predict the Sigma column. Once trained, the difficulty estimator is used to predict the difficulty of the calibration and test sets. Finally, the calibration and test sets are fed into the Conformal Regression node, assigning the column with predicted sigma values as difficulty column. Finally, the results are evaluated using the Conformal Scorer (Regression) node.

Looking at the conformal predictors on an individual data set, the Stock data set in this case, Figure 9 show the results using four different line plots for a standard conformal regressor defined using significance level $\epsilon = 0.05$. The two subplots to the left are sorted on target values and the two subplots to the right are sorted on prediction values. The plot show the result for standard conformal regression, where the prediction intervals are equally wide for all conformal predictions. The intervals are equally wide is most clearly seen in the two subplots to the right, since they are sorted on the prediction values. In the two top plots, the prediction from the underlying model and the conformal regression are compared to the true target (in black).



⬛	REGRESSION
🟩	Prediction
🟦	Upper bound
🟥	Lower bound

Figure 9: Standard Conformal Regression without Normalization

Using the same set of plots for normalized conformal prediction using prediction variance as difficulty estimate can be seen in Figure 10. As can be seen, the prediction intervals are clearly affected by the difficulty estimate used, with some intervals being extremely narrow and others being extremely wide. The width of the interval can be used as an estimate of how difficult an instance is.

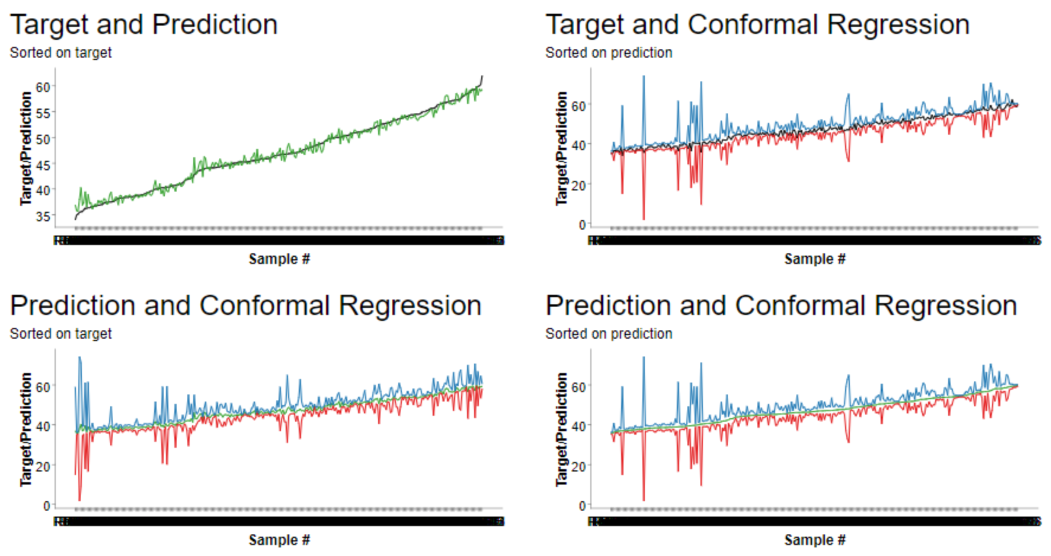


Figure 10: Normalized Conformal Regression using the prediction variance as difficulty estimate

The final use case demonstrates the conformal regression using cross-calibration, i.e., using the conformal loops as described above (see Figure 11). This workflow is following the same logic as the Redfield workflow for conformal classification[3] that was presented in 2020. The Stock data set was used as in the previous example. However instead of using the Conformal Partitioning node, Conformal Calibration and Conformal Prediction loops were used. The Conformal Calibration Loop nodes were used to run five iterations, each training a regression model and predicting the corresponding calibration table. Then in the Conformal Prediction Loop, the models were synchronized with calibration tables to obtain conformal regression predictions for each iteration. Finally, the predictions from all iterations were aggregated by taking the median of the results from the different loops. This approach helps getting more robust predictions, which are less dependent on the sampling during the training. The results obtained for the Stock data set can be seen in Figure 12, and they are very similar to the use case shown in Figure 9.

The solutions presented above, along with additional example workflows, can be downloaded and used as templates. No knowledge of coding is required to use the Conformal Prediction nodes introduced above (even if some of the supplementary material include nodes with short snippets of code). A video tutorial for the original implementation focusing solely on classification was held at a KNIME Webinar in 2020[4].

3. URL: https://kni.me/w/BaWdZnceB7O4sqjA
4. Url: https://www.youtube.com/watch?v=_ZVuEWEfwuw

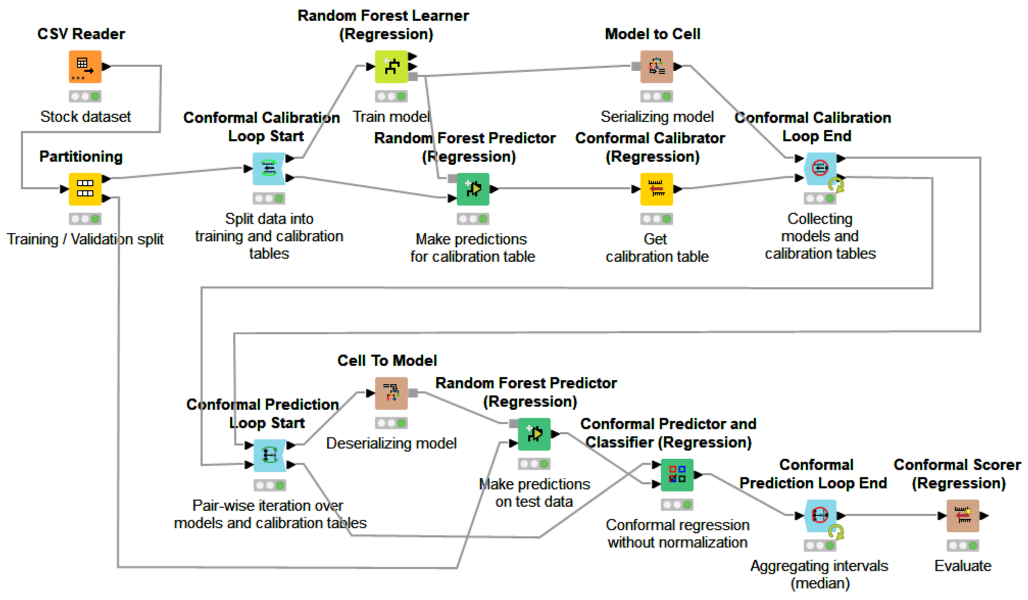

CSV Reader
Stock dataset
Partitioning
Training / Validation split
Conformal Calibration Loop Start
Split data into training and calibration tables
Random Forest Learner (Regression)
Train model
Random Forest Predictor (Regression)
Make predictions for calibration table
Model to Cell
Serializing model
Conformal Calibrator (Regression)
Get calibration table
Conformal Calibration Loop End
Collecting models and calibration tables
Cell To Model
Deserializing model
Conformal Prediction Loop Start
Pair-wise iteration over models and calibration tables
Random Forest Predictor (Regression)
Make predictions on test data
Conformal Predictor and Classifier (Regression)
Conformal regression without normalization
Conformal Prediction Loop End
Aggregating intervals (median)
Conformal Scorer (Regression)
Evaluate

Figure 11: Conformal regression with cross-calibration - workflow overview

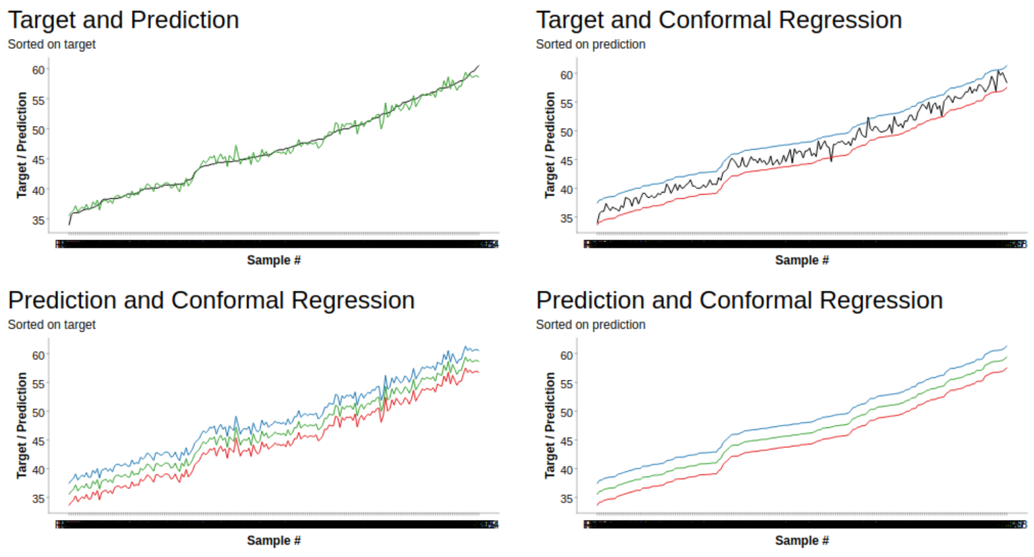


Figure 12: Conformal regression with cross-calibration

5. Experimental setup

The Conformal Prediction package has also been evaluated in experiments using multiple benchmarking data sets. 25 binary data sets were used for classification and 33 data sets were used for regression. The selection criteria was based on convenience as these data sets had been used in previous studies. The target variables for all regression data sets were Min-Max Normalized, i.e., they were scaled to the range $[0, 1]$. By doing that, all interval sizes will be comparable and will represent the proportion of the entire range covered by the prediction intervals in the same scale.

Both classification and regression are run using significance values

$$\epsilon \in (0, 1) = \{0.01, 0.02, \ldots, 0.99\}$$

to allow plotting significance levels vs error rates. The tabulated results only include $\epsilon \in \{0.01, 0.05, 0.10\}$. Both experiments are using 10-fold cross validation and the experiments are repeated multiple times. Regression is evaluated using the three setups presented in Figures 6, 7 and 8.

6. Results

In the sections below, the results from the two experiments are presented.

6.1. Classification

Table 1 present the results for classification. The *Error rate* is the proportion of instances where the true target was not in the prediction region and *Efficiency* is the proportion of instances with a prediction region containing a single class label. The Conformal Scorer node output *Validity*, which is $1 - \textit{error rate}$. The three columns per metric are the significance levels used, i.e., significance levels used are $\epsilon \in \{0.01, 0.05, 0.1\}$.

When averaging over multiple data sets and trials, the error rate is exactly corresponding to the significance level, as expected. We also see that the proportion of predictions with a single class increase as the significance level increase.

Figure 13 show the error rate vs the significance levels for each individual class as well as in total. As expected, the error rate is aligned along the diagonal in black representing the significance level, indicating valid results for individual classes as well as in total.

6.2. Regression

As three setups have been evaluated for regression, results for error rate and efficiency are separated into Tables 2 and 3. The results are grouped based on significance level (first row). Std stands for **St**andar**d** Conformal Regression, NPV stands for **N**ormalized Conformal Regression using **P**rediction **V**ariance from the random forest to estimate difficulty, and NPR stands for **N**ormalized Conformal Regression using **P**olynomial **R**egression to estimate difficulty. The error rate is the fraction of instances where the true target value is outside the prediction interval from the conformal predictor. As can be seen in Table 2, almost all results, as well as the Mean error rate, are the same as the significance levels. The error rates for heating are slightly conservative and are rounded to 0.00 for significance level 0.01. The actual error rates are 0.004 when rounded to three decimals.

Table 1: Results from classification experiment.

	Error rate			**Efficiency**		
Data sets	ϵ =**0.01**	ϵ =**0.05**	ϵ =**0.10**	ϵ =**0.01**	ϵ =**0.05**	ϵ =**0.10**
colic	0.01	0.06	0.11	0.16	0.54	0.75
creditA	0.01	0.05	0.10	0.21	0.67	0.92
diabetes	0.01	0.06	0.11	0.11	0.43	0.64
german	0.01	0.05	0.10	0.03	0.16	0.28
haberman	0.01	0.06	0.11	0.04	0.20	0.35
heartC	0.01	0.04	0.09	0.12	0.53	0.75
heartH	0.01	0.05	0.11	0.18	0.60	0.79
heartS	0.01	0.05	0.11	0.11	0.51	0.80
hepati	0.01	0.06	0.11	0.10	0.41	0.68
iono	0.01	0.05	0.10	0.38	0.90	0.95
je4042	0.01	0.05	0.11	0.07	0.36	0.59
je4243	0.01	0.06	0.11	0.05	0.25	0.43
kc1	0.01	0.05	0.10	0.05	0.23	0.39
kc2	0.01	0.07	0.12	0.13	0.49	0.65
kc3	0.01	0.04	0.10	0.04	0.24	0.51
liver	0.01	0.05	0.10	0.05	0.25	0.44
pc1req	0.01	0.04	0.09	0.03	0.17	0.37
pc4	0.01	0.06	0.11	0.27	0.73	0.91
sonar	0.01	0.05	0.10	0.15	0.59	0.75
spect	0.01	0.04	0.09	0.04	0.15	0.29
spectf	0.01	0.05	0.11	0.09	0.42	0.65
transfusion	0.01	0.05	0.09	0.03	0.20	0.34
ttt	0.01	0.04	0.09	0.78	0.97	0.91
vote	0.01	0.05	0.10	0.21	0.62	0.80
wbc	0.01	0.07	0.12	0.40	0.95	0.91
Mean	**0.01**	**0.05**	**0.10**	**0.15**	**0.46**	**0.63**

Table 2: Error rates from regression experiment.

	ϵ =**0.01**			ϵ =**0.05**			ϵ =**0.10**		
Data sets	**Std**	**NPV**	**NPR**	**Std**	**NPV**	**NPR**	**Std**	**NPV**	**NPR**
abalone	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
anacalt	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
bank8fh	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
bank8fm	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
bank8nh	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
bank8nm	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
boston	0.01	0.01	0.01	0.04	0.05	0.05	0.09	0.09	0.08
comp	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
concreate	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.09
cooling	0.01	0.01	0.01	0.04	0.04	0.04	0.08	0.09	0.08
deltaA	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
deltaE	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
friedm	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
heating	0.00	0.00	0.00	0.04	0.04	0.04	0.09	0.09	0.09
istanbul	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
kin8fh	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
kin8fm	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
kin8nh	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
kin8nm	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
laser	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
mg	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
mortage	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
plastic	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
puma8fh	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
puma8fm	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
puma8nh	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
puma8nm	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
quakes	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
stock	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
treasury	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
wineRed	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
wineWhite	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
wizmir	0.01	0.01	0.01	0.05	0.05	0.05	0.10	0.10	0.10
Mean	**0.01**	**0.01**	**0.01**	**0.05**	**0.05**	**0.05**	**0.10**	**0.10**	**0.10**

Table 3: Efficiency from regression experiment.

	ϵ =**0.01**			ϵ =**0.05**			ϵ =**0.10**		
Data sets	**Std**	**NPV**	**NPR**	**Std**	**NPV**	**NPR**	**Std**	**NPV**	**NPR**
abalone	0.54	0.52	0.52	0.31	0.30	0.30	0.23	0.22	0.22
anacalt	0.60	0.45	0.54	0.25	0.23	0.23	0.16	0.15	0.15
bank8fh	0.55	0.54	0.53	0.39	0.38	0.38	0.31	0.30	0.30
bank8fm	0.33	0.32	0.31	0.20	0.19	0.19	0.15	0.15	0.15
bank8nh	0.79	0.78	0.77	0.45	0.44	0.44	0.33	0.32	0.32
bank8nm	0.43	0.40	0.40	0.23	0.22	0.22	0.16	0.15	0.15
boston	0.77	0.62	0.73	0.33	0.31	0.33	0.23	0.22	0.23
comp	0.20	0.19	0.19	0.12	0.12	0.12	0.09	0.09	0.09
concreate	0.52	0.50	0.50	0.30	0.29	0.29	0.23	0.23	0.23
cooling	0.32	0.32	0.31	0.25	0.24	0.23	0.19	0.19	0.17
deltaA	0.25	0.25	0.25	0.15	0.15	0.15	0.12	0.12	0.12
deltaE	0.32	0.32	0.32	0.21	0.21	0.21	0.17	0.17	0.17
friedm	0.38	0.38	0.38	0.28	0.28	0.28	0.23	0.23	0.23
heating	0.34	0.33	0.33	0.13	0.13	0.13	0.10	0.10	0.10
istanbul	0.56	0.55	0.56	0.34	0.34	0.34	0.26	0.26	0.26
kin8fh	0.41	0.40	0.40	0.30	0.30	0.29	0.25	0.24	0.24
kin8fm	0.27	0.27	0.27	0.19	0.19	0.18	0.15	0.15	0.15
kin8nh	0.64	0.63	0.64	0.50	0.49	0.50	0.42	0.42	0.42
kin8nm	0.56	0.55	0.56	0.42	0.41	0.42	0.35	0.35	0.35
laser	0.34	0.30	0.33	0.10	0.09	0.10	0.06	0.05	0.05
mg	0.67	0.61	0.66	0.40	0.37	0.40	0.28	0.26	0.28
mortage	0.09	0.09	0.09	0.05	0.05	0.05	0.03	0.03	0.03
plastic	0.85	0.86	0.85	0.65	0.65	0.65	0.54	0.54	0.54
puma8fh	0.76	0.75	0.75	0.57	0.56	0.57	0.48	0.47	0.47
puma8fm	0.36	0.36	0.36	0.27	0.27	0.27	0.22	0.22	0.22
puma8nh	0.75	0.73	0.74	0.54	0.53	0.54	0.44	0.43	0.44
puma8nm	0.39	0.37	0.38	0.28	0.27	0.28	0.23	0.23	0.23
quakes	1.06	1.08	1.06	0.71	0.70	0.71	0.55	0.54	0.55
stock	0.18	0.18	0.18	0.11	0.11	0.11	0.09	0.09	0.09
treasury	0.12	0.12	0.12	0.06	0.06	0.06	0.03	0.03	0.03
wineRed	0.74	0.72	0.73	0.50	0.49	0.50	0.38	0.38	0.38
wineWhite	0.66	0.65	0.66	0.44	0.43	0.43	0.34	0.33	0.34
wizmir	0.15	0.15	0.15	0.08	0.08	0.08	0.07	0.07	0.07
Mean	**0.48**	**0.47**	**0.47**	**0.31**	**0.30**	**0.30**	**0.24**	**0.23**	**0.24**

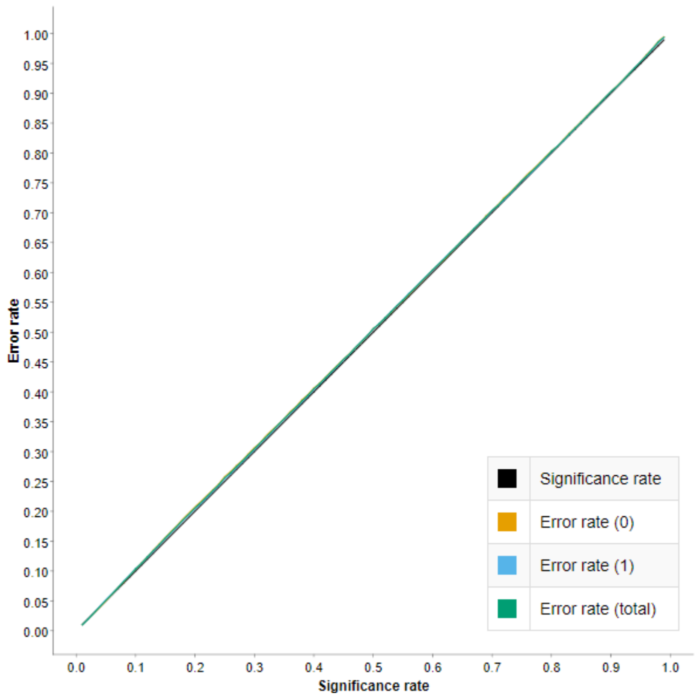


Figure 13: Error rates vs significance levels - Classification

Looking at the efficiency in Table 3, the results are organized in the same way. Efficiency are defined as the Mean Interval Size. As all targets are in the range $[0, 1]$, the interval sizes represent the per cent of the entire output range that is covered by an interval (on average). As expected, smaller significance levels result in wider prediction intervals. When being 99% confident in the prediction (i.e., when $\epsilon = 0.01$), the interval will on average cover almost $1/2$ of the target range. Reducing confidence to 95% or 90% will reduce the interval sizes to cover almost $1/3$ or $1/4$ of the target range, respectively. It is known from previous research that normalized conformal regression tend to produce smaller intervals on average. When testing differences in efficiency using a Friedman test followed by a Nemenyi post-hoc test, normalization using prediction variance result in significantly smaller prediction intervals than standard conformal regression in all three comparisons. Using polynomial regression as difficulty estimate is only significantly smaller for $\epsilon = 0.01$. No significant difference could be detected between the two normalization methods at the 0.05 significance level.

Figure 14 shows the error rate vs the significance levels for the three regression setups used. 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.

7. Conclusions

In this paper, the extended Conformal Prediction package in KNIME has been introduced and explained using a number of straight-forward use cases and experiments. The package

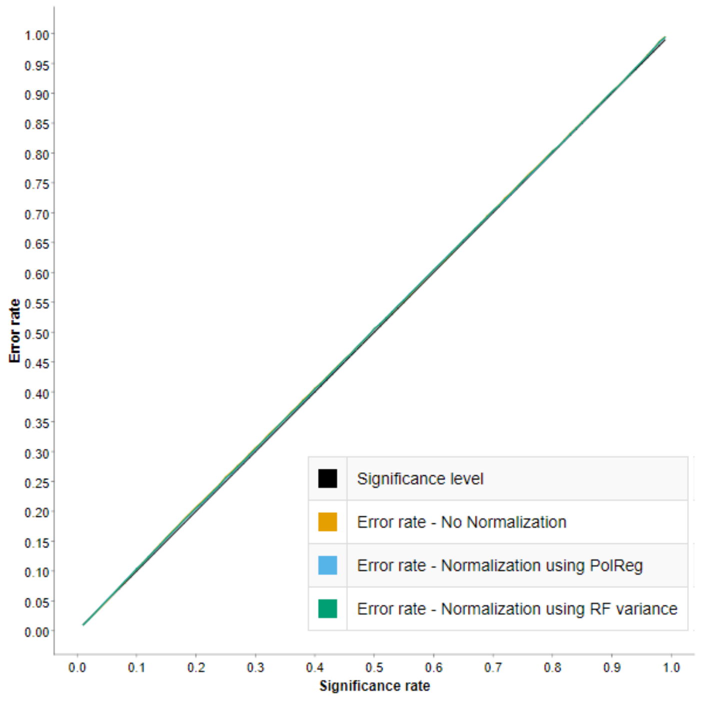


Figure 14: Error rates vs significance levels - Regression

offer the benefits of conformal prediction in an accessible and easy-to-use way while still providing advanced options for more demanding needs.

7.1. Future Work

Future planned expansions of the package include adding support for Venn and Venn-Abers predictors for classification and standard and normalized conformal predictive systems for regression. Other potential expansions could be adding support for Mondrian categories for both conformal classification, conformal regression and conformal predictive systems. Smoothed p-values for classification and interpolation are two other updates that are also on the future work list.

Combining conformal prediction with interpretable models like decision trees and rule sets have previously been proposed in many papers. Adding nodes making it possible to enrich interpretable models with conformal predictions 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. Project: AFAIR (20200223). Redfield have also contributed with time and experience in disucssions 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.

Lars Carlsson, Ernst Ahlberg, Henrik Boström, Ulf Johansson, and Henrik Linusson. Modifications to p-values of conformal predictors. In Alexander Gammerman, Vladimir Vovk, and Harris Papadopoulos, editors, *Statistical Learning and Data Sciences*, pages 251–259, Cham, 2015. Springer International Publishing. ISBN 978-3-319-17091-6.

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.

Tuve Löfström, Henrik Boström, Henrik Linusson, and Ulf Johansson. Bias reduction through conditional conformal prediction. *Intelligent Data Analysis*, 19(6):1355–1375, 2015.

Valery Manokhin and Rahul Vishwakarma. valeman/awesome-conformal-prediction: A professionally curated list of awesome conformal prediction videos, tutorials, books, papers, phd theses, articles and open-source libraries. URL https://github.com/valeman/awesome-conformal-prediction.

Ulf Norinder, Lars Carlsson, Scott Boyer, and Martin Eklund. Introducing conformal prediction in predictive modeling for regulatory purposes. a transparent and flexible alternative to applicability domain determination. 2015. doi: 10.1016/j.yrtph.2014.12.021. URL http://dx.doi.org/10.1016/j.yrtph.2014.12.021.

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

Vladimir Vovk, Ilia Nouretdinov, Valentina Fedorova, Ivan Petej, and Alex Gammerman. Criteria of efficiency for set-valued classification. *Annals of Mathematics and Artificial Intelligence*, 81(1):21–46, 2017.