





Conformal Prediction with Orange

Tomaž Hočevar 
University of Ljubljana

Blaž Zupan 
University of Ljubljana

Jonna Stålring 
Medivir AB

Abstract

Conformal predictors estimate the reliability of outcomes made by supervised machine learning models. Instead of a point value, conformal prediction defines an outcome region that meets a user-specified reliability threshold. Provided that the data are independently and identically distributed, the user can control the level of the prediction errors and adjust it following the requirements of a given application. The quality of conformal predictions often depends on the choice of nonconformity estimate for a given machine learning method. To promote the selection of a successful approach, we have developed **Orange3-Conformal**, a Python library that provides a range of conformal prediction methods for classification and regression. The library also implements several nonconformity scores. It has a modular design and can be extended to add new conformal prediction methods and nonconformities.

Keywords: conformal prediction, nonconformity, machine learning, Python, **Orange**.

1. Introduction

The application of supervised machine learning tools is growing, and an increasing number of decisions in a wide range of industries are based on computer-generated predictions. The effects of prediction errors vary between different applications. For example, forecasts of recommender systems that aim to increase sales in e-commerce are less critical than those of diagnostic systems in the intensive care unit. In applications where the reliability of individual predictions is crucial, quantification of the prediction confidence should complement the output of a machine learning method.

A range of current prediction confidence or reliability methods employs sensitivity analysis, inference of reliability from predictions of near neighbors, or explore properties of individual machine learning algorithms, such as tree variance in a random forest (Bosnić and Kononenko 2009). Conformal predictions (Vovk *et al.* 2005) are conceptually different from other reliability methods in predicting a range of values rather than reporting a confidence value. Provided

that the data are independent and identically distributed (IID), the conformal prediction framework assures that the true value lies within the predicted range at a given significance level. Consequently, the user can control the level of prediction error and adjust it following the requirements of a given application.

The inputs to conformal prediction are a set of labeled examples and a nonconformity function that measures the strangeness of a labeled example. The output is a model that for a new example predicts a set of labels with low nonconformity. The nonconformity threshold is specified by the user through the desired significance level. For example, at significance level 0.05, a conformal predictor produces a prediction region that contains the true label with probability of at least 95%. Provided IID data sets, conformal prediction methods guarantee the ratio of correct predictions under the given significance level. The approach was introduced by [Vovk *et al.* \(2005\)](#) and presented through a tutorial published in the Journal of Machine Learning Research ([Shafer and Vovk 2008](#)). It found its use in practical fields such as chemoinformatics ([Eklund *et al.* 2012](#)) and molecular biology ([Cortés-Ciriano *et al.* 2015](#)).

The quality of conformal prediction depends on the choice of nonconformity function. For example, for classification ([Shafer and Vovk 2008](#)), nonconformity can be estimated by the predicted probability of the correct class, the difference of predicted probabilities of the two most likely classes, or the distance to the closest neighbors with the same class label. For regression ([Papadopoulos *et al.* 2011](#)), a common nonconformity measure is an absolute error of a point prediction. A nonconformity threshold for a given significance is estimated from the distribution of nonconformity scores. A variety of sampling approaches, such as inductive, transductive and cross-validation methods have been proposed to estimate this distribution ([Vovk 2015](#)).

We present here **Orange3-Conformal**, a Python-based framework ([van Rossum *et al.* 2011](#)) for conformal prediction that supports various combinations of conformal prediction approaches and nonconformity scores. The library is an add-on to the **Orange** data mining framework ([Demšar *et al.* 2013](#)) and can also accommodate data and predictors from **scikit-learn** ([Pedregosa *et al.* 2011](#)). Several implemented approaches are specifically tailored to individual nonconformity scores, while others are general and can be used with any classifier or regressor. The library is designed to support applications of conformal prediction and facilitates research in novel calibration methods and nonconformities.

2. Conformal prediction

In this section we briefly introduce the theory of conformal predictions and describe the sampling methods and nonconformity measures that are implemented in the **Orange3-Conformal** library.

2.1. Conformal prediction theory

Conformal prediction theory is based on a nonconformal measure, which is a real-valued function $F(B, z)$ that measures the difference between an example z and the examples in the bag B . The example $z(\mathbf{x}, y)$ consists of a feature vector \mathbf{x} and an actual or assumed response y . For regression the response is a scalar, while for classification y can take any of the values in the set of n labels, such that $y \in \{l_1, \dots, l_n\}$. For classification, the nonconformal measure

is computed for each possible class label. In the following, the nonconformal measures will be defined for an example z and its class value y , denoted by $F(B, z)_y$.

Evaluation of the nonconformity measure for a given example and response results in a scalar nonconformity score (NCS). The p value over a set of NCSs estimates the probability that a given label y for an example z is accurate. Given the law of large numbers this probability is correct under the assumption of exchangeability (Vovk *et al.* 2005). In the application of a conformal predictor, a significance level ϵ is specified, yielding a prediction set (Γ^ϵ) including the true label with the probability $1 - \epsilon$.

For a binary classifier, the prediction set can include one label, both labels or no label, while regression conformal prediction defines the highest and lowest value of the response by identifying the y values that yield a NCS with a rank corresponding to the significance level ϵ . Hence, for a set of m NCSs, $\{\alpha_1, \dots, \alpha_m\}$, a p value for label y (Equation 1) corresponds to the fraction of NCSs greater than the NCS of the example being predicted (α^*). The label y is included in the prediction set Γ^ϵ if and only if the p value is greater than the significance level ($p \text{ value}_y > \epsilon$; Shafer and Vovk 2008). This means that there is a significant amount of “stranger” examples, therefore this label is not so unusual and we should include it.

$$p \text{ value}_y = \frac{\#\{1, \dots, m \mid \alpha_i \geq \alpha^*\}}{m} \quad (1)$$

The quality of a conformal predictor model is quantified by its validity and efficiency. A conformal predictor is valid if the true label is within the prediction range for a fraction of at least $1 - \epsilon$ of the examples in the validation set. For regression, the efficiency is described by the tightness of the prediction ranges. The tighter the interval the more efficient the conformal predictor. For classification, an individual prediction set can be empty, it can consist of a single label (singleton) or multiple labels. In an ideal case we would expect most of the predictions to be singletons. Therefore, the fraction of singletons is a feasible efficiency measure. A too low significance level ϵ or a too hard prediction problem will lead to more multiple predictions. Conversely, a too high significance level, which allows a low accuracy of the conformal predictor, can lead to empty predictions.

2.2. Sampling methods

NCSs are computed on a set of examples that are sampled by one of the three conceptually different approaches, denoted as transductive, inductive and cross conformal predictions. A transductive conformal predictor recalculates NCSs for all examples upon every prediction. Conversely, an inductive conformal predictor reuses a static set of NCSs for examples constituting a so-called calibration set, left outside of the training set. In cross conformal prediction, cross validation sampling is used to select multiple calibration sets, thereby increasing the number of available NCSs in comparison with the inductive approach. We provide some guidelines on the choice of the sampling method in Section 3.5.

2.3. Nonconformity measures

Table 1 lists different nonconformity measures for classification and regression implemented in **Orange3-Conformal**. Many of the measures are analogous between regression and classification, replacing the response in regression (y) with the predicted probability of the class

Name	Response	Type
Inverse probability	Classification	Prediction error
Probability margin	Classification	Prediction error
SVM distance	Classification	Algorithm specific
Leave-one-out classification	Classification	Normalized prediction error
KNN distance	Classification	Distance to NN
KNN fraction	Classification	NN responses
Absolute error	Regression	Prediction error
Absolute error RF	Regression	Prediction error
Absolute error normalized	Regression	Normalized prediction error
Leave-one-out regression	Regression	Normalized prediction error
Error model	Regression	Normalized prediction error
Absolute error KNN	Regression	NN responses
Average error KNN	Regression	NN responses

Table 1: Summary of nonconformity measures.

label (p_y) in classification. For a detailed description of individual nonconformities please refer to the documentation of our software.

Inverse probability (`InverseProbability`) uses an underlying classification model and the probability (p_y) the model assigns to the class label (y). Hence, the nonconformity score for label y is defined as $F(B, z)_y = 1 - p_y$.

Probability margin (`ProbabilityMargin`) is similar to `InverseProbability` but uses a normalized difference between the probability of the label (p_y) and the maximal probability of any other class.

SVM distance (`SVMDistance`) uses the distance in the space of the SVM model from the predicted example to the SVM’s decision boundary.

KNN distance (`KNNDistance`) is only defined for binary classification problems and is expressed as the ratio between the sum of distances to the set of k nearest neighbors ($N_k(z)_y$) that belong to the same class as the example z and the sum of distances to the set of neighbors in the opposite class ($N_k(z)_{label \neq y}$).

KNN fraction (`KNNFraction`) identifies k nearest neighbors of the target data instance for which the NCS is calculated ($N_k(z)$) and is expressed as a fraction of k nearest neighbors that share the class label y with the target data instance. Alternatively, the neighbors can be weighted by their distances to the target data instance.

Leave-one-out classification (`LOOClassNC`) measures the predictability of an instance from a set of its nearest neighbors. The value is normalized by the predictability of nearest neighbors. This is an experimental nonconformity score based on the study by [Toplak et al. \(2014\)](#).

Absolute error (`AbsError`) for regression models uses the difference between the predicted value \hat{y} and the response y of the example for which the NCS is calculated: $F(B, z)_y = |\hat{y} - y|$.

Absolute error RF (`AbsErrorRF`) uses the standard deviation of the predictions of all constituting trees, σ_{RF} , of an underlying random forest model. The standard deviation, together with a scaling factor, β , is used to normalize the absolute prediction error.

Error model (`ErrorModelNC`) is based on two underlying regressors. The first one is trained to predict the value of the example for which the NCS is calculated, while the second is used for predicting logarithms of the errors made by the first model, as described by Papadopoulos and Haralambous (2011).

Absolute error normalized (`AbsErrorNormalized`) uses an underlying regression model to predict the response of the target example, which is then normalized by distances and variances of the nearest neighbors as described by Papadopoulos *et al.* (2011).

Leave-one-out regression (`LOORegrNC`) is analogous to the classification version. However, instead of using the predicted probability of the correct class, it relies on the absolute error of the prediction.

Absolute error KNN (`AbsErrorKNN`) computes the difference between the average response of the k nearest neighbors, \bar{y} , and the response of the example for which the NCS is calculated: $F(B, z)_y = |\bar{y} - y|$. It is implemented with an average- and/or variance-normalization option based on the nearest neighbors.

Average error KNN (`AvgErrorKNN`) calculates an average absolute difference in responses between the example for which the NCS is calculated and the nearest neighbors: $F(B, z)_y = 1/k \sum_{z_i \in N_k(z)} |y - y_i|$.

3. The Orange3-Conformal package

We propose a Python-based implementation of conformal prediction. In the following sections, we compare it with related software packages, describe the architecture of **Orange3-Conformal** and provide an example of its use.

3.1. Related work

Several modular and extensible frameworks for conformal prediction have been proposed and vary both in a breadth of implemented functionality and in development activity. A Python library **PyCP** (Balasubramanian *et al.* 2013) implemented core techniques for classification, but the package has recently become unavailable. More functionality is provided in an R (R Core Team 2021) package **conformal** (Cortes 2016), which covers both classification and regression and was published as a side product of an application in proteomics (Cortés-Ciriano *et al.* 2015). The most extensive of the available libraries is the Python library **nonconformist** (Linusson 2018) that implements a range of conformity prediction approaches, but couples them with few nonconformity scores.

Table 2 provides a brief comparison of these three libraries. **Orange3-Conformal** is conceptually most similar to **nonconformist**, but includes a richer set of parameterized nonconformity functions. Nonconformity functions should discriminate well between conformal and nonconformal data instances for a given outcome label (Shafer and Vovk 2008). Their implementation

Library	nonconformist		PyCP		conformal		Orange3	
Nonconformities	2	2	4	0	1	1	6	7
Transductive	✗	✗	✓	✗	✗	✗	✓	✗
Inductive	✓	✓	✗	✗	✗	✗	✓	✓
Cross/aggregated	✓	✓	✗	✗	✓	✓	✓	✓
Mondrian	✓		✓		✓		✓	
Documentation	✓		✗		✓		✓	
Availability	PyPI ¹		✗		CRAN ²		PyPI ³	

Table 2: Comparison of functionalities implemented in existing conformal prediction libraries. The first column for each library corresponds to classification and the second to regression. For example, **PyCP** implements a transductive approach for classification, but not for regression. In rows, we report on the number of nonconformity scores that are available. In the second section, we indicate the availability of different approaches for the selection of calibration sets for computation of nonconformity scores (see Section 2.2). The last section provides information on documentation and availability of the software package. Notice that **PyCP** is no longer available. A mondrian setting is not applicable to regression and hence the corresponding entries are left blank.

is a critical component of conformal prediction. Their quality dictates the narrowness of the prediction range and with that the utility of the approach. **Orange3-Conformal** aims to cover a wide range of nonconformity approaches and to provide their implementation for further research on their differences and utility.

In most cases, the frameworks implement conformal prediction methods through an additional layer over underlying classification or regression models. With clearly defined approaches to nonconformity scoring, the main distinguishing factor of specific implementation is its usability. The nature of the overhead of the conformal prediction framework leaves very little room for speed improvements, which is why comparing the packages by the response time would rank them according to the speed of the employed package for machine learning.

3.2. Design

Orange3-Conformal wraps learners and classifiers from the **Orange** (Demšar *et al.* 2013) data mining suite. Besides being a pure Python-based library, **Orange** is essentially a visual programming toolbox that features a graphical user interface for the construction of interactive data analysis workflows (Curk *et al.* 2004). **Orange3-Conformal** is thus also a stepping stone towards interactive visualizations and exploratory analysis of conformal predictions.

Conformal predictions in **Orange3-Conformal** are obtained by selecting a combination of two main components:

Conformal prediction method which can be transductive (`TransductiveClassifier`), inductive (`InductiveClassifier` and `InductiveRegressor`) or cross (`CrossClassifier` and `CrossRegressor`).

¹<https://pypi.python.org/pypi/nonconformist>

²<https://CRAN.R-project.org/package=conformal>

³<https://pypi.python.org/pypi/Orange3-Conformal>

Nonconformity measure which is one of many provided measures of how unusual is a specific data instance. **Orange3-Conformal** includes general-purpose nonconformity measures like `InverseProbability`, `ProbabilityMargin` for classification, and `AbsError`, `AbsErrorNormalized` for regression. These measures work in combination with any underlying prediction model. There is also a set of measures that are predictor-specific and include, among others, `SVMDistance`, `KNNFraction`, `AbsErrorRF`, and `AvgErrorKNN`.

The library is modular with a range of implemented nonconformity measures, which can also serve as examples for those who wish to experiment and research other possible nonconformities. There are two core modules which implement methods for conformal classification (**conformal.classification**) and regression (**conformal.regression**). A separate module implements the nonconformity scores (**conformal.nonconformity**). Several evaluation scores were proposed to estimate the performance of conformal predictors (Lofstrom *et al.* 2013; Papadopoulos *et al.* 2011). For classification, these also include confidence, credibility, and confusion. Module **conformal.evaluation** includes implementations of scoring functions for regression and classification, together with sampling-based evaluation procedures. The library is accompanied by a tutorial and reference documentation (<http://orange3-conformal.readthedocs.io/>).

3.3. Example

Installation

Orange3-Conformal is an add-on for the **Orange** machine learning and data mining suite in Python and requires installation of **Orange**:

- Use the installer from the official web page (<http://orange.biolab.si/>).
- With an existing Python installation and a C/C++ compiler (Windows users can get one at <http://landinghub.visualstudio.com/visual-cpp-build-tools>) use the **pip** package manager to install **Orange3** from PyPI (Python Package Index):

```
pip install Orange3
```

- Get the latest development version from the source code repository (<https://github.com/biolab/orange3>) and follow the instructions there.

After installation try to import the **Orange** library in Python:

```
>>> import Orange
>>> print(Orange.__version__)
```

To install **Orange3-Conformal**, use **pip** that belongs to the Python instance used to run **Orange**:

```
pip install Orange3-Conformal
```

Finally, check that the installation was successful:

```
>>> import orangecontrib.conformal as cp
>>> print(cp.__version__)
```


We will also set a random seed to make the demonstration replicable:

```
>>> import numpy as np
>>> np.random.seed(12345)
```

Classification

Here we demonstrate how to use an inductive conformal classifier with the nonconformity score computed as the inverse probability of logistic regression. We illustrate its use on the Iris data set from [Anderson \(1936\)](#). This classic data set describes 150 iris flowers with four features (sepal length, sepal width, petal length, petal width) and their species (Iris Setosa, Iris Versicolor, Iris Virginica), which is the class variable.

First, we need to import the libraries:

```
>>> import Orange
>>> import orangecontrib.conformal as cp
```

Next, we load the data set, split it into a train, calibration and test set, and extract a single data instance for which we will make a conformal prediction:

```
>>> iris = Orange.data.Table('iris')
>>> train, test = next(cp.evaluation.RandomSampler(iris, 1, 1))
>>> train, calibrate = next(cp.evaluation.RandomSampler(train, 1, 1))
>>> test_instance = Orange.data.Instance(test[-1])
>>> print(test_instance)
```

```
[4.7, 3.2, 1.6, 0.2 | Iris-setosa]
```

We use a `LogisticRegressionLearner` and the inverse probability nonconformity score in an inductive conformal prediction classifier:

```
>>> lr = Orange.classification.LogisticRegressionLearner()
>>> ip = cp.nonconformity.InverseProbability(lr)
>>> icp = cp.classification.InductiveClassifier(ip, train, calibrate)
```

Two prediction regions are obtained at the 0.1 and 0.02 significance levels:

```
>>> print(test_instance.get_class())
>>> print(icp(test_instance, 0.1))
>>> print(icp(test_instance, 0.02))
```

```
Iris-setosa
['Iris-setosa']
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
```

We can see that at the 0.1 significance level only the correct class of `Iris-setosa` was predicted (singleton prediction). By changing the significance level to 0.02, which implies a

lower tolerance for errors, the model loosens its prediction to a multiple prediction with three possible classes, *Iris-setosa*, *Iris-versicolor*, and *Iris-virginica*.

There are 16 multiple predictions at 0.05 significance from the test set consisting of 75 instances. All these cases belong to the boundary region between the *Iris-virginica* and *Iris-versicolor* classes, while the instances from the *Iris-setosa* class are assigned a correct singleton prediction.

```
>>> multiple = []
>>> for instance in test:
>>>     prediction = icp(instance, 0.05)
>>>     if len(prediction) > 1:
>>>         multiple.append((instance, prediction))
>>> print(len(multiple), len(test))
>>> for instance, prediction in multiple:
>>>     print(instance, '->', prediction)

16 75
[6.1, 2.9, 4.7, 1.4 | Iris-versicolor] -> ['Iris-versicolor',
'Iris-virginica']
[7.4, 2.8, 6.1, 1.9 | Iris-virginica] -> ['Iris-versicolor',
'Iris-virginica']
...
[7.2, 3.0, 5.8, 1.6 | Iris-virginica] -> ['Iris-versicolor',
'Iris-virginica']
```

Evaluating the conformal predictor on a given train and test set confirms that the predicted regions at 0.1 significance are indeed correct approximately 90% of the time. Besides, 95% of the predictions consist of a single value, 5% are empty predictions and there are no multiple predictions.

```
>>> res = cp.evaluation.run_train_test(icp, 0.1, train, test, calibrate)
>>> print(res.accuracy())
>>> print(res.singleton_criterion())
>>> print(res.empty_criterion())
>>> print(res.multiple_criterion())

0.93333
0.94667
0.05333
0.0
```

See Figure 1 for a visualization of predictions made by the conformal predictor on the test set. As expected, the mistakes and non-singleton predictions appear on the boundary between different class values.

Another interesting data set for demonstration purposes is the handwritten digits data set (<https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+>

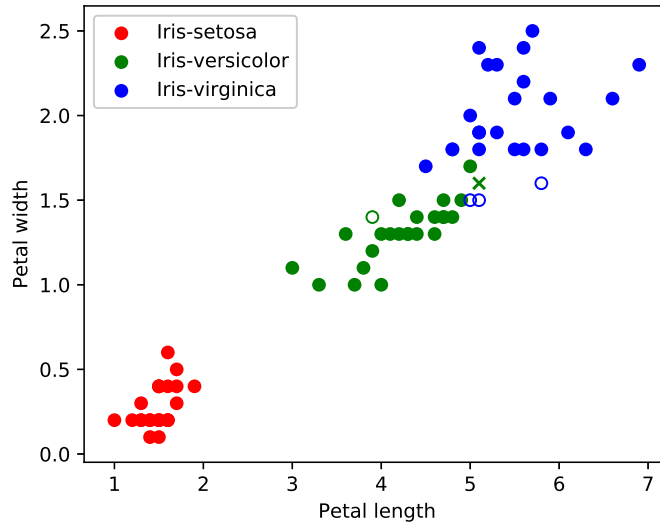


Figure 1: Visualization of conformal prediction on the test set. Examples are colored according to their originally assigned class value. Filled circles indicate a correct singleton prediction, crosses indicate wrong singleton predictions and empty circles indicate empty prediction sets. There were no multiple predictions at the significance level of 0.1.

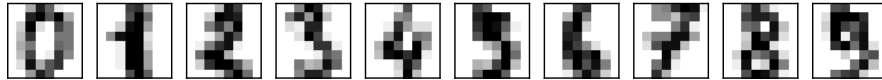


Figure 2: Examples of handwritten digits.

Digits) available from the UCI Machine Learning Repository (Dua and Graff 2017). The data set consists of 8×8 images of handwritten digits with pixel values in the range $[0, 16]$ (Figure 2). For brevity, we omit the parts of the code that import libraries and draw visualizations. Please see the replication code supplement for a complete example.

A subset of the digits data set is available from **scikit-learn**. We load the data set, convert it to **Orange** format and normalize the values to a $[0, 1]$ range.

```
>>> digits = sklearn.datasets.load_digits()
>>> domain = Domain([ContinuousVariable(f"A{i}") for i in range(64)],
... DiscreteVariable("digit", map(str, range(10))))
>>> tab = Table(domain, digits.data, digits.target)
>>> normalizer = Normalize(zero_based = True,
... norm_type = Normalize.NormalizeBySpan)
>>> tab = normalizer(tab)
```

Support vector machines (SVM) perform well on the handwritten digits recognition problem. We will use an inductive conformal prediction with probabilities of a SVM model as nonconformity scores. The training and testing set consist of 1198 and 599 examples, respectively. The `run_train_test` method reserves one third of the training set for calibration purposes. Promised accuracy is close to the expected 0.97, and there are only 12 multiple predictions.

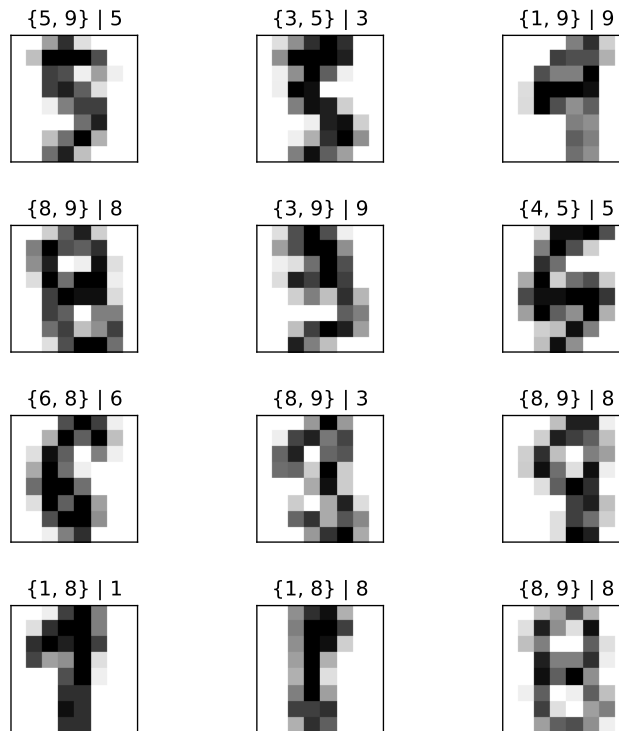


Figure 3: Digits where the conformal method predicted more than one class. In the annotation of each image, the figure shows the set of predicted classes and the true class, separated by ‘|’.

```
>>> learner = SVMLearner(probability = True)
>>> nc = cp.nonconformity.InverseProbability(learner)
>>> ic = cp.classification.InductiveClassifier(nc)
>>> train, test = next(cp.evaluation.RandomSampler(tab, 2, 1))
>>> r = cp.evaluation.run_train_test(ic, 0.03, train, test)
>>> multi = []
>>> for example, pred, ref in zip(test, r.preds, r.refs):
...     if len(pred.classes()) >= 2:
...         multi.append((example, pred.classes(), ref))
>>> print(r.accuracy(), len(multi))
```

0.97329 12

The list `multi` contains the necessary data about the image, predicted classes, and the true class from which we can visualize these uncertain examples as in Figure 3. By visual inspection of these examples, it is easy to agree with the uncertainty of the conformal predictions as all the proposed classes look likely. Interestingly, looking at the images from farther away, humans can determine the right digit in most of these cases.

Regression

Here we demonstrate the use of **Orange3-Conformal** on the regression problem of Boston house prices (<https://archive.ics.uci.edu/ml/machine-learning-databases/housing/>, Harrison and Rubinfeld 1978) and show the interaction with the **scikit-learn** library. The Boston house prices data set consists of 506 instances of Boston towns or suburbs each described by 13 features such as the crime rate per capita, average number of rooms per dwelling and pupil-teacher ratio by town. The target variable is the median value of owner-occupied homes in thousands of dollars (values range from 5 to 50).

The data set is available in **scikit-learn**, which we split into the training and testing set. We train a nearest neighbor regressor on the training set and evaluate it on the testing set.

```
>>> from sklearn import datasets
>>> from sklearn.metrics import mean_absolute_error
>>> boston = datasets.load_boston()
>>> mask = np.random.choice([False, True], len(boston.data))
>>> X_train, y_train = boston.data[mask], boston.target[mask]
>>> X_test, y_test = boston.data[~mask], boston.target[~mask]
>>> from sklearn.neighbors import KNeighborsRegressor
>>> neigh = KNeighborsRegressor(n_neighbors = 10)
>>> neigh.fit(X_train, y_train)
>>> pred = neigh.predict(X_test)
>>> print(mean_absolute_error(y_test, pred))
```

5.59316

To use a conformal prediction method on the Boston housing data, we have to convert the **scikit-learn**'s data set to **Orange** format. For conversion we do not have to specify the domain (use `None`) and let the library construct appropriate variables from the data and target arrays of the **scikit-learn**'s data set. To use the constructed nearest neighbors regressor, we simply reshape the **Orange**'s row instance to 2D and employ the previously constructed regressor from **scikit-learn**.

```
>>> import Orange
>>> data = Orange.data.Table.from_numpy(None, boston.data, boston.target)
>>> train, test = data[mask], data[~mask]
>>> test_instance = test[-1]
>>> print(test_instance)
>>> print(neigh.predict(test_instance.x.reshape((1, -1))))
```

```
[0.04527, 0, 11.93, 0, 0.573, ..., 9.08 | 20.6]
[22.74]
```

Now we are ready to set up a conformal predictor. As a nonconformity measure, we will use a non-normalized absolute error of 10 nearest neighbors determined by the Euclidean distance. To demonstrate another method, we will use this nonconformity in a cross conformal setting with 5 folds. Usually, we would employ a cross conformal instead of an inductive predictor

in situations where the data set is too small for a fixed partition into separate training, calibration and testing sets. Predicting the test instance with stronger (smaller) significance levels results in wider prediction ranges. Note that, in this case, the ranges are centered on the same value that was predicted by **scikit-learn**'s nearest neighbors regressor (22.74).

```
>>> import orangecontrib.conformal as cp
>>> from Orange.distance import Euclidean
>>> nc = cp.nonconformity.AbsErrorKNN(Euclidean(), 10)
>>> ccp = cp.regression.CrossRegressor(nc, 5, train)
>>> print(ccp(test_instance, 0.1))
>>> print(ccp(test_instance, 0.05))
```

```
(13.79, 31.69)
(10.66, 34.82)
```

To evaluate our conformal predictor at a 0.1 significance level more thoroughly, we will randomly sample the training and testing set in ratio 3:2 and repeat the entire experiment three times. The accuracy is close to 0.9, as expected, and the average width of the predicted range is 21.6 (i.e., about ± 11).

```
>>> res = cp.evaluation.run(ccp, 0.1,
...   cp.evaluation.RandomSampler(data, 3, 2), rep = 3)
>>> print(res.accuracy())
>>> print(res.median_range())
```

```
0.88834
21.6
```

3.4. Nonconformity scores

In this section we use an artificial data set proposed in [Friedman \(1991\)](#) to illustrate the advantages of different nonconformity scores. The Friedman data set is available from the **KEEL** repository (<http://keel.es/datasets.php>, [Alcalá-Fdez et al. 2011](#)).

It is defined by

$$y = \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon, \quad (2)$$

with $\epsilon \sim \mathcal{N}(0, 1)$ and where y is the dependent variable and x_1, \dots, x_5 are the independent variables drawn from a uniform distribution over $[0.0, 1.0]$ (Figure 4). $\mathcal{N}(0, 1)$ indicates a random noise from a zero-mean normal distribution with variance 1.

We experimented with six different nonconformity scores to demonstrate their differences. Refer to Section 2.3 for their descriptions. Some of them depend on underlying regressors for which we used a linear regression (LR) and a K -nearest neighbors (KNN) model (we used $k = 20$ in all such cases). We split the data set (1200 instances) into training (907 instances) and testing set (293 instances) in a 3:1 ratio. Because we used an inductive conformal regressor, we set aside one half (453 instances) of the training set for calibration. The output of conformal regressors for every instance is a range of predicted values. We observed the median width

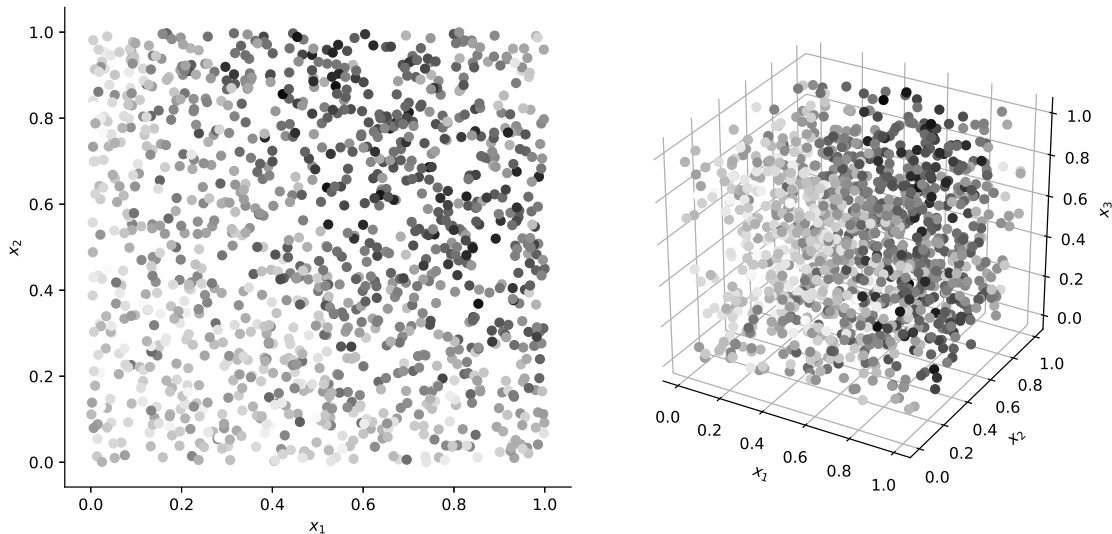


Figure 4: Scatter plot showing the distribution of the dependent variable with respect to the first two or three independent variables (left and right image, respectively). Darker points indicate larger values of the dependent variable.

$\epsilon = 0.1$	Accuracy	Median range	Interdecile range
AbsError (LR)	0.860	8.429	0.000
AbsError (KNN)	0.887	7.403	0.000
AbsErrorNormalized (LR)	0.857	8.268	1.442
AvgErrorKNN	0.877	7.957	1.715
ErrorModelNC (LR, KNN)	0.853	7.771	4.389
LOORegrNC (LR)	0.881	5.161	3.948

Table 3: Differences among nonconformity scores.

of predicted ranges (median range) in the test set and an interdecile range, which is the difference in widths of predicted ranges of the first and ninth decile.

To observe the interdecile range, we performed a single experiment, although the accuracy of all nonconformities is slightly below 0.9, where we would expect it to be. Several repetitions of the experiment should remedy that but would invalidate the interdecile range measure. The **AbsError** method always outputs a range of the same width; therefore, the interdecile range is zero. **AbsError** based on the K -nearest neighbors model outperforms LR in terms of the median range. This is probably because the data set has a non-linear structure but is locally homogeneous, which benefits the KNN model. Other nonconformity scores output ranges of different widths to output wider ranges for difficult or unusual instances while keeping the ranges narrow in the other (normal) cases. **AbsErrorNormalized**, **AvgErrorKNN** and **ErrorModelNC** all use the nearest neighbors in some way and improve the median range results of the **AbsError** with LR. The best performing method regarding the median range is the new **LOORegrNC** nonconformity measure which is based on the predictability of an instance from its neighborhood, in our case using an LR model. A possible interpretation is that the method discovers non-homogeneous regions and successfully adapts the width of the predicted

range. However, note that this comes at the cost of a larger interdecile range compared to some other nonconformities.

3.5. Best practice guidelines

Here, we propose some guidelines for applying the conformal prediction framework and the presented **Orange3-Conformal** package.

We first have to choose a conformal prediction method among transductive, inductive, and cross approaches for a given data set. The transductive prediction has the strongest theoretical foundation but is by far the most computationally demanding. In practice, we would use transduction on small data sets that include, for example, at most few hundred instances. The results of transductive conformal predictions are similar to the other two approaches unless an addition of a single labeled data instance can have a substantial impact on the trained model. On the other hand, the inductive method is the most practical due to its efficiency. It should be applied where the data set is sufficiently large to be split into a training and calibration set. In-between transductive and inductive approach is the cross prediction method, which we can use when the data set is too big for transductive prediction and too small for a sufficiently representative split into a training and calibration data set required by inductive prediction. The cross method reuses the training set similarly as the well-known cross validation approach in machine learning.

It is difficult to generalize about the success of various nonconformity scores, and instead – given a specific data set – a range of nonconformity methods should be evaluated and scored. For this reason, **Orange3-Conformal** implements many of the nonconformity methods from the current literature. These methods are conceptually different, rely on various properties of the data and thereby complement each other. The NCSs based on prediction error (**AbsError**) and model probabilities (**InverseProbability**) are the simplest and least computationally demanding, therefore providing a good starting point. Many of the methods use properties of the nearest neighbors such as responses, predicted values, and distances. Some NCS approaches develop local leave-one-out models over a broader set of nearest neighbors, increasing the computational time. The methods based on error models require an additional model of the full data set predicting the prediction error. While we suggest comparing all available nonconformity scores, the choice for inclusion may also depend on the available time and computational resources.

Our library may aspire to provide broader access to conformal predictions and multiple NCSs within the Python modeling community and its application might aid in future development of generalized guidelines for the selection of NCS. The application of the outlined best practices is illustrated by the case study in the following section, exploring the NCSs for classification.

4. Case study: AMES mutagenicity

Assessment of the carcinogenic potential of new molecular structures is vital in several industries, including those from chemistry, cosmetics, and pharmaceuticals. Within the pharmaceutical industry, the tolerance for compounds with carcinogenic potential is very low, and it is crucial to identify such compounds early within the drug discovery process to terminate their development and reduce costs. For a long period of time, the AMES mutagenicity test has been the first in vitro assay routinely used within the pharmaceutical industry to

gauge carcinogenicity. For an extensive review of this method refer to [Mortelmans and Zeiger \(2000\)](#). The widespread use of the AMES test has resulted in its standardization, promoted by the Organization for Economic Co-operation and Development (OECD) and the International Commission on Harmonization. The standardization improves the uniformity of test procedures between laboratories and enables a comparison of data originating from different laboratories.

4.1. Preliminaries

The AMES assay uses several strains of Salmonella bacteria to detect a range of genetic damage caused by chemical substances. The Salmonella strains included in the assay have preexisting mutations that prevent these strains from producing the amino acid histidine required for the bacteria to grow and form colonies. Chemicals with a tendency to cause mutations might revert this inability to produce histidine, which would be indicated by a promoted growth and colony formation compared to the negative controls. Prudent treatment of mutagenicity requires a compound to be categorized as mutagenic if it promotes growth in any of the Salmonella strains.

Because of the routine use of the AMES test within several industries across an extended period, a substantial amount of AMES data has accumulated in the public domain. Also, most larger pharmaceutical companies have AMES mutagenicity data sets from multiple projects, supporting the inference of predictive models of the outcome of the AMES in vitro assay.

A computational model that uses only the information about the molecular structure has the potential to filter out the compounds with an increased risk for mutagenicity before they are considered to be synthesized. The cost of synthesis and testing of drug candidates is substantial, and a model predicting AMES mutagenicity could not only save time and money but also increase the quality of the pursued compounds. However, false negative predictions could result in investments in poor compounds, while an incorrect positive prediction might rule out the next blockbuster drug. Because of the potential significant harm of wrong predictions, methods assessing not only the average accuracy of the model but also the confidence in individual predictions are highly valuable.

In drug discovery, new chemical entities are continuously explored, resulting in the violation of the fundamental requirement of IID data. In the quantitative structure-activity relationship (QSAR) literature, such compounds are considered outside of the so-called applicability domain of the model, and for such compounds, the validation statistics do not apply nor do model-based confidence prediction methods. Consequently, in drug discovery, a model-based prediction confidence method should be complemented by an assessment of the IID assumption between the training set and the test set. Once the IID assumption has been assessed, application of the conformal prediction framework, reliant on the IID criterion, provides additional information about the extent of the “strangeness” of the example. Furthermore, using the framework of conformal prediction assesses not only the prediction confidence but the error of the model can be controlled, facilitating risk analysis and planning in drug discovery projects.

[Hansen *et al.* \(2009\)](#) have compiled an AMES mutagenicity data set from several public data sources, removing duplicates and compounds occurring in several data sets with contradictory classification. This data set contains the molecular structure of about 6500 compounds,

Classifier	CA	AUC	Precision	Recall	F1
Logistic regression	0.748	0.819	0.773	0.752	0.762
Naive Bayes	0.669	0.721	0.706	0.658	0.681
Nearest neighbors	0.775	0.843	0.788	0.796	0.792
Random forest	0.765	0.839	0.813	0.731	0.770
Support vector machine	0.772	0.772	0.799	0.771	0.785

Table 4: Accuracy scores of classifiers on the AMES data set.

together with their AMES classification as positive or negative. The data set is reasonably balanced containing about 3500 positive and 3000 negative compounds.

Developing an empirical model based solely on information about chemical structure to predict a molecular property is referred to as QSAR modeling. As the chemical structure cannot be directly used for machine learning, it is instead represented numerically by a vector of so-called molecular descriptors. For an example of conformal prediction application to QSAR modeling, see [Eklund *et al.* \(2012\)](#). In our experiment, the set of 177 well established RD-Kit descriptors ([Landrum 2006](#), Release_2012_12_1) were used as features to represent the molecular structure. This descriptor set contains the counts of various functional groups, molecular indices and calculated bulk properties such as lipophilicity and charge.

4.2. Experiments

We preprocessed the AMES data with QSAR features. We removed eleven features that were the same for all instances and one that contained spurious values. The resulting data set (available in the supplementary material) contains 6494 chemicals described by 165 features. We used an arbitrarily selected 80% (5220) of the instances for the training and the remaining 20% (1274) for the testing set.

To estimate the difficulty of the classification problem that we are presented with, we first evaluated several standard classification models (Table 4). We used the default parameters of each method. The observed values for each classifier are its classification accuracy (CA), the area under the ROC curve (AUC), precision, recall, and the F1 score. We can see that the classifiers obtain comparable results except for Naive Bayes, which under-performs. It relies on the independence of features, which does not seem to be the case in this problem. The obtained AUC values are also similar to those previously reported ([Hansen *et al.* 2009](#)); we attribute slight differences observed to different molecular descriptors and classifier parameters. From here on, we use logistic regression as our choice of a classification model, also because of its computational efficiency.

We will use the inductive conformal prediction approach, which requires a separate calibration set. For this purpose, we further split the training set in half into an actual training set (2610 instances) and a calibration set (2610 instances).

To confirm that the conformal predictor performs as expected, we construct a validation plot (Figure 5). We measure the predictor’s accuracy at several significance levels and check whether the observed error rate equals the specified significance level. Ideally, we would get a straight line with a slope of -1 . For this purpose, we used an `InductiveClassifier` based on the `InverseProbability` nonconformity score of the underlying logistic regression model. Besides validity, we are mostly interested in the efficiency of conformal predictors ([Vovk *et al.*](#)

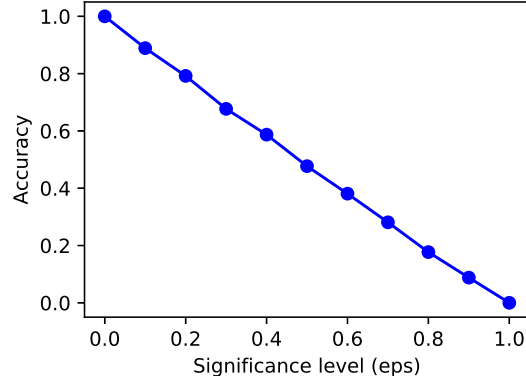


Figure 5: A valid conformal classifier should exhibit observed accuracy equal to $1 - \textit{eps}$, where \textit{eps} is the specified significance level.

$\epsilon = 0.05$	Accuracy	Singletons	Correct singletons	Multiple	Empty
InverseProbability	0.941	0.405	0.855	0.595	0.000
ProbabilityMargin	0.941	0.405	0.855	0.595	0.000
SVMDistance	0.947	0.458	0.883	0.542	0.000
KNNDistance	0.960	0.389	0.897	0.611	0.000
KNNFraction	0.951	0.452	0.891	0.548	0.000
LOOClassNC	0.942	0.440	0.877	0.556	0.004
$\epsilon = 0.1$	Accuracy	Singletons	Correct singletons	Multiple	Empty
InverseProbability	0.889	0.614	0.820	0.386	0.000
ProbabilityMargin	0.889	0.614	0.820	0.386	0.000
SVMDistance	0.903	0.641	0.849	0.359	0.000
KNNDistance	0.908	0.611	0.850	0.389	0.000
KNNFraction	0.904	0.627	0.847	0.373	0.000
LOOClassNC	0.905	0.597	0.851	0.397	0.006
$\epsilon = 0.2$	Accuracy	Singletons	Correct singletons	Multiple	Empty
InverseProbability	0.792	0.884	0.765	0.116	0.000
ProbabilityMargin	0.792	0.884	0.765	0.116	0.000
SVMDistance	0.794	0.922	0.777	0.078	0.000
KNNDistance	0.808	0.883	0.783	0.117	0.000
KNNFraction	0.810	0.874	0.783	0.126	0.000
LOOClassNC	0.806	0.832	0.812	0.130	0.038

Table 5: Efficiency of conformal classifiers.

2016). In classification problems, we can measure this as the number of singleton predictions (prediction sets of size one), the ratio of correct singleton predictions, number of multiple and empty predictions (Table 5).

Conformal classifier	Confidence	Credibility
<code>InverseProbability</code>	0.910	0.580
<code>ProbabilityMargin</code>	0.910	0.580
<code>SVMDistance</code>	0.918	0.572
<code>KNNDistance</code>	0.910	0.569
<code>KNNFraction</code>	0.911	0.591
<code>LOOClassNC</code>	0.907	0.583

Table 6: Confidence and credibility of conformal classifiers.

We can observe the same results for inverse probability and probability margin nonconformities. These two nonconformities are equivalent in binary classification problems. The measured accuracy corresponds well to the specified significance level (error rate) ϵ . As we increase ϵ , the number of singleton predictions grows, while the number of predictions containing multiple classes declines. There are almost no cases with an empty set of predicted classes, which would be assigned to instances too unusual for every possible class under the given significance level. The purpose of nonconformity measures is to distinguish between easy and difficult or unusual instances. The results for different nonconformity scores are quite similar in this case with the `SVMDistance` standing out slightly.

Because of the importance of correct predictions in drug discovery projects, high confidence in the predictions, at the expense of uninformative predictions, is often preferable. Selecting the 0.05 significance level and the `KNNFraction` method would result in 54.8% of the predictions being uninformative. However, the accuracy of the singleton predictions would be 89.1% compared to the classification accuracy of the logistic regression model that is 74.8%. Many drug discovery projects consider an accuracy level of 90% as sufficient and could discard about 55% of the compounds from expensive experimental testing.

Table 6 presents conformal classifier’s average confidence and credibility, which are measures that are independent of the specified significance level. Confidence of a single prediction is equal to $1 - \epsilon_1$, where ϵ_1 represents the minimum significance level that would still result in a prediction of a single label. Hence, for `KNNFraction` nonconformity over the AMES test data set, we can exclude the least probable class with an average certainty of 91%. Similarly, the credibility of a prediction is the minimum ϵ that would result in an empty prediction set. Small credibility indicates an unusual example. For a complete introduction to confidence and credibility in conformal prediction refer to [Saunders *et al.* \(1999\)](#).

5. Conclusion

We have proposed a Python-based implementation of several conformal prediction methods. Conformal prediction is an interesting formal statistical approach that addresses the reliability problem in classification and regression, and has already been used in various applications in natural sciences. Compared to other software packages, **Orange3-Conformal** covers a wider range of nonconformity scores and is designed in a modular way to foster experimentation with new approaches and scoring techniques. Our implementation has purposely been implemented within the **Orange** data mining framework, opening a possibility of extending the implemented functionality with a graphical user interface and incorporating it with **Orange**’s interactive data analytics framework.

Acknowledgments

This work was supported by the Slovenian Research Agency (ARRS, grant number P2-0209) and by a research grant from Medivir AB.

References

- Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011). “**KEEL** Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework.” *Journal of Multiple-Valued Logic and Soft Computing*, **17**(2–3), 255–287.
- Anderson E (1936). “The Species Problem in Iris.” *The Annals of the Missouri Botanical Garden*, **23**(3), 457–509. doi:10.2307/2394164.
- Balasubramanian VN, Baker A, Yanez M, Chakraborty S, Panchanathan S (2013). “**PyCP**: An Open-Source Conformal Predictions Toolkit.” In *Artificial Intelligence Applications and Innovations*, pp. 361–370. Springer-Verlag. doi:10.1007/978-3-642-41142-7_37.
- Bosnić Z, Kononenko I (2009). “An Overview of Advances in Reliability Estimation of Individual Predictions in Machine Learning.” *Intelligent Data Analysis*, **13**(2), 385–401.
- Cortes I (2016). **conformal**: *Conformal Prediction for Regression and Classification*. R package version 0.2, URL <https://CRAN.R-project.org/package=conformal>.
- Cortés-Ciriano I, Bender A, Malliavin T (2015). “Prediction of PARP Inhibition with Proteochemometric Modelling and Conformal Prediction.” *Molecular Informatics*, **34**(6–7), 357–366. doi:10.1002/minf.201400165.
- Curk T, Demsar J, Xu Q, Leban G, Petrovic U, Bratko I, Shaulsky G, Zupan B (2004). “Microarray Data Mining with Visual Programming.” *Bioinformatics*, **21**(3), 396–398. doi:10.1093/bioinformatics/bth474.
- Demšar J, Curk T, Erjavec A, Gorup Č, Hočevar T, Milutinovič M, Možina M, Polajnar M, Toplak M, Starič A, Štajdohar M, Umek L, Žagar L, Žbontar J, Žitnik M, Zupan B (2013). “**Orange**: Data Mining Toolbox in Python.” *Journal of Machine Learning Research*, **14**(35), 2349–2353. URL <http://jmlr.csail.mit.edu/papers/v14/demsar13a.html>.
- Dua D, Graff C (2017). “UCI Machine Learning Repository.” URL <https://archive.ics.uci.edu/ml>.
- Eklund M, Norinder U, Boyer S, Carlsson L (2012). “Application of Conformal Prediction in QSAR.” In L Iliadis, I Maglogiannis, H Papadopoulos, K Karatzas, S Sioutas (eds.), *Artificial Intelligence Applications and Innovations: AIAI 2012*, pp. 166–175. Springer-Verlag. doi:10.1007/978-3-642-33412-2_17.
- Friedman JH (1991). “Multivariate Adaptive Regression Splines.” *The Annals of Statistics*, **19**(1), 1–67. doi:10.1214/aos/1176347963.

- Hansen K, Mika S, Schroter T, Sutter A, ter Laak A, Stefer-Hartmann T, Heinrich N, Müller KR (2009). “Benchmark Data Set for in Silico Prediction of Ames Mutagenicity.” *Journal of Chemical Information and Modelling*, **49**(9), 2011–2081. doi:10.1021/ci900161g.
- Harrison D, Rubinfeld DL (1978). “Hedonic Housing Prices and the Demand for Clean Air.” *Journal of Environmental Economics and Management*, **5**(1), 81–102. doi:10.1016/0095-0696(78)90006-2.
- Landrum G (2006). “RDKit: Open-Source Cheminformatics.” URL <http://www.rdkit.org>.
- Linusson H (2018). *nonconformist: Python Implementation of the Conformal Prediction Framework*. URL <https://github.com/donlnz/nonconformist>.
- Lofstrom T, Johansson U, Bostrom H (2013). “Effective Utilization of Data in Inductive Conformal Prediction Using Ensembles of Neural Networks.” In *The 2013 International Joint Conference on Neural Networks (IJCNN)*. doi:10.1109/IJCNN.2013.6706817.
- Mortelmans K, Zeiger E (2000). “The Ames Salmonella/Microsome Mutagenicity Assay.” *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, **455**(1–2), 29–60. doi:10.1016/S0027-5107(00)00064-6.
- Papadopoulos H, Haralambous H (2011). “Reliable Prediction Intervals with Regression Neural Networks.” *Neural Networks*, **24**(8), 842–851. doi:10.1016/j.neunet.2011.05.008.
- Papadopoulos H, Vovk V, Gammerman A (2011). “Regression Conformal Prediction with Nearest Neighbours.” *Journal of Artificial Intelligence Research*, **40**(1), 815–840. doi:10.1613/jair.3198.
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011). “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research*, **12**(85), 2825–2830. URL <http://www.jmlr.org/papers/v12/pedregosa11a.html>.
- R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Saunders C, Gammerman A, Vovk V (1999). “Transduction with Confidence and Credibility.” In T Dean (ed.), *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, volume 2, pp. 722–726. Morgan Kaufmann.
- Shafer G, Vovk V (2008). “A Tutorial on Conformal Prediction.” *Journal of Machine Learning Research*, **9**(12), 371–421. URL <http://www.jmlr.org/papers/v9/shafer08a.html>.
- Toplak M, Močnik R, Polajnar M, Bosnić Z, Carlsson L, Hasselgren C, Demšar J, Boyer S, Zupan B, Stålring J (2014). “Assessment of Machine Learning Reliability Methods for Quantifying the Applicability Domain of QSAR Regression Models.” *Journal of Chemical Information and Modeling*, **54**(2), 431–441. doi:10.1021/ci4006595.
- van Rossum G, et al. (2011). *Python Programming Language*. URL <https://www.python.org/>.

- Vovk V (2015). “Cross-Conformal Predictors.” *The Annals of Mathematics and Artificial Intelligence*, **74**(1–2), 9–28. doi:10.1007/s10472-013-9368-4.
- Vovk V, Fedorova V, Nouretdinov I, Gammerman A (2016). “Criteria of Efficiency for Conformal Prediction.” In A Gammerman, Z Luo, J Vega, V Vovk (eds.), *Conformal and Probabilistic Prediction with Applications*, pp. 23–39. Springer-Verlag, Cham. doi:10.1007/978-3-319-33395-3_2.
- Vovk V, Gammerman A, Shafer G (2005). *Algorithmic Learning in a Random World*. Springer-Verlag, Boston. doi:10.1007/b106715.

Affiliation:

Tomaž Hočevar
Faculty of Computer and Information Science
University of Ljubljana
Večna pot 113
1000 Ljubljana, Slovenia
E-mail: tomaz.hocevar@fri.uni-lj.si
URL: <https://fri.uni-lj.si/en/employees/tomaz-hocevar>