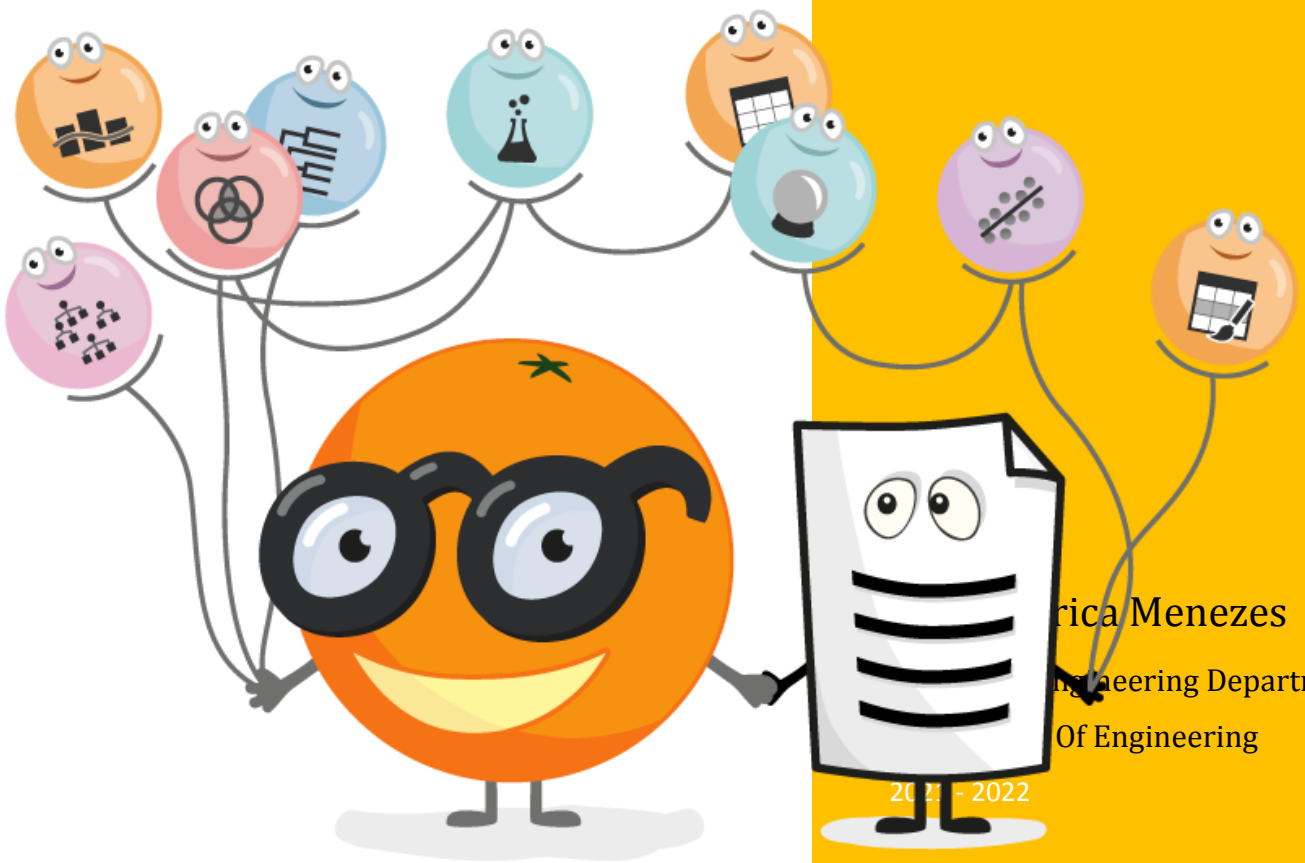
2021 - 2022

DATA MINING USING ORANGE



rica Menezes

ngineering Department

Of Engineering

2021 - 2022

1. : Introduction to Orange

Orange is a machine learning and data mining suite for data analysis through Python scripting and visual programming. Here we report on the scripting part, which features interactive data analysis and component-based assembly of data mining procedures. In the selection and design of components, we focus on the flexibility of their reuse: our principal intention is to let the user write simple and clear scripts in Python, which build upon C++ implementations of computationally intensive tasks. Orange is intended both for experienced users and programmers, as well as for students of data mining.

Orange library is a hierarchically organized toolbox of data mining components. The low-level procedures at the bottom of the hierarchy, like data filtering, probability assessment and feature scoring, are assembled into higher-level algorithms, such as classification tree learning. This allows developers to easily add new functionality at any level and fuse it with the existing code. The main branches of the component hierarchy are:

- **data management and preprocessing** for data input and output, data filtering and sampling, imputation, feature manipulation (discretization, continuization, normalization, scaling and scoring), and feature selection
- **classification** with implementations of various supervised machine learning algorithms (trees, forests, instance-based and Bayesian approaches, rule induction), borrowing from some well-known external libraries such as LIBSVM (Chang and Lin, 2011)
- **regression** including linear and lasso regression, partial least square regression, regression trees and forests, and multivariate regression splines
- **association** for association rules and frequent itemsets mining,
- **ensembles** implemented as wrappers for bagging, boosting, forest trees, and stacking
- **clustering**, which includes k-means and hierarchical clustering approaches
- **evaluation** with cross-validation and other sampling-based procedures, functions for scoring the quality of prediction methods, and procedures for reliability estimation,
- **projections** with implementations of principal component analysis, multi-dimensional scaling and self-organizing maps.

The library is designed to simplify the assembly of data analysis workflows and crafting of data mining approaches from a combination of existing components. Besides broader range of features, Orange differs from most other Python-based machine learning libraries by its maturity (over 15 years of active development and use), a large user community supported through an active forum, and extensive documentation that includes tutorials, scripting examples, data set repository, and documentation for developers. Orange scripting library is also a foundation for its visual programming platform with graphical user interface components for interactive data visualization. [1]

Download link: https://orangedatamining.com/download/#windows

2. : Widgets Catalog

2.1.Data

2.1.1. File

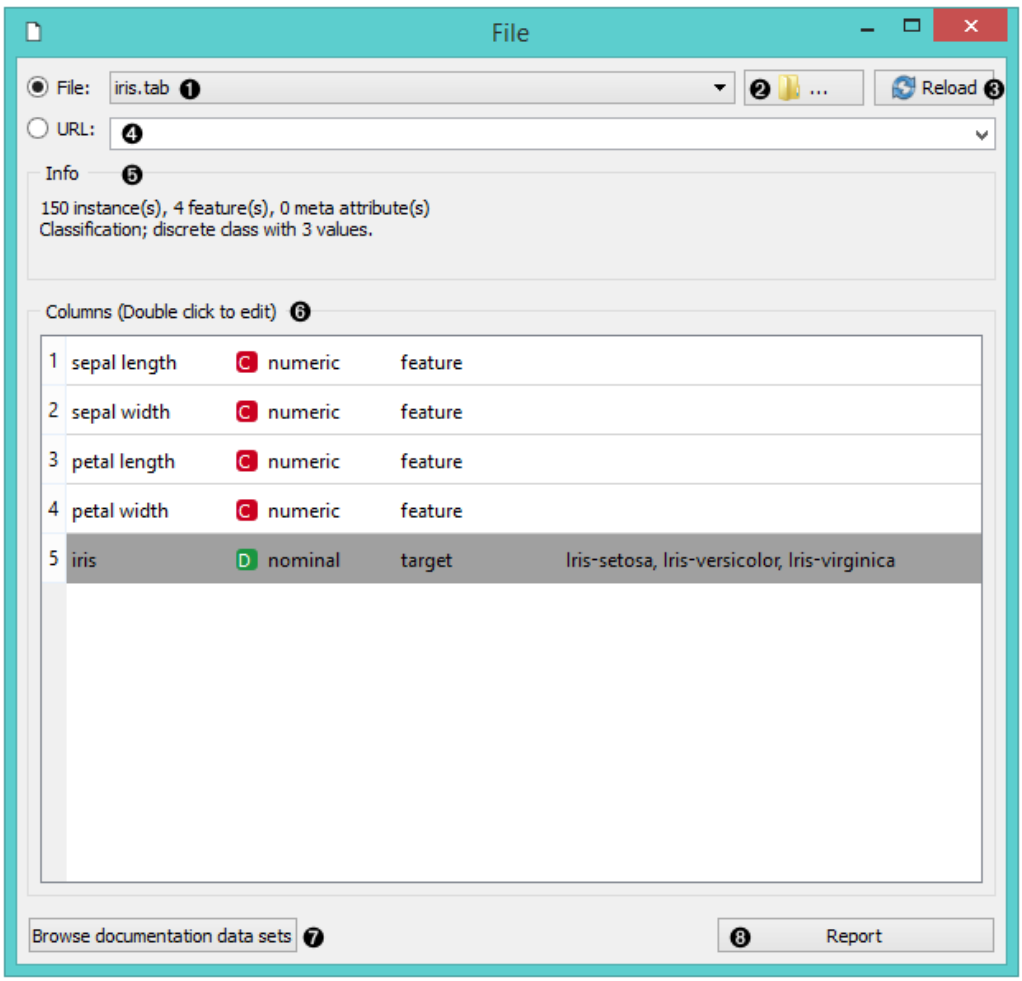
Reads attribute-value data from an input file.

Outputs

Data: dataset from the file

The **File** widget reads the input data file (data table with data instances) and sends the dataset to its output channel. The history of most recently opened files is maintained in the widget. The widget also includes a directory with sample datasets that come pre-installed with Orange.

The widget reads data from Excel (**.xlsx**), simple tab-delimited (**.txt**), comma-separated files (**.csv**) or URLs. For other formats see Other Formats section below.

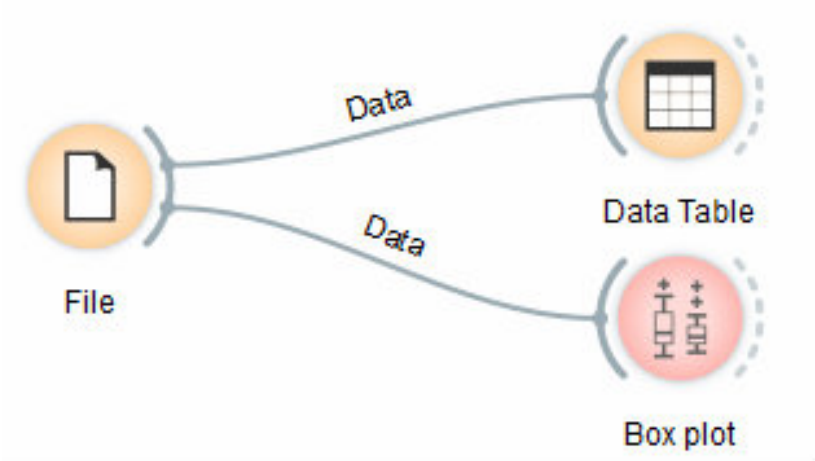


1. Browse through previously opened data files, or load any of the sample ones.
2. Browse for a data file.
3. Reloads currently selected data file.
4. Insert data from URL addresses, including data from Google Sheets.
5. Information on the loaded dataset: dataset size, number and types of data features.

6. Additional information on the features in the dataset. Features can be edited by double-clicking on them. The user can change the attribute names, select the type of variable per each attribute (*Continuous*, *Nominal*, *String*, *Datetime*), and choose how to further define the attributes (as *Features*, *Targets* or *Meta*). The user can also decide to ignore an attribute.
7. Browse documentation datasets.
8. Produce a report.

Example

Most Orange workflows would probably start with the **File** widget. In the schema below, the widget is used to read the data that is sent to both the Data Table and the Box Plot widget.



Loading your data

Orange can import any comma, .xlsx or tab-delimited data file or URL. Use the **File** widget and then, if needed, select class and meta attributes.

To specify the domain and the type of the attribute, attribute names can be preceded with a label followed by a hash. Use c for class and m for meta attribute, i to ignore a column, and C, D, S for continuous, discrete and string attribute types. Examples: C#mpg, mS#name, i#dummy.

Orange's native format is a tab-delimited text file with three header rows. The first row contains attribute names, the second the type (*continuous*, *discrete* or *string*), and the third the optional element (*class*, *meta* or *time*).

2.1.2. Data Table

Displays attribute-value data in a spreadsheet.

Inputs

- Data: input dataset

Outputs

- Selected Data: instances selected from the table

The **Data Table** widget receives one or more datasets in its input and presents them as a spreadsheet. Data instances may be sorted by attribute values. The widget also supports manual selection of data instances.

1. The name of the dataset (usually the input data file). Data instances are in rows and their attribute values in columns. In this example, the dataset is sorted by the attribute “sepal length”.
2. Info on current dataset size and number and types of attributes
3. Values of continuous attributes can be visualized with bars; colors can be attributed to different classes.
4. Data instances (rows) can be selected and sent to the widget’s output channel.
5. Use the *Restore Original Order* button to reorder data instances after attribute-based sorting.
6. Produce a report.
7. While auto-send is on, all changes will be automatically communicated to other widgets. Otherwise, press *Send Selected Rows*.

2.1.3. Select Columns

Manual selection of data attributes and composition of data domain.

Inputs

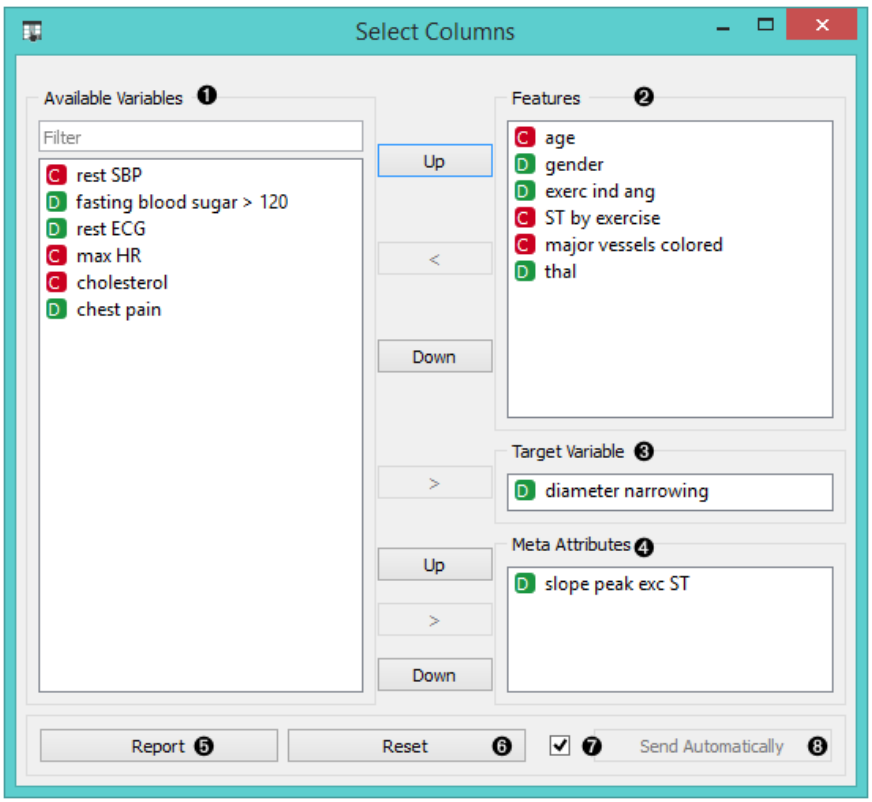
- Data: input dataset

Outputs

- Data: dataset with columns as set in the widget

The **Select Columns** widget is used to manually compose your data domain. The user can decide which attributes will be used and how. Orange distinguishes between ordinary attributes, (optional) class attributes and meta attributes. For instance, for building a classification model, the domain would be composed of a set of attributes and a discrete class attribute. Meta attributes are not used in modeling, but several widgets can use them as instance labels.

Orange attributes have a type and are either discrete, continuous or a character string. The attribute type is marked with a symbol appearing before the name of the attribute (D, C, S, respectively).



1. Left-out data attributes that will not be in the output data file
2. Data attributes in the new data file
3. Target variable. If none, the new dataset will be without a target variable.
4. Meta attributes of the new data file. These attributes are included in the dataset but are, for most methods, not considered in the analysis.

5. Produce a report.
6. Reset the domain composition to that of the input data file.
7. Tick if you wish to auto-apply changes of the data domain.
8. Apply changes of the data domain and send the new data file to the output channel of the widget.

2.1.4. Select Rows

Selects data instances based on conditions over data features.

Inputs

- Data: input dataset

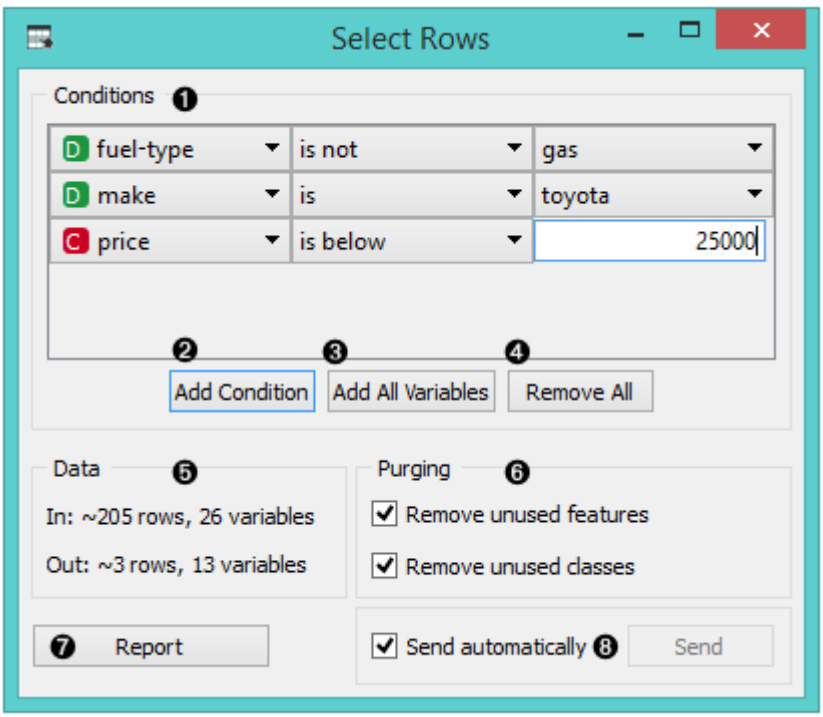
Outputs

- Matching Data: instances that match the conditions
- Non-Matching Data: instances that do not match the conditions
- Data: data with an additional column showing whether a instance is selected

This widget selects a subset from an input dataset, based on user-defined conditions. Instances that match the selection rule are placed in the output *Matching Data* channel.

Criteria for data selection are presented as a collection of conjunct terms (i.e. selected items are those matching all the terms in '*Conditions*').

Condition terms are defined through selecting an attribute, selecting an operator from a list of operators, and, if needed, defining the value to be used in the condition term. Operators are different for discrete, continuous and string attributes.



1. Conditions you want to apply, their operators and related values
2. Add a new condition to the list of conditions.
3. Add all the possible variables at once.
4. Remove all the listed variables at once.
5. Information on the input dataset and information on instances that match the condition(s)
6. Purge the output data.
7. When the *Send automatically* box is ticked, all changes will be automatically communicated to other widgets.
8. Produce a report.

Any change in the composition of the condition will update the information pane (*Data Out*).

If *Send automatically* is selected, then the output is updated on any change in the composition of the condition or any of its terms.

2.1.5. Preprocess

Preprocesses data with selected methods.

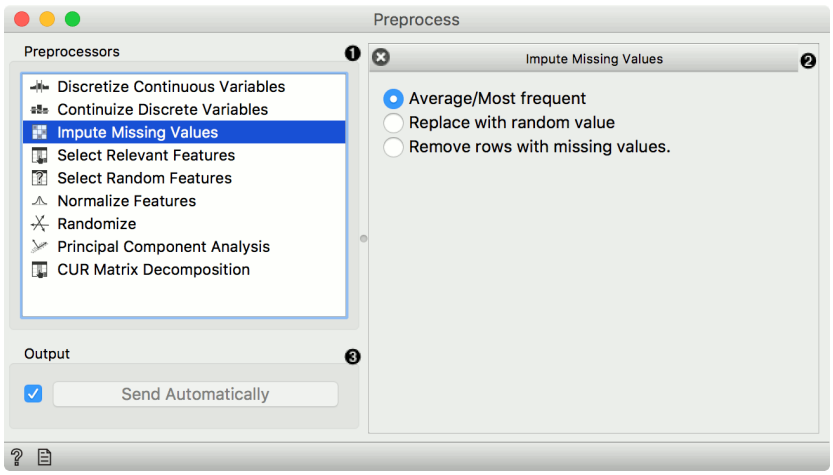
Inputs

- Data: input dataset

Outputs

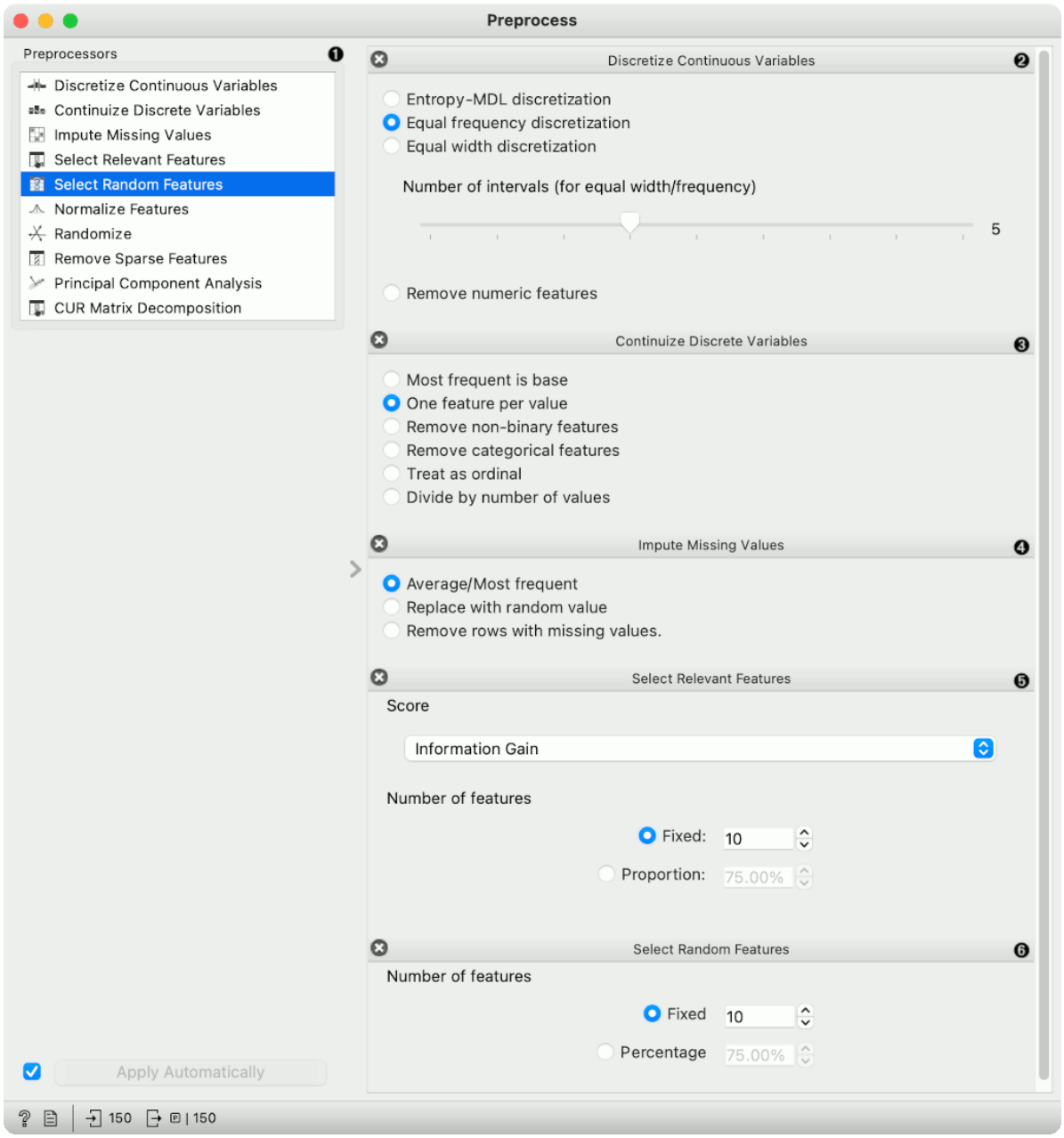
- Preprocessor: preprocessing method
- Preprocessed Data: data preprocessed with selected methods

Preprocessing is crucial for achieving better-quality analysis results. The **Preprocess** widget offers several preprocessing methods that can be combined in a single preprocessing pipeline. Some methods are available as separate widgets, which offer advanced techniques and greater parameter tuning.



1. List of preprocessors. Double click the preprocessors you wish to use and shuffle their order by dragging them up or down. You can also add preprocessors by dragging them from the left menu to the right.
2. Preprocessing pipeline.
3. When the box is ticked (*Send Automatically*), the widget will communicate changes automatically. Alternatively, click *Send*.

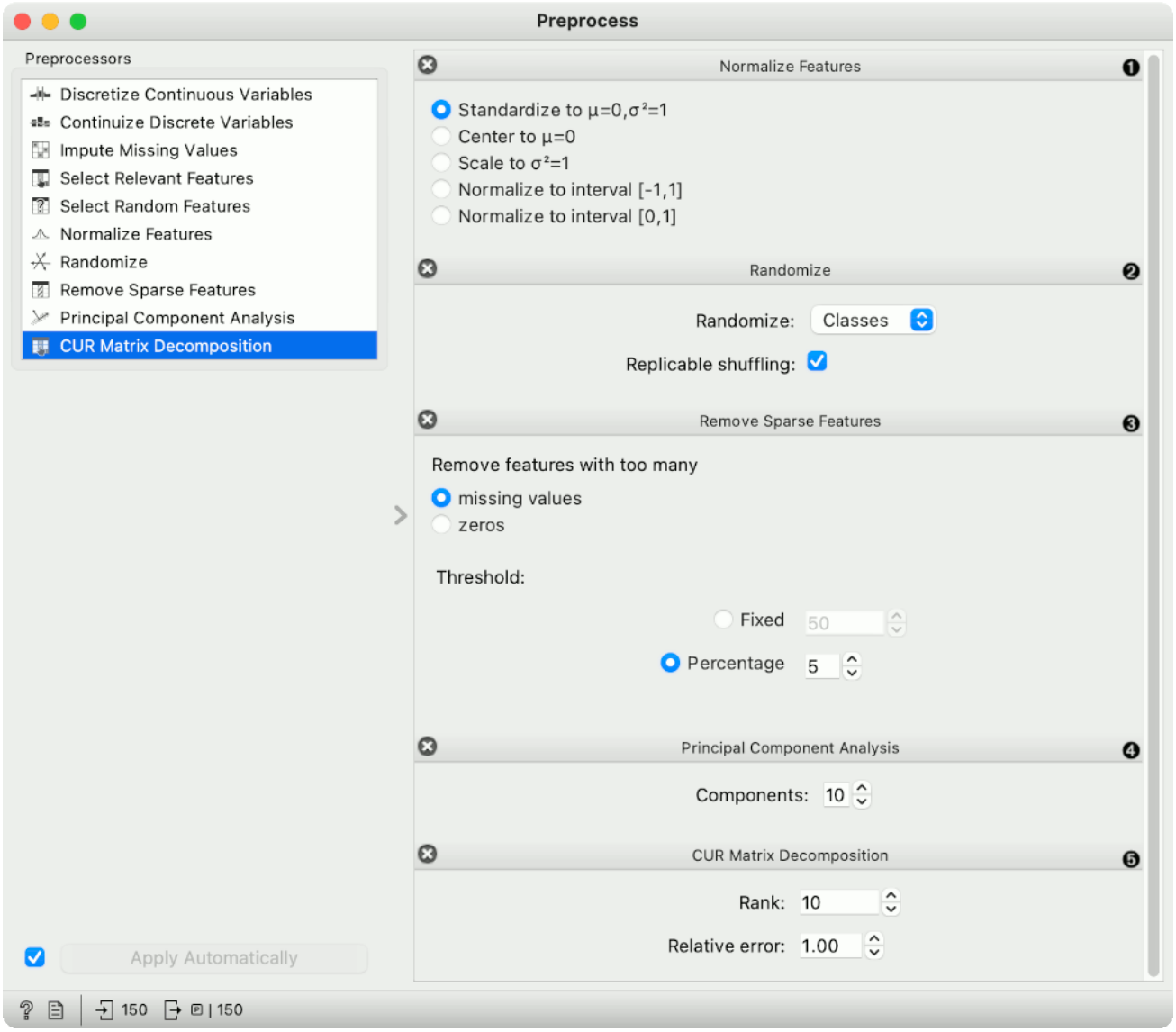
Preprocessors



1. List of preprocessors.
2. Discretization of continuous values:

- Entropy-MDL discretization by Fayyad and Irani that uses expected information to determine bins.
- *Equal frequency discretization* splits by frequency (same number of instances in each bin.
- *Equal width discretization* creates bins of equal width (span of each bin is the same).
- *Remove numeric features* altogether.

3. Continuization of discrete values:
   - *Most frequent as base* treats the most frequent discrete value as 0 and others as 1. The discrete attributes with more than 2 values, the most frequent will be considered as a base and contrasted with remaining values in corresponding columns.
   - *One feature per value* creates columns for each value, place 1 where an instance has that value and 0 where it doesn't. Essentially One Hot Encoding.
   - *Remove non-binary features* retains only categorical features that have values of either 0 or 1 and transforms them into continuous.
   - *Remove categorical features* removes categorical features altogether.
   - *Treat as ordinal* takes discrete values and treats them as numbers. If discrete values are categories, each category will be assigned a number as they appear in the data.
   - *Divide by number of values* is similar to treat as ordinal, but the final values will be divided by the total number of values and hence the range of the new continuous variable will be [0, 1].
4. Impute missing values:
   - *Average/Most frequent* replaces missing values (NaN) with the average (for continuous) or most frequent (for discrete) value.
   - *Replace with random value* replaces missing values with random ones within the range of each variable.
   - *Remove rows with missing values*.
5. Select relevant features:
   - Similar to Rank, this preprocessor outputs only the most informative features. Score can be determined by information gain, gain ratio, gini index, ReliefF, fast correlation based filter, ANOVA, Chi2, RReliefF, and Univariate Linear Regression.
   - *Strategy* refers to how many variables should be on the output. *Fixed* returns a fixed number of top scored variables, while *Percentile* return the selected top percent of the features.
6. *Select random features* outputs either a fixed number of features from the original data or a percentage. This is mainly used for advanced testing and educational purposes.



1. Normalize adjusts values to a common scale. Center values by mean or median or omit centering altogether. Similar for scaling, one can scale by SD (standard deviation), by span or not at all.
2. Randomize instances. Randomize classes shuffles class values and destroys connection between instances and class. Similarly, one can randomize features or meta data. If replicable shuffling is on, randomization results can be shared and repeated with a saved workflow. This is mainly used for advanced testing and educational purposes.
3. *Remove sparse features* retains features that have more than a number/percentage of non-zero/missing values. The rest are discarded.
4. Principal component analysis outputs results of a PCA transformation. Similar to the PCA widget.
5. CUR matrix decomposition is a dimensionality reduction method, similar to SVD.

For the complete reference on Widget visit

3. : Workflows in Orange

4. : Lab Work using Orange Data Mining Tool

4.1. Lab Session 1: Getting Started with Orange

4.1.1. Aim:

Explore the Orange Application
Develop workflows using the Data Mining Tool
Visualize the data using visualization tools

4.1.2. Problem Description:

For the given dataset, perform the following tasks:

- Load the dataset
- View the data
- Filter the data based on rows and columns
- Visualize the data using Scatter Plot and Distributions

4.1.3. Widgets Used:

- File (Refer 2.1.1)
- Data Table (Refer 2.1.2)
- Select Columns (Refer 2.1.3)
- Select Rows (Refer 2.1.4)

4.1.4. Data Workflow (Paste your workflow here size: 9 by14 in)

4.1.5. Visualizing the data(Paste the visual plots here. Each plot size: 5 by 6 in, two plots per row)

4.1.6. Conclusion/Inferences: (With reference to the fig captions write 5 points of observations per plot)

4.2. Lab Session 2: Data Pre-processing using Orange

4.2.1. Aim: Perform data cleaning and pre processing on the given dataset

4.2.2. Problem Description: Perform the following on the given dataset

- Impute Missing values
- Discretize continuous values
- Continuize discrete values
- Select relevant attributes
- Select random attributes

4.2.3. Widgets Used: Preprocess (Refer 2.1.5)

4.2.4. Data Workflow

4.2.5. Output: (Screenshot of part original data and corresponding cleaned data)

4.2.6. Conclusion