**Neural Network**

A multi-layer perceptron (MLP) algorithm with backpropagation.

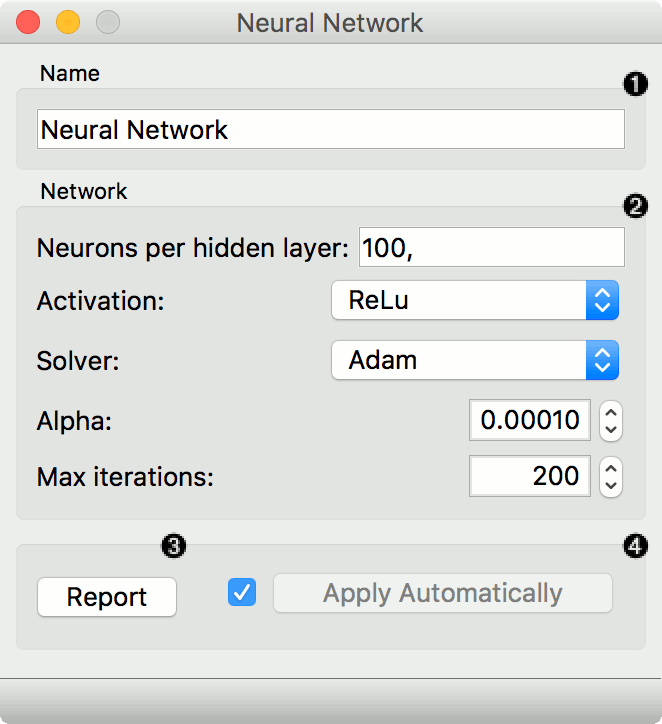
**Inputs**

* Data: input dataset
* Preprocessor: preprocessing method(s)

**Outputs**

* Learner: multi-layer perceptron learning algorithm
* Model: trained model

The **Neural Network** widget uses sklearn's [Multi-layer Perceptron algorithm](http://scikit-learn.org/stable/modules/neural_networks_supervised.html) that can learn non-linear models as well as linear.



1. A name under which it will appear in other widgets. The default name is "Neural Network".
2. Set model parameters:
   * Neurons per hidden layer: defined as the ith element represents the number of neurons in the ith hidden layer. E.g. a neural network with 3 layers can be defined as 2, 3, 2.
   * Activation function for the hidden layer:
     + Identity: no-op activation, useful to implement linear bottleneck
     + Logistic: the logistic sigmoid function
     + tanh: the hyperbolic tan function
     + ReLu: the rectified linear unit function
   * Solver for weight optimization:
     + L-BFGS-B: an optimizer in the family of quasi-Newton methods
     + SGD: stochastic gradient descent
     + Adam: stochastic gradient-based optimizer
   * Alpha: L2 penalty (regularization term) parameter
   * Max iterations: maximum number of iterations

Other parameters are set to [sklearn's defaults](http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html).

1. Produce a report.
2. When the box is ticked (*Apply Automatically*), the widget will communicate changes automatically. Alternatively, click *Apply*.

**Preprocessing**

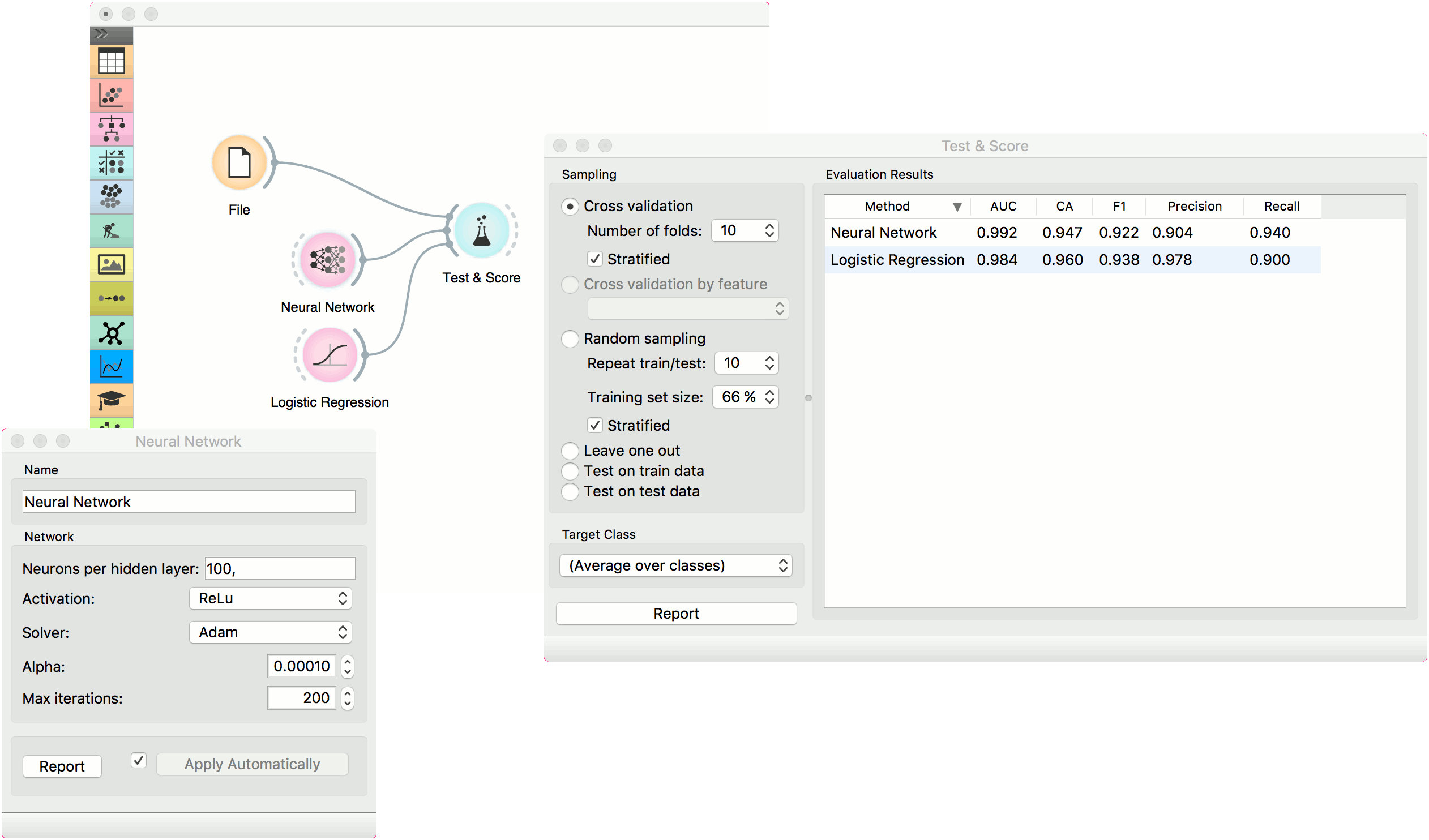
Neural Network uses default preprocessing when no other preprocessors are given. It executes them in the following order:

* removes instances with unknown target values
* continuizes categorical variables (with one-hot-encoding)
* removes empty columns
* imputes missing values with mean values
* normalizes the data by centering to mean and scaling to standard deviation of 1

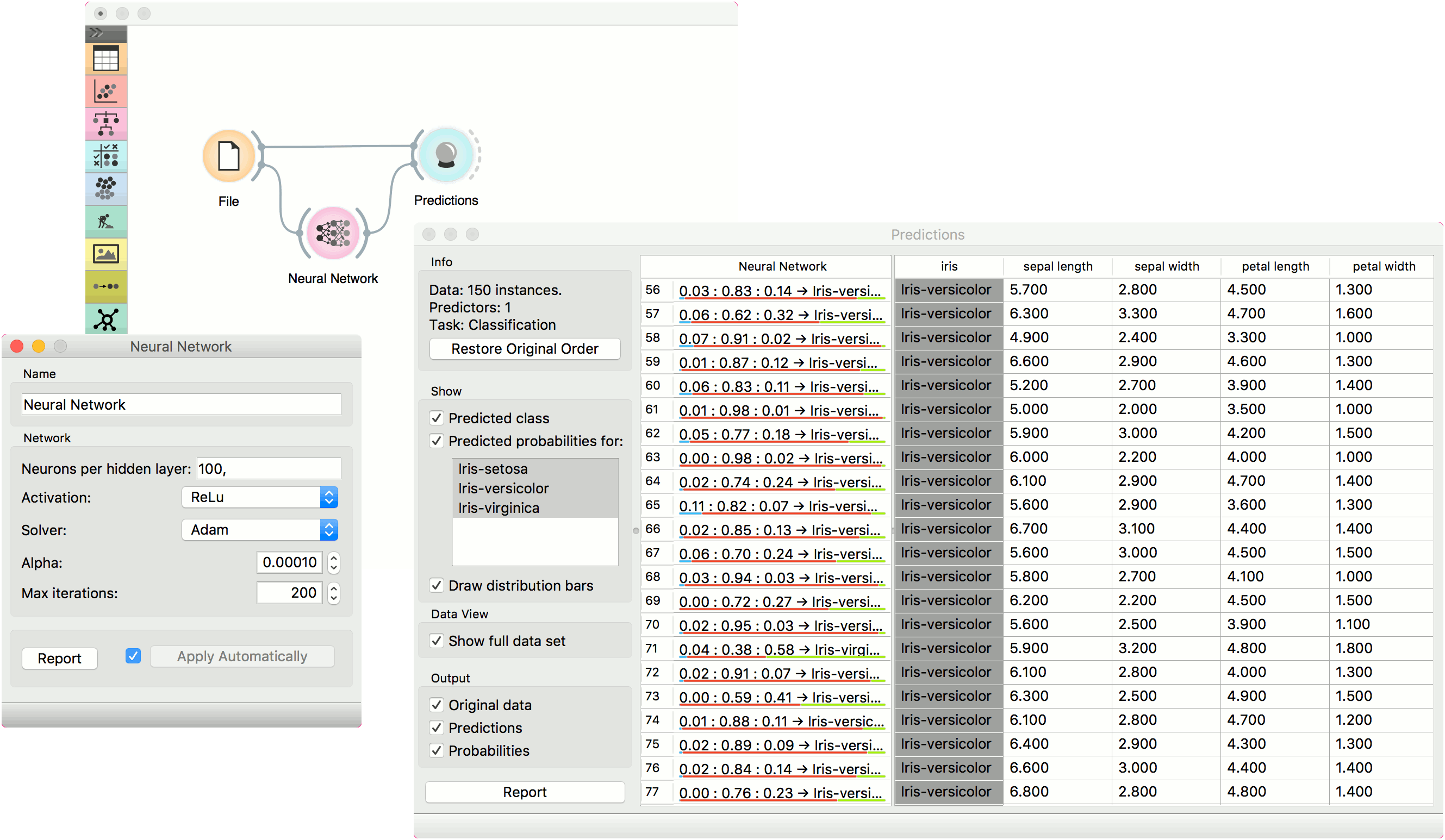
To remove default preprocessing, connect an empty [Preprocess](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/preprocess.html) widget to the learner.

**Examples**

The first example is a classification task on *iris* dataset. We compare the results of **Neural Network** with the [Logistic Regression](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/logisticregression.html).



The second example is a prediction task, still using the *iris* data. This workflow shows how to use the *Learner* output. We input the **Neural Network** prediction model into [Predictions](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/evaluate/predictions.html) and observe the predicted values.



**SVM**

Support Vector Machines map inputs to higher-dimensional feature spaces.

**Inputs**

* Data: input dataset
* Preprocessor: preprocessing method(s)

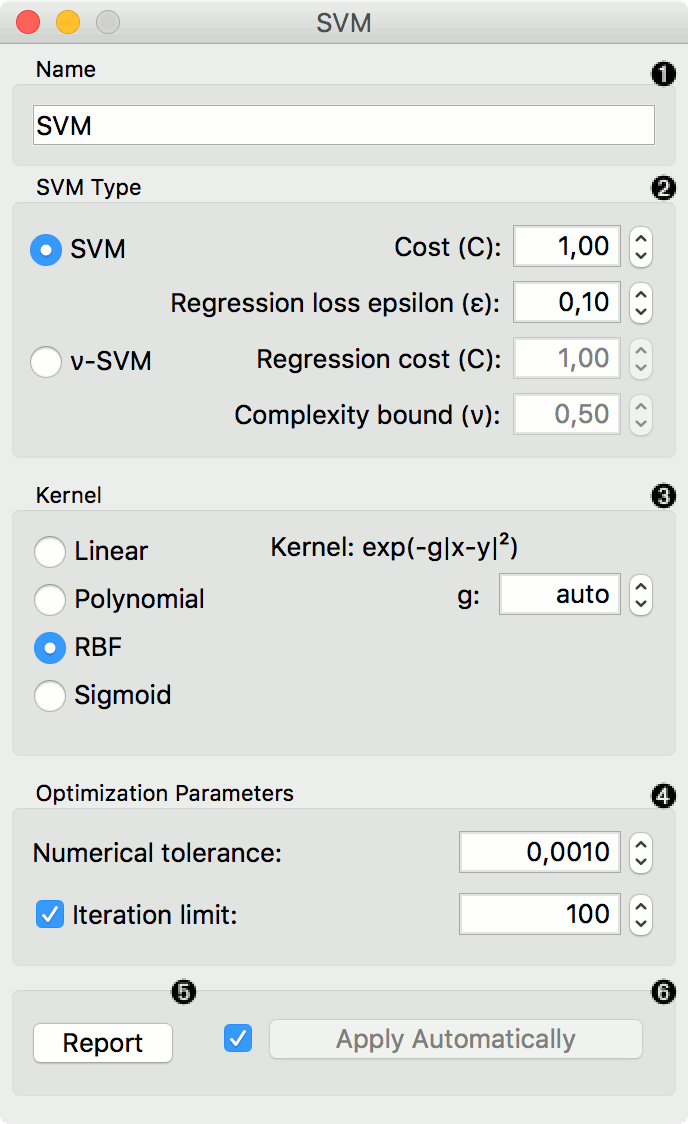
**Outputs**

* Learner: linear regression learning algorithm
* Model: trained model
* Support Vectors: instances used as support vectors

[Support vector machine](https://en.wikipedia.org/wiki/Support_vector_machine) (SVM) is a machine learning technique that separates the attribute space with a hyperplane, thus maximizing the margin between the instances of different classes or class values. The technique often yields supreme predictive performance results. Orange embeds a popular implementation of SVM from the [LIBSVM](https://www.csie.ntu.edu.tw/~cjlin/libsvm/) package. This widget is its graphical user interface.

For regression tasks, **SVM** performs linear regression in a high dimension feature space using an ε-insensitive loss. Its estimation accuracy depends on a good setting of C, ε and kernel parameters. The widget outputs class predictions based on a [SVM Regression](https://en.wikipedia.org/wiki/Support_vector_machine#Regression).

The widget works for both classification and regression tasks.



1. The learner can be given a name under which it will appear in other widgets. The default name is "SVM".
2. SVM type with test error settings. *SVM* and *ν-SVM* are based on different minimization of the error function. On the right side, you can set test error bounds:
   * [SVM](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html):
     + [Cost](http://www.quora.com/What-are-C-and-gamma-with-regards-to-a-support-vector-machine): penalty term for loss and applies for classification and regression tasks.
     + ε: a parameter to the epsilon-SVR model, applies to regression tasks. Defines the distance from true values within which no penalty is associated with predicted values.
   * [ν-SVM](http://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVR.html#sklearn.svm.NuSVR):
     + [Cost](http://www.quora.com/What-are-C-and-gamma-with-regards-to-a-support-vector-machine): penalty term for loss and applies only to regression tasks
     + ν: a parameter to the ν-SVR model, applies to classification and regression tasks. An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors.
3. Kernel is a function that transforms attribute space to a new feature space to fit the maximum-margin hyperplane, thus allowing the algorithm to create the model with [Linear](https://en.wikipedia.org/wiki/Linear_model), [Polynomial](https://en.wikipedia.org/wiki/Polynomial_kernel), [RBF](https://en.wikipedia.org/wiki/Radial_basis_function_kernel) and [Sigmoid](http://crsouza.com/2010/03/kernel-functions-for-machine-learning-applications/#sigmoid) kernels. Functions that specify the kernel are presented upon selecting them, and the constants involved are:
   * g for the gamma constant in kernel function (the recommended value is 1/k, where k is the number of the attributes, but since there may be no training set given to the widget the default is 0 and the user has to set this option manually),
   * c for the constant c0 in the kernel function (default 0), and
   * d for the degree of the kernel (default 3).
4. Set permitted deviation from the expected value in Numerical Tolerance. Tick the box next to Iteration Limit to set the maximum number of iterations permitted.
5. Produce a report.
6. Click *Apply* to commit changes. If you tick the box on the left side of the *Apply*button, changes will be communicated automatically.

Preprocessing

SVM uses default preprocessing when no other preprocessors are given. It executes them in the following order:

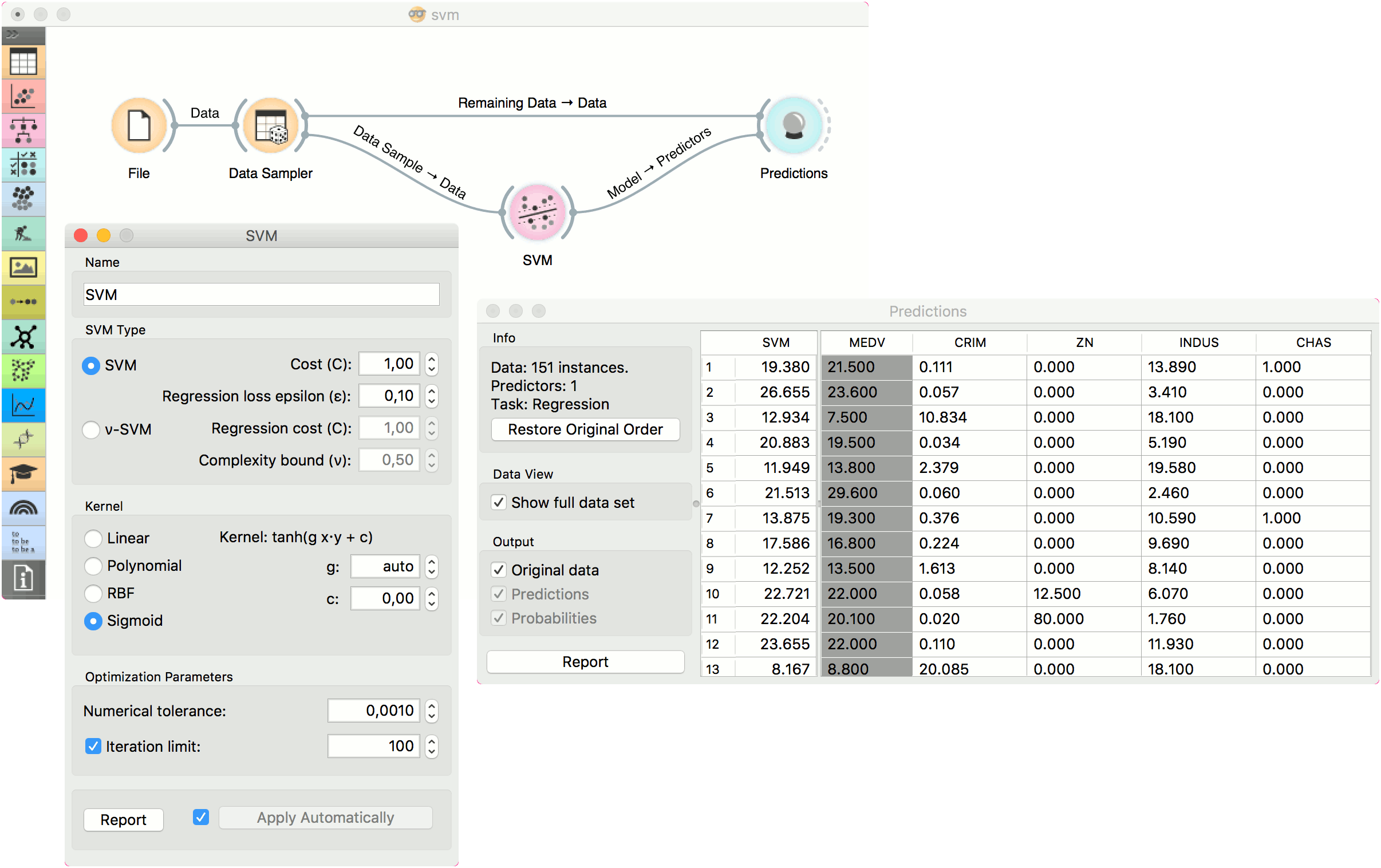
* removes instances with unknown target values
* continuizes categorical variables (with one-hot-encoding)
* removes empty columns
* imputes missing values with mean values

For classification, SVM also normalizes dense and scales sparse data.

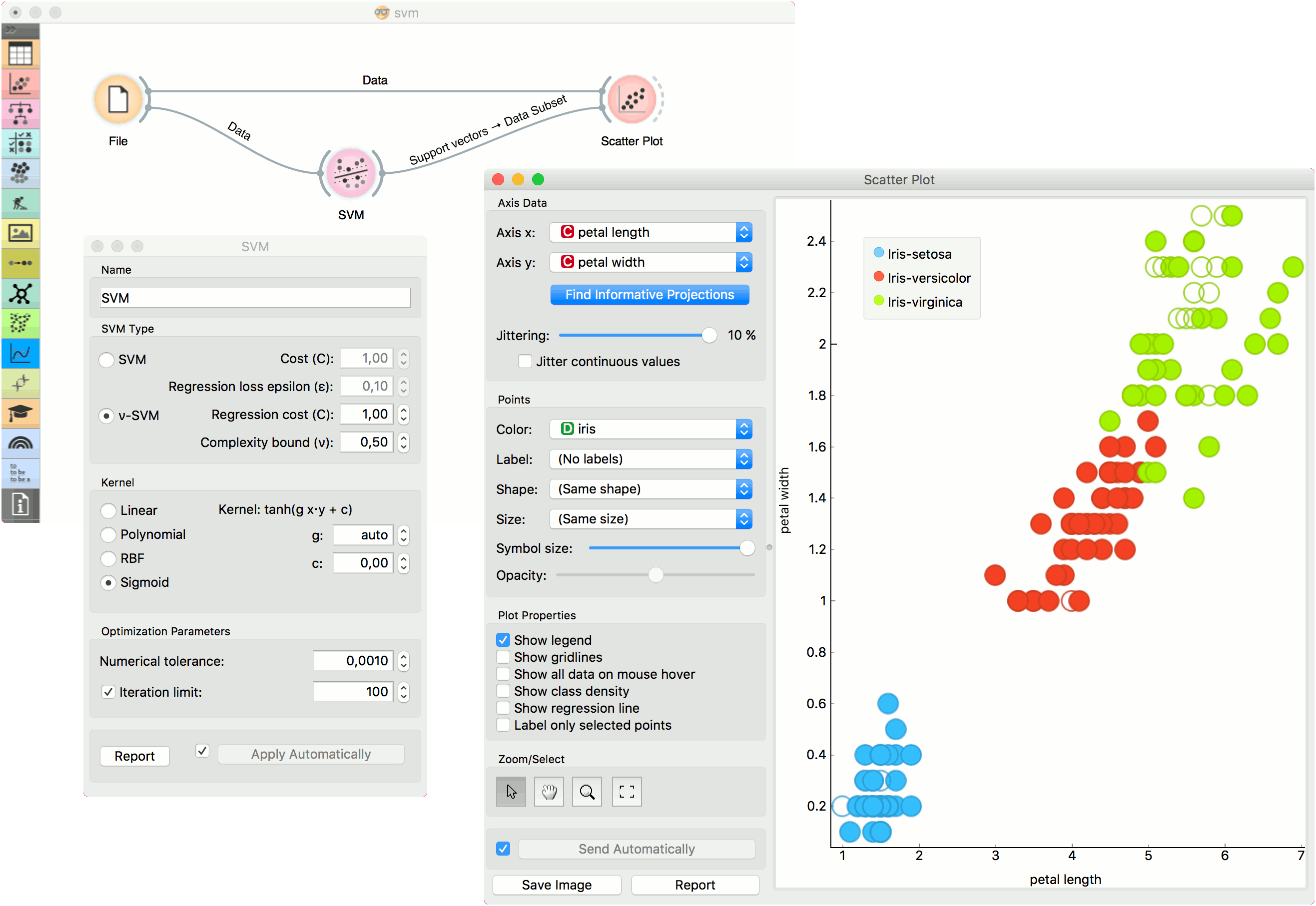
To remove default preprocessing, connect an empty [Preprocess](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/preprocess.html) widget to the learner.

Examples

In the first (regression) example, we have used *housing* dataset and split the data into two data subsets (*Data Sample* and *Remaining Data*) with [Data Sampler](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/datasampler.html). The sample was sent to SVM which produced a *Model*, which was then used in [Predictions](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/evaluate/predictions.html) to predict the values in *Remaining Data*. A similar schema can be used if the data is already in two separate files; in this case, two [File](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/file.html) widgets would be used instead of the [File](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/file.html) - [Data Sampler](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/datasampler.html) combination.



The second example shows how to use **SVM** in combination with [Scatter Plot](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/visualize/scatterplot.html). The following workflow trains a SVM model on *iris* data and outputs support vectors, which are those data instances that were used as support vectors in the learning phase. We can observe which are these data instances in a scatter plot visualization. Note that for the workflow to work correctly, you must set the links between widgets as demonstrated in the screenshot below.



# **LASSO / Ridge Regression**

The logistic regression classification algorithm with LASSO (L1) or ridge (L2) regularization.

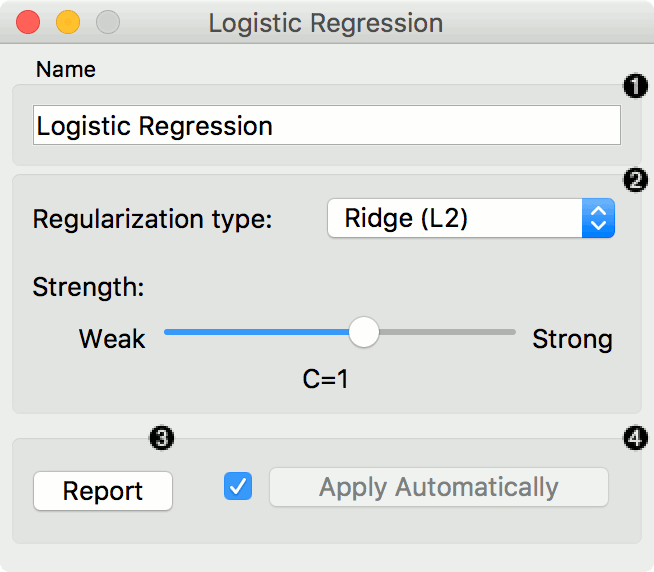
**Inputs**

* Data: input dataset
* Preprocessor: preprocessing method(s)

**Outputs**

* Learner: logistic regression learning algorithm
* Model: trained model
* Coefficients: logistic regression coefficients

**Logistic Regression** learns a [Logistic Regression](https://en.wikipedia.org/wiki/Logistic_regression) model from the data. It only works for classification tasks.



1. A name under which the learner appears in other widgets. The default name is "Logistic Regression".
2. [Regularization](https://en.wikipedia.org/wiki/Regularization_(mathematics)) type (either [L1](https://en.wikipedia.org/wiki/Least_squares#Lasso_method) or [L2](https://en.wikipedia.org/wiki/Tikhonov_regularization)). Set the cost strength (default is C=1).
3. Press Apply to commit changes. If Apply Automatically is ticked, changes will be communicated automatically.

**Preprocessing**

Logistic Regression uses default preprocessing when no other preprocessors are given. It executes them in the following order:

* removes instances with unknown target values
* continuizes categorical variables (with one-hot-encoding)
* removes empty columns
* imputes missing values with mean values

To remove default preprocessing, connect an empty [Preprocess](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/preprocess.html) widget to the learner.

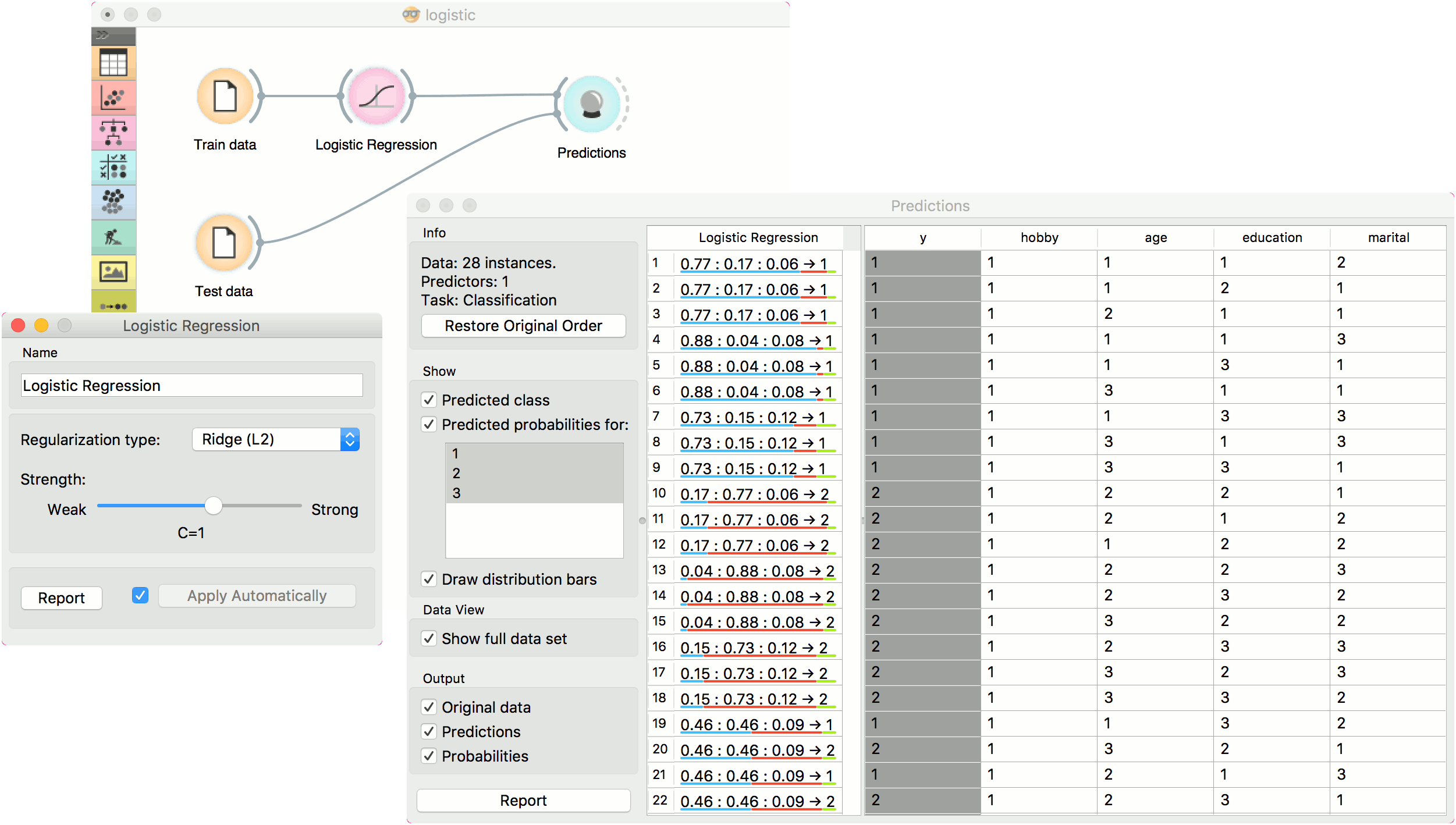
**Feature Scoring**

Logistic Regression can be used with Rank for feature scoring. See [Learners as Scorers](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/learners-as-scorers/index.html) for an example.

**Examples**

The widget is used just as any other widget for inducing a classifier. This is an example demonstrating prediction results with logistic regression on the hayes-roth dataset. We first load hayes-roth\_learn in the [File](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/file.html) widget and pass the data to **Logistic Regression**. Then we pass the trained model to [Predictions](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/evaluate/predictions.html).

Now we want to predict class value on a new dataset. We load hayes-roth\_test in the second **File** widget and connect it to **Predictions**. We can now observe class values predicted with **Logistic Regression** directly in **Predictions**.



# **Naive Bayes**

A fast and simple probabilistic classifier based on Bayes' theorem with the assumption of feature independence.

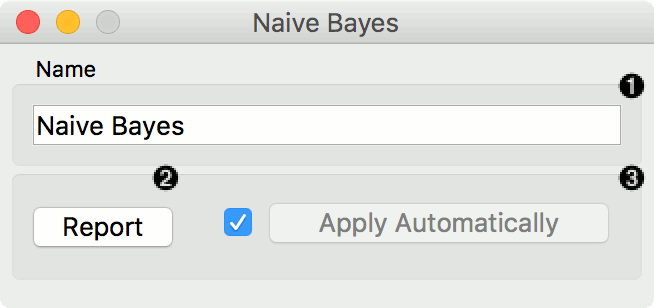
**Inputs**

* Data: input dataset
* Preprocessor: preprocessing method(s)

**Outputs**

* Learner: naive bayes learning algorithm
* Model: trained model

**Naive Bayes** learns a [Naive Bayesian](https://en.wikipedia.org/wiki/Naive_Bayes_classifier) model from the data. It only works for classification tasks.



This widget has two options: the name under which it will appear in other widgets and producing a report. The default name is Naive Bayes. When you change it, you need to press Apply.

**Preprocessing**

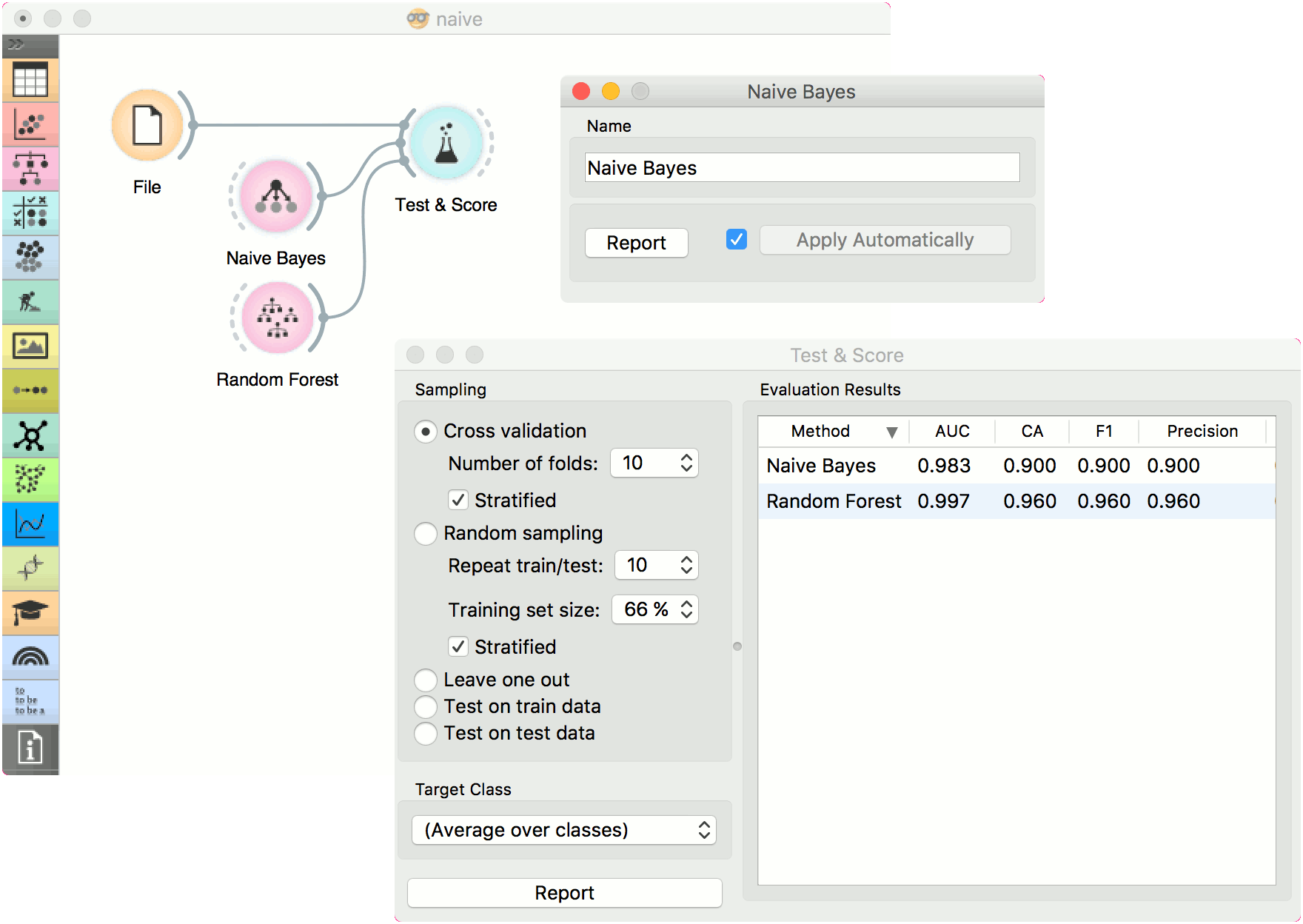
Naive Bayes uses default preprocessing when no other preprocessors are given. It executes them in the following order:

* removes empty columns
* discretizes numeric values to 4 bins with equal frequency

To remove default preprocessing, connect an empty [Preprocess](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/preprocess.html) widget to the learner.

**Examples**

Here, we present two uses of this widget. First, we compare the results of the **Naive Bayes** with another model, the [Random Forest](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/randomforest.html). We connect iris data from [File](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/data/file.html) to [Test & Score](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/evaluate/testandscore.html). We also connect **Naive Bayes** and [Random Forest](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/randomforest.html) to **Test & Score** and observe their prediction scores.



The second schema shows the quality of predictions made with **Naive Bayes**. We feed the [Test & Score](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/evaluate/testandscore.html) widget a Naive Bayes learner and then send the data to the [Confusion Matrix](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/evaluate/confusionmatrix.html). We also connect [Scatter Plot](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/visualize/scatterplot.html) with **File**. Then we select the misclassified instances in the **Confusion Matrix** and show feed them to [Scatter Plot](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/visualize/scatterplot.html). The bold dots in the scatterplot are the misclassified instances from **Naive Bayes**.

