
Orange3 Educational Add-on Documentation

Biolab

Apr 05, 2023

Contents

1	Widgets	3
2	Indices and tables	31

Widgets in Educational Add-on demonstrate several key data mining and machine learning procedures. The widgets are useful for beginners to understand the inner working of key algorithms in the data mining and for teachers to be able to visually explain various methods in a classroom.

1.1 Google Sheets

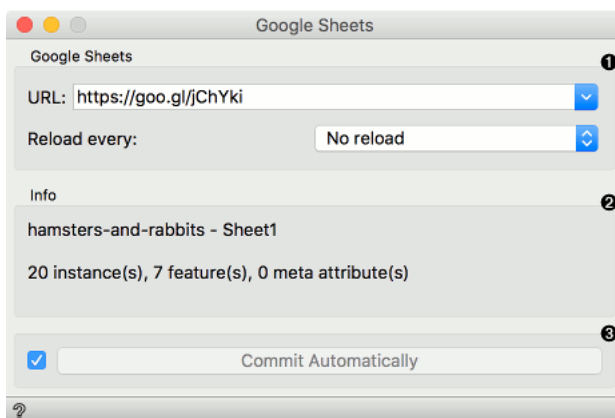
Read data from a Google Sheets spreadsheet.

Outputs

- Data: data set from the Google Sheets service.

1.1.1 Description

The widget reads data from the [Google Sheets service](#). To use the widget, click the Share button in a selected spreadsheet, copy the provided link and paste it into the widget's URL line. Press enter to load the data. To observe the data in real time, use the Reload function.

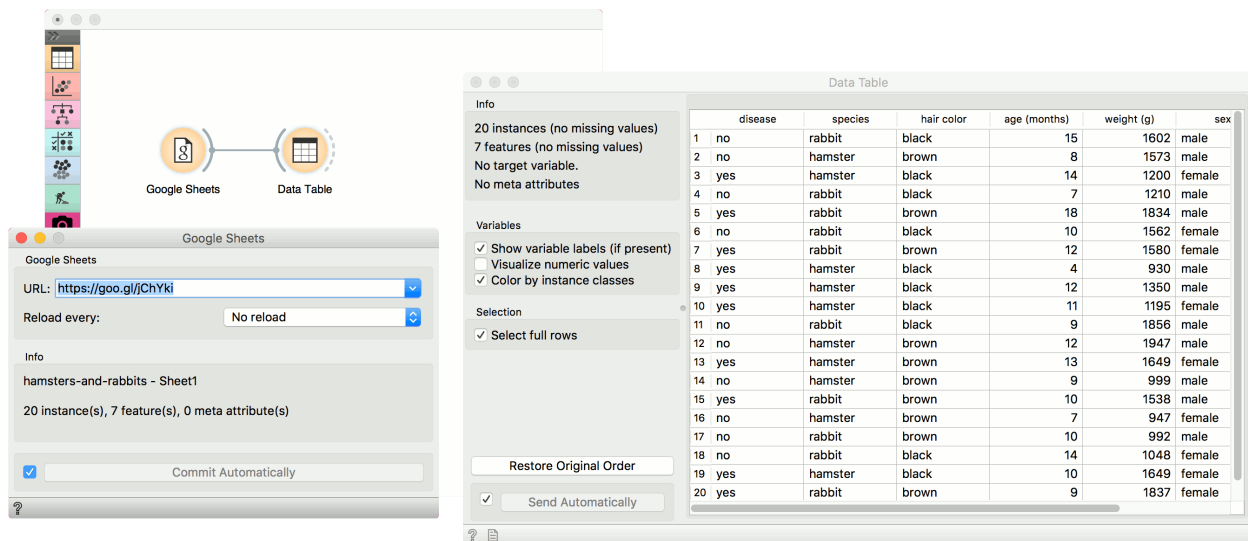


1. Enter the link to the spreadsheet. Press Enter to load the data. Set reload if you wish to observe the updates in real time.
2. Information on the data set: name and attributes.

3. If *Commit Automatically* is ticked, the data will be automatically communicated downstream. Alternatively, press *Commit*.

1.1.2 Example

This widget is used for loading the data. We have used the link from the Google Sheets: <https://goo.gl/jChYki>. This is a fictional data on hamsters and rabbits, of which some have the disease and some don't. Use the **Data Table** to observe the loaded data in a spreadsheet.



The screenshot shows the Orange3 workflow editor. A **Google Sheets** widget is connected to a **Data Table** widget. The **Google Sheets** widget has the URL <https://goo.gl/jChYki> entered. The **Data Table** widget displays the following data:

	disease	species	hair color	age (months)	weight (g)	sex
1	no	rabbit	black	15	1602	male
2	no	hamster	brown	8	1573	male
3	yes	hamster	black	14	1200	female
4	no	rabbit	black	7	1210	male
5	yes	rabbit	brown	18	1834	male
6	no	rabbit	black	10	1562	female
7	yes	rabbit	brown	12	1580	female
8	yes	hamster	black	4	930	male
9	yes	hamster	black	12	1350	male
10	yes	hamster	black	11	1195	female
11	no	rabbit	black	9	1856	male
12	no	hamster	brown	12	1947	male
13	yes	hamster	brown	13	1649	female
14	no	hamster	brown	9	999	male
15	yes	rabbit	brown	10	1538	male
16	no	hamster	brown	7	947	female
17	no	rabbit	brown	10	992	male
18	no	rabbit	black	14	1048	female
19	yes	hamster	black	10	1649	female
20	yes	rabbit	brown	9	1837	female

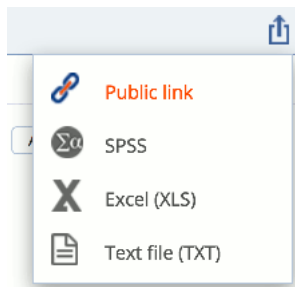
1.2 EnKlik Anketa

Import data from EnKlikAnketa (1ka.si) public URL.

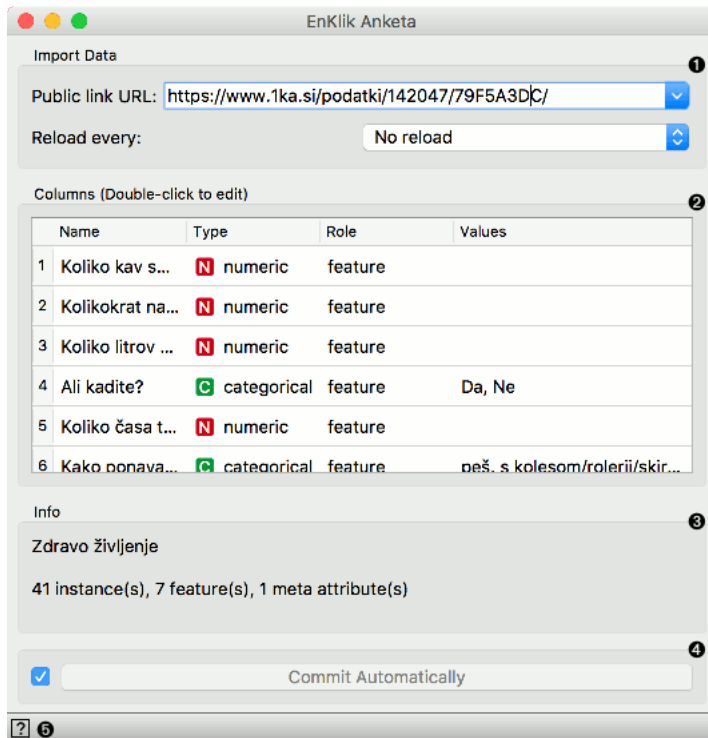
Outputs

- Data: survey results

The **EnKlik Anketa** widget retrieves survey results obtained from the [EnKlikAnketa](https://www.1ka.si/) service. You need to create a public link to retrieve the results. Go to the survey you wish to retrieve, then select Data (Podatki) tab and create a public link (javna povezava) at the top right corner.



Then insert the link into the Public link URL field. The link should look something like this: <https://www.1ka.si/podatki/123456/78A9B1CD/>.



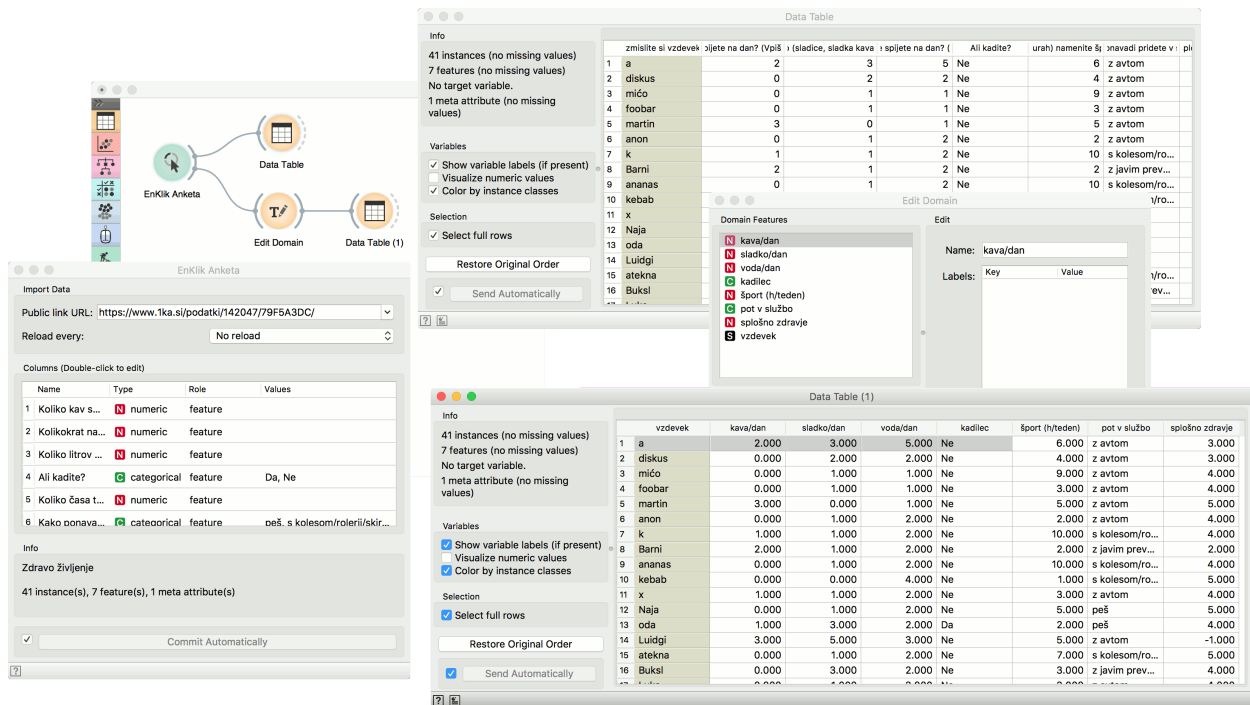
1. A public link to the survey results. To observe the results live, set the reload rate (5s - 5 min).
2. Attribute list. You can change the attribute type and role, just like in the **File** widget.
3. Survey meta information.
4. Tick the box on the left to commit the changes automatically. Alternatively, click *Commit*.
5. Access widget help.

1.2.1 Example

EnKlik Anketa widget is great for observing results from online surveys. We have created a sample survey and imported it into the widget. We have 41 responses and we have asked 8 questions, 7 of which were recognized as features and 1 as a meta attribute.

The widget sets questions from the survey as feature names. This, however, might be slightly impractical for analytical purposes, as we can see in the **Data Table**. We will shorten the names with **Edit Domain** widget.

Edit Domain enables us to change attribute names and even rename attribute values for discrete attributes. Now our attribute names are much easier to work with, as we can see in **Data Table (1)**.



1.3 Interactive k-Means

Educational widget that shows the working of a k-means clustering.

Inputs

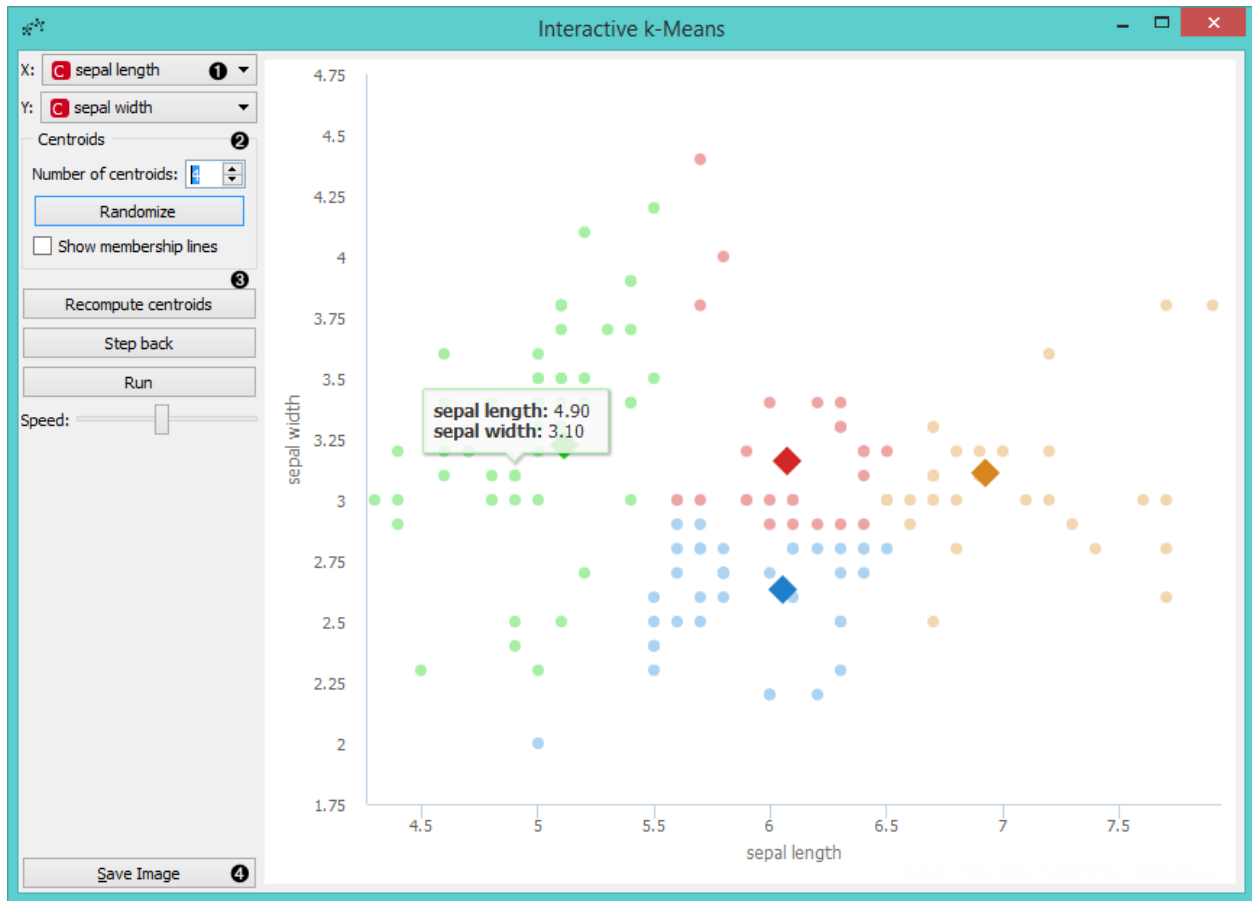
- Data: input data set

Outputs

- Data: data set with cluster annotation
- Centroids: centroids position

1.3.1 Description

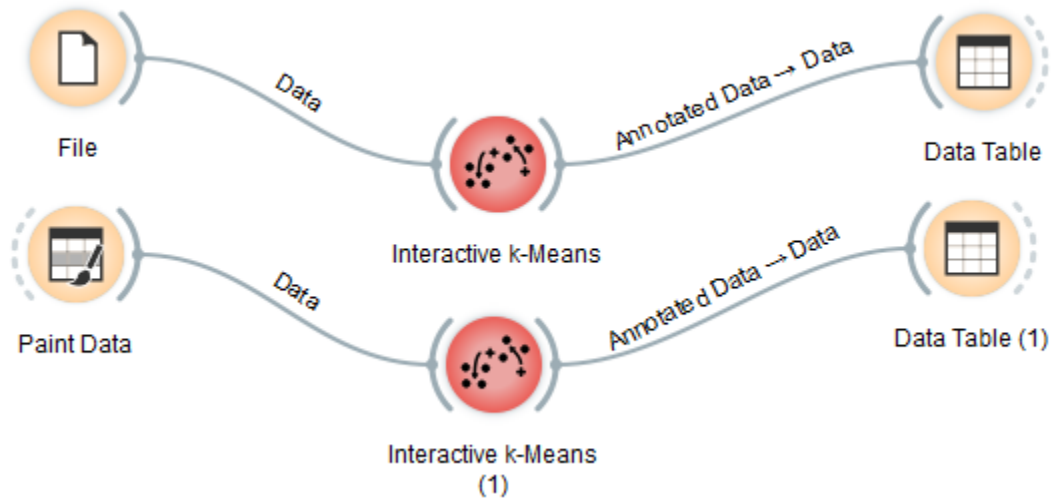
The aim of this widget is to show the working of a **k-means clustering algorithm** on two attributes from a data set. The widget applies k-means clustering to the selected two attributes step by step. Users can step through the algorithm and see how it works.



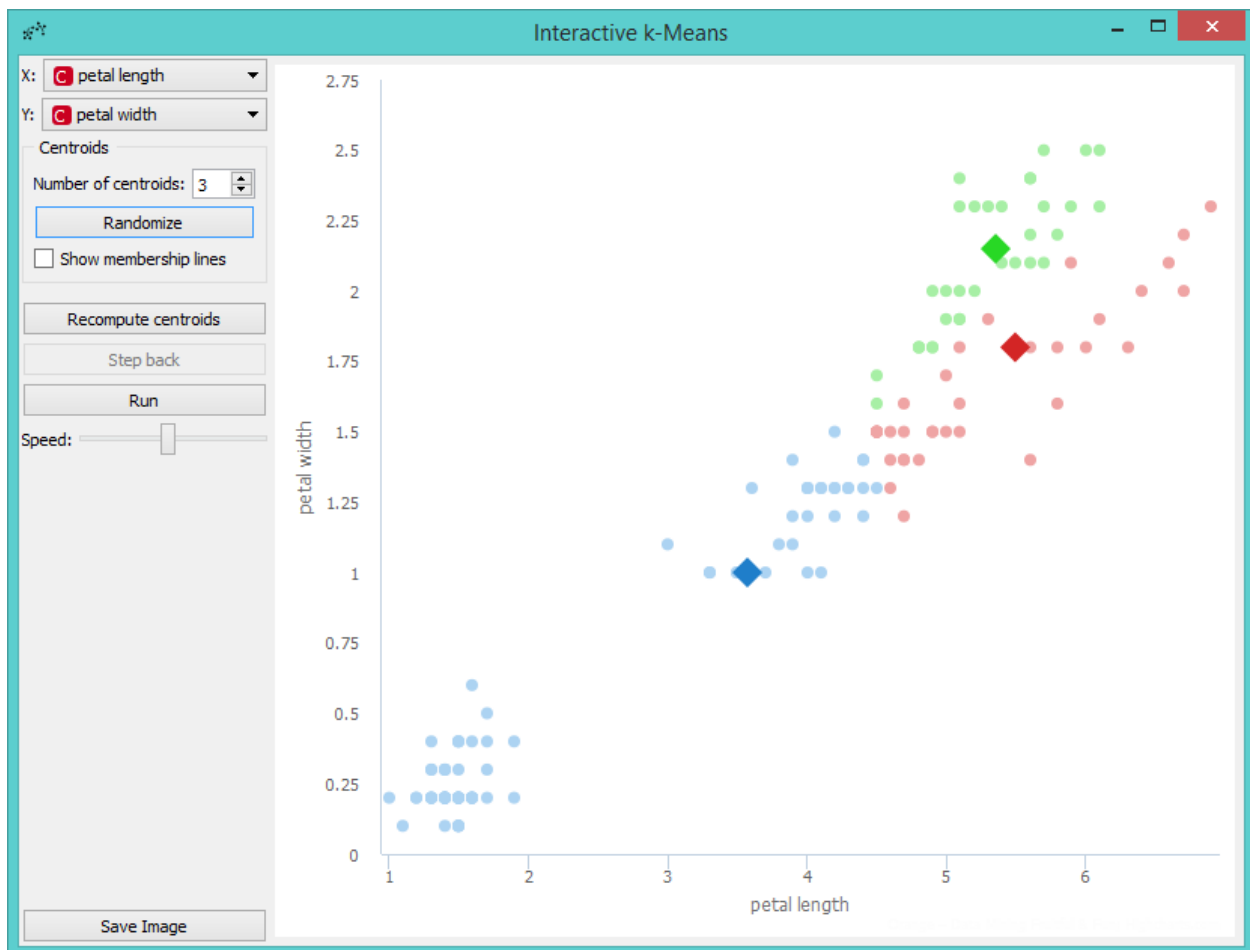
1. Select attributes for **x** and **y** axis.
2. **Number of centroids**: set the number of centroids. **Randomize**: randomly assigns position of centroids. If you want to add centroid on a particular position in the graph, click on this position. If you want to move the centroid, drag and drop it on the desired position. **Show membership lines**: if ticked, connection between data points and closest centroids are shown.
3. **Recompute centroids** or **Reassign membership**: step through different stages of the algorithm. *Recompute centroids* moves centroids to new positions, based on the most central position of the data assigned to the centroid. *Reassign membership* reassigns data points to the centroid they are the closest to. **Step back**: make a step back in the algorithm. **Run**: step through the algorithm automatically. **Speed**: set the speed of automatic stepping.
4. **Save Image** saves the image to the computer in a .svg or .png format.

1.3.2 Example

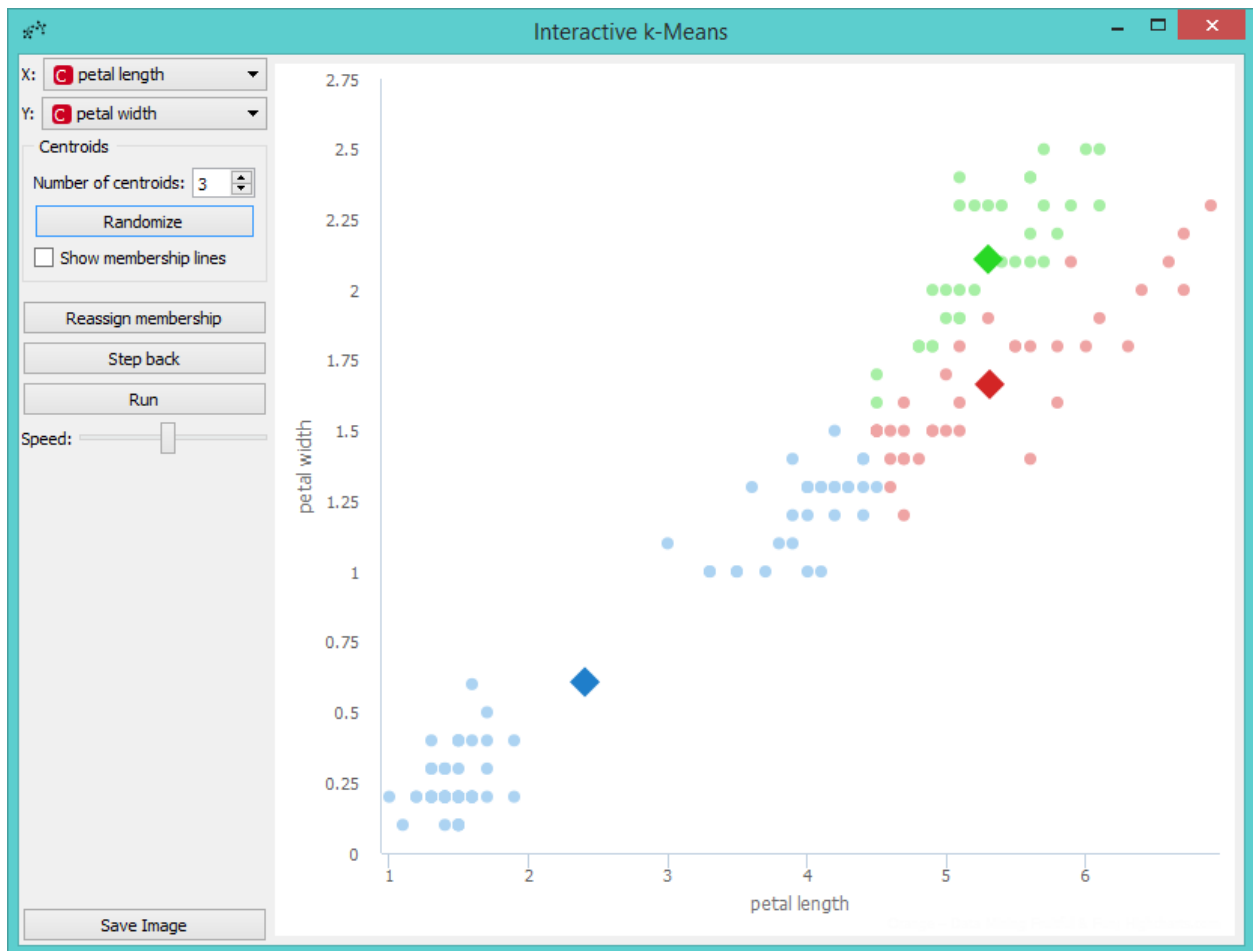
Here are two possible schemas that show how the **Interactive k-Means** widget can be used. You can load the data from **File** or use any other data source, such as **Paint Data**. Interactive k-Means widget also produces a data table with results of clustering and a table with centroids positions. These data can be inspected with the **Data Table** widget.



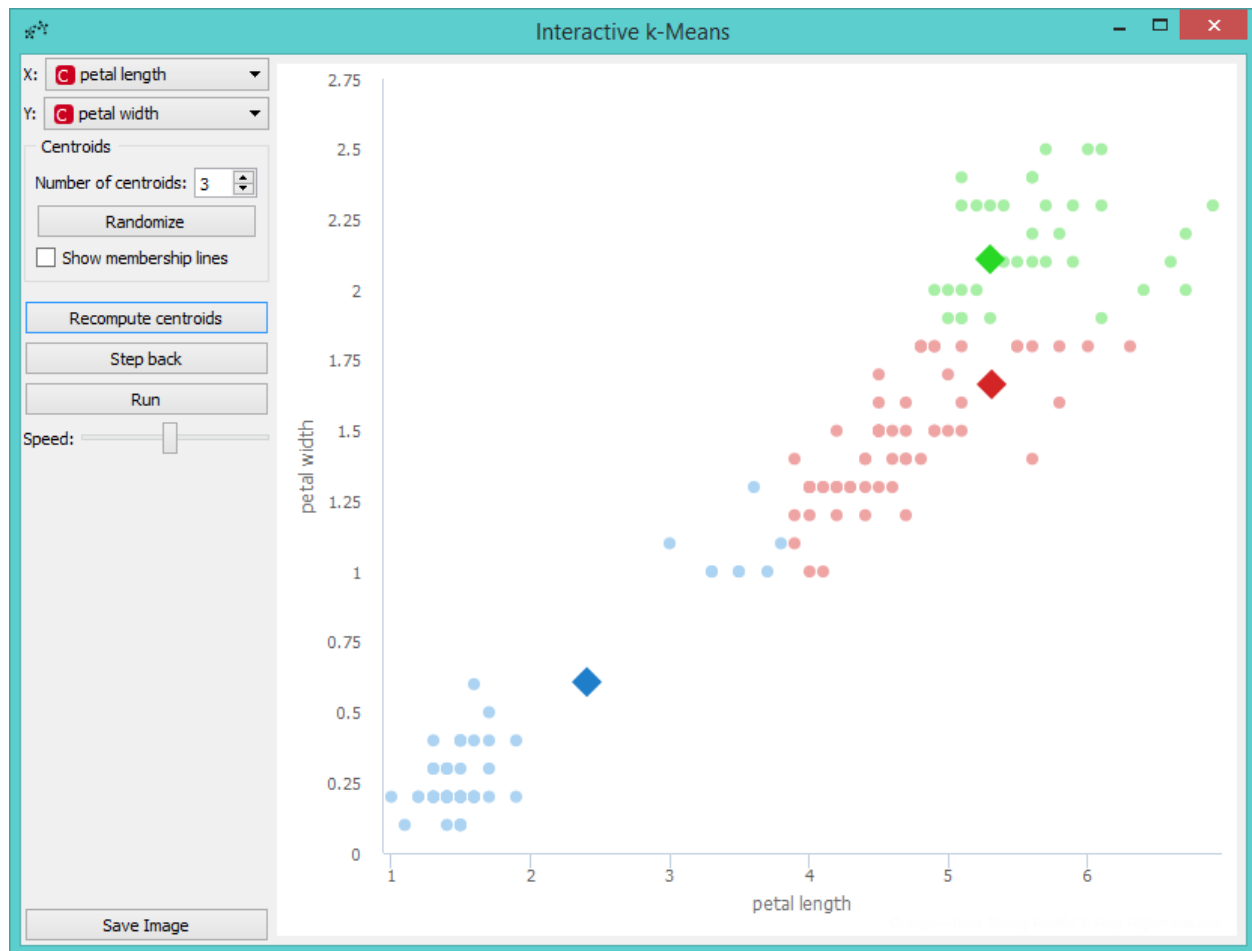
Let us demonstrate the working of the widget on *Iris* data set. We provide the data using **File**. Then we open **Interactive k-Means**. Say, we will demonstrate k-Means on *petal length* and *petal width* attributes, so we set them as *X* and *Y* parameters. We also decided to perform clustering for 3 clusters. This is set as the *Number of centroids*.



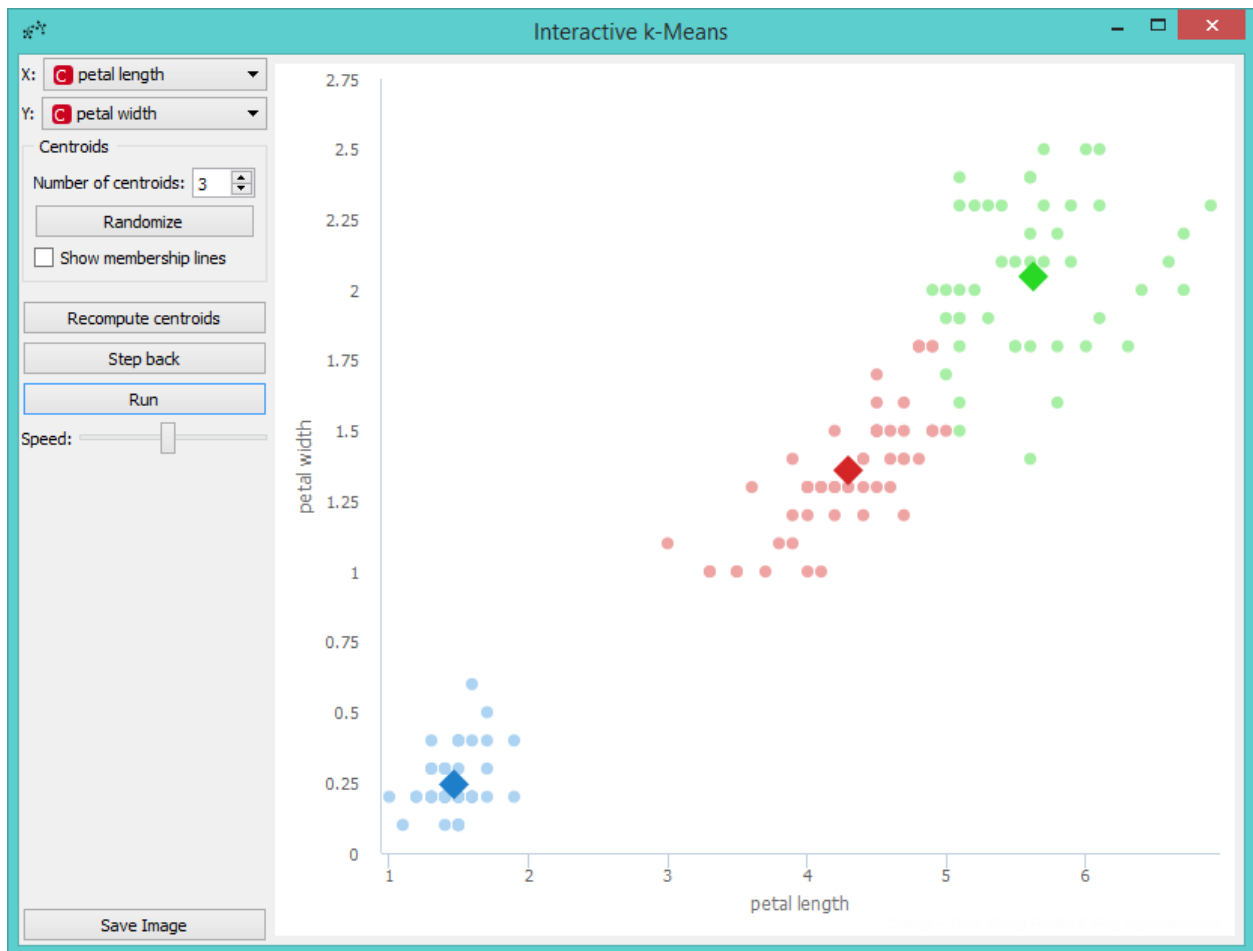
If we are not satisfied with positions of centroids we can change them with a click on the **Randomize** button. Then we perform the first recomputing of centroids with a click on the **Recompute centroids**. We get the following image.



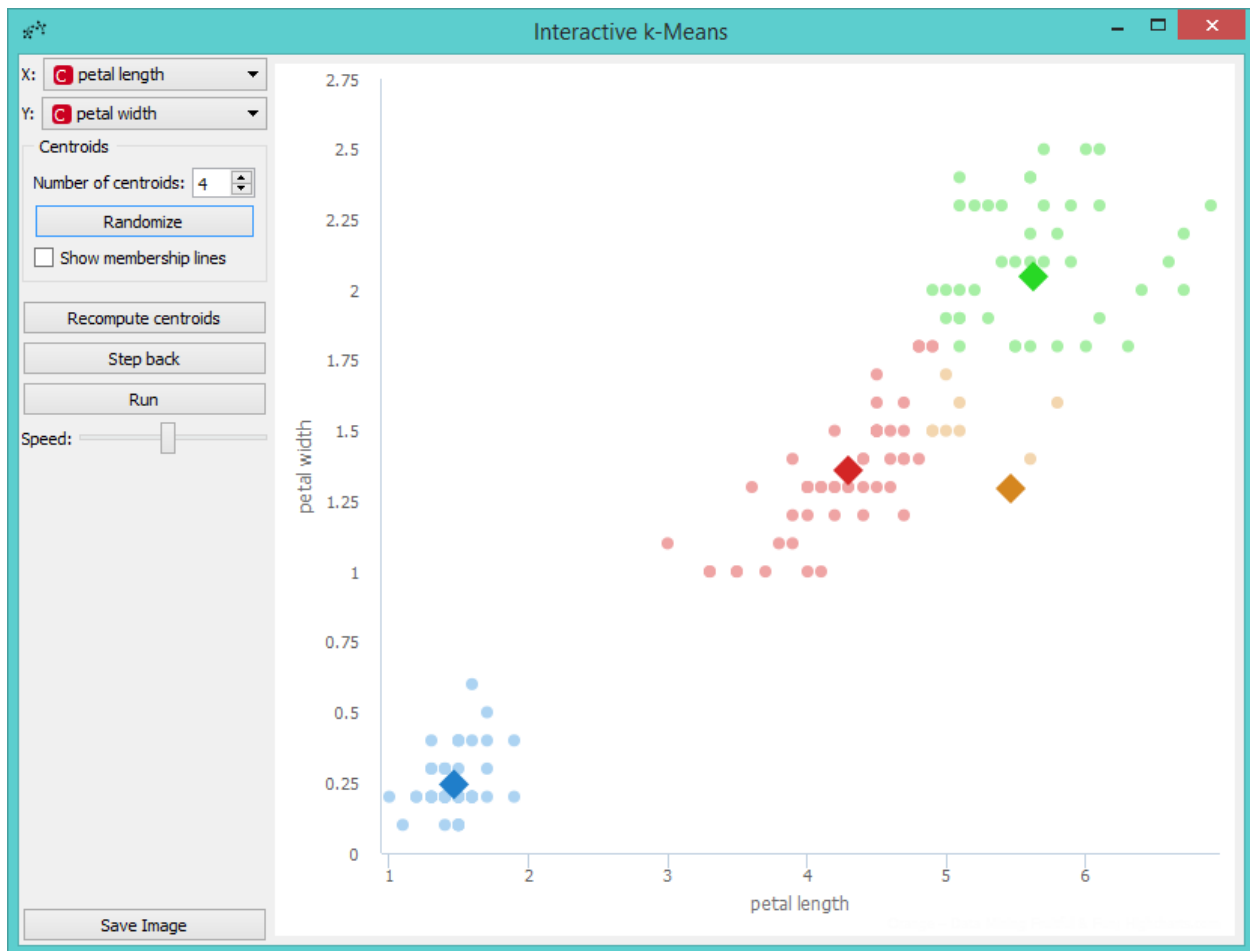
The next step is to reassign membership of all points to the closest centroid. This is performed with a click on the **Reassign membership** button.



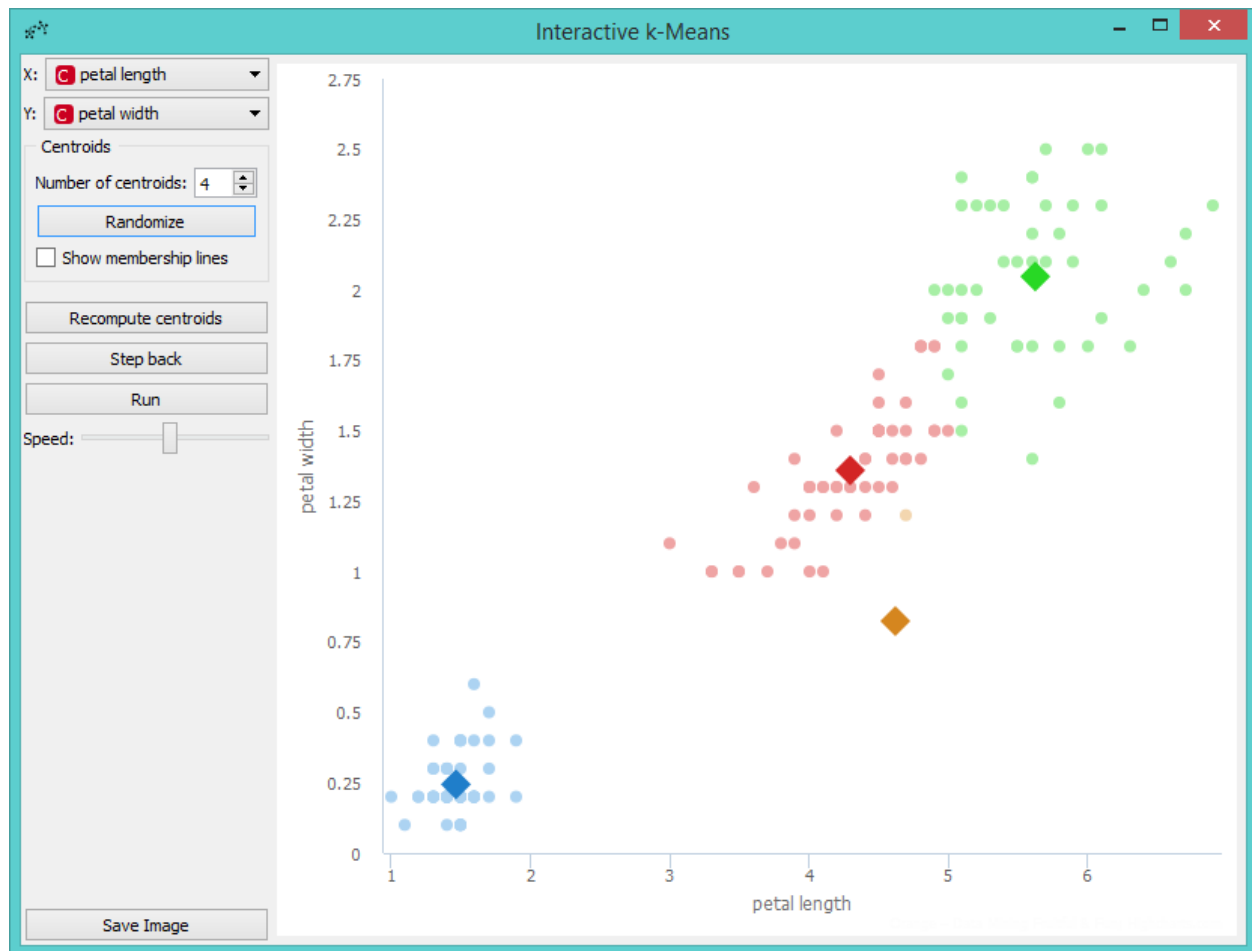
Then we repeat these two steps until the algorithm converges. This is the final result.



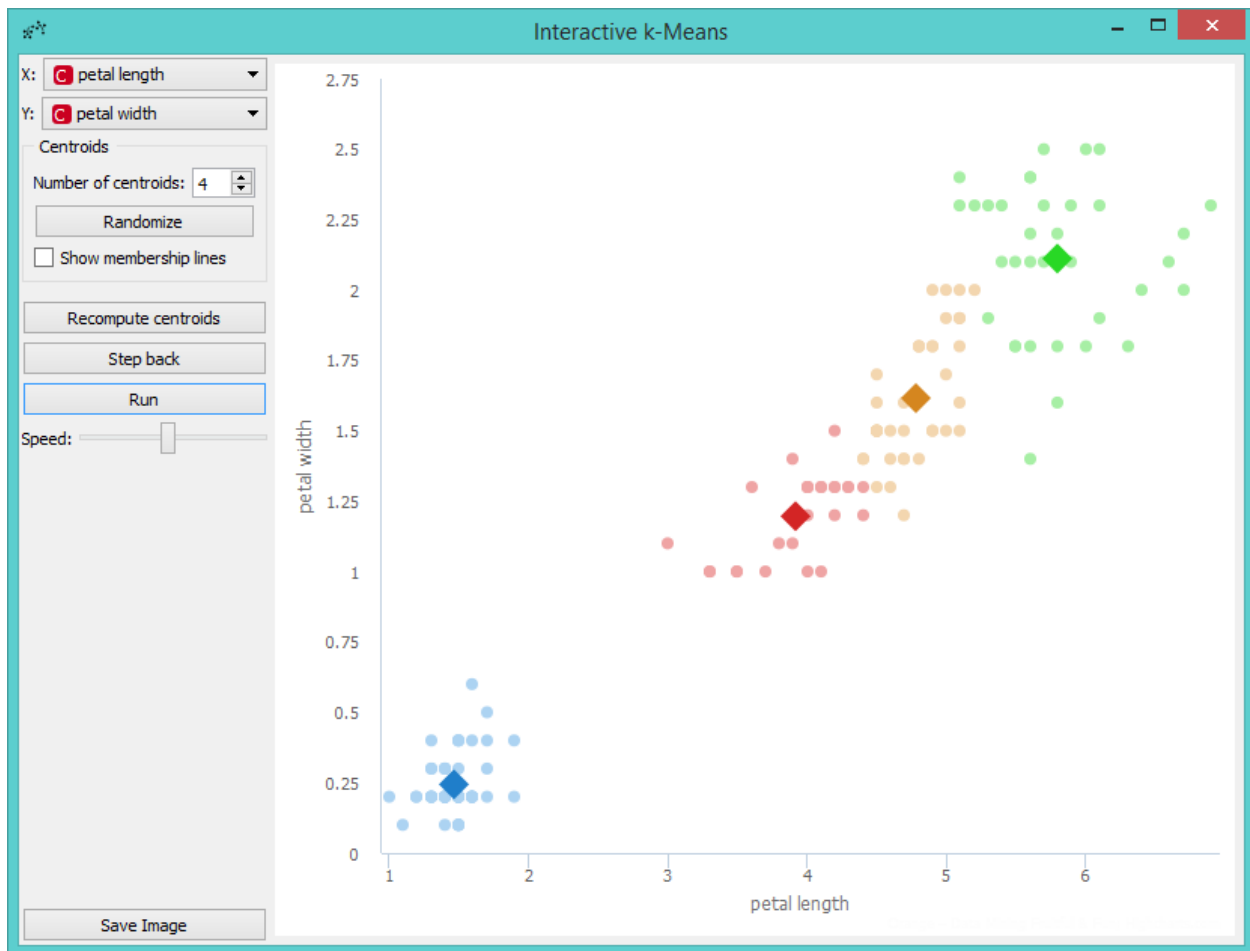
Perhaps we are not satisfied with the result because we noticed that maybe classification into 4 clusters would be better. So we decided to add a new centroid. We can do this by increasing the number of centroids in the control menu or with a click on the position in the graph where we want to place the centroid. We decided to add it with a click. The new centroid is the orange one.



Now we can repeat running the algorithm until it converges again, but before that we will move the new centroid to change the behavior of the algorithm. We grabbed the orange centroid and moved it to the desired position.



Then we press *Run* and observe the centroids while the algorithm converges again.



1.4 Gradient Descent

Educational widget that shows the gradient descent algorithm on a logistic or linear regression.

Inputs

- Data: input data set

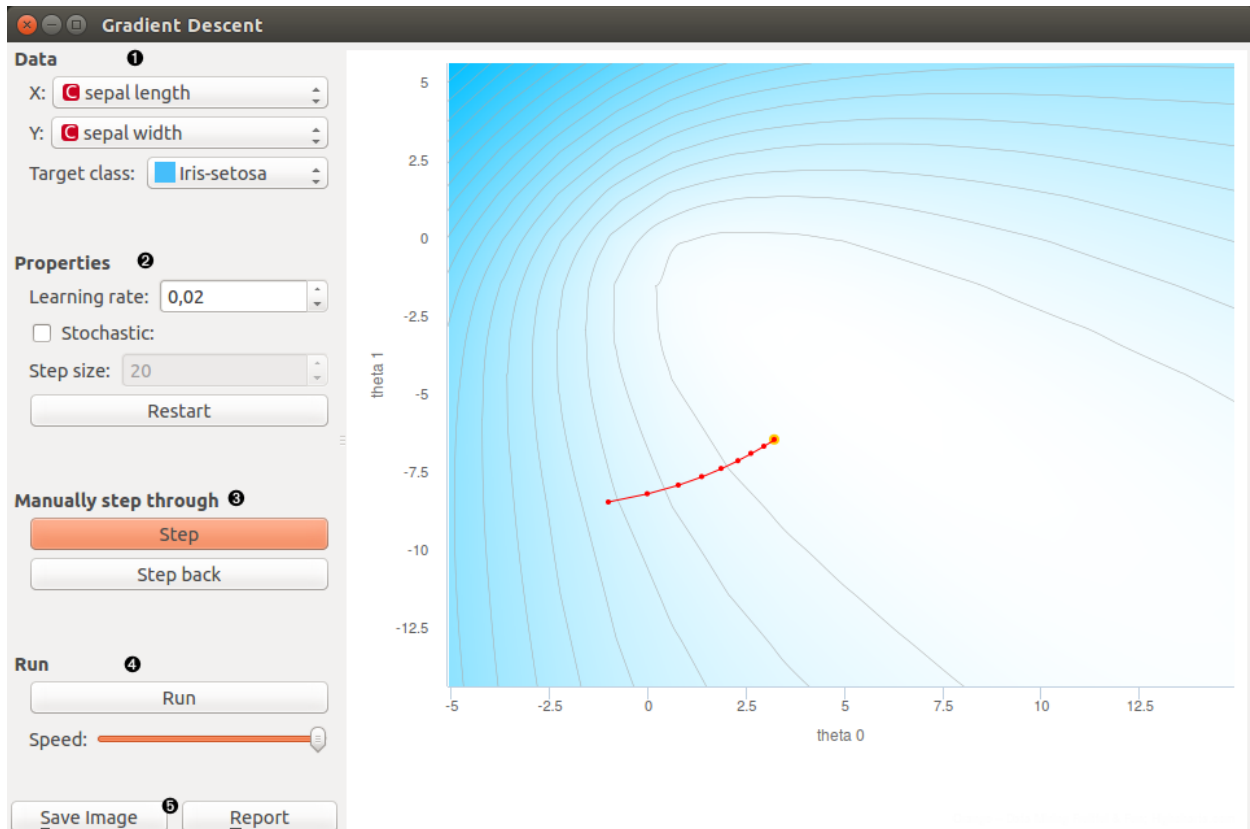
Outputs

- Data: data with columns selected in the widget
- Classifier: model produced at the current step of the algorithm.
- Coefficients: coefficients at the current step of the algorithm.

1.4.1 Description

This widget incrementally shows steps of [gradient descent](#) for a logistic or linear regression. Gradient descent is demonstrated on two attributes that are selected by the user.

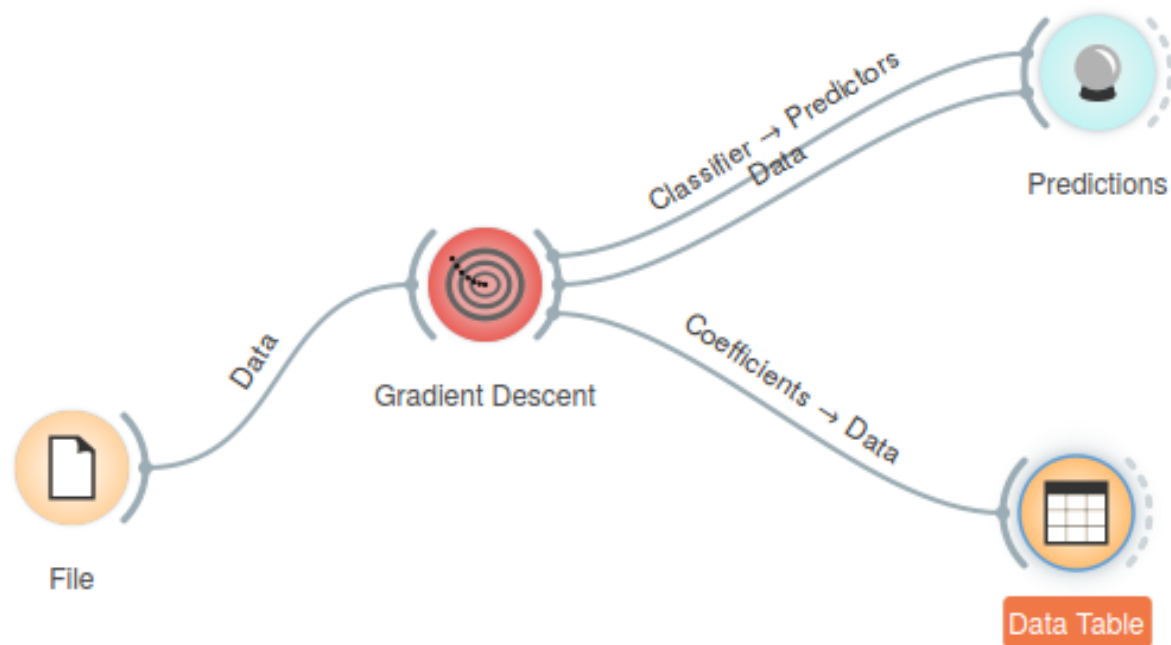
Gradient descent is performed on logistic regression if the class in the data set is categorical and linear regression if the class is numeric.



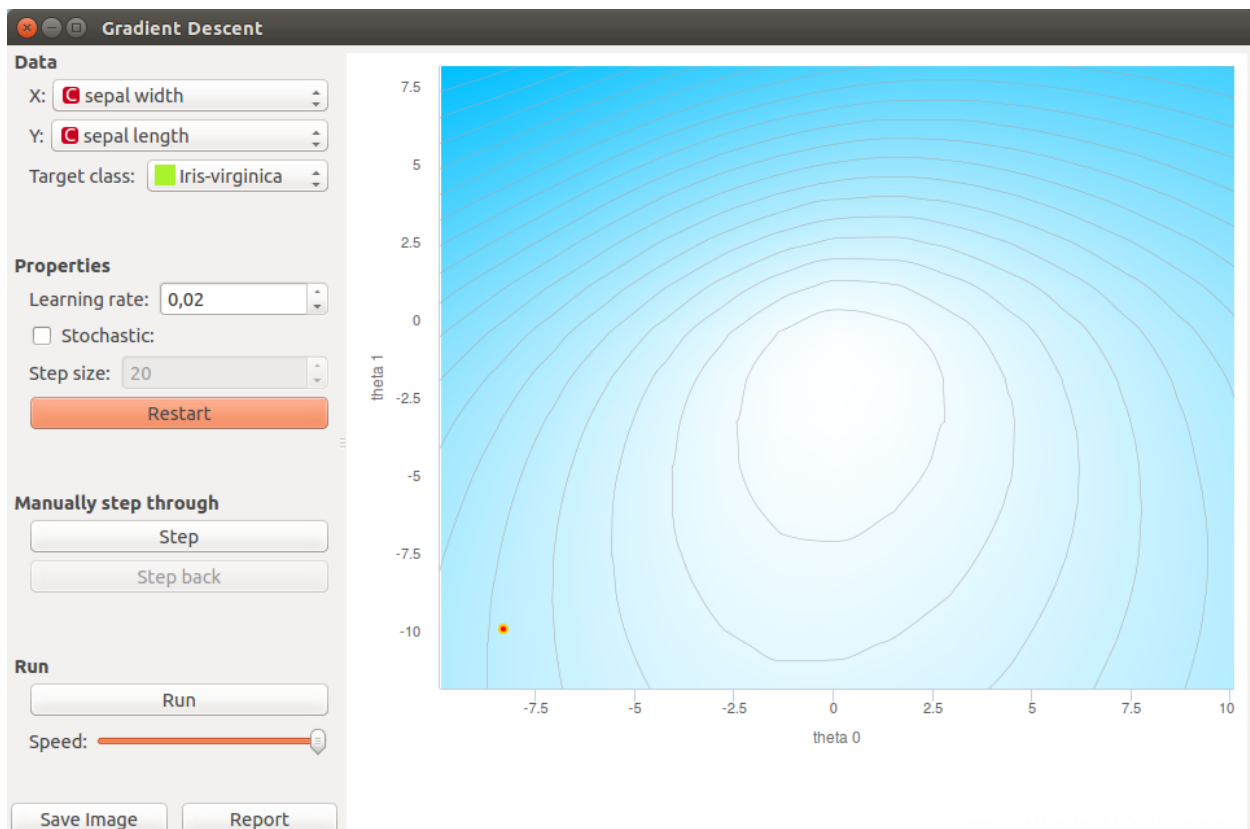
1. Select two attributes (**x** and **y**) on which the gradient descent algorithm is preformed. Select the **target class**. It is the class that is classified against all other classes.
2. **Learning rate** is a step size in the gradient descent With **stochastic** checkbox you can select whether gradient descent is **stochastic** or not. If stochastic is checked you can set **step size** that is amount of steps of stochastic gradient descent performed in one press on step button. **Restart**: start algorithm from the beginning
3. **Step**: perform one step of the algorithm **Step back**: make a step back in the algorithm
4. **Run**: automatically perform several steps until the algorithm converges **Speed**: set speed of the automatic stepping
5. **Save Image** saves the image to the computer in a .svg or .png format. **Report** includes widget parameters and visualization in the report.

1.4.2 Example

In Orange we connected *File* widget with *Iris* data set to the *Gradient Descent* widget. Iris data set has discrete class, so *Logistic regression* will be used this time. We connected outputs of the widget to *Predictions* widget to see how the data are classified and the *Data Table* widget where we inspect coefficients of logistic regression.

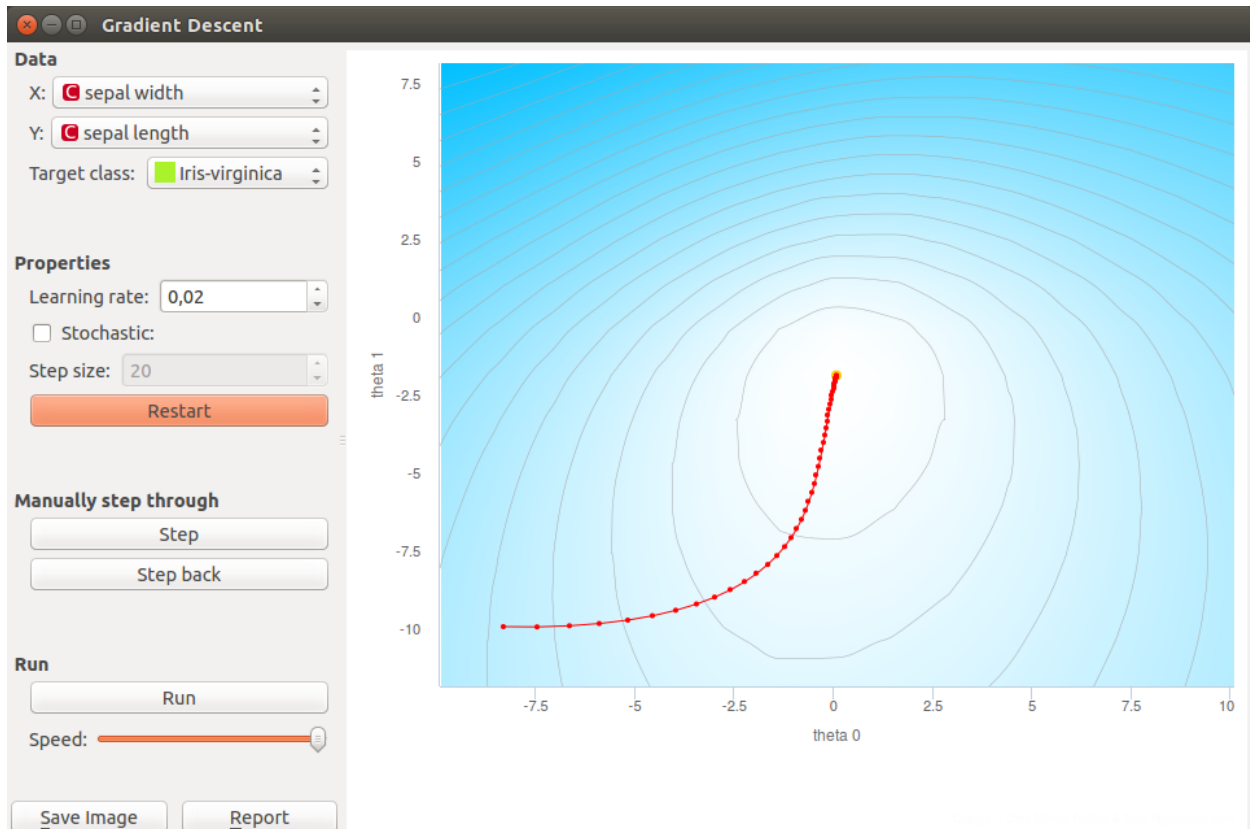


We open the *Gradient Descent* widget and set X to *sepal width* and Y to *sepal length*. Target class is set to *Iris-virginica*. We set *learning rate* to 0.02. With a click in the graph we set the initial coefficients (red dot).

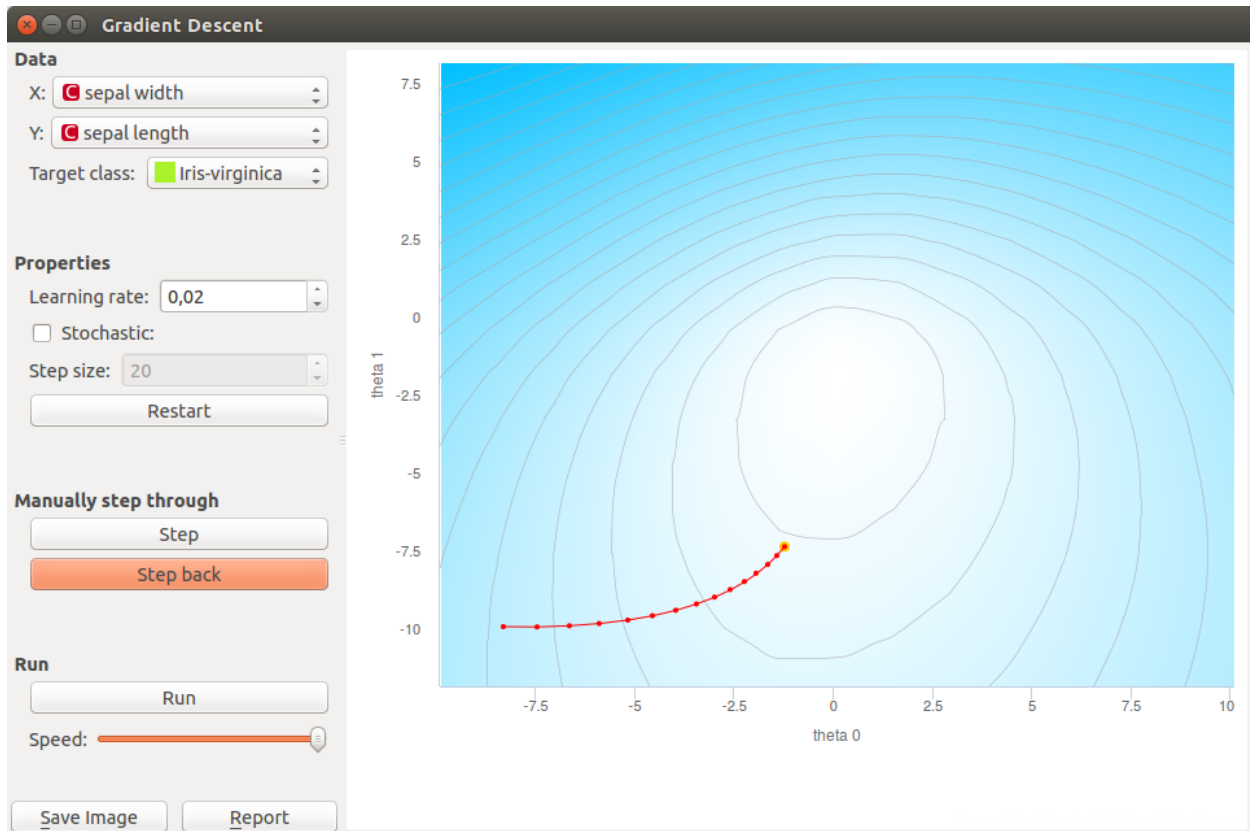


We perform step of the algorithm by pressing the **Step** button. When we get bored with clicking we can finish stepping

by pressing on the **Run** button.



If we want to go back in the algorithm we can do it by pressing **Step back** button. This will also change the model. Current model uses positions of last coefficients (red-yellow dot).



In the end we want to see the predictions for input data so we can open the *Predictions* widget. Predictions are listed in the left column. We can compare this predictions to the real classes.

Predictions

Info
 Data: 150 instances.
 Predictors: 1
 Task: Classification
 Restore Original Order

Options (classification)
☒ Show predicted class
☒ Show predicted probabilities
 Iris-virginica
 Others
☒ Draw distribution bars

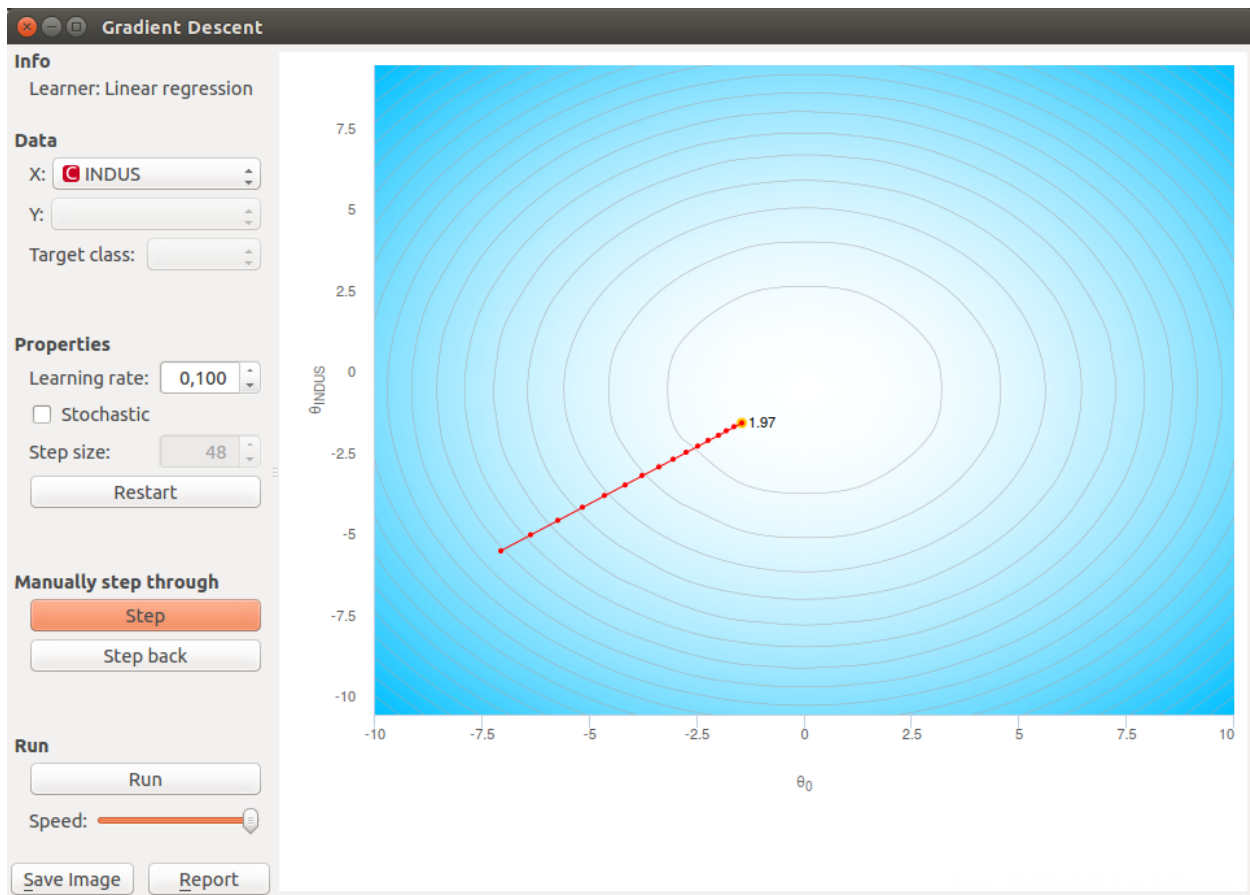
Data View
☒ Show full data set

Output
☒ Original data
☒ Predictions
☒ Probabilities
 Report

	Logistic Regression	iris-bin	iris	sepal width
100	0.43 : 0.57 → Others	Others	Iris-versicolor	-0.588
101	0.72 : 0.28 → Iris-vi...	Iris-virginica	Iris-virginica	0.569
102	0.49 : 0.51 → Others	Iris-virginica	Iris-virginica	-0.819
103	0.94 : 0.06 → Iris-vi...	Iris-virginica	Iris-virginica	-0.125
104	0.73 : 0.27 → Iris-vi...	Iris-virginica	Iris-virginica	-0.356
105	0.81 : 0.19 → Iris-vi...	Iris-virginica	Iris-virginica	-0.125
106	0.98 : 0.02 → Iris-vi...	Iris-virginica	Iris-virginica	-0.125
107	0.13 : 0.87 → Others	Iris-virginica	Iris-virginica	-1.282
108	0.96 : 0.04 → Iris-vi...	Iris-virginica	Iris-virginica	-0.356
109	0.88 : 0.12 → Iris-vi...	Iris-virginica	Iris-virginica	-1.282
110	0.94 : 0.06 → Iris-vi...	Iris-virginica	Iris-virginica	1.263
111	0.80 : 0.20 → Iris-vi...	Iris-virginica	Iris-virginica	0.338
112	0.78 : 0.22 → Iris-vi...	Iris-virginica	Iris-virginica	-0.819
113	0.89 : 0.11 → Iris-vi...	Iris-virginica	Iris-virginica	-0.125
114	0.45 : 0.55 → Others	Iris-virginica	Iris-virginica	-1.282
115	0.49 : 0.51 → Others	Iris-virginica	Iris-virginica	-0.588
116	0.76 : 0.24 → Iris-vi...	Iris-virginica	Iris-virginica	0.338
117	0.81 : 0.19 → Iris-vi...	Iris-virginica	Iris-virginica	-0.125
118	0.98 : 0.02 → Iris-vi...	Iris-virginica	Iris-virginica	1.726
119	0.98 : 0.02 → Iris-vi...	Iris-virginica	Iris-virginica	-1.051
120	0.62 : 0.38 → Iris-vi...	Iris-virginica	Iris-virginica	-1.976
121	0.90 : 0.10 → Iris-vi...	Iris-virginica	Iris-virginica	0.338
122	0.38 : 0.62 → Others	Iris-virginica	Iris-virginica	-0.588

If we want to demonstrate the *linear regression* we can change the data set to *Housing*. This data set has a continuous class variable. When using linear regression we can select only one feature which means that our function is linear. Another parameter that is plotted in the graph is *intercept* of a *linear function*.

This time we selected *INDUS* as an *independent variable*. In the widget we can make the same actions as before. In the end we can also check the predictions for each point with the *Predictions* widget. And check coefficients of linear regression in a *Data Table*.



1.5 Polynomial Regression

Educational widget that interactively shows regression line for different regressors.

Inputs

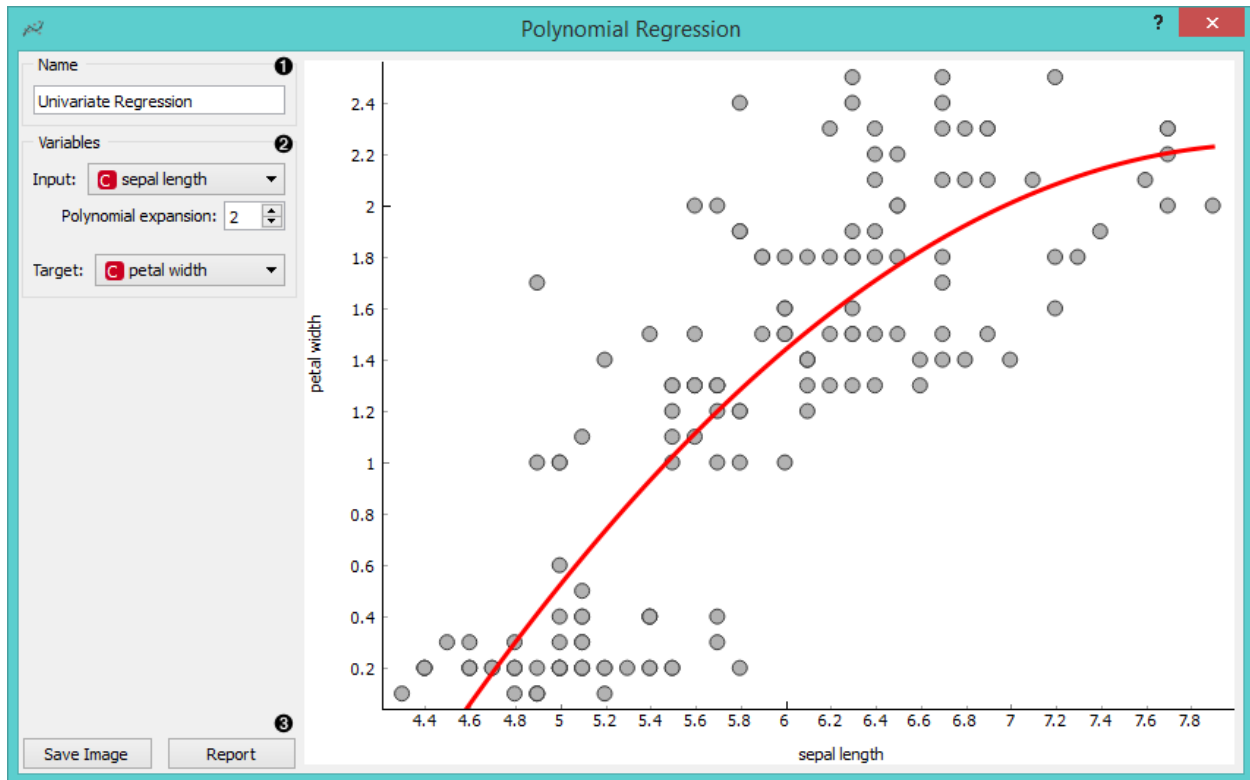
- Data: input data set. It needs at least two continuous attributes.
- Preprocessor: data preprocessors
- Learner: regression algorithm used in the widget. Default set to Linear Regression.

Outputs

- Learner: regression algorithm used in the widget
- Predictor: trained regressor
- Coefficients: regressor coefficients if any

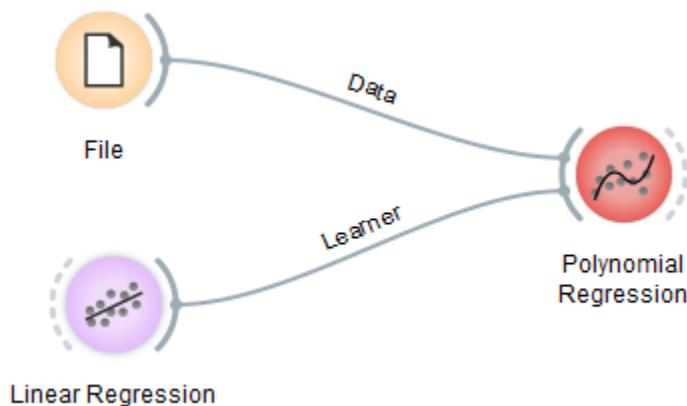
1.5.1 Description

This widget interactively shows the regression line using any of the regressors from the *Model* module. In the widget, [polynomial expansion](#) can be set. Polynomial expansion is a regulation of the degree of the polynomial that is used to transform the input data and has an effect on the shape of a curve. If polynomial expansion is set to 1 it means that untransformed data are used in the regression.

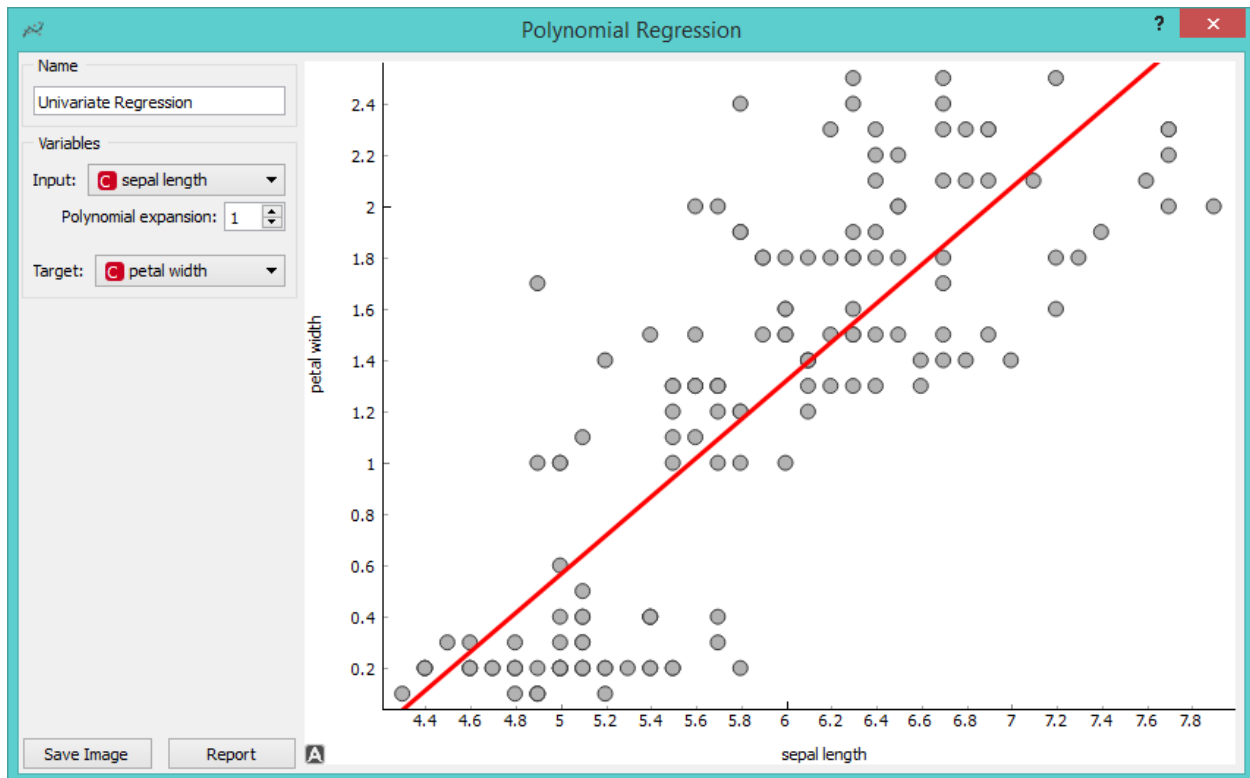


1. Regressor name.
2. *Input*: independent variable on axis x. *Polynomial expansion*: degree of polynomial expansion. *Target*: dependent variable on axis y.
3. *Save Image* saves the image to the computer in a .svg or .png format. *Report* includes widget parameters and visualization in the report.

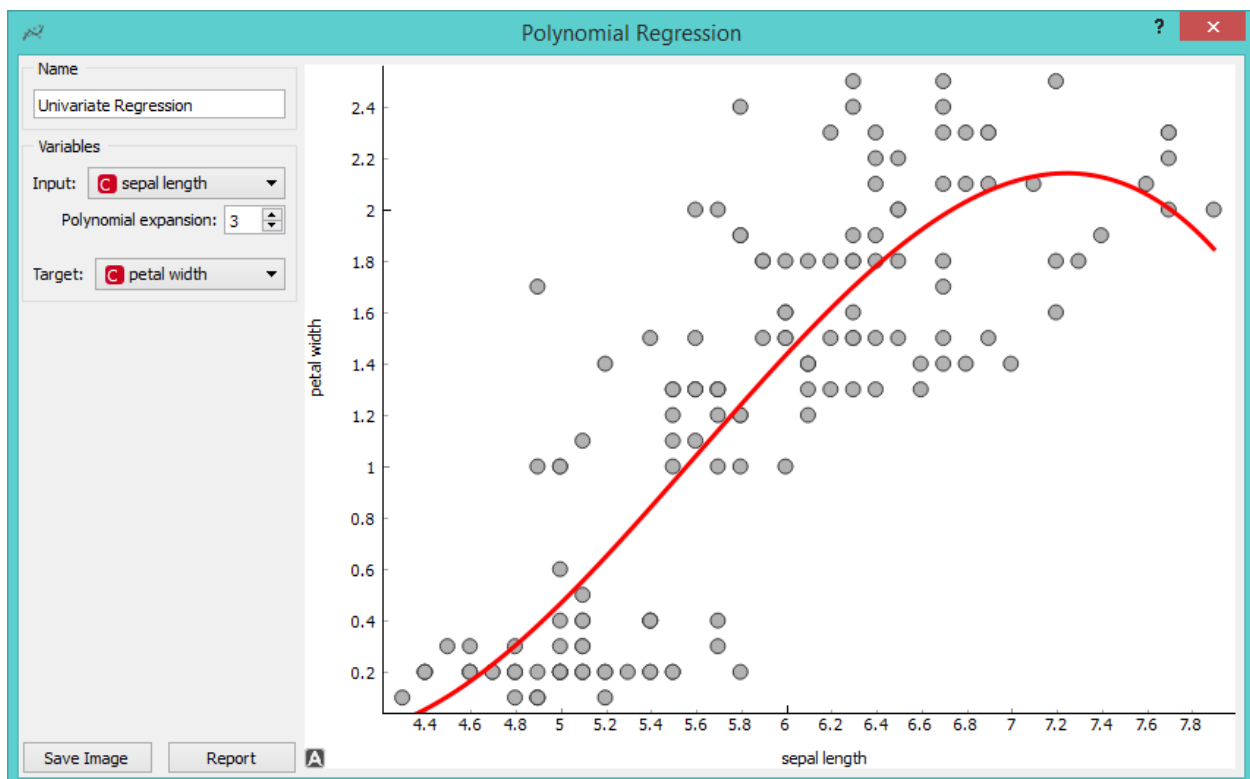
1.5.2 Example



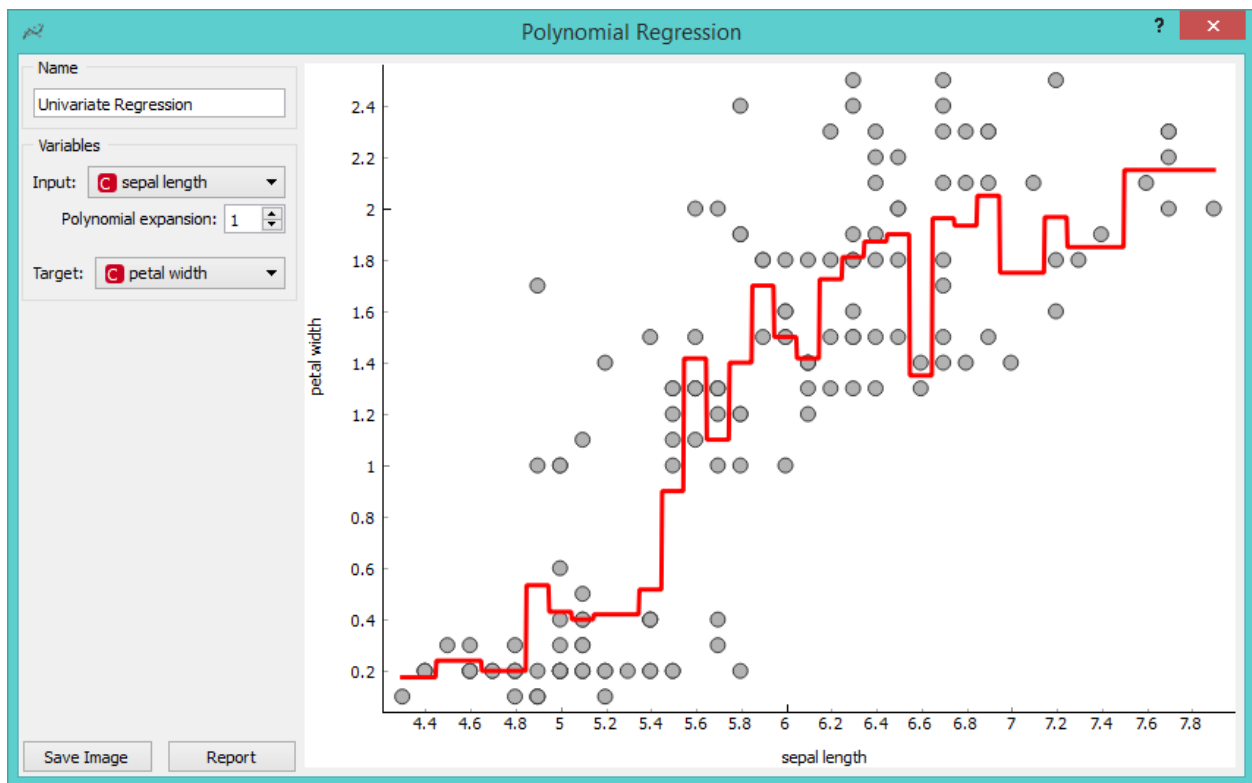
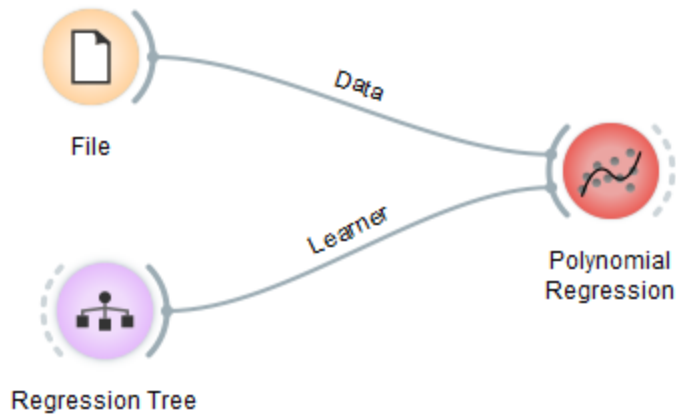
We loaded *iris* data set with the **File** widget. Then we connected **Linear Regression** learner to the **Polynomial Regression** widget. In the widget we selected *petal length* as our *Input* variable and *petal width* as our *Target* variable. We set *Polynomial expansion* to 1 which gives us a linear regression line. The result is shown in the figure below.



The line can fit better if we increase the **Polynomial expansion** parameter. Say, we set it to 3.



To observe different results, change **Linear Regression** to any other regression learner from Orange. Example below is done with the **Tree** learner.



1.6 Polynomial Classification

Educational widget that visually demonstrates classification in two-dimensional space.

Inputs

- Data: input data set
- Preprocessor (optional): data preprocessors
- Learner (optional): classification algorithm used in the widget (default: Logistic Regression)

Outputs

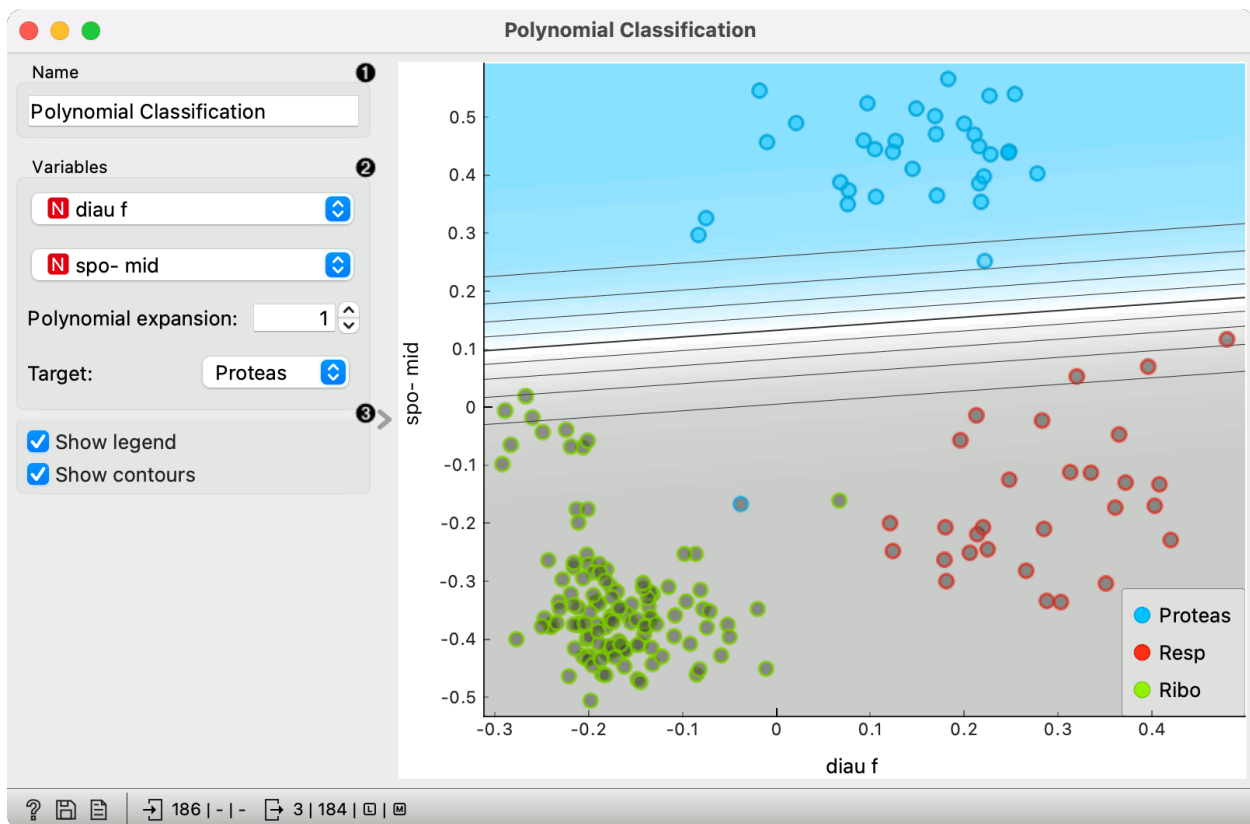
- Learner: classification algorithm used in the widget
- Classifier: trained classifier
- Coefficients: classifier coefficients if it has them

1.6.1 Description

This widget interactively shows classification probabilities using contours and color gradient for any classifier. The widget is particularly useful for showing effects of polynomial expansion (by adding terms like $x_i y_j$ where $i + j$ is at most the selected degree) and of regularization*.

By default, the widget uses non-regularized logistic regression. Manually attaching learner, for instance the Logistic Regression widget, allows us to control the regularization strength.

The outline of the shown data points indicates the actual class, and the inside shows the prediction by the model. In non-binary classification, points predicted to non-target classes are painted gray.



1. Classifier name.
2. Variables: variables used for classification; options shown only if data contains more than two independent variables. *Polynomial expansion*: Degree of polynomial expansion. *Target class*: the target to which the shown probabilities apply. In non-binary classification, other classes are merged.
3. *Show legend*: Show color legend. *Show contours*: Show contour lines for probabilities.

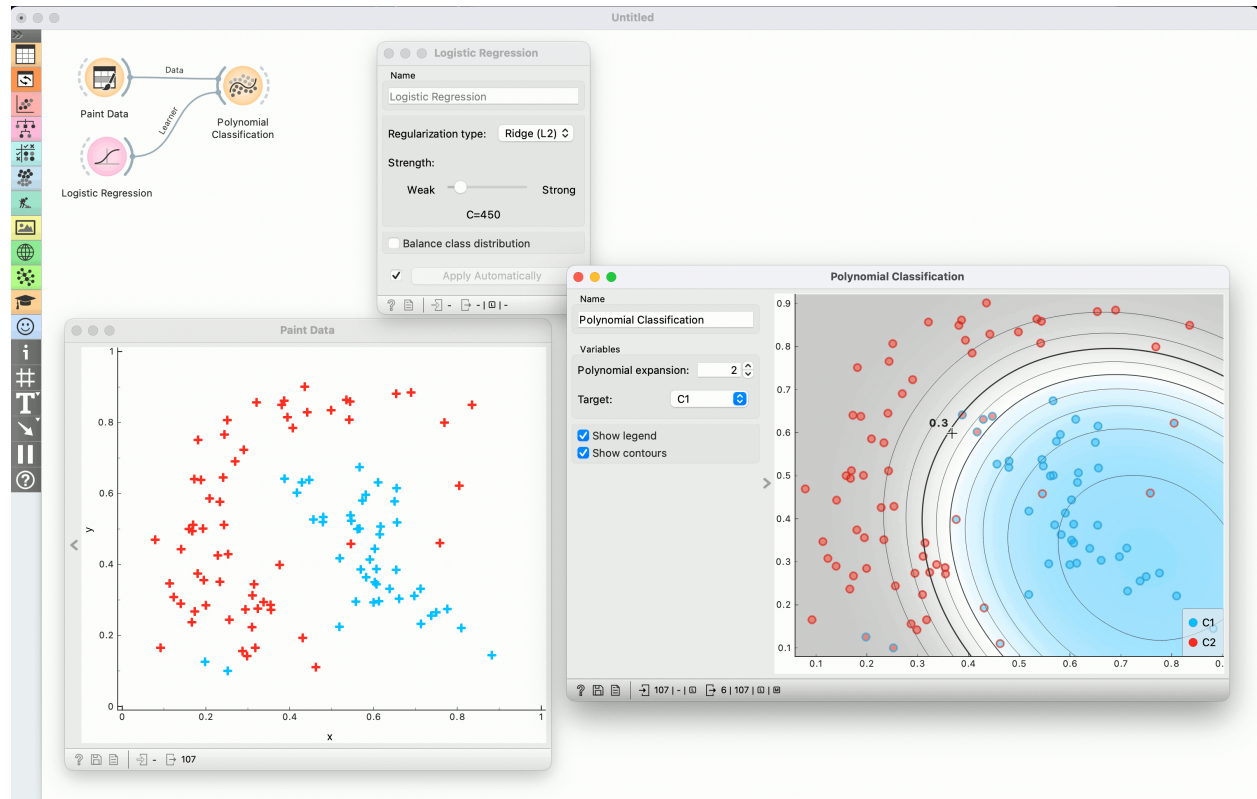
1.6.2 Example

We painted some data using the Paint widget and fed it to Polynomial Classification. We also added Logistic Regression to control the regularization.

Setting the polynomial expansion to 2 allows the classifier to construct boundaries as 2 degree polynomials. Hovering over a contour line shows the predicted probability of the target class (in this case C1) for points on that line. Moving the mouse elsewhere shows a probability at some particular point.

Red points with blue outline are “blue” data instances that are misclassified as red, and vice versa.

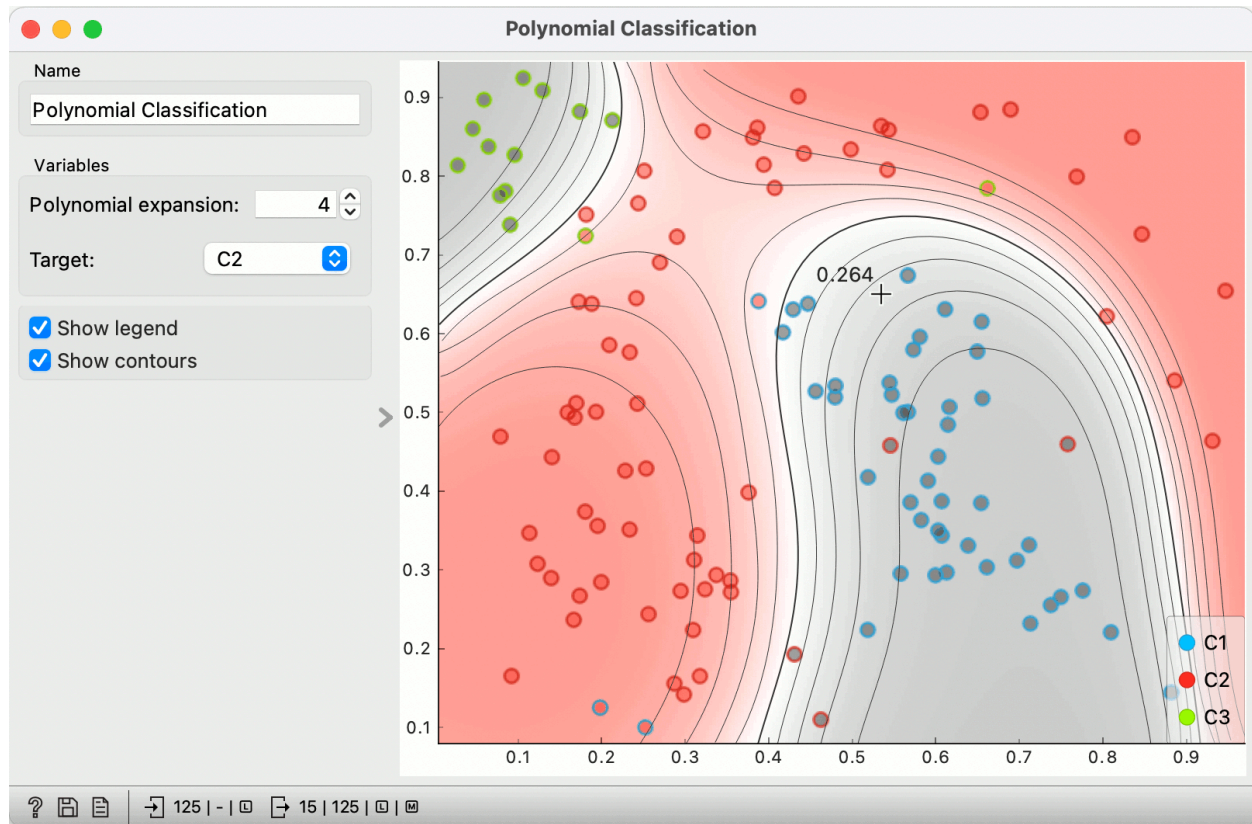
Changing the regularization (in Logistic regression widget) allows us to observe how the contour lines spread and shrink.



We added another class, chose C2 as the target, increased the polynomial expansion to 4 and weaken the regularization (in Logistic regression widget).

Outlines still represent the original classes. Instances of the target classes are colored red, the other two classes are gray.

Hovering at any point shows us the probability for red (e.g. 0.264).



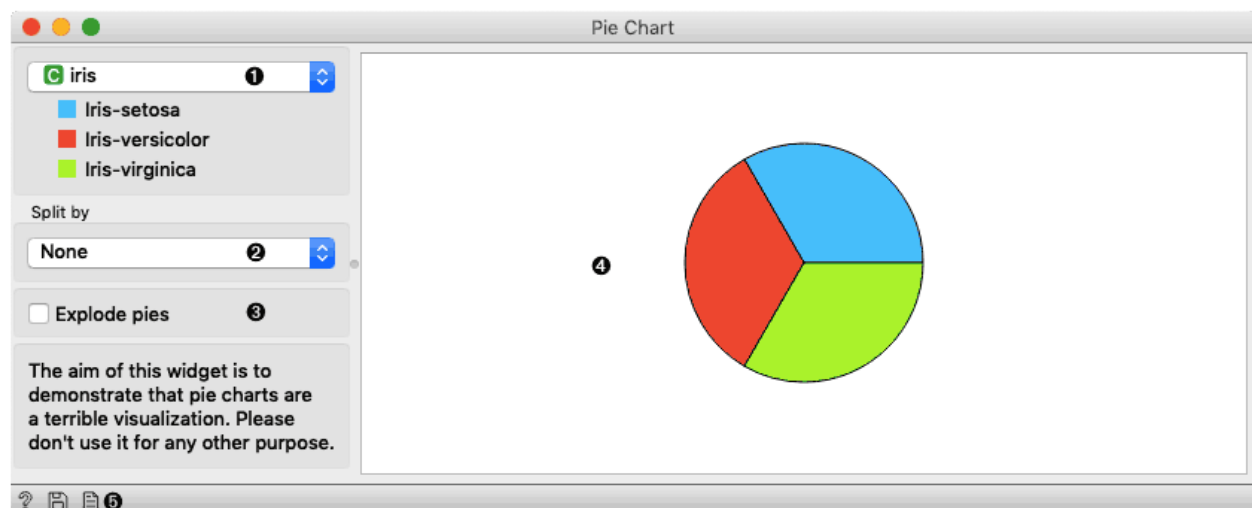
1.7 Pie Chart

The widget for visualizing discrete attributes in the pie chart.

Inputs

- Data: input data set

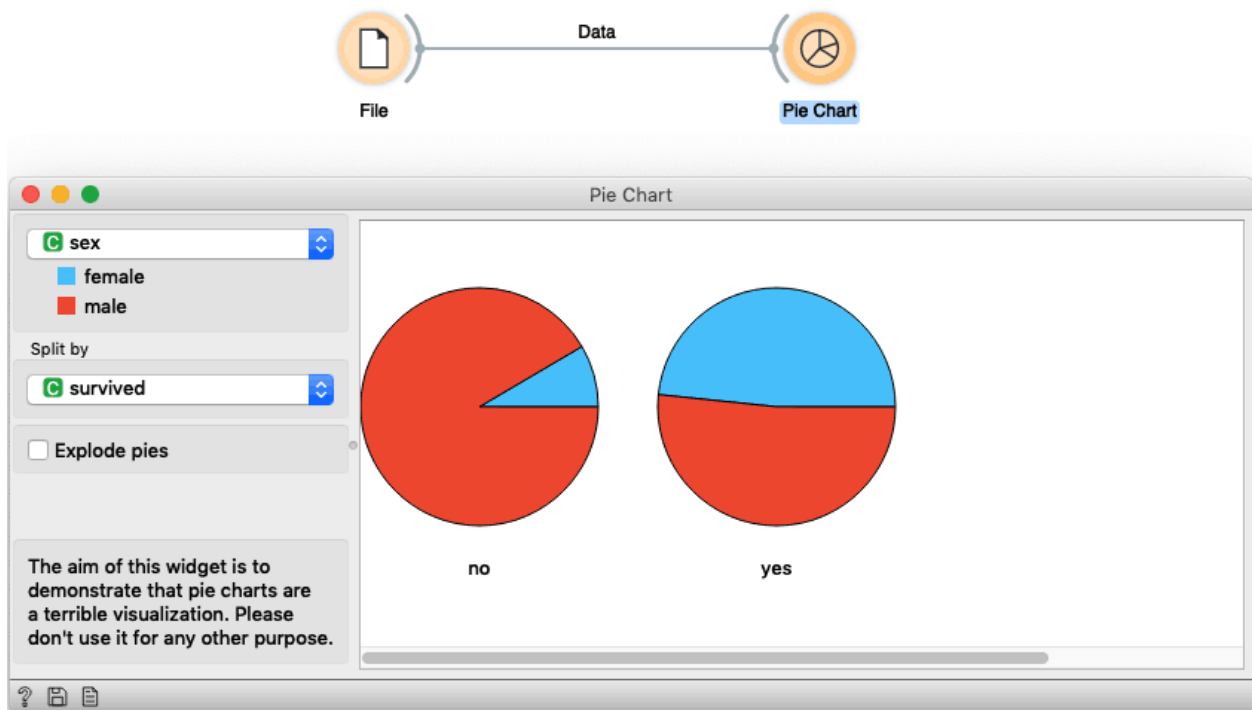
The aim of this widget is to demonstrate that pie charts are a terrible visualization. Please don't use it for any other purpose.



1. Select the attribute you want to visualize.
2. Select the attribute which is used to split data in more charts.
3. Check if you want pies to be exploded (parts of the pie will have space in between).
4. You will see your data visualized here.
5. With those buttons, you can either get help, save the plot, or include plots in the report.

1.7.1 Example

We load the Titanic dataset in File widget and connected the data to Pie Chart. Here we show the distribution of gender data and split pies by survived attributes. We notice that in the group of passengers that did not survive there are mainly male while there is a higher proportion of women in the group of people that survived. While the pie chart can shed some light of data we still suggest using more informative visualizations, e.g. Box Plot.



1.8 Random Data

Generate random data sample.

Inputs

- None

Outputs

- Data: randomly generated data

Random Data allows creating random data sets, where variables correspond to the selected distributions. The user can specify the number of rows (samples) and the number of variables for each distribution. Distributions from the Scipy's *stats* module are used.

1. **Normal**: A normal continuous random variable. Set the number of variables, the mean and the variance.
2. **Bernoulli**: A Bernoulli discrete random variable. Set the number of variables and the probability mass function.
3. **Binomial**: A binomial discrete random variable. Set the number of variables, the number of trials and probability of success.
4. **Uniform**: A uniform continuous random variable. Set the number of variables and the lower and upper bound of the distribution.
5. **Discrete uniform**: A uniform discrete random variable. Set the number of variables and the number of values per variable.
6. **Multinomial**: A multinomial random variable. Set the probabilities and the number of trials. The probabilities should sum to one. The number of probabilities corresponds to the final number of variables generated.
7. **Add more variables...** enables selecting new distributions from the list and with that adding additional variables. Distributions can be removed by pressing an X in the top left corner of each distribution.
8. Define the sample size (i.e. number of rows, default 1000) and press *Generate* to output the data set.

Random Data

Hypergeometric distribution ①

Variables Number of objects
 Name prefix Number of positives
 Number of trials

Negative binomial distribution ②

Variables Number of successes
 Name prefix Probability of success

Poisson distribution ③

Variables Event rate (λ)
 Name prefix

Exponential distribution ④

Variables
 Name prefix

Gamma distribution ⑤

Variables Shape (α)
 Name prefix Scale

Student's t distribution ⑥

Variables Degrees of freedom
 Name prefix

Bivariate normal distribution ⑦

Variables Mean x
 Name prefix Variance x
 Mean y
 Variance y
 Covariance

Add more variables ...

Sample size

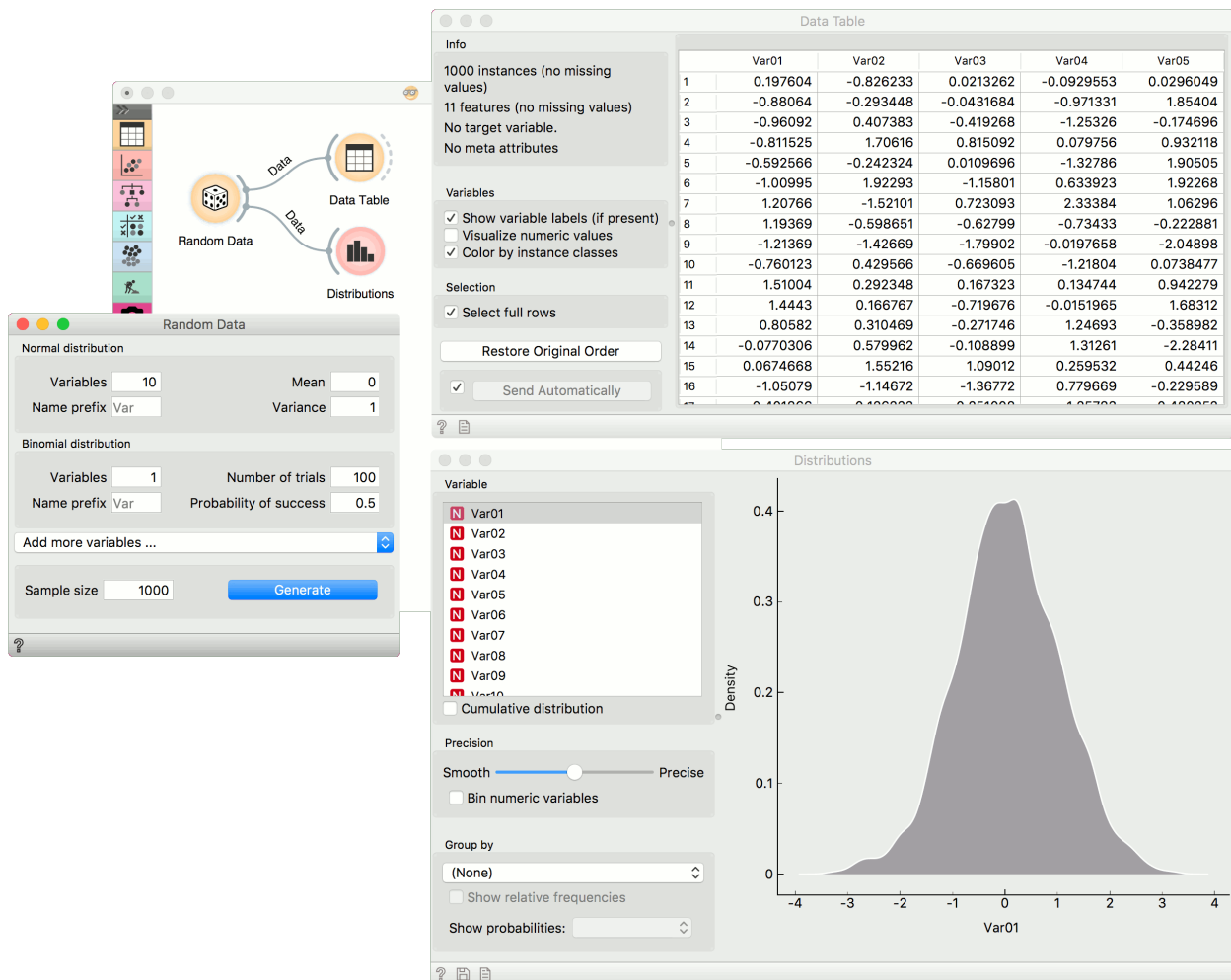
1. **Hypergeometric**: A hypergeometric discrete random variable. Set the number of variables, number of objects, positives and trials.
2. **Negative binomial**: A negative binomial discrete random variable. Set the number of variables, number of successes and the probability of a success.
3. **Poisson**: A Poisson discrete random variable. Set the number of variables and the event rate (expected number of occurrences).
4. **Exponential**: An exponential continuous random variable. Set the number of variables.
5. **Gamma**: A gamma continuous random variable. Set the number of variables, the shape and scale. The larger the scale parameter, the more spread out the distribution.
6. **Student's t**: A Student's t continuous random variable. Set the number of variables and the degrees of freedom.
7. **Bivariate normal**: A multivariate normal random variable where the number of variables is fixed to 2. The number of variables is set to two and cannot be changed. Set the mean and variance of each variable and the

covariance matrix of the distribution.

1.8.1 Example

We normally wouldn't create a data set with so many different distributions but rather, for instance, a set of normally distributed variables and perhaps a binary variable, which we will use as the target variable. In this example, we use the default settings, which generate 10 normally distributed variables and a single binomial variable.

We observe the generated data in a **Data Table** and in **Distributions**.



CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`