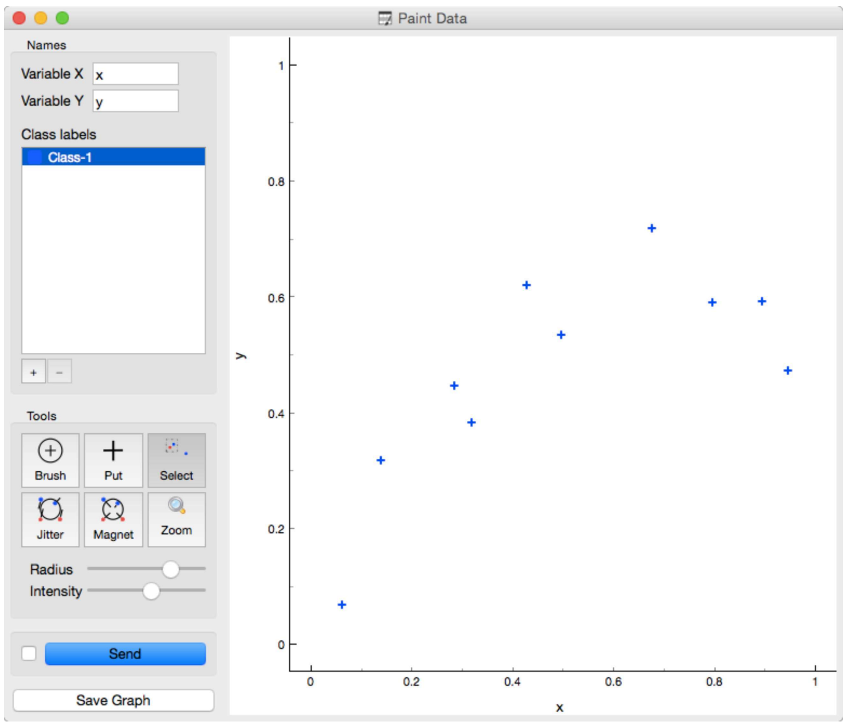
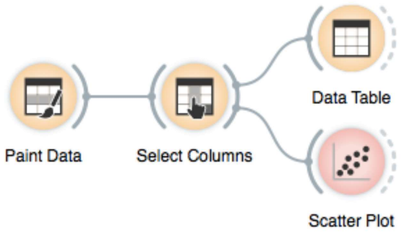
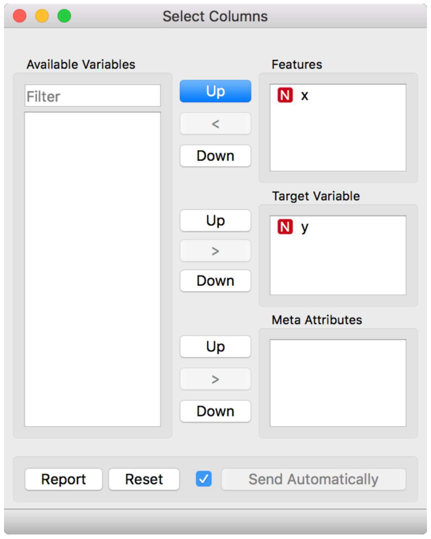
Lesson 17: Linear Regression

In the Paint Data widget, remove the Class-2 label from the list. If you have accidentally left it while painting, don't despair. The class variable will appear in the Select Columns widget, but you can "remove" it by dragging it into the Available Variables list.

For a start, let us construct a very simple data set. It will contain a just one continuous input feature (let's call it x) and a continuous class (let's call it y). We will use Paint Data, and then reassign one of the features to be a class by using Select Column and moving the feature y from the list of "Features" to a field with a target variable. It is always good to check the results, so we are including Data Table and Scatter Plot in the workflow at this stage. We will be modest this time and only paint 10 points and will use Put instead of the Brush tool.







Features: x

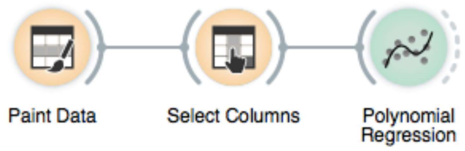
Target Variable: y

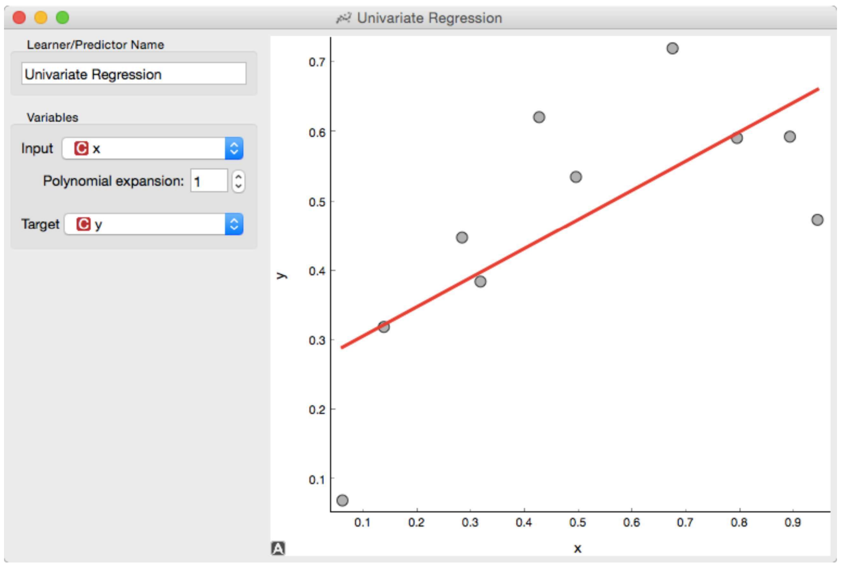
Meta Attributes:

We would like to build a model that predicts the value of class y from the feature x. Say that we would like our model to be linear, to mathematically express it as $h(x)=\theta_0+\theta_1 x$. Oh, this is the equation of a line. So we would like to draw a line through our data points. The θ_0 is then an intercept, and θ_1 is a slope. But there are many different lines we could draw. Which one is the best one? Which one is the one that fits our data the most?

The question above requires us to define what a good fit is. Say, this could be the error the fitted model (the line) makes when it predicts the value of y for a given data point (value of x). The prediction is h(x), so the error is h(x) - y. We should treat the negative and positive errors equally, plus, let us agree, we would prefer punishing larger errors more severely than smaller ones. Therefore, it is perfectly ok if we square the errors for each data point and then sum them up. We got our objective function! Turns out that there is only one line that minimizes this function. The procedure that finds it is called linear regression. For cases where we have only one input feature, Orange has a special widget in the educational add-on called Polynomial Regression.

Do not worry about the strange name of the widget Polynomial Regression, we will get there in a moment.





Learner/Predictor Name: Univariate Regression

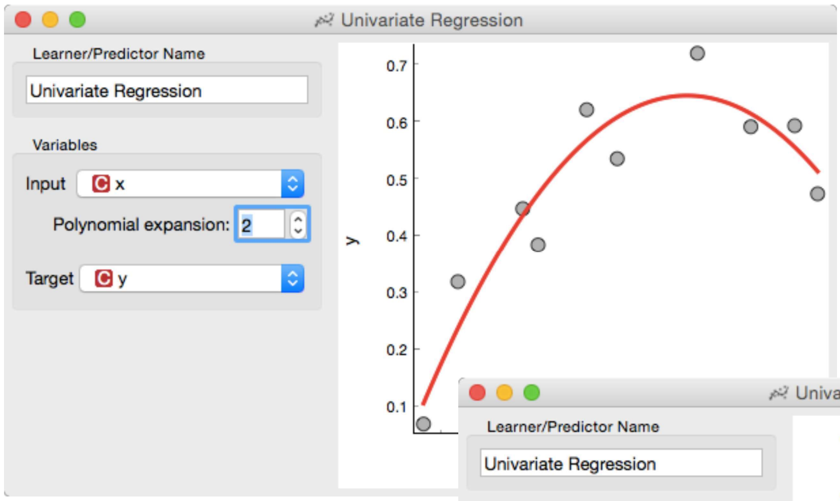
Input: x

Polynomial expansion: 1

Target: y

Looks ok. Except that these data points do not appear exactly on the line. We could say that the linear model is perhaps too simple for our data sets. Here is a trick: besides column x, the widget Univariate Regression can add columns x^2, x^3... x^n to our data set. The number n is a degree of polynomial expansion the widget performs. Try setting this number to higher values, say to 2, and then 3, and then, say, to 9. Witho the degree of 3, we are then fitting the data to a linear function $h(x) = \theta_0 + \theta_1 x + \theta_1 x^2 + \theta_1 x^3$.

The trick we have just performed (adding the higher order features to the data table and then performing linear regression) is called Polynomial Regression. Hence the name of the widget. We get something reasonable with polynomials of degree 2 or 3, but then the results get really wild. With higher degree polynomials, we totally overfit our data.



Univariate Regression

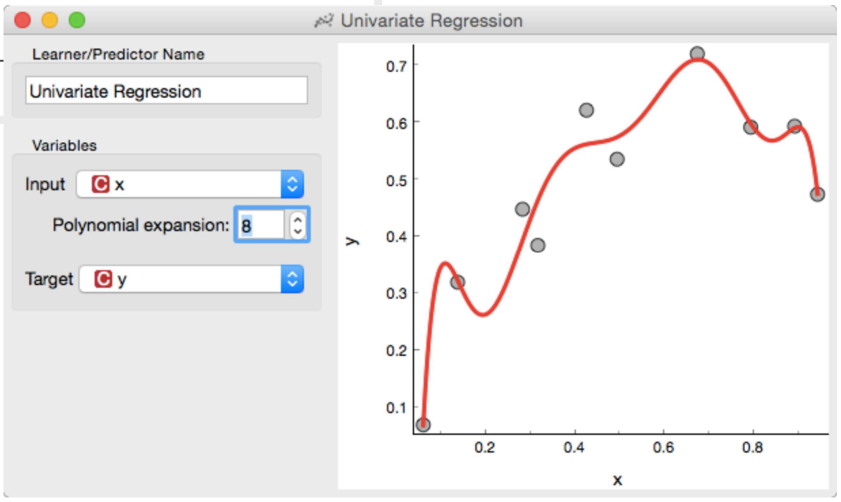
Learner/Predictor Name: Univariate Regression

Variables

Input: x

Polynomial expansion: 2

Target: y



Univariate Regression

Learner/Predictor Name: Univariate Regression

Variables

Input: x

Polynomial expansion: 8

Target: y

It is quite surprising to see that linear regression model can result in fitting non-linear (univariate) functions. That is, functions with curves, such as those on the figures. How is this possible? Notice though that the model is actually a hyperplane (a flat surface) in the space of many features (columns) that are powers of x. So for the degree 2, $\mathbf{h}(x)=\theta_0+\theta_1 x+\theta_1 x^2$ is a (flat) hyperplane. The visualization gets curvy only once we plot $\mathbf{h}(x)$ as a function of x.

Overfitting is related to the complexity of the model. In polynomial regression, the models are defined through parameters θ. The more parameters, the more complex is the model.

Obviously, the simplest model has just one parameter (an intercept), ordinary linear regression has two (an intercept and a slope), and polynomial regression models have as many parameters as is the degree of the polynomial. It is easier to overfit with a more complex model, as this can adjust to the data better. But is the overfitted model really discovering the true data patterns? Which of the two models depicted in the figures above would you trust more?