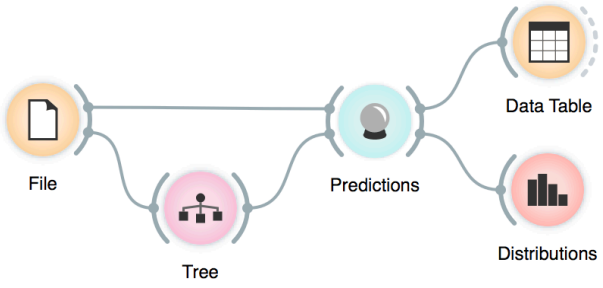
Lesson 18: Classification Accuracy

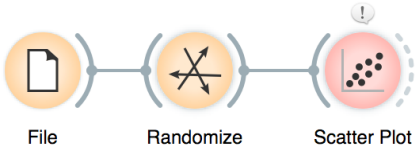
Now that we know a few learning algorithms, the next question is what is the quality of their predictions. For beginning, we need to define what we mean by quality. In classification, the simplest measure of quality is classification accuracy expressed as the proportion of data instances for which the classifier correctly guessed the value of the class. Let's see if we can estimate, or at least get a feeling for, classification accuracy with the widgets we already know.

Measuring of accuracy is such an important concept that it would require its widget. But wait a while, there's educational value in reusing the widgets we already know.



Let us try this schema with the brown-selected data set. The Predictions widget outputs a data table augmented with a column that includes predictions. In the Data Table widget, we can sort the data by any of these two columns, and manually select data instances where the values of these two features are different (this would not work on big data). Roughly, visually estimating the accuracy of predictions is straightforward in the Distribution widget, if we set the features in view appropriately.

This lesson has a strange title and it is not obvious why it was chosen. Maybe you, the reader, should tell us what does this lesson have to do with cheating.

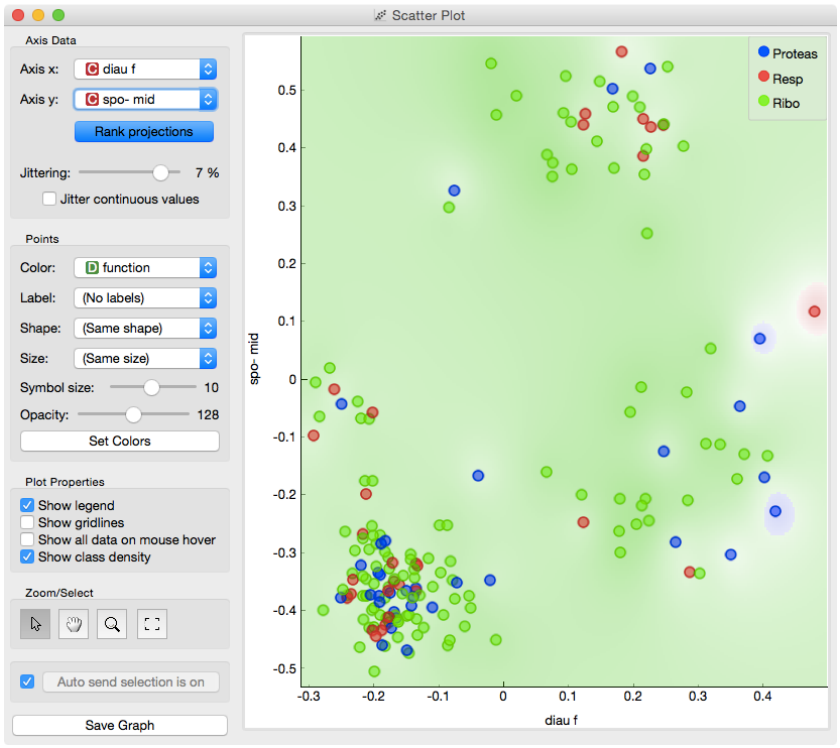


Randomize widget shuffles the column in the data table. It can shuffle the class column, columns with data features or columns with meta information. Shuffling the class column breaks any relation between features and the class, keeping the data points (genes profiles) intact.

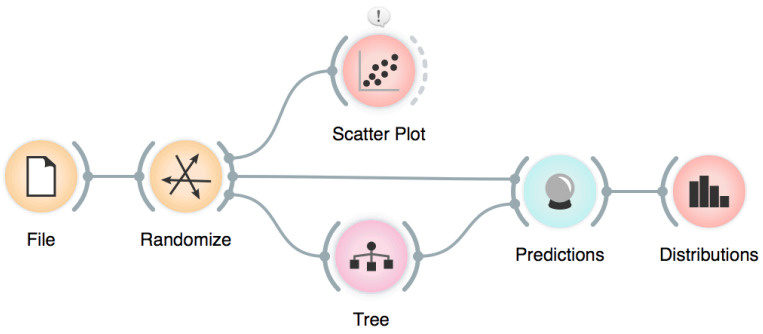
Why is the background in this scatter plot so green, and only green? Why have the other colors disappeared after the class randomization?

Lesson 19: How to Cheat

At this stage, the classification tree looks very good. There's only one data point where it makes a mistake. Can we mess up the data set so bad that the trees will ultimately fail? Like, remove any existing correlation between gene expression profiles and class? We can! There's the Randomize widget that can shuffle the class column. Check out the chaos it creates in the Scatter Plot visualization where there were nice clusters before randomization!

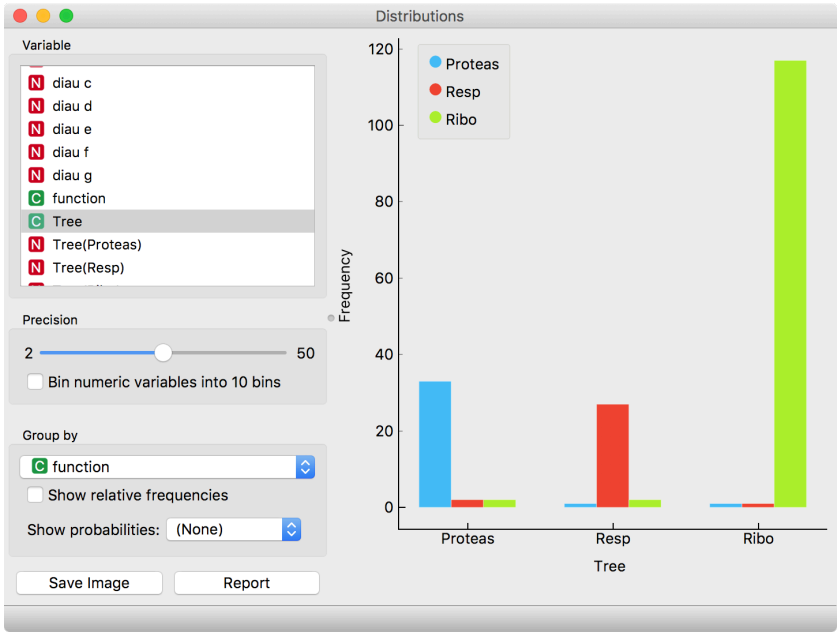


Fine. There can be no classifier that can model this mess, right? Let us test this. We will build classification tree and check its performance on the messed-up data set.



And the result? Here is a screenshot of the Distributions:

At this stage, it may be worthwhile checking how do the trees look. Try comparing the tree inferred from original and shuffled data!



Most unusual. Almost no mistakes. How is this possible? On a class-randomized data set?

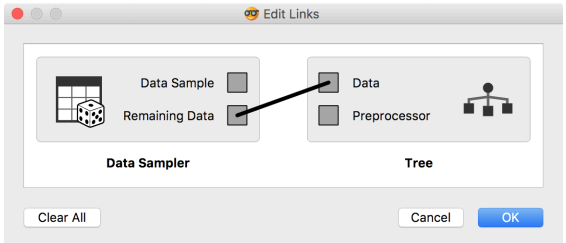
To find the answer to this riddle, open the Tree Viewer and check out the tree. How many nodes does it have? Are there many data instances in the leaf nodes?

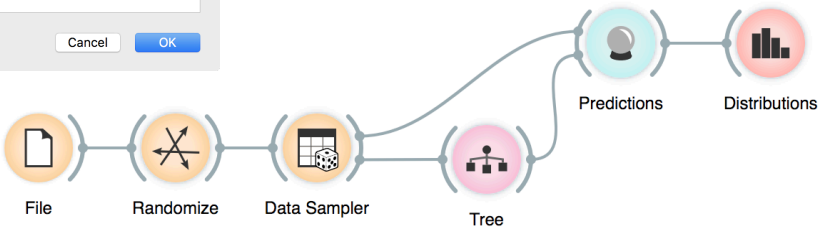
It looks like the tree just memorized every data instance from the data set. No wonder the predictions were right. The tree makes no sense, and it is complex because it simply remembered everything.

This should be a bit of déjà vu. Is not this the same as regression modelling with high degree polynomials?

If a classifier remembers everything from a data set but without discovering any general patterns, it should perform miserably on any new data set, right? Let us check this out. We will split our data set into two sets, training and testing, train the classification tree on the training data set and then estimate its accuracy on the test data set.

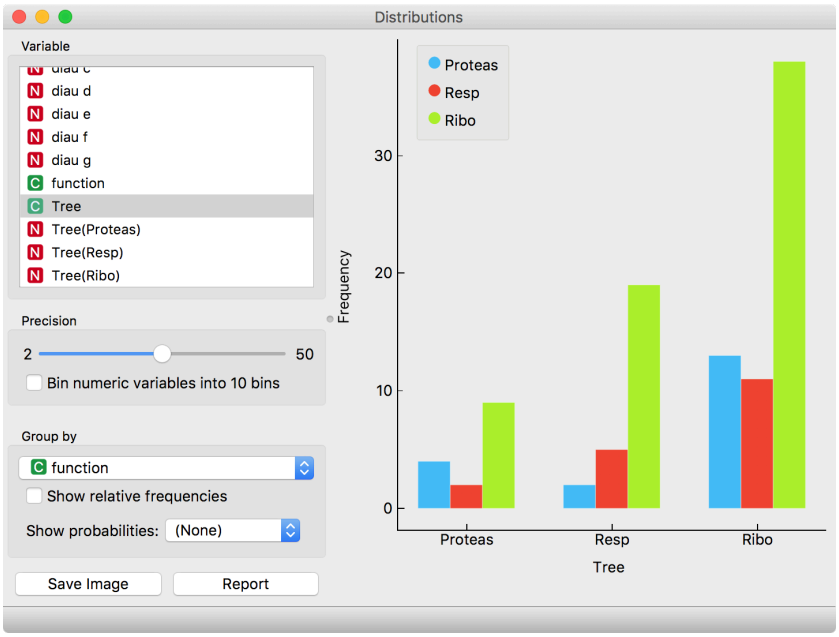
The signals from the Data Sampler widget have not been named in our workflow to save space. The Data Sampler splits the data to a sample and out-of-sample (so called remaining data). The sample was given to the Tree widget, while the remaining data was handed to the Predictions widget. Set the Data Sampler so that the size of these two data sets is about equal.





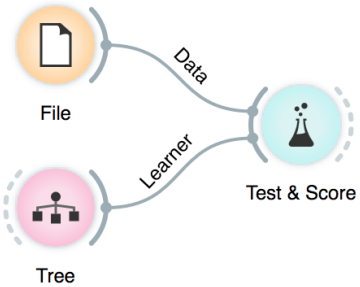
Let's check how the Distributions widget looks after testing the classifier on the test data.

Turns out that for every class value the majority of data instances has been predicted to the ribosomal class (green). Why? Green again (like green from the Scatter Plot of the messed-up data)? Here is a hint: use the Box Plot widget to answer this question.



The first two classes are a complete fail. Predictions for ribosomal genes are a bit better, but still with lots of mistakes. On class-randomized training data, our classifier fails miserably. Finally, this is just as we would expect.

To test the performance (accuracy) of the classification technique, we have just learned that we need to train the classifiers on the training set and then test it on a separate test set. With this test, we can distinguish between those classifiers that just memorize the training data and those that learn a useful model.



Learning is not only remembering. Rather, it is discovering patterns that govern the data and apply to new data as well. To estimate the accuracy of a classifier, we, therefore, need a separate test set. This assessment should not depend on just one division of the input data set to training and test set (here's a place for cheating as well). Instead, we need to repeat the process of estimation several times, each time on a different train/test set and report on the average score.

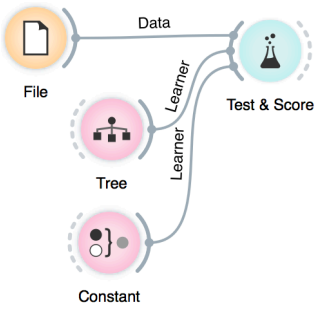
Testing classification models is thus the same as testing regression models, just with a different score. All other techniques we have seen before, such as cross-validation, apply here, too.

Lesson 20: Model Scoring

In multiple choice exams, you are graded according to the number of correct answers. The same goes for classifiers: the more correct predictions they make, the better they are. Nothing could make more sense. Right?

Maybe not. Dr. Smith is a specialist of a type and his diagnosis is correct in 98% of the cases. Would you consider visiting him if you have some symptoms related to his speciality?

Not necessarily. His specialty, in fact, are rare diseases (2 out of 100 of his patients have it) and, being lazy, he always dismisses everybody as healthy. His predictions are worthless — although extremely accurate. Classification accuracy is not an absolute measure, which can be judged out of context. At the very least, it has to be compared with the frequency of the majority class, which is, in case of rare diseases, quite ... major.



For instance, on GEO data set GDS 4182, the classification tree achieves 78% accuracy on cross validation, which may be reasonably good. Let us compare this with the Constant model, which implements Dr. Smith's strategy by always predicting the majority. It gets 83%. Classification trees are not so good after all, are they?

On the other hand, their accuracy on GDS 3713 is 57%, which seems rather good in comparison with the 50% achieved by predicting the majority.

What do other columns represent? Keep reading!

Method	AUC	CA	F1	Precision	Recall
Classification Tree	0.573	0.570	0.585	0.571	0.600
Majority	0.500	0.506	0.672	0.506	1.000

The problem with classification accuracy goes deeper, though.

Classifiers usually make predictions based on probabilities they compute. If a data instance belongs to class A with a probability of 80% and to B with a probability of 20%, it is classified as A. This makes sense, right?

Maybe not, again. Say you fall down the stairs and your leg hurts. You open Orange, enter some data into your favorite model and

compute a 20% of having your leg broken. So you assume your leg is not broken and you take an aspirin. Or perhaps not?

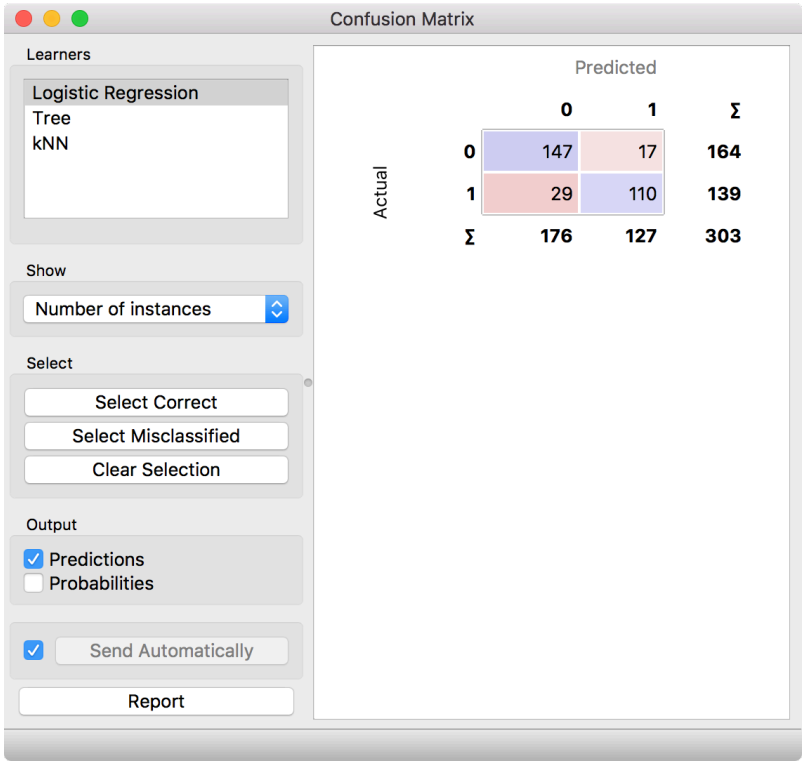
What if the chance of a broken leg was just 10%? 5%? 0.1%?

Say we decide that any leg with a 1% chance of being broken will be classified as broken. What will this do to our classification threshold? It is going to decrease badly — but we apparently do not care. What do we do care about then? What kind of "accuracy" is important?

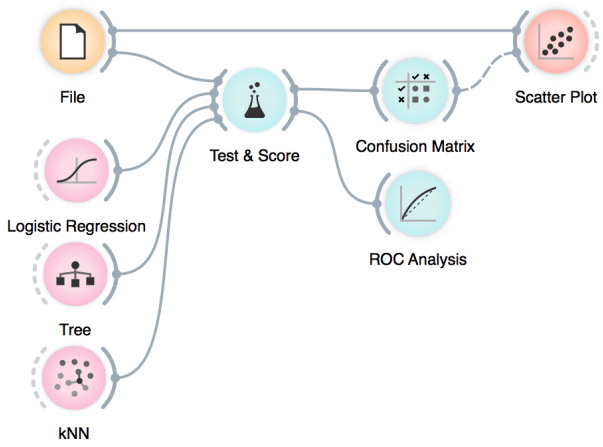
Not all mistakes are equal. We can summarize them in the Confusion Matrix. Here is one for logistic regression on the heart disease data.

These numbers in the Confusion Matrix have names. An instance can be classified as positive or negative; imagine this as being positive or negative when being tested for some medical condition. This classification can be true or false. So there are four options, *true positive* (TP), *false positive* (FP), *true negative* (TN) and *false negative* (FN).

Identify them in the table!



Use the output from Confusion Matrix as a subset for Scatter plot to explore the data instances that were misclassified



Logistic regression correctly classifies 147 healthy persons and 110 of the sick, the numbers on the diagonal. Classification accuracy is then 257 out of 303, which is 85%.

17 healthy people were unnecessarily scared. The opposite error is worse: the heart problems of 29 persons went undetected. We need to distinguish between these two kinds of mistakes.

We are interested in the probability that a person who has some problem will be correctly diagnosed. There were 139 such cases, and 110 were discovered. The proportion is 110 / 139 = 0.79. This measure is called *sensitivity* or *recall* or *true positive rate (TPR)*.

If you were interested only in sensitivity, though, here's Dr. Smith's associate partner — wanting to be on the safe side, she considers everybody ill, so she has a perfect sensitivity of 1.0.

To counterbalance the sensitivity, we compute the opposite: what is the proportion of correctly classified *negative* instances? 147 out of 164, that is, 90%. This is called *specificity* or *true negative rate*.

If you are interested in a complete list, see the Wikipedia page on Receiver operating characteristic, https://en.wikipedia.org/wiki/Receiver_operating_characteristic

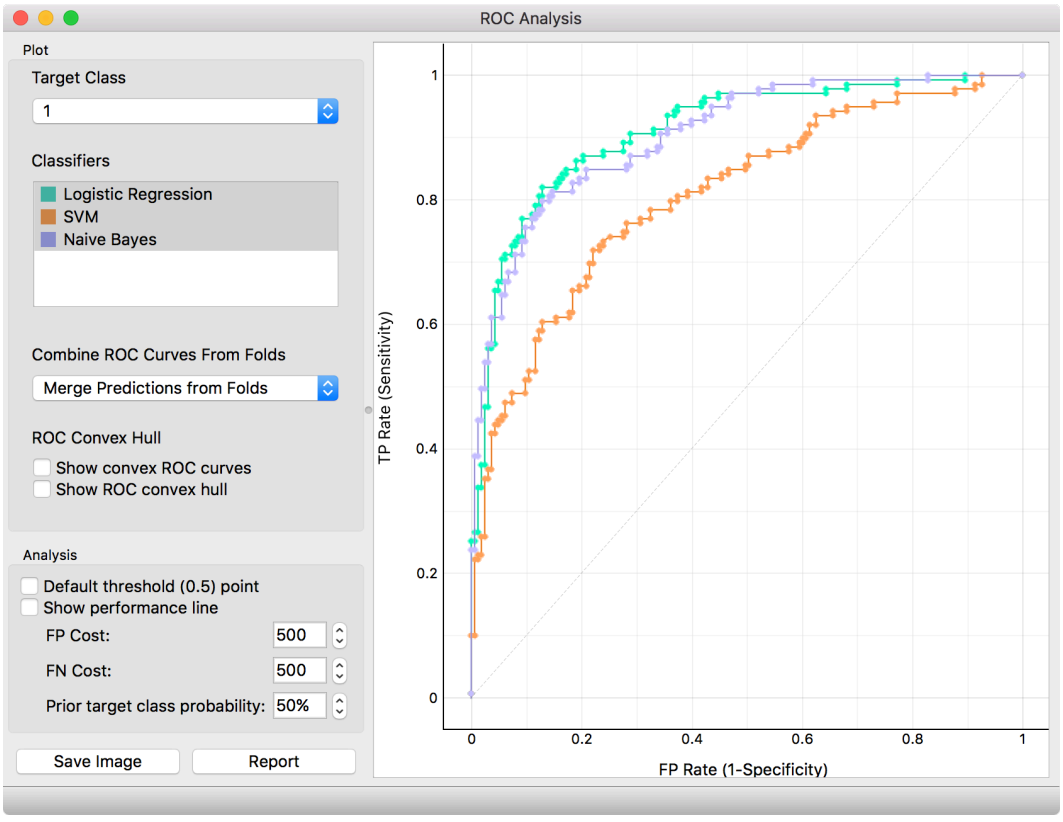
So, if you're classified as OK, you have a 90% chance of actually being OK? No, it's the other way around: 90% is the chance of being classified as OK, if you are OK. (Think about it, it's not as complicated as it sounds). If you're interested in your chance of being OK if the classifier tells you so, you look for the *negative predictive value*. Then there's also *precision*, the probability of being positive if you're classified as such. And the *fall-out* and *negative likelihood ratio* and ... a whole list of other indistinguishable fancy names, each useful for some purpose.

Lesson 21: Choosing the Decision Threshold

The common property of scores from the previous lesson is that they depend on the threshold we choose for classifying an instance as positive. By adjusting it, we can balance between them and find, say, the threshold that gives us the required sensitivity at an acceptable specificity. We can even assign costs (monetary or not) to different kinds of mistakes and find the threshold with the minimal expected cost.

A useful tool for this is the Receiver-Operating Characteristic curve. Don't mind the meaning of the name, just call it the ROC curve.

Here are the curves for logistic regression, SVM with linear kernels and naive Bayesian classifier on the same ROC plot.



The curves show how the sensitivity (y-axis) and specificity (x-axis, but from right to left) change with different thresholds.

Sounds complicated? If it helps: perhaps you remember the term *parametric curve* from some of your math classes. ROC is a parametric curve where x and y (the sensitivity and 1 - specificity) are a function of the same parameter, the decision threshold.

There exists, for instance, a threshold for logistic regression (the green curve) that gives us 0.65 sensitivity at 0.95 specificity (the curve shows 1 - specificity). Or 0.9 sensitivity with a specificity of 0.8. Or a sensitivity of (almost) 1 with a specificity of somewhere around 0.3.

The optimal point would be at top left. The diagonal represents the behavior of a random guessing classifier.

Which of the three classifiers is the best now? It depends on the specificity and sensitivity we want; at some points we prefer logistic regression and at some points the naive bayesian classifier. SVM doesn't cut it, anywhere.

There is a popular score derived from the ROC curve, called Area under curve, AUC. It measures, well, the area under the curve. This curve. If the curve goes straight up and then right, the area is 1; such an optimal AUC is not reached in practice. If the classifier guesses at random, the curve follows the diagonal and AUC is 0.5. Anything below that is equivalent to guessing + bad luck.

AUC has a kind of absolute scale. As a rule of a thumb: 0.6 is bad, 0.7 is bearable, 0.8 is publishable and 0.9 is suspicious.

ROC curves and AUC are fascinating tools. To learn more, read T. Fawcett: ROC Graphs: Notes and Practical Considerations for Researchers

AUC also has a nice probabilistic interpretation. Say that we are given two data instances and we are told that one is positive and the other is negative. We use the classifier to estimate the probabilities of being positive for each instance, and decide that the one with the highest probability is positive. It turns out that the probability that such a decision is correct equals the AUC of this classifier. Hence, AUC measures how well the classifier discriminates between the positive and negative instances.

From another perspective: if we use a classifier to rank data instances, then AUC of 1 signifies a perfect ranking, an AUC of 0.5 a random ranking and an AUC of 0 a perfect reversed ranking.