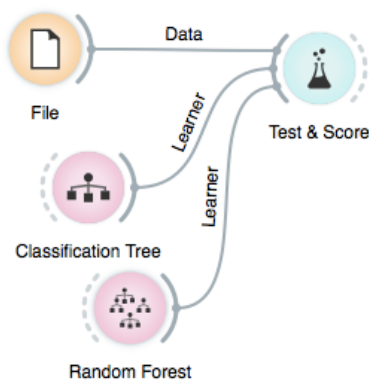


## Lesson 11: A Few More Classifiers

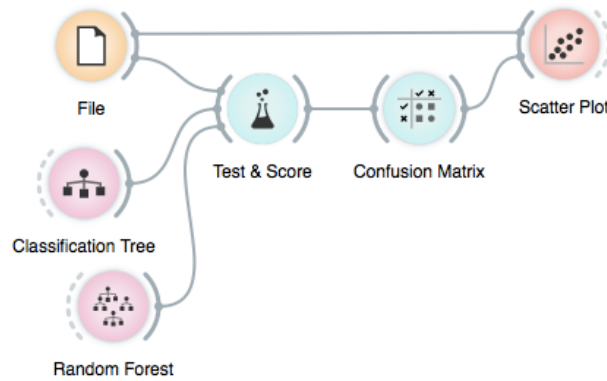
We have ended the previous lesson with cross-validation and classification trees. There are many other, much more accurate classifiers. A particularly interesting one is Random Forest, which averages across predictions of hundreds of classification trees. It uses two tricks to construct different classification trees. First, it infers each tree from a sample of the training data set (with replacement). Second, instead of choosing the most informative feature for each split, it randomly selects from a subset of most informative features. In this way, it randomizes the tree inference process. Think of each tree shedding light on the data from a different perspective. Just like in the wisdom of the crowd, an ensemble of trees (called a forest) usually performs better than a single tree.



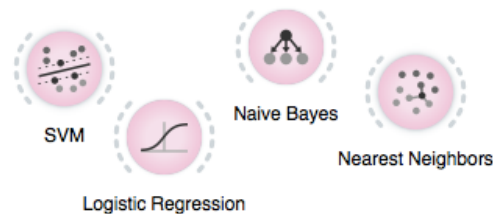
Let us see if this is really so. We give two learners to the Test Learners widget and check if cross-validated classification accuracy is indeed higher for random forest. Choose different classification data sets for this comparison, starting with those we already know (hearth disease, iris, brown selected).

It may be interesting to compare where different classification methods make mistakes. We can use Confusion Matrix for this purpose, and then pass the signal from this widget to the Scatter Plot.

What kind of object is sent from the Test & Score widget to the Confusion Matrix widget? So far, we have used widgets that send data, or even learners. But what could the Test & Score widget communicate to other widgets?

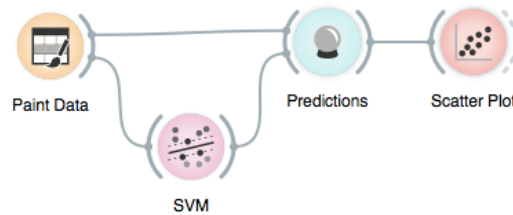


There are other classifiers we can try. We will briefly mention a few more, but instead of diving into what they do (we could spend a semester on this!), we'll pass on to other important topics in data mining. At this point, just add them to the workflow above and see how they perform.

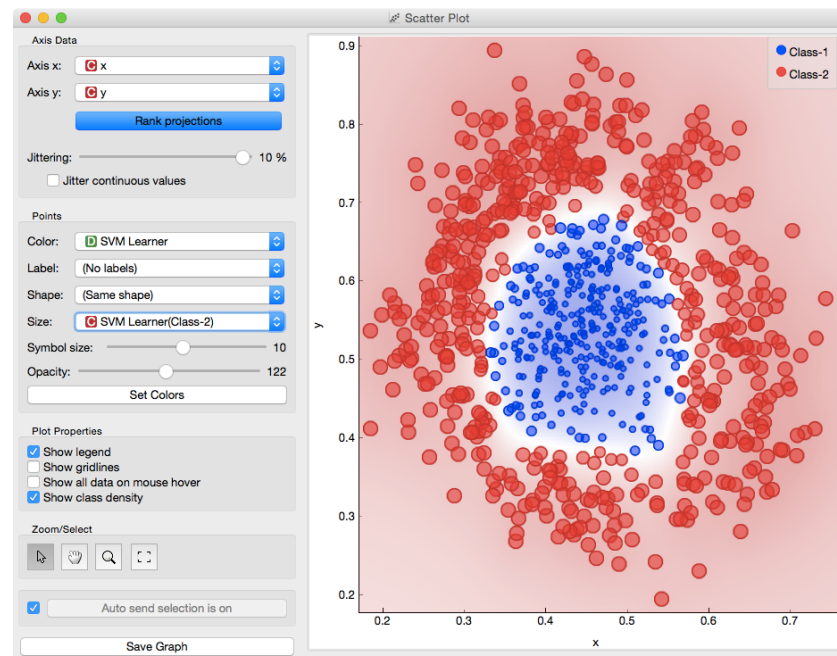
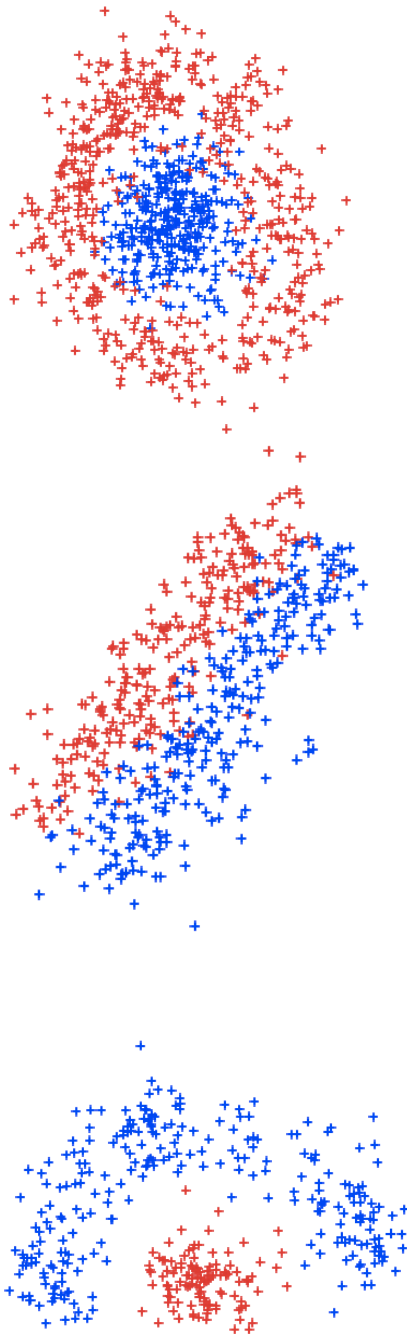


It would be nice if we could, at least on the intuitive level, understand the differences between all these methods and their variants (every method has some parameters). Remember, the classification tree finds hyperplanes orthogonal to the axis; those hyperplanes split the data space to regions with different class probabilities. The tree's decision boundaries are flat. Nearest neighbors classifies the data instance according to the few neighboring data instances in the training set. Decision boundaries with this approach could be very complex. Logistic regression infers just one hyperplane (decision boundary) in an arbitrary direction. This is similar to support vector machines with linear kernel, but then again, the kernels with SVM can be changed, resulting in more complex decision boundaries.

Ok, we have to admit: the above paragraph reads almost like gibberish. We would need a workflow where we could actually see the decision boundaries. And perhaps invent the data sets to test the classifiers. Best in 2D. Maybe, for a start, we could just paint the data. Time to stop writing this long passage of text, end the suspense, and construct a workflow that does this all.



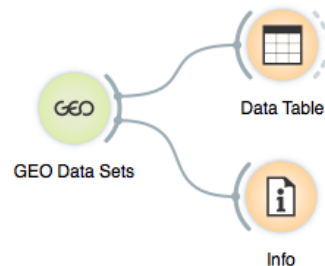
Be creative when painting the data! Also, instead of SVM, use different classifiers. Also, try changing the parameters of the classifiers. Like, limit the depth of the decision tree to 2, or 3, 4. Or switch from SVM with linear kernel to the radial basis function. Appropriately set up the scatter plot to observe the changes.



## Lesson 12: GEO Data Sets

A slight detour here. Let us get some fresh data. The bioinformatics add-on provides access to NCBI's Gene Expression Omnibus (GEO), more precisely, to GEO's collection of data set, or GDS. In the GEO Data Sets widget we can browse through the collection and download a selected data set from NCBI. Let's use the following workflow and inspect a few data sets.

In the GEO Data Sets widget, try changing the setting of what data will be represented in rows. Check the output in the Data Table and Info widgets. Which setting would be appropriate for creating a data set for classification?



The data sets that have already been downloaded on your computer are marked with a bullet in the first column of the table.

ID	Title	Organism	Samples	Features	Genes	Subsets	PubMedID
• CDS360	Breast cancer and docetaxel...	Homo sapiens	24	12625	9459	2	
• CDS3713	Smoking effect on B lympho...	Homo sapiens	79	22283	14047	2	20217071
• CDS1210	Gastric cancer	Homo sapiens	30	7129	6172	2	11782383
CDS2526	c-MYC depletion effect on c...	Homo sapiens	18	54675	31396	8	17159920
CDS2524	Effect of gonadal steroids o...	Mus musculus	48	22690	13916	10	16714546
CDS2525	Foxp3 ablation effect on m...	Mus musculus	4	45101	26722	2	17220892
CDS2522	Pyocyanin treatment: dose r...	Saccharomyc...	6	9335	8714	5	17185230

**Description**

Analysis of peripheral circulating B cells from smoking and non-smoking healthy US white females. B cells are directly associated with the onset and development of many smoking-induced diseases. Results provide insight into the molecular basis of B cell involvement in smoking-related pathogenesis.

**Sample Annotations**

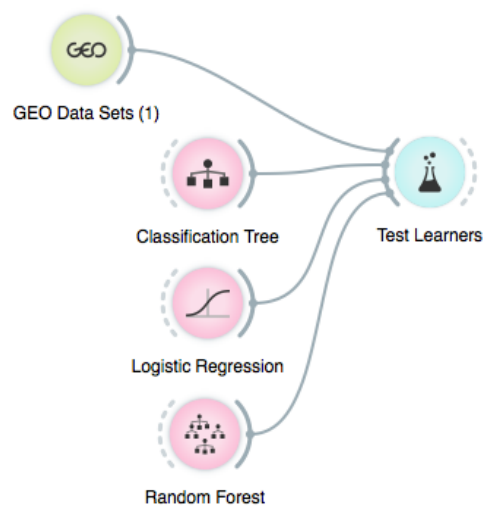
Type (Sample annotations)	Sample count
stress	
control	40
cigarette smoke	39

Some data sets from GEO are very large and may take quite a while to download. Try experimenting with smaller ones first.

## Lesson 13: GEO Data Sets and Classification

From the GEO Data Sets widget, select the data on breast cancer (GDS360) with 14 treatment resistant and 10 treatment sensitive tumors. Can we predict the treatment sensitivity from gene expression profiles? To do this, we need to set the Output of GEO Data Sets widgets to Samples (default is set to Genes or spots).

The Random Forest classifier often achieves good accuracy on gene expression data. Try changing the number of classification trees in the forest. How does the accuracy change? Does random forest beat a single classification tree? How does logistic regression compare with the other two methods?



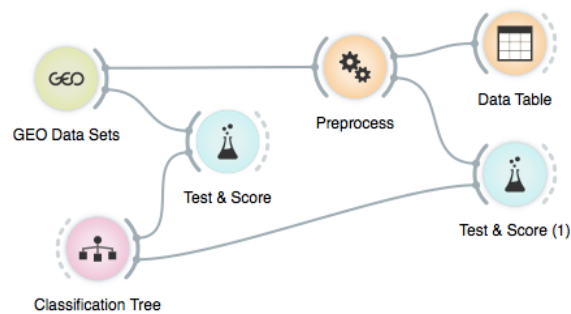
If you are concerned only with accuracy and ignore any extra knowledge on the data domain, a recent report by Fernández-Delgado *et al.* (JMRL, 2014) says that Random Forests are your best bet. However, each of the machine learning methods has its own good and bad points. Some can be nicely visualized, some have higher accuracy, and others, like deep neural nets, perform well on huge data sets. Knowing which one to use when is close to an art, but luckily, even if not tuned to perfection, most are useful.

We will test the accuracy of three learners: classification tree, logistic regression, and random forest. We recommend starting with smaller data sets as some of the learning algorithms require a lot of time.

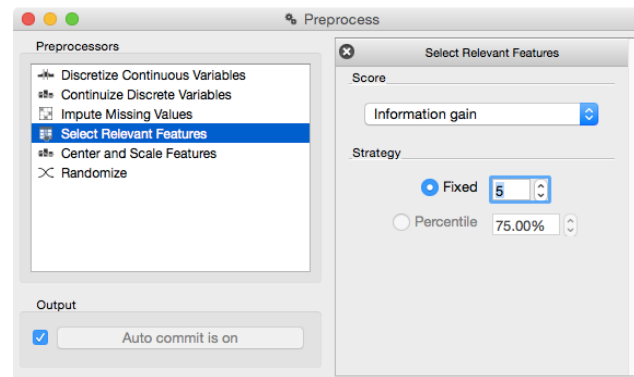
## Lesson 14: Overfitting on GDS

Consider GDS data sets where we have samples in rows and genes expressions in columns. These data sets are usually fat: there are many more genes than samples. Fat data sets are almost typical for systems biology. When samples are classified, and our task is classification, many features (genes) will be irrelevant and most often only a few will be highly correlated with class. So why not simply first select a set of most informative features, and then do the whole analysis? At least cross-validation will then work much faster, as the model inference algorithms will deal with much smaller data tables. Cool. What a nice trick! Let's try it out in the following workflow.

GEO data sets can be very large. To avoid long computation times, start this lesson by setting the Test & Score widget to 5-fold cross validation. Use a smaller GEO data set, say GDS360.



The workflow above uses the data preprocessing widget, which we have configured to select 5 most informative features.



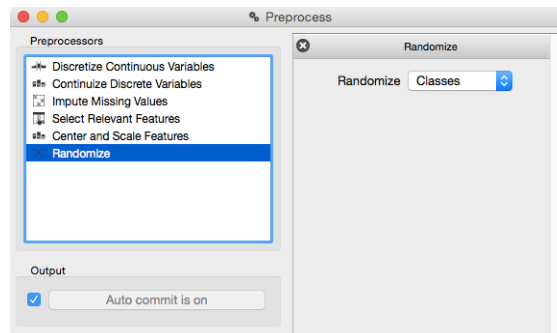
Observe the classification accuracy obtained on the original data set, and on the data set with five best selected features. What is happening? Why?

## Lesson 15: Overfitting to the Extreme

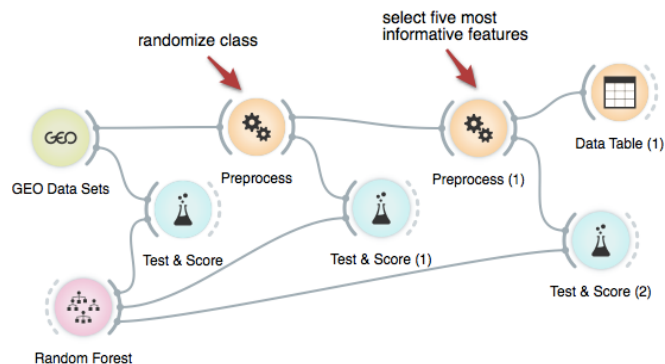
We can push the example from our previous lesson to the extreme. We will randomize the classification data. That is, we will take the column with the class values and randomly permute it. We will use the Preprocess widget to do this.

Later, we will do classification on this data set. We expect really low classification accuracy on this data set. Then, we will select 5 most relevant features. Even though we have randomly permuted the classes, there have to be some features that are weakly correlated with the class. Simply because we have tens of thousands of features, and we have only a few samples. There are enough features that some of them correlate with class simply by chance. Finally, we will score a random forest on a randomized data set with selected features.

The Preprocess widget takes an input data set and applies requested transformation. You can configure it for a pipeline of stacked transformations, from top to bottom.



Compare the scores reported by cross-validation on different data sets in this pipeline. Why is the accuracy in the final one rather high? Would adding more “most informative features” improve or degrade the cross-validated performance on a randomized data set?



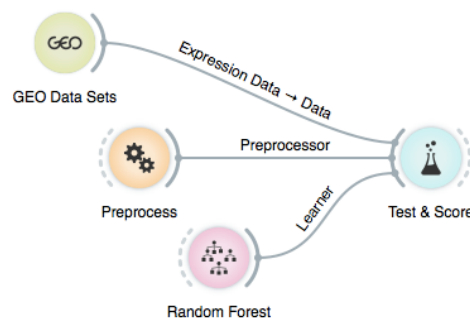
## Lesson 16: How to avoid Overfitting

The writing on the right looks straightforward. But actually one needs to be extremely careful not to succumb to overfitting when reporting results of cross-validation tests. The literature on systems biology is polluted with reporting on overly optimistic results, and high impact factors provide no guarantee that studies were carried out correctly (in fact, due to a lack of reviewers from the field of machine learning, mistakes likely stay overlooked).

Simon et al. (2003) provides a great read on this topic. He found that many of the early papers in gene expression analysis reported high accuracy simply due to overfitting.

To put it simply: never, in any way, transform the data prior to cross-validation. Any transformation should happen within cross-validation, first on the training set, and then, if required, on a test set. In a relaxed form: it's ok to transform the data, but the transformation should be done independently on the train and the test set and the transformation on the test set should in no way use the information about the class value. Data imputation could be an example of such operation, but again it should be carried out separately for the train and test set and should not consider classes.

But how do we then correctly apply preprocessing in Orange? The idea of reducing the number of features prior to inferring a predictive model may be still appealing, now that we know we can use it on training data sets (leaving the test set alone). Following are two workflows that do this correctly.



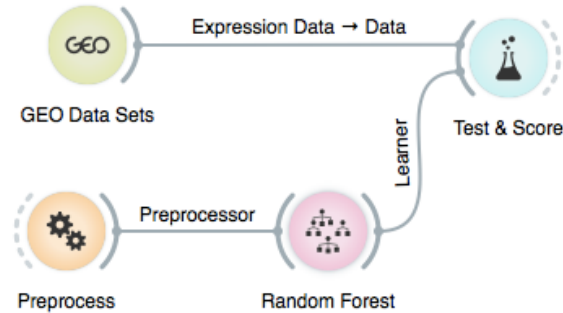
In this first workflow, we gave the Test & Score widget a preprocessor (feature selection was used in this example). The Test & Score widget uses it correctly only on the training sets. This type of workflow is preferred if we would like to test the effect of preprocessing on a number of different learning algorithms.



The Preprocess widget does not necessarily require a data set on its input. An alternative use of this widget is to output a method for data preprocessing, which we can then pass to either a learning method or to a widget for cross validation.

This is not the first time we have used a widget that instead of a data passes forward a computation method. All the learners, like Random Forest, do so. A learner could get data on its input and pass a classifier to its output, or simply pass an instance of itself, that is, pass a learning algorithm to whichever widget could use it. For instance, to the Test & Score widget.

Alternatively, we can include a preprocessor in a learning method? The preprocessor is now called on the training data set just before this learner performs inference of the predictive model.



Can you extend this workflow to such an extent that you can test both a learner with preprocessing by feature subset selection and the same learner without this preprocessing? How does the number of selected features affect the cross-validated accuracies? Does the success of this particular combination of machine learning technique depend on the input data set? Does it work better for some machine learning algorithms? Try its performance on k-nearest neighbors learner (warning: use small data sets, this classifier could be very slow).

Somehow, in a shy way, we have also introduced a technique for feature selection, and pointed to its possible utility for classification problems. Feature subset selection, or FSS in short, was and still is, to some extent, an important topic in machine learning. Modern classification algorithms, though, perform it implicitly, and can deal with a large number of features without the help of external procedures for their advanced selection. Random forest is one such technique. Yet, to see the positive effect of FSS, say with classification trees, try the workflow below.

