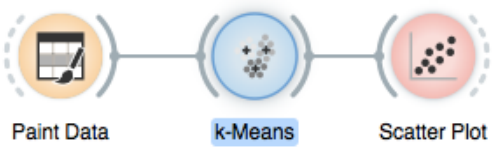
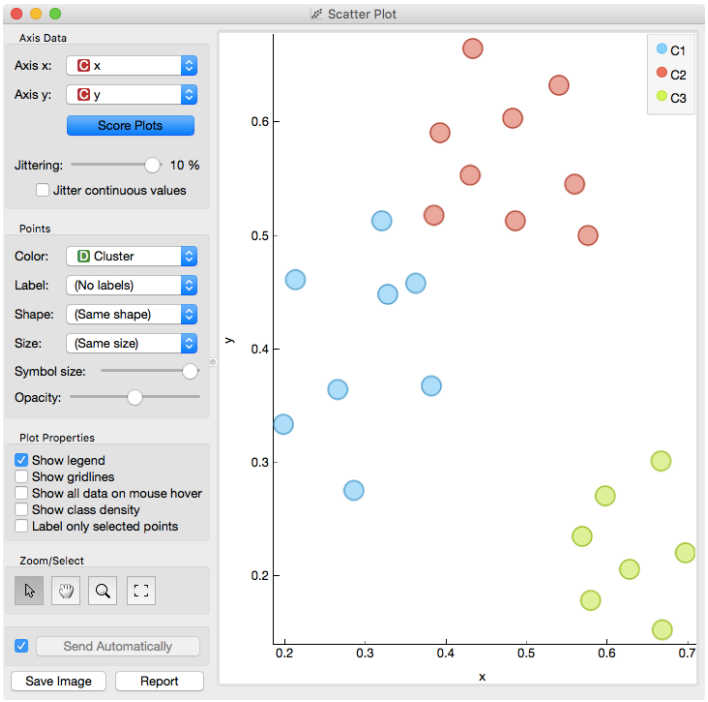
Lesson 26: Silhouettes

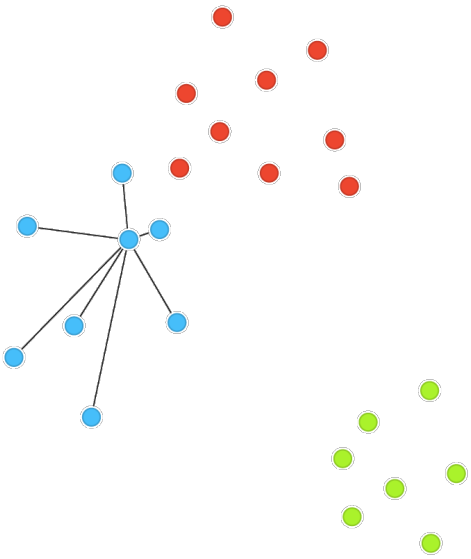
Don't get confused: we paint data and/or visualize it with Scatter plots, which show only two features. This is just for an illustration! Most data sets contain many features and methods like k-Means clustering take into account all features, not just two.



Consider a two-feature data set which we have painted in the Paint Data widget. We send it to the k-means clustering, tell it to find three clusters, and display the clustering in the scatterplot.

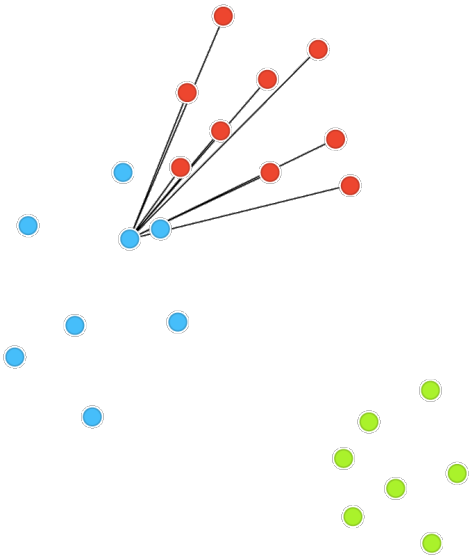


The data points in the green cluster are well separated from those in the other two. Not so for the blue and red points, where several points are on the border between the clusters. We would like to quantify the degree of how well a data point belongs to the cluster to which it is assigned.



We will invent a scoring measure for this and we will call it a *silhouette* (because this is how it's called). Our goal: a silhouette of 1 (one) will mean that the data instance is well rooted in the cluster, while the score of 0 (zero) will be assigned to data instances on the border between two clusters.

For a given data point (say the blue point in the image on the left), we can measure the distance to all the other points in its cluster and compute the average. Let us denote this average distance with A. The smaller A, the better.



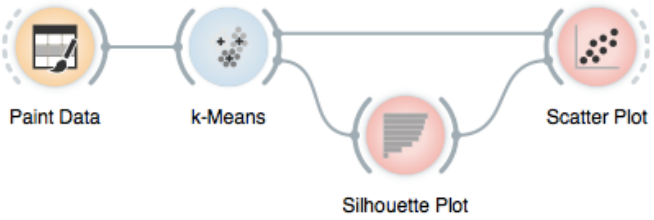
C3 is the green cluster, and all its points have large silhouettes. Not so for the other two.

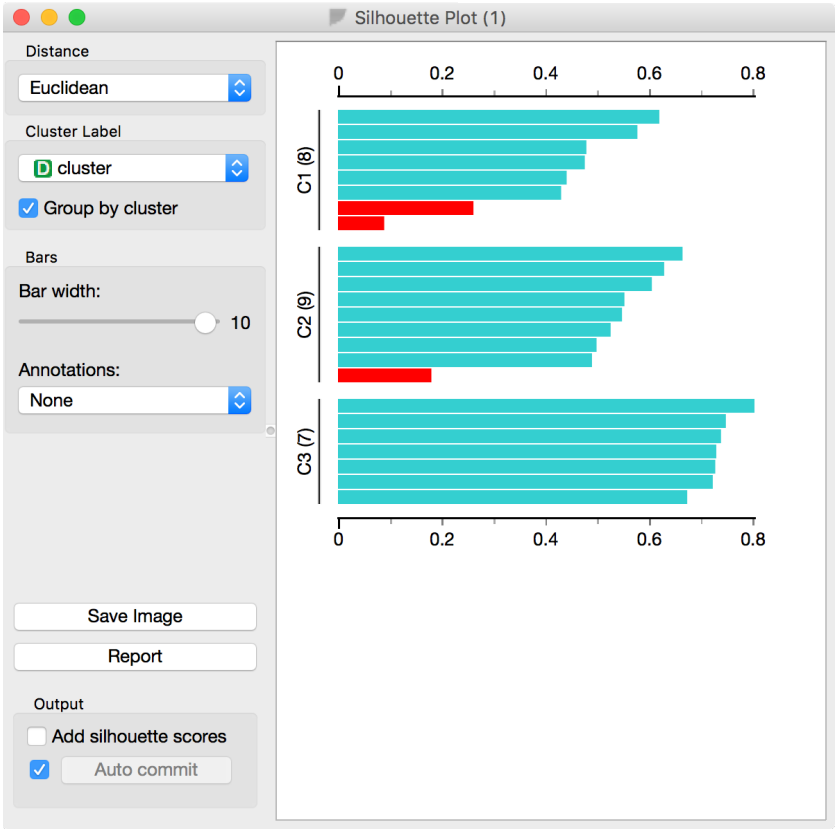
Below we selected three data instances with the worst silhouette scores. Can you guess where the lie in the scatter plot?

On the other hand, we would like a data point to be far away from the points in the closest neighboring cluster. The closest cluster to our blue data point is the red cluster. We can measure the distances between the blue data point and all the points in the red cluster, and again compute the average. Let us denote this average distance as B. The larger B, the better.

The point is well rooted within its own cluster if the distance to the points from the neighboring cluster (B) is much larger than the distance to the points from its own cluster (A), hence we compute $B-A$. We normalize it by dividing it with the larger of these two numbers, $S = (B - A) / \max\{A, B\}$. Voilá, S is our silhouette score.

Orange has a Silhouette Plot widget that displays the values of the silhouette score for each data instance. We can also choose a particular data instance in the silhouette plot and check out its position in the scatter plot.

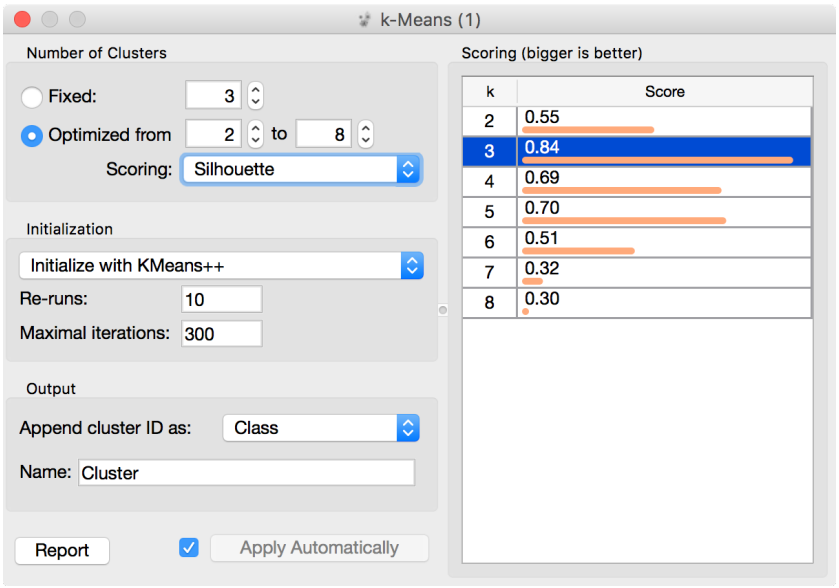




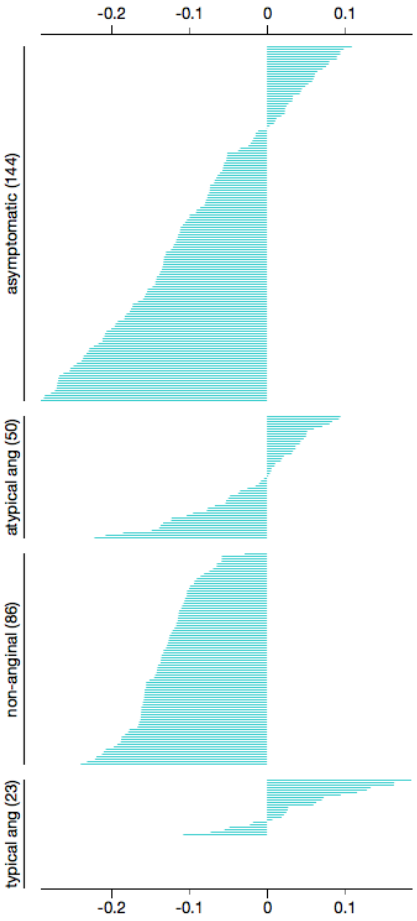
This of course looks great for data sets with two features, where the scatter plot reveals all the information. In higher-dimensional data, the scatter plot shows just two features at a time, so two points that seem close in the scatter plot may be actually far apart when all features - perhaps thousands of gene expressions - are taken into account.

The total quality of clustering - the silhouette of the clustering - is the average silhouette across all points. When the k-Means widget searches for the optimal number

of clusters, it tries different number of clusters and displays the corresponding silhouette scores.



Ah, one more thing: Silhouette Plot can be used on any data, not just on data sets that are the output of clustering. We could use it with the iris data set and figure out which class is well separated from the other two and, conversely, which data instances from one class are similar to those from another.

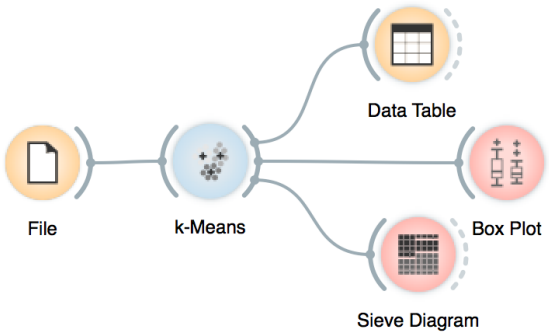


We don't have to group the instances by the class. For instance, the silhouette on the left would suggest that the patients from the heart disease data with typical anginal pain are similar to each other (with respect to the distance/similarity computed from all features), while those with other types of pain, especially non-anginal pain are not clustered together at all.

Lesson 27: Interpretation of Clustering

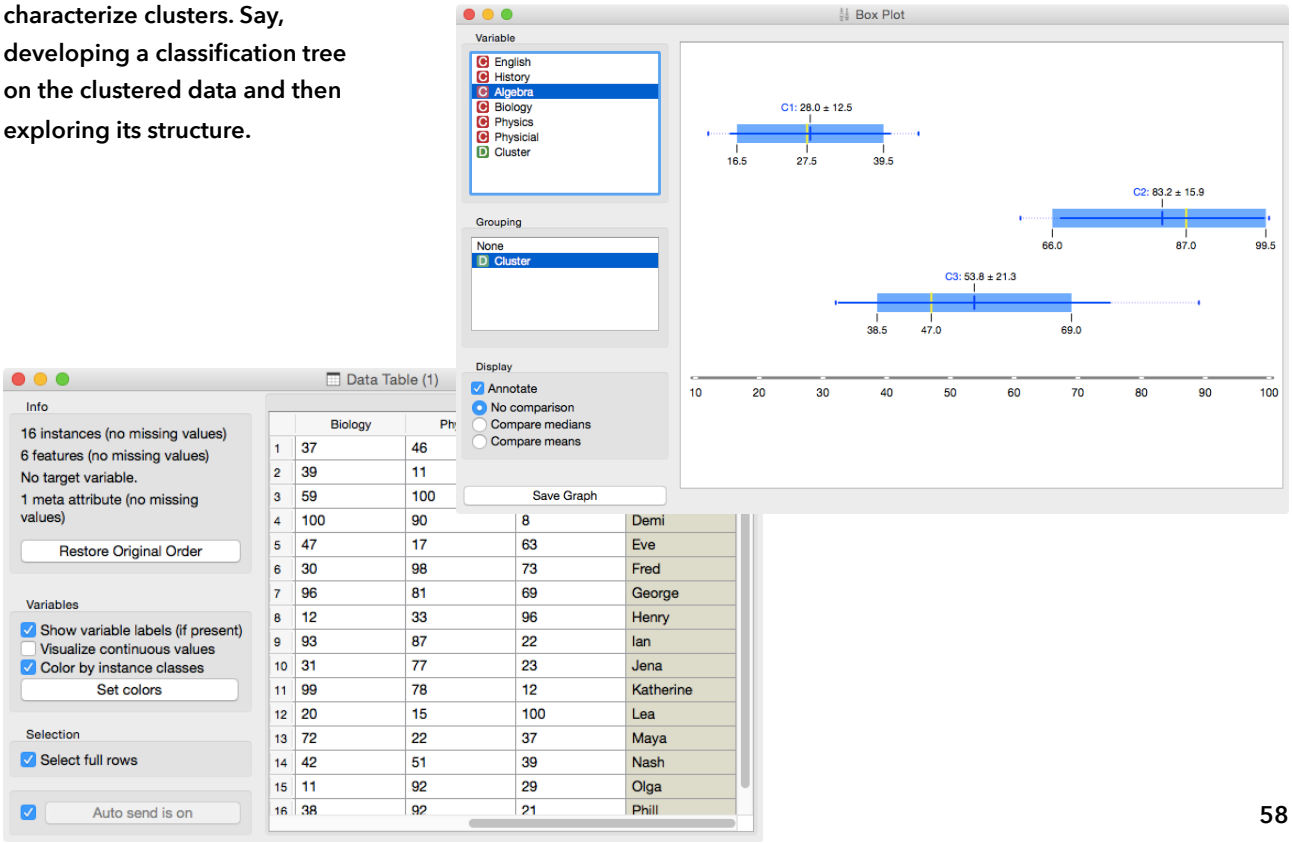
Clustering provides means for identifying groups of data samples. But this is just the beginning. Once we have the clusters, we should figure out what they really are, what are their characteristics. If a data set does not contain too many features, then a simple visualization will get the job done.

The data file used on this page can be downloaded from http://bit.ly/1K9nsi2.



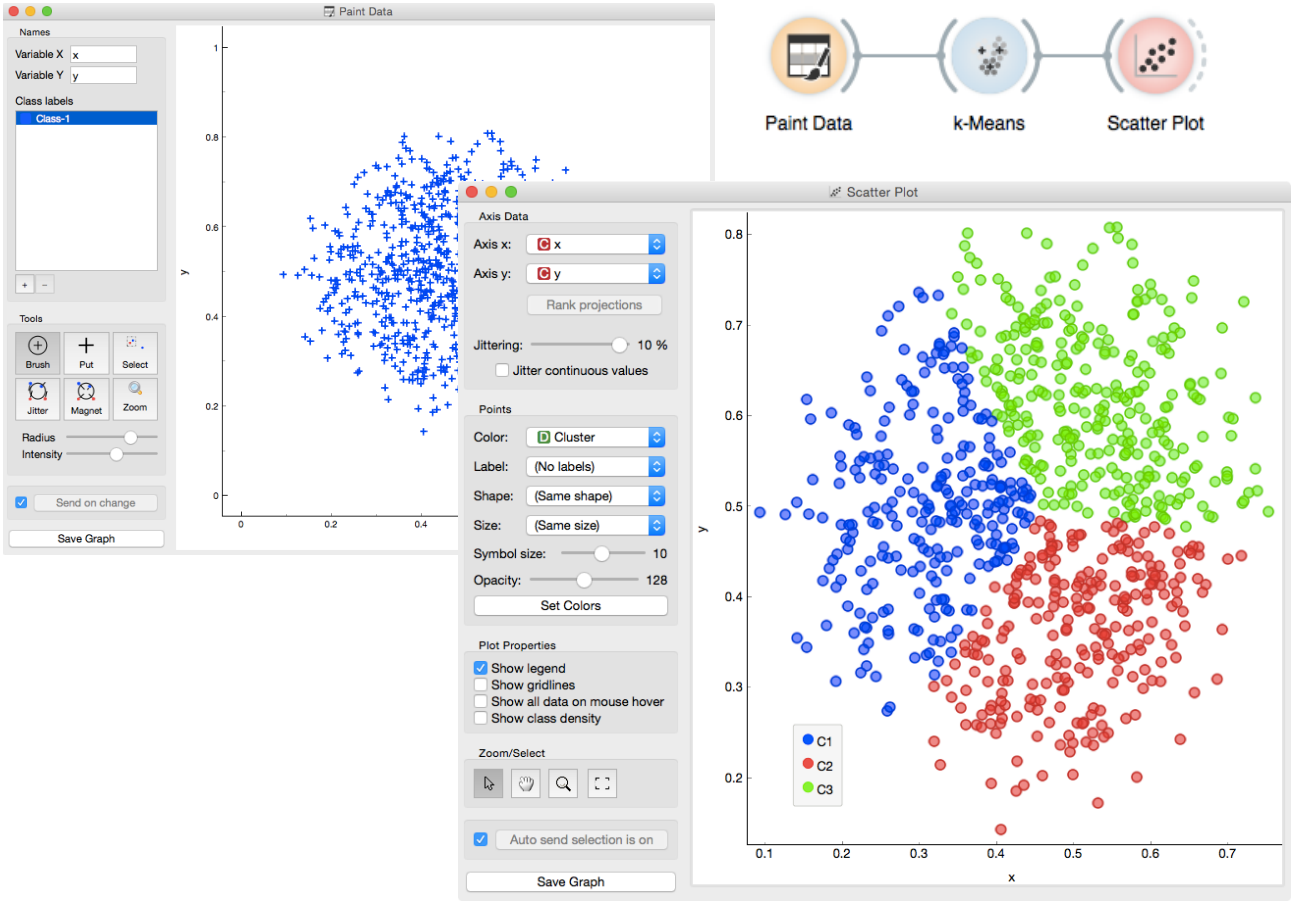
Like, for example, in the data on school grades (kids and different subjects), the box plot reveals that there is a cluster where kids score low in Algebra but high in Physical education.

We could also use supervised data mining techniques to characterize clusters. Say, developing a classification tree on the clustered data and then exploring its structure.



Lesson 28: Finding Clusters When There Are None

We have to be very careful when interpreting clusters. Algorithms like k-means will always find them even when they do not actually exist.



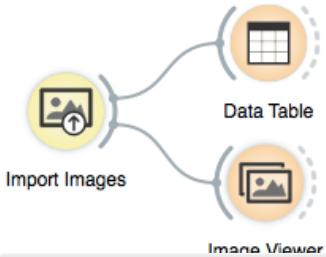
Playing with Paint Data and k-Means can be quite fun. Try painting the data where there are clusters, but k-means does not find them. Or, actually, finds the wrong ones. What kind of clusters are easy to find for k-means? Are these the kind of clusters we would actually find in real data sets?

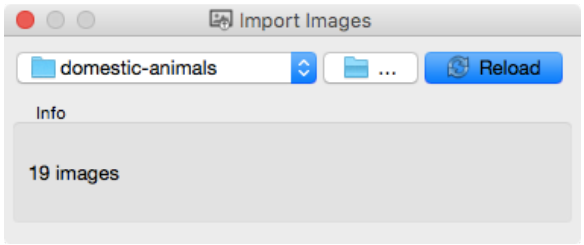
We cannot verify whether the clusters we found are "real". Data mining methods like clustering can serve only as hints that can help forming new hypotheses, which must make biological sense and be verified on new, independent data. We cannot make conclusions based only on "discovering" clusters.

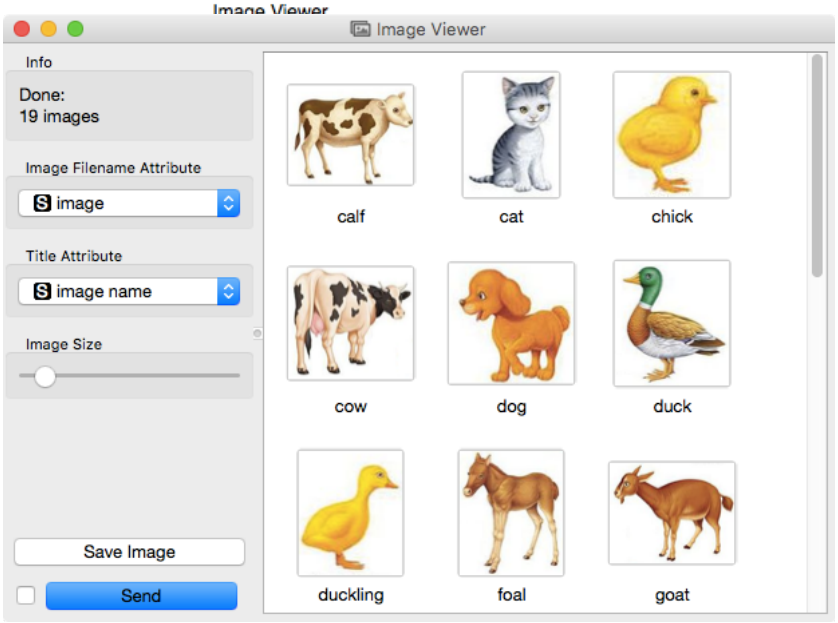
Lesson 29: Images

We will use images on domestic animals in one folder, and cities in the other folder. Load them from http://bit.ly/2cl7uvT and unzip to your computer.

Let us load some images. We can use Import Images widget from Image Analytics add-on. The widget loads the images from the specified folder and all its subfolders. For example, we can load the 19 images from the domestic-animals folder:







We can see these images in Image Viewer. Hm, pack of domestic animals, as expected.

How are these images sent from one widget to another? Let us see what Data Table tells us. Nothing really useful. The table contains only meta features with the image name (from the filename), the location of the image, file size, and the image's width and height. We can not use this for any data mining! We need to convert these images to some numbers, say a vector representation of something. This is called image embedding and is up next.

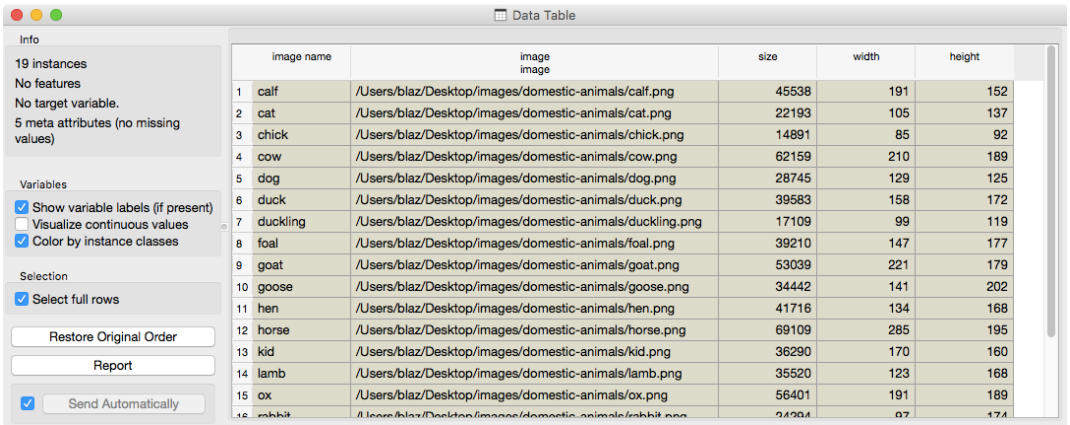
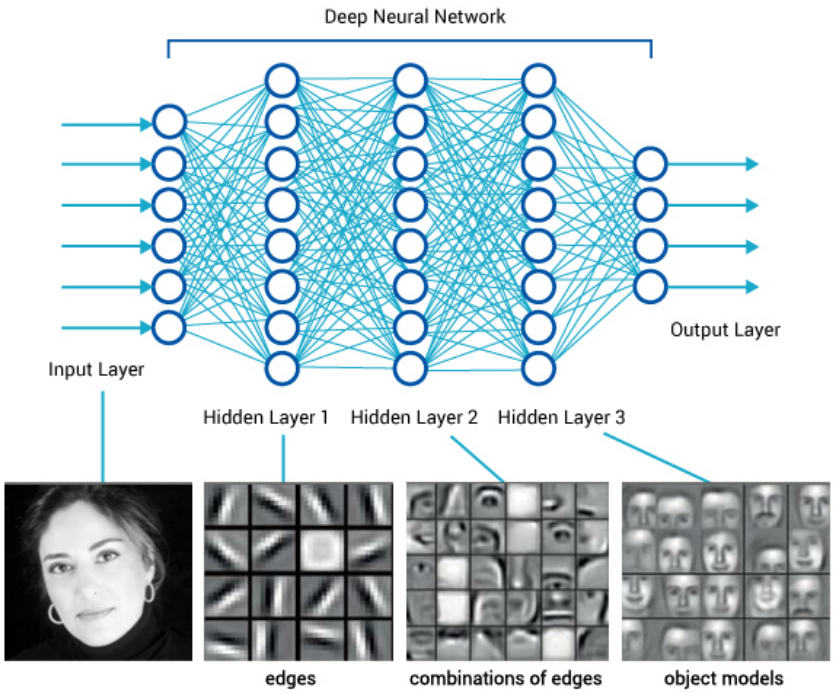


	image name	image image	size	width	height
1	calf	/Users/blaz/Desktop/images/domestic-animals/calf.png	45538	191	152
2	cat	/Users/blaz/Desktop/images/domestic-animals/cat.png	22193	105	137
3	chick	/Users/blaz/Desktop/images/domestic-animals/chick.png	14891	85	92
4	cow	/Users/blaz/Desktop/images/domestic-animals/cow.png	62159	210	189
5	dog	/Users/blaz/Desktop/images/domestic-animals/dog.png	28745	129	125
6	duck	/Users/blaz/Desktop/images/domestic-animals/duck.png	39583	158	172
7	duckling	/Users/blaz/Desktop/images/domestic-animals/duckling.png	17109	99	119
8	foal	/Users/blaz/Desktop/images/domestic-animals/foal.png	39210	147	177
9	goat	/Users/blaz/Desktop/images/domestic-animals/goat.png	53039	221	179
10	goose	/Users/blaz/Desktop/images/domestic-animals/goose.png	34442	141	202
11	hen	/Users/blaz/Desktop/images/domestic-animals/hen.png	41716	134	168
12	horse	/Users/blaz/Desktop/images/domestic-animals/horse.png	69109	285	195
13	kid	/Users/blaz/Desktop/images/domestic-animals/kid.png	36290	170	160
14	lamb	/Users/blaz/Desktop/images/domestic-animals/lamb.png	35520	123	168
15	ox	/Users/blaz/Desktop/images/domestic-animals/ox.png	56401	191	189

Lesson 30: Image Embedding

Every data set so far came in the matrix (tabular) form: objects (say, tissue samples, students, flowers) were described by row vectors representing a number of features. Not all the data is like this; think about collections of text articles, nucleotide sequences, voice recordings or images. It would be great if we could represent them in the same matrix format we have used so far. We would turn collections of, say, images, into matrices and explore them with the familiar prediction or clustering techniques.

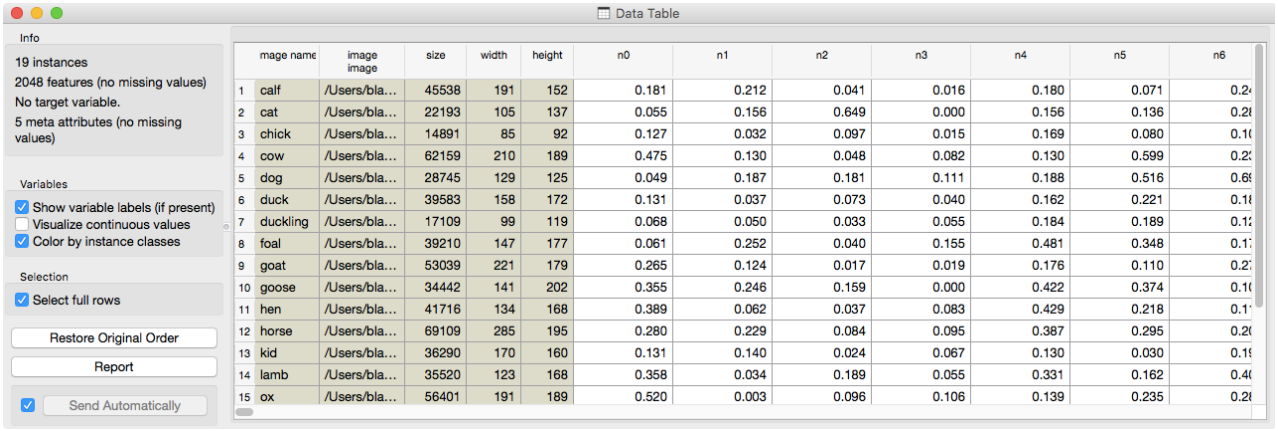
This depiction of deep learning network was borrowed from http://www.amax.com/blog/?



Until very recently, finding useful representation of complex objects such as images was a real pain. Now, technology called deep learning is used to develop models that transform complex objects to vectors of numbers. Consider images. When we, humans, see an image, our neural networks go from pixels, to spots, to patches, and to some higher order representations like squares, triangles, frames, all the way to representation of complex objects. Artificial neural networks used for deep learning emulate these through layers of computational units (essentially, logistic regression models and some other stuff we will ignore here). If we put an image to an input of such a network and collect the outputs from the higher levels, we get vectors containing an abstraction of the image. This is called embedding.

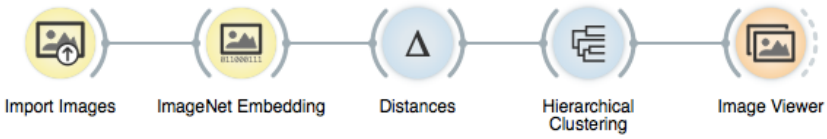
Deep learning requires a lot of data (thousands, possibly millions of data instances) and processing power to prepare the network. We will use one which is already prepared. Even so, embedding takes time, so Orange doesn't do it locally but uses a server invoked through the ImageNet Embedding widget.

Image embedding describes the images with a set of 2048 features appended to the table with meta features of images.



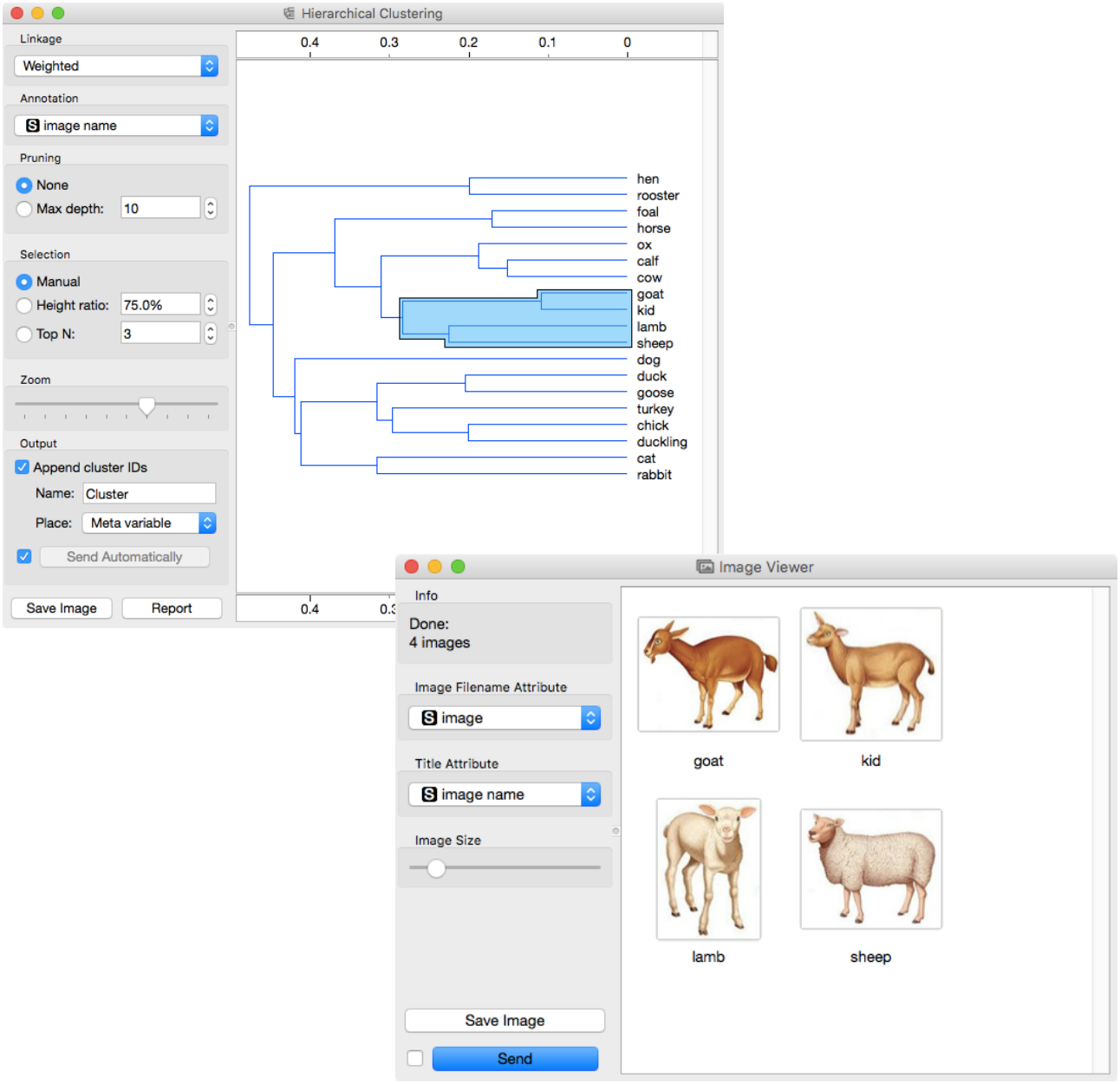
	mage name	image image	size	width	height	n0	n1	n2	n3	n4	n5	n6
1	calf	/Users/bla...	45538	191	152	0.181	0.212	0.041	0.016	0.180	0.071	0.2
2	cat	/Users/bla...	22193	105	137	0.055	0.156	0.649	0.000	0.156	0.136	0.2
3	chick	/Users/bla...	14891	85	92	0.127	0.032	0.097	0.015	0.169	0.080	0.1
4	cow	/Users/bla...	62159	210	189	0.475	0.130	0.048	0.082	0.130	0.599	0.2
5	dog	/Users/bla...	28745	129	125	0.049	0.187	0.181	0.111	0.188	0.516	0.6
6	duck	/Users/bla...	39583	158	172	0.131	0.037	0.073	0.040	0.162	0.221	0.1
7	duckling	/Users/bla...	17109	99	119	0.068	0.050	0.033	0.055	0.184	0.189	0.1
8	foal	/Users/bla...	39210	147	177	0.061	0.252	0.040	0.155	0.481	0.348	0.1
9	goat	/Users/bla...	53039	221	179	0.265	0.124	0.017	0.019	0.176	0.110	0.2
10	goose	/Users/bla...	34442	141	202	0.355	0.246	0.159	0.000	0.422	0.374	0.1
11	hen	/Users/bla...	41716	134	168	0.389	0.062	0.037	0.083	0.429	0.218	0.1
12	horse	/Users/bla...	69109	285	195	0.280	0.229	0.084	0.095	0.387	0.295	0.2
13	kid	/Users/bla...	36290	170	160	0.131	0.140	0.024	0.067	0.130	0.030	0.1
14	lamb	/Users/bla...	35520	123	168	0.358	0.034	0.189	0.055	0.331	0.162	0.4
15	ox	/Users/bla...	56401	191	189	0.520	0.003	0.096	0.106	0.139	0.235	0.2

We have no idea what these features are, except that they represent some higher-abstraction concepts in the deep neural network (ok, this is not very helpful in terms of interpretation). Yet, we have just described images with vectors that we can compare and measure their similarities and distances. Distances? Right, we could do clustering. Let's cluster the images of animals and see what happens.



To recap: in the workflow about we have loaded the images from the local disk, turned them into numbers, computed the distance matrix containing distances between all pairs of images, used the distances for hierarchical clustering, and displayed the images that correspond to the selected branch of the dendrogram in the image viewer. We used cosine similarity to assess the distances (simply because of the dendrogram looked better than with the Euclidean distance).

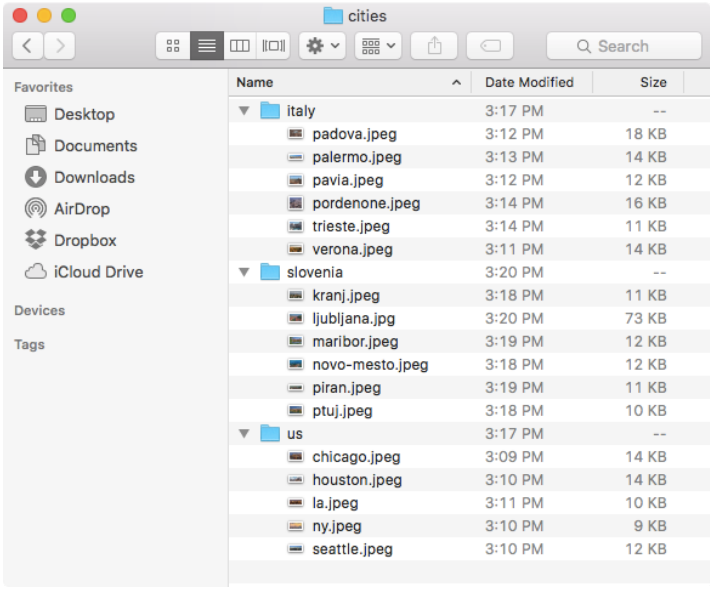
Even the lecturers of this course were surprised at the result. Beautiful!



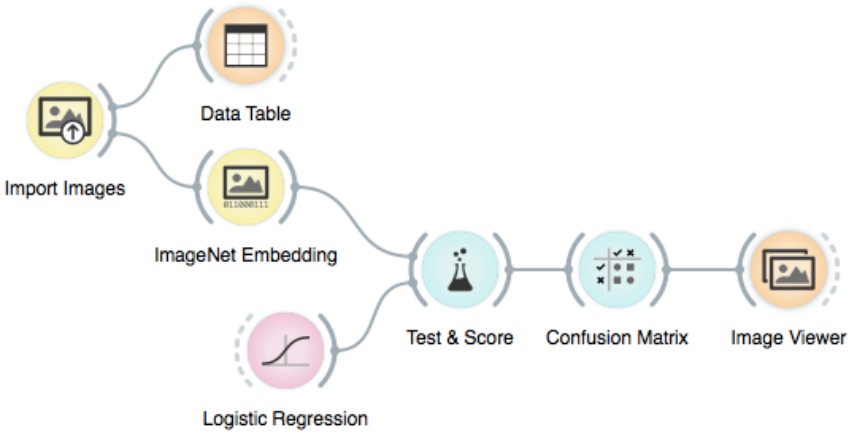
Lesson 31: Images and Classification

In this lesson, we are using images of cities (http://bit.ly/2cl7uvT) in the classification setup. But this same data set could be explored in clustering as well. The workflow would be the same as the one from previous lesson. Try it out! Do Italian cities cluster next to American or are their photos more similar to Slovene cities?

We can use image data for classification. For that, we need to associate every image with the class label. The easiest way to do this is by storing images of different classes in different folders. Take, for instance, images of cities in US, Italy and Slovenia. This is how we have stored them on the disk:



Country names (Italy, Slovenia, US) will now become class labels for the images. We are just a step away from testing if logistic regression can classify city images to its corresponding country. The data set is small: make sure you use leave-one-out for evaluation in Test & Score widget instead of cross validation.



The AUC score is not the highest, but we are able to check where the mistakes are made. It turns out the cities in Italy and Slovenia look quite similar. We could get much better accuracy if we would classify only the cities in Italy vs. US, or Slovenia vs US. Check if this is really so. Use Select Row widget.