BIOLAB AND COLLABORATORS

DATA SCIENCE FOR ARCHAEOLOGY

BIOLAB

First printing, March 2021

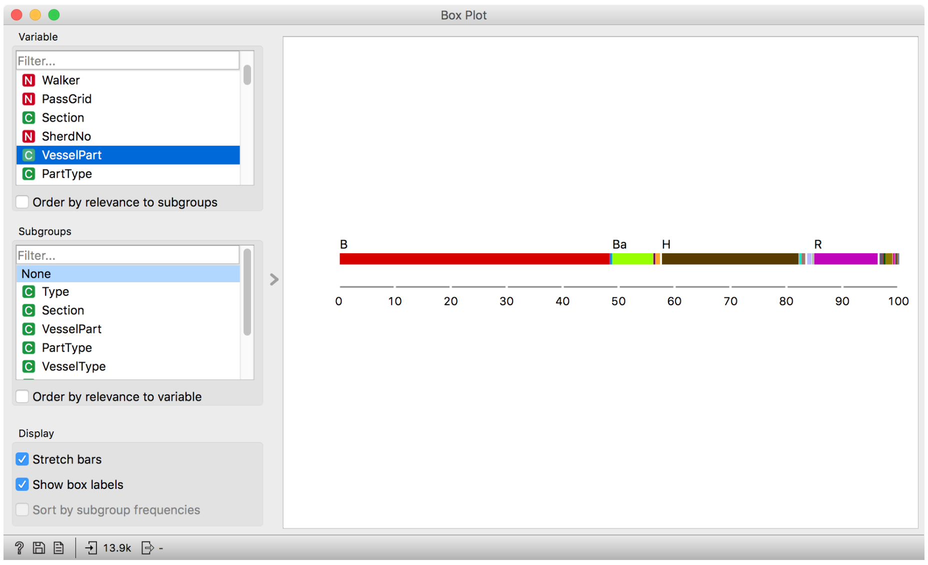
Contents

Assignment: Data Preprocessing

Data preprocessing is crucial in archeology. The data often contains many diverse labels, some of which are certain and others speculative. Machine learning doesn't handle categorical data with many different labels well, hence we need to preprocess the data.

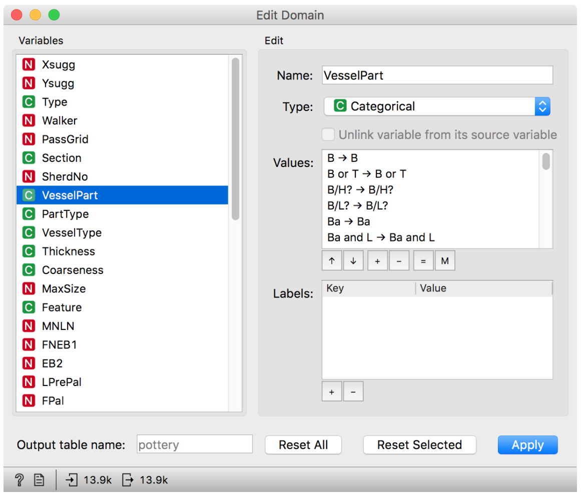
We will use the pottery data from the Antikythera data base. The data contains 13,865 data instances and 36 features describing the pottery collected across the full extent of the Greek island of Antikythera.

Let us look at the VesselPart feature, which describes the part of the ceramic vessel involved and it has 34 values.



But looking at the *Box Plot*, we notice there are 4 fairly frequent values, while others are rare. We would like to keep B (body), Ba (base), H (handle), and R (rim), and consider all the others as one group.

We will use *Edit Domain* to merge infrequent values together. Select *VesselPart* and press M. This opens a dialogue for merging values. You can use any technique you wish. Here, since we know there are 4 frequent values, we will use the final option, *Group all except 4 most frequent values*. We can also give the name to the new value, but let's leave it a *other* for now.


Variables: Xsugg, Ysugg, Type, Walker, PassGrid, Section, SherdNo, VesselPart, PartType, VesselType, Thickness, Coarseness, MaxSize, Feature, MNLN, FNEB1, EB2, LPrePal, FPal

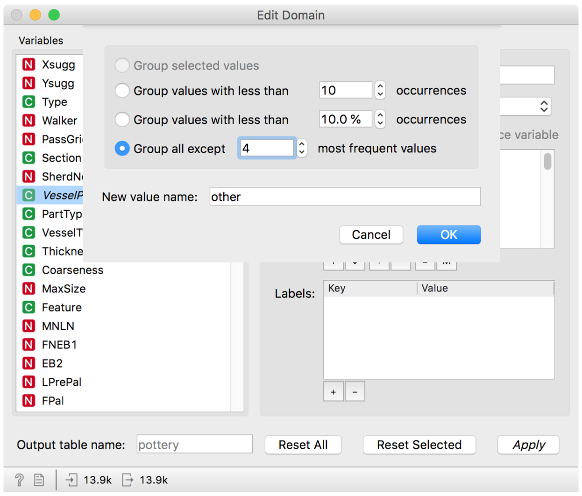
Name: VesselPart

Type: Categorical

Unlink variable from its source variable

Values:
B → B
B or T → B or T
B/H? → B/H?
B/L? → B/L?
Ba → Ba
Ba and L → Ba and L

Output table name: pottery

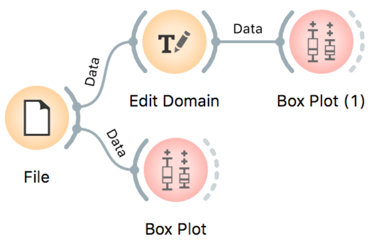

Group selected values

Group values with less than 10 occurrences

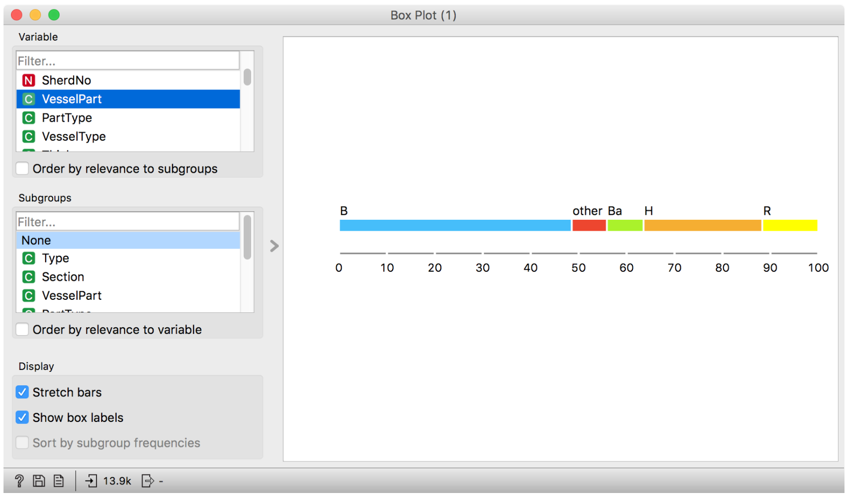
Group values with less than 10.0 % occurrences

[✓ Group all except 4 most frequent values]

New value name: other



You can see the less frequent values have been remapped. Do not forget to press *Apply* once you have finished editing. Looking at the Box Plot, the values are now much easier to interpret.


Variable: VesselPart

Subgroups: None

Display:
[✓ Stretch bars]
[✓ Show box labels]
Sort by subgroup frequencies

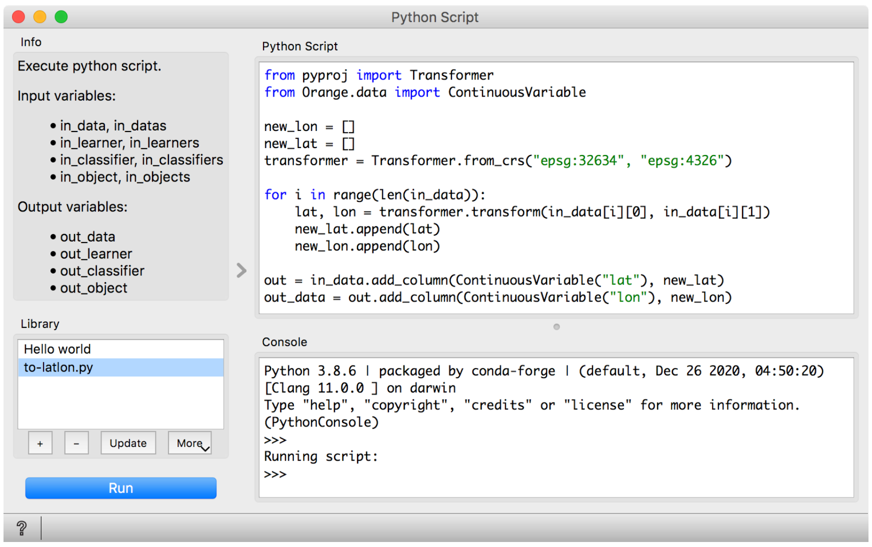
Assignment

Preprocess the pottery data using the technique from above and answer the following questions:

- Put metadata to meta attributes using Select Columns. Which features did you put there? Why?
- Which features had to be preprocessed by merging infrequent values?
- Is there another way we could merge values?
- Use Box Plot after Edit Domain and

Assignment: Geolocations

The Antikythera data set contains information on the geolocation of artefacts. The location is encoded as *Xsugg*, *Ysugg* coordinates in *epsg:32634* system. We need to transform the data to the more standard *EPSG: 4326* projection, which enables us to see the data on a map. We will do this with the help of a *Python Script*.



Execute python script.

Input variables:

- in_data, in_datas
- in_learner, in_learners
- in_classifier, in_classifiers
- in_object, in_objects

Output variables:

- out_data
- out_learner
- out_classifier
- out_object

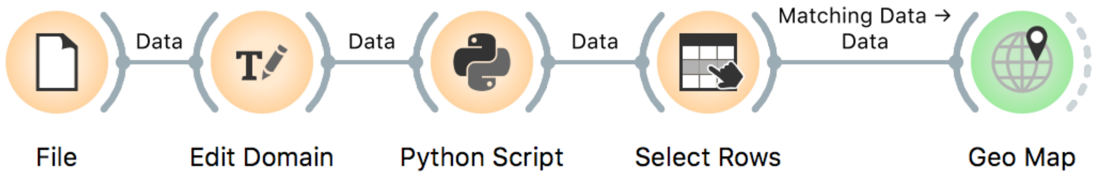
```
from pyproj import Transformer
from Orange.data import ContinuousVariable

new_lon = []
new_lat = []
transformer = Transformer.from_crs("epsg:32634", "epsg:4326")

for i in range(len(in_data)):
    lat, lon = transformer.transform(in_data[i][0], in_data[i][1])
    new_lat.append(lat)
    new_lon.append(lon)

out = in_data.add_column(ContinuousVariable("lat"), new_lat)
out_data = out.add_column(ContinuousVariable("lon"), new_lon)
```

```
Python 3.8.6 | packaged by conda-forge | (default, Dec 26 2020, 04:50:20)
[Clang 11.0.0 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(PythonConsole)
>>>
Running script:
>>>
```

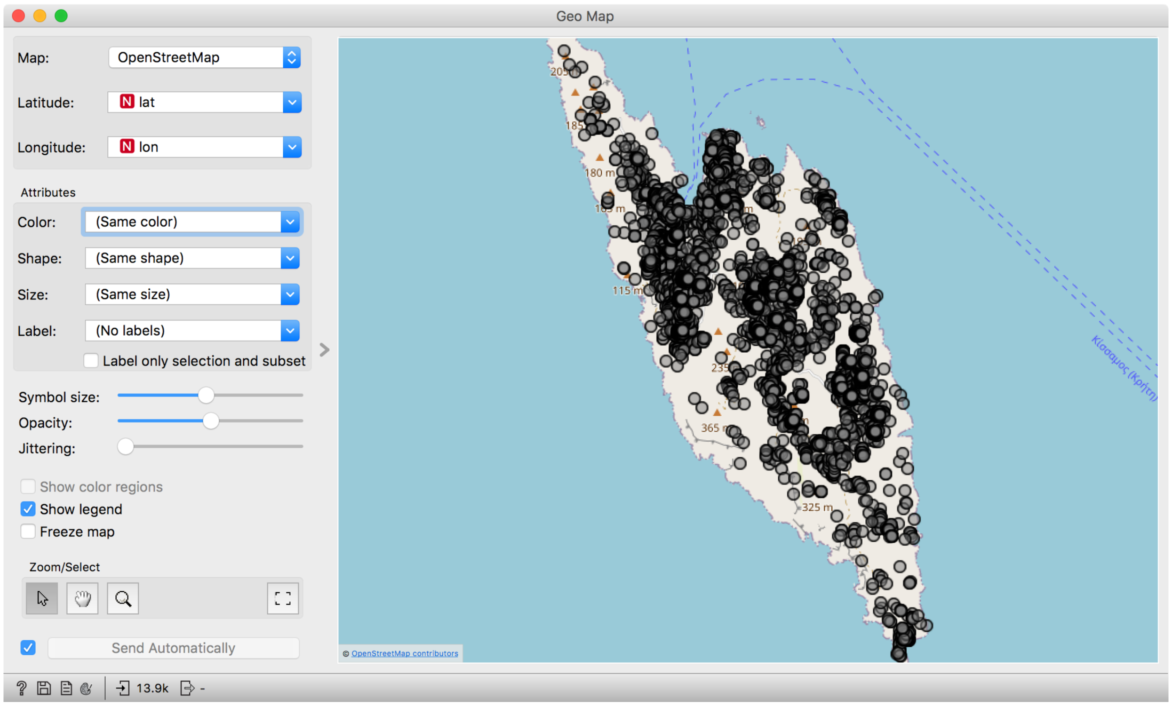
Now, we can plot the data on the map. We will use the Geo add-on for this. If you have not installed the add-on yet, go to Options –> Add-ons and install it. Restart Orange for the add-on to appear.

Connect *Geo Map* to Python Script. The widget should automatically find latitude and longitude attributes, but if it doesn't, you can set them manually. The plot shows where the pottery shards were collected.

For Python Script to work, one needs to have the 'pyproj' package installed in the Orange environment. To bypass this, here are the preprocessed data you can load with File and connect directly to Geo Map.



Assignment

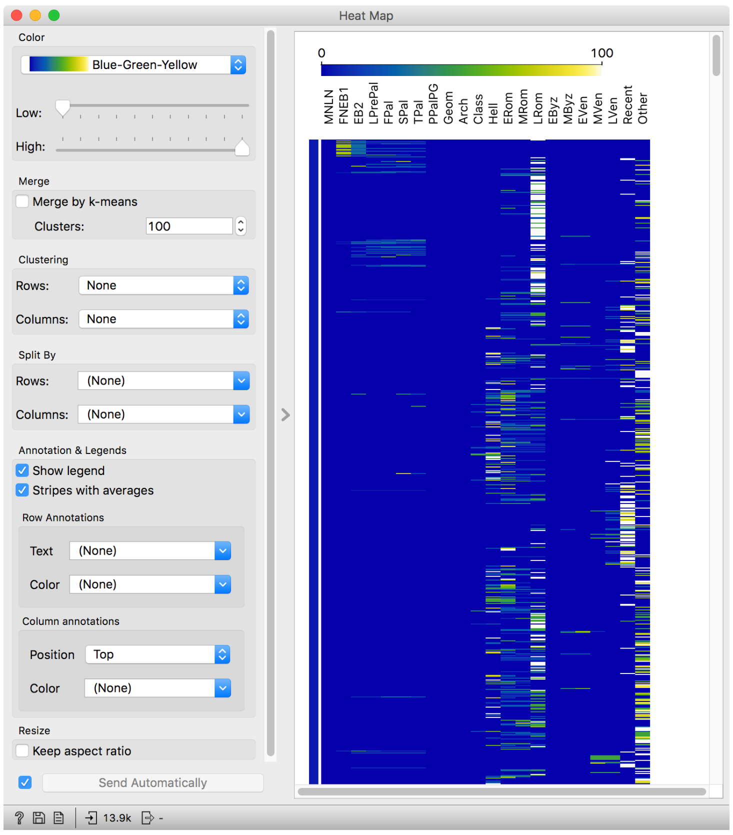


Use Geo Map widget and color the points by an appropriate attribute to answer the following questions:

- Does the location of the shard correspond to the way it was collected?
- Use *Select Rows* to keep only data instances with over 75% certainty that they belong to the a certain period. Go through different periods (attributes from MNLN to Other) and observe if items from one period were found in a specific location of the island. Some periods will produce no results (no items with such high certainty), but that's ok. Skip them.
- Use *Select Rows* to explore categorical variables. For example, set the filter to *VesselPart is B* and observe where in the island the shards belonging to the body of the vessel were found. Explore variables of your interest and draw conclusions.

Assignment: Hierarchical Clustering

SHARDS BELONG TO DIFFERENT TIME PERIODS. Periods are described with 22 numeric variables, each defining the likelihood of a shard belonging to the said period. First, let us visualize the data in a visualization that shows the value of numeric variables with color - the higher the value, the brighter the number. Such a visualization is called a *Heat Map*.


Heat Map

Color
Blue-Green-Yellow
Low:
High:

Merge
☐ Merge by k-means
Clusters: 100

Clustering
Rows: None
Columns: None

Split By
Rows: (None)
Columns: (None)

Annotation & Legends
☑ Show legend
☑ Stripes with averages

Row Annotations
Text (None)
Color (None)

Column annotations
Position Top
Color (None)

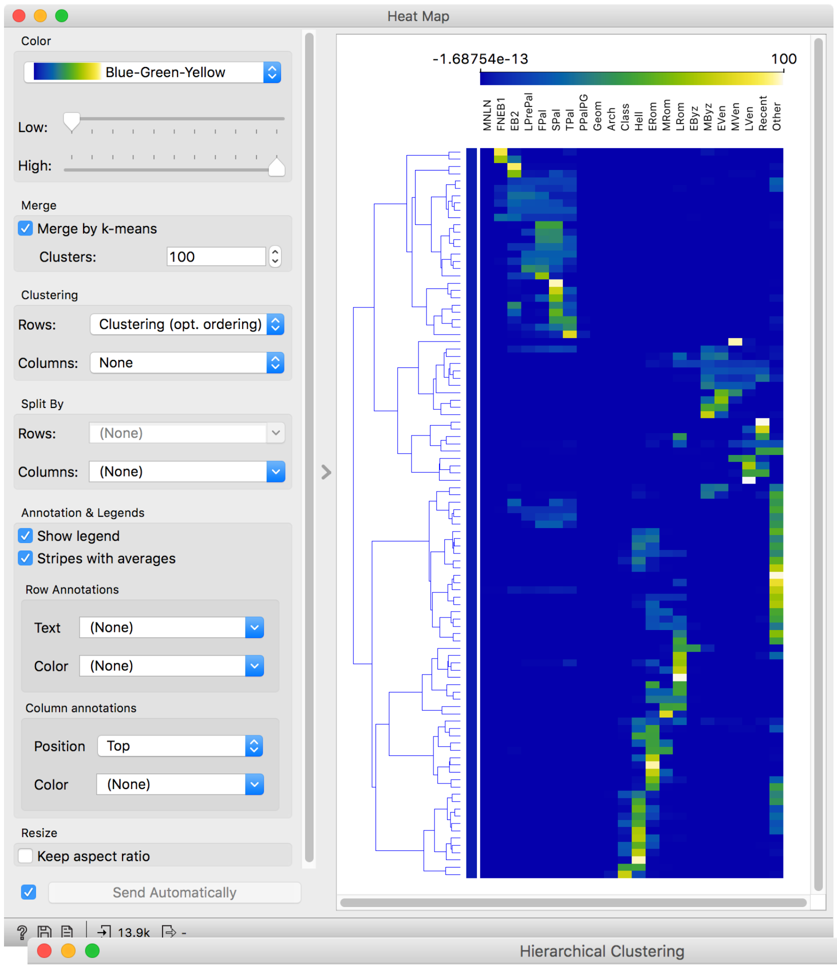
Resize
☐ Keep aspect ratio

☑ Send Automatically

13.9k -

By default, Heat Map shows each data instance in a row and maps the values to the color scale. But our data is so unorganized, there is no way we can make sense of it! First, since we have many rows in our data set, let us join together rows that are the most similar. We will do this with *Merge by k-means* option and set the number of clusters to 100. This is keep only 100 most different rows.

Visualization got simpler, but it is still unorganized. Would it be nice to have similar rows closer together? Of course it would! We can use hierarchical clustering to re-order rows, so that similar rows will be put next to each other. Use *Clustering (opt. ordering)* - the plot now makes much more sense!



Color
Blue-Green-Yellow
Low:
High:

Merge
☑ Merge by k-means
Clusters: 100

Clustering
Rows: Clustering (opt. ordering)
Columns: None

Split By
Rows: (None)
Columns: (None)

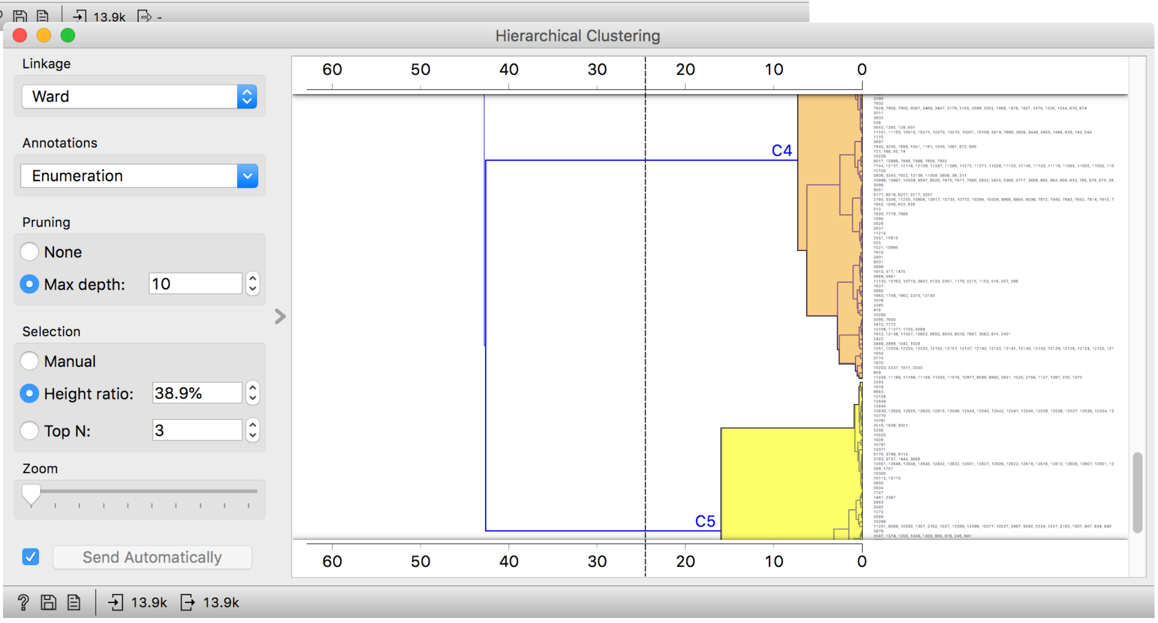
Annotation & Legends
☑ Show legend
☑ Stripes with averages

Row Annotations
Text (None)
Color (None)

Column annotations
Position Top
Color (None)

Resize
☐ Keep aspect ratio

☑ Send Automatically



Linkage
Ward

Annotations
Enumeration

Pruning
○ None
● Max depth: 10

Selection
○ Manual
● Height ratio: 38.9%
○ Top N: 3

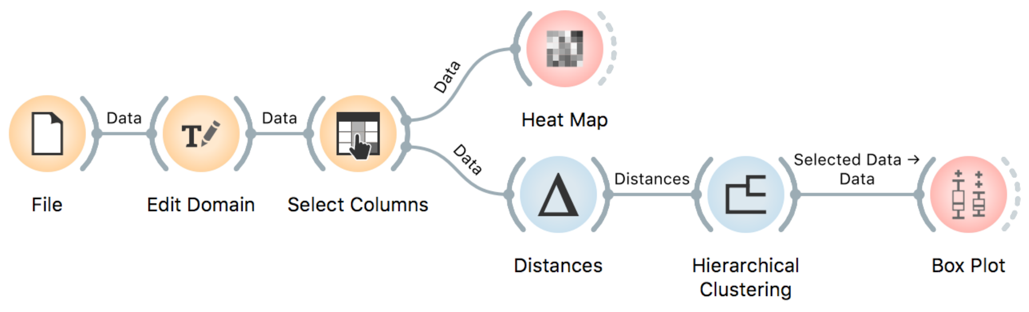
Zoom

☑ Send Automatically

In the top left corner, we have shards from the early periods. In the bottom central part shards from Roman and Hellenic periods. And in the middle right part recent or unidentified shards.

We can repeat the same, but a more exact procedure, with *Hierarchical Clustering*. First, let us pass the data to *Distances* to compute the distance matrix. As our data has a fair number of dimensions (22 to be precise), we will use *cosine distance* and *Ward linkage* for the measure of cluster similarity.

In Hierarchial Clustering, we see a large dendrogram. To make it easier to read, set *Max depth* to 10 (this will cut the tree at the tenth split). Use Zoom to zoom out as much as possible.



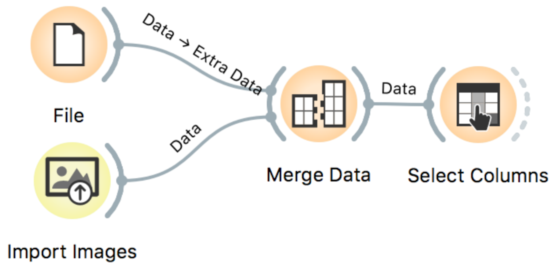
Assignment

Use Box Plot to explore the clusters:

- What is a good number of clusters? Why?
- Cut the dendrogram at the desired level (up to you). Which period(s) does each of the clusters represent? Could you name them?
- What defines each period? Use Box Plot with *Data* output to find out.

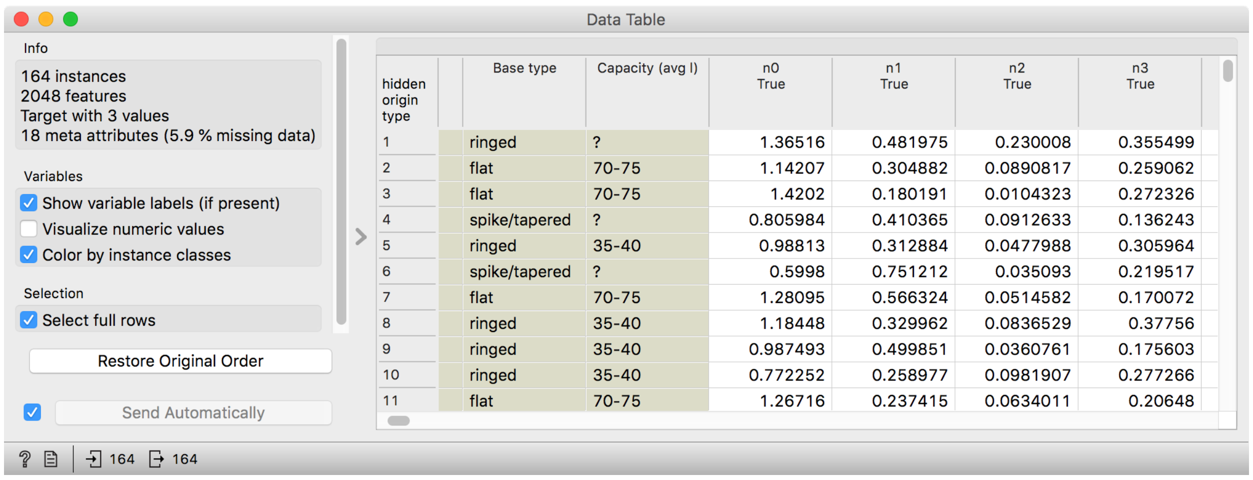
Assignment: Image Analytics

Archaeologists often analyze images of artefacts, from sketches of items to photographs of sites and findings. We retrieved 164 digital sketches of amphorae belonging to 3 different types - Dressel, Gauloise, and Keay. Consider we have an archive of such images and we wish to automatically label new sketches with their corresponding amphora type.



First, we load images with *Import Images* from the Image Analytics add-on. Our images are organized, each type of amphora being in it own folder. Folders will be treated as categories upon loading. This means each image will get a special label with the name of the folder - the target variable.

Second, we have metadata belonging to each subtype of amphora (Dressel 6b, Gauloise 13, etc.). Load this with *File* and join the two tables with *Merge Data*. The widget should automatically recognize the attribute to merge by, but it case it doesn't, set both to *image name*. Finally, we will use *Select Columns* to put all the metadata to metas. Remember, our aim is to classify amphorae based on their image, not their description.



Info
164 instances
2048 features
Target with 3 values
18 meta attributes (5.9 % missing data)

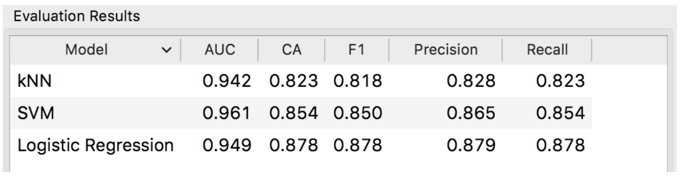
Variables
☑ Show variable labels (if present)
☐ Visualize numeric values
☑ Color by instance classes

Selection
☑ Select full rows

hidden origin type		Base type	Capacity (avg l)	n0 True	n1 True	n2 True	n3 True
1		ringed	?	1.36516	0.481975	0.230008	0.355499
2		flat	70-75	1.14207	0.304882	0.0890817	0.259062
3		flat	70-75	1.4202	0.180191	0.0104323	0.272326
4		spike/tapered	?	0.805984	0.410365	0.0912633	0.136243
5		ringed	35-40	0.98813	0.312884	0.0477988	0.305964
6		spike/tapered	?	0.5998	0.751212	0.035093	0.219517
7		flat	70-75	1.28095	0.566324	0.0514582	0.170072
8		ringed	35-40	1.18448	0.329962	0.0836529	0.37756
9		ringed	35-40	0.987493	0.499851	0.0360761	0.175603
10		ringed	35-40	0.772252	0.258977	0.0981907	0.277266
11		flat	70-75	1.26716	0.237415	0.0634011	0.20648

But machine learning works with numbers and images are not numbers, but complex objects. We need a way to transform images to a numeric description. We will use *Image Embedding*, a widget that uses pre-trained deep models to infer image profiles. A common embedder is Google's Inception v3, which we will use for this exercise. Finally, we have our numbers! Specifically, a 2048-dimensional vector for each image.

Now, we can train a few classifiers. Common models for image classification are *kNN*, *Logistic Regression*, and *SVM*, so let's use these. It looks like SVM has the highest AUC score, but Logistic Regression performs the best in other metrics.

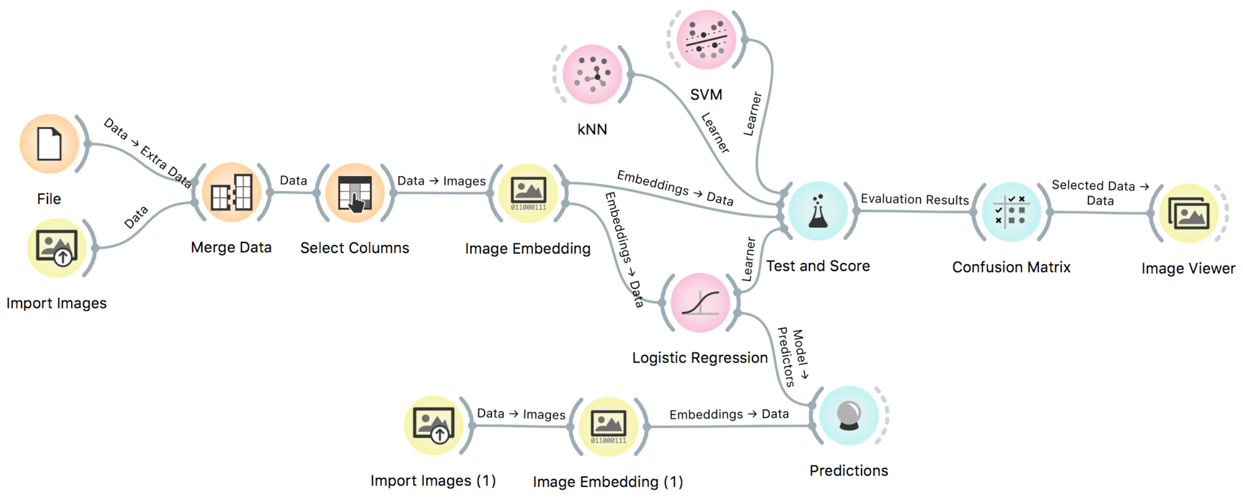


Evaluation Results

Model	AUC	CA	F1	Precision	Recall
kNN	0.942	0.823	0.818	0.828	0.823
SVM	0.961	0.854	0.850	0.865	0.854
Logistic Regression	0.949	0.878	0.878	0.879	0.878

Assignment

- Observe *Confusion Matrix* and determine which model is better to use in practice and why.
- Which type of amphorae is the most difficult to predict? How could we improve the performance? Select different misclassifications in Confusion Matrix and explore them in Image Viewer.
- There are three new images to predict. Use amphorae-predict and load it with Import Images. Pass it to Image Embedding, then use Predictions and the chosen model to predict image type. How accurate is the model? Would you use it in practice? Why (not)?



Bibliography