Claudio Sartori

Elaboration from the example given in Sebastian Raschka, 2015

https://github.com/rasbt/python-machine-learning-book

Machine Learning - Lab

Working with Unlabeled Data – Clustering Analysis

Use **DBSCAN**

Overview

In this example we will use an *artificial* data set

1. load the data
2. check the shape and plot the content
3. observe the plot and decide which are the most interesting columns, to use in the plots of the clusters
    - make a 2d plot of the two most promising columns
    - exclude from the dataset the columns which seem to be only noise</br>
4. initialize and `fit_predict` an estimator for `DBSCAN`, using the default parameters, then print the results</br>
    - print the estimator to check the parameter values
    - the labels are the unique values of the predicted values
    - print if there is noise
        - if there is noise the first cluster label will be `-1`
    - print the number of clusters (noise excluded)
        - the other clusters are labeled starting from `0`
    - for each cluster (noise excluded) compute the **centroid**
        - plot the data with the centroids and the colors representing clusters
        - use the `plot_clusters` function provided
5. find the best parameters using `ParameterGrid`
    - prepare a dictionary with the parameters lists
        - generate the list of the parameter combinations with `ParameterGrid`
    - for each combination of parameters
        - initialize the DBSCAN estimator
        - `fit_predict`
        - extract the labels and the number of clusters excluding the *noise*
        - compute the silhouette score and the number of unclustered objects (noise)
        - filter and print the parameters and the results
            - print if the silhouette score is above a threshold and the percentage of unclustered is below a threshold
6. observe visually the most promising combination of parameters

- fit and predict the estimator
- plot the clusters
- compute the silouette scores for the individual samples using the function `silhouette_samples`
- plot the silhouette scores for each sample using the function `plot_silhouette`

In [ ]:
```
from IPython.display import Image
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score, silhouette_samples
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import ParameterGrid
from sklearn.preprocessing import MinMaxScaler

%matplotlib inline

rnd_state = 42 # This variable will be used in all the procedure calls allowing a r
               # in this way the running can be perfectly reproduced
               # just change this value for a different experiment

# the .py files with the functions provided must be in the same directory of the .
from plot_clusters import plot_clusters      # python script provided separately
```

In [ ]:
```
# help(plot_clusters)
```

In [ ]:
```
# help(plot_silhouette)
```

1. Load the data

In [ ]:
```
# data_file = 'ex1_4dim_data.csv'
# data_file = 'ex1_4dim_mod_data.csv'
# data_file = 'ex1_data.csv'
data_file = 'ex1_4dim_data.csv'
delimiter = ','
```

In [ ]:
```
# cov_mat = np.cov(X, rowvar = False) # occorre la trasposta
# import seaborn as sns
# sns.heatmap(cov_mat, cmap="YlGnBu", annot=True)
```

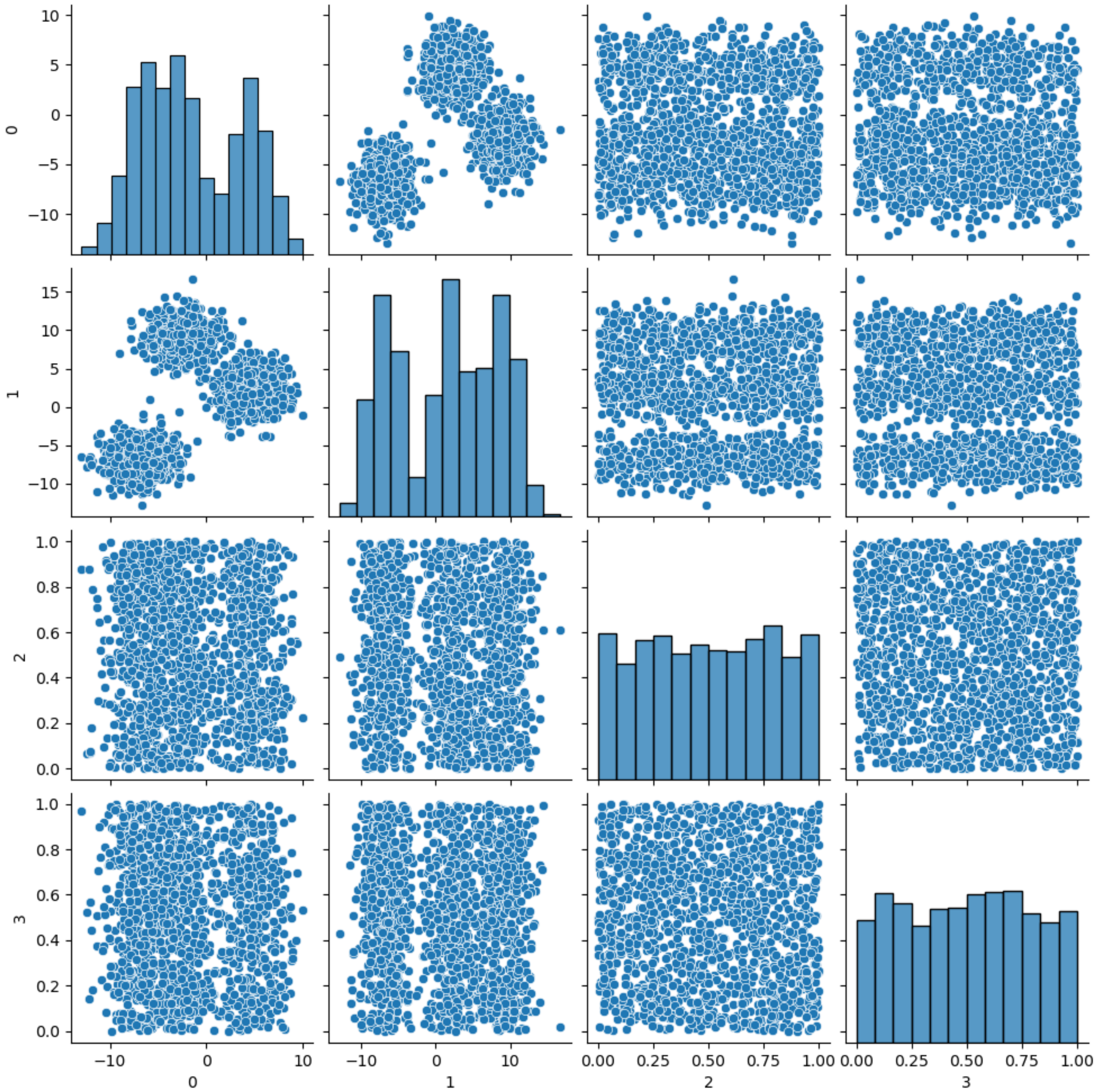
2. Inspect

In [ ]:

Out[ ]:
```
(1500, 4)
```

In [ ]:

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x1f9919c04c0>
```



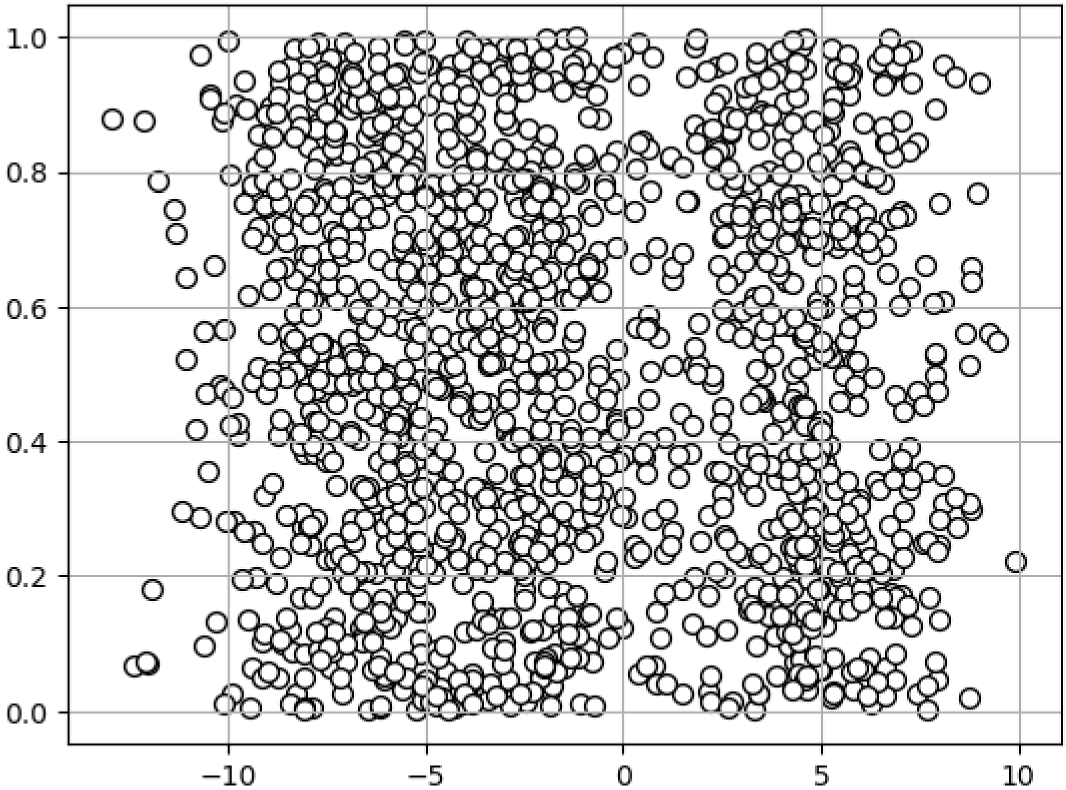
In [ ]:
```
X = X[:,[0,2]]
focus = [0,1]
# focus = [0,2]
```

3. Observing the pairplots

In this simple example you can easily see which are the two most interesting columns.

All the plots will focus on those columns

In [ ]:
```
plt.scatter(X[:,focus[0]], X[:,focus[1]]
            , c='white'          # color filling the data markers
            , edgecolors='black' # edge color for data markers
            , marker='o'         # data marker shape, e.g. triangles (v<>^), square
            , s=50)              # data marker size
plt.grid()  # plots a grid on the data
plt.show()
```



4. Initialize, fit_predict and plot the clusters

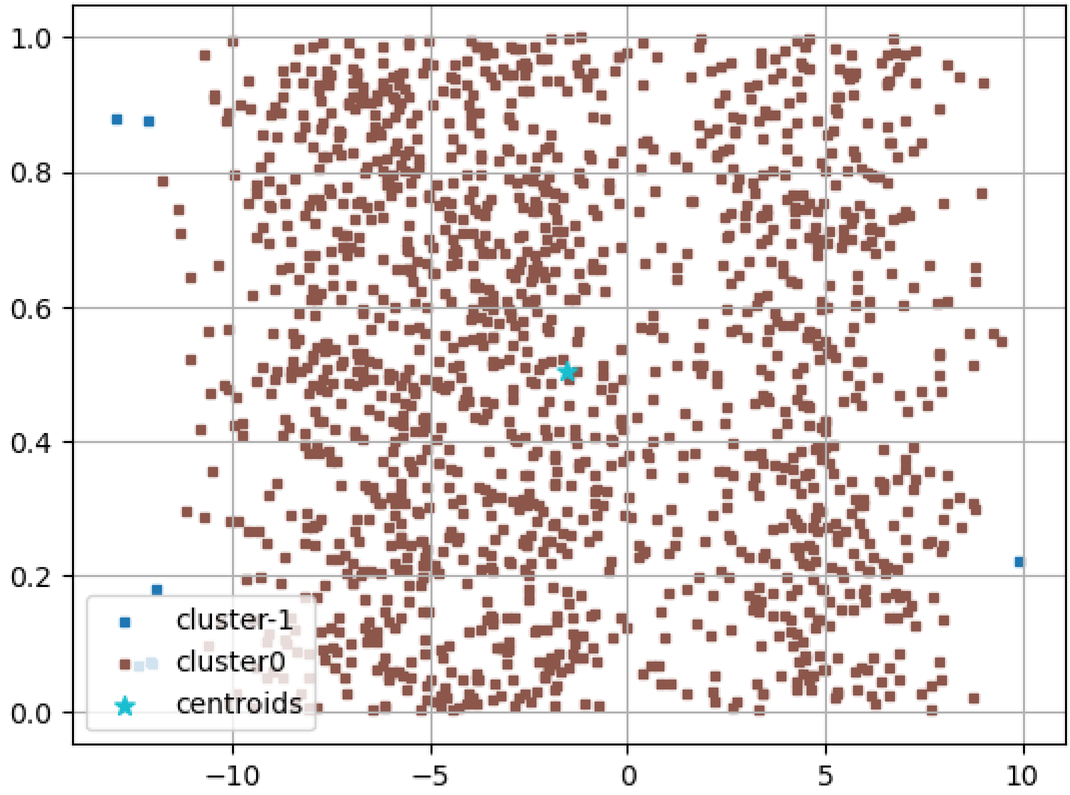
In [ ]:

```
DBSCAN()
```

In [ ]:

```
There is noise
There is/are 0 cluster(s)
```

In [ ]:



5. Find the best parameters using `ParameterGrid`

In [ ]:

Arrange DBSCAN results in a dataframe, for easier presentation and filtering

In [ ]:

In [ ]:

```
sil_thr = 0  # visualize results only for combinations with silhouette above the th
unc_thr = 10 # visualize results only for combinations with unclustered% below the
```

Out[ ]:

	eps	min_samples	n_clusters	silhouette	unclust%
131	0.24	6.0	2.0	0.291228	2.800000
140	0.25	6.0	2.0	0.291267	2.733333
149	0.26	6.0	2.0	0.289177	2.266667
150	0.26	7.0	2.0	0.261636	3.066667
158	0.27	6.0	2.0	0.289177	2.266667
...	...	...	...	...	...
244	0.39	9.0	2.0	0.288878	1.000000
246	0.40	2.0	2.0	0.327134	0.266667
247	0.40	3.0	2.0	0.327134	0.266667
248	0.40	4.0	2.0	0.327443	0.333333
249	0.40	9.0	2.0	0.288878	1.000000

62 rows × 5 columns

6. Observe

- Observe visually the most promising combination of parameters.
- Plot the clusters with the centers
- Plot the silhouette indexs for all the clustered samples

In [ ]:

```
There are 24 clusters
```

In [ ]:

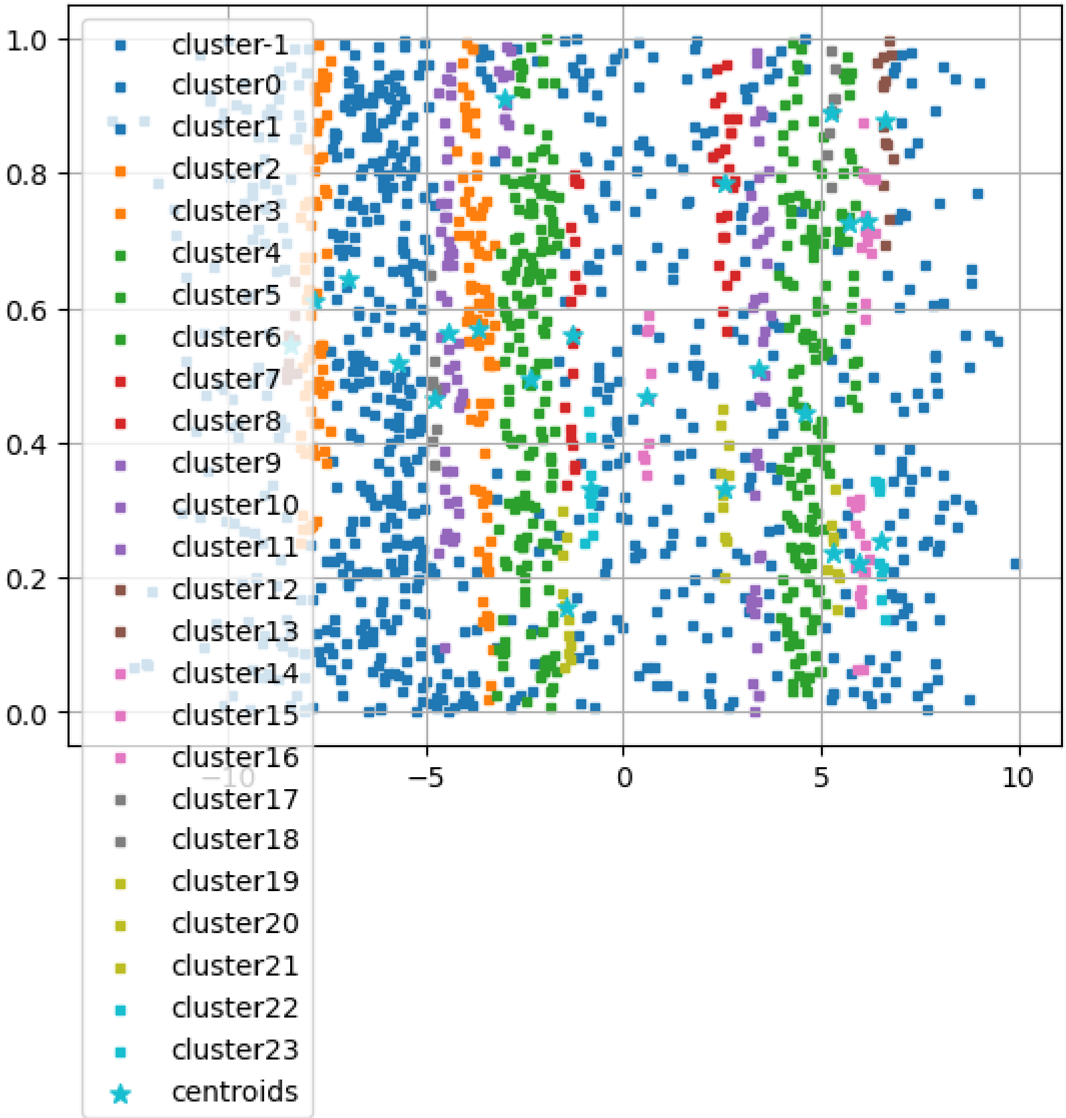
```
The cluster labels are [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 1
9 20 21 22 23]
```

In [ ]:

Out[ ]:

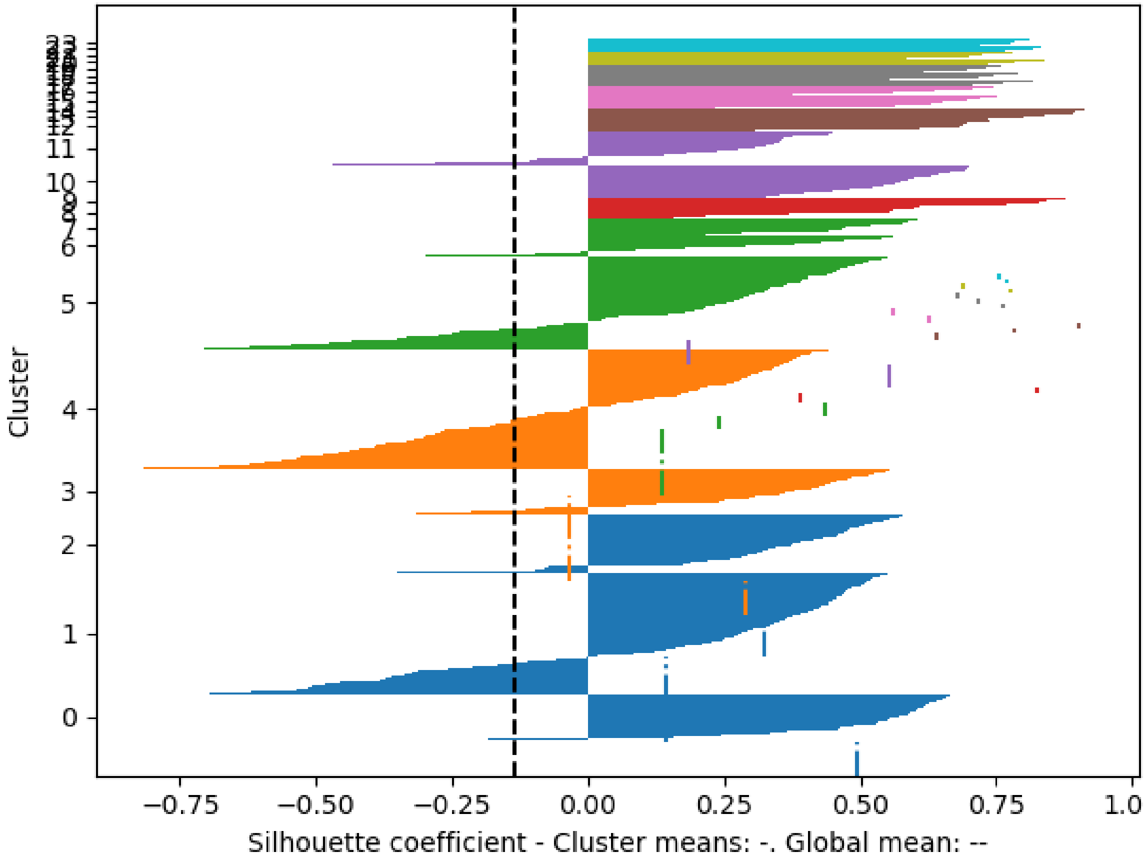
```
array([[-6.97417201,  0.64263635],
       [-5.6901368 ,  0.51840531],
       [-3.66759545,  0.57000177],
       [-7.81821402,  0.61023013],
       [-2.34880835,  0.49459378],
       [ 4.56923456,  0.44549656],
       [ 5.7123243 ,  0.7267118 ],
       [ 2.57406902,  0.78610145],
       [-1.26612109,  0.56139268],
       [-2.98343408,  0.90987031],
       [ 3.43364259,  0.50821984],
       [-4.41823681,  0.56166179],
       [ 6.633802  ,  0.87812558],
       [-8.41879611,  0.54583064],
       [ 0.6088741 ,  0.46699471],
       [ 5.98235741,  0.22087922],
       [ 6.16510539,  0.72873768],
       [ 5.26210782,  0.89027269],
       [-4.79510344,  0.46500772],
       [-1.43949617,  0.15547103],
       [ 5.32206087,  0.23615664],
       [ 2.55382203,  0.33192892],
       [-0.8339198 ,  0.33004399],
       [ 6.50099844,  0.25441864]])
```

In [ ]:



In [ ]:

```
# from plot_silhouette import plot_silhouette  # python script provided separately
from plot_silhouette_w_mean import plot_silhouette  # python script provided separa
plot_silhouette(silhouette,y_db)
```



A quick look to the width of data ranges

In [ ]:

Out[ ]:
```
array([22.84721703,  0.99878403])
```