Estimation, Detection and Analysis II

03 - Group analysis

K-means

k-medoids

DBSCAN

Hierarchical Clustering

K-means

The purpose of this exercise is to reproduce "Lab 2 – Data Mining: Clustering #1" solved with Weka. At first, you should recall the results of that exercise.

```
kMeans
…
Within cluster sum of squared errors: 113.58260073260074
…
Final cluster centroids:
                           Cluster#
Attribute         Full Data         0          1          2          3          4
                    (100.0)     (26.0)     (27.0)      (5.0)     (14.0)     (28.0)
=============================================================================
Dealership               0.6     0.9615     0.6667          1     0.8571          0
Showroom                0.72     0.6923     0.6667          0     0.5714          1
ComputerSearch          0.43     0.6538          0          1     0.8571     0.3214
M5                      0.53     0.4615      0.963          1     0.7143          0
3Series                 0.55     0.3846     0.4444        0.8     0.0714          1
Z4                      0.45     0.5385          0        0.8     0.5714     0.6786
Financing               0.61     0.4615     0.6296        0.8          1        0.5
Purchase                0.39          0     0.5185        0.4          1     0.3214

…

Clustered Instances

0       26 ( 26%)
1       27 ( 27%)
2        5 (  5%)
3       14 ( 14%)
4       28 ( 28%)
```

1. Upload the "bmw-browsers.csv" file found in Moodle

```
import pandas as pd
data = pd.read _csv("file_path/bmw-browsers.csv")
```

2. Run k-means with 5 clusters

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, random_state=0).fit(data)
```

3. View the clustering score (Sum of distances of samples to their closest cluster center)

```
print("SSE: ",kmeans .inertia_)
```

Result:

```
SSE: 101.59821734329748
```

Note : the intrinsic randomness of k-means operation (different initializations lead to different results) will result in different results

4. Visualize the centroids (it is necessary to define that the floats appear with 2c.d.)

```
import numpy as np
np.set_printoptions(formatter={'float_kind': '{:.3f}'.format})
print("Centroids:")
print(kmeans.cluster_centers_)
```

Result:

```
Centroids:
[[0.923 0.731 0.692 0.808 0.308 0.923 0.769 0.346]
[0.174 1.000 0.522 0.043 1.000 0.522 0.217 0.000]
[0.727 0.591 0.318 0.864 0.455 0.000 1.000 0.955]
[0.000 1.000 0.100 -0.000 1.000 0.900 1.000 0.900]
[0.842 0.368 0.263 0.632 0.211 0.000 0.211 0.000]]
```

Note : the matrix is transposed with respect to what is shown by Weka. In this case, each row represents a cluster, and the columns represent dataset variables.

5. Determine how many instances were inserted into each cluster

```
data['kmean'] = kmeans.labels_
print( "Clustered instances:")
print(data['kmean'].value_ counts( ))
```

Result:

```
Clustered instances:
0 26
1 23
2 22
4 19
3 10
```

6. Compare the results obtained with the results obtained previously with Weka.

Suggestion:

Attribute	Full Data	W0	P0	W1	P1	W2	P4	W3	P2	W4	P3
	100%	26%	26%	27%	23%	5%	10%	14%	22%	28%	19%
Dealership	0,6	0,962	0,923	0,667	0,174	1	0,842	0,857	0,727	0	0
Showroom	0,72	0,692	0,731	0,667	1	0	0,368	0,571	0,591	1	1
ComputerSearch	0,43	0,654	0,692	0	0,522	1	0,263	0,857	0,318	0,321	0,1
M5	0,53	0,462	0,808	0,963	0,043	1	0,632	0,714	0,864	0	0
3Series	0,55	0,385	0,308	0,444	1	0,8	0,211	0,071	0,455	1	1
Z4	0,45	0,539	0,923	0	0,522	0,8	0	0,571	0	0,679	0,9
Financing	0,61	0,462	0,769	0,63	0,217	0,8	0,211	1	1	0,5	1
Purchase	0,39	0	0,346	0,519	0	0,4	0	1	0,955	0,321	0,9

It is difficult to map the clusters between the two tools.

Comparison of clustering methods

Review the scikit learn documentation for the algorithms:

- k-means: https://scikit-learn.org/stable/modules/clustering.html#k-means
- k-medoids: https://scikit-learn-extra.readthedocs.io/en/stable/generated/sklearn_extra.cluster.KMedoids.html
- DBSCAN: https://scikit-learn.org/stable/modules/clustering.html#dbscan
- Agglomerative Clustering: https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering

Based on the iris dataset, create a python script that, for each of the algorithms:

- Make the parameter selection.
- Run the algorithm with the selected parameters (**note** : the Agglomerative Clustering algorithm allows several types of linkage. Try several).
- Allows you to view actual values and forecasts.

In addition, the script must accumulate in a dataset the labels assigned by clustering for later evaluation of the results

Below is an example using k-means

1. Load the dataset and define that we are only going to use the independent variables for clustering (the dependent one will be used later just to compare the clustering result with the real value)

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

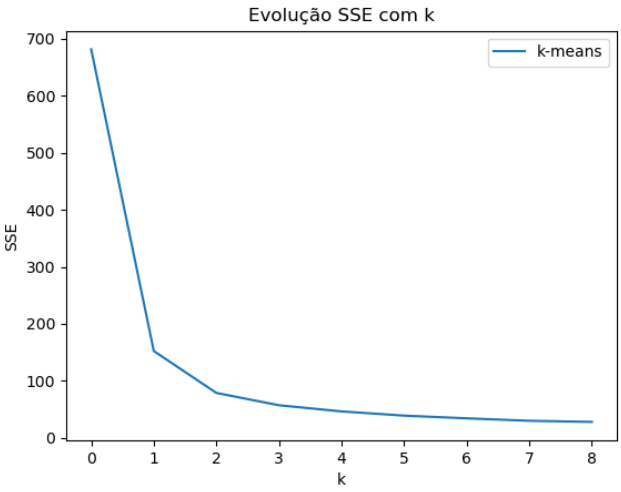
2. Create a dataframe to store the labels. For now, this dataframe only contains the information of the independent and dependent variables of the iris dataset

```
from pandas import DataFrame
date = DataFrame(X)
date['y'] = y
```

3. Import k-means and run the algorithm for different values of k. At the end, visualize the graph of the evolution of the SSE with the change in the value of k

```
from sklearn.cluster import KMeans
scores_kmeans = []
for k in range(1,10):
kmeans = KMeans(n_clusters=k, random_state=0).fit(X)
scores_ kmeans.append (kmeans.inertia_)
import matplotlib.pyplot as plt
plt.plot (scores_kmeans, label="k-means")
plt.xlabel('k')
plt.ylabel ('SSE')
plt.title ('SSE evolution with k')
plt.legend()
plt.show()
```

Result:



Based on domain knowledge, we know that the iris dataset has 3 different classes. So we would choose k=3. As you can see in the graph, there is an inflection (albeit small) for k=3. So, we can choose k=3 to do the clustering.

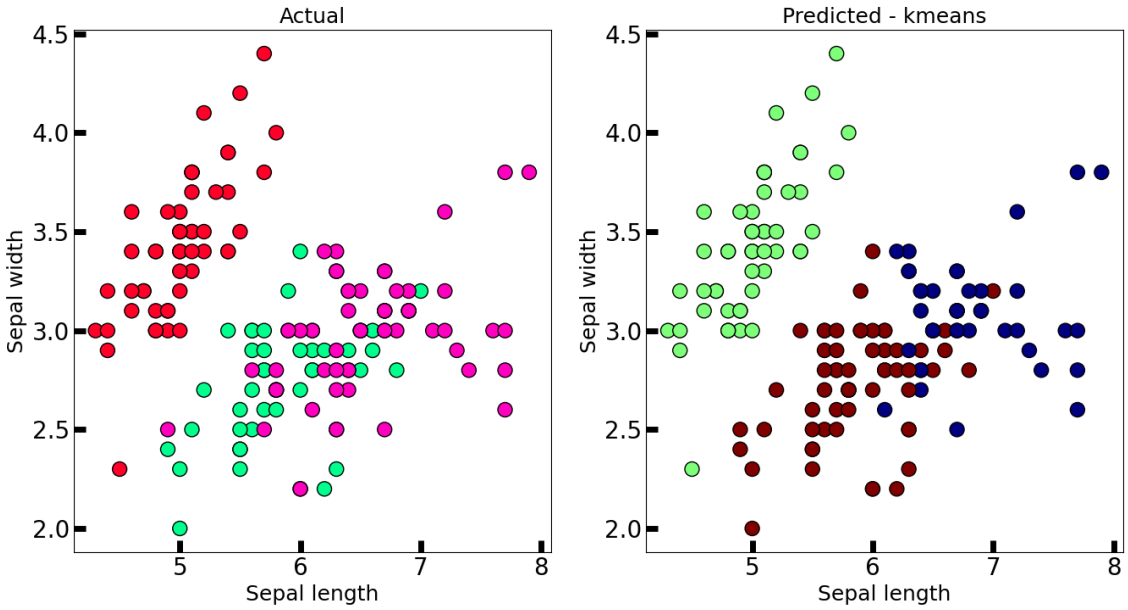
4. Execute the clustering with the selected parameter, save the labels in the dataframe

```
kmeans = KMeans( n_clusters=3, random_state=0).fit(X)
data['kmeans'] = kmeans.labels_
```

5. Show actual and assigned values by clustering side by side. In this case, we will do the mapping only considering the first two columns of the dataset, since with more than two dimensions it would be difficult to visualize the results.

```
fig, axes = plt.subplots (1, 2, figsize=(16,8))

axes[0].scatter(X[:, 0], X[:, 1], c=y, cmap='gist_rainbow',edgecolor='k', s=150)
axes[ 1].scatter(X[:, 0], X[:, 1], c=data['kmeans'], cmap='jet',edgecolor='k', s=150)

axes[ 0].set_xlabel('Sepal length', fontsize=18)
axes[ 0].set_ylabel('Sepal width', fontsize=18)
axes[ 0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[ 1].set_xlabel('Sepal length', fontsize=18)
axes[ 1].set_ylabel('Sepal width', fontsize=18)
axes[ 1].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[ 0].set_title('Actual', fontsize=18)
axes[ 1].set_title('Predicted - kmeans', fontsize=18)

plt.show ()
```

Result:



6. Analyze the results obtained

7. Based on the labels that were stored in the "data" dataframe, how could one evaluate the clustering methods regarding the External Indexes (slides 53 to 55)? **Note** : the algorithms assign different label values to the same record.