Estimação, Deteção e Análise II

03 – Análise de grupos

K-means

K-medoids

DBSCAN

Clustering hierárquico

K-means

O objetivo deste exercício é reproduzir o "Laboratório 2 – Data Mining: Clustering #1" resolvido com o Weka. Num primeiro momento, deve relembrar os resultados desse exercício.

```
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. Carregar o ficheiro "bmw-browsers.csv" que se encontra no Moodle

```
import pandas as pd
data = pd.read_csv("caminho_fich/bmw-browsers.csv")
```

2. Executar o k-means com 5 clusters

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

3. Visualizar o score (Sum of distances of samples to their closest cluster center) do clustering

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

Resultado:

```
SSE: 101.59821734329748
```

Nota: a aleatoriedade intrínseca ao funcionamento do k-means (diferentes inicializações levam a diferentes resultados) irá resultar em resultados diferentes

4. Visualizar os centroides (é necessário definir que os floats aparecem com 2c.d.)

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

Resultado:

```
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]]
```

Nota: a matriz é transposta em relação à que é mostrada pelo Weka. Neste caso, cada linha representa um cluster, e as colunas representam as variáveis do dataset.

5. Determinar quantas instâncias foram inseridas em cada cluster

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

Resultado:

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

6. Comparar os resultados obtidos com os resultados obtidos anteriormente com o Weka.

Sugestão:

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

É difícil mapear os clusters entre as duas ferramentas.

Comparação de métodos de clustering

Analisar a documentação do scikit learn para os algoritmos:

- 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

Tendo como base o dataset iris, criar um script python que, para cada um dos algoritmos:

- Faça a seleção de parâmetros.
- Execute o algoritmo com os parâmetros selecionados (**nota**: o algoritmo Agglomerative Clustering permite diversos tipos de linkage. Experimentar vários).
- Permita visualizar os valores reais e as previsões.

Para além disso, o script deve ir acumulando num dataset os labels atribuídos pelo clustering para posterior avaliação dos resultados

Abaixo encontra-se um exemplo utilizando o k-means

1. Carregar o dataset e definir que só vamos usar as variáveis independentes para clustering (a dependente será utilizada depois apenas para comparar o resultado do clustering com o valor real)

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

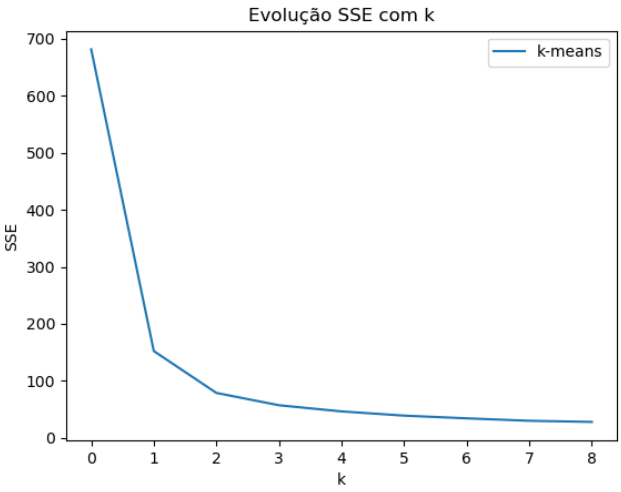
2. Criar um dataframe para guardar os labels. Para já, este dataframe fica só com a informação das variáveis independentes e dependente do dataset iris

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

3. Importar o k-means e correr o algoritmo para diversos valores de k. No fim, visualizar o gráfico da evolução do SSE com a alteração do valor de 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('Evolução SSE com k')
plt.legend()
plt.show()
```

Resultado:



Com base no conhecimento de domínio, sabemos que o dataset iris tem 3 classes diferentes. Logo, iriamos escolher k=3. Como se pode ver no gráfico, existe uma inflexão (embora pequena) para k=3. Logo, podemos escolher k=3 para fazer o clustering.

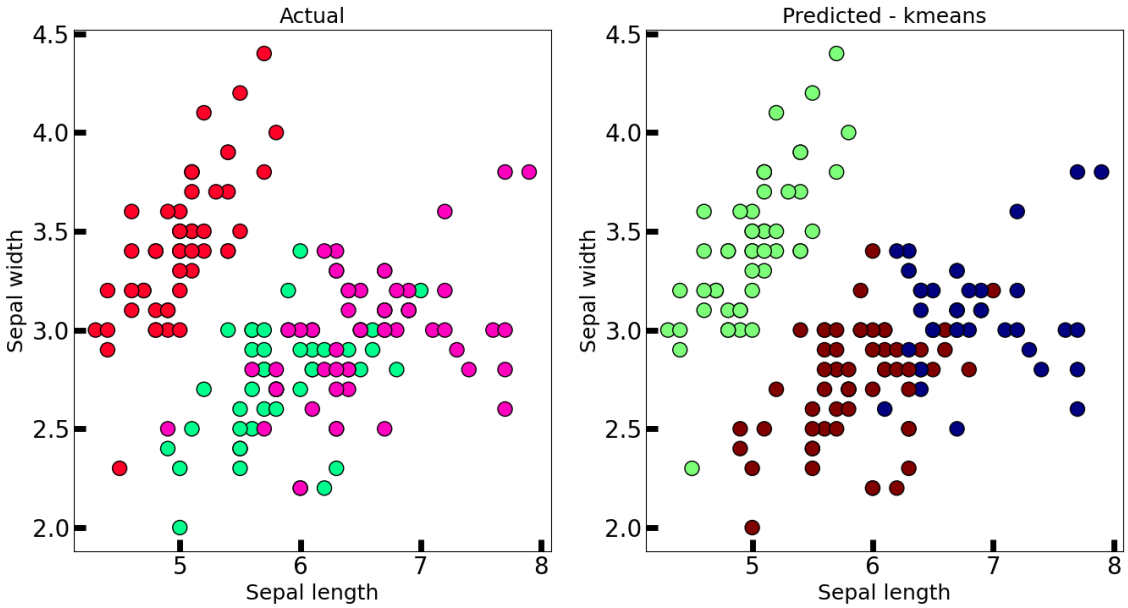
4. Executar o clustering com o parâmetro selecionado, guardar os labels no dataframe

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

5. Mostrar os valores reais e atribuídos pelo clustering lado a lado. Neste caso vamos fazer o mapeamento apenas considerando as duas primeiras colunas do dataset, dado que com mais do que duas dimensões seria complicado visualizar os resultados.

```
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()
```

Resultado:



6. Analisar os resultados obtidos

7. Tendo por base os labels que ficaram guardados no dataframe "data", como se poderia avaliar os métodos de clustering quanto aos External Indexes (slides 53 a 55)? **Nota**: os algoritmos atribuem diferentes valores de labels ao mesmo registo.