Ficha laboratorial - Clustering

Ex1. Corre o código abaixo

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn import datasets

from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering as aggClus
from scipy.cluster.hierarchy import dendrogram

Iris = datasets.load_iris()
X=Iris.data.T
print(X.shape)

# Num problema de aprend. não supervisionada não teríamos acesso a esta informação
trueClass=Iris.target
```

Na célula seguinte coloca a instrução:

```
kmeans = KMeans(n_clusters=3, init='k-means++', n_init =5, max_iter=500, tol = 0.0001, verbose=0,random_state=42)
```

e corre o código seguinte:

```
1 kmeans = kmeans.fit(X.T)

1 C= kmeans.cluster_centers_
2 y=kmeans.labels_
```

Imprime o conteúdo das variáveis C e y e analisa o seu conteúdo.

Ex.2 Corre os dois códigos apresentados abaixo. Analisa os gráficos obtidos.

```
plt.figure(figsize=(10,5))
plt.title('Clusters encontrados')
plt.plot(X[0,y==0],X[1,y==0],'g.')
plt.plot(X[0,y==1],X[1,y==1],'b.')
plt.plot(X[0,y==2],X[1,y==2],'r.')
plt.plot(C[:,0],C[:,1],'k*')
plt.axis([2.5, 8, 1.3, 5.2])
plt.grid()
plt.show()

1  y1=trueClass
plt.figure(figsize=(10,5))
plt.title('Classes originais')
plt.plot(X[0,y1=0],X[1,y1=0],'b.')
plt.plot(X[0,y1=1],X[1,y1=1],'g.')
plt.plot(X[0,y1=2],X[1,y1=2],'r.')
plt.axis([2.5, 8, 1.3, 5.2])
plt.grid()
plt.show()
```



Ex. 3 Corre o código abaixo. A primeira rotina pode ser obtida em:

https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html

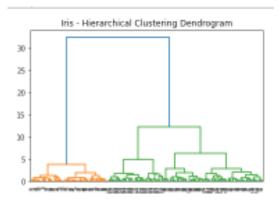
Para a rotina correr corretamente é necessário importar: que já foi importado no exercício 1.

```
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering as aggClus
from scipy.cluster.hierarchy import dendrogram
```

```
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram
 4 5 6 7
            # create the counts of samples under each node
           counts = np.zeros(model.children_.shape[0])
n_samples = len(model.labels_)
for i, merge in enumerate(model.children_):
                  current_count = 0
for child_idx in merge:
    if child_idx < n_samples:
10
                                current_count += 1 # leaf node
11
12
                         else:
                                current_count += counts[child_idx - n_samples]
                  counts[i] = current_count
14
15
           linkage_matrix = np.column_stack(
    [model.children_, model.distances_, counts]
).astype(float)
16
18
19
           # Plot the corresponding dendrogram
dendrogram(linkage_matrix, **kwargs)
20
```

```
# Colocando distance_threshold=0 garante que obtemos a árvore completa.
model = aggClus(distance_threshold = 0, n_clusters = None, linkage = 'ward')
model = model.fit(X.T)
plt.title("Iris - Hierarchical Clustering Dendrogram")
# Visualizar todos os 150 níveis do dendrograma
plot_dendrogram(model, labels = None, truncate_mode = "level", p=150)
```

Gráfico que deve ser obtido com o código:





Ex. 4 O código abaixo aplica as experiências anteriores a um novo conjunto de dados.

Executa o código replicando o código já introduzido e adapta observando os resultados.

```
import numpy as np
from matplotlib import pyplot as plt
import pickle

from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering as aggClus
from scipy.cluster.hierarchy import dendrogram

file = open("/mix9gaus2d.p","rb")

dados = pickle.load(file)

X=dados

kmeans = KMeans(n_clusters=9, init='k-means++', n_init =5, max_iter=500, tol = 0.0001, verbose=0,random_state=42

kmeans = kmeans.fit(X.T)

C=kmeans.cluster_centers_
```

```
#setting distance_threshold=0 ensures we computee the full tree.
model = aggClus(distance_threshold = 0, n_clusters = None, linkage = 'ward')
model = model.fit(X.T)

plt.title("Hierarchical Clustering Dendrogram")
# plot the top 10 levels of the dendrogram
plot_dendrogram(model, labels = None, truncate_mode = "level", p=10)
```

