Analyse de données - TP 4

ISEP – 10 Novembre 2020

Instructions : Préparez un rapport incluant le code source et vos résultats, et déposez-le sur Moodle. Pas plus de 2 personnes par groupe. N'oubliez pas de mettre les 2 noms sur le rapport, ou de faire 2 rendus.

Librairies

Ce TP nécessite les librairies pandas, matplotlib, numpy, scipy et sklearn.

A Analyse des Iris de Fisher avec l'algorithme K-Means

1. Ouvrez les données iris du TP2, ou récupérez les données iris directement dans le module datasets de sklearn.
2. La dernière colonne de vos données représente le label correspondant à l'espèce d'Iris associée. Stockez ces labels dans un vecteur séparé et enlevez-le de votre jeu de données.
3. Utilisez la commande **PCA(·)** de sklearn.decomposition pour réaliser une analyse en composante principale de vos données. Puis, afin de récupérer vos données projetées dans 2 composantes, utilisez les commandes suivantes :

```
pca = PCA(n_components=2)
PCA_val = pca.fit_transform(X)
df_iris_PCA = pd.DataFrame(data = PCA_val,columns = ['PC1', 'PC2'])
df_iris_PCA_class = pd.concat([df_iris_PCA, Y], axis = 1)
```

Vous devriez maintenant avoir 2 jeux de données, le jeux de données original stocké dans une première variable ("*data*" par exemple), et le même jeu de données projeté sur 2 composantes stockée dans la variable *df_iris_PCA*.

4. En utilisant les 2 premières composantes, projetez les labels des 3 espèces d'iris en 2 dimensions. Que pouvez-vous dire sur la séparabilité des classes ? Quelle peuvent être les implications si on utilise un algorithme de clustering sur ces données ?
5. Utilisez l'algorithme K-Means de sklearn.cluster sur vos données originales afin d'obtenir 3 clusters et visualisez les résultats. Pour ce faire, vous pouvez utiliser le code ci-dessous. Expliquez les paramètres de la fonction KMeans.

```
#Kmeans code, change X by your dataset
kmeans = KMeans(n_clusters=3, n_init=5, max_iter=300,random_state=0).↘
    fit(X)
kmeans.score(X)
prediction = kmeans.predict(X)
```

6. Répétez la question 4) plusieurs fois avec des valeurs différentes de random_state. Que constatez-vous ? Expliquez.
7. Affichez la matrice de confusion comparant vos résultats aux labels théoriques.
8. Choisissez le clustering donnant la meilleure matrice de confusion et évaluez le avec l'indice de Silhouette associé à vos données (*silhouette_score()* dans sklearn.metrics). Commentez.

B Clustering hiérarchique

Cet exercise est un tutoriel sur l'utilisation du clustering hiérarchique avec Scipy.

1. Commencez par importer les packages nécessaires.

```
# needed imports
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from scipy.cluster.hierarchy import cophenet, inconsistent, maxRstat
from scipy.spatial.distance import pdist
import numpy as np
```

2. Nous allons maintenant générer un jeu de deonnées aléatoire avec 2 classes à partir du code ci-dessous. Essayez de modifier les paramètres des fonctions pour créer a et b afin de voir à quoi ils correspondent.

```
# np.random.seed(0)  # uncomment for repeatability
a = np.random.multivariate_normal([10, 0], [[3, 1], [1, 4]], size↘
    =[100,])
b = np.random.multivariate_normal([0, 20], [[3, 1], [1, 4]], size↘
    =[50,])
X = np.concatenate((a, b),)
plt.scatter(X[:,0], X[:,1])
plt.title('My data distribution')
plt.show()
```

3. Le code ci-après permet de choisir un linkage (ward dans cet exemple), de générer la matrice du dendrogramme et d'afficher celui-ci. Après avoir fixé la seed aléatoire dans le morceau de code précédent, essayez plusieurs linkages. Commentez.

```
# alternative linkage methods: 'single', 'complete', 'average', '↘
    euclidean' (default), 'cityblock' aka Manhattan, 'hamming', '↘
    cosine'...

Z = linkage(X, 'ward', optimal_ordering=True)


c, coph_dists = cophenet(Z, pdist(X))
print('Cophenetic Correlation: %1.2f' % c)


plt.figure(figsize=(25, 10))
plt.title('Hierarchical Clustering Dendrogram (full)')
plt.xlabel('sample clusters')
plt.ylabel('distance')

dendrogram( Z, leaf_rotation=90., leaf_font_size=8.,)

plt.show()

# truncate the dendrogram for better visibility

R = inconsistent(Z)
```

4. Dans le code ci-dessous, que contient la variable Z. Affichez son contenu et commentez.
5. Cherchez la définition du coefficient cophénétique décrit dans le code précédent. A quoi sert-il et comment l'interpréter ?
6. Comme vous le voyez, le dendrogramme original est massif et difficile à interpréter. Il est possible d'obtenir une version rédutie en utilisant le code ci-après. Dans ce code, expliquez le role des paramètres "truncate_mode" et "p".

```
#display truncated dendrogram
plt.title('Hierarchical Clustering Dendrogram (truncated)')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(
    Z,
    truncate_mode='lastp', #to explain
    p=12,  #to explain
    show_leaf_counts=False,
    leaf_rotation=90.,
    leaf_font_size=12.,
    show_contracted=True,
)
plt.show()
```

7. Enfin, on souhaite maintenant couper le dendrogramme afin d'obtenir les clusters. Deux méthodes possibles sont proposées dans le code ci-après. Expliquez en quoi elles diffèrent, les avantages de l'une ou de l'autre, et ce que font les différents paramètres. Essayez de modifier les paramètres pour voir ce qui change.

```
max_d = 14
clusters = fcluster( Z, max_d, criterion='distance')

plt.figure(figsize=(10, 8))
plt.scatter(X[:,0], X[:,1], c=clusters, cmap='prism')
plt.show()
```

OR

```
k=4
clusters = fcluster(Z,k,criterion='maxclust')

plt.figure(figsize=(10, 8))
plt.scatter(X[:,0], X[:,1], c=clusters, cmap='prism')
plt.show()
```

8. Refaite les questions 3 à 7 avec d'autres linkages et commentez sur les différences de résultats.

C Données Wisconsin Data Breast Cancer

En vous inspirant des exercices précédents, ouvrez les données WDBC, analysez les données et la séparabilité des classes. Faites tourner une ou plusieures méthodes de clustering, et analysez la qualité de vos résultats. N'hésitez pas à faire les analyses classiques des attributs de ces données, et à faire des projections pour visualiser.

D Nombre optimal de clusters sur les données atmosphère d'exoplanète

Dans cet exercice, on va évaluer le nombre de clusters optimal dans un jeu de données artificiel décrivant des compositions atmosphérique d'exoplanètes.

1. Ouvrez le fichier *exo4_ atm_ extr.csv* et retirez la dernière colonne qui contient les labels.
2. Rappelez les propriétés de l'indice de Calinski-Harabasz et celles de l'indice de Davies-Boouldin.
3. Utilisez sklearn.metrics pour déterminer le nombre optimal de clusters pour l'algorithme des K-Means à partir des indices de Davies-Bouldin et Calinski-Harabasz. Expliquez avec vos mots ce que fait cette fonction.
4. Commentez vos résultats. Vous pourrez utiliser des graphes, des projections ACP et les labels pour appuyer vos explications.