TP (projet) Clustering

Envoi du rapport (complet) et code : 24/03/2023 minuit

Date de présentation : 29/03/2023

Les algorithmes de clustering cherchent à apprendre, à partir des propriétés des données, une division optimale ou un étiquetage discret de groupes de points. De nombreux algorithmes de regroupement sont disponibles dans Scikit-Learn et ailleurs, mais le plus simple à comprendre, comme déjà évoqué en cours, est le *k-means*, qui est implémenté dans sklearn.cluster.KMeans.

Le K-means

L'algorithme k-means recherche un nombre prédéterminé de clusters dans un ensemble de données multidimensionnelles non étiquetées. Il y parvient en utilisant une conception simple de ce à quoi ressemble le regroupement optimal :

- Le "centroïde du cluster" est la moyenne arithmétique de tous les points appartenant au cluster.
- Chaque point est plus proche du centroïde de son cluster que de ceux des autres clusters.

Ces deux hypothèses sont à la base du modèle k-means. Nous nous pencherons bientôt sur la manière exacte dont l'algorithme parvient à cette solution, mais pour l'instant, examinons un simple ensemble de données et voyons le résultat du k-means.

Exercice 1

Construction de l’ensemble des données

Afin de tester l’efficacité du k-means, nous allons créer des regroupements cohérents à partir d’un ensemble de données, puis, nous ferons appel à l’algorithme du k-means pour qu’il effectue le clustering de ces mêmes données.

Notre ensemble de données sera donc étiqueté mais nous ne passerons pas à l’algorithme du k-means les étiquettes associées aux données. Elles pourront cependant nous servir pour évaluer la qualité des clusters obtenus. Pour générer un ensemble de données, nous ferons appel à :

sklearn.datasets permet de générer des (“contenus”) blobs gaussiens isotropes pour le regroupement (voir https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html pour plus d’informations)

Q 1

from sklearn.datasets import make_blobs

X, y_true = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0) # crée 4 clusters à partir d'un échantillon de 300 exemples

n_samples : s'il s'agit d'un *entier* alors nous aurons un nombre total de points également distribués dans les clusters, si *array* alors ses éléments correspondent au nombre de points à placer dans chaque groupe

centers : nombre de centroïdes

cluster_std : l'écart-type des clusters.

random_state : si entier représente le *seed* et permet de guarantir la reproducibilité de votre algorithme (n'oubliez pas que les données sont construites de façon aléatoire, par conséquent, si vous voulez reproduire une exécution, il faudra faire en sorte que les même données soient générées lors des prochaines exécutions) ; si instance RandomState, random_state est le générateur de nombres aléatoires ; Si None, le générateur de nombres aléatoires est l'instance RandomState utilisée par np.random. (int, instance RandomState ou None (default=None))

Les paramètres suivants n'ont pas été spécifiés dans l'exemple ci-dessus:

n_features : le nombre de caractéristiques (int, par défaut=2)

center_box : la boîte de délimitation pour chaque centre de grappe (paire de flotteurs (min, max), par défaut=-10.0, 10.0)

shuffle : mélanger les échantillons (booléen, par défault=True)

Renvoie :

- **X :** ndarray of shape (n_samples, n_features) : les échantillons générés
- **y :** ndarray of shape (n_samples,) : les labels (nombres entiers) qui indiquent l'appartenance aux clusters pour chaque membre de l'échantillon.

Le processus ci-dessus, pourra vous être utile si vous avez besoin de générer des groupes de données spécifiques.

Q 2

Pour les affichages :

import matplotlib.pyplot as plt

Vous pouvez utiliser la méthode *scatter* pour afficher les clusters des points ainsi obtenus comme suit :

plt.scatter(X[:, 0], X[:, 1], s=50); # X a deux colonnes car nous laissé le *n_features* avec sa valeur par défaut (2)

s correspond à la taille des points qui seront affichés dans la figure.

Q 3

Vérifier les contenus des valeurs renvoyées, X et y.

Q 4

Compléter l'affichage, grâce à l'application de la fonction show() à l'objet plt, les clusters ainsi formés.

Mise en pratique du k-means

Votre algorithme de k-means devra maintenant trouver (une approximation) des groupes formés ci-dessus. Il fait cela automatiquement, et dans Scikit-Learn la méthode appropriée s'appelle Kmeans() : `from sklearn.cluster import KMeans`

Q 5

Importer la librairie `KMeans`

Q 6

Construire, à partir du jeu de données X obtenu à la Q 1, et en utilisant la méthode .fit() le modèle qui vous permettra d'obtenir les 4 clusters.

Q 7

Quelles sont les classes predictes pour les éléments de X d'après le modèle que vous venez de construire ? Pour répondre à cette question, appliquez la méthode predict() au modèle obtenu ci-dessus et affectez le résultat à la variable `y_kmeans`.

Q 8

Quelle est la classe prédicte pour le deuxième élément de X ? Quelle sa vraie classe ? Que constatez vous ?

Q 9

Quelle est la classe prédicte pour cinquième élément de X ? Quelle sa vraie classe ? Que constatez vous ?

Q 10

Afficher les classes de tous les éléments de X.

Q 11

Afficher les classes predictes pour tous les éléments de X.

Exercice 2

Les exemples du jeu de données “**wine.csv**” sont le résultat d'une analyse chimique de vins cultivés dans une même région en Italie mais provenant de trois cultivateurs différents. L'analyse a permis de déterminer les quantités de 13 composants présents dans chacun des trois types de vins analysés.

Les attributs sont (selon Riccardo Leardi), sont :
1) Alcohol
2) Malic acid
3) Ash : dosage nécessaire pour que le vin soit certifié.
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline

Nous avons aussi la classe ou label (attribut “Class”) à laquelle le vin appartient. Ce jeu de données est adapté pour faire de la classification (apprentissage supervisé). Cependant, nous allons faire le nécessaire pour qu’il puisse être utilisé pour faire du clustering (apprentissage non supervisé).

Premisses : si on suppose que tous les vins sont adaptés à la consomation, la variable “Ash” n’est pas utile. De même, puisqu’il s’agit d’une méthode d’apprenstissage non supervisé, nous allons enlever la classe des vins pour obtenir ainsi l’ensemble des données avec leur caractéristiques mais sans les labels.

Commencons par analyser les données ...

Analyse du jeu de données

Q 1

Importer “Pandas” : import pandas as pd

Q 2

Lire le fichier wine.csv et l’affecter à la variable df

Q3

Afficher les 10 premiers éléments du jeu de données

Q 4

Afficher les 10 derniers éléments du jeu de données

Q 5

Afficher toutes les informations concernant le jeu de données (méthode info())

Q 6

Jetez un coup d'oeuil au résultat. Y-a-t-il des valeurs nulles ?

Q 7

Utiliser les méthodes isnull() et sum() pour répondre à la question Q 6.

```
df.isnull().sum()
```

Q 8

Quel sont les types des variables dans le jeu de données ?

Q 9

Quelles sont les labels du dataframe ? Vous pouvez utiliser la méthode unique()

Reduction de la dimension des données

Contrairement au jeu de données que nous avons construit lors de l'Exercice 1, les données ici sont représentées grâce à 13 attributs. Il serait donc impossible de les représenter dans un repère à deux dimensions. Par conséquent, nous allons devoir utiliser une technique de réduction de dimension pour pouvoir les représenter sur deux dimensions. Nous allons utiliser la PCA (Principal Component Analysis) à cet effet. Voici les importations que vous devez effectuer :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

Q 10

Numpy Array est très utile pour exécuter certaines fonctions mathématiques de haut niveau prises en charge par le paquet Numpy. Pour profiter de ces fonctionalités, nous allons convertir notre dataframe en Numpy Array.
Appliquez la méthode *to_numpy()* au dataframe df. Ceci renvoi un objet de type Numpy ndarray. Affecter ce résultat à une variable que vous allez nommer *donnees*.

Q 11

Créer un object nommé pca de type PCA. Il sera ensuite utilisé pour transformer vos données en dimension 2 (méthode fit_transform()) :

Nous pouvons maintenant construire nos clusters.

Q 12

Importez la librairie Kmeans : from sklearn.cluster import KMeans

Q 13

Instanciez un objet de type kmeans avec 3 clusters et l'affectez à la variable *kmeans*

Q 14

Utiliser la méthode *fit_predict()* pour obtenir les labels predits par le kmeans pour les différents enregistrements du jeu de données. Affectez les à la variable *classe*.

Q 15

Quels sont les valeurs des classes ? Vous pouvez utiliser la méthode unique() pour répondre à cette question et affecter le résultat à la variable *labels_uniques*.

Avant de passer à l'affichage des résultats, essayons de comprendre les fonctionnalités des méthodes d'affichage. La plus simple, est la méthode *scatter()*. Elle sera la première à être utilisée pour représenter les nuages de points.

La méthode scatter() (Scatter Plot)

Un nuage de points est une figure dans laquelle chaque valeur de l'ensemble de données est représentée par un point (x,y).

Le module Matplotlib dispose d'une méthode pour dessiner des nuages de points. Pour cela, il a besoin de deux vecteurs de **même longueur**, l'un pour les valeurs de l'axe x, et l'autre pour les valeurs de l'axe y : objet.scatter(x, y).

Le fichier World_happiness_dataset_2019.csv contient des informations sur plusieurs pays, comme les données sur les aides sociales, la perception de corruption, PIB ("GDP per capita"), etc. mais aussi un *score* qui reflète l'*impression d'être heureux*.

QS1

Effectuez les importations suivantes :

```
import matplotlib.pyplot as plt
import pandas as pd
```

QS 2

Lire l'ensemble des données contenues dans le fichier World_happiness_dataset_2019.csv et affecter le résultat à la variable df (`pd.read_csv`).

QS 3

Appliquer la méthode *scatter()* à l'objet plt, pour construire le nuage des points correspondant à la relation entre la richesse d'un pays "Economy (GDP per Capita)", et l'impression d'être heureux.

Vous pouvez aussi créer d'autres nuages de points pour d'autres couples d'attributs.

QS 4
Il est aussi possible de créer des nuages de points avec une librairie (plus générale) seaborn. Il y a plusieurs façons de dessiner un nuage de points dans seaborn. La plus simple, qui doit être utilisée lorsque les deux variables sont numériques, est la fonction *scatterplot()*.

Faire, en plus l'importation suivante :
import seaborn as sns

QS 5
Exécuter l'instruction suivante :

```
sns.scatterplot(data = df, x = "Economy (GDP per Capita)", y = "Happiness Score")
```

QS 6
Il est aussi possible de visualiser les données par catégorie. Exécuter l'instruction suivante :

```
sns.catplot(x="Economy (GDP per Capita)", y="Happiness Score", data=df)
```

--
Retour à l'Exercice 2.

Notre but maintenant est de construire les deux vecteurs à passer à la méthode *scatter()* pour pouvoir afficher notre nuage des mots. Il s'agit, dans ce cas, d'extraire les coordonnées x et y des points (les éléments de chaque classe prédicte). Les valeurs de x correspondront ainsi à celles de la première colonne du dataframe alors que celles de y correspondeont à la deuxième colonne du dataframe.

Q 16
Ecrire une boucle sur les différentes valeurs de labels_unique, qui permettra d'afficher les nuages des points dans df correspondants à tous les labels.

Q 17
Afficher le résultat grâce à la méthode *show()*.

Projet associé à la séance

Tout travail démontrant votre esprit d'initiative sera pris en compte.

Partie A : il s'agit, dans un premier temps, de construire des clusters (à vous de choisir le nombre de clusters, pensez à justifier votre choix) correspondant au jeu de données digits que vous pouvez récupérer grâce à l'instruction load_digits().

TRAVAIL DEMANDÉ :

1. analyse et description du jeu de données (toutes les informations sur le jeu données).

2. votre programme devra contenir toutes les étapes nécessaires pour la construction des clusters.

3. **toutes les étapes doivent être systématiquement décrites, illustrées et justifiées.**

Partie B : allez sur le site https://archive.ics.uci.edu/ml/datasets.php et choisissez un jeu de données adapté à l'apprentissage non supervisé. Vous devez me communiquer votre choix dès que vous aurez choisi votre jeu de données. Réalisez le même travail que celui effectué pour le jeu de données *digit*.

Partie C : mise à jour du rapport (10 pages plus annexe).