TP de classification non supervisée

17 octobre 2021

Résumé

L'objectif du TP est de réaliser un clustering sur des données de façon à regrouper automatiquement les données qui se ressemblent d'un point de vue caractéristiques et à dégager plusieurs catégories. Les deux méthodes à implémenter et tester sont le clustering ascendant hiérarchique et l'algorithme K-means.

1 Lecture des données

Le jeu de données est stocké dans le fichier `hotels.csv` disponible sur MyLearningSpace. Commencez par visualiser le fichier. Il regroupe des informations sur un ensemble d'hôtels : pays, étoiles, confort, nombre de chambres, qualité niveau cuisine, sport et plage, prix (par nuit en euro).

Importez le fichier en mémoire à l'aide de la fonction `read_csv()` de `pandas` pour pouvoir manipuler les données. Pour rappel, la fonction `read_csv()` renvoie un objet de la classe `DataFrame`. Assurez-vous que l'importation est correcte en utilisant les fonctionnalités offertes pour les objets `DataFrame`, notamment l'affichage des informations sur les variables, ainsi que l'affichage des premières lignes : `info()`, `describe()`, `shape`, `head()`...

Quelles variables proposez-vous de conserver pour effectuer le clustering ?

Stockez les colonnes NOM et ETOILE à part de façon à conserver les noms et les étoiles des différents hôtels.

Eliminez les colonnes qui ne seront pas utilisées avec la méthode `drop()` de la classe `DataFrame`.

Stockez les noms des colonnes restantes de façon à conserver les noms des variables utilisées pour le clustering avec l'attribut `columns` de la classe `DataFrame`.

2 Examen et transformation des données

Après avoir importé et sélectionné les données pour la classification, vous pouvez faire une première analyse des données en recherchant les relations qui existent entre les variables.

Calculez le coefficient de corrélation entre chaque couple de variables numériques en utilisant la méthode `corr()` de la classe `DataFrame`.

Il est aussi possible de visualiser graphiquement les corrélations entre variables grâce à la fonction `scatter_matrix()` de `pandas`. Cette fonction croise deux à deux les variables numériques et affiche les nuages de points correspondants sur un plan.

Quelles sont les variables les plus corrélées positivement ? négativement ? Quelles sont les variables les moins corrélées ?

Les variables étant très hétérogènes, il faut réduire et centrer les données avant d'effectuer le clustering. Il existe différentes façons de réaliser ces opérations. La plus simple est la suivante :

Commencez par convertir l'objet `DataFrame` en tableau de type `array` en utilisant la méthode `to_numpy()` de la librairie `numpy`.

Ensuite pour centrer et réduire les données, utilisez les fonctionnalités de la classe `StandardScaler` de `sklearn.preprocessing`. Effectuez la normalisation des données avec les méthodes `fit()` et `transform()`.

Vérifiez que la moyenne et la variance des variables centrées réduites valent bien 0 et 1 respectivement.

3 Classification ascendante hiérarchique (CAH)

Pour mettre en oeuvre la classification ascendante hiérarchique en Python, on utilise le module `scipy.cluster.hierarchy` et principalement les fonctions `dendrogram`, `linkage` et `fcluster`. Etudiez ces fonctions.

Pour cette étude, on considère la distance euclidienne entre points et différentes méthodes de calcul de distance entre clusters : single, complete, average, centroid.

Pour chacune de ces méthodes :

— Effectuez la classification ascendante hiérarchique complète à l'aide de la fonction `linkage` (paramètre optimal_ordering=True pour obtenir une

meilleure visualisation des résultats). Cette fonction retourne une matrice de liaison.
— Représentez cette matrice de liaison sous forme de dendrogramme à l'aide de la fonction `dendrogram` (paramètre color_threshold=0 pour obtenir le dendrogramme complet). Analysez visuellement le résultat obtenu. On rappelle que le dendrogramme représente non seulement les liaisons entre les classes mais aussi la distance entre les classes fusionnées via la hauteur des branches.

En déduire les distances entre clusters qui permettent d'obtenir les meilleures partitions (clusters homogènes et bien séparés).

Pour les distances entre clusters retenues, proposez (toujours visuellement) une ou deux valeurs pertinentes pour le nombre K de clusters. Déterminez les seuils t qui permettent de couper l'arbre aux niveaux requis pour obtenir ces valeurs de K. Affichez les classes en couleur sur le dendrogramme en exécutant à nouveau la fonction `dendrogram` avec color_threshold=t.

Effectuez le clustering proprement dit en utilisant la fonction `fcluster` paramétrée par les seuils t requis pour obtenir les nombres de clusters K voulus et le critère de distance (criterion='distance'). Affichez le résultat de la fonction qui est un vecteur contenant pour chaque donnée le label (ou numéro) du cluster qui lui a été affecté.

Pour évaluer la qualité des partitions obtenues, calculez le coefficient de silhouette en utilisant la fonction `silhouette.score` du module `sklearn.metrics`. Comparez les résultats obtenus et déterminez la meilleure partition (par rapport à cette métrique).

Pour cette partition de meilleure qualité, listez les noms et étoiles des hôtels de chaque cluster. Pour récupérer les indices des hôtels affectés aux clusters, le plus simple est d'utiliser la fonction `argsort` de `numpy` : les indices retournés correspondent aux labels ordonnés du plus petit au plus grand.

Analysez les clusters obtenus au regard des données du fichier.

4 Méthode des K-means

Pour mettre en oeuvre la méthode des K-means en Python, on utilise la classe `KMeans` du module `sklearn.cluster`.

Etudiez cette classe :
— paramètres d'appel, en particulier `n_clusters`, `init` et `n_init`,
— attributs, en particulier `labels_`, `inertia_` et `cluster_centers_`,
— méthodes, en particulier `fit()`.

Appliquez la méthode des K-means pour un nombre de clusters K que vous choisirez, avec d'abord une initialisation aléatoire (init = "random") et une seule exécution (n_init = 1). Affichez le résultat de l'algorithme : labels (ou numéros) des clusters affectés à chaque donnée grâce à l'attribut `labels_` et inertie de la partition obtenue grâce à l'attribut `inertia_`.

Exécutez l'algorithme une 2ème fois, affichez le résultat obtenu et comparez avec la partition précédente en calculant l'indice de Rand ajusté (ARI) grâce à la fonction `adjusted_rand_score` du module `sklearn.metrics`.

Appliquez ensuite l'algorithme avec toujours une initialisation aléatoire mais plusieurs exécutions (n_init = 10), puis en utilisant la méthode d'initialisation k-means++ (et n_init = 10). Pour chaque paramétrage, comparez les résultats obtenus pour 2 exécutions successives en calculant l'indice de Rand ajusté. Commentez la stabilité de l'algorithme K-means.

Pour déterminer expérimentalement la valeur de K "optimale", appliquez l'algorithme des K-means pour K variant de 2 à 10 avec des paramètres que vous choisirez d'après les résultats précédents et représentez l'évolution du coefficient de silhouette en fonction du nombre de clusters. En déduire le nombre de classes "optimal".

Pour la valeur de K choisie, listez les noms et étoiles des hôtels de chaque classe.

Pour visualiser graphiquement les clusters, il faut projeter les données sur un plan. Effectuez une ACP et représentez les points obtenus sur le premier plan principal défini par les axes 1 et 2. Identifiez les clusters en utilisant une couleur différente pour représenter les points qu'ils regroupent. Interprétez les résultats.

Vérifiez s'il est possible ou non de classer automatiquement les hôtels en fonction de leurs étoiles. Pour cela, appliquez la méthode des K-means pour un nombre de clusters $K = 6$, puisqu'il existe 6 catégories d'étoiles (0 à 5) et vérifiez le lien entre les étoiles des hôtels et les clusters auxquels ils sont affectés. Pour comparer le résultat de clustering (donné par l'attribut `labels_`) et la classification en étoiles (donnée par la colonne ETOILE), vous pouvez utiliser l'indice de Rand ajusté.