Université Hassiba Benbouali de Chlef
Faculté des Sciences Exactes et Informatique
Département de l'Informatique

La Méthode K-means

Préparé par

BELBACHIR Moundir-Oussama
ALLOUACHE Imadeddine

Décembre 12, 2023

Contents

1. Introduction

Dans le domaine de la science des données, on explore comment tirer des idées importantes à partir de grandes quantités de données. L'analyse de données est une méthode qui nous aide à comprendre des schémas et des tendances dans ces données. Dans ce projet, on va se pencher sur le regroupement, une façon de mettre ensemble des données similaires. On va se concentrer sur une méthode appelée K-means, et on va même essayer de l'utiliser pour comprendre les groupes de clients. On va faire cela en utilisant un langage de programmation Python.

2. Définition

1. Clustering :

Clustering (ou partitionnement des données) : Cette méthode de classification non supervisée rassemble un ensemble d'algorithmes d'apprentissage dont le but est de regrouper entre elles des données non étiquetées présentant des propriétés similaires. Isoler ainsi des schémas ou des familles permet aussi de préparer le terrain pour l'application ultérieure d'algorithmes d'apprentissage supervisé (comme le KNN). [1]

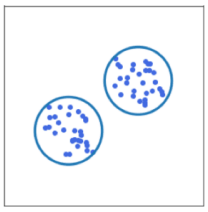


Figure 1 : K-Means simplifie le regroupement des données en deux cercles

2. K-Means :

K-means est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper en K clusters distincts les observations du data set. Ainsi les données similaires se retrouveront dans un même cluster. Par ailleurs, une observation ne peut se retrouver que dans un cluster à la fois (exclusivité d'appartenance). Une même observation, ne pourra donc, appartenir à deux clusters différents. [2]

3. Découverte de K-Means:

1. Comprendre les Similarités dans le Regroupement K-Means :

Pour organiser des données en différents groupes avec l'algorithme K-Means, on doit voir à quel point les observations se ressemblent. Si deux choses se ressemblent, leur distance est petite, mais si elles sont différentes, la distance est grande. Cela nous aide à décider dans quel groupe mettre chaque chose.

Pour le regroupement, nous utilisons des méthodes mathématiques pour mesurer la distance, parmi lesquelles

- **La distance Euclidienne**

  d (M1,M2) = √ ((x 2 − x 1) ^ 2 + (y 2 − y 1) ^2)

- **La distance de Manhattan**

  d (A, B) = |XB - XA| + |YB - YA|

2. *Choisir K : le nombre de clusters :*

Il existe deux méthodes empiriques qui nous permettent de déterminer une valeur de k optimale lorsque l'on ne sait pas exactement combien de clusters il existerait dans le jeu de données.

- **Méthode de coude**
- **Méthode de silhouette**

3. L'algorithme K-Means :

Entrée *:*

K le nombre de cluster à former

Le Training Set (matrice de données)

DEBUT

Choisir aléatoirement K points (une ligne de la matrice de données). Ces points sont les centres des clusters (nommé centroïde).

REPETER

Affecter chaque point (élément de la matrice de donnée) au groupe dont il est le plus proche au son centre

Recalculer le centre de chaque cluster et modifier le centroïde

JUSQU'A *CONVERGENCE*

OU *(stabilisation de l'inertie totale de la population)*

FIN ALGORITHME

4. Scénarios d'utilisation de l'algorithme K-means :

- **Segmentation de marché** : K-means groupe les consommateurs selon leurs préférences, facilitant la personnalisation du marketing.
- **Classification d'images** : Il classe les images en fonction de caractéristiques, simplifiant la catégorisation.
- **Détection d'anomalies** : Identifie les comportements suspects, utile pour la prévention de fraudes en ligne.
- **Optimisation de la chaîne d'approvisionnement** : Regroupe les données logistiques pour optimiser la gestion des stocks.
- **Recommandation de produits** : Analyse les historiques d'achats pour suggérer des produits similaires aux utilisateurs.
- **Analyse de sentiment** : Regroupe les avis en fonction de la tonalité émotionnelle, facilitant l'analyse des opinions.

4. Mise en œuvre :

Dans notre projet, nous allons appliquer l'algorithme K-means pour résoudre le défi de segmentation des clients.

1. Les bibliothèques utilisées :

```
1  import numpy as np  #Linear ALgebra
2  import pandas as pd #data processing , CSV file I/O
3  import matplotlib.pyplot as plt #visio
4  import seaborn as sb  #visio
5  from sklearn.cluster import KMeans #Kmeans algorithm
```

- **numpy** : Bibliothèque pour les opérations algébriques linéaires en Python.
- **Pandas** : Outil pour la manipulation et l'analyse de données tabulaires, notamment pour les fichiers CSV.
- **matplotlib.pyplot** : Librairie de visualisation permettant de créer des graphiques et des visualisations.
- **Seaborn** : Bibliothèque de visualisation de données basée sur Matplotlib, offrant une interface plus esthétique.
- **sklearn.cluster ,KMeans**: fait référence à une classe dans la bibliothèque scikit-learn, plus précisément dans le module de regroupement (cluster).

2. Ensemble de données utilisées :

```
1  #Loading the data from csv file to pandas DataFrame
2  custemer_data = pd.read_csv('Mall_Customers.csv')
3  #first 5 rows in the dataframe
4  custemer_data.head()
```

L'ensemble de données clients contient probablement des informations telles que l'identifiant du client, le genre, l'âge, le revenu annuel et le score de dépenses, qui peuvent être utilisées pour la segmentation et l'analyse.

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

3. Vue d'ensemble des données:

```
#getting some information about the dataset
custemer_data.info()
# finding the number of rows and columns
custemer_data.shape
```

Nous recueillons des informations sur l'ensemble de données, notamment en déterminant le nombre de lignes et de colonnes

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB

(200, 5)
```

Évidemment, c'est une taille modeste pour un DataFrame avec 200 entrées et 5 colonnes. Aucune valeur NaN ni objet anormal n'est présent, ce qui simplifie le nettoyage des données et les prétraitements.

1. *Le choix des colonnes de revenu et de score :*

Il est facile de constater que le Revenu Annuel (k$) et le Score de Dépenses (1-100) sont les caractéristiques clés pour analyser et regrouper les clients.

```
1  X = custemer_data.iloc[:,[3,4]].values
2  print(X)
```

La première colonne représente le revenu annuel et la deuxième colonne représente le score de dépenses. Ce sont ces deux valeurs que nous allons utiliser dans notre regroupement

2. *Regroupement par K-means :*

Les tâches principales de KMeans sont les suivantes :

1. Extraire un nombre k d'échantillons comme centroïde initial de manière aléatoire.
2. Commencer une boucle.
3. Attribuer chaque point d'échantillon au centroïde le plus proche d'eux, générant ainsi k clusters.
4. Pour chaque cluster, calculer la moyenne de tous les points d'échantillon assignés au cluster comme nouveau centroïde.
5. Lorsque la position du centroïde ne change plus, l'itération s'arrête et le regroupement est terminé

1. Choix du nombre de clusters :

Nous ne connaissons pas le nombre optimal et correct de clusters. Pour cela, nous allons utiliser un paramètre appelé WCSS (Within-Cluster Sum of Squares), qui mesure la somme des carrés des distances entre chaque point de données et le centre de ces clusters. Lorsque nous essayons de trouver le meilleur nombre K, le WCSS doit être le plus bas possible.

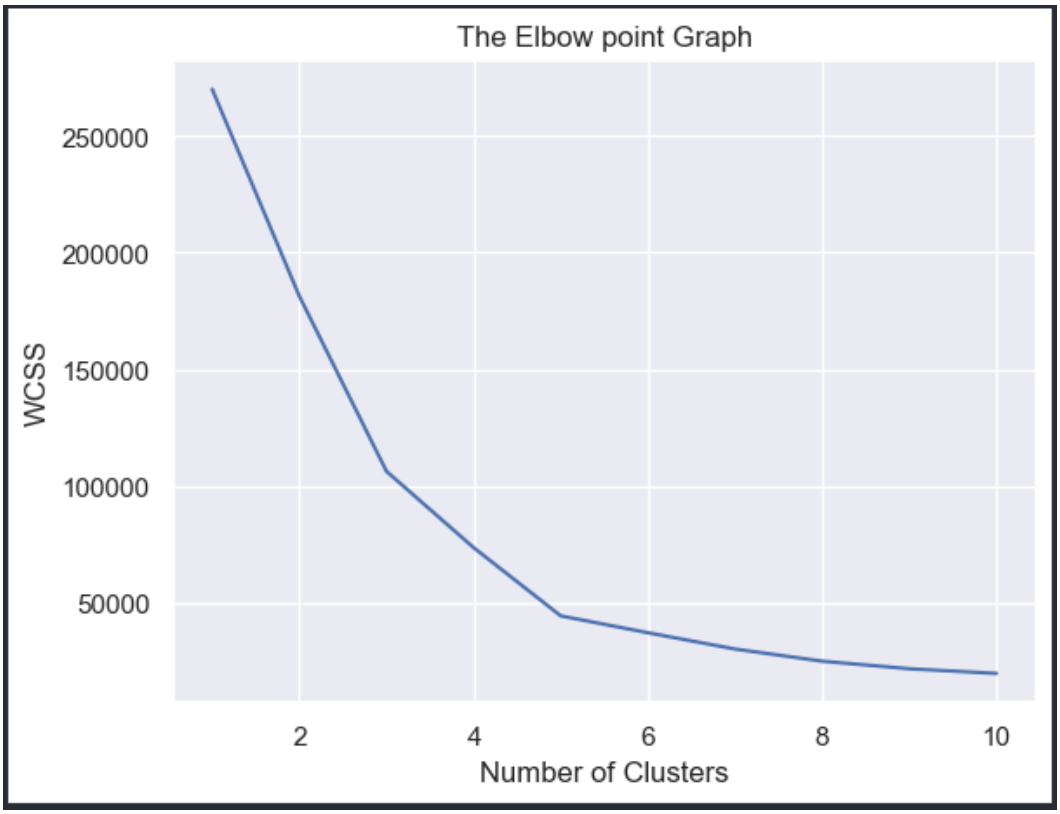
Nous allons utiliser **La Méthode de coude :**

Nous essayons de trouver la valeur de WCSS pour différents nombres de clusters.

```
#finding the wcss value for difrent number of clusters

wcss = [] #empty list

 # 1 to 10 loop
for i in range(1,11):

  #variable kmeans & k-means++ model
  kmeans = KMeans(n_clusters=i,init='k-means++',random_state=65,n_init=10)
  kmeans.fit(X)

  #will give us wcss values for each clusters and the value will be stored in the list
  wcss.append(kmeans.inertia_)
```

```
#plot the elbow graph

sb.set()
plt.plot(range(1,11),wcss)
plt.title('The Elbow point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

Après avoir calculé le WCSS pour chaque cluster et tracé le graphique, voici le résultat.



Il y a une baisse aux points 3 et 5, ce sont donc des points de coude. Nous allons choisir 5 car il n'y a pas d'autre baisse de la valeur après cela. Le nombre optimal de clusters sera donc 5.

2. Le traitement de k-means model :

```python
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0, n_init=10)

# Fit the model and get labels and cluster centers
Y = kmeans.fit_predict(X)
centroids = kmeans.cluster_centers_ #clusters coordinates
final_iter = kmeans.n_iter_ #final iteration
# Print the results
print("Labels:\n", Y)
print("\n")
print("Final Cluster Centers:\n", centroids)
print("\n")
print("Number of iterations to converge:", final_iter)
```

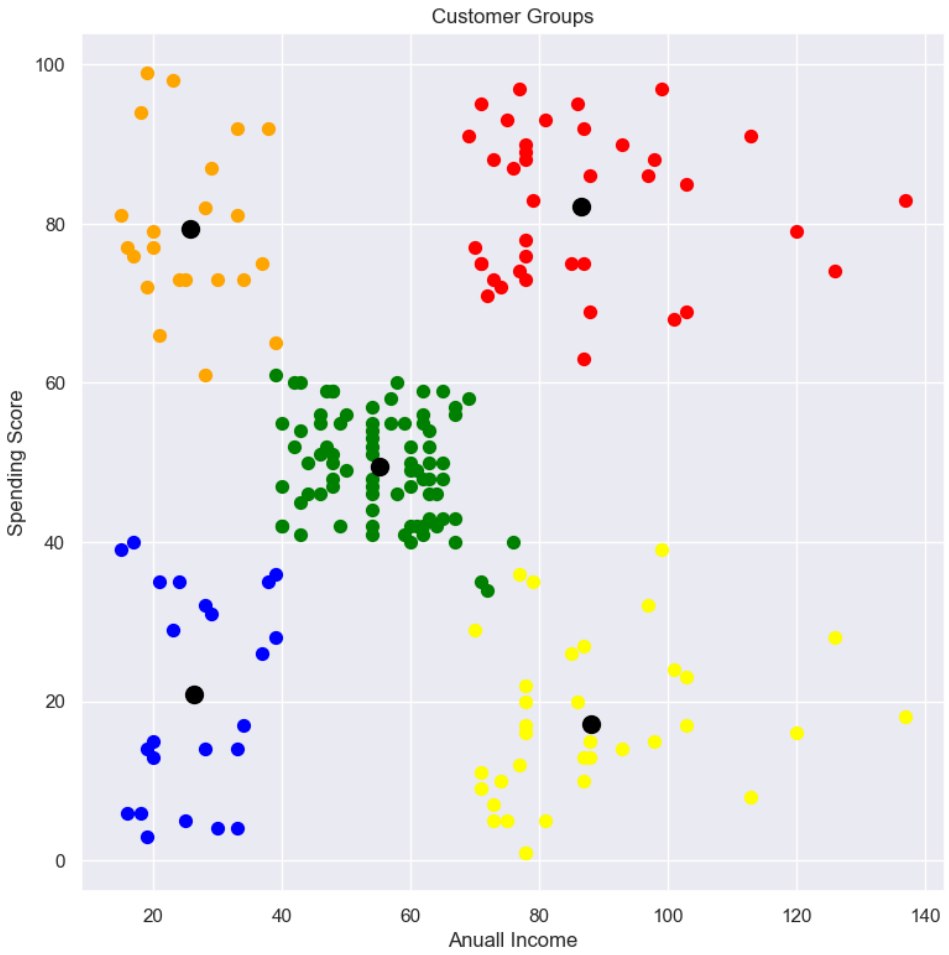
```
1-Labels:
 [3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3
 4 3 4 3 4 3 0 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 0 1 2 1 2 1 0 1 2 1 2 1 2 1 2 1 0 1 2 1 2 1
 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1]
2-Final Cluster Centers:
 [[55.2962963  49.51851852]
 [86.53846154 82.12820513]
 [88.2        17.11428571]
 [26.30434783 20.91304348]
 [25.72727273 79.36363636]]
3-Number of iterations to converge: 6
```

Il y a 5 clusters : 0, 1, 2, 3, 4.

3. Visualisation de tous les clusters :

```python
# plotting all the clusters and their centroids
plt.figure(figsize=(9,9))
plt.scatter(X[Y==0,0], X[Y==0,1], s=50, c='green', label='Cluster 1')
plt.scatter(X[Y==1,0], X[Y==1,1], s=50, c='red', label='Cluster 2')
plt.scatter(X[Y==2,0], X[Y==2,1], s=50, c='yellow', label='Cluster 3')
plt.scatter(X[Y==3,0], X[Y==3,1], s=50, c='blue', label='Cluster 4')
plt.scatter(X[Y==4,0], X[Y==4,1], s=50, c='orange', label='Cluster 5')
#plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=100,
c='black',label='Centroids')
plt.title('Customer Groups')
plt.xlabel('Anuall Income')
plt.ylabel('Spending Score')
plt.show()
```

The output:



Tous les clusters sont bien répartis, quelques points de données sont proches les uns des autres. Notre ensemble de données représente cinq groupes distincts de clients.

Conclusion :

En conclusion, l'algorithme K-means se révèle être un outil puissant pour la segmentation client, facilitant l'identification et la compréhension de groupes distincts de clients en fonction de leurs caractéristiques. Sa simplicité et sa facilité de mise en œuvre Bien que le K-means ait ses limites, comme sa sensibilité aux centroïdes initiaux, ses avantages en termes de simplicité, d'efficacité et de polyvalence en font l'un des meilleurs algorithmes pour les tâches de segmentation.

Bibliographie

[1] 2023. [Online]. Available: https://dataanalyticspost.com/Lexique/clustering/.

[2] 2023. [Online]. Available: https://mrmint.fr/algorithme-k-means.