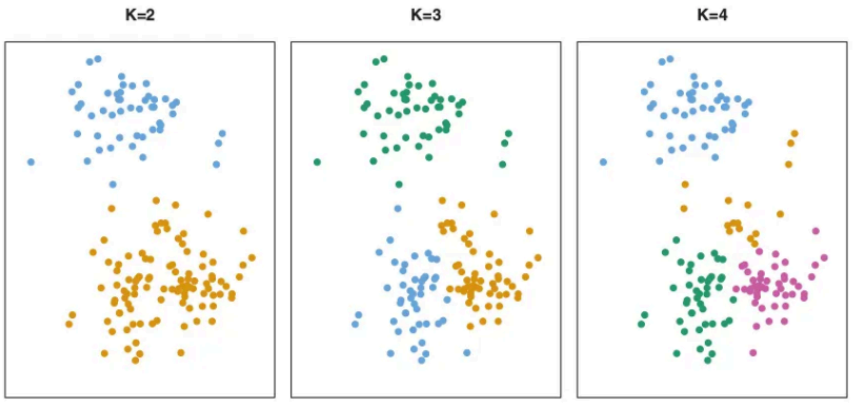
RAPPORT TP FOUILLE DE DONNÉES
L'ALGORITHME K-MEANS



Réalisé par :

- TAREB Selma 191931089198
- ADLAOUI Anis 181831052692

Section :
M1 IV groupe 01

Année universitaire :
2023/2024

INTRODUCTION

L'algorithme K-Means est l'un des algorithmes de clustering les plus largement utilisés en analyse de données et en apprentissage automatique. Développé par James MacQueen en 1967, il est populaire en raison de sa simplicité, de sa rapidité d'exécution et de sa facilité d'interprétation des résultats. Le K-Means est largement utilisé dans divers domaines tels que le marketing, la bioinformatique, la reconnaissance de formes, etc., pour découvrir des structures sous-jacentes dans les données non étiquetées.

L'objectif du clustering est de diviser un ensemble de données en groupes homogènes appelés clusters, de telle sorte que les points de données au sein d'un même cluster soient similaires les uns aux autres, tandis que les points de données dans des clusters différents sont différents. L'algorithme K-Means vise à minimiser la variance intra-cluster et à maximiser la variance inter-cluster en déplaçant itérativement les centroids des clusters et en assignant les points de données aux clusters les plus proches.

Dans ce rapport, nous examinerons en détail l'algorithme K-Means, ses principes de fonctionnement, ainsi que ses avantages et limites. Nous présenterons également une méthodologie pour varier le paramètre K, le nombre de clusters, et comparer les résultats obtenus dans le contexte de différentes applications. Enfin, nous illustrerons l'utilisation de l'algorithme K-Means à travers des exemples concrets et des cas d'utilisation réels.

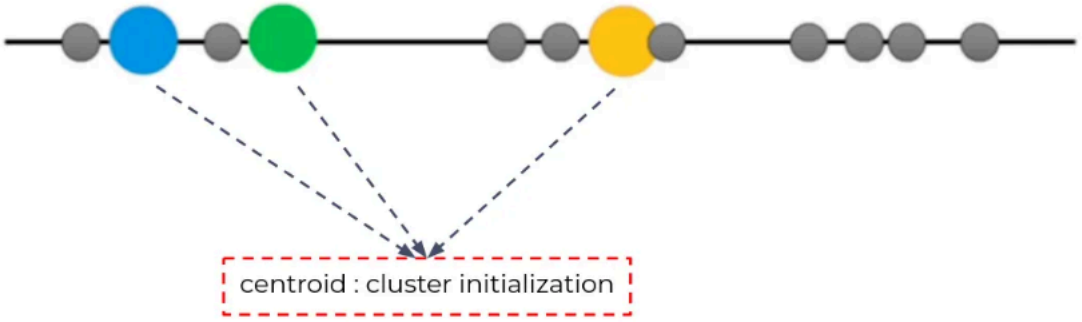
L’algorithme K-means

Définition :
K-Means est un type d'algorithme de partitionnement en grappes qui appartient à la catégorie de l'apprentissage non supervisé. Son objectif principal est de diviser un ensemble de données en K grappes, chaque grappe représentant un groupe de points de données présentant des similitudes. Le "K" de K-Means fait référence au nombre de grappes que vous souhaitez créer, et il s'agit d'un hyperparamètre que vous devez spécifier avant d'exécuter l'algorithme.

Comment fonctionne K-Means ?
Il y a plusieurs étapes pour l'expliquer. Par exemple, les données suivantes sont fournies :

Étape 1. Déterminer la valeur "K", la valeur "K" représente le nombre de Clusters.
dans ce cas, nous choisirons K=3. En d'autres termes, nous voulons identifier 3 clusters. Existe-t-il un moyen de déterminer la valeur de K ? Oui, mais nous en reparlerons plus tard.

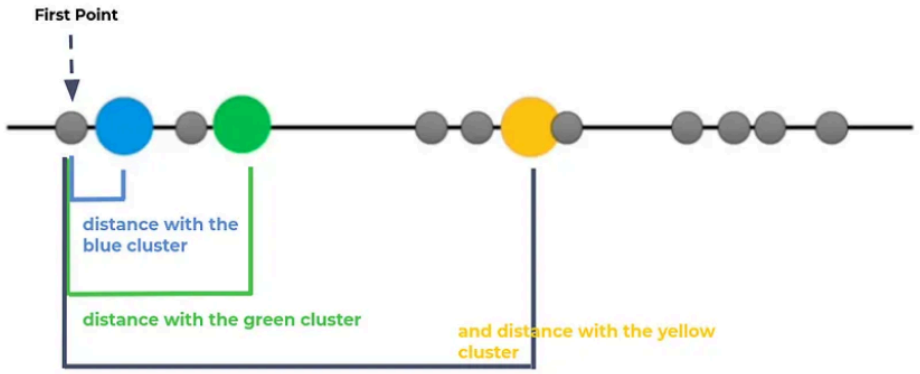
Étape 2. Sélection aléatoire de 3 centroïdes distincts (nouveaux points de données pour l'initialisation de la cluster).
Par exemple - tentatives 1. "K" est égal à 3 donc il y a 3 centroïdes, dans ce cas il s'agit de l'initialisation du cluster



Étape 3. Mesurer la distance (distance euclidienne) entre chaque point et le centroïde

par exemple, mesurer la distance entre le premier point et le centroïde.

$$d = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$



Étape 4. Affectation de chaque point à la grappe la plus proche

par exemple, mesurer la distance entre le premier point et le centroïde.

Étape 5. Calculer la moyenne de chaque cluster comme nouveau centroïde

Mise à jour du centroïde avec la moyenne de chaque grappe

Étape 6. Répétez les étapes 3 à 5 avec le nouveau centre de la grappe.

Répéter jusqu'à l'arrêt :

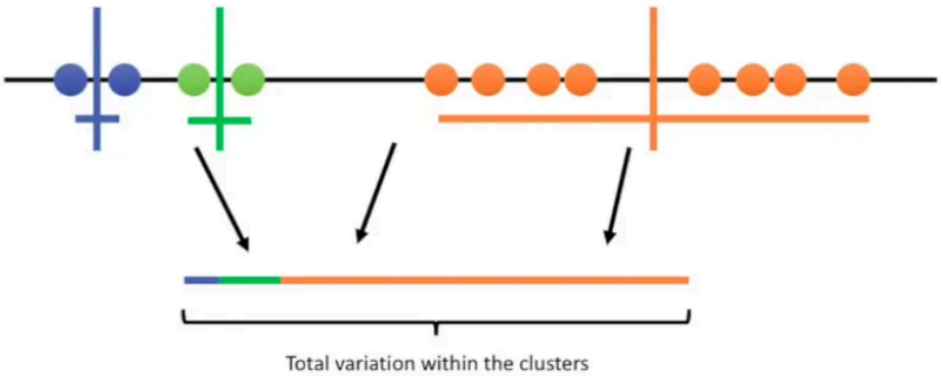
- Convergence. (Pas d'autres changements)
- Nombre maximal d'itérations.

Si le regroupement ne change pas du tout au cours de la dernière itération, on s'arrête.

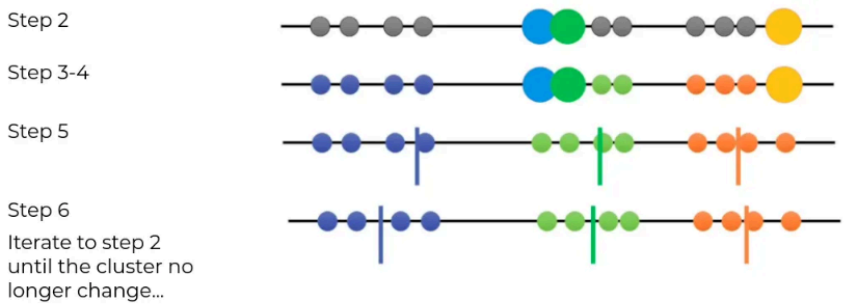
Ce processus est-il achevé ? Bien sûr que non !
l'algorithme K-means vise à choisir le centroïde qui minimise l'inertie, ou le critère de la somme des carrés à l'intérieur d'un groupe.

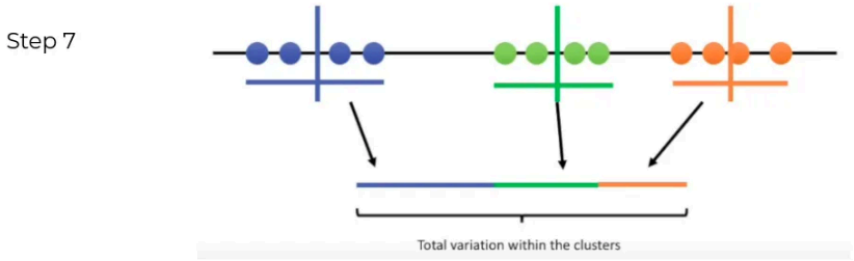
Méthode du coude (Elbow method) : Tracez la variance expliquée par le modèle en fonction du nombre de clusters. Le coude du graphique représente le point où l'ajout d'un cluster supplémentaire ne donne plus une réduction significative de la variance. Cela peut être considéré comme un bon choix pour K.

Étape 7. Calculer la variance de chaque grappe
Étant donné que le regroupement K-Means ne peut pas "voir" le meilleur regroupement, sa seule option est de garder une trace de ces regroupements et de leur variance totale, et de recommencer l'opération avec d'autres points de départ.



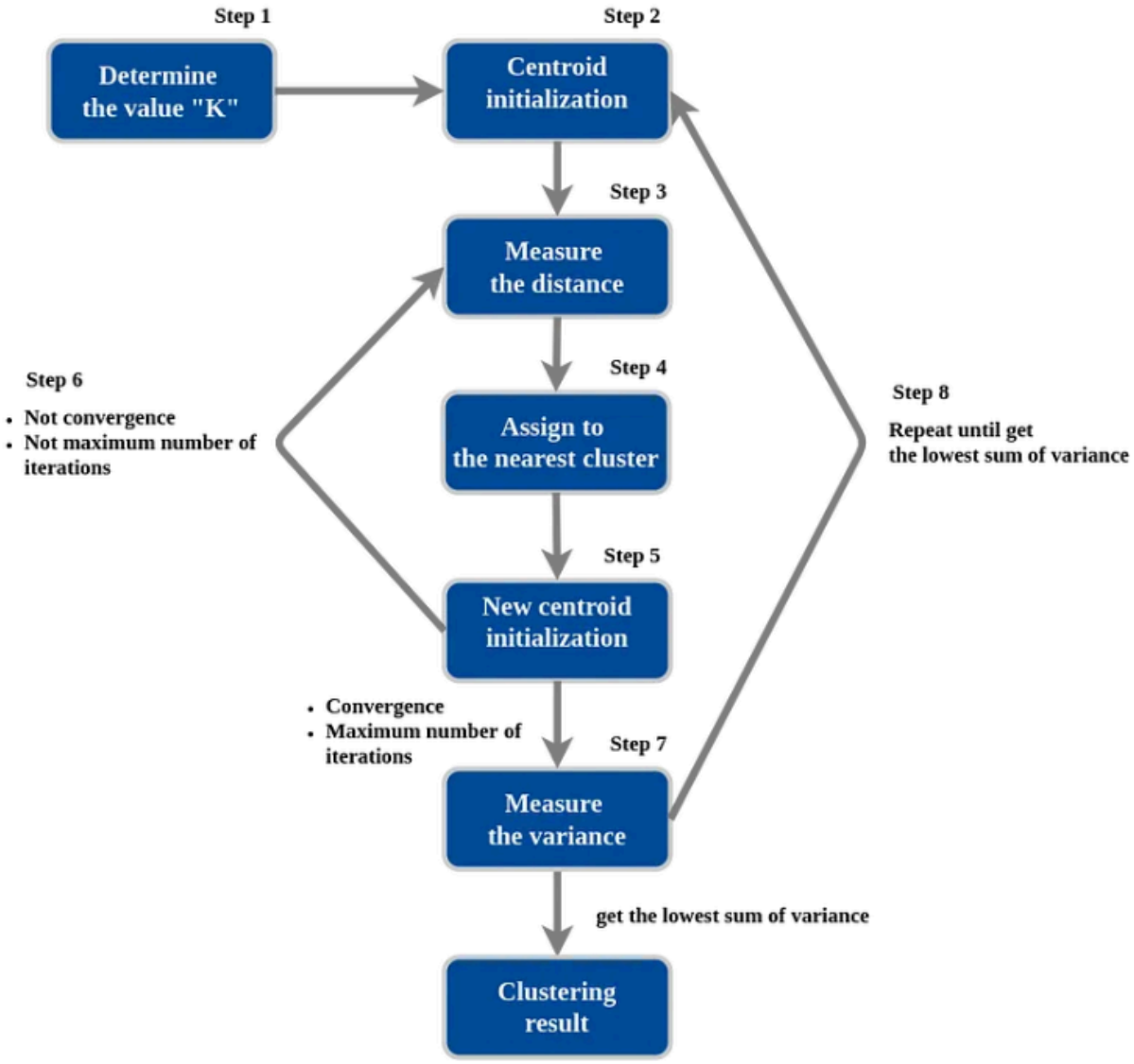
Étape 8. Répétez les étapes 2 à 7 jusqu'à ce que vous obteniez la plus petite somme de variance.
Par exemple - tentatives 2 avec un centroïde aléatoire différent





Répéter jusqu'à l'arrêt :
Jusqu'à ce que nous obtenions la somme de variance la plus faible et que nous choisissions ces groupes comme résultat.

OVERVIEW :



Implementation :

Dataset utilisé :
https://drive.google.com/file/d/19g4T97_dXKU6xkAUFT-A4OnPYtHW2qg_/view?usp=sharing

Les fonction implémenté :

- Kmeans_custom(k,X,max_iter=300) :

Implémente l'algorithme K-Means pour le clustering de données. Elle sélectionne aléatoirement K points dans l'ensemble de données pour servir de centroids initial, puis itère jusqu'à ce que les centroids convergent vers une solution stable ou que le nombre maximal d'itérations soit atteint. À chaque itération, elle assigne chaque point de données au cluster dont le centroid est le plus proche, met à jour les centroids en prenant la moyenne des points de données assignés à chaque cluster, et vérifie la convergence. Enfin, elle retourne les centroids finaux et les étiquettes de cluster pour chaque point de données.

```python
def kmeans_custom(k, X, max_iter=300):
    # Initialisation des centroids : sélectionnez les k premiers points comme centroids initiaux
    centroids_indices = np.random.choice(len(X), k, replace=False)
    centroids = X[centroids_indices]

    for _ in range(max_iter):
        # Assignation des points au cluster le plus proche
        labels = assign_clusters(X, centroids)

        # Mise à jour des centroids
        new_centroids = []
        for i in range(k):
            cluster_points = X[labels == i]
            if len(cluster_points) > 0:
                new_centroid = np.mean(cluster_points, axis=0)
                new_centroids.append(new_centroid)
            else:
                new_centroids.append(centroids[i])
        new_centroids = np.array(new_centroids)

        # Vérification de la convergence
        if np.allclose(centroids, new_centroids):
            break
        centroids = new_centroids

    return centroids, labels
```

- Assign_clusters(X,centroids) :

Est responsable de l'affectation de chaque point de données à un cluster en calculant la distance entre chaque point de données et tous les centroids, puis en attribuant le point au cluster dont le centroid est le plus proche.

```
def assign_clusters(X, centroids):
    distances = []
    for centroid in centroids:
        dist = np.linalg.norm(X - centroid, axis=1)
        distances.append(dist)
    distances = np.array(distances)
    return np.argmin(distances, axis=0)
```

- Lecture et sélection des données :

```
df=pd.read_csv('Mall_Customers.csv')
df.head()
```

	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

```
dfa=df[['Age',"Annual Income (k$)"]]
dfa
```

	Age	Annual Income (k$)
0	19	15
1	21	15
2	20	16
3	23	16
4	31	17
...	...	...
195	35	120
196	45	126
197	32	126
198	32	137
199	30	137

200 rows × 2 columns

- Standardisation des données :

Standardiser les données avant d'appliquer l'algorithme K-Means est essentiel pour assurer que toutes les caractéristiques contribuent de manière égale à la mesure de la distance entre les points de données. Sans standardisation, les caractéristiques avec des échelles plus grandes peuvent dominer la distance, introduisant ainsi des biais dans le clustering. La standardisation met toutes les caractéristiques sur la même échelle, garantissant ainsi une contribution équilibrée de chaque caractéristique à la formation des clusters.

```
scdfa=StandardScaler()
dfa_std=scdfa.fit_transform(dfa.astype(float))
dfa_std
```

```
array([[-1.42456879, -1.73899919],
       [-1.28103541, -1.73899919],
       [-1.3528021 , -1.70082976],
       [-1.13750203, -1.70082976],
       [-0.56336851, -1.66266033],
       [-1.20926872, -1.66266033],
       [-0.27630176, -1.62449091],
       [-1.13750203, -1.62449091],
       [ 1.80493225, -1.58632148],
       [-0.6351352 , -1.58632148],
       [ 2.02023231, -1.58632148],
       [-0.27630176, -1.58632148],
       [ 1.37433211, -1.54815205],
       [-1.06573534, -1.54815205],
       [-0.13276838, -1.54815205],
       [-1.20926872, -1.54815205],
       [-0.27630176, -1.50998262],
       [-1.3528021 , -1.50998262],
       [ 0.94373197, -1.43364376],
       [-0.27630176, -1.43364376],
       [-0.27630176, -1.39547433],
       [-0.99396865, -1.39547433],
       [ 0.51313183, -1.3573049 ],
       [-0.56336851, -1.3573049 ],
       [ 1.08726535, -1.24279661],
...
       [-0.27630176,  2.26879087],
       [ 0.44136514,  2.49780745],
       [-0.49160182,  2.49780745],
       [-0.49160182,  2.91767117],
       [-0.6351352 ,  2.91767117]])
```

- Visualisation des données :

Pour k=[1…10]

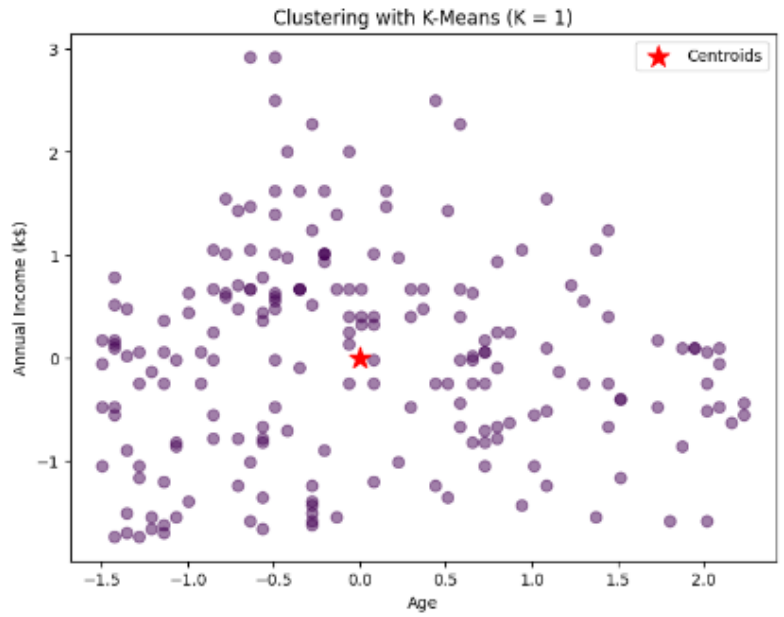
```
X = dfa_std
for k in range(1, 11):
    # Call your K-Means function
    centroids, labels = kmeans_custom(k, X)

    # Visualize the clusters
    plt.figure(figsize=(8, 6))
    plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50, alpha=0.5)
    plt.scatter(centroids[:, 0], centroids[:, 1], marker='*', c='red', s=200, label='Centroids')
    plt.xlabel('Age')
    plt.ylabel('Annual Income (k$)')
    plt.title(f'Clustering with K-Means (K = {k})')
    plt.legend()
    plt.show()
```

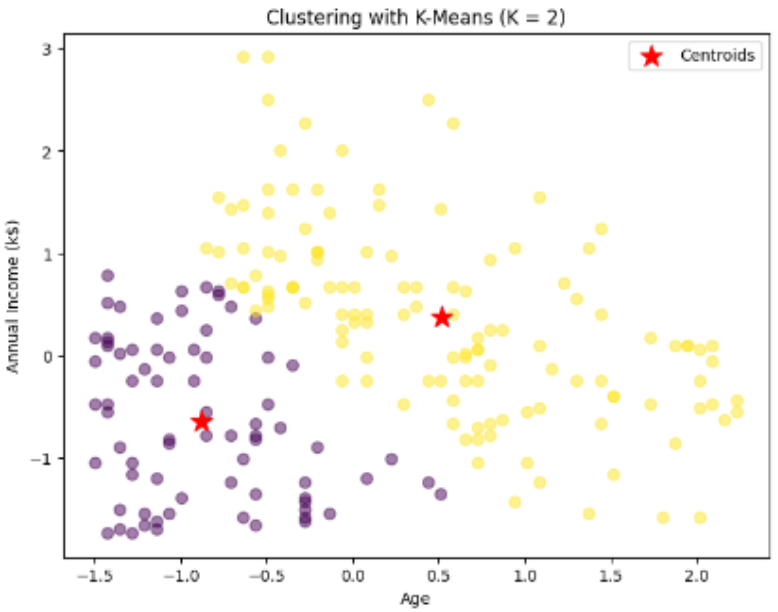
- Méthode de courde :

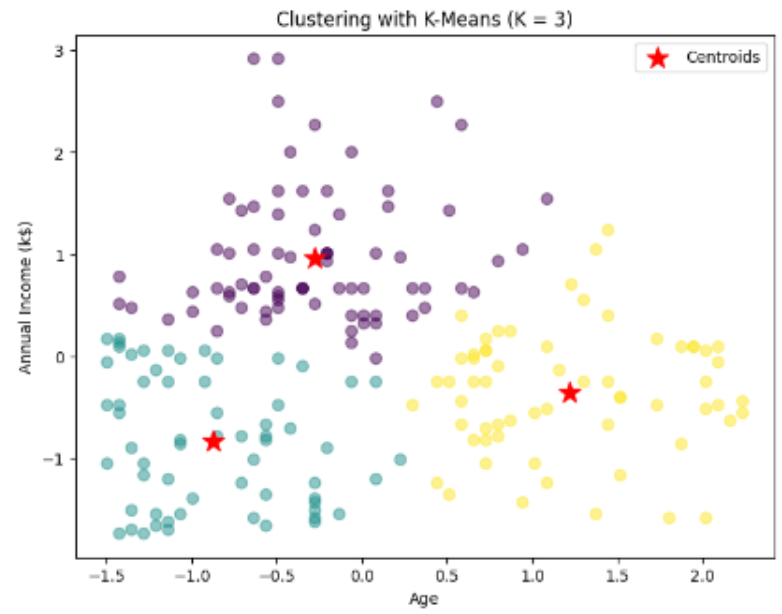
Pour retrouver le K ideal

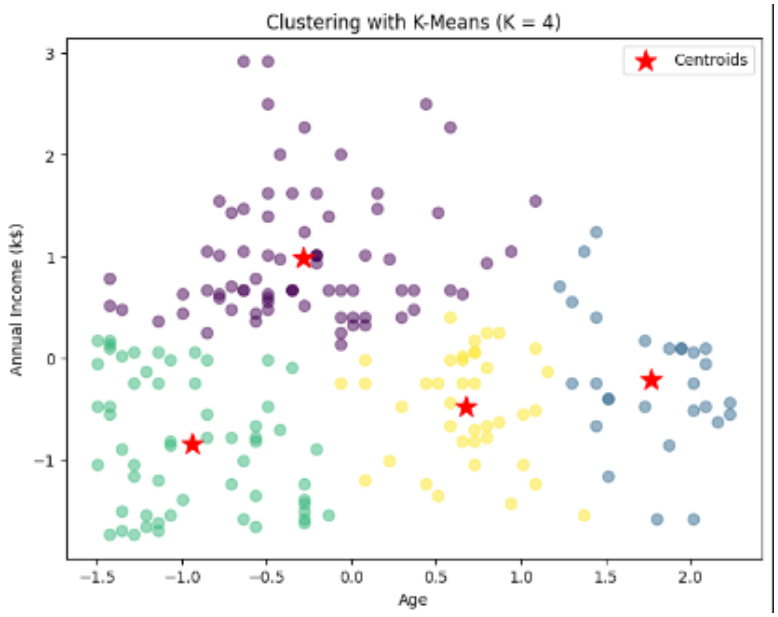
```
sse = []
for k in range(1, 11):
    centroids, labels = kmeans_custom(k, X)
    sse.append(np.sum((X - centroids[labels])**2))

# Plot the elbow graph
plt.plot(range(1, 11), sse, marker='o')
plt.xlabel('Number of clusters (K)')
plt.ylabel('Sum of Squared Distances (SSE)')
plt.title('Elbow Method for Optimal K')
plt.xticks(range(1, 11))
plt.grid(True)
plt.show()
```

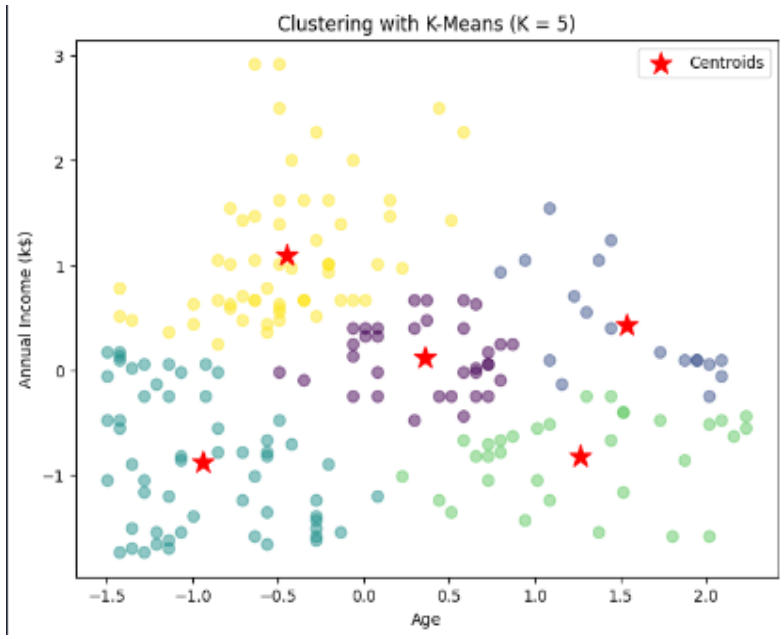
Résultats obtenu:

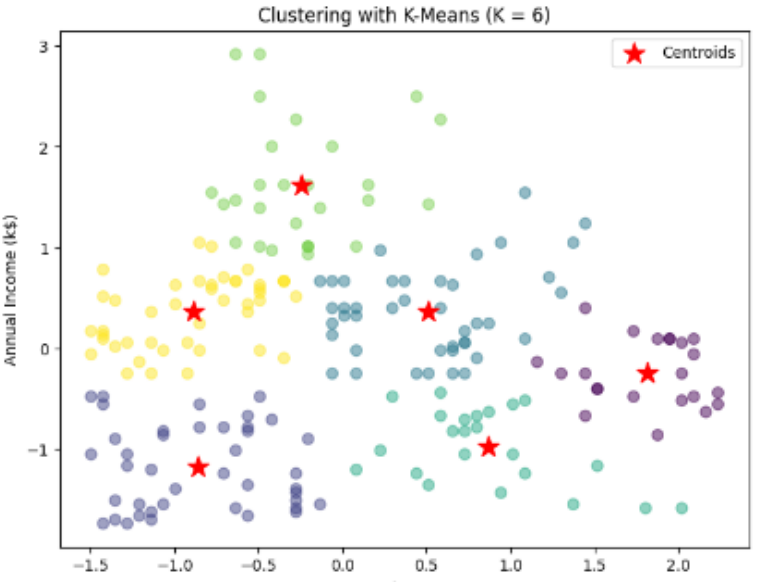


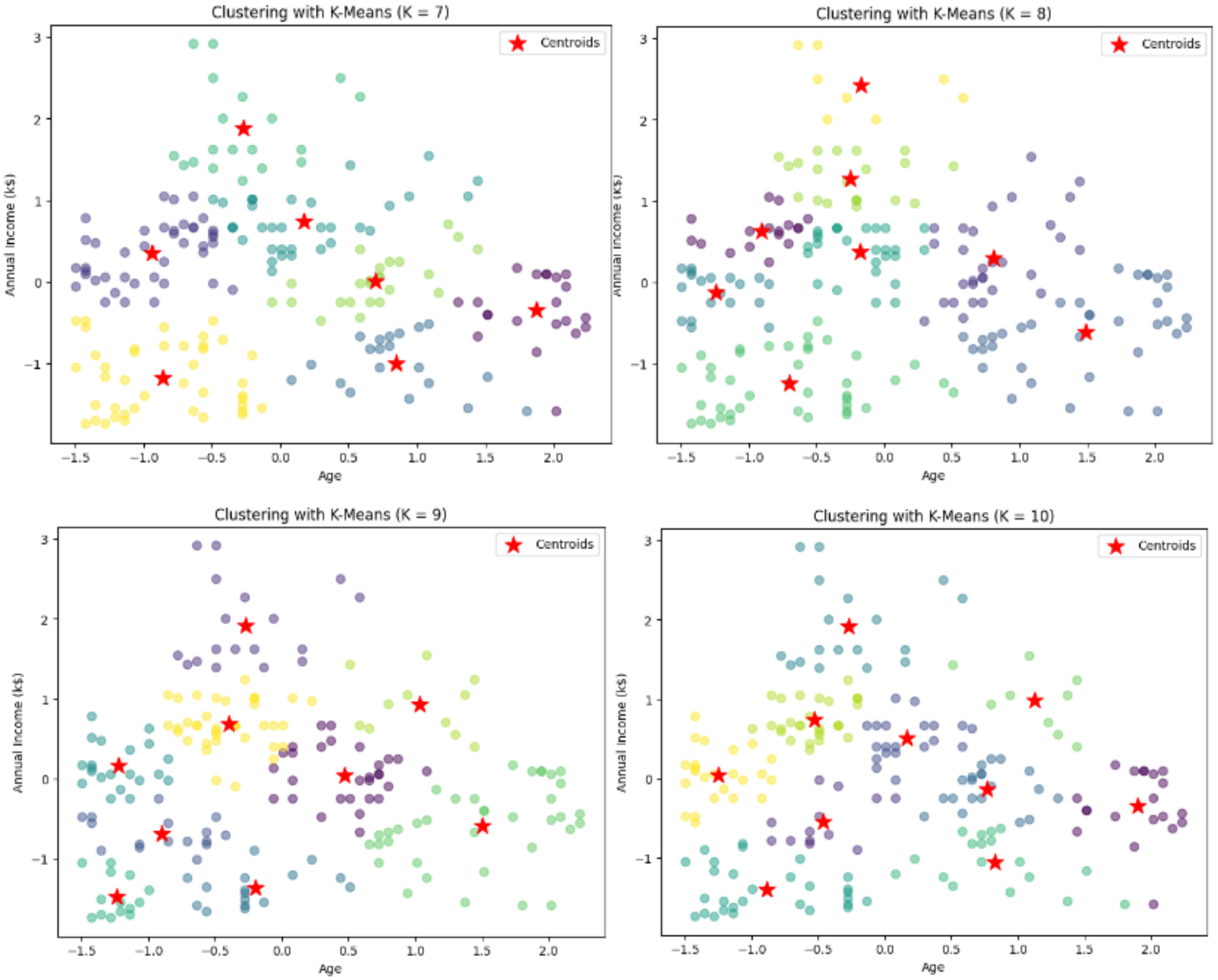










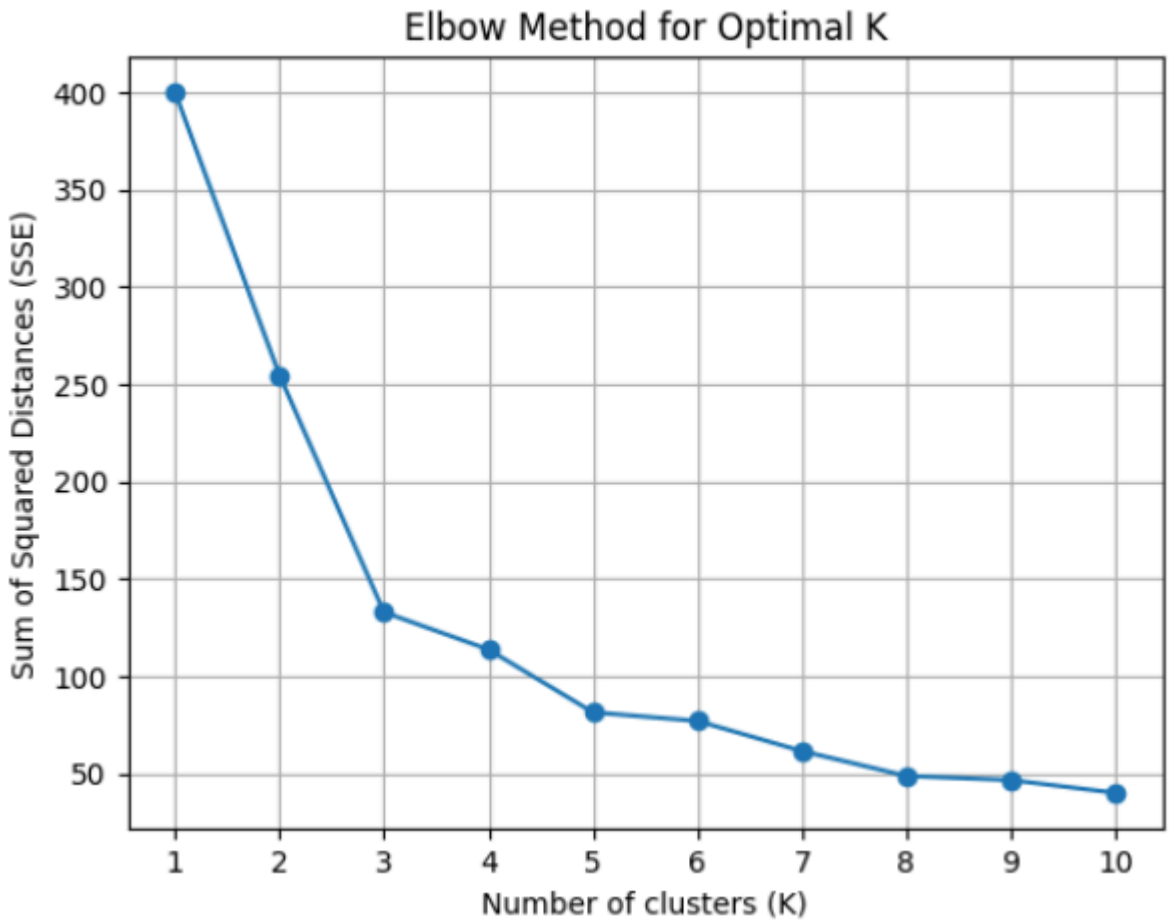


Remarques :

Augmenter K peut conduire à un surajustement des données, où des clusters trop nombreux sont créés et des modèles sont trouvés dans le bruit des données. Cela peut conduire à une perte de généralisation et à des clusters peu significatifs.

- pour trouver le K idéal, la méthode du point de course est utilisé

Voici le graphe obtenu après avoir appliqué cette méthodes sur les résultat du clustering des different K :

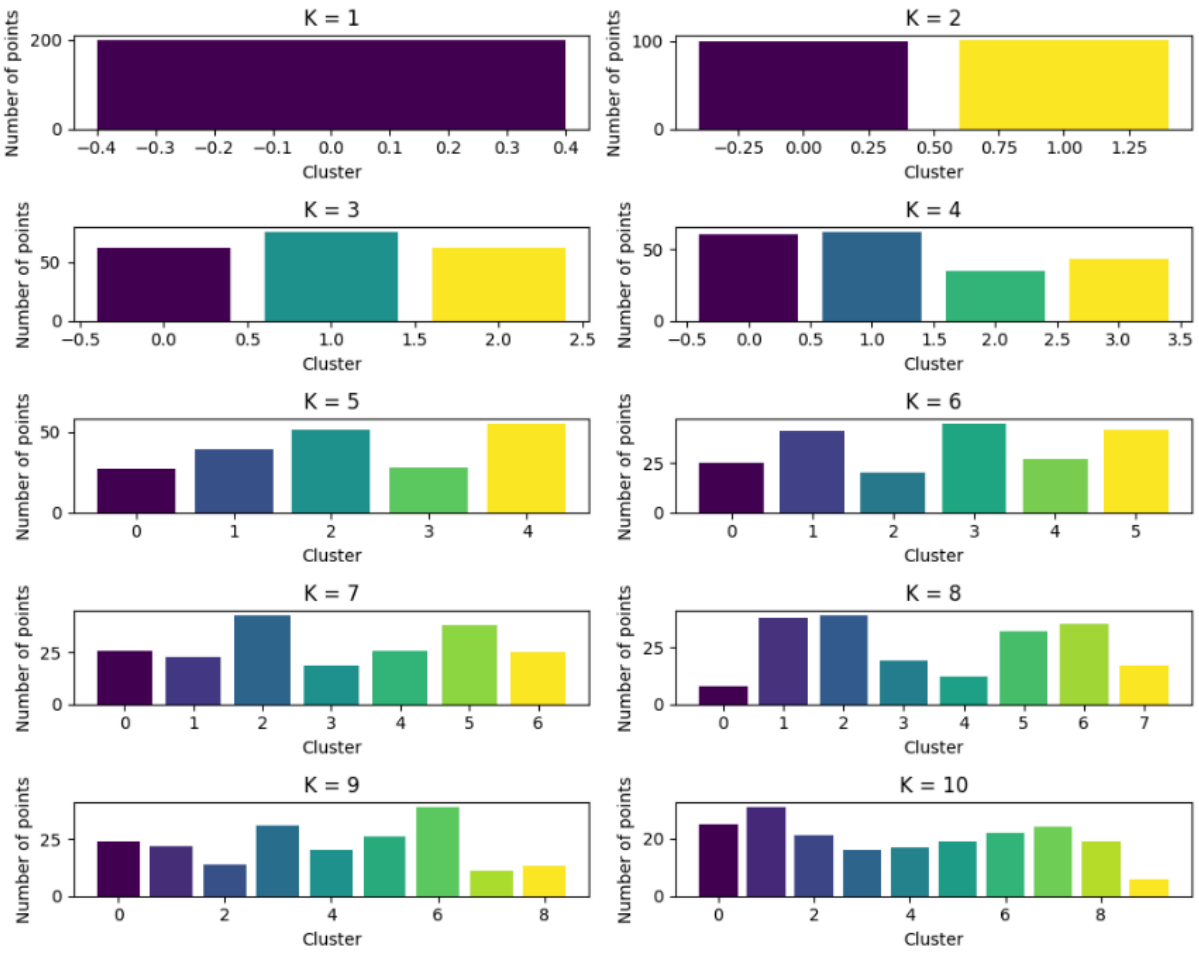


Dans le graphique du coude, on peut rechercher un point où la décroissance de la SSE ralentit, formant une courbe qui ressemble à un coude. Ce point est souvent considéré comme un candidat pour le nombre optimal de clusters.

Interprétation du graphique :

On peut remarquer que la décroissance de la SSE se ralenti dans le cas de k=3, donc on peut dire que 3 est considéré comme un candidat pour le nombre optimal de clusters

Représentation en forme de ligne verticals :
Pour pouvoire voir bien les clusters de désigner le nombre K optimal



Interprétation :
Comme on déjà vu dans la représentation graphique de la course, ici aussi on peut voir que le nombre idéal c'est 3 car il a la meilleur division de données en clusters .

Avantages du regroupement K-Means :

- Simplicité : K-Means est facile à comprendre et à mettre en œuvre, ce qui en fait un choix approprié pour les débutants.
- Rapidité : il est efficace sur le plan des calculs et peut traiter efficacement de grands ensembles de données.
- Évolutivité : K-Means s'adapte bien au nombre de points de données et de grappes.
- Polyvalence : Il peut être appliqué à un large éventail de types de données, ce qui en fait un outil polyvalent pour l'analyse des données.

Limites du regroupement K-Means

- Nombre de clusters (K) : La sélection de la valeur optimale de K peut s'avérer difficile et nécessite souvent une connaissance du domaine ou des techniques supplémentaires.
- Sensibilité à l'initialisation : Les résultats de l'algorithme peuvent varier en fonction de l'emplacement initial des centroïdes, ce qui conduit à des solutions sous-optimales.
- Hypothèse de grappes sphériques : K-Means suppose que les grappes sont sphériques et de taille égale, ce qui n'est pas toujours le cas.
- Sensibilité aux valeurs aberrantes : K-Means peut être sensible aux valeurs aberrantes, qui peuvent influencer de manière disproportionnée la position des centroïdes.

CONCLUSION

Le regroupement K-Means est un algorithme fondamental dans le domaine de l'apprentissage non supervisé, avec un large éventail d'applications. Comprendre ses principes et la manière de l'utiliser efficacement peut permettre aux scientifiques et aux analystes de données d'extraire des informations précieuses de leurs données. Cependant, il est essentiel de garder à l'esprit ses limites et la nécessité d'un réglage réfléchi des paramètres, en particulier lorsqu'il s'agit d'ensembles de données réels. K-Means n'est qu'un outil parmi d'autres dans la vaste boîte à outils de l'apprentissage automatique, mais il s'agit sans aucun doute d'un outil précieux pour la segmentation des données et la découverte de modèles.