Kenza BELALOUI B3 Dev Data & IA

TP – K-Means

- Objectif :

L'objectif de ce TP est de réaliser une segmentation de clients à l'aide de l'algorithme de clustering non supervisé K-Means à partir du dataset **Mall_Customers.csv**.
Le but est d'identifier des groupes de clients ayant des comportements similaires en fonction de leurs caractéristiques socio-économiques et de leur comportement de dépense.

Le dataset contient les variables suivantes :

- **CustomerID** : identifiant du client
- **Gender** : genre
- **Age** : âge du client
- **Annual Income (k$)** : revenu annuel en milliers de dollars
- **Spending Score (1-100)** : score de dépense attribué par le centre commercial

Partie 1 – Préparation des données

1.1 Chargement et exploration :

Le dataset **Mall_Customers.csv** est chargé avec pandas.read_csv().
On affiche un aperçu (head()), des informations générales (info()) et les valeurs manquantes (isnull().sum()).

1.2 Sélection des variables pertinentes :

Pour la segmentation, on choisit les variables :

- **Age**
- **Annual Income (k$)**

- **Spending Score (1-100)**

Justification :

- **CustomerID** est un simple identifiant, il n'a pas de signification pour le clustering.
- **Gender** est catégorielle (texte) et ne se prête pas directement à K-Means sans encodage, et elle n'est pas indispensable pour une première segmentation simple.
- **Age**, **Revenu annuel** et **Spending Score** sont des variables **numériques et explicatives** du comportement client, et sont donc pertinentes pour la segmentation.

On crée donc un sous-dataframe :

```
df_selected = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
```

1.3 Standardisation des variables :

L'algorithme K-Means est sensible à l'échelle des variables (il utilise les distances).
On applique donc une **standardisation** avec StandardScaler de sklearn :

```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_selected)
```

Chaque variable est transformée pour avoir une moyenne proche de 0 et un écart-type proche de 1.
Cela permet de donner **le même poids** à l'âge, au revenu et au score de dépense dans le calcul des distances.

Partie 2 – Analyse exploratoire

2.1 Visualisation des relations entre les variables :

Plusieurs visualisations ont été réalisées :

1. **Scatter plot 2D + couleur (3 variables)**
    - X : Revenu annuel
    - Y : Spending Score
    - Couleur : Âge

Cela permet de repérer visuellement des regroupements de clients en fonction du revenu et du score de dépense.

2. **Histogrammes des variables** (df_selected.hist())
    - montrent la distribution des âges, des revenus et des scores de dépense
    - permettent de voir si les variables sont plutôt concentrées ou dispersées.
3. **Pairplot** (sns.pairplot(df_selected))
    - affiche les relations 2 à 2 entre Age, Income, Spending Score
    - permet d'apercevoir des **nuages de points distincts**, surtout sur le plan Revenu / Spending Score.
4. **Heatmap de corrélation** (sns.heatmap(df_selected.corr(), ...))
    - montre les corrélations entre les trois variables.
    - Les corrélations sont modérées ou faibles, ce qui indique que chaque variable apporte une information complémentaire.

2.2 Tendances et regroupements visuels :

Les scatter plots (surtout Revenu vs Spending Score) montrent des **zones de concentration** de points, ce qui suggère l'existence de plusieurs groupes de clients:

- des clients à **revenu élevé et score de dépense élevé**,
- des clients à **revenu élevé mais score faible**,
- des clients à **revenu moyen ou faible, avec score variable**.

On observe donc visuellement plusieurs **tendances** et potentiels **clusters**.

2.3 Hypothèses sur le nombre de segments possibles :

À partir des visualisations :

- on peut supposer qu'il existe **plusieurs segments**, probablement entre 4 et 6 groupes.
- ces hypothèses seront validées ensuite par la méthode du coude et le score de silhouette.

Partie 3 – Application de K-Means :

3.1 K-Means pour différents nombres de clusters (k = 2 à 10) :

On applique K-Means sur les données standardisées pour k allant de 2 à 10 :

Pour chaque k, on stocke :

- l'inertie (somme des distances au carré aux centroïdes),
- le **score de** silhouette.

3.2 Méthode du coude :

On trace la courbe Inertia en fonction de k.
On observe que l'inertie diminue fortement au début, puis la courbe s'aplatit à partir de k = 6.
Ce point d'inflexion (le "coude") indique que au-delà de 6 clusters, l'amélioration est faible.

3.3 Score de silhouette :

On trace de même le score de silhouette en fonction de k.
Le score de silhouette est maximal pour k = 6, ce qui signifie que :

- la séparation entre les clusters est bonne,
- les clients d'un même cluster sont relativement proches les uns des autres
- et bien séparés des autres clusters.

3.4 Choix du nombre de clusters :

Les deux méthodes (coude et silhouette) indiquent que k = 6 est un bon choix :

- coude visible à k = 6,
- score silhouette maximal à k = 6.

Nous retenons donc k = 6 comme nombre optimal de segments.

3.5 Entraînement du modèle final et ajout des labels :

```
k_optimal = 6
model_final = KMeans(n_clusters=k_optimal, random_state=42)
model_final.fit(df_scaled)

clusters = model_final.labels_
```

- Une nouvelle colonne **Cluster** est ajoutée au dataset, indiquant le numéro de cluster de chaque client.

Partie 4 – Interprétation des clusters :

4.1 Profils moyens par cluster :

On calcule les moyennes des variables pour chaque cluster :

```
cluster_profiles = df.groupby('Cluster')[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']].mean()
```

Ce tableau permet de voir, pour chaque cluster :

- l'âge moyen,
- le revenu annuel moyen,
- le score de dépense moyen.

4.2 Description qualitative des segments :

À partir de ces moyennes, on peut décrire des **profils typiques**. Par exemple (à adapter selon tes valeurs exactes) :

- **Cluster 0 : Jeunes dépensiers**
  Âge plutôt faible, revenu moyen, **spending score élevé**.
- **Cluster 1 : Clients économes**
  Revenu moyen ou élevé, mais **score de dépense faible**.

- **Cluster 2 : Clients à haut revenu et forte dépense**
  Revenu élevé et **spending score élevé** → clients très intéressants pour le business.
- **Cluster 3 : Jeunes à faible revenu et forte dépense**
  Revenu plus faible, mais **score de dépense assez élevé** → comportement impulsif.
- **Cluster 4 : Clients plus âgés et peu dépensiers**
  Âge plus élevé, dépenses plus faibles.
- **Cluster 5 : Profil intermédiaire**
  Variables proches de la moyenne → clients "moyens", sans comportement extrême.

(Les numéros de cluster peuvent changer, l'important est l'interprétation à partir des moyennes.)

4.3 Étiquettes descriptives :

On peut donner des noms marketing aux segments, par exemple :

- Jeunes dépensiers
- Riches gros consommateurs
- Riches peu dépensiers
- Clients économes
- Jeunes impulsifs
- Clients intermédiaires

Ces étiquettes facilitent la compréhension pour un responsable marketing.

4.4 Visualisation des clusters en 2D et 3D :

- Visualisation 2D :

On représente les clients sur le plan :

- X : Revenu annuel
- Y : Spending Score
- Couleur : Cluster

- et on ajoute les centroïdes (croix noires) :

```
centers = model_final.cluster_centers_
centers_original = scaler.inverse_transform(centers)

plt.figure(figsize=(8,6))
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-
100)'], c=df['Cluster'], cmap='tab10')
plt.scatter(centers_original[:, 1], centers_original[:, 2],
s=200, c='black', marker='X')
plt.title("Clusters et centroïdes")
plt.xlabel("Revenu annuel (k$)")
plt.ylabel("Spending Score")
plt.show()
```

- Visualisation 3D :

On représente ensuite les clusters en 3D :

- X : Âge
- Y : Revenu annuel
- Z : Spending Score
- Couleur : Cluster
- Centroïdes : croix noires

```
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df['Age'], df['Annual Income (k$)'], df['Spending
Score (1-100)'],
           c=df['Cluster'], cmap='tab10')

# Calcul des centroïdes remis à l'échelle originale
centers = model_final.cluster_centers_
centers_original = scaler.inverse_transform(centers)

# Scatter 3D des centroïdes
ax.scatter(centers_original[:, 0], centers_original[:, 1],
centers_original[:, 2],c='black', s=300, marker='X',
label='Centroïdes')

ax.set_xlabel('Âge')
ax.set_ylabel('Revenu annuel (k$)')
ax.set_zlabel('Spending Score')
plt.title("Clusters en 3D avec centroïdes")
ax.legend()

plt.show()
```

Partie 5 – Extension :

5.1 Comparaison avec un autre algorithme : DBSCAN :

En complément de K-Means, l'algorithme **DBSCAN** a été appliqué sur les données standardisées.
Contrairement à K-Means, DBSCAN ne nécessite pas de choisir le nombre de clusters et peut détecter des points isolés (outliers).
Cependant, sur ce dataset, DBSCAN produit moins de groupes et identifie de nombreux outliers.
Conclusion : K-Means est mieux adapté que DBSCAN pour ce type de données.

5.2 Stabilité du modèle :

K-Means a été testé avec plusieurs valeurs de random_state.
Les scores de silhouette obtenus sont très proches, ce qui montre que le clustering est **stable** et que le choix de **k = 6** est fiable.

- Conclusion :

- L'algorithme K-Means a permis de **segmenter les clients en 6 groupes** distincts à partir de leur âge, revenu et score de dépense.

- Le nombre optimal de clusters a été choisi en combinant **la méthode du coude** et le **score de silhouette**, ce qui rend la démarche rigoureuse.

- Les visualisations 2D et 3D montrent des **comportements différents** entre les segments, ce qui peut être exploité en **marketing (ciblage, offres personnalisées, fidélisation)**.

- Limites possibles :

  - le genre n'a pas été pris en compte,
  - les données proviennent d'un seul centre commercial,
  - K-Means suppose des clusters de forme "globulaire" et peut être sensible aux outliers.