FACULTÉ DES SCIENCES ET TECHNIQUES
DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Fondements et applications de l'apprentissage automatique

B. HILALI - brahill79@gmail.com / T. SABRI – sabritarik@gmail.com
Filière: Ingénierie Logicielle et Intégration des Systèmes Informatiques (ILISI)
24-25

TP 5 : Clustering (Regroupement non supervisé)

- **Objectif** : Découvrir les techniques de clustering pour segmenter des données non étiquetées.
- **Contenu** :
  - Concepts de base du clustering.
  - Implémentation d'algorithme K-Means.
  - Évaluation des clusters : méthode du coude, silhouette score, inertie.

1. Apprentissage non supervisé

- Les données ne sont pas étiquetées.
- Le modèle doit découvrir des structures ou des patterns par lui-même, sans guide.
- Explorer un territoire inconnu sans carte : on découvre des patterns cachés.

1. Apprentissage non supervisé

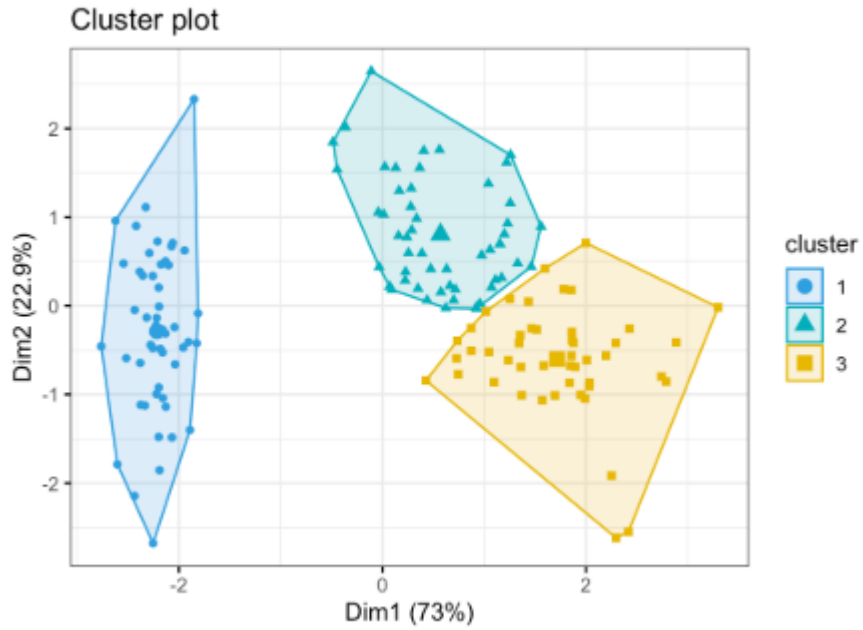
- Cette approche se divise également en deux grandes catégories :
  - ✓ **Le regroupement (ou clustering)** : le modèle regroupe les données en fonction de leurs similarités. Par exemple, segmenter des clients en groupes homogènes pour du marketing ciblé.
  - ✓ **La réduction de dimension** : l'objectif est de simplifier les données en réduisant le nombre de variables, tout en conservant l'essentiel de l'information. Cela est utile pour visualiser des données complexes ou accélérer les calculs.

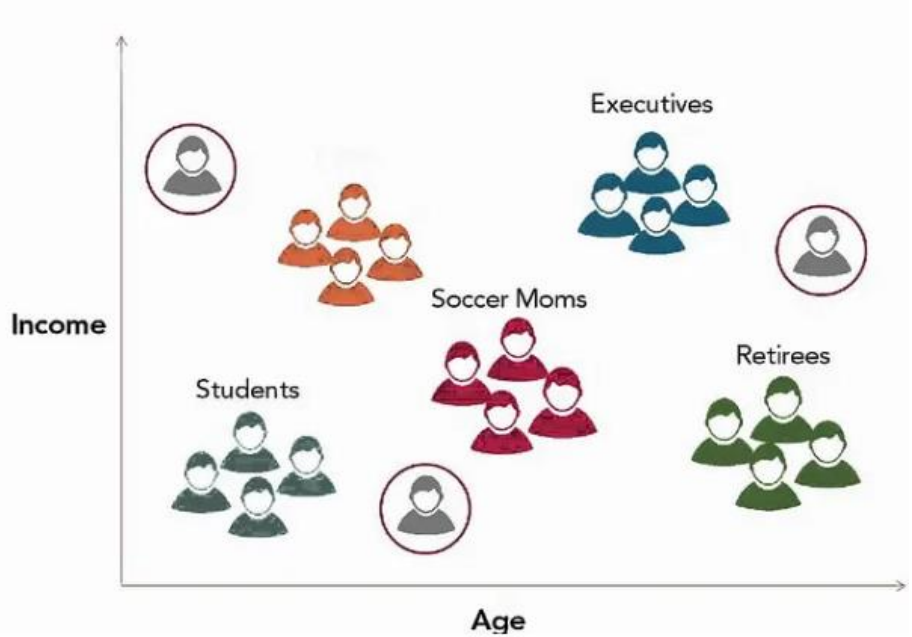
2. Qu'est-ce que K-Means ?

- K-Means est un algorithme de clustering utilisé dans l'apprentissage non supervisé.
- Il regroupe des données similaires en clusters (groupes) basés sur leurs caractéristiques.
- Il identifie des groupes naturels dans les données sans intervention humaine.

2. Qu'est-ce que K-Means ?

- **Applications :**
  - ✓ *Marketing* : Classification des clients selon leurs préférences.
  - ✓ *Bioinformatique :* Regroupement de gènes ayant des expressions similaires.





3. Étapes de l’algorithme K-Means

1. **Initialisation :**
   - Choisissez le nombre de clusters k.
   - Initialisez aléatoirement k centroïdes (points représentant les centres des clusters).
2. **Assignation :**
   - Attribuez chaque point de données au cluster dont le centroïde est le plus proche (basé sur la distance euclidienne).
3. **Mise à Jour :**
   - Recalculez les centroïdes en prenant la moyenne des points dans chaque cluster.
4. **Répétez :**
   - Répétez les étapes 2 et 3 jusqu'à ce que les centroïdes ne changent plus (convergence).

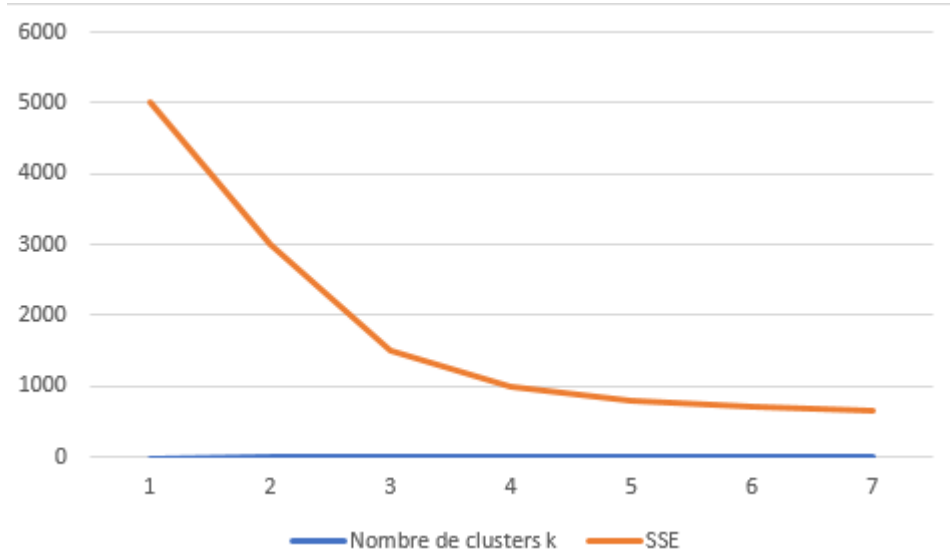
4. Avantages et limitations

1. **Avantages :**
   - ✓ Simplicité et rapidité d'exécution.
   - ✓ Très efficace pour les grands ensembles de données.
   - ✓ Résultats interprétables.

2. **Limitations :**
   - ✓ Nécessite de spécifier le nombre de clusters k à l'avance.
   - ✓ Ne fonctionne pas bien avec des clusters de formes complexes ou des données non linéaires.

5. Méthode du coude (Elbow Method)

- La méthode du coude (Elbow Method) est une technique couramment utilisée pour déterminer le nombre optimal de clusters k. Elle repose sur l'analyse de la variance intra-cluster, mesurée par la somme des carrés des erreurs (SSE) ou inertie , pour différentes valeurs de k.
- La méthode Elbow s'appuie sur l'idée suivante :
  - ✓ Lorsque vous augmentez le nombre de clusters k, la variance intra-cluster (SSE) diminue car chaque cluster devient plus spécifique et contient moins de points.
  - ✓ Cependant, au-delà d'un certain point, ajouter plus de clusters **n'apporte pas** de réduction significative de la variance intra-cluster. Ce point "optimal" correspond à un "coude" (elbow) dans la courbe.



5. Méthode du coude (Elbow Method)

- **Étapes de la méthode Elbow:**
- Étape 1 : Calculer la SSE pour différentes valeurs de k

  Pour chaque valeur de k (par exemple, k=1,2,3,…,10), exécutez l'algorithme K-Means. Calculez la somme des carrés des erreurs SSE (Sum of Squared Errors) ou inertie.

$$SSE = \sum_{i=1} \sum_{x \in C_i} ||x - \mu_i||^2$$

où :

- C_i est le cluster i,
- μ_i est le centroïde du cluster i,
- x est un point appartenant au cluster i.

5. Méthode du coude (Elbow Method)

- Étape 2 : Tracer la courbe de la SSE en fonction de k

  Tracez un graphique avec k sur l'axe des abscisses et la SSE sur l'axe des ordonnées.

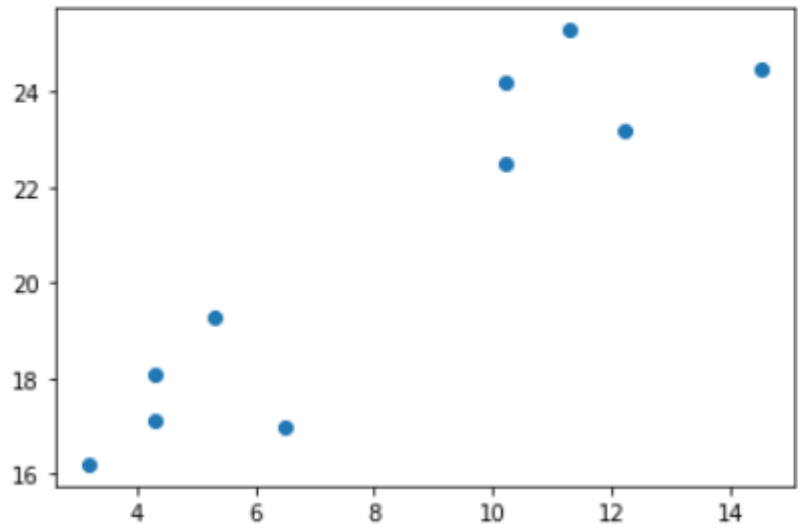
- Étape 3 : Identifier le "coude"

  Recherchez le point où la courbe commence à se stabiliser (le "coude"). Ce point indique le nombre optimal de clusters k.

6. Application 1

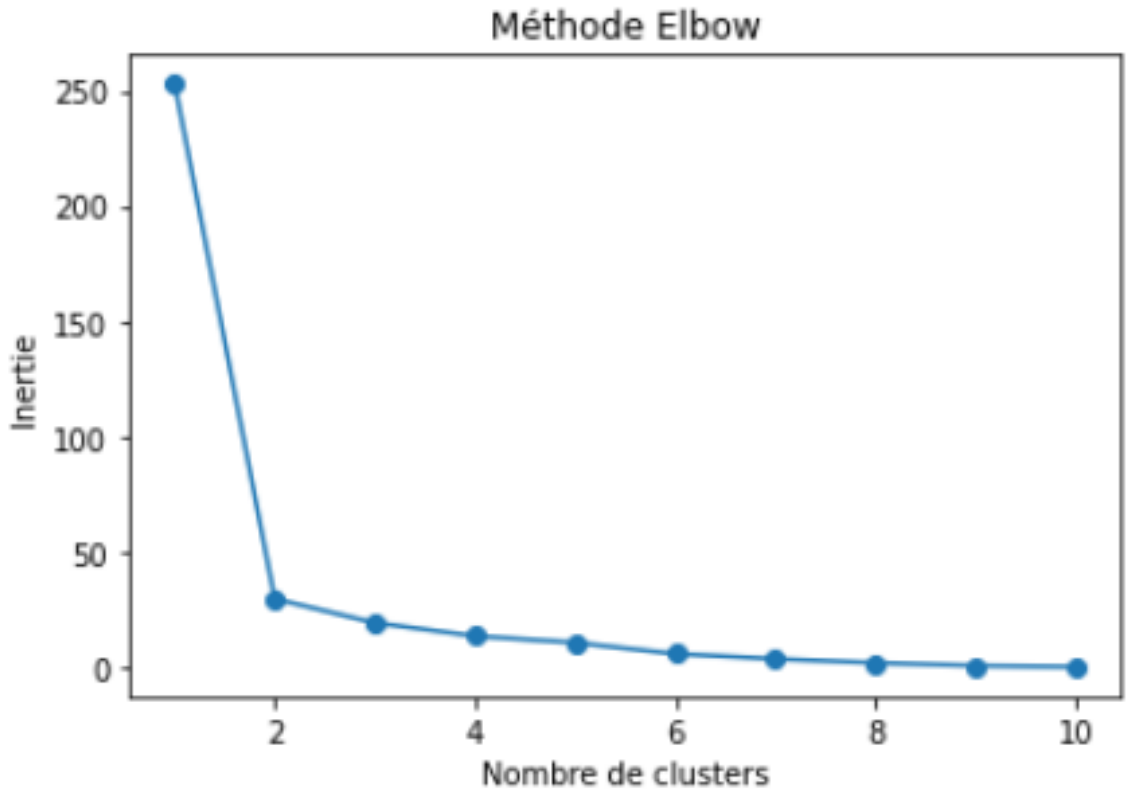
- ***Objectifs:***
  1. Découvrir et visualiser les groupes clusters présents dans un ensemble de points bidimensionnels (x,y) à l'aide de l'algorithme K-Means.
  2. Utiliser la méthode Elbow pour déterminer le nombre optimal de clusters.
  3. Afficher les clusters finaux ainsi que leurs centres.

```
import matplotlib.pyplot as plt
x = [4.3, 5.3, 10.2, 4.3, 3.2, 11.3, 14.5 , 6.5, 10.2, 12.2]
y = [18.1, 19.3, 24.2, 17.1, 16.2, 25.3, 24.5, 17, 22.5, 23.2]
plt.scatter(x, y)
plt.show()
```



6. Application 1

```
from sklearn.cluster import KMeans
data = list(zip(x, y))
inertias = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)
plt.plot(range(1,11), inertias, marker='o')
plt.title('Méthode Elbow')
plt.xlabel('Nombre de clusters')
plt.ylabel('Inertie')
plt.show()
```



6. Application 1

```
# Appliquer K-Means avec k=2
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(data)

# Récupérer les labels des clusters
labels = kmeans.labels_
labels
```

```
array([1, 1, 0, 1, 1, 0, 0, 1, 0, 0])
```

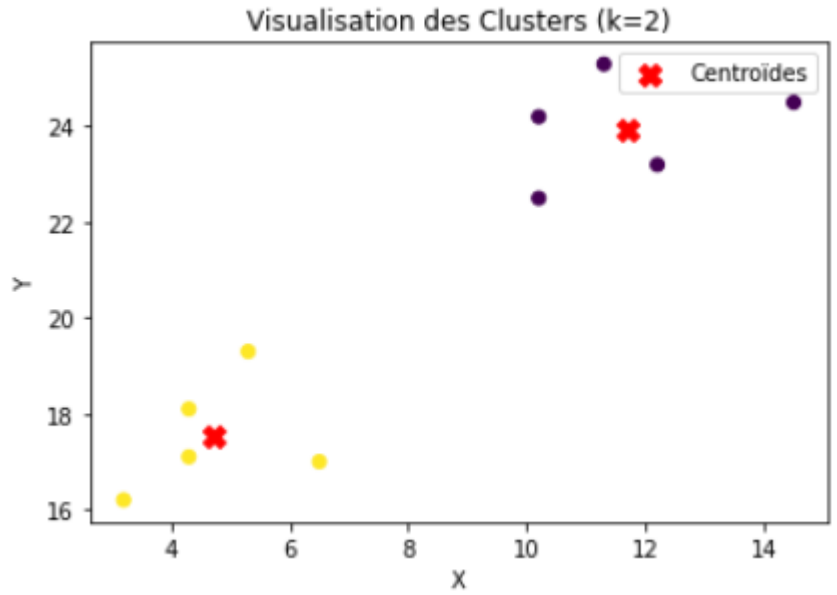
```
kmeans.cluster_centers_
```

```
array([[11.68, 23.94],
       [ 4.72, 17.54]])
```

```
kmeans.cluster_centers_[:, 0]
```

```
array([11.68,  4.72])
```

```
# Visualiser les clusters
plt.scatter(x, y, c=labels)  # Colorer les points par cluster
plt.scatter(kmeans.cluster_centers_[:, 0],
            kmeans.cluster_centers_[:, 1],
            c='red', s=100, marker='X', label='Centroïdes')  # Ajouter les centroïdes
plt.title('Visualisation des Clusters (k=2)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```



7. Coefficient de silhouette (Silhouette Score)

- Le coefficient de silhouette mesure à quel point chaque point est similaire aux points de son propre cluster par rapport aux points des autres clusters.
- Formule :
- Pour un point x :

$$s(x) = \frac{b(x) - a(x)}{\max(a(x), b(x))}$$

où :

- a(x) est la distance moyenne entre x et les autres points de son cluster (cohésion intra-cluster).
- b(x) est la distance moyenne minimale entre x et les points d'un autre cluster (séparation inter-clusters).

7. Coefficient de silhouette (Silhouette Score)

- Interprétation :
  - s(x) varie entre -1 et 1 :
    - ✓ Proche de 1 : le point est bien attribué à son cluster.
    - ✓ Proche de 0 : le point est à la limite entre deux clusters.
    - ✓ Négatif : le point est probablement mal attribué.
- Avantages :
  - Fournit une mesure globale de la qualité du clustering.
  - Ne nécessite pas de vérité terrain.

8. Application 2

- ***Problème:***

  Vous disposez d'un jeu de données artificiel contenant des points répartis en plusieurs groupes (clusters). Votre objectif est d'appliquer l'algorithme K-means pour identifier ces clusters et d'évaluer la qualité du clustering en calculant le coefficient de silhouette.

8. Application 2

1. *Générer un jeu de données artificiel*

- *Importez la classe KMeans depuis sklearn.cluster.*
- Importez la fonction silhouette_score depuis sklearn.metrics.
- *Utilisez la fonction make_blobs de sklearn.datasets pour générer un jeu de données contenant 300 points répartis en 4 clusters.*

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# 1. Générer un jeu de données artificiel
# Créons un jeu de données avec 300 points et 4 clusters bien séparés
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=42)
# cluster_std = Écart type des clusters (contrôle leur dispersion)

X.shape
```

```
(300, 2)
```

8. Application 2

2. *Appliquer l'algorithme K-means*

- *Créez une instance de K-means avec n_clusters=4 (nombre de clusters à identifier) et random_state=42.*
- *Entraînez le modèle sur les données X et prédisez les labels des clusters pour chaque point, stockez les labels prédits dans une variable labels.*

```
# 2. Appliquer K-means avec un nombre de clusters fixé (k=4)
kmeans = KMeans(n_clusters=4, random_state=42)
labels = kmeans.fit_predict(X)  # Labels prédits par K-means
labels[:10]
```

8. Application 2

3. *Calculer le coefficient de silhouette*

- Calculez le coefficient de silhouette moyen pour les clusters identifiés en utilisant les données X et les labels prédits labels.

```
# 3. Calculer le coefficient de silhouette
silhouette_avg = silhouette_score(X, labels)
print("Coefficient de silhouette moyen :",round(silhouette_avg,2))
```

```
Coefficient de silhouette moyen : 0.88
```

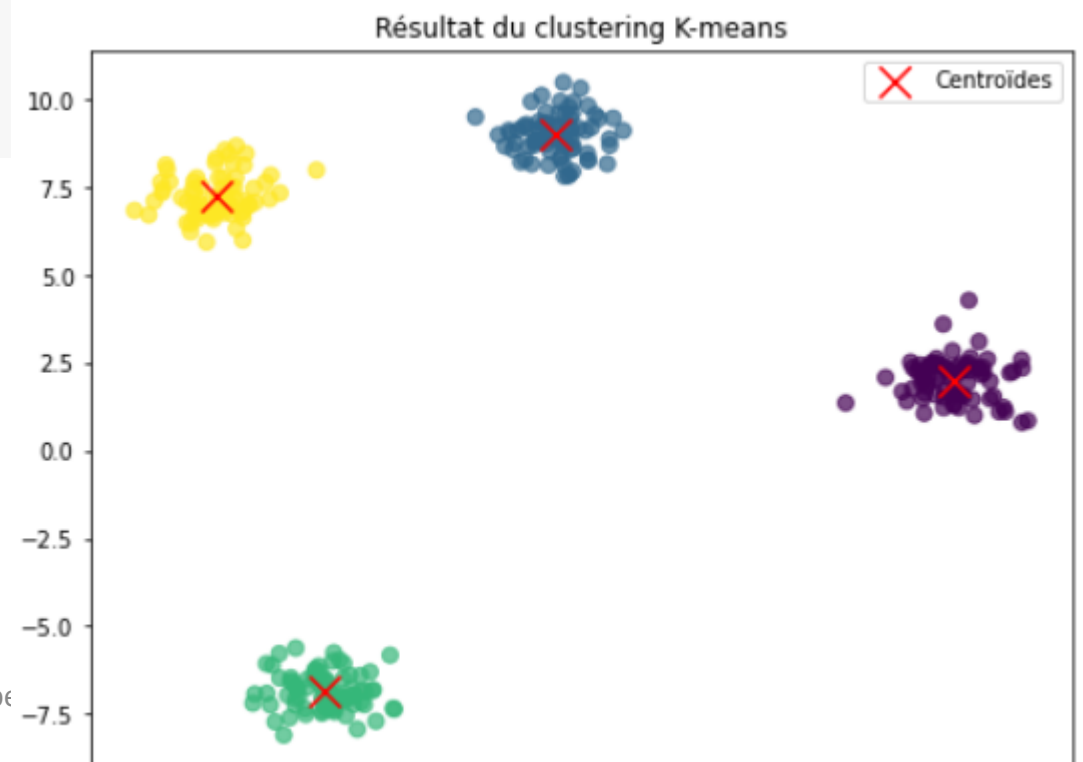
8. Application 2

4. Visualiser les clusters

- Importez la bibliothèque matplotlib.pyplot pour créer des graphiques.
- Tracez les points des données X en les colorant selon leurs labels (labels) :
- Utilisez le paramètre c=labels pour attribuer une couleur différente à chaque cluster.
- Ajoutez les centroïdes des clusters au graphique :
- Récupérez les centroïdes depuis l'attribut cluster_centers_ de l'objet K-means.
- Tracez les centroïdes avec des croix rouges (c='red', marker='x', s=200).
- Ajoutez un titre au graphique ("Résultat du clustering K-means"), une légende pour les centroïdes, et affichez le graphique.

8. Application 2

```python
# 4- Visualiser les clusters et leurs centroïdes
plt.figure(figsize=(8, 6))

# Afficher les points colorés selon leur cluster
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50, alpha=0.7)

# Afficher les centroïdes
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='x', s=200, label='Centroïdes')

plt.title("Résultat du clustering K-means")
plt.legend()
plt.show()
```



8. Application 2

5. Analyser la qualité du clustering pour différents nombres de clusters

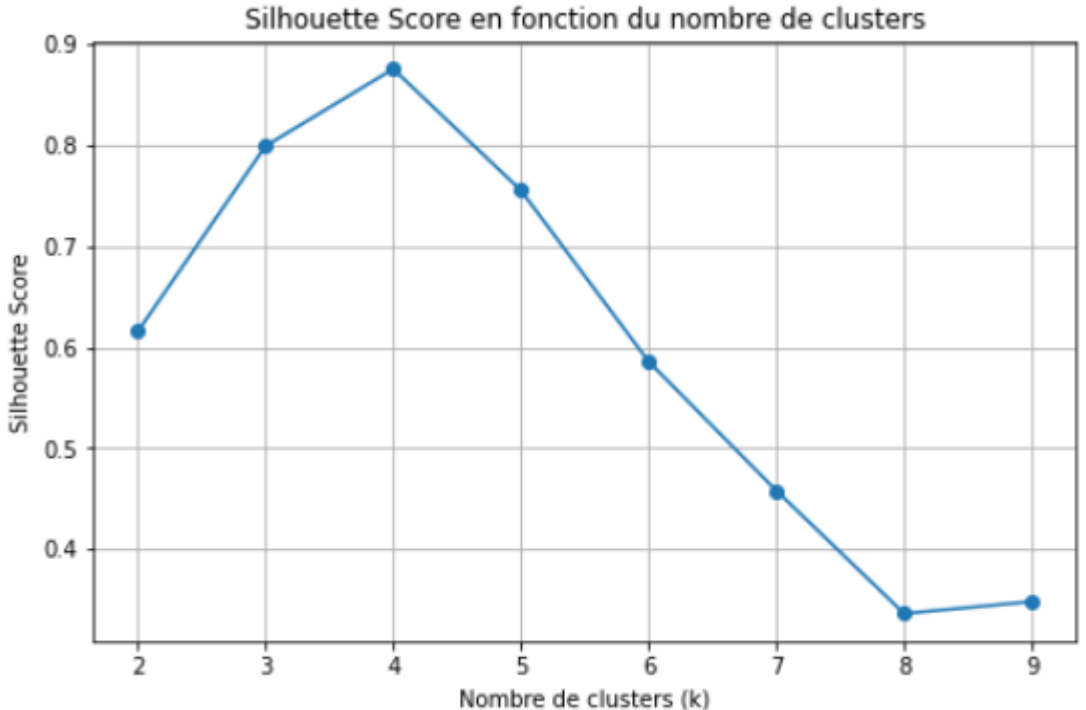
- Testez différentes valeurs de k (nombre de clusters) entre 2 et 9 pour trouver le nombre optimal de clusters.
- Pour chaque valeur de k :
  - ✓ Appliquez K-means avec le nombre de clusters actuel.
  - ✓ Calculez le coefficient de silhouette moyen.
  - ✓ Stockez les scores dans une liste.
  - ✓ Affichez le score de silhouette pour chaque k.
- Tracez un graphique montrant l'évolution du coefficient de silhouette en fonction de k

8. Application 2

```python
from sklearn.metrics import silhouette_score

# Tester différentes valeurs de k
range_n_clusters = range(2, 10)
silhouette_scores = []

for n_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    labels = kmeans.fit_predict(X)
    silhouette_avg = silhouette_score(X, labels)
    silhouette_scores.append(silhouette_avg)
    print(f"Nombre de clusters = {n_clusters}, Silhouette Score = {silhouette_avg:.2f}")

# Tracer les scores de silhouette
plt.figure(figsize=(8, 5))
plt.plot(range_n_clusters, silhouette_scores, marker='o')
plt.title("Silhouette Score en fonction du nombre de clusters")
plt.xlabel("Nombre de clusters (k)")
plt.ylabel("Silhouette Score")
plt.grid()
plt.show()
```

8. Application 2

```
Nombre de clusters = 2, Silhouette Score = 0.62
Nombre de clusters = 3, Silhouette Score = 0.80
Nombre de clusters = 4, Silhouette Score = 0.88
Nombre de clusters = 5, Silhouette Score = 0.76
Nombre de clusters = 6, Silhouette Score = 0.59
Nombre de clusters = 7, Silhouette Score = 0.46
Nombre de clusters = 8, Silhouette Score = 0.34
Nombre de clusters = 9, Silhouette Score = 0.35
```



8. Application 2

6. *Interprétations:*

- Pour k=4, le coefficient de silhouette moyen est généralement autour de 0.88 (selon les données générées).
- Un score de silhouette proche de 1 indique un clustering de haute qualité, où les points sont bien groupés dans leurs clusters respectifs et bien séparés des autres clusters.
- Un score de 0.88 est donc considéré comme bon , car il est relativement proche de 1.
- Le nombre optimal de clusters correspond au point où le coefficient de silhouette est maximal.
- Dans l'exemple donné, le score de silhouette atteint son maximum pour k=4. Par conséquent, le nombre optimal de clusters est k=4 .

8. Application 2

le coefficient de silhouette est une métrique utile pour évaluer la qualité d'un clustering

- Le coefficient de silhouette mesure à quel point chaque point est similaire aux points de son propre cluster par rapport aux points des autres clusters.
- Il combine deux aspects importants du clustering :
  1. Cohésion **intra-cluster** : Les points d'un même cluster doivent être proches les uns des autres.
  2. Séparation **inter-clusters** : Les clusters doivent être bien distincts les uns des autres.
- Un score de silhouette proche de 1 indique que les points sont bien attribués à leur cluster et que les clusters sont bien séparés.
- À l'inverse, un score proche de 0 ou négatif indique que les points sont mal attribués ou que les clusters se chevauchent.
- Cette métrique est donc très utile pour évaluer la qualité globale d'un clustering sans avoir besoin de vérité terrain.