Université IBN TOFAIL – Kénitra
ECOLE NATIONALE DES SCIENCES APPLIQUEES

Cours Intelligence Artificielle

CH3: Machine Learning - Apprentissage Non-Supervisé: K-moyennes (K-means)

Par:

Hassan EL FADIL

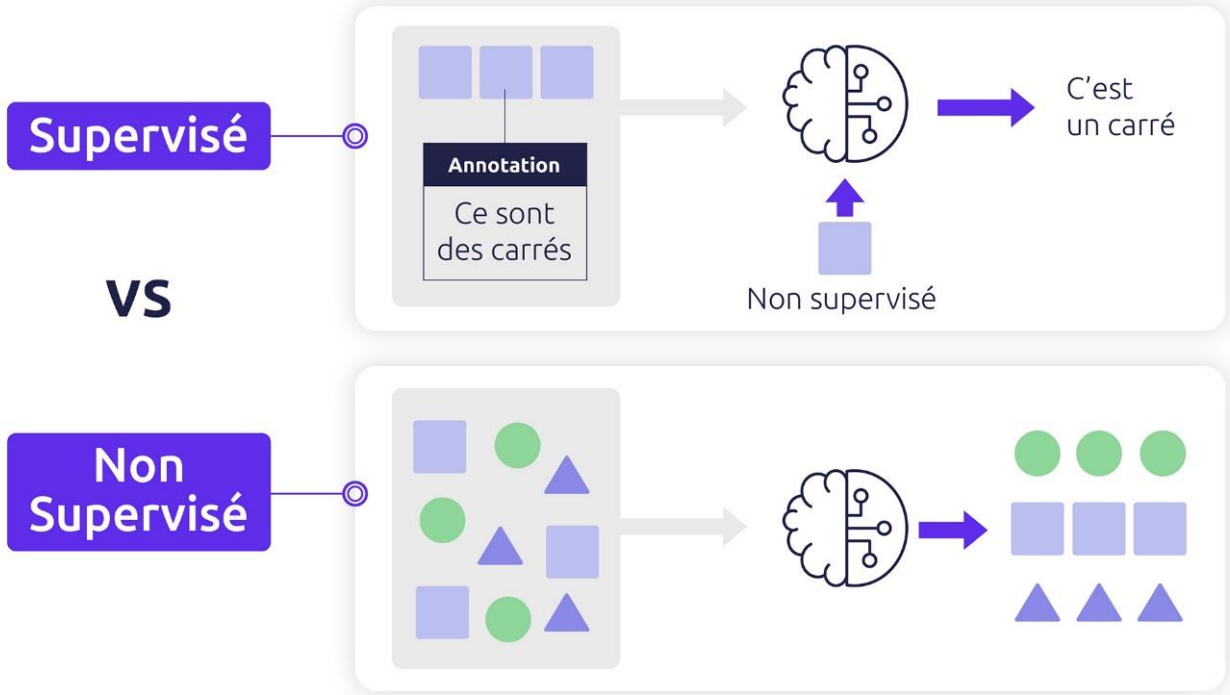
elfadil.h@gmail.com

1. Apprentissage Non-Supervisé

1.1. Définition

- Un **algorithme non-supervisé** est un type d'algorithme d'apprentissage automatique qui analyse des données sans avoir besoin de labels ou d'étiquettes prédéfinis.

- Contrairement aux algorithmes supervisés, où les données d'entraînement sont étiquetées avec des résultats connus (par exemple, une catégorie ou une valeur), les algorithmes non-supervisés cherchent à identifier des structures cachées, des motifs ou des relations dans des données non étiquetées.



1.2. Caractéristiques principales des algorithmes non-supervisés:

- **Aucune étiquette (non supervisé)** : Les données d'entrée n'ont pas de labels associés, ce qui signifie que l'algorithme doit apprendre directement à partir des données.
- **Exploration des données** : Ils sont utilisés pour explorer les données, trouver des modèles sous-jacents, des regroupements, des tendances ou des associations sans qu'on ait besoin d'une réponse explicite.

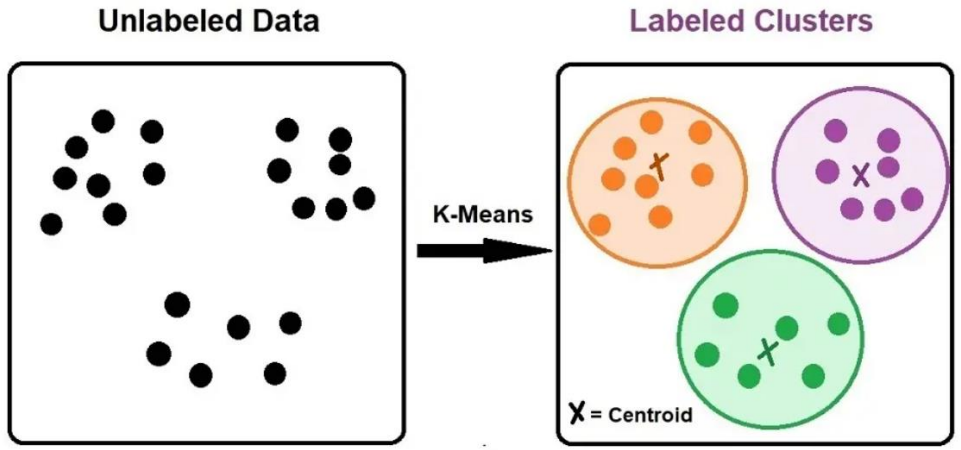
1.3. Exemples d'algorithmes non-supervisés:

- **K-means** : Regroupement de données en k clusters.
- **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise): Clustering basé sur la densité, capable de détecter des formes de clusters complexes.
- **PCA** (Principal Component Analysis) : Réduction de dimension des données pour les rendre plus faciles à analyser.
- **Autoencodeurs** (Autoencoders) : Utilisés pour la compression des données ou la détection d'anomalies.

2. Algorithme K-means

2.1. Objectif de l'algorithme

- Le clustering est une tâche d'apprentissage non supervisé qui vise à regrouper des données en clusters homogènes selon une métrique donnée.
- Parmi les algorithmes, **K-means** est l'un des plus utilisés en raison de sa simplicité et de son efficacité.



- L'objectif de K-means est de minimiser la **variance intra-cluster**, définie comme la somme des distances au carré entre chaque point et le centre de son cluster :

$$J = \sum_{j=1}^{k} \sum_{x_i \epsilon C_j} \|x_i - c_j\|^2$$

- J : Fonction objectif (variance intra-cluster).
- k : Nombre de clusters.
- C_j : Ensemble des points appartenant au cluster j.
- c_j : Centre du cluster j.

2.2. Étapes de l'algorithme

1. Initialisation :
- Choisir k centres initiaux $\{c_1, c_2, \dots, c_k\}$
- Méthodes possibles :
  - Aléatoire.
  - K-means++ : amélioration pour éviter les initialisations défavorables.

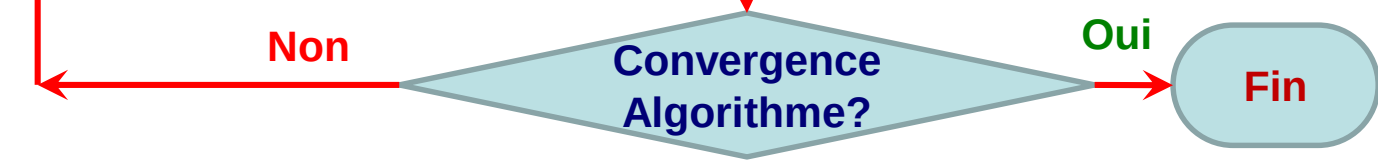
2. Assignation des points :
- Assigner chaque point x_i au cluster C_j le plus proche
- Calcul :

$$C(x_i) = \underset{j}{\operatorname{argmin}} \|x_i - c_j\|^2$$

3. Mise à jour des centres:
- Recalculer le centre de chaque cluster

$$c_j = \frac{1}{|C_j|} \sum_{x_i \epsilon C_j} x_i$$

Répéter les étapes 2 et 3 jusqu'à la convergence



- C_j : L'ensemble des points assignés au cluster j.
- $|C_j|$: Le nombre de points dans le cluster j.

2.3. Mesure de la qualité des clusters:

1. **Somme des erreurs quadratiques (SSE) :**

   Plus SSE est faible, meilleure est la qualité du clustering.

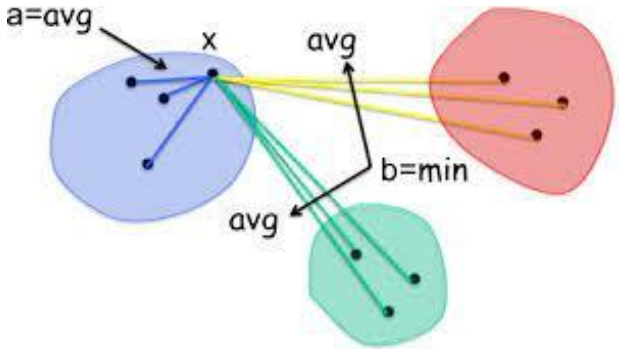
$$SSE = \sum_{j=1}^{k} \sum_{x_i \epsilon C_j} \|x_i - c_j\|^2$$

2. **Indice de silhouette:**

   Mesure la cohésion intra-cluster et la séparation inter-cluster

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- $S(i) = -1$ (mauvaise affectation, x_i est plus proche d'un autre cluster que du sien).
- $S(i) = 0$ (affectation ambiguë, x_i est équidistant de son cluster et d'un autre cluster).
- $S(i) = +1$ (bonne affectation, x_i est beaucoup plus proche de son cluster que des autres).



- $a(i)$: Distance moyenne entre x_i et les points de son cluster.
- $b(i)$: Distance moyenne entre x_i et les points du cluster le plus proche.

3. Application: Implémentation de K-means avec Google Colab

3.1. Objectif:

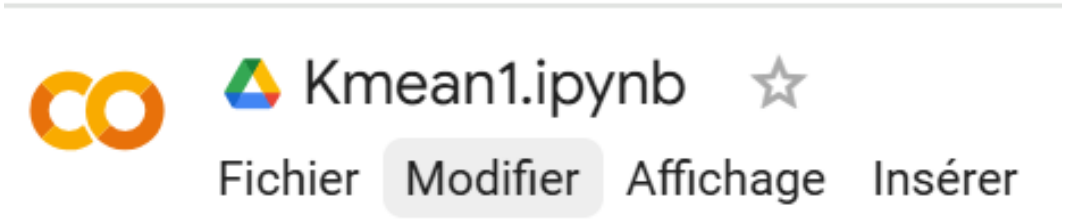
Objectif : Segmentation des clients

Nous avons un jeu de données simulé contenant les colonnes suivantes :

1. Âge des clients.
2. Revenu annuel.
3. Dépenses annuelles (en pourcentage du revenu).
4. Nous appliquerons l'algorithme K-means pour segmenter ces clients.

3.2. Nouveau Notebook:

- Ouvrir Google Colab et créer un **Nouveau Notebook**.
  Renommer le **Kmeans1.ipynb**



3.3. Code à utiliser dans Google Colab: …….. 1/3:

1 Importer les Bibliothèques Nécessaires:

```
# Installer les bibliothèques nécessaires (si non déjà installées)
# !pip install matplotlib scikit-learn pandas

# Importer les bibliothèques
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

3.3. Code à utiliser dans Google Colab: …….. 1/3:

2 Génération de Données et leur Affichage:

```
# Génération de données simulées
data = {
    'Age': [25, 40, 23, 60, 50, 22, 27, 45, 37, 58, 33, 41, 29, 54, 32],
    'Revenu Annuel (kDH)': [200, 350, 150, 800, 700, 180, 220, 400, 320, 750, 300, 380, 250, 720, 310],
    'Dépenses Annuelles (%)': [50, 40, 60, 30, 20, 70, 45, 35, 50, 25, 55, 30, 65, 15, 48]
}
df = pd.DataFrame(data)

# Afficher les premières lignes du jeu de données
print("Données initiales :")
print(df.head())
```

```
Données initiales :
   Age  Revenu Annuel (kDH)  Dépenses Annuelles (%)
0   25                  200                      50
1   40                  350                      40
2   23                  150                      60
3   60                  800                      30
4   50                  700                      20
```

3.3. Code à utiliser dans Google Colab: …….. 1/3:

3 Normalisation:

Standardisation (Z-score)

$$Valeur\ normalisée = \frac{X - \mu}{\sigma}$$

```
# Normalisation des données
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
```

4 Appliquer l'Algorithme K-means:

```
# Appliquer K-means
k = 3  # Nombre de clusters
kmeans = KMeans(n_clusters=k, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_data)
```

3.3. Code à utiliser dans Google Colab: …….. 1/3:

5 Affichages:

```
# Afficher les centres des clusters
print("\nCentres des clusters (après normalisation) :")
print(kmeans.cluster_centers_)

# Afficher le DataFrame avec les clusters
print("\nDonnées segmentées :")
print(df)
```

```
Centres des clusters (après normalisation) :
[[-0.9699941  -0.85304172  0.93814058]
 [ 0.04890727 -0.23013715 -0.1211852 ]
 [ 1.39385707  1.56723402 -1.25572937]]
```

Données segmentées :

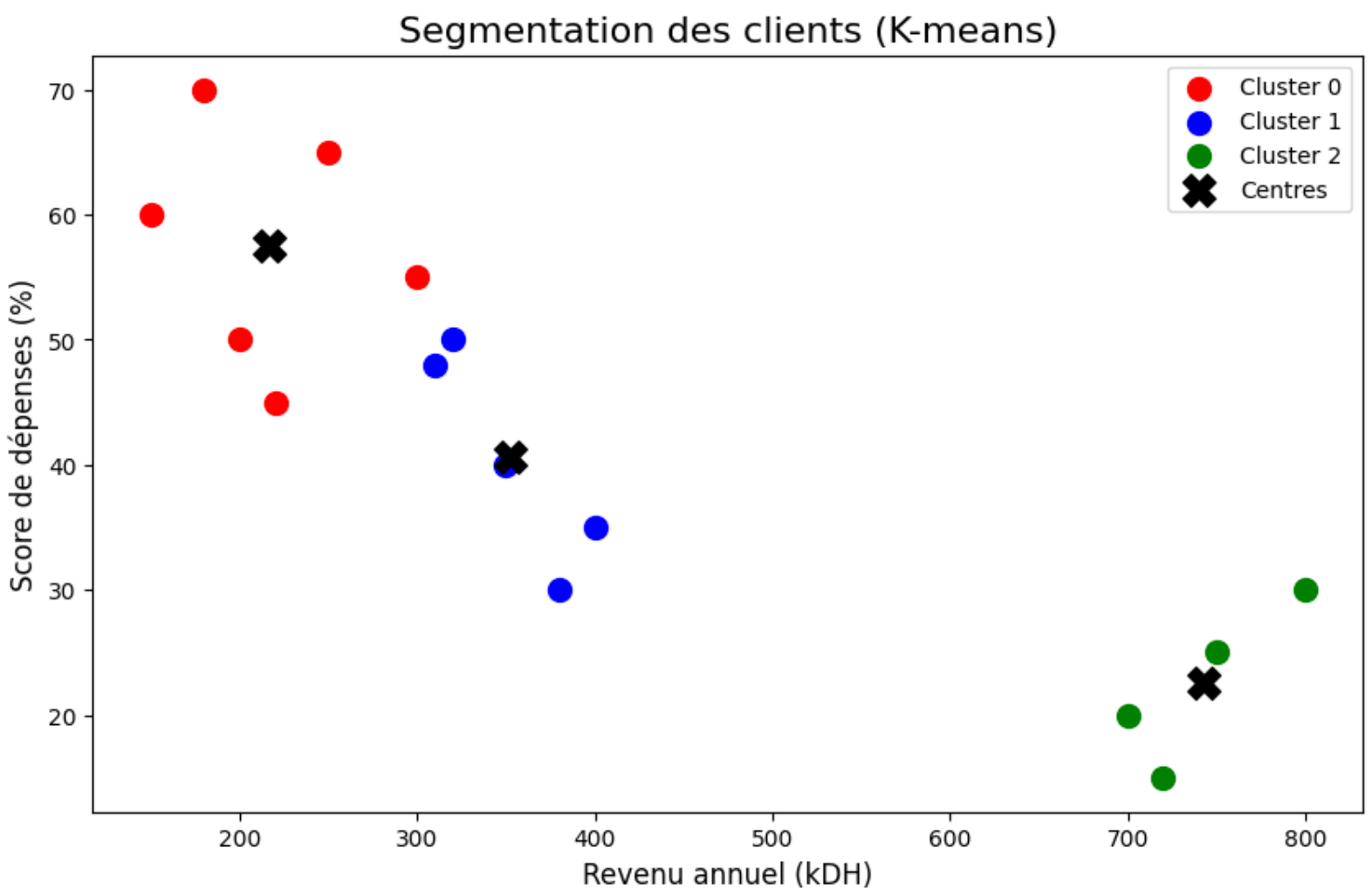
	Age	Revenu Annuel (kDH)	Dépenses Annuelles (%)	Cluster
0	25	200	50	0
1	40	350	40	1
2	23	150	60	0
3	60	800	30	2
4	50	700	20	2
5	22	180	70	0
6	27	220	45	0
7	45	400	35	1
8	37	320	50	1
9	58	750	25	2
10	33	300	55	0
11	41	380	30	1
12	29	250	65	0
13	54	720	15	2
14	32	310	48	1

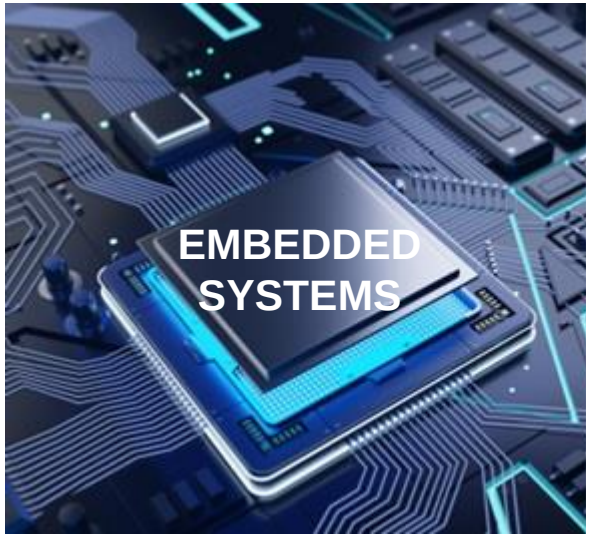
3.3. Code à utiliser dans Google Colab: …….. 1/3:

6 Visualisation des Clusters:

```
# Visualisation des clusters
plt.figure(figsize=(10, 6))
colors = ['red', 'blue', 'green']

for i in range(k):
    cluster_data = df[df['Cluster'] == i]
    plt.scatter(cluster_data['Revenu Annuel (kDH)'], cluster_data['Dépenses Annuelles (%)'],
                color=colors[i], label=f'Cluster {i}', s=100)

# Ajouter les centres de clusters
centers = scaler.inverse_transform(kmeans.cluster_centers_)
plt.scatter(centers[:, 1], centers[:, 2], c='black', marker='X', s=200, label='Centres')

plt.title('Segmentation des clients (K-means)', fontsize=16)
plt.xlabel('Revenu annuel (kDH)', fontsize=12)
plt.ylabel('Score de dépenses (%)', fontsize=12)
plt.legend()
plt.show()
```

… K-means : Application





FIN !
Questions ?