esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

Apprentissage Non Supervisé – Clustering

UP : Mathématiques

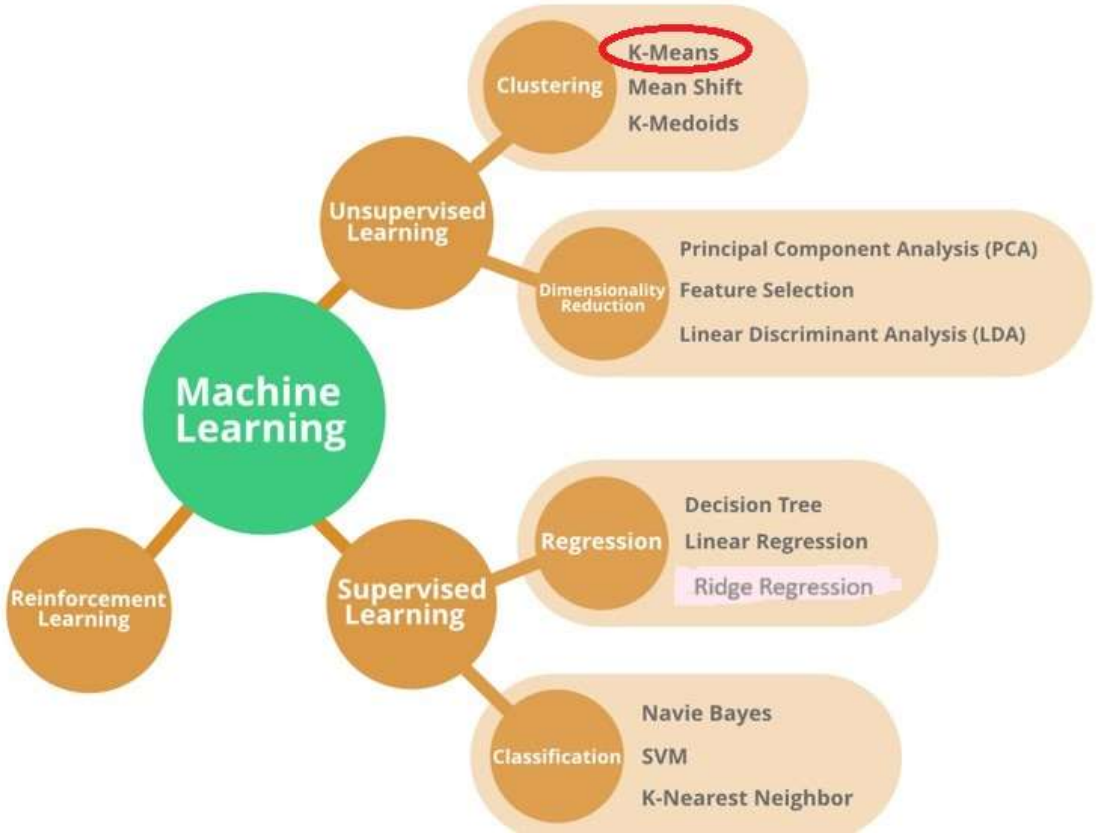
Année universitaire 2024-2025

United Nations Educational, Scientific and Cultural Organization
UNESCO Chair "Project-based learning" ESPRIT School of engineering, Tunisia
uniTwin
EUR-ACE® Délivrée par la Commission des Titres d'Ingénieur
CONFERENCE DES GRANDES ÉCOLES
cdio

PLAN

1. Rappel: Types d’apprentissage et algorithmes
2. Notion du regroupement (clustering)
3. Types de Clustering
4. Algorithme K-MEANS, Comment ça marche ?
5. Exemple
6. Quand utiliser l’algorithmes K-MEANS ?
7. Hyperparamètres de K-MEANS

RAPPEL: TYPES D'APPRENTISSAGE ET ALGORITHMES

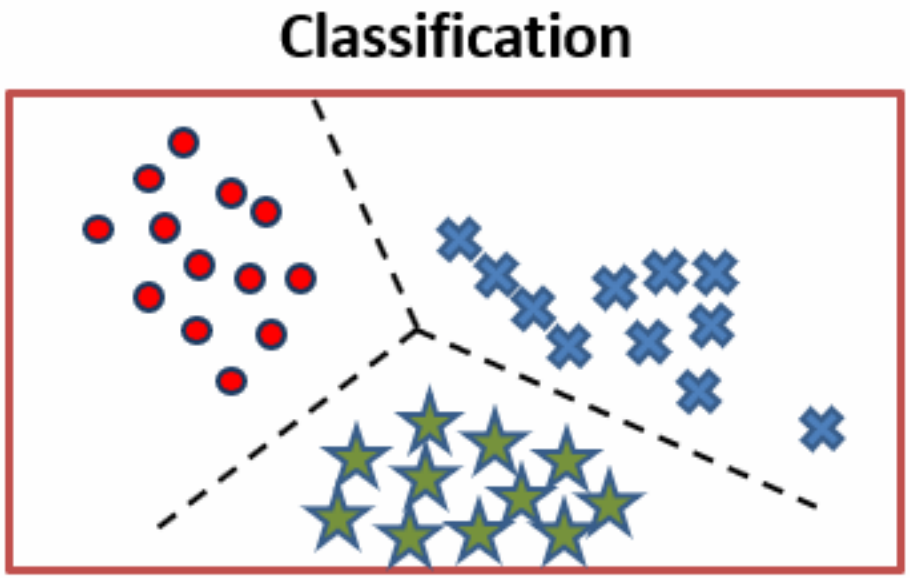


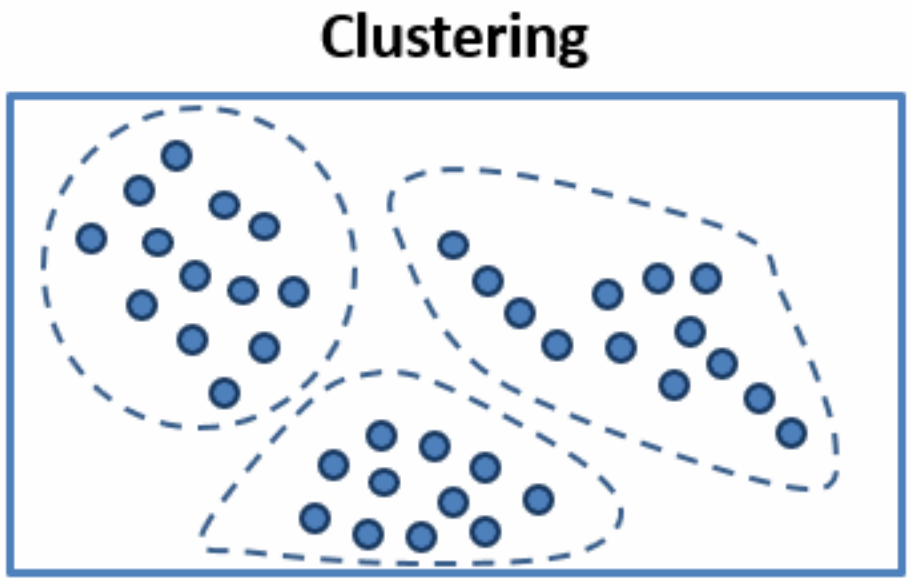
NOTION DU REGROUPEMENT (*CLUSTERING*)

- Lorsque l'on cherche à s'informer à propos de quelque chose, comme des livres, une approche possible pourrait être de chercher des groupes ou des collections significatives **plus ou moins semblables**. Par exemple, on pourra organiser ces derniers par auteurs, ou bien par genre. Une autre personne pourra choisir de les organiser par date de publication.

- La façon dont on choisit de regrouper ses éléments permettra de s'informer d'avantages à propos de chacune des observations individuellement. On pourra même créer des groupes qui n'existaient pas tout en combinant des critères bien choisis.

- De même en Machine Learning, nous devons séparer nos données en groupes pour les comprendre. Il existe deux façons pour séparer les données :
- **Si les données sont étiquetées**, il s'agit d'une classification, et les données sont regroupées selon leurs étiquettes (ou *labels*) → **Apprentissage supervisé**.
- **Si les données ne sont pas étiquetées**, il s'agit d'un regroupement (**clustering**). Le principe consiste à séparer les données en groupes homogènes ayant des caractéristiques communes. → **Apprentissage non supervisé**.





Le **regroupement**, ou **clustering**, est une méthode d'apprentissage non supervisée en machine learning qui consiste à regrouper des points de données en fonction de leurs similarités.

Ce principe est applicable dans divers domaines. Les applications incluent :

- **Segmentation des clients** : Utilisé en marketing pour regrouper les clients ayant des comportements d'achat similaires.
- **Analyse des réseaux sociaux** : Pour identifier des communautés ou des groupes d'utilisateurs ayant des intérêts communs.
- **Détection de fraudes** : Pour repérer des transactions ou des comportements anormaux.
- **Biologie** : Pour classer des gènes ou des espèces en fonction de leurs caractéristiques.
- **Imagerie médicale** : Pour segmenter des images en différentes régions d'intérêt.

TYPES DE CLUSTERING

- **Clustering basé sur le centroïde** : Utilise des centroïdes pour représenter chaque cluster. L'algorithme k-means est un exemple populaire.

- **Clustering basé sur la densité** : Identifie des zones de haute densité de points de données et les regroupe. DBSCAN est un algorithme couramment utilisé.

- **Clustering hiérarchique** : Crée une hiérarchie de clusters, souvent représentée sous forme d'arbre.

ALGORITHME K-MEANS

comment ça marche ?

K-MEANS

- L'algorithme **K-means** est une méthode de clustering non supervisée qui vise à partitionner un ensemble de données en **K** clusters, où chaque observation appartient au cluster avec le centroïde le plus proche.

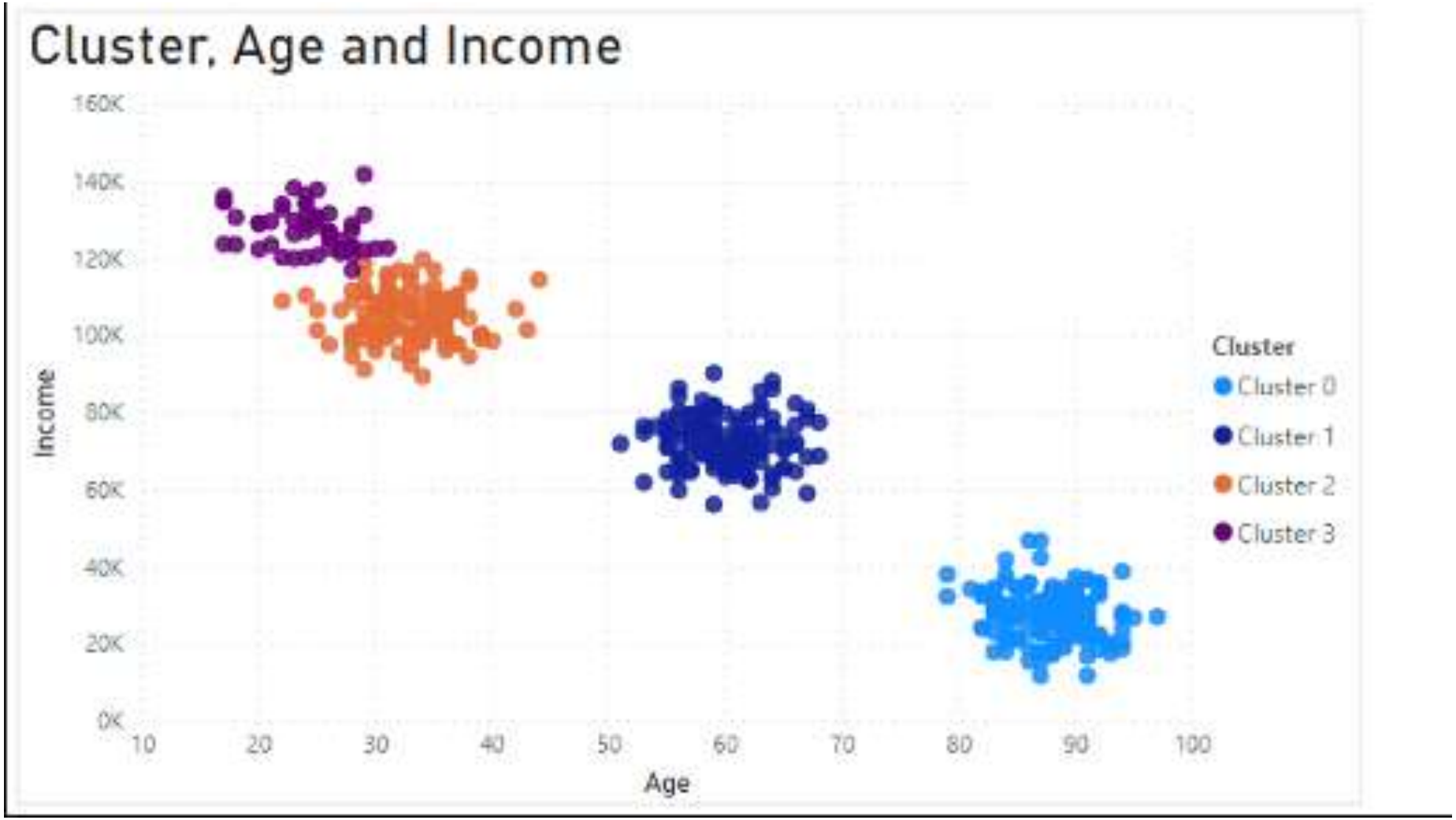
- Le nombre de groupes (clusters) représenté par la variable k n'est pas connu : c'est le Data Scientist qui est censé le préciser en fonction de son jeu de données, en estimant par lui-même le nombre de groupe qui existerait dans le jeu de données.

PRINCIPE DE FONCTIONNEMENT DE L'ALGORITHME K-MEANS

- Le cas d’usage le plus classique pour les méthodes de clustering est la **segmentation de bases de clients**. Nous allons considérer l'exemple d'une clientèle d'une banque.

- Les détails de chaque client, et à partir de ces renseignements, décident qu'elle offre devrait être faite à quel client.

- Cette méthode prendra énormément de temps et de main-d'œuvre. Ce qui sera plus logique est efficace, ce serait d'utiliser l'algorithme k-means pour regrouper les clients. On prendra deux caractéristiques en premier pour simplifier l'exemple : **l'âge** et le **revenu annuel**.


Cluster, Age and Income
160K
140K
120K
100K
80K
60K
40K
20K
0K
Income
10
20
30
40
50
60
70
80
90
100
Age
Cluster
Cluster 0
Cluster 1
Cluster 2
Cluster 3

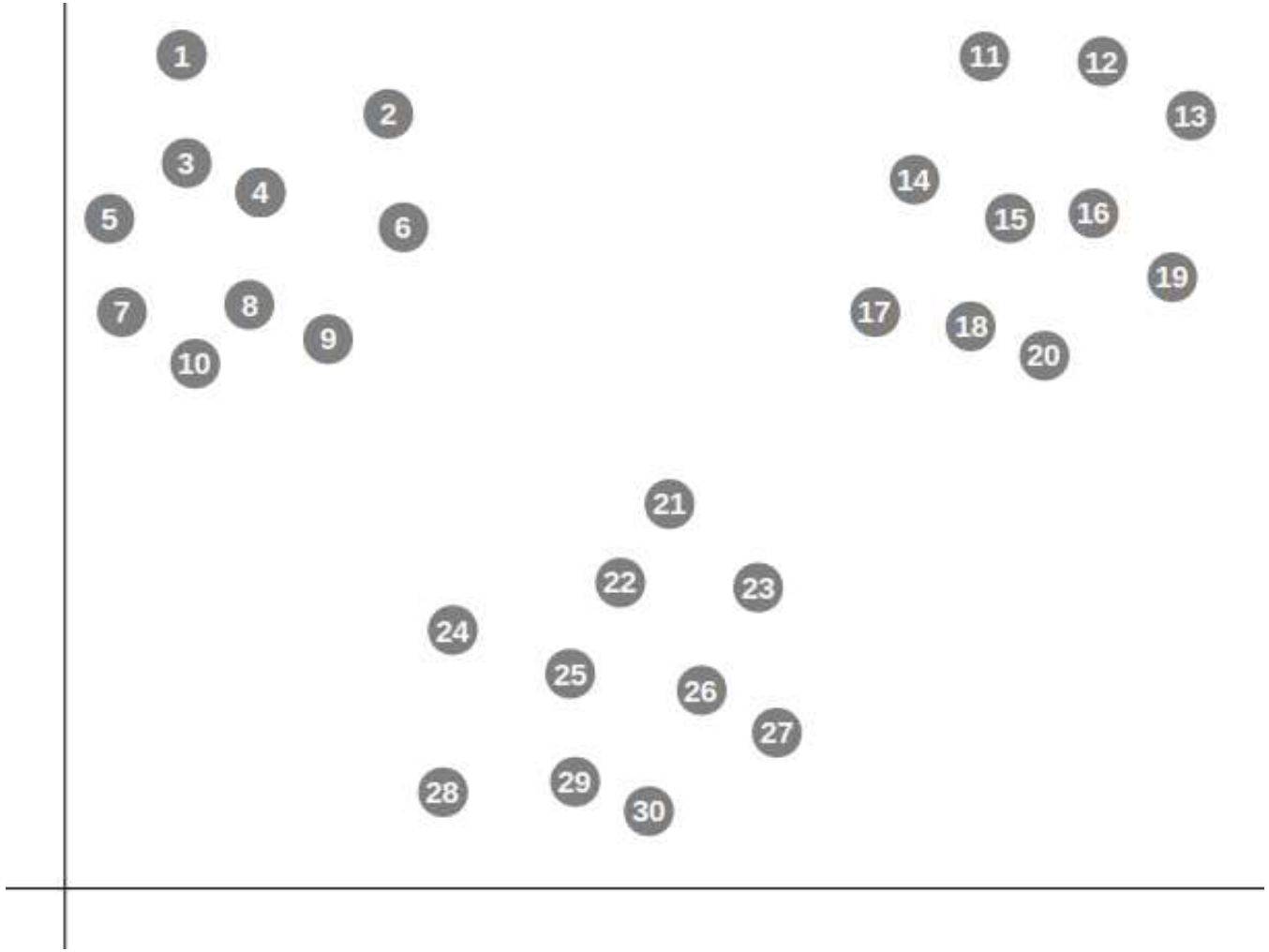
- Visuellement, les clients peuvent se séparer en 4 groupes distincts :
- Un premier groupe (en violet) : c'est le groupe des clients qui sont dans leurs vingtaines ayant un revenu annuel élevé (entre 120k et 140k).
- Un deuxième groupe (en orange) : c'est le groupe des clients qui sont dans leurs trentaines/quarantaines ayant un revenu annuel moyen (entre 80k et 120k).
- Un troisième groupe (en bleu foncé) : c'est le groupe des clients dont l'âge varie entre 40 et 60 ayant un revenu annuel moyen (entre 40k et 100K).
- Un dernier groupe (en bleu clair) : c'est le groupe des clients dont l'âge varie entre 80 et 100 ayant un revenu annuel bas (entre 0k et 60K).

La banque peut maintenant préparer quatre stratégies ou offres différentes, une pour chaque groupe au lieu de créer des stratégies différentes pour chaque client. Cela réduira l'effort ainsi que le temps.

- Mais comment les groupes sont-ils choisis réellement?
- Comment savoir quel élément à mettre dans tel et tel groupe ?
- Comment choisir le nombre de groupes optimale pour son jeu de données ?

DÉTAILS DE L'ALGORITHME K-MEANS

- Prenons un exemple simple avec les données réparties sous cette configuration.

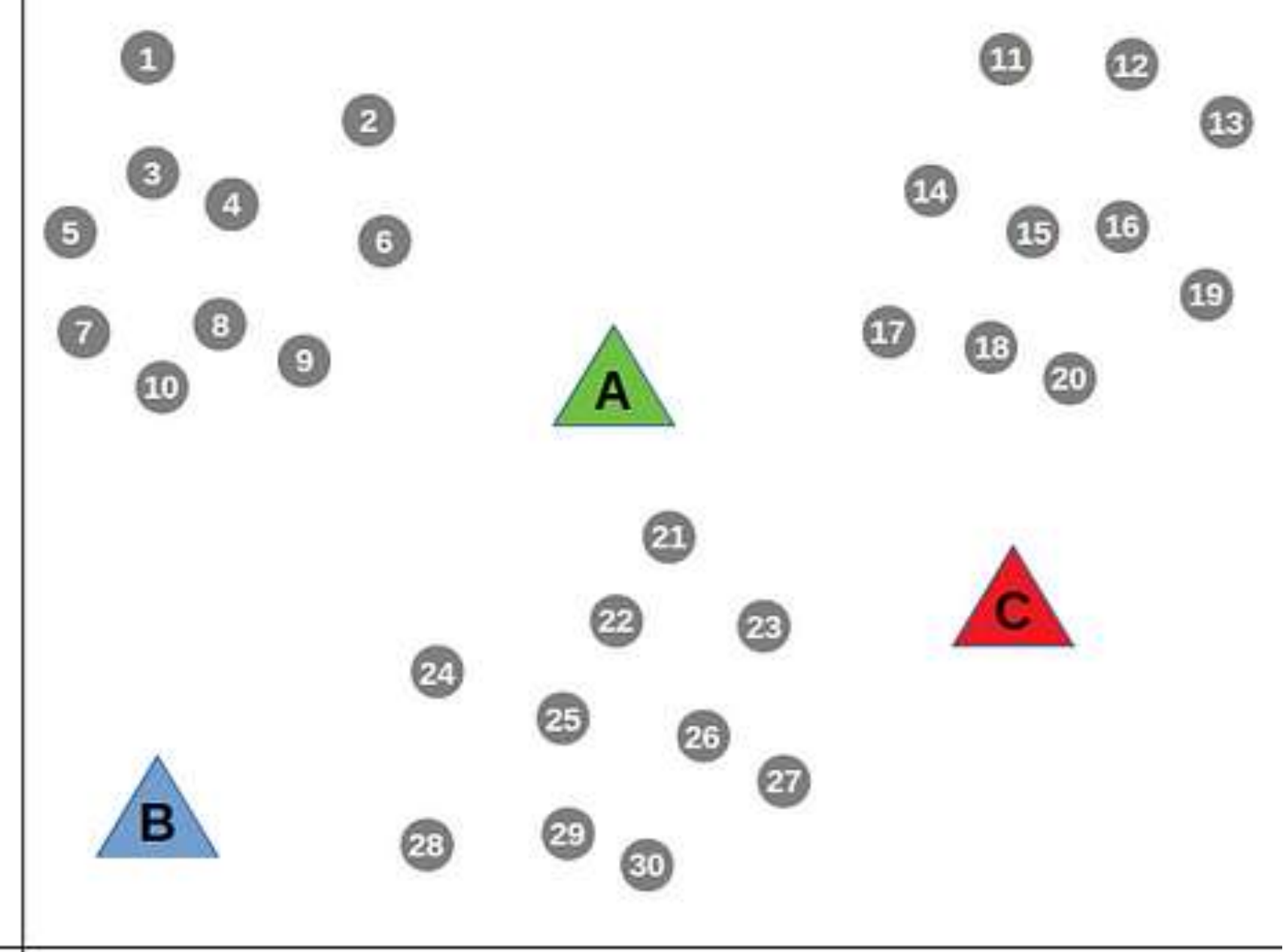


- Visuellement, un humain peut distinguer trois groupes, mais la machine ne peut pas les voir, car les données dessus sont simplement des points, et les nombres affectés n'ont aucune signification réelle.

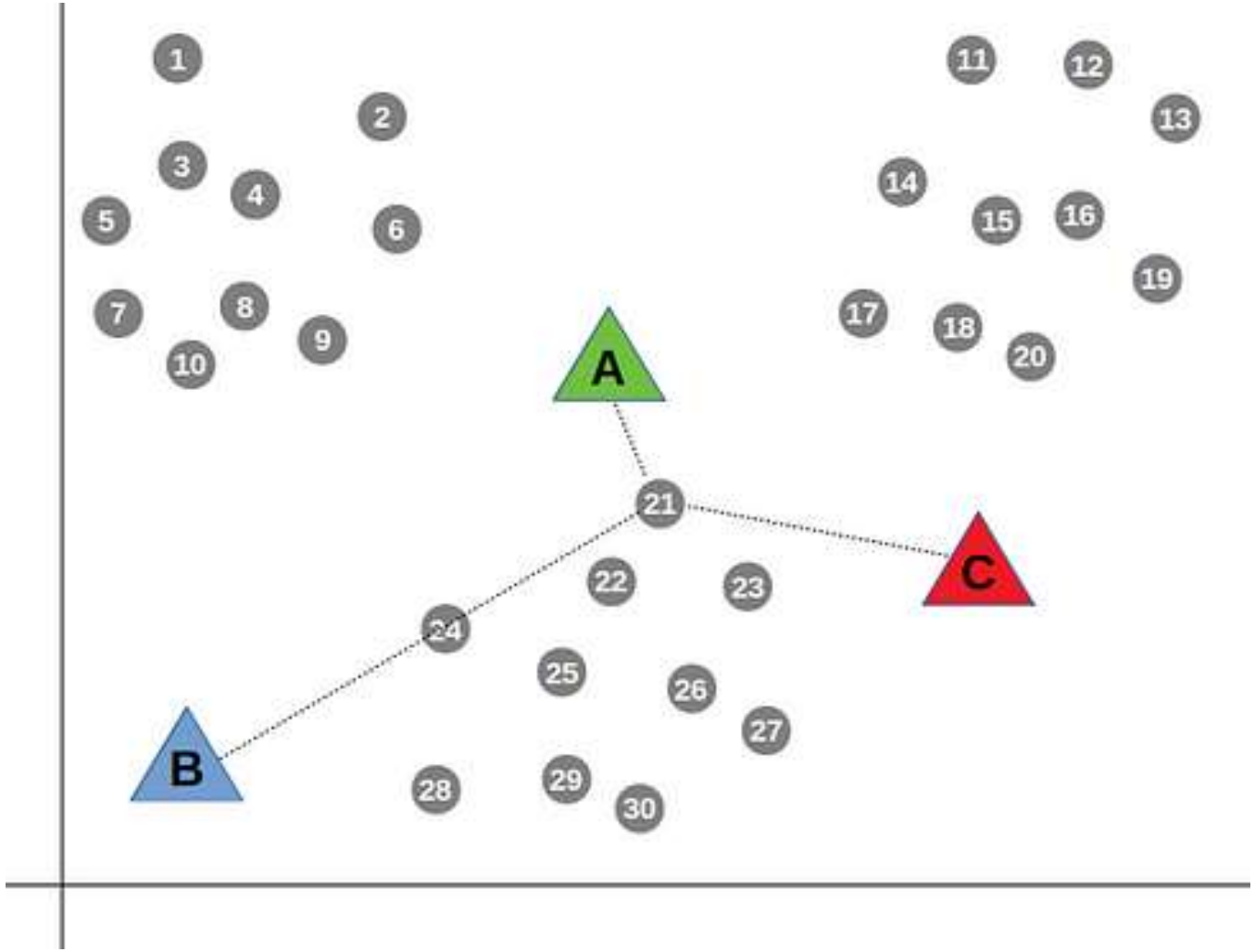
- On cherche alors à construire des groupes homogènes à partir de ces points.

- On commencera par spécifier manuellement la valeur de k, le nombre de groupes qui seront générés par l'algorithme. Le nombre k est généralement choisi aléatoirement au début (sauf en cas de connaissance approfondie sur le contexte), et on essayera ensuite de trouver la valeur de k optimale selon les premiers résultats que l'on obtiendrait, ou alors en utilisant des méthodes d'optimisation.

- Dans notre cas, nous choisissons k=3. Ces 3 points de données « prototypes » s'appellent des **centroïdes** et représentent les centres des groupes. Au début, ils seront choisis et placés aléatoirement comme suit.



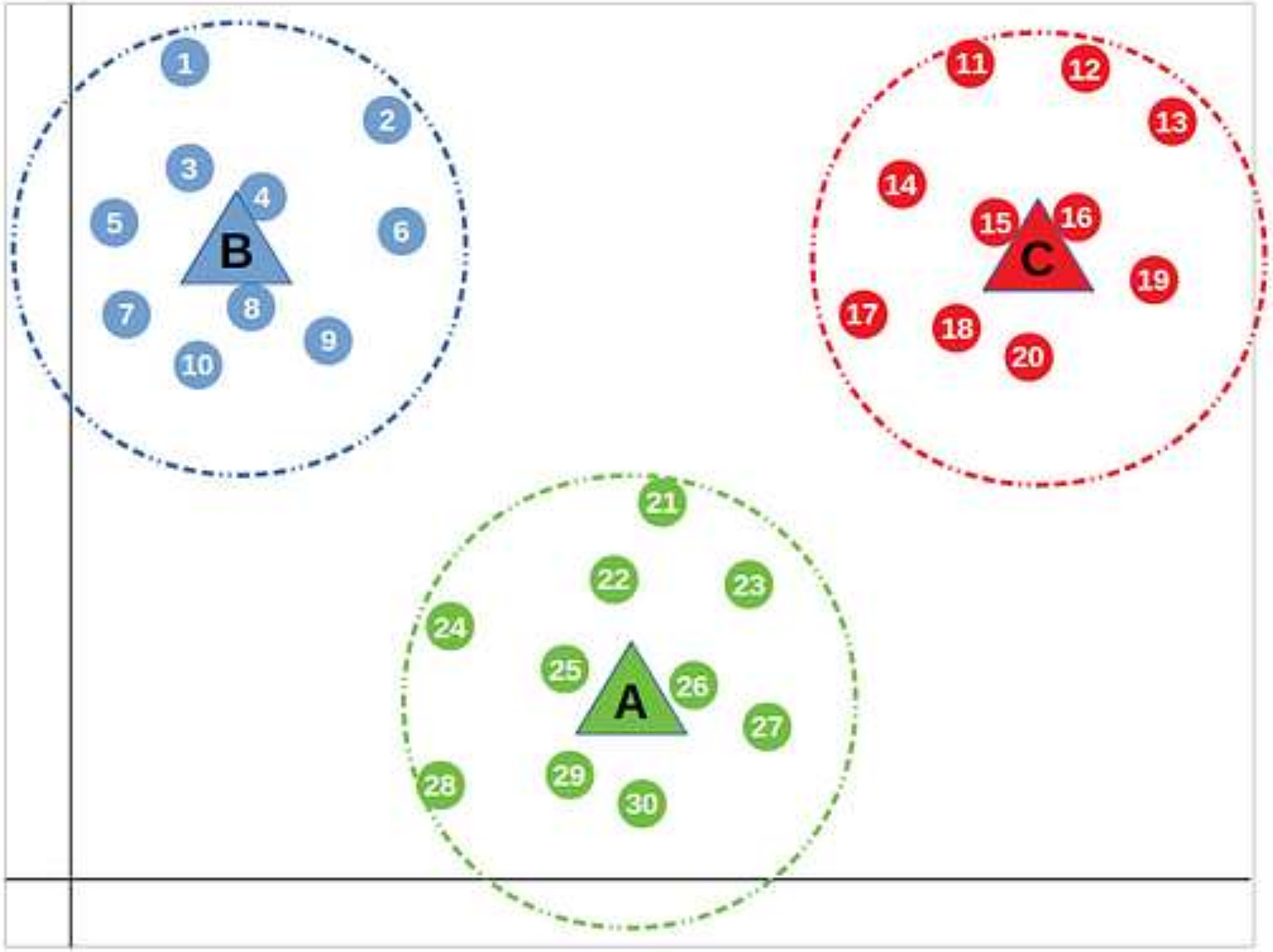
- L’algorithme passe par les points de données un par un, en mesurant la distance entre chaque point par rapport aux trois centroïdes (A, B et C).
- L’algorithme regroupe ensuite le point de données avec le centroïde le plus proche (c'est-à-dire celui le plus proche en termes de la distance).



PAR EXEMPLE, LE POINT N°21 APPARTIENDRA AU GROUPE DE A CAR IL EST PLUS PROCHE AU CENTROÏDE A QUE C.

- Puisque l'on vient d'affecter chaque point à un des 3 groupes, les centres ne sont plus les mêmes ! Il faut donc recalculer **les 3 nouveaux centroïdes** qui seront **les centres de gravité de chaque groupe** calculé dans l'étape précédente.

- On répète ces étapes jusqu’à ce que les nouveaux centroïdes soient stabilités, et ne changent plus de groupe.


1
2
3
4
5
6
B
7
8
9
10
11
12
13
14
15
16
C
17
18
19
20
21
22
23
24
25
26
A
27
28
29
30

QUELLE MÉTRIQUE UTILISER POUR ÉVALUER LA DISTANCE ENTRE LES POINTS ET LES CENTROÏDES ?

- Pour pouvoir regrouper un jeu de données en k groupes distincts, l'algorithme k-means a besoin d'un moyen pour **comparer le degré de similarité** entre les différentes observations.
- L'algorithme des k-means fait généralement intervenir la distance euclidienne. Soient deux groupes d'éléments p=$(p_1,\ldots,p_n)$ et q=$(q_1,\ldots,q_n)$, alors la distance d entre les points p et q se calcule avec cette formule.

$$d(p,q)=\sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

- Dans l'exemple précédent, nous avons calculé la distance euclidienne entre un point et chacun des centroïdes, puis nous l'avons associé au centroïde le plus proche, c'est-à-dire celui avec la plus petite distance. Ce calcul est effectué pour chacun des points du jeu de données.

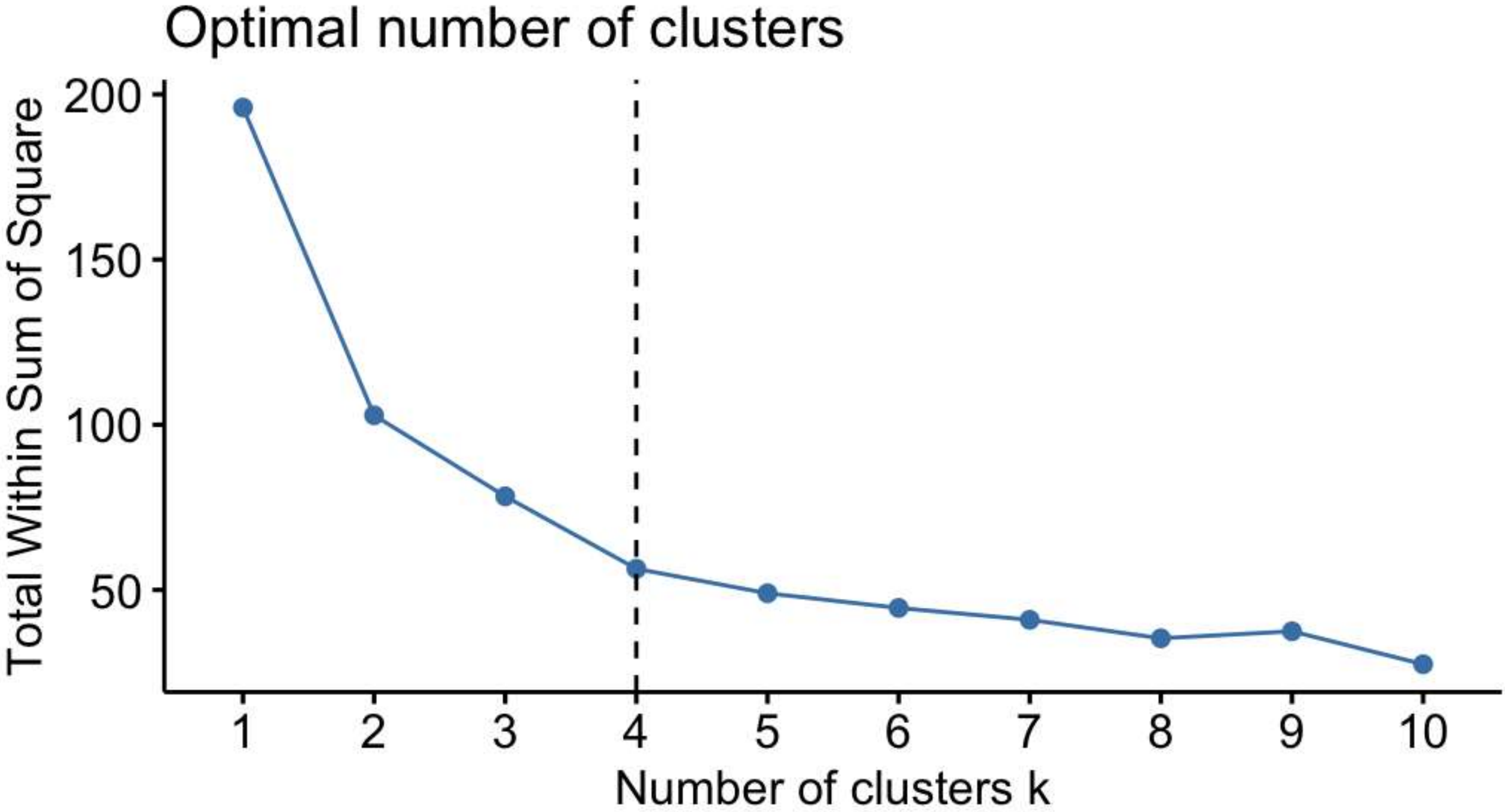
QUEL EST LE NOMBRE DE CLUSTERS À CHOISIR ?

- Lorsque la taille du jeu de données est importante, nous n'avons pas forcément d'hypothèses sur les données, et par la suite, nous ne pouvons pas choisir la valeur de k intuitivement. Comment faire dans ce cas ?

- 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

- Il existe une méthode populaire connue sous le nom de « *The Elbow method* » ou **la méthode du coude** qui est utilisée pour déterminer la valeur optimale de k. L'idée de base derrière cette méthode se décompose en plusieurs étapes.
- On lance l'algorithme k-means avec différentes valeurs de k.
- On calcule **la distance moyenne entre les points et leurs centroïdes respectives au carré**. On appellera cette valeur WSS (**Within Sum of Squares**)
- On place les différents nombres de clusters k en fonction de la valeur WSS sur un graphique.
- Nous obtenons alors une visualisation en forme de coude.


Optimal number of clusters
Total Within Sum of Square
200
150
100
50
1
2
3
4
5
6
7
8
9
10
Number of clusters k

- L’emplacement d’une courbe (coude) dans le graphique est généralement considéré comme un **indicateur du nombre approprié de groupes**. Les distances ne varient plus rapidement après cette inflexion, c'est-à-dire qu'une augmentation de k n'aura pas un grand effet sur les distances entre les points de données et leurs centroïdes respectives.

- Par conséquent, nous pouvons estimer à cet endroit, que l'on arrive à la valeur optimale de k.

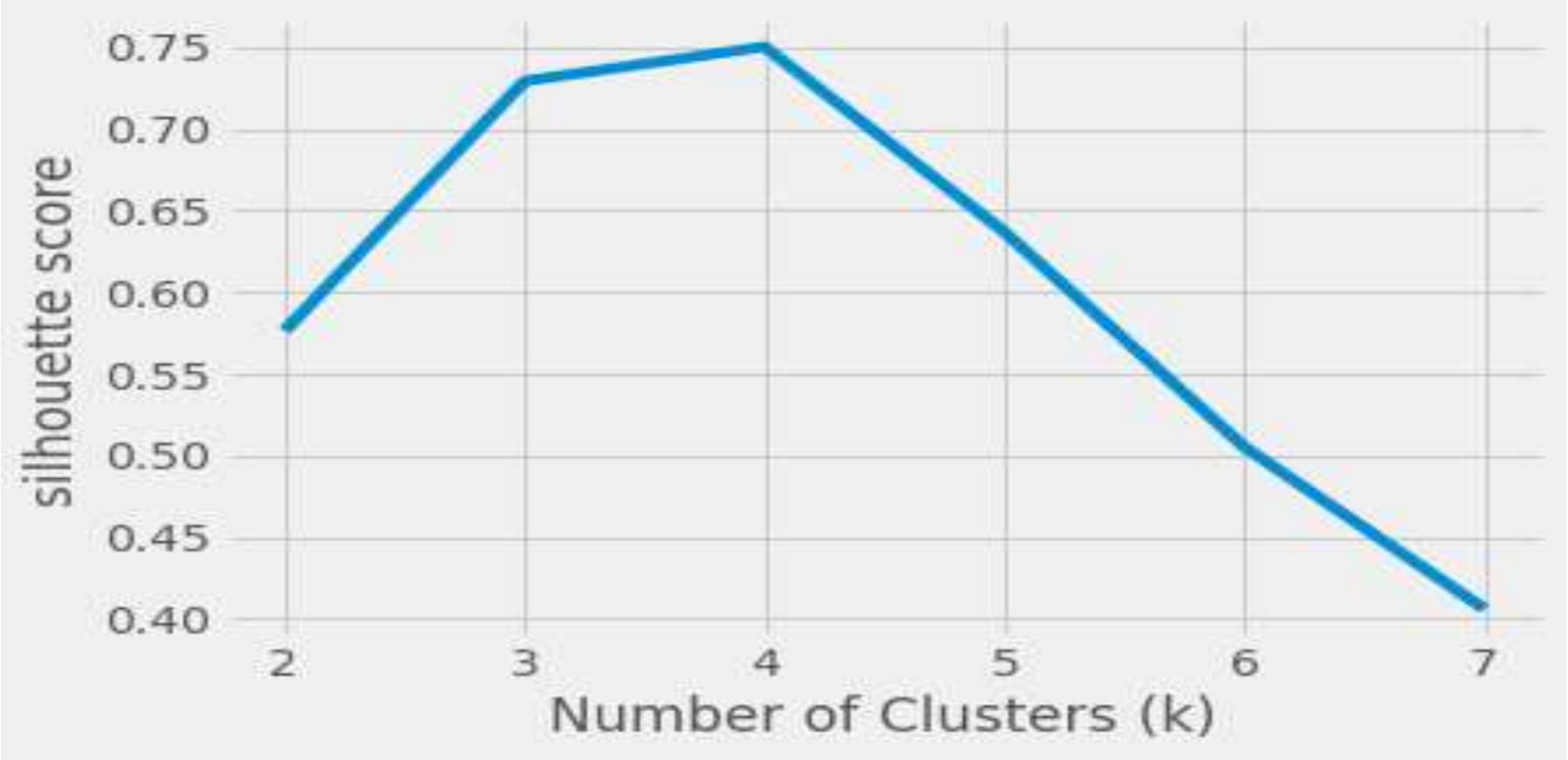
MÉTHODE DE SILHOUETTE

- La **méthode de silhouette** est également une méthode pour trouver le nombre optimal de clusters. Le **coefficient de silhouette** est une mesure de la similitude d'un point de données à l'intérieur d'un groupe par rapport à d'autres groupes.
- Ce coefficient doit être calculer pour chaque point du jeu de données. Pour trouver ce coefficient du i-ème point :
- On commence par calculer a_i: la distance moyenne du point i avec tous les autres points du même cluster.
- Ensuite, on calcule b_i : la distance moyenne du point i avec tous les points du cluster le plus proche de son cluster.
- Finalement, on calcule s_i (coefficient de silhouette) pour cet i-ème point en utilisant la formule ci-dessous.

$$s_i = \frac{(b_i - a_i)}{\max(b_i, a_i)}$$

- On choisira une plage de valeurs **k** candidates, puis on lancera l'algorithme k-means pour ces valeurs de **k** en calculant à chaque fois le coefficient de silhouette moyen.

- Dans l'exemple qui suit, on remarque que le coefficient de silhouette est maximal pour `k=4`, le nombre de clusters adéquats dans ce cas alors est 4.



EXEMPLE

- Soient les points donnés suivants:
- M0(1,1), M1(1,2), M2(2,1), M3(8,8), M4(8,9),M5(9,8),M6(0,8),M7(1,9),M8(2,8)
- Etape 1: Initialisation
- Choisissons aux Hazard 3 centroïdes initiaux: C1(1,1), C2(8,8) et C3(0,8)
- Etape2: Attribution aux clusters
- Pour chaque point, calculons la distance euclidienne à chacun des centroïdes,
- Ensuite , on assigne chaque point au centroïde le plus proche au sens de la distance euclidienne:

DISTANCES DES POINTS AUX CENTROÏDES:

points	C1(1,1)	C2(8,8)	C3(0,8)	Cluster assigné
(1,1)	0	9.9	7.07	C1
(1,2)	1	9.21	6.08	C1
(2,1)	1	9.21	7.28	C1
(8,8)	9.9	0	8	C2
(8,9)	10.63	1	8.06	C2
(0,8)	7.07	8	0	C3
(1,9)	8	7.07	1.41	C3
(9,8)	10.63	1	9	C2
(2,8)	7.07	6	2	C3

REGROUPEMENT INITIAL

On affecte les points aux centroïdes le plus proche:

- Cluster 1 (C1) : (1 , 1) , (1 , 2), (2 , 1)
- Cluster 2 (C2) : (8 , 8) , (8 , 9), (9 , 8)
- Cluster3 (C3): (0 , 8), (1 , 9), (2 , 8)

RECALCUL DES CENTROÏDES

- Pour chaque cluster , recalculons les centroïdes comme la moyenne des coordonnées des points de ce cluster:
- Nouveau C1:
- $C1=(\frac{1+1+2}{3}, \frac{1+2+1}{3})=(1.33, 1.33)$
- $C2=(\frac{8+8+9}{3}, \frac{8+9+8}{3})=(8.33, 8.33)$
- $C3=(\frac{0+1+2}{3}, \frac{8+9+8}{3})=(1, 8.33)$

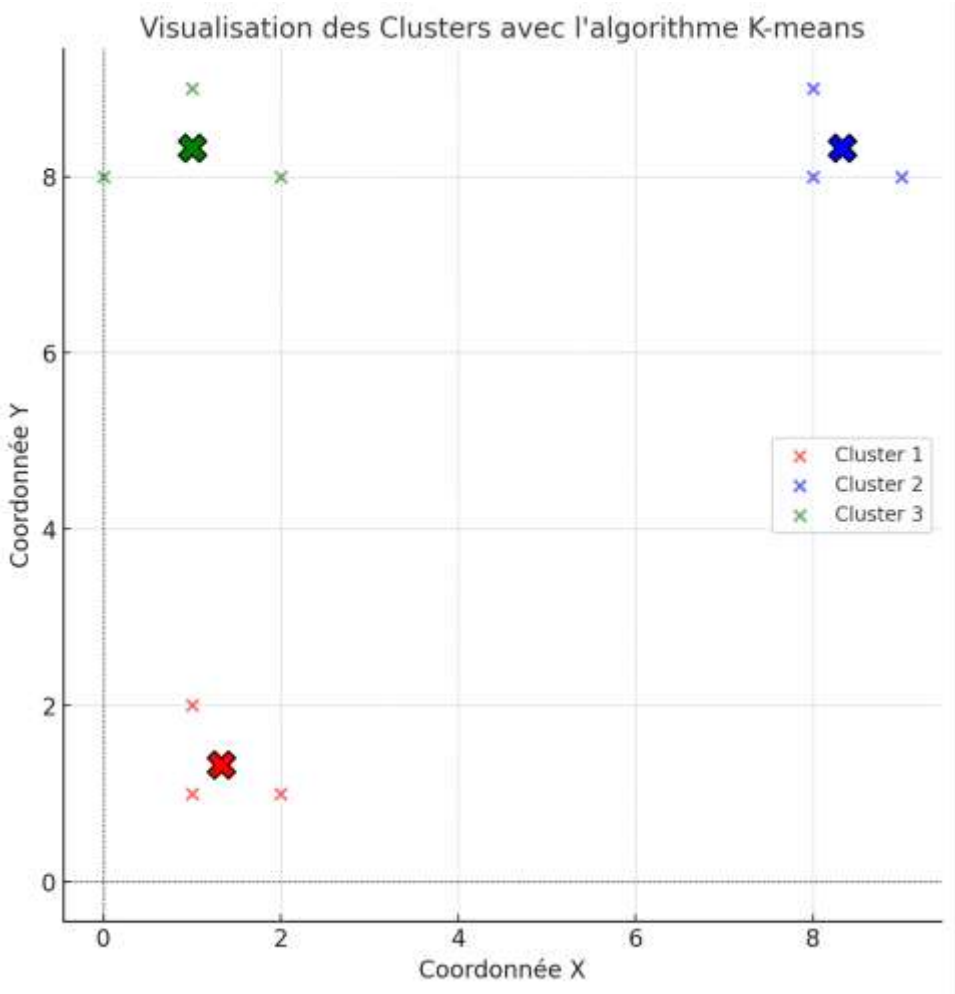
RÉATTRIBUTION DES POINTS

- Avec ces nouveaux centroïdes , on répète l'attribution des points aux clusters comme précédemment, On calcule les distances pour chaque point et réattribue les clusters, Cette étape se répète jusqu'à la convergence

points	C1(1.33 , 1.33)	C2(8.33 , 8.33)	C3(1 , 8.33)		Cluster assigné
(1,1)	0.47	10.37	7.33		C1
(1,2)	0.75	9.68	6.33		C1
(2,1)	0.75	9.68	7.4		C1
(8,8)	9.43	0.47	7.01		C2
(8,9)	10.16	0,74	7.03		C2
(9,8)	10,16	0.74	8.01		C2
(0,8)	6.8	8.33	1.05		C3
(1,9)	7.68	7.36	0.67		C3
(2,8)	6.7	6.34	1.05		C3

Regroupement des points :

- Les points sont à nouveaux regroupés selon la proximité des centroïdes recalculés
- Les clusters restent stables, et les points sont assignés de manière similaire à la première attribution,
- Après quelques itérations , on obtient:
- Cluster 1: (1,1),(1,2),(2,1)
- Cluster 2:(8,8), (8,9) , (9,8)
- Cluster 3: (0,8),(1,9),(2,8)



Condition d’arrêt:

On répète ces étapes jusqu’à ce que les nouveaux centroïdes se stabilisent, et ne changent plus de groupe.

QUAND UTILISER L'ALGORITHME K-MEANS ?

- L'algorithme k-means est un modèle prédictif très simple et non hiérarchique : **il ne nécessite donc pas beaucoup de temps d'exécution**. Il est aussi **rapide et efficace** en termes de coûts de calcul. Cela en fait la **solution meilleure dans le cas de données volumineuses**.

- Il ne peut être exécuté **qu'avec des données numériques**. Il est donc déconseillé de l'utiliser avec des données d'autres types.

- Dans la situation où le jeu de données est **de grande dimension**, c'est-à-dire lorsque le nombre de variables ou d'attributs est beaucoup plus élevé que le nombre d'observations, un modèle de Machine Learning aura besoin d'un énorme ensemble d'entrainement (*train set*) pour apprendre. Cet algorithme est le meilleur choix dans ce cas

- Enfin, les k-means sont fréquemment utilisés dans le domaine de **l'imagerie médicale**, la segmentation des couleurs et parfois pour la détection des anomalies.
- Néanmoins, il y a **aucune garantie de convergence** avec l'algorithme des k-means. En effet, puisque l'étape d'initialisation calcule des centroïdes de manière aléatoire, il se peut qu'aucun groupe ne se stabilise à long terme, et donc qu'aucune convergence ne se produire.

- L'algorithme des k-means est l'un des **algorithmes de clustering les plus populaires** et généralement la première chose que les Data Scientists appliquent pour résoudre des tâches de regroupement. Il est non seulement **simple et facile** à comprendre mais aussi **rapide et économique** en matière de coût de calcul.

HYPERPARAMÈTRES DE K-MEANS:

***CLASS* SKLEARN.CLUSTER.KMEANS(*N_CLUSTERS*=*8*, *, *INIT*='*K-MEANS++*', *N_INIT*='*AUTO*', *MAX_ITER*=*300*, *TOL*=*0.0001*, *RANDOM_STATE*=*NONE*, *COPY_X*=*TRUE*, *ALGORITHM*='*LLOYD*')**

- ***n_clusters:*** type int, default=8

Le nombre de clusters à former ainsi que le nombre de centroïdes à générer.

- ***init*=*'k-means++'***

sélectionne les centroïdes initiaux des clusters en utilisant un échantillonnage basé sur une distribution de probabilité empirique de la contribution des points à l'inertie globale,

- ***n_init* :'auto' ou int, default='auto'**

Nombre de fois où l'algorithme k-means est exécuté avec différentes graines de centroïdes. Le résultat final est la meilleure sortie de n_init exécutions consécutives en termes d'inertie. Il est recommandé de procéder à plusieurs exécutions pour les problèmes à haute dimension peu denses.

- ***max_iter****: type(int), default =300*

Nombre maximal d'itérartions

- **tol**: type(*float), default=1e-4*

précision pour arrêter l'algorithme

- ***Random _state**: type(int), RandomState instance or None, default=None*

Détermine la génération de nombres aléatoires pour l'initialisation du centroïde. Utilisez un int pour que le hasard soit déterministe.

- ***copy_x***: bool, default=True

Lors du pré-calcul des distances, il est plus précis numériquement de centrer d'abord les données. Si copy_x est True (par défaut), les données originales ne sont pas modifiées. Si False, les données originales sont modifiées et remises en place avant le retour de la fonction,

- ***Algorithme*** :{"lloyd", "elkan"}, default="lloyd"

Choisir comment on va faire les calculs:

'lloyd': Méthode standard qui calcule directement les distances et met à jour les centres,

'elkan' : Une version optimisée qui va plus vite en évitant certains calculs inutiles. Utiles pour les grands ensembles de données