



ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES D'AGADIR

Méthodes d'Analyse de données : Kmeans et Classification Ascendante Hiérarchique

ANALYSE DE DONNÉES

Élèves :

Daniel Kodjovi ANONWODJI
1ère année Finance et Ingénierie
Décisionnelle

Enseignant :

M. Brahim El ASRI



Résumé

Notre projet consiste en l'implémentation de techniques d'analyse de données, notamment la Classification Ascendante Hiérarchique (CAH) et l'algorithme K-means, en utilisant Python. Le projet est organisé en différentes étapes, avec une partie théorique expliquant les concepts de base de la CAH et de K-means, suivie d'une mise en œuvre pratique en Python.

La méthode de la CAH est décrite en détail, avec différentes stratégies d'agrégation des clusters telles que la méthode de Ward, la stratégie du saut minimum et du saut maximum. L'implémentation de la CAH est réalisée en utilisant des étapes telles que la création d'une matrice de distance, le choix des éléments les plus éloignés comme premiers centres, et l'affectation des individus aux clusters en fonction de la distance minimale.

En ce qui concerne l'algorithme K-means, une méthode d'initialisation par le mal classé est présentée, où les individus sont ajoutés aux centres de manière itérative en choisissant ceux qui sont mal classés. L'implémentation de K-means comprend également une étape de choix aléatoire des centres initiaux, suivie d'une itération de l'algorithme jusqu'à la convergence.

En plus de l'implémentation des algorithmes, une interface utilisateur est développée en utilisant Flask, permettant aux utilisateurs de choisir la méthode de classification, la mesure de dissimilarité et de charger un jeu de données pour l'analyse. Des exemples de résultats de classification sont présentés pour illustrer l'utilisation de l'interface.

En conclusion, le projet démontre l'importance de l'analyse de données dans divers domaines et met en évidence l'application pratique des techniques de clustering pour découvrir des modèles cachés dans les données et prendre des décisions stratégiques.

Table des matières

I	Le clustering	5
1	Définition et principe	5
2	Les différentes méthodes de clustering	5
II	Les distances utilisées dans notre projet	7
1	La distance euclidienne	7
1.1	Formule en deux dimensions	7
1.2	Formule en n dimensions	7
2	Distance de Manhattan	7
2.1	Formule en deux dimensions	7
2.2	Formule en n dimensions	8
III	La Classification Ascendante Hiérarchique (CAH)	8
1	La méthode	8
1.1	La méthode CAH	8
1.2	Domaines d'application	8
2	Algorithme de CAH	9
3	Choix de l'indice d'agrégation	10
3.1	Méthode de Ward	10
3.2	Stratégie du saut minimum (single linkage)	10
3.3	Stratégie du saut maximum (complete linkage)	10
4	Implémentation avec python	10
4.1	Le choix du langage	10
4.2	Le code sur VsCode	11
IV	Kmeans ou Kmoyennes	13
1	La méthode et ses domaines d'apllcation	13
1.1	La méthode Kmeans	13
1.2	Domaines d'application	13
2	Algorithme de Kmeans	13
3	Convergence de l'Algorithme	14
3.1	Théorème de convergence des K-means	15
4	Quelques versions améliorée de Kmeans	15
4.1	Global Kmeans	15
4.2	Principe de l'initialisation par le mal classé	16
5	Implémentation avec python	18
5.1	Initialisation aléatoire	19
5.2	Initialisation par le mal classé	20
5.3	Global Kmeans	21
V	Réalisation d'un test avec notre Application Python Flask (et Exemple de données du cours)	22
1	Présentation de l'interface	22
2	Présentation du jeu de données et Chargement depuis ordinateur	23
3	Visualisation des résultats	24

Table des figures

III.1 Ma classe Cluster	11
III.2 Ma classe des Fonctions	11
IV.1 Exemple	16
IV.2 Ma classe des Fonctions	17
V.1 Interface utilisateur	22
V.2 Interface utilisateur'	23
V.3 Jeu de données	23
V.4 Recherche de la Table et Récupération de valeurs	24
V.5 Chargement fichier	24
V.6 Les entrées 1	25
V.7 Les entrées 2	25
V.8 Résultats avec Distance Euclidienne	26
V.9 Résultats avec Distance de Manatthan	26

Liste des tableaux

Listings

Introduction

L'analyse de données occupe une place de plus en plus cruciale dans de nombreux domaines, de la finance à la santé en passant par le marketing et la recherche scientifique. Identifier des modèles, segmenter des données et extraire des informations significatives à partir de vastes ensembles de données sont des défis de taille pour les professionnels de l'analyse de données.

Dans ce contexte, notre projet se concentre sur l'implémentation de deux techniques d'analyse de données largement utilisées : la Classification Ascendante Hiérarchique (CAH) et l'algorithme K-means. Ces deux méthodes de clustering permettent de regrouper des données similaires ensemble, facilitant ainsi leur interprétation et leur analyse.

L'objectif de ce projet est double : d'une part, comprendre les principes théoriques sous-jacents à la CAH et à K-means, et d'autre part, mettre en œuvre ces techniques en utilisant le langage de programmation Python. Nous explorerons les différentes étapes de chaque algorithme, de la préparation des données à l'interprétation des résultats, en passant par l'implémentation des algorithmes eux-mêmes.

En plus de l'implémentation des algorithmes, nous développerons également une interface utilisateur conviviale permettant aux utilisateurs d'expérimenter avec différents jeux de données et paramètres d'analyse. Cette interface offrira une expérience interactive pour explorer les résultats de la CAH et de K-means, facilitant ainsi la compréhension et l'interprétation des clusters générés.

Grâce à ce projet, nous espérons non seulement acquérir une compréhension approfondie des concepts fondamentaux de l'analyse de données, mais aussi fournir un outil pratique et efficace pour les professionnels souhaitant explorer et analyser leurs propres ensembles de données.

I Le clustering

1 Définition et principe

Le clustering, également appelé regroupement en français, est une technique d'analyse de données non supervisée visant à regrouper des données similaires dans des ensembles distincts appelés clusters. Son principe repose sur la mesure de la similarité ou de la dissimilarité entre les observations et la formation de groupes homogènes en fonction de ces mesures.

Le principe du clustering peut être illustré comme suit :

1. **Initialisation** : Le processus commence par l'initialisation de l'algorithme de clustering. Selon la méthode utilisée, cela peut impliquer le choix aléatoire de centres de clusters (comme dans K-means) ou la construction d'une hiérarchie de clusters à partir de la matrice de distances entre les observations (comme dans la CAH).

2. **Assignment des observations aux clusters** : Les observations sont assignées à des clusters en fonction de leur similarité avec les centres de clusters ou avec d'autres observations. Dans le cas de K-means, chaque observation est assignée au cluster dont le centre est le plus proche en termes de distance euclidienne. Dans la CAH, les observations sont assignées à des clusters en fonction de leur similarité ou de leur dissimilarité calculée à partir de la matrice de distances.

3. **Mise à jour des centres de clusters** : Une fois les observations assignées à des clusters, les centres de clusters sont mis à jour en fonction des observations qui leur sont assignées. Dans K-means, les centres de clusters sont recalculés comme la moyenne des observations appartenant à chaque cluster. Dans d'autres méthodes, comme la CAH, les centres de clusters peuvent être déterminés à différents niveaux de la hiérarchie de clusters.

4. **Répétition des étapes précédentes** : Les étapes d'assignation et de mise à jour sont répétées itérativement jusqu'à ce qu'un critère d'arrêt soit atteint, tel que la convergence des centres de clusters ou un nombre maximal d'itérations.

5. **Évaluation des clusters** : Une fois que les clusters ont été formés, il est souvent nécessaire d'évaluer leur qualité. Cela peut être fait en examinant la cohérence interne des clusters (homogénéité des observations au sein d'un cluster et séparation entre les clusters) ou en comparant les clusters à des informations externes si elles sont disponibles.

Exemple : Supposons que nous ayons un ensemble de données sur les clients d'un magasin en ligne, comprenant des informations telles que le montant total des achats, la fréquence d'achat, etc. En utilisant le clustering, nous pourrions regrouper ces clients en segments homogènes, tels que les clients à faible fréquence d'achat et montant élevé, les clients à fréquence d'achat élevée mais montant faible, etc. Cela permettrait au magasin de mieux comprendre ses clients et de personnaliser ses stratégies de marketing en fonction des caractéristiques de chaque segment de clients.

2 Les différentes méthodes de clustering

Il existe plusieurs méthodes de clustering, chacune ayant ses propres caractéristiques et domaines d'application. Voici quelques-unes des méthodes les plus couramment utilisées :

1. **K-means** : Une méthode populaire qui divise les données en k clusters en minimisant la variance intra-cluster.

2. **CAH (Classification Ascendante Hiérarchique)** : Construit une hiérarchie de clusters en fusionnant progressivement des clusters similaires.
3. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** : Identifie les clusters basés sur la densité des points dans l'espace.
4. **Clustering hiérarchique basé sur la densité** : Des méthodes comme OPTICS étendent DBSCAN pour détecter des clusters de densités différentes.
5. **Clustering basé sur les modèles** : Utilise des distributions probabilistes pour trouver les clusters.
6. **Clustering basé sur la grille** : Divise l'espace de données en une grille de cellules.
7. **Clustering spectral** : Utilise la structure spectrale des données pour regrouper des observations similaires.

Ces méthodes offrent une variété d'approches pour regrouper les données en fonction de leurs caractéristiques et de leurs structures.

Dans la suite nous allons aborder en détails les méthodes Kmeans et de la Classification hiérarchique.

II Les distances utilisées dans notre projet

1 La distance euclidienne

La distance euclidienne, également connue sous le nom de distance L^2 ou distance euclidienne standard, est une mesure de distance entre deux points dans un espace euclidien. Elle est définie comme la longueur de la ligne droite la plus courte entre les deux points dans cet espace.

1.1 Formule en deux dimensions

La formule de la distance euclidienne entre deux points (x_1, y_1) et (x_2, y_2) dans un plan cartésien est donnée par :

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

1.2 Formule en n dimensions

Dans un espace de dimension n , la distance euclidienne entre deux points $\mathbf{p} = (p_1, p_2, \dots, p_n)$ et $\mathbf{q} = (q_1, q_2, \dots, q_n)$ est donnée par :

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

Cette formule mesure la longueur de la ligne droite reliant les deux points dans l'espace n -dimensionnel. La distance euclidienne est souvent utilisée dans de nombreux domaines, tels que la géométrie, l'analyse de données, la vision par ordinateur, et bien d'autres, en raison de sa simplicité et de son interprétation intuitive. Elle satisfait les propriétés d'une distance, notamment la non-négativité, l'identité des indiscernables, la symétrie et l'inégalité triangulaire.

2 Distance de Manhattan

La distance de Manhattan, également connue sous le nom de distance L^1 ou distance de la ville, est une mesure de distance entre deux points dans un espace euclidien. Contrairement à la distance euclidienne qui mesure la distance la plus courte entre deux points en suivant une ligne droite, la distance de Manhattan mesure la somme des longueurs des segments horizontaux et verticaux parcourus pour se déplacer d'un point à un autre.

2.1 Formule en deux dimensions

La formule de la distance de Manhattan entre deux points (x_1, y_1) et (x_2, y_2) dans un plan cartésien est donnée par :

$$d((x_1, y_1), (x_2, y_2)) = |x_2 - x_1| + |y_2 - y_1|$$

2.2 Formule en n dimensions

Dans un espace de dimension n , la distance de Manhattan entre deux points $\mathbf{p} = (p_1, p_2, \dots, p_n)$ et $\mathbf{q} = (q_1, q_2, \dots, q_n)$ est donnée par :

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

Cette formule mesure la somme des différences absolues des coordonnées entre les deux points dans chaque dimension. La distance de Manhattan est souvent utilisée dans les domaines de l'analyse de données, de la robotique et de la modélisation de circuits, entre autres, en raison de sa simplicité et de son interprétation intuitive.

III La Classification Ascendante Hiérarchique (CAH)

1 La méthode

1.1 La méthode CAH

La classification ascendante hiérarchique (CAH) est une méthode de classification non supervisée utilisée en apprentissage automatique et en statistiques pour regrouper des objets similaires en clusters hiérarchiques. Contrairement aux méthodes de clustering comme le k-means qui nécessitent un nombre prédéfini de clusters, la CAH construit une hiérarchie de clusters sans avoir besoin de spécifier le nombre de clusters à l'avance. La CAH commence par considérer chaque observation comme un cluster individuel et fusionne progressivement les clusters les plus similaires jusqu'à ce qu'un seul cluster global soit formé.

1.2 Domaines d'application

La classification ascendante hiérarchique (CAH) peut être appliquée dans divers domaines et contextes. Voici quelques exemples spécifiques :

1. **Finance quantitative** : En finance quantitative, la CAH peut être utilisée pour regrouper des actifs financiers similaires en fonction de leurs rendements, de leur volatilité, de leur corrélation, etc. Cela peut aider les gestionnaires de portefeuille à diversifier leurs investissements et à gérer les risques.
2. **Analyse des marchés financiers** : La CAH peut être utilisée pour regrouper des titres financiers ou des instruments dérivés en fonction de leurs caractéristiques de marché, telles que la capitalisation boursière, le secteur d'activité, la liquidité, etc. Cela peut aider les traders et les analystes à identifier des tendances et des opportunités d'investissement.
3. **Gestion des risques** : En finance, la CAH peut être utilisée pour regrouper des risques similaires ou corrélés dans un portefeuille d'investissement. Cela permet aux gestionnaires de risques de mieux comprendre les expositions au risque et de prendre des décisions informées sur la gestion des risques.
4. **Analyse de crédit** : Dans le domaine de la banque et des services financiers, la CAH peut être utilisée pour regrouper des emprunteurs en fonction de leurs caractéristiques de crédit, telles que le score de crédit, le niveau d'endettement, le

revenu, etc. Cela peut aider les prêteurs à évaluer le risque de crédit et à prendre des décisions de prêt.

5. **Technologies financières (FinTech) :** En FinTech, la CAH peut être utilisée pour segmenter les clients en fonction de leurs comportements financiers, de leurs préférences en matière de services bancaires ou de leurs habitudes d'utilisation des applications mobiles. Cela peut aider les entreprises FinTech à personnaliser leurs offres de produits et services.
6. **Analyse des données de marché :** En utilisant la CAH, les analystes financiers peuvent regrouper les données de marché, telles que les cours des actions, les taux d'intérêt ou les volumes de transactions, afin de mieux comprendre les tendances du marché et les comportements des investisseurs.

La CAH peut être un outil précieux dans le domaine de la finance et de la technologie pour l'analyse des données, la gestion des risques, la prise de décision et la personnalisation des services financiers.

2 Algorithme de CAH

Initialisation :

- Chaque observation est considérée comme un cluster individuel.
- Calcul des distances entre toutes les paires d'observations à l'aide d'une mesure de similarité (euclidienne, corrélation, etc.).

Construction de la matrice de distance :

- Création d'une matrice de distance ou de similarité qui stocke les distances entre chaque paire d'observations.

Fusion de clusters :

- Identifie les deux clusters les plus similaires en fonction de la matrice de distance.
- Fusionne ces clusters pour former un nouveau cluster.
- Met à jour la matrice de distance pour refléter la similarité entre le nouveau cluster et les autres clusters.

Répétition du processus :

- Répéter les étapes précédentes jusqu'à ce qu'il ne reste plus qu'un seul cluster global ou jusqu'à ce qu'un critère d'arrêt prédéfini soit atteint (nombre de clusters souhaité, distance de fusion minimale, etc.).

Construction du dendrogramme :

- Un dendrogramme est construit pour représenter la hiérarchie des clusters. Les axes horizontaux représentent les observations, tandis que les axes verticaux représentent les distances entre les clusters fusionnés.

Choix du nombre de clusters :

- Le nombre de clusters peut être déterminé en coupant le dendrogramme à un niveau approprié. Cela peut être fait en fonction de critères tels que la distance de coupure, le coefficient de corrélation, etc.

Attribution des observations aux clusters :

- Une fois le nombre de clusters choisi, chaque observation est attribuée au cluster auquel elle appartient dans la configuration finale.

L'algorithme de CAH est itératif et agglomératif, ce qui signifie qu'il fusionne progressivement les clusters les plus similaires jusqu'à ce qu'un seul cluster global soit formé.

L'algorithme est assez flexible car il peut utiliser différentes mesures de similarité et peut être adapté à différents types de données et de problèmes. Il est également capable de gérer des ensembles de données de grande taille, bien qu'il puisse devenir computationnellement coûteux pour des ensembles de données très volumineux.

3 Choix de l'indice d'agrégation

Dans le processus de regroupement des éléments lors de la Classification Ascendante Hiérarchique (CAH), il est courant de choisir un indice d'agrégation pour déterminer comment les clusters sont fusionnés à chaque étape de l'algorithme. Plusieurs méthodes sont disponibles pour ce faire.

3.1 Méthode de Ward

La méthode de Ward est l'une des méthodes les plus connues pour choisir l'indice d'agrégation en CAH. Elle vise à minimiser la variance intra-cluster lors de la fusion des clusters, ce qui conduit généralement à des clusters compacts et homogènes.

3.2 Stratégie du saut minimum (single linkage)

Dans la stratégie du saut minimum, également appelée single linkage, les clusters sont fusionnés en regroupant les deux éléments présentant la plus petite distance entre les éléments des deux classes. Cette approche favorise la fusion des clusters dont les éléments sont les plus proches les uns des autres.

3.3 Stratégie du saut maximum (complete linkage)

Dans la stratégie du saut maximum, également connue sous le nom de complete linkage ou du diamètre, les clusters sont fusionnés en regroupant les deux éléments présentant la plus grande distance entre les éléments des deux classes. Cette approche favorise la fusion des clusters dont les éléments sont les plus éloignés les uns des autres.

Dans la suite de notre travail, nous allons nous baser sur la Stratégie du saut minimum pour notre algorithme de la CAH

4 Implémentation avec python

4.1 Le choix du langage

Nous avons choisi Python pour coder l'algorithme de CAH et K-means en raison de sa popularité et de sa flexibilité en matière de programmation. Voici quelques raisons qui justifient ce choix :

1. **Facilité d'apprentissage** : Python est reconnu pour sa syntaxe simple et intuitive, ce qui en fait un langage de programmation idéal pour les débutants. Cela nous a permis de nous familiariser rapidement pour l'implémentation de nos algorithmes.
2. **Grande communauté de développeurs** : Python bénéficie d'une vaste communauté de développeurs et d'utilisateurs, il nous a été facile de trouver des ressources, des tutoriels et des exemples de code.

3. **Extensibilité et modularité** : Python permet de créer des codes modulaires et réutilisables, ce qui est particulièrement avantageux car cela permet de diviser le code en fonctions spécifiques et de le réutiliser dans d'autres projets.

4.2 Le code sur VsCode

```
class Cluster:
    nbrCluster = 0

    def __init__(self, data):
        Cluster.nbrCluster += 1
        self.numeroCluster = Cluster.nbrCluster
        self.indicesIndividus = [Cluster.nbrCluster] # Initialise avec un tableau contenant seulement le numéro de cluster
        self.data = data
```

FIGURE III.1 – Ma classe Cluster

```
class Fonction:
    @staticmethod
    def distanceEuclidienne(arg1, arg2): ...

    @staticmethod
    def extract_data_from_excel(file_path): ...

    @staticmethod
    def distanceManatthan(arg1, arg2): ...

    @staticmethod
    def aleatoire(k, tabIndividus): ...

    @staticmethod
    def mean2DTab(tab): ...

    @staticmethod
    def kmeans(k, tabIndividus, distanceChoisie): ...

    @staticmethod
    def hierarchy(clusters, distanceChoisie): ...
```

FIGURE III.2 – Ma classe des Fonctions

Voici le code Python pour la fonction hierarchy :

```

1 @staticmethod
2     def hierarchy(clusters,distanceChoisie):
3         doc = Document(os.path.join(save_path, 'rapport.docx'))
4         doc.add_page_break()
5         doc.add_paragraph("\n\n\n")
6         hierarchy = []
7         indices = []
8         iteration = 0
9         while len(clusters)-2*iteration > 1:
10             distanceMin = float('inf')
11             indice1, indice2 = None, None
12             for i in range(len(clusters)):
13                 if i not in indices:
14                     for j in range(i + 1, len(clusters)):
15                         if j not in indices:
16                             if distanceChoisie == 'distanceEuclidienne':
17                                 distance = Fonction.distanceEuclidienne(
clusters[i], clusters[j])
18                             elif distanceChoisie == 'distanceManatthan'
:
19                                 distance = Fonction.distanceManatthan(
clusters[i], clusters[j])
20                             if distance < distanceMin:
21                                 distanceMin = distance
22                                 indice1, indice2 = i, j
23                             indices.append(indice1)
24                             indices.append(indice2)
25                             iteration +=1
26                             new_data = [(clusters[indice1].data[i] + clusters[indice2].
data[i]) / 2 for i in range(len(clusters[indice1].data))]
27                             newCluster = Cluster(new_data) # Cr e un nouveau cluster
avec les donn es du barycentre des deux clusters
28                             newCluster.indicesIndividus = clusters[indice1].
indicesIndividus + clusters[indice2].indicesIndividus
29                             clusters.append(newCluster)
30                             doc.add_paragraph(f"{iteration}-- Liaison du Cluster {
clusters[indice1].numeroCluster-1} et du Cluster {clusters[indice2].
numeroCluster-1} pour former le cluster {newCluster.numeroCluster-1}
regroupant ainsi {len(newCluster.indicesIndividus)} elements.\n")
31                             # Enregistrer le document Word dans le dossier static
32                             doc.save(os.path.join(save_path, 'telechargeable.docx'))
33                             hierarchy.append([indice1, indice2, distanceMin, len(
newCluster.indicesIndividus)])
34                             return hierarchy

```

IV Kmeans ou Kmoyennes

1 La méthode et ses domaines d'application

1.1 La méthode Kmeans

La méthode K-means est un algorithme de partitionnement de données largement utilisé en apprentissage automatique non supervisé. Son objectif principal est de diviser un ensemble de données en k clusters distincts, où chaque point de données appartient au cluster dont le centroïde est le plus proche.

1.2 Domaines d'application

- **Segmentation de marché** : K-means est utilisé pour segmenter les clients en groupes homogènes en fonction de leurs caractéristiques et comportements d'achat.
- **Analyse de l'image** : En traitement d'images, K-means peut être utilisé pour la segmentation d'images, la compression d'images et la détection de motifs.
- **Recherche de documents** : Dans le domaine de la recherche d'informations, K-means peut être utilisé pour regrouper des documents similaires ensemble.
- **Classification de texte** : K-means peut être utilisé pour classer des documents texte en fonction de leur similitude.
- **Biologie** : En biologie, K-means est utilisé pour classer des données génétiques et pour l'analyse de séquences d'ADN.
- **Marketing** : Dans le domaine du marketing, K-means est utilisé pour analyser les comportements des consommateurs et pour cibler des segments de marché spécifiques.
- **Détection d'anomalies** : K-means peut être utilisé pour détecter des anomalies ou des valeurs aberrantes dans un ensemble de données.
- **Prévision de la demande** : En logistique et gestion de la chaîne d'approvisionnement, K-means peut être utilisé pour prévoir la demande de produits.

2 Algorithme de Kmeans

L'algorithme K-means est un algorithme de partitionnement de données largement utilisé en apprentissage automatique non supervisé. Il cherche à diviser un ensemble de données en un nombre prédéfini de clusters (groupes) distincts. Voici une description détaillée de l'algorithme :

1. Initialisation :

- Choisir le nombre de clusters k à former.
- Sélectionner aléatoirement k points parmi les données comme centres de clusters initiaux, appelés centroïdes.

2. Affectation des points aux clusters :

- Pour chaque point de données, calculer la distance entre ce point et chaque centroïde.
- Affecter le point au cluster correspondant au centroïde le plus proche, en minimisant la distance. Cela crée k groupes initiaux.

3. Mise à jour des centroïdes :

- Recalculer les nouveaux centroïdes de chaque cluster en prenant la moyenne des points qui lui sont attribués. Les nouveaux centroïdes représentent le centre de gravité de chaque cluster.

4. Répéter :

- Répéter les étapes 2 et 3 jusqu'à ce qu'une condition d'arrêt soit atteinte. Les conditions d'arrêt courantes sont :
 - Convergence : Aucun point ne change de cluster entre deux itérations successives.
 - Nombre d'itérations maximum : Fixer un nombre maximum d'itérations à effectuer.

5. Fin de l'algorithme :

- Une fois la condition d'arrêt atteinte, les clusters sont considérés comme stabilisés et l'algorithme est terminé.
- Les points de données sont désormais regroupés en k clusters, avec chaque cluster représenté par son centroïde.

L'algorithme K-means est sensible au choix initial des centroïdes, ce qui peut influencer les clusters obtenus. Il est souvent exécuté plusieurs fois avec différents points de départ pour améliorer la robustesse des résultats. De plus, la qualité de la partition dépend souvent du nombre de clusters choisi, qui doit être déterminé a priori ou à l'aide de critères de validation.

3 Convergence de l'Algorithme

Une question se pose quant à la convergence de l'algorithme K-means, surtout lorsque les centres initiaux sont choisis de manière aléatoire. Pour répondre à cette question, nous proposons de représenter l'algorithme de la manière suivante :

Soit un ensemble de points \mathbf{X}_i , $1 \leq i \leq P$, dans R^N , où P est le nombre de points. À chaque point est associée une classe c_i , $1 \leq i \leq P$, dans $\{1, \dots, C\}$, où C est le nombre de classes.

Nous définissons les barycentres des classes \mathbf{G}_i , $1 \leq i \leq C$, dans R^N .

Initialisation L'initialisation consiste à choisir aléatoirement une classe pour chaque point parmi $\{1, \dots, C\}$. On pose $t = 0$.

Calcul des barycentres Pour $k \in \{1, \dots, C\}$, \mathbf{G}_k^t est calculé comme suit :

$$\mathbf{G}_k^t = \frac{\sum_{i=1}^P \mathbf{X}_i \cdot 1\{c_i^t = k\}}{\sum_{i=1}^P 1\{c_i^t = k\}}$$

Calcul de l'inertie L'inertie I^t est calculée comme suit :

$$I^t = \sum_{i=1}^P d^2(\mathbf{X}_i, \mathbf{G}_{c_i^t}^t)$$

où $d(\mathbf{X}_i, \mathbf{G}_{c_i^t}^t)$ est la distance entre \mathbf{X}_i et $\mathbf{G}_{c_i^t}^t$.

Arrêt de l'algorithme Si $s_i^t > 0$ et $I^t \sim I^{t-1}$, on arrête l'algorithme.

Attribution des classes Pour chaque point $i \in \{1, \dots, P\}$, c_i^{t+1} est attribué selon :

$$c_i^{t+1} = \arg \min d(\mathbf{X}_i, \mathbf{G}_k^t)$$

On retourne ensuite à l'étape de calcul des barycentres jusqu'à la convergence de l'inertie I^t .

3.1 Théorème de convergence des K-means

Quelle que soit l'initialisation choisie, la suite $(I^t)_{t \geq 0}$ construite par l'algorithme des K-means converge.

La démonstration du théorème nécessite le lemme suivant.

Lemme de l'inertie minimum Soit $(\mathbf{X}_1, \dots, \mathbf{X}_P)$ dans R^N , P points, le minimum de la quantité $Q(\mathbf{Y})$, $\mathbf{Y} \in R^N$, est atteint pour $\mathbf{Y} = \mathbf{G} = \frac{1}{P} \sum_{i=1}^P \mathbf{X}_i$, le barycentre des points $(\mathbf{X}_1, \dots, \mathbf{X}_P)$.

Le lemme précédent permet d'affirmer que la suite $(I^t)_{t \geq 0}$ est décroissante et, puisqu'elle est minorée par 0, elle est donc convergente.

L'algorithme cherche à attribuer à chaque point de l'ensemble une classe parmi celles disponibles. La solution trouvée dépend de l'initialisation et n'est pas forcément celle qui minimise l'inertie intra-classe : l'inertie finale est un minimum local. Néanmoins, elle assure que la partition est formée de classes convexes.

4 Quelques versions améliorée de Kmeans

4.1 Global Kmeans

Le "Global K-means" est une méthode d'initialisation pour l'algorithme K-means qui vise à trouver une solution globalement optimale en ajoutant dynamiquement de nouveaux centres de cluster tout en appliquant itérativement l'algorithme K-means jusqu'à la convergence. Voici un résumé de son fonctionnement selon l'algorithme fourni :

Algorithm 1 Global K-means

Require: Ensemble de N données, notées x ; Nombre de groupes souhaités, noté k ;

Ensure: Une partition de k groupes $\{C_1, C_2, \dots, C_k\}$

- 1: $C \leftarrow$ Centre de gravité de l'ensemble des données ;
 - 2: **repeat**
 - 3: Initialiser les centres $i - 1$ par le résultat de l'étape précédente ;
 - 4: **for all** données x **do**
 - 5: Considérer x comme étant le i ème centre ;
 - 6: Affecter les données au plus proche centre ;
 - 7: Calculer l'erreur quadratique pour $C_i = x$;
 - 8: Garder le centre $C_i = x$ qui minimise l'erreur quadratique ;
 - 9: **end for**
 - 10: Appliquer l'algorithme K-means jusqu'à la convergence ;
 - 11: **until** obtenir une partition en k groupes
-

Dans cet algorithme, chaque donnée peut être candidate pour devenir un centre, et le meilleur candidat est celui qui minimise l'erreur quadratique. Cela permet d'obtenir une solution globalement optimale en ajoutant de nouveaux centres de cluster de manière itérative.

Cependant, les auteurs ont constaté que cette approche peut être lourde en raison de la stratégie de choix du nouveau centre. Pour accélérer le processus, ils ont proposé le "Fast Global K-means" avec une nouvelle stratégie. Cette stratégie conserve la même

philosophie que la précédente, mais évite d'affecter les données aux centres les plus proches (centres déjà existants en plus du centre candidat) et de calculer l'erreur quadratique. Au lieu de cela, le nouveau centre est choisi en maximisant un taux basé sur la diminution de l'erreur quadratique en fonction du nombre de centres. Selon les expérimentations des auteurs, cette technique améliore le temps d'exécution tout en garantissant de bons résultats, presque aussi bons que ceux fournis par la stratégie précédente.

4.2 Principe de l'initialisation par le mal classé

L'absence d'un signe indiquant si l'optimum global est atteint ou pas fait penser à la possibilité d'améliorer les résultats. Observant l'équation :

$$x_i \in R^n, \quad s_i = \arg \min_j ||x_i - \mu_j||^2$$

un objet est affecté à un groupe s'il lui est le plus proche, plus la distance diminue plus la probabilité d'appartenance à ce groupe augmente. Dans le cas contraire, l'objet le plus loin de son groupe d'appartenance est considéré comme étant mal classé, il fera certainement un bon candidat afin de former le nouveau centre.

Le global K-means est amorcé par un seul groupe ayant pour représentant le centre de gravité de l'ensemble des données. Dans certains cas, cette partie de l'espace est vide (voir Figure 1), ce qui peut dégrader la classification. Nous proposons d'amorcer l'initialisation du K-means avec deux groupes, les centres de ces groupes doivent assurer la séparabilité des données au cours de la classification. Il est évident de choisir les deux données les plus éloignées.

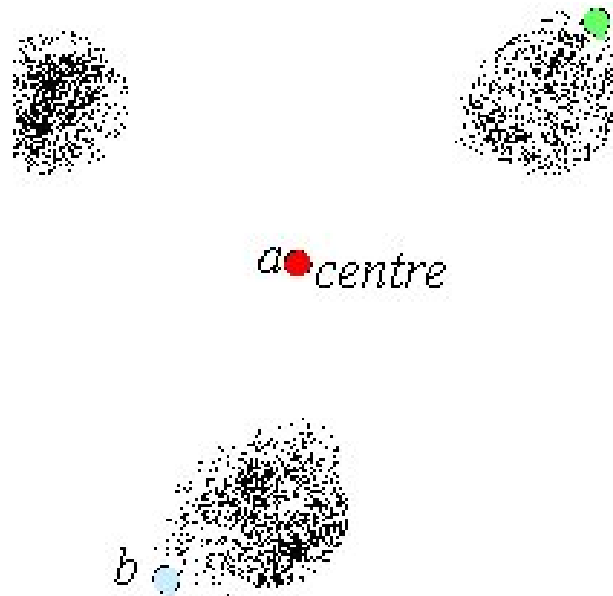


FIGURE IV.1 – Exemple

Algorithm 2 Initialisation par le mal classé

- 1: Création d'une matrice de distance
 - 2: Choisir les deux éléments les plus éloignés (ils représentent les deux premiers centres)
 - 3: **while** le nombre de classes souhaité n'est pas atteint **do**
 - 4: Affecter les individus aux noyaux disponibles
 - 5: Sélectionner un élément mal classé (celui qui possède la plus grande distance de son centre le plus proche)
 - 6: Ajouter cet individu à l'ensemble des noyaux
 - 7: Augmenter le nombre des noyaux
 - 8: **end while**
-

```
class Fonction:
    @staticmethod
    def distanceEuclidienne(arg1, arg2): ...

    @staticmethod
    def extract_data_from_excel(file_path): ...

    @staticmethod
    def distanceManatthan(arg1, arg2): ...

    @staticmethod
    def aleatoire(k, tabIndividus): ...

    @staticmethod
    def mean2DTab(tab): ...

    @staticmethod
    def kmeans(k, tabIndividus, distanceChoisie): ...

    @staticmethod
    def hierarchy(clusters, distanceChoisie):|...
```

FIGURE IV.2 – Ma classe des Fonctions

5 Implémentation avec python

La fonction aléatoire est mis en dehors de Kmeans car nous allons proposer plus tard une façon plus améliorée d'initialiser les centres conformément à la version améliorée énoncée précédemment.

5.1 Initialisation aléatoire

```

1  def aleatoire(k, tabIndividus):
2      centroidTab = []
3      indiceTab = []
4      for i in range(0,k):
5          indice = random.randint(0,len(tabIndividus)-1)
6          while indice in indiceTab:
7              indice = random.randint(0,len(tabIndividus)-1)
8          indiceTab.append(indice)
9          centroidTab.append(Centroid(tabIndividus[indice]))
10     return centroidTab

1  @staticmethod
2     def kmeans(k,tabIndividus,distanceChoisie):
3         doc = Document(os.path.join(save_path, 'rapport.docx'))
4         doc.add_page_break()
5         doc.add_paragraph("\n\n\n")
6         centroidTab,indiceTab = Fonction.aleatoire(k, tabIndividus)
7         print(indiceTab)
8         #choix aléatoire des centroides initiaux
9         nbreModification = 10
10        while nbreModification != 0 :
11            nbreModification = 0
12            #Affectation des individus aux groupes
13            for i in range(0,len(tabIndividus)):
14                if distanceChoisie == 'distanceEuclidienne':
15                    distMin = Fonction.distanceEuclidienne(tabIndividus[
16                    i],centroidTab[tabIndividus[i].group]) #distance entre l'individu et
17                    son ancien centre
18                    for j in range(0,len(centroidTab)):
19                        if distMin > Fonction.distanceEuclidienne(
20                        tabIndividus[i],centroidTab[j]):
21                            distMin = Fonction.distanceEuclidienne(
22                            tabIndividus[i],centroidTab[j])
23                            tabIndividus[i].group = j
24                            nbreModification += 1
25                elif distanceChoisie == 'distanceManatthan':
26                    distMin = Fonction.distanceManatthan(tabIndividus[i
27                    ],centroidTab[tabIndividus[i].group]) #distance entre l'individu et
28                    son ancien centre
29                    for j in range(0,len(centroidTab)):
30                        if distMin > Fonction.distanceManatthan(
31                        tabIndividus[i],centroidTab[j]):
32                            distMin = Fonction.distanceManatthan(
33                            tabIndividus[i],centroidTab[j])
34                            tabIndividus[i].group = j
35                            nbreModification += 1
36            if nbreModification != 0 : # cas ou les groupes ont chang
37                #Calcul des nouveaux centres
38                for i in range(k): # k est le nombre de centres
39                    centroid_data_sum = np.zeros_like(centroidTab[i].
40                    data) # Initialiser la somme des données des individus pour chaque
41                    centre
42                    ctr = np.zeros(len(centroidTab[i].data)) #
43                    Initialiser le compteur pour chaque dimension

```

```

34         # Calculer la somme des données des individus pour
chaque centre
35         for l in range(len(tabIndividus)):
36             if tabIndividus[l].group == i:
37                 centroid_data_sum += tabIndividus[l].data
38                 ctr += 1
39
40         # Mettre à jour les centres en calculant la moyenne
des données des individus pour chaque centre
41         centroidTab[i].data = np.where(ctr != 0,
centroid_data_sum / ctr, centroidTab[i].data)
42
43     TabCenters = [centroidTab[i].data for i in range(len(centroidTab
))]
44     resultat = [] #resultat sous forme de liste indiquant le
groupe auquel chaque individu appartient
45     for i in range(0, len(tabIndividus)):
46         resultat.append(tabIndividus[i].group)
47         doc.add_paragraph(f"{i+1}-- Individu {i+1} appartient au
groupe {tabIndividus[i].group}.\\n")
48         doc.save(os.path.join(save_path, 'telechargeable.docx'))
49     return resultat, TabCenters

```

5.2 Initialisation par le mal classé

```

1 @staticmethod
2     def initialisation_par_mal_classe(k, tab_individus):
3         # Création d'une matrice de distance
4         distance_matrix = np.zeros((len(tab_individus), len(
tab_individus)))
5         for i in range(len(tab_individus)):
6             for j in range(i+1, len(tab_individus)):
7                 distance_matrix[i][j] = distance_matrix[j][i] = np.
linalg.norm(tab_individus[i].data - tab_individus[j].data)
8
9         # Choisir les deux éléments les plus loigns
10        max_distance = np.max(distance_matrix)
11        indices_max_distance = np.unravel_index(np.argmax(
distance_matrix), distance_matrix.shape)
12        centroids = [tab_individus[indices_max_distance[0]],
tab_individus[indices_max_distance[1]]]
13
14        # Initialisation du nombre de noyaux
15        nbre_noyaux = 2
16
17        # Boucle jusqu' ce que le nombre de classes souhait soit
atteint
18        while nbre_noyaux < k:
19            # Affecter les individus aux noyaux disponibles
20            for individu in tab_individus:
21                individu.group = np.argmin([np.linalg.norm(individu.data
- centroid.data) for centroid in centroids])
22
23            # Sélectionner un élément mal classé
24            mal_classe = max(tab_individus, key=lambda x: np.linalg.norm
(x.data - centroids[x.group].data))

```

```

25
26         # Ajouter cet individu à l'ensemble des noyaux
27         centroids.append(mal_classe)
28
29         # Augmenter le nombre des noyaux
30         nbre_noyaux += 1
31
32     return centroids

```

5.3 Global Kmeans

```

1 @staticmethod
2     def global_kmeans(data, k):
3         # Initialisation des centres avec le centre de gravité de l'
4         ensemble des données
5         centers = [np.mean(data, axis=0)]
6
7         # Répéter jusqu'à obtenir une partition en k groupes
8         while len(centers) < k:
9             # Initialisation des centres pour cette itération
10            new_centers = centers.copy()
11
12            # Itération sur toutes les données
13            for x in data:
14                # Considérer x comme tant le ième centre
15                new_center = x
16
17                # Affecter les données au plus proche centre
18                min_distance = float('inf')
19                for center in new_centers:
20                    distance = np.linalg.norm(x - center)
21                    if distance < min_distance:
22                        min_distance = distance
23                        new_center = center
24
25                # Calculer l'erreur quadratique pour le nouveau centre
26                error = sum(np.linalg.norm(x - center) ** 2 for center
27                in new_centers)
28
29                # Garder le centre qui minimise l'erreur quadratique
30                if error < sum(np.linalg.norm(x - center) ** 2 for
31                center in new_centers[:-1]):
32                    new_centers[-1] = new_center
33
34            # Appliquer l'algorithme K-means jusqu'à la convergence
35            # (Non implémenté ici car dépendant de l'implémentation
36            spécifique de K-means)
37
38            # Ajouter le nouveau centre à la liste des centres
39            centers.append(new_centers[-1])
40
41    return centers

```

Pour notre application web, nous allons juste inclure le code du kmeans standard...

V Réalisation d'un test avec notre Application Python Flask (et Exemple de données du cours)

Dans cette partie nous allons présenter les divers résultats de nos algorithmes de CAH et de Kmeans en utilisant un jeu de données exploitées lors des travaux pratiques.

1 Présentation de l'interface

Nous avons prévu un interface qui propose à l'utilisateur de :

- Choisir la méthode qu'il souhaite utiliser pour sa classification (soit la CAH ou Kmeans) ;
- Choisir une mesure de dissimilarité, nous avons proposé la distance euclidienne et la distance de Manatthan ;
- Charger un jeu de données en forme de Table Excel organisée en tableau 2 dimensions standard de représentation des réalisations des variables pour chaque individu ;
- Enfin, choisir le nombre de groupes souhaité dans le cas d'un clustering avec Kmeans

Voici un aperçu de l'interface :

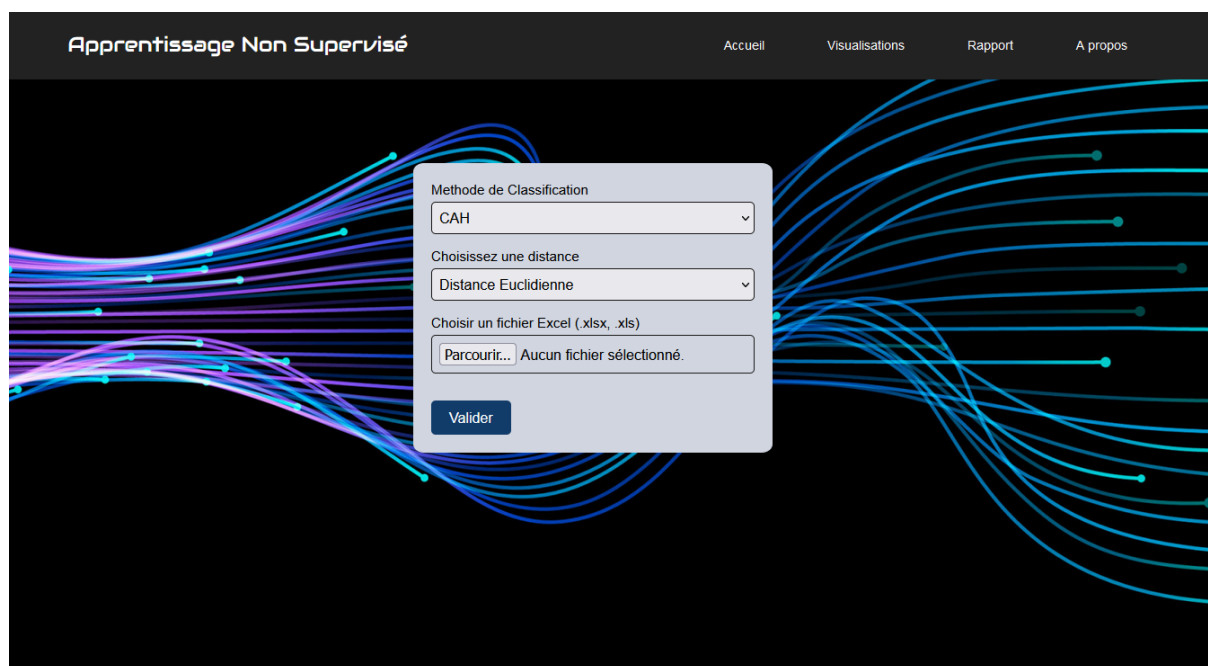


FIGURE V.1 – Interface utilisateur

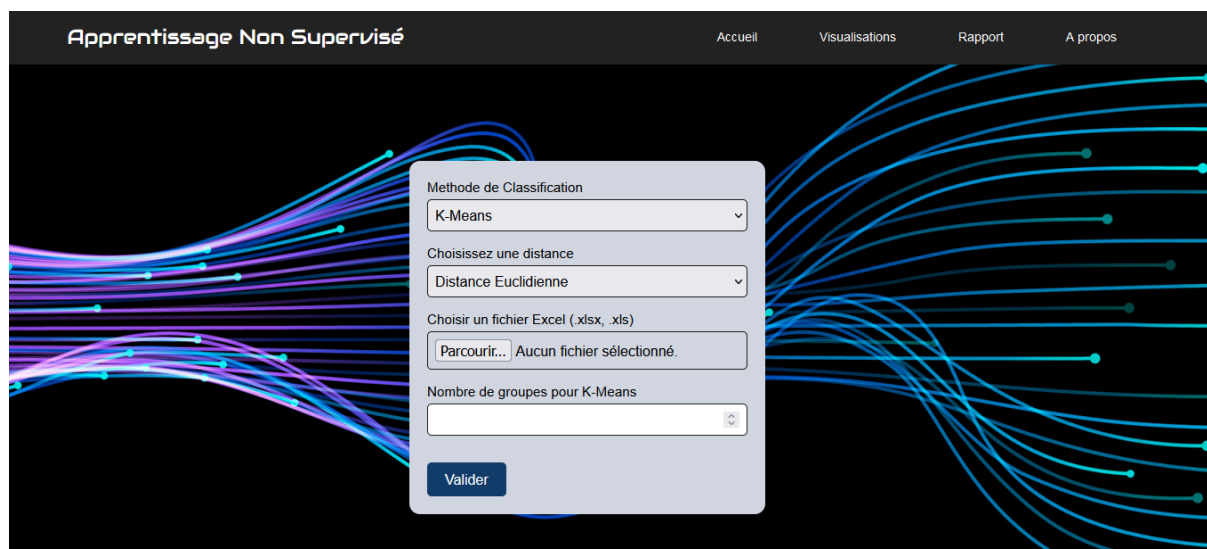


FIGURE V.2 – Interface utilisateur

2 Présentation du jeu de données et Chargement depuis ordinateur

Voici la table de données que nous allons exploiter :

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
2															
3															
4															
5			N	Arabe	Français	Anglais	Philosophie	Math	PC	SVT	ES				
6			etu1	16	14.5	14.75	13.5	10	11.75	11	16.5				
7			etu2	11.5	14	13	10	16	17.5	15.75	15.5				
8			etu3	12	11.75	14	11.25	14.75	16	16.5	16				
9			etu4	13.5	12	15	12.5	15	14.5	14.75	15				
10			etu5	15.75	16.5	15	14.25	9.75	11	11.25	14.75				
11			etu6	11.75	10.75	13	12.75	15	14.5	16	15.5				
12			etu7	14.5	13.75	16	15.5	15	14	13.75	13				
13			etu8	12.5	10	11	13	13.5	12.75	15	14				
14			etu9	13.5	15	12.75	10	12.5	11.75	10	15				
15			etu10	17	14.25	16	15.75	11.75	13	12.5	16.75				
16			etu11	15.5	16	14.75	13.25	12	12.5	12.75	13.75				
17			etu12	13.75	16	17.5	13.5	16.75	17	16.5	15				
18			etu13	14	11.75	14.5	12.5	13	11.75	14	16.25				
19			etu14	10.5	9.5	11.75	13	11	12.5	10	17				
20			etu15	15.5	13.25	14	14	13.75	15	14.5	14				
21			etu16	10	12	10	8	14	13	13	12				
22															

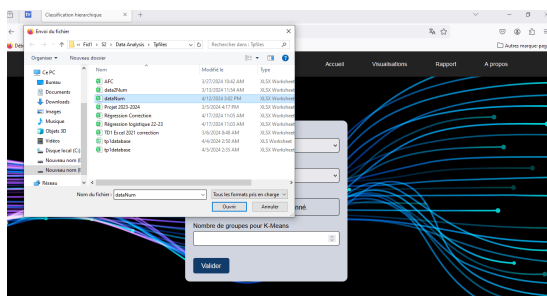
FIGURE V.3 – Jeu de données

Une remarque à noter est que les données peuvent ne pas se situer à partir de la première cellule du fichier Excel, alors nous avons réalisé un code intelligent qui permet de rechercher la cellule initiale de la table....

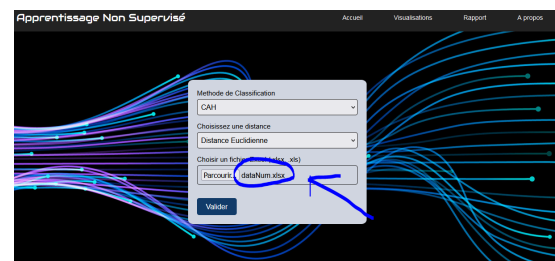

```
class Fonction:
    @staticmethod
    def extract_data_from_excel(file_path):
        # Charger le fichier Excel
        dataframe = pd.read_excel(file_path, engine='openpyxl')
        # Identifier la position du tableau de données en recherchant les index des premières cellules non vides
        start_row = None
        start_col = None
        for i in range(dataframe.shape[0]):
            for j in range(dataframe.shape[1]):
                if pd.notna(dataframe.iloc[i, j]):
                    start_row = i
                    start_col = j
                    break
            if start_row is not None:
                break
        # Collecter les noms des individus (ligne d'index 0)
        individus = dataframe.iloc[start_row+1:, start_col].tolist()
        # Collecter les noms des variables (première ligne après les noms des individus)
        variables = dataframe.iloc[start_row, start_col+1:].tolist()
        # Extraire les valeurs dans un tableau à deux dimensions
        values = dataframe.iloc[start_row+1:, start_col+1:]
        return individus, variables, values
```

FIGURE V.4 – Recherche de la Table et Récupération de valeurs

Maintenant nous pouvons charger ce fichier depuis notre Ordinateur (ou en ligne)...



(a) Appuyer sur Parcourir et choisir un fichier



(b) Fichier chargé avec succès

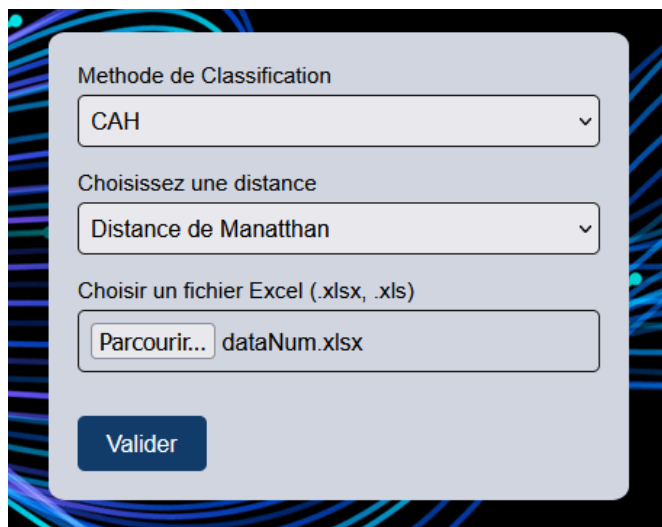
FIGURE V.5 – Chargement fichier

Alors que notre fichier est chargé nous pouvons passer à la visualisation de notre résultat.

3 Visualisation des résultats

Une fois les données chargées et tous les choix (distance, méthode, nombre de classe) faits, nous pouvons obtenir le résultat de la classification en fonction des données d'entrée :

- **Cas de la CAH**
Présentation des entrées pour le test :
- **Cas de Kmeans**
Présentation des entrées pour le test :



Methode de Classification

CAH

Choisissez une distance

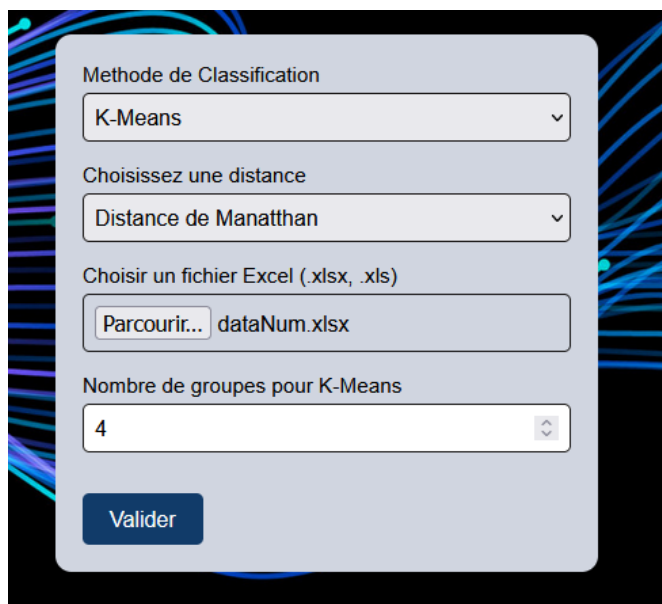
Distance de Manatthan

Choisir un fichier Excel (.xlsx, .xls)

Parcourir... dataNum.xlsx

Valider

FIGURE V.6 – Les entrées 1



Methode de Classification

K-Means

Choisissez une distance

Distance de Manatthan

Choisir un fichier Excel (.xlsx, .xls)

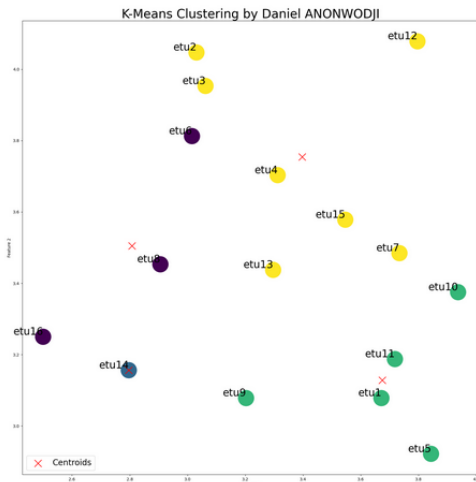
Parcourir... dataNum.xlsx

Nombre de groupes pour K-Means

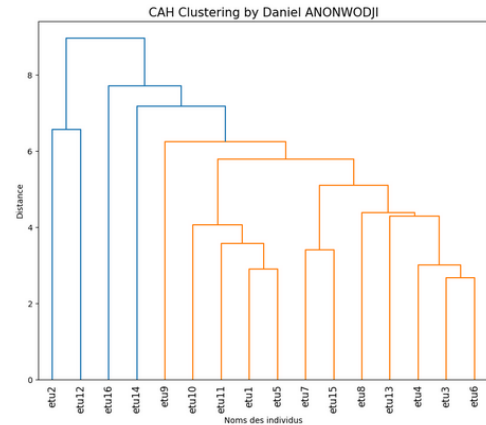
4

Valider

FIGURE V.7 – Les entrées 2

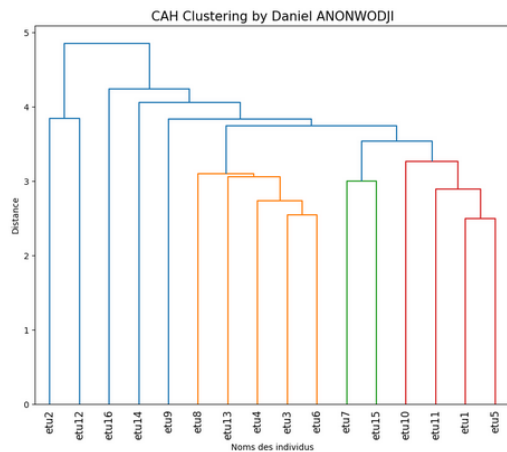


(a) Kmeans à 4 groupes

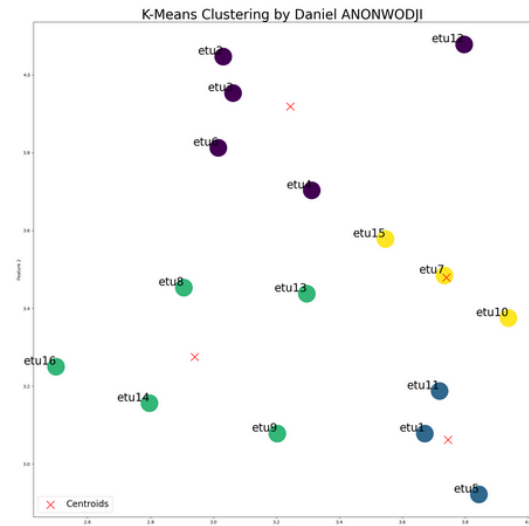


(b) CAH

FIGURE V.8 – Résultats avec Distance Euclidienne



(a) CAH



(b) Kmeans à 4 groupes

FIGURE V.9 – Résultats avec Distance de Manatthan

Conclusion

Somme toute, ce projet d'implémentation des techniques d'analyse de données, notamment la Classification Ascendante Hiérarchique (CAH) et l'algorithme K-means, a été une expérience enrichissante et instructive. À travers la combinaison d'une exploration théorique approfondie et d'une mise en œuvre pratique en utilisant le langage de programmation Python, nous avons acquis une compréhension plus profonde des concepts fondamentaux du clustering et de leur application dans le domaine de l'analyse de données.

La méthodologie rigoureuse adoptée tout au long du projet, comprenant une explication détaillée des algorithmes, leur mise en œuvre en Python et le développement d'une interface utilisateur conviviale, a permis une approche holistique de l'analyse de données. De plus, l'utilisation de différentes mesures de dissimilarité et de stratégies d'agrégation des clusters nous a permis d'explorer divers aspects du clustering et d'apprécier leur impact sur les résultats finaux.

L'interface utilisateur développée en utilisant Flask a ajouté une dimension interactive à notre projet, offrant aux utilisateurs la possibilité d'explorer et d'analyser leurs propres ensembles de données en utilisant les méthodes de clustering mises en œuvre. Les exemples de résultats de classification présentés ont illustré l'efficacité et la pertinence des techniques de clustering pour découvrir des modèles cachés dans les données et prendre des décisions stratégiques.

Enfin, ce projet a ouvert la voie à de nombreuses opportunités futures, notamment l'exploration de techniques de clustering avancées, l'intégration de fonctionnalités supplémentaires dans l'interface utilisateur et l'application de ces méthodes à des ensembles de données du monde réel dans divers domaines d'application. Nous espérons que ce projet servira de base solide pour de futures recherches et développements dans le domaine de l'analyse de données et du clustering.

En résumé, ce projet a été une expérience passionnante qui a permis d'approfondir nos connaissances en analyse de données et de mettre en lumière l'importance des techniques de clustering pour explorer, interpréter et tirer des conclusions significatives à partir de vastes ensembles de données.

[1] [2] [3] [4]

Références

- [1] Fabien Chevalier-Jérôme Le Bellac. *Classification*. 2012.
- [2] Angelina Roche. *Classification Ascendante Hiérarchique*.
- [3] B. El Asri. *Analyse de données*. 1ère année Finance et ingénierie décisionnelle, 2023.
- [4] Faïcel Chamroukhi. Algorithme des centres mobiles (k-means).