

المدرسة الوطنية للعلوم التطبيقية - أكادير
ⵜⴰⵎⴰⵔⵜ ⵜⴰⵎⴰⵏⴰⵢⵜ ⵜⴰⵖⴰⵏⴰⵏⵜ ⵜⴰⵙⴰⵎⴰⵏⴰⵢⵜ - ⵎⴰⵖⴰⵔⵉⵏ
ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES - AGADIR



PROJET ANALYSE DES DONNÉES

ÉCOLE NATIONAL DES SCIENCES APPLIQUÉES

AGADIR

Réalisé Par :

Ziad Ouahbi

Encadré Par :

Professeur Mr Brahim Al ASRI

Resumé

Le projet envisagé a pour but la création d'un site web interactif destiné à l'analyse de données, employant pour cela les méthodes de classification hiérarchique et K-means afin de fournir aux utilisateurs une plateforme accessible pour le traitement et l'analyse d'ensembles de données complexes. L'accent est mis sur la facilité avec laquelle les utilisateurs peuvent visualiser les résultats des groupements créés par ces techniques de clustering. Le développement du site sera réalisé en Python, un langage de programmation particulièrement bien adapté au traitement des données et à l'analyse statistique, ce qui garantit une manipulation efficace des opérations complexes de données. Pour l'interface utilisateur, le choix s'est porté sur Streamlit, une solution idéale pour mettre en place rapidement des applications web orientées data science. Cette technologie est reconnue pour sa capacité à simplifier la visualisation de données complexes et à rendre les interactions avec les analyses de données plus intuitives, offrant ainsi aux utilisateurs une expérience enrichissante et directement applicable à divers contextes professionnels ou académiques.

Contents

1	Introduction	4
2	Classification Hiérarchique	5
2.1	Définition	5
2.2	Partie Théorique	5
2.2.1	Fondements théoriques	5
2.2.2	Métrique de distance	6
2.2.3	Méthodes de liaison	6
2.2.4	Algorithme	7
2.3	Partie Codage	7
2.3.1	Packages utilisés	7
2.3.2	Fonction Read CSV	7
2.3.3	Calcul Distance Euclidean	8
2.3.4	Fonction de Linkage	8
2.3.5	Fonction De Classification Hiérarchique	8
2.4	Conclusion	9
3	K-means	10
3.1	Définition	10
3.2	Partie Théorique	10
3.2.1	Initialisation	10
3.2.2	Affectation des clusters	10
3.2.3	Convergence	11
3.3	Partie Codage	11
3.3.1	Packages utilisés	11
3.3.2	Fonction Read CSV	11
3.3.3	Fonction Mean	12

3.3.4	Fonction Transpose	12
3.3.5	Fonction Distance	12
3.3.6	Fonction Assign Data	13
3.3.7	Fonction Compute Centroids	13
3.3.8	Fonction Principale	13
3.3.9	Fonction Quality	14
3.4	Conclusion	14
4	Interface	15
4.1	Outils Utilisées	15
4.1.1	Python	15
4.1.2	Streamlit	15
4.1.3	HTML (HyperText Markup Language)	15
4.1.4	CSS	16
4.1.5	JavaScript	16
4.1.6	Bootstrap	16
4.2	Pages	17
4.2.1	HomePage	17
4.2.2	Classification Hiérarchique Page	17
4.2.3	K-means Page	19
4.3	Lien De L'interface	20
5	Conclusion	21

Chapter 1

Introduction

En constante évolution de la science des données, la capacité à analyser et interpréter de vastes ensembles de données devient un élément crucial pour la prise de décisions éclairées dans divers secteurs industriels et académiques. Face à des volumes de données toujours plus importants, les entreprises et les chercheurs sont souvent confrontés à des défis complexes lorsqu'il s'agit de grouper efficacement les informations pour en extraire des insights pertinents. Par exemple, une entreprise de commerce électronique cherchant à améliorer l'expérience client pourrait vouloir segmenter ses utilisateurs en fonction de leurs comportements d'achat pour cibler plus précisément ses campagnes marketing. Cependant, sans les outils appropriés pour analyser ces données, il peut être difficile d'identifier les patterns qui permettent de former des groupes homogènes et significatifs.

C'est dans ce contexte que notre projet de développement d'un site web interactif pour l'analyse de données prend tout son sens. Utilisant des méthodes éprouvées de classification telles que la classification hiérarchique et K-means, notre plateforme vise à rendre l'analyse de données à la fois accessible et intuitive, cette plateforme permettra aux utilisateurs de visualiser les résultats de leurs analyses de données de manière claire et interactive. Ainsi, des utilisateurs de divers horizons, qu'ils soient spécialistes du marketing, économistes ou chercheurs, pourront exploiter pleinement le potentiel de leurs données pour prendre des décisions mieux informées, tout en surmontant les défis posés par la complexité des ensembles de données modernes.

Chapter 2

Classification Hiérarchique

2.1 Définition

La classification hiérarchique est une méthode d'analyse des données qui vise à construire une hiérarchie ou une structure en arbre de groupes appelés clusters. Imaginez un arbre généalogique, mais au lieu de personnes, chaque branche représente un groupe de données similaires. La beauté de cette méthode réside dans sa simplicité conceptuelle et sa capacité à visualiser les relations entre les données..

2.2 Partie Théorique

2.2.1 Fondements théoriques

La classification hiérarchique peut être effectuée selon deux approches principales : agglomérative (de bas en haut) ou divisive (de haut en bas).

Approche agglomérative : Elle commence par traiter chaque observation comme un groupe isolé et, à chaque étape, fusionne les deux groupes les plus proches jusqu'à ce que tous les groupes soient fusionnés en un seul.

Approche divisive : Elle commence avec toutes les observations dans un seul groupe, puis divise progressivement le groupe en clusters plus petits jusqu'à ce que chaque observation forme un groupe isolé.

2.2.2 Métrique de distance

La clé de la classification hiérarchique est la définition d'une mesure de similarité ou de distance entre les observations. Les métriques utilisées comprennent :

Distance euclidienne : Convient pour les mesures continues.

$$d(p, q) = \sqrt{(p_1 - q_1)^2}$$

Distance de Manhattan : Utile dans les grilles comme les images. Si vous avez deux points dans un espace n-dimensionnel, $p=(p_1, p_2, \dots, p_n)$ et $q=(q_1, q_2, \dots, q_n)$

$$d(p, q) = |p_1 - q_1| + |p_2 - q_2| + \dots + |p_n - q_n|$$

Distance de Minkowski : Une généralisation des distances euclidienne et de Manhattan. Si vous avez deux points dans un espace n-dimensionnel, $p=(p_1, p_2, \dots, p_n)$ et $q=(q_1, q_2, \dots, q_n)$

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

2.2.3 Méthodes de liaison

Lorsque des groupes de plus de deux observations sont formés, il faut décider de la distance entre les groupes. Les méthodes de liaison courantes incluent :

Liaison minimum (single-linkage) : La distance entre deux groupes est égale à la plus petite distance entre un membre de chaque groupe.

Liaison maximum (complete-linkage) : La distance est définie comme la plus grande distance entre les membres des deux groupes.

Liaison moyenne (average-linkage) : La distance est la moyenne de toutes les distances inter-groupes.

Liaison du centroïde : La distance entre le centre de masse de deux groupes.

2.2.4 Algorithme

Voici une description de l'algorithme de classification hiérarchique agglomérative :

1. **Initialisation** : Commencer avec nn groupes, chaque groupe contenant une observation.
2. **Calcul de la matrice de distance** : Calculer la distance entre chaque paire de groupes selon la métrique choisie.
3. **Fusion des groupes** : Identifier la paire de groupes avec la plus petite distance et les fusionner pour former un nouveau groupe.
4. **Mise à jour de la matrice de distance** : Mettre à jour la matrice de distance pour refléter la fusion des deux groupes..
5. **Répétition** : Répéter les étapes 3 et 4 jusqu'à ce que tous les groupes soient fusionnés en un seul.

2.3 Partie Codage

2.3.1 Packages utilisés

Les Packages utilisés dans ce Code sont :

```
1 import csv
2 from io import StringIO
3 from math import sqrt
```

Listing 2.1: Package Utilisées

NB : Tous les packages utilisés sert soit à des calculs mathématiques ou à l'importation des données

2.3.2 Fonction Read CSV

Cette fonction lit des données numériques à partir d'un fichier CSV. Elle ignore les valeurs non numériques et les lignes qui ne peuvent pas être converties en nombres flottants. La fonction utilise StringIO pour traiter le contenu textuel du fichier.

```
1 def read_csv(file):
2     """Reads numeric data from a CSV file object, ignoring non-numeric values."""
3     data = []
```



```

4     text_stream = StringIO(file.read().decode('utf-8')) # Decode binary file to
      text
5     reader = csv.reader(text_stream)
6     for row in reader:
7         try:
8             data.append([float(item) for item in row])
9         except ValueError:
10            continue # Skip rows with non-numeric values
11     return data

```

Listing 2.2: Read CSV Function

2.3.3 Calcul Distance Euclidean

Cette fonction calcule la distance euclidienne entre deux points.

```

1 def euclidean_distance(point1, point2):
2     """Calculate the Euclidean distance between two points."""
3     return sqrt(sum((p - q) ** 2 for p, q in zip(point1, point2)))

```

Listing 2.3: Calcul Distance Euclidean

2.3.4 Fonction de Linkage

Cette fonction calcule la distance de liaison moyenne entre deux clusters. Elle détermine la moyenne des distances euclidiennes entre tous les points du premier cluster et tous les points du second cluster.

```

1 def average_linkage(cluster1, cluster2, data):
2     """Calculate the average linkage distance between two clusters."""
3     distances = [euclidean_distance(data[i], data[j]) for i in cluster1 for j in
      cluster2]
4     return sum(distances) / len(distances)

```

Listing 2.4: Fonction de Linkage

2.3.5 Fonction De Classification Hiérarchique

Cette fonction réalise un clustering hiérarchique agglomératif. Partant de clusters initiaux où chaque point de données est considéré comme un cluster individuel, elle fusionne progressivement les clusters les plus proches jusqu'à ce que le nombre désiré de clusters soit atteint. La fusion est basée sur la distance de liaison moyenne calculée précédemment.

```

1 def agglomerative_clustering(data, n_clusters):
2     clusters = [[i] for i in range(len(data))]
3     while len(clusters) > n_clusters:
4         # Find the two closest clusters based on average linkage
5         min_distance = float('inf')
6         to_merge = (0, 1) # Default pair of clusters to merge
7         for i in range(len(clusters)):
8             for j in range(i + 1, len(clusters)):
9                 distance = average_linkage(clusters[i], clusters[j], data)
10                if distance < min_distance:
11                    min_distance = distance
12                    to_merge = (i, j)
13
14        # Merge the two closest clusters
15        clusters[to_merge[0]].extend(clusters[to_merge[1]])
16        del clusters[to_merge[1]]
17
18    # Convert cluster indices to original data points
19    clustered_data = [[data[index] for index in cluster] for cluster in clusters]
20    return clustered_data

```

Listing 2.5: Fonction De Classification Hiérarchique

2.4 Conclusion

La classification hiérarchique est une méthode puissante et versatile de l'analyse de données qui permet de découvrir des structures intrinsèques et des relations entre les données. Sur le plan théorique, cette technique se distingue par sa capacité à produire une hiérarchie de clusters, qui peut être visualisée sous forme de dendrogramme, offrant ainsi une compréhension intuitive et détaillée des niveaux de regroupement des données.

Du point de vue du codage, la mise en œuvre de la classification hiérarchique requiert une attention particulière à la gestion des distances entre les points de données ou les clusters existants. Les fonctions telles que le calcul de la distance euclidienne, la liaison moyenne, et l'algorithme agglomératif illustrent bien la manière dont les concepts théoriques sont traduits en procédures opérationnelles. Ces fonctions sont cruciales pour garantir que le clustering reflète fidèlement les distances et les similarités entre les données.

Chapter 3

K-means

3.1 Définition

L'algorithme **K-means** est une méthode de clustering non supervisée, utilisée pour regrouper des données en K clusters distincts. C'est une technique très utilisée en data science pour segmenter des données selon des caractéristiques similaires. Voici une explication détaillée de l'algorithme K-means, incluant les aspects théoriques et mathématiques.

3.2 Partie Théorique

3.2.1 Initialisation

Choix de K : Le nombre de clusters K , est déterminé à l'avance. Ce choix peut dépendre de la connaissance du domaine ou être déterminé par des méthodes comme la méthode du coude.

Sélection des centroids initiaux : Les centroids de départ sont sélectionnés aléatoirement parmi les points de données ou selon une méthode plus sophistiquée pour améliorer la convergence de l'algorithme.

3.2.2 Affectation des clusters

Calcul de la distance : Pour chaque point de données, calculez la distance (généralement euclidienne) entre ce point et chaque centroid.

Attribution au cluster : Chaque point est attribué au cluster dont le centroid est le plus proche. Cette étape minimise la distance entre les points de données et leur centroid associé, dans le but de former des clusters compacts.

3.2.3 Convergence

Critère d'arrêt : L'algorithme répète les étapes d'affectation des clusters et de mise à jour des centroids jusqu'à ce que l'un des critères suivants soit rempli : les centroids ne changent plus, les changements dans la fonction objectif sont minimales, ou un nombre maximal d'itérations est atteint.

Evaluation de la stabilité : La stabilité des centroids entre les itérations indique que l'algorithme a convergé.

3.3 Partie Codage

3.3.1 Packages utilisés

Les Packages utilisés dans ce Code sont :

```
1 import csv
2 from math import sqrt, fsum
3 from random import sample
4 from collections import defaultdict
5 from functools import partial
6 from io import StringIO
```

Listing 3.1: Package Utilisées

NB : Tous les packages utilisés sert soit à des calculs mathématiques ou à l'importation des données

3.3.2 Fonction Read CSV

Cette fonction lit des données numériques à partir d'un objet fichier CSV, en ignorant les valeurs non numériques. Elle prend en entrée un fichier (sous forme d'objet fichier), lit le fichier ligne par ligne, et convertit chaque ligne en une liste de nombres flottants. Elle retourne une liste de ces listes numériques.

```

1 def read_data_from_csv(file):
2     """Reads numeric data from a CSV file object, ignoring non-numeric values."""
3     data = []
4     text_stream = StringIO(file.read().decode('utf-8')) # Decode binary file to
5     text
6     reader = csv.reader(text_stream)
7     for row in reader:
8         try:
9             data.append([float(item) for item in row])
10        except ValueError:
11            continue # Skip rows with non-numeric values
12    return data

```

Listing 3.2: Fonction Read CSV

3.3.3 Fonction Mean

La fonction Mean calcule la moyenne arithmétique de manière précise. la moyenne est utilisée pour calculer les nouveaux centroids à partir des points attribués à chaque cluster

```

1 def mean(data):
2     'Accurate arithmetic mean'
3     if isinstance(data, list) == False:
4         data = list(data)
5     return fsum(data) / len(data)

```

Listing 3.3: Fonction Mean

3.3.4 Fonction Transpose

La transposition d'une matrice est nécessaire pour la fonction de recalcul des centroids. En K-means, les centroids sont mis à jour en calculant la moyenne de chaque dimension des points dans un cluster.

```

1 def transpose(matrix):
2     'Swap rows with columns for a 2-D array'
3     return zip(*matrix)

```

Listing 3.4: Fonction Transpose

3.3.5 Fonction Distance

La fonction distance calcule la distance euclidienne entre deux points. La mesure de

distance est essentielle pour attribuer chaque point de données au centroid le plus proche dans le processus de clustering.

```
1 def distance(p, q, sqrt=sqrt, fsum=fsum, zip=zip):
2     'Multi-dimensional euclidean distance between points p and q'
3     return sqrt(fsum((x1 - x2) ** 2.0 for x1, x2 in zip(p, q)))
```

Listing 3.5: Fonction Distance

3.3.6 Fonction Assign Data

La fonction assign data est cruciale pour le processus de clustering. Elle attribue chaque point à un cluster en fonction du centroid le plus proche, en utilisant la fonction distance. Cette étape est au cœur de l'algorithme K-means.

```
1 def assign_data(centroids, data):
2     """Assign data the closest centroid"""
3     d = defaultdict(list)
4     for point in data:
5         # Convert centroid to a tuple to use it as a dictionary key
6         centroid = min(centroids, key=partial(distance, point))
7         d[tuple(centroid)].append(point)
8     return dict(d)
```

Listing 3.6: Fonction Assign DATA

3.3.7 Fonction Compute Centroids

Cette fonction calcule les centroids de chaque cluster en prenant la moyenne des points attribués à chaque cluster. Le recalcul des centroids est une étape vitale dans chaque itération de K-means, car elle vise à optimiser la position des centroids pour mieux représenter les points de données de chaque cluster.

```
1 def compute_centroids(groups):
2     """Compute the centroid of each group"""
3     return [list(map(mean, transpose(group))) for group in groups]
```

Listing 3.7: Fonction Compute Centroids

3.3.8 Fonction Principale

K-means est la fonction principale qui orchestre l'ensemble du processus de clustering. Elle initialise les centroids, attribue les points aux clusters, recalcule les centroids,

et répète ces étapes pour un nombre défini d'itérations. Cette fonction encapsule la logique itérative de K-means.

```
1 def k_means(data, k, iterations=10):
2     """Return k-centroids for the data"""
3     data = list(data)
4     # Ensure centroids are initialized as tuples if needed
5     centroids = [tuple(point) for point in sample(data, k)]
6     for i in range(iterations):
7         labeled = assign_data(centroids, data)
8         # Convert tuples back to lists if necessary
9         centroids = [list(centroid) for centroid in compute_centroids(labeled.values
10                                ())]
11     return centroids
```

Listing 3.8: Fonction Principale

3.3.9 Fonction Quality

La fonction quality évalue la qualité des clusters formés en calculant la moyenne des carrés des distances de chaque point à son centroid. Cette métrique de qualité aide à évaluer la compacité des clusters formés et est souvent utilisée pour ajuster les paramètres de l'algorithme

```
1 def quality(labeled):
2     'Mean value of squared distances from data to its assigned centroid'
3     return mean(distance(c, p) ** 2 for c, pts in labeled.items() for p in pts)
```

Listing 3.9: Fonction Quality

3.4 Conclusion

L'algorithme K-means est un outil de clustering puissant et largement utilisé dans le domaine de l'analyse de données et du machine learning pour grouper des données en clusters distincts. Sa popularité est due à sa simplicité, son efficacité et sa facilité d'implémentation, même sur de grands ensembles de données. Cependant, K-means n'est pas exempt de défis et de limitations.

Chapter 4

Interface

4.1 Outils Utilisées

4.1.1 Python

Python est un langage de programmation interprété, de haut niveau, et généraliste. Connu pour sa syntaxe claire et sa lisibilité, il est largement utilisé pour le développement web, l'analyse de données, l'intelligence artificielle, le calcul scientifique, et plus encore. Python supporte plusieurs paradigmes de programmation, y compris la programmation impérative, orientée objet et fonctionnelle.

4.1.2 Streamlit

Streamlit est un framework open-source pour créer des applications web pour la visualisation de données et le machine learning en Python. Il permet aux développeurs de créer des applications interactives et élégantes avec peu de code, et il se concentre sur la rapidité et la facilité d'utilisation, facilitant le partage de résultats d'analyses de données et de modèles de machine learning.

4.1.3 HTML (HyperText Markup Language)

HTML est le langage de balisage standard pour créer et structurer des pages web. Il utilise des balises pour délimiter les éléments de contenu tels que les titres, paragraphes, liens, listes, et autres éléments incorporés comme des images et des vidéos. HTML forme la structure de base de toute page web.

4.1.4 CSS

CSS est un langage de feuille de style utilisé pour décrire la présentation d'un document écrit en HTML ou XML. CSS permet de contrôler le style et la mise en page des pages web, incluant des ajustements comme les couleurs, la taille des polices, l'espacement, et la réactivité du design aux différents appareils et tailles d'écran.

4.1.5 JavaScript

JavaScript est un langage de programmation de scripts principalement connu pour sa capacité à créer des interactions dynamiques sur les pages web. Il est exécuté sur le navigateur et peut manipuler le contenu du document, permettre des interactions en temps réel sans recharger la page, et beaucoup plus, rendant les sites web plus interactifs et fonctionnels.

4.1.6 Bootstrap

Bootstrap est un framework front-end open-source pour développer des applications web et des sites mobiles. Il fournit des modèles de conception basés sur HTML et CSS, ainsi que des composants JavaScript optionnels. Bootstrap facilite la création de designs web responsifs et modernes avec une grille flexible et des composants préconçus tels que des boutons, des cartes, des alertes, et plus encore.

4.2 Pages

4.2.1 HomePage

HomePage

 Hierarchy Clustering

 Kmeans Clustering

Deploy

“Doesn't matter how much data you have, it's whether you use it successfully that counts.”



Ziad Ouahbi

Filière : FID1


Année Universitaire : 2023/2024


Rapport Du Projet

Download Rapport

4.2.2 Classification Hiérarchique Page

HomePage


 Hierarchy Clustering

 Kmeans Clustering


Deploy

Hierarchy Clustering

Upload Your Dataset:

 Drag and drop file here
Limit 200MB per file

Browse files

 synthetic_dataset.csv 1.7KB

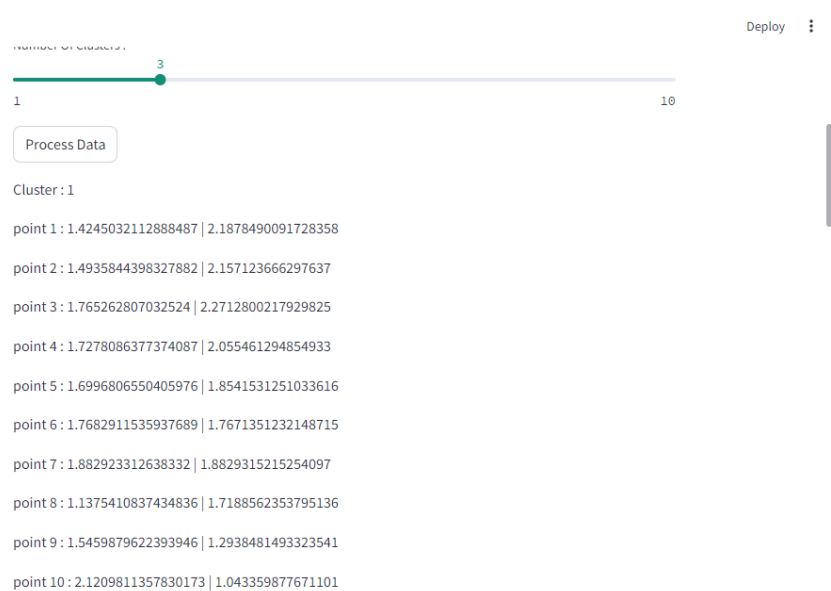
	X	Y
0	1.4243	2.1878
1	2.649	5.8362
2	7.2607	7.6401
3	2.7328	1.8871
4	1.6991	2.9261
5	2.8039	5.2682
6	8.4113	7.3896
7	7.5804	7.8454

×

HomePage

Hierarchy Clustering

Kmeans Clustering

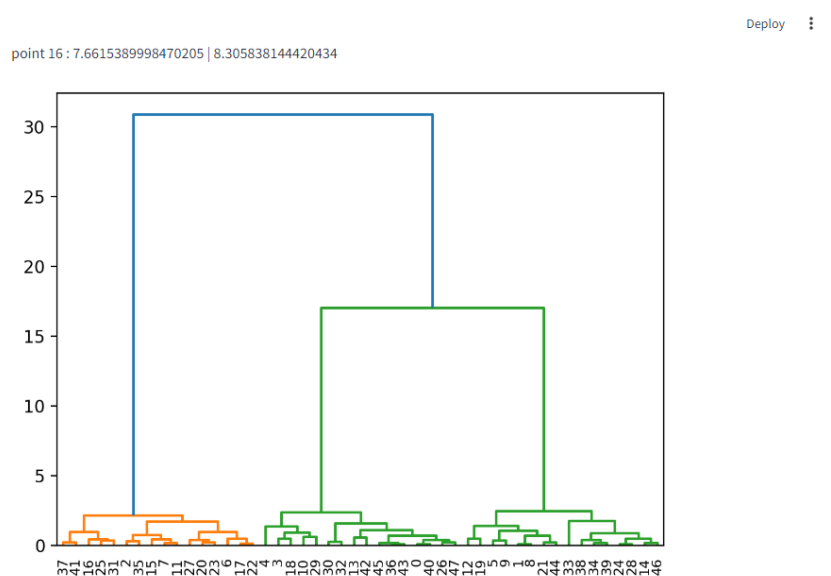


×

HomePage


Hierarchy Clustering


Kmeans Clustering



4.2.3 K-means Page


HomePage

 Hierarchy Clustering


 Kmeans Clustering

K-means Clustering

Upload Your Dataset :

 Drag and drop file here
Limit 200MB per file

Browse files

 synthetic_dataset.csv 1.7KB

X

	X	Y
0	1.4245	2.1878
1	2.649	5.8362
2	7.2607	7.6401
3	2.7328	1.8871
4	1.6991	2.9261
5	2.8039	5.2682
6	8.4113	7.3896



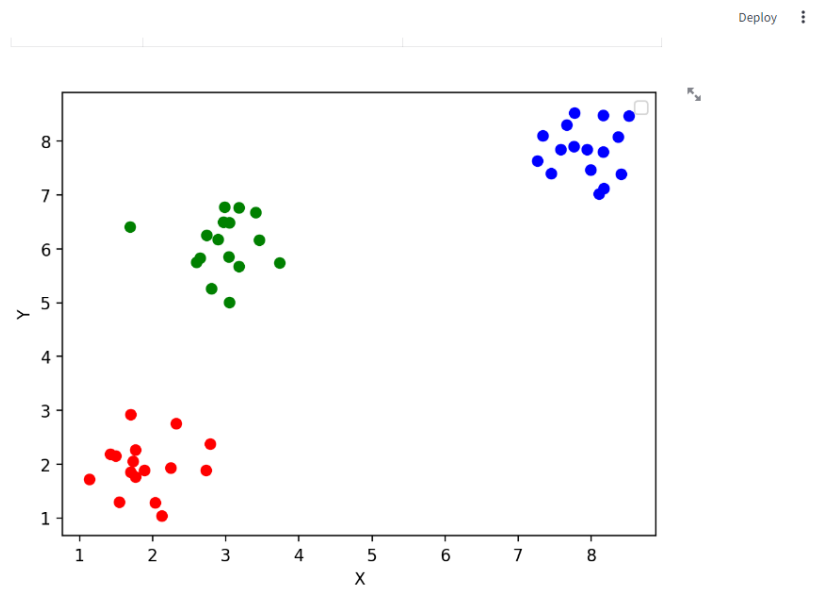
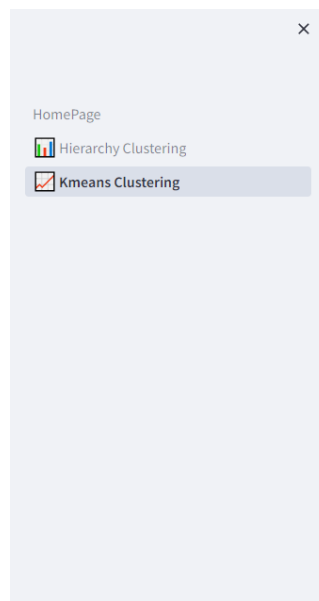
Number Of Clusters :



Process Data

Centroid: (1.8996317083081165, 1.9630609384652717)

	x	y
0	1.4245	2.1878
1	2.7328	1.8871
2	1.6991	2.9261
3	2.7896	2.3837
4	1.5460	1.2938
5	2.2484	1.9309
6	1.7653	2.2713
7	2.3238	2.7615
8	2.1210	1.0434



Chapter 5

Conclusion

Le projet détaillé dans le document est une initiative admirable qui fusionne l'analyse de données avec le développement web pour créer un site web interactif utilisant des méthodes de classification hiérarchique et K-means. Cette plateforme, conçue en Python et intégrant le framework Streamlit, vise à simplifier et à démocratiser l'accès à des analyses de données complexes pour une audience variée, allant des professionnels du marketing aux chercheurs académiques. L'accent est mis sur l'interface utilisateur, développée avec des technologies telles que HTML, CSS, JavaScript, et Bootstrap, pour offrir une expérience utilisateur fluide et interactive. Ce projet ne se contente pas seulement de fournir des outils d'analyse de données avancés, mais il contribue également à la formation et à l'éducation des utilisateurs en rendant les techniques de data science plus accessibles et compréhensibles, ce qui est essentiel dans un monde de plus en plus piloté par les données.