Université Cadi Ayyad
École Supérieure De Technologie-Safi
Département : Informatique
Filière : LP

Option : Ingénierie des Systèmes d'Information et Réseaux

Rapport sur le Projet de L'Intelligence Artificielle

Segmentation des Clients du Mall : Implémentation et Comparaison des Algorithmes K-Means et DBSCAN

Réalisé par :

M. DOUKKANI Mohamed
M. ZAGHOU Mohamed

Encadré par :

Mme. Mounir Ilham

Année Universitaire : 2024-2025

Remerciements

La réalisation de ce projet a été le fruit d'une collaboration exceptionnelle au sein d'une équipe engagée et passionnée. Chaque membre a contribué par son expertise, sa créativité et son dévouement, enrichissant ainsi la qualité et la portée de notre travail. Ce projet a également été marqué par un esprit d'équipe solide, favorisant l'échange d'idées et l'apprentissage mutuel à chaque étape, de la conception à la finalisation.
Nous exprimons également notre profonde gratitude à Mme. Mounir Ilham pour son soutien précieux tout au long de ce projet. Ses conseils avisés, son engagement constant et sa confiance en nos capacités ont été une source majeure de motivation et d'inspiration, nous permettant de relever les défis et de mener ce travail à son aboutissement.

Table des matières

Table des figures

Liste des Abréviations

- **K-Means** : K-means Clustering Algorithm
- **PCA** : Principal Component Analysis
- **EDA** : Exploratory Data Analysis
- **API** : Application Programming Interface
- **Jupyter** : Jupyter Notebook (interactive computing environment)
- **GitHub** : GitHub (platform for version control and collaboration)
- **DBI** : Davies-Bouldin Index
- **SS** : Silhouette Score
- **ML** : Machine Learning
- **AI** : Artificial Intelligence
- **DBSCAN** : Density-Based Spatial Clustering of Applications with Noise
- **CSV** : Comma-Separated Values (data file format)
- **SVM** : Support Vector Machine
- **ROC** : Receiver Operating Characteristic (curve)
- **TPR** : True Positive Rate
- **FPR** : False Positive Rate
- **GPU** : Graphics Processing Unit
- **CPU** : Central Processing Unit
- **KDD** : Knowledge Discovery in Databases
- **XML** : Extensible Markup Language

Résumé

Le rapport documente le projet de segmentation des clients en utilisant les algorithmes K-Means et DBSCAN, en détaillant chaque étape du processus, les outils et langages utilisés, ainsi que les résultats obtenus.
Après une introduction sur la segmentation des clients et une présentation des algorithmes, le rapport explore les aspects techniques de leur implémentation.
Il commence par présenter les outils et langages utilisés, notamment Anaconda, Jupyter, Python, ainsi que des bibliothèques comme pandas, numpy, seaborn, et scikit-learn. Ensuite, une présentation détaillée du processus de segmentation est fournie, incluant la préparation des données, l'analyse exploratoire et la normalisation des données.
Les algorithmes K-Means et DBSCAN sont expliqués en profondeur, avec des détails sur leur fonctionnement respectif et leur mise en œuvre. Le rapport met en évidence l'évaluation des clusters pour les deux algorithmes, avec des indicateurs comme le Silhouette Score et l'Indice de Davies-Bouldin, et fournit des visuels pour illustrer la qualité de la segmentation.
Enfin, une comparaison des performances entre K-Means et DBSCAN est réalisée, permettant d'analyser les avantages et inconvénients de chaque méthode dans le contexte des données étudiées. Le rapport conclut sur les résultats obtenus, en soulignant les insights pertinents pour les entreprises, les applications possibles dans la personnalisation du marketing, et les recommandations pour le choix de l'algorithme selon les besoins.

Introduction Générale

L'analyse des données occupe une place essentielle dans de nombreux secteurs, notamment en marketing et en gestion de la relation client. Parmi les domaines clés, la segmentation des clients joue un rôle crucial en permettant aux entreprises de mieux comprendre les comportements et besoins de leurs clients, afin d'optimiser leurs stratégies marketing et d'améliorer l'expérience utilisateur. Pour atteindre cet objectif, les algorithmes de clustering, tels que K-Means et DBSCAN, s'avèrent être des outils puissants pour diviser un ensemble de clients en groupes homogènes en fonction de leurs caractéristiques et comportements.

Ce rapport présente un projet de segmentation des clients d'un centre commercial en utilisant et comparant les algorithmes K-Means et DBSCAN. L'objectif est d'analyser un ensemble de données de clients, d'identifier des groupes distincts (clusters) et de fournir des recommandations basées sur les comportements d'achat et les données démographiques. La segmentation ainsi obtenue pourra être utilisée pour personnaliser les campagnes marketing, améliorer la gestion des relations client, et mieux comprendre les différences entre les deux approches algorithmiques.

Le rapport est structuré en plusieurs sections, détaillant les différentes étapes du projet. Après une présentation des outils, langages et bibliothèques utilisés, nous explorerons la préparation des données, l'analyse exploratoire et la normalisation des variables pertinentes. Ensuite, nous appliquerons les algorithmes K-Means et DBSCAN, en expliquant leur fonctionnement respectif, en déterminant les paramètres optimaux, et en évaluant la qualité des segmentations obtenues à l'aide de métriques telles que le Silhouette Score. Des visualisations des clusters permettront d'interpréter les groupes identifiés et de mettre en évidence les forces et limites des deux méthodes.

En conclusion, le rapport fournira des insights précieux pour les entreprises, en démontrant comment le choix d'un algorithme de clustering peut influencer les résultats et leur pertinence. Cette analyse comparative mettra en lumière les situations où K-Means ou DBSCAN s'avèrent plus adaptés, selon les caractéristiques des données et les objectifs du projet. En résumé, ce rapport illustre l'importance et la puissance des techniques de clustering dans l'analyse des données clients et leur application pratique dans la segmentation de marché.

Chapitre I

Généralités

1 Introduction

Dans cette section, nous explorerons les fondements du data mining, de l'apprentissage automatique et de la science des données, trois domaines interconnectés qui jouent un rôle crucial dans l'analyse intelligente des données. Le data mining, en tant qu'approche clé, vise à extraire des informations utiles à partir de grands ensembles de données, révélant des tendances, des modèles et des relations cachées. Parallèlement, l'apprentissage automatique permet aux machines d'apprendre à partir de données, facilitant ainsi la création de modèles prédictifs. La science des données fusionne ces disciplines pour offrir une compréhension globale des processus d'analyse de données.

Nous mettrons en lumière les avantages distincts de chaque composante de ce triptyque. Le data mining, en identifiant des schémas cachés, ouvre la voie à une prise de décision informée. L'apprentissage automatique se concentre sur la construction de modèles capables de généraliser à partir des données, renforçant ainsi la capacité prédictive. Enfin, la science des données adopte une approche intégrée, combinant ces disciplines pour une analyse approfondie des phénomènes complexes.

Dans le cadre du data mining, nous nous pencherons sur deux approches principales : les méthodes descriptives, qui résument les caractéristiques des données, et les méthodes prédictives, qui anticipent des événements futurs. Ces distinctions jouent un rôle fondamental dans la compréhension des techniques utilisées. Parmi ces techniques, le clustering occupe une place centrale. Ce dernier permet de regrouper des données selon des similarités, facilitant ainsi leur interprétation.

L'algorithme K-Means, au cœur de notre projet d'implémentation, est une méthode de partitionnement largement utilisée pour le clustering. Nous examinerons ses principes fondamentaux, ses avantages, ainsi que ses applications dans des contextes variés, tout en posant les bases nécessaires pour comprendre son importance dans le traitement de données complexes.

2 Data Mining

2.1 Définition de Data Mining

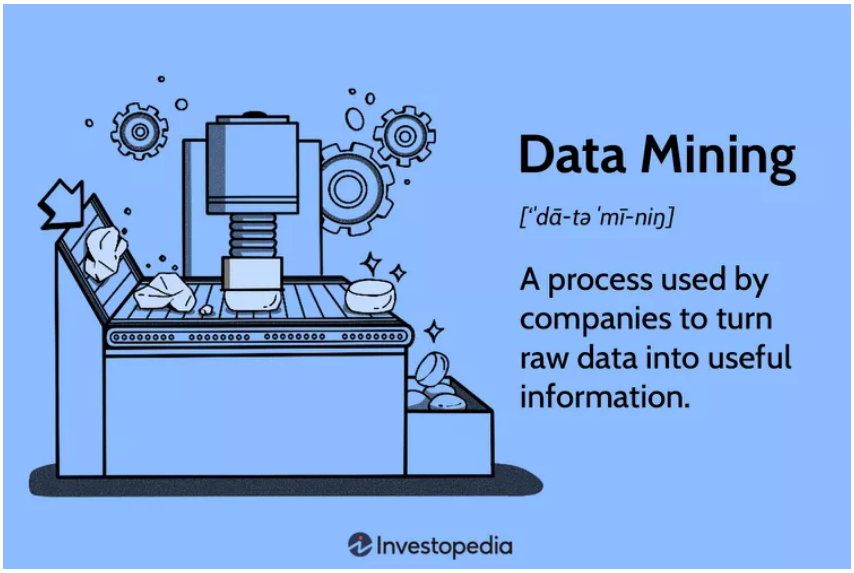


FIGURE I.1 – Data Mining

Le Data Mining, également connu sous le nom de fouille de données, est une discipline puissante qui repose sur l'exploration approfondie et l'analyse de vastes ensembles de données afin de découvrir des modèles, des tendances et des relations significatives. Il s'agit d'un processus itératif qui utilise des méthodes statistiques, mathématiques et informatiques avancées pour extraire des informations utiles à partir de données brutes. Le Data Mining vise à révéler des connaissances enfouies dans les données, offrant ainsi des perspectives nouvelles et exploitables pour la prise de décision.

2.2 Fondamentaux et Bases du Data Mining

Les fondamentaux du Data Mining reposent sur deux approches principales : les méthodes descriptives et les méthodes prédictives. Ces approches permettent de tirer parti des données pour révéler des informations utiles et orienter la prise de décision.

- **Méthodes Descriptives** : se concentrent sur une compréhension globale des données. Elles visent à résumer et à décrire les caractéristiques essentielles d'un jeu de données. Parmi les techniques descriptives les plus couramment utilisées, on trouve :

    1. *Clustering* : Une méthode qui regroupe les données en ensembles homogènes. Par exemple, le clustering peut être utilisé pour segmenter les clients en groupes similaires basés sur leurs comportements d'achat.

    2. *Règles d'Association* : Une technique qui identifie des relations fréquentes entre différentes variables. Par exemple, elle peut révéler une corrélation entre l'achat

de pain et de lait dans un supermarché.

- **Méthodes Prédictives** : en revanche, sont axées sur la modélisation des données afin d'anticiper des événements futurs. Ces méthodes incluent plusieurs techniques avancées telles que :

  1. *Régression* : Utilisée pour modéliser les relations fonctionnelles entre les variables et effectuer des prédictions basées sur ces relations.

  2. *Arbres de Décision* : Des structures arborescentes permettant de prendre des décisions séquentielles basées sur les caractéristiques des données.

  3. *Réseaux de Neurones* : Inspirés du fonctionnement du cerveau humain, ces modèles sont capables d'apprendre et de reproduire des modèles complexes à partir des données.

2.3 Perspective vers le Clustering, K-Means et DBSCAN

Cette exploration des approches descriptives et prédictives constitue une base solide pour comprendre les concepts fondamentaux du Data Mining. Dans les sections suivantes, nous approfondirons l'étude du clustering, en mettant un accent particulier sur les algorithmes K-Means et DBSCAN, deux méthodes puissantes et complémentaires pour segmenter les données en groupes homogènes.

K-Means, largement utilisé pour sa simplicité et son efficacité, jouera un rôle central dans notre projet. Il permettra d'illustrer comment diviser les données en clusters bien définis en se basant sur des centres de gravité. Cependant, nous intégrerons également DBSCAN (Density-Based Spatial Clustering of Applications with Noise), un algorithme basé sur la densité, capable de détecter des structures plus complexes et d'identifier des points de bruit souvent négligés par K-Means.

Cette double approche permettra d'évaluer les forces et limites de chaque méthode et de déterminer leur applicabilité selon les caractéristiques des données et les objectifs du projet. En combinant l'étude de ces deux algorithmes, nous viserons à fournir une vision globale et approfondie des techniques de clustering, en mettant en lumière leurs apports respectifs dans le traitement et l'analyse de données complexes.

3 Algorithmes de Clustering

3.1 Algorithmes de Partitionnement

Les algorithmes de partitionnement visent à diviser un ensemble de données en plusieurs groupes distincts, appelés partitions, en minimisant une mesure de dissimilarité intra-cluster. Ces méthodes nécessitent généralement la spécification préalable du nombre de clusters souhaités (k). Voici quelques-uns des algorithmes de partitionnement les plus

courants :

- **K-Means** : Cet algorithme regroupe les données en k clusters en minimisant la variance intra-cluster. Il fonctionne en calculant les barycentres des clusters et en réaffectant les points pour optimiser la configuration.
- **K-Medoids** : Similaire à K-Means, mais utilise des objets réels des clusters comme centres, rendant cette méthode plus robuste aux valeurs aberrantes.

3.2 Algorithmes Hiérarchiques

Les algorithmes hiérarchiques construisent une structure arborescente de clusters, souvent visualisée à l'aide d'un dendrogramme. Ces méthodes ne nécessitent pas de spécification préalable du nombre de clusters. Voici les principales variantes :

- **CAH (Classification Ascendante Hiérarchique)** : Commence par considérer chaque point comme un cluster individuel, puis fusionne itérativement les clusters les plus similaires jusqu'à ce qu'un seul cluster global soit formé.
- **CDH (Classification Descendante Hiérarchique)** : Part de l'ensemble des données comme un cluster global, puis le divise progressivement en clusters plus petits.

3.3 Algorithmes Basés sur la Densité

Les algorithmes basés sur la densité détectent des zones de forte densité dans l'espace des données, tout en identifiant les points bruyants comme des anomalies. Ces méthodes sont particulièrement utiles pour des ensembles de données ayant des formes complexes :

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** : Regroupe les points densément connectés en clusters et identifie les points isolés comme du bruit.

3.4 Comparaison des Algorithmes de Clustering

Voici une comparaison des principaux algorithmes de clustering sur différents critères tels que la complexité, la sensibilité aux valeurs aberrantes, et les types de données traitées :

TABLE I.1 – Comparaison des Catégories d'Algorithmes de Clustering

Catégorie	**Structure des Clusters**	**Avantages**	**Inconvénients**
Partitionnement (e.g., K-Means, K-Medoids)	Forme sphérique et bien séparée	Simplicité, rapidité, efficacité sur grands ensembles	Sensible aux valeurs aberrantes et nécessite k
Hiérarchique (e.g., CAH, CDH)	Arborescente, capture des sous-structures	Ne nécessite pas k, visualisation intuitive via dendrogramme	Complexité élevée pour grands ensembles
Basé sur la Densité (e.g., DBSCAN)	Forme arbitraire, zones densément peuplées	Insensibilité aux valeurs aberrantes, efficace pour clusters non linéaires	Sensible aux paramètres comme ε et $MinPts$

4 Algorithme K-Means

4.1 Définition

L'algorithme K-Means est une méthode de partitionnement largement utilisée dans le clustering. Il vise à regrouper n observations en k clusters de manière à minimiser la somme des variances intra-cluster. Chaque cluster est représenté par son centroïde, qui correspond au barycentre des points appartenant à ce cluster.

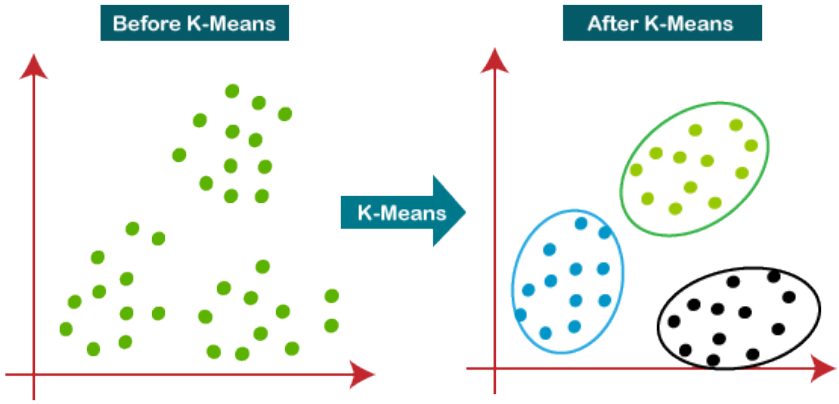


FIGURE I.2 – Algorithme K-Means

4.2 Processus de K-Means

Le processus de K-Means peut être décrit en cinq étapes principales :

1. **Initialisation :** Sélectionner k centroïdes initiaux, soit de manière aléatoire, soit en utilisant des heuristiques spécifiques.

2. **Assignation des points :** Chaque point est assigné au cluster dont le centroïde est le plus proche selon une métrique de distance, généralement la distance euclidienne.

La règle d'assignation est donnée par :

$$C_i = \{x_j : \|x_j - \mu_i\|^2 \leq \|x_j - \mu_h\|^2, \forall h = 1, \ldots, k\}$$

où C_i est le i-ième cluster, x_j est un point de données, et μ_i est le centroïde du i-ième cluster.

3. **Mise à jour des centroïdes :** Calculer le nouveau centroïde de chaque cluster comme la moyenne des points qui lui sont assignés :

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$

où $|C_i|$ est le nombre de points dans le cluster C_i.

4. **Itération :** Répéter les étapes 2 et 3 jusqu'à ce que les centroïdes ne changent plus ou qu'un critère de convergence soit atteint (par exemple, un nombre maximal d'itérations).

5. **Résultat final :** Les clusters finaux et leurs centroïdes respectifs sont obtenus.

4.3 Avantages de K-Means

- **Simplicité et Rapidité :** L'algorithme est simple à implémenter et efficace pour des ensembles de données de grande taille.

- **Convergence Rapide :** Il converge rapidement dans la majorité des cas grâce à son approche itérative.

- **Flexibilité :** Il peut être utilisé dans divers contextes, comme la segmentation de clients, l'analyse d'images, ou la classification non supervisée.

4.4 Limites de K-Means

- **Dépendance à k :** Le nombre de clusters (k) doit être défini à l'avance, ce qui peut nécessiter des tests multiples.

- **Sensibilité aux valeurs aberrantes :** Les points aberrants peuvent affecter considérablement les centroïdes.

- **Forme des clusters :** K-Means fonctionne mieux avec des clusters sphériques et de tailles similaires.

4.5 Exemple d'Application

Un exemple typique de l'utilisation de K-Means est la segmentation de clients dans un cadre commercial. Supposons qu'une entreprise souhaite regrouper ses clients en fonction de leur comportement d'achat (par exemple, montant dépensé et fréquence des achats).

1. Les données des clients sont collectées, incluant des attributs tels que le montant total dépensé et le nombre d'achats.

2. L'algorithme K-Means est appliqué avec $k = 3$ pour segmenter les clients en trois groupes : gros acheteurs, acheteurs modérés et petits acheteurs.

3. Les résultats permettent à l'entreprise de cibler chaque segment avec des stratégies marketing spécifiques.

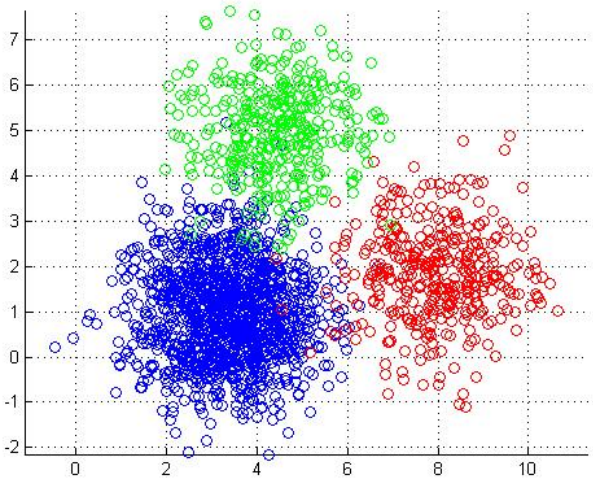


FIGURE I.3 – Exemple de segmentation client avec K-Means ($k = 3$).

Cette approche démontre l'efficacité de K-Means dans la simplification et l'interprétation des données complexes.

5 Algorithme DBSCAN

5.1 Définition

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) est un algorithme de clustering qui identifie les clusters dans un ensemble de données en se basant sur la densité des points. Contrairement aux méthodes de clustering traditionnelles, DBSCAN peut identifier des clusters de formes complexes et est résistant aux outliers. Il catégorise les points en trois types : les points noyaux (core points), les points frontières (border points), et les points de bruit (noise points). La densité d'un cluster est mesurée par le

nombre de points à proximité d'un point donné, défini par un rayon spécifique

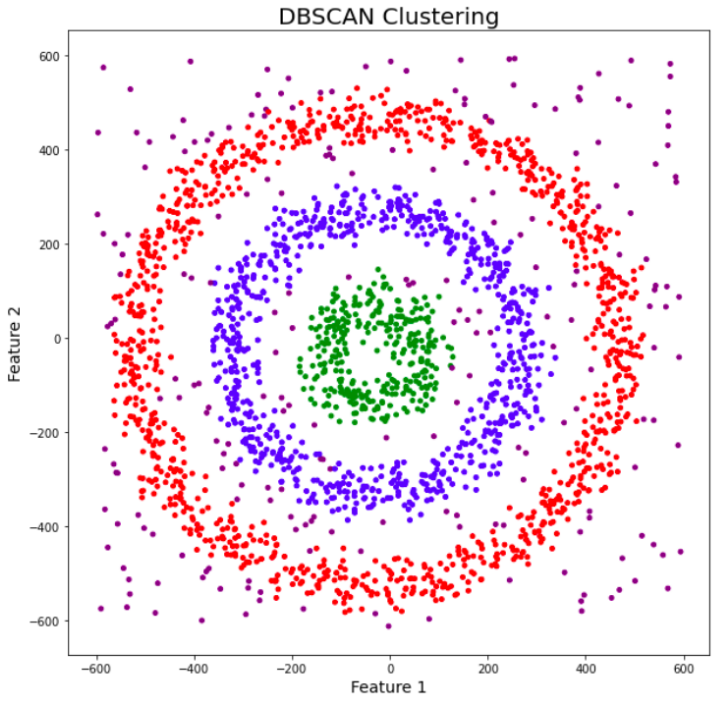


FIGURE I.4 – Algorithme DBSCAN

5.2 Fonctionnement de DBSCAN

L'algorithme DBSCAN utilise deux paramètres : la distance ϵ et le nombre minimum de points « MinPts » qui doivent se trouver dans un rayon ϵ pour que ces points soient considérés comme formant un cluster.

DBSCAN fonction de la maniere suivante :

1. DBSCAN commence par un point de données de départ arbitraire qui n'a pas été visité. Le voisinage de ce point est extrait en utilisant une distance epsilon ϵ.

2. S'il y a un nombre suffisant de points (selon les minPoints) dans ce voisinage, le processus de mise en cluster démarre et le point de données actuel devient le premier point du nouveau cluster. Sinon, le point sera étiqueté comme bruit (plus tard, ce point bruyant pourrait devenir la partie du cluster). Dans les deux cas, ce point est marqué comme «visité».

3. Pour ce premier point du nouveau cluster, les points situés dans son voisinage à distance se joignent également au même cluster. Cette procédure est ensuite répétée pour tous les nouveaux points qui viennent d'être ajoutés au groupe de cluster.

4. Ce processus des étapes 2 et 3 est répété jusqu'à ce que tous les points du cluster soient déterminés, c'est-à-dire que tous les points à proximité du ϵ voisinage du

cluster ont été visités et étiquetés.

5. Une fois terminé avec le cluster actuel, un nouveau point non visité est récupéré et traité, ce qui permet de découvrir un nouveau cluster ou du bruit. Ce processus se répète jusqu'à ce que tous les points soient marqués comme étant visités. A la fin de tous les points visités, chaque points a été marqué comme appartenant à un cluster ou comme étant du bruit.

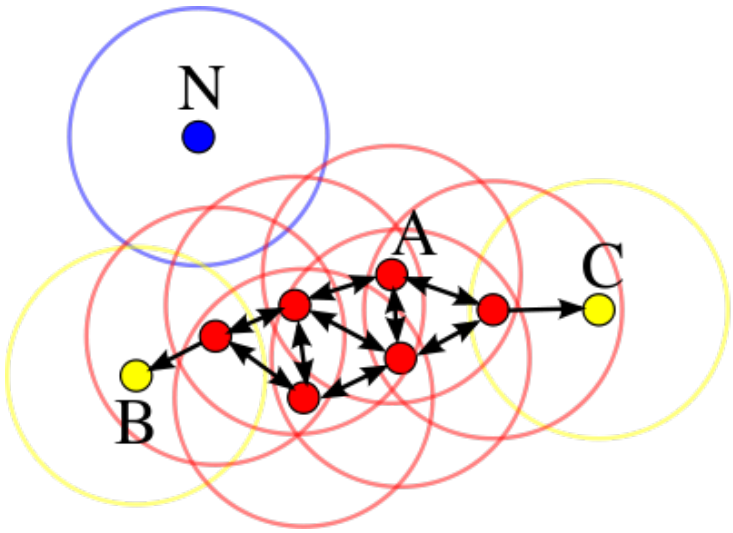


FIGURE I.5 – Fonctionnement de DBSCAN

5.3 Avantages de DBSCAN

L'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise) présente plusieurs avantages qui en font une méthode puissante pour le clustering, notamment :

- **Détection des formes arbitraires de clusters** : Contrairement à des algorithmes comme K-Means qui supposent des clusters de formes sphériques, DBSCAN est capable de détecter des clusters de formes non régulières. Cela en fait un choix privilégié pour des données complexes où les clusters peuvent avoir des structures variées.

- **Gestion du bruit** : DBSCAN permet d'identifier et d'isoler les points considérés comme du bruit (points aberrants), en les étiquetant comme des points "non attribués". Cela aide à améliorer la qualité des clusters en excluant les données non pertinentes ou erronées.

- **Pas de besoin de spécifier le nombre de clusters** : L'un des principaux avantages de DBSCAN est qu'il ne nécessite pas de définir à l'avance le nombre de clusters, contrairement à des algorithmes comme K-Means. L'algorithme détecte automatiquement le nombre de clusters en fonction de la densité des points.

- **Scalabilité et efficacité** : DBSCAN peut être efficace sur des jeux de données relativement grands, surtout lorsqu'il est implémenté avec des structures de données adaptées, comme les arbres KD ou les index R-tree, permettant ainsi une recherche rapide des voisins.

5.4 Limites de DBSCAN

Bien que DBSCAN soit un algorithme robuste pour de nombreuses situations, il présente également des limites qui peuvent limiter son utilisation dans certains contextes :

- **Sensibilité aux paramètres** : DBSCAN dépend de deux paramètres clés, à savoir la distance maximale entre deux points pour qu'ils soient considérés comme voisins ϵ et le nombre minimum de points pour former un cluster (MinPts). Le choix de ces paramètres est crucial et peut influencer la qualité du clustering. Si ces paramètres ne sont pas bien choisis, DBSCAN peut soit créer trop de petits clusters, soit ne pas détecter de clusters significatifs.

- **Difficulté avec des clusters de densité variable** : DBSCAN a du mal à traiter des clusters ayant des densités différentes. Si un ensemble de données contient des clusters de densité variée, DBSCAN peut avoir du mal à les détecter correctement, car il suppose que tous les clusters ont la même densité.

- **Performance avec de très grands jeux de données** : Bien que DBSCAN soit relativement rapide avec des jeux de données modérés, son efficacité peut diminuer avec des ensembles de données très volumineux, surtout sans l'utilisation d'index spécialisés. Dans ce cas, la complexité algorithmique peut entraîner des délais de calcul importants.

- **Difficulté avec les données à haute dimensionnalité** : Comme la plupart des algorithmes de clustering, DBSCAN souffre de la malédiction de la dimensionnalité. À mesure que le nombre de dimensions augmente, la notion de densité devient plus difficile à définir, ce qui rend l'algorithme moins performant et moins fiable sur des données de haute dimension.

6 Conclusion

Ce premier chapitre a permis de poser les bases théoriques indispensables à la compréhension du clustering, une technique fondamentale en data mining. Nous avons exploré les différentes catégories d'algorithmes de clustering, leurs caractéristiques distinctives ainsi que leurs avantages et limitations. Plus particulièrement, les algorithmes K-Means et DBSCAN ont été mis en avant. K-Means a été présenté comme une méthode de partitionnement puissante et largement utilisée dans diverses applications, tandis que DBSCAN a été valorisé pour sa capacité à détecter des clusters de formes arbitraires et à gérer le bruit de manière efficace.

L'étude des concepts théoriques constitue une étape cruciale avant de passer à la mise en œuvre pratique. Dans le chapitre suivant, nous appliquerons les algorithmes K-Means et DBSCAN au dataset *Mall Customers* pour démontrer leur utilité dans un cas concret de segmentation client. Cette implémentation permettra d'illustrer les concepts abordés, tout en mettant en lumière les défis et opportunités liés à l'utilisation de ces deux algorithmes sur des données réelles.

Chapitre II

Implémentation

1 Introduction

Dans ce deuxième chapitre, nous faisons la transition de la théorie à la pratique. L'implémentation devient l'occasion de concrétiser les concepts théoriques abordés dans le premier chapitre et de les appliquer dans un contexte réel. Cette section met en lumière les choix technologiques qui ont soutenu notre projet, en détaillant les outils, les langages de programmation et les bibliothèques que nous avons choisis pour réaliser cette implémentation.

Le Data Mining, étant un domaine vaste et complexe, nécessite des décisions technologiques judicieuses pour transformer les idées théoriques en solutions pratiques et fonctionnelles. Nous commencerons par analyser les technologies spécifiques que nous avons sélectionnées, expliquant les raisons derrière chaque choix et comment elles ont contribué à la mise en œuvre du projet.

Ensuite, nous présenterons le projet en lui-même, en décrivant ses objectifs, sa portée et son impact potentiel dans le domaine du Data Mining. Cette section offre un aperçu détaillé de la manière dont nous avons appliqué les algorithmes K-Means et DBSCAN au dataset *Mall Customers*, mettant en évidence les différences et complémentarités entre ces deux méthodes de clustering. Nous illustrerons comment chaque algorithme permet de segmenter les clients en groupes significatifs et comment leur efficacité peut varier selon la structure des données.

Ainsi, ce chapitre sert de pont entre la théorie et la pratique, montrant comment les concepts abordés sont intégrés dans un projet réel, transformant la théorie en applications tangibles et concrètes.

2 Outils, Langages et Bibliothèques Utilisés

2.1 Outils Utilisés

Pour mener à bien ce projet, plusieurs outils ont été utilisés afin de faciliter le développement, l'analyse et la gestion des données.

Le premier outil clé utilisé est **Anaconda**, une plateforme de gestion des environnements de développement Python. Anaconda permet une gestion efficace des bibliothèques et des versions de Python, tout en offrant des outils de science des données comme Jupyter

Notebook.

FIGURE II.1 – Anaconda

Nous avons utilisé **Jupyter Notebook** pour coder de manière interactive, visualiser les résultats et documenter chaque étape de notre travail.

FIGURE II.2 – Jupyter Notebook

En outre, **GitHub** a été utilisé pour la gestion du code source et la collaboration, permettant de versionner le projet et de faciliter le partage et le suivi des évolutions du projet au fur et à mesure de son développement.

FIGURE II.3 – Github

2.2 Langages de programmation

Le principal langage de programmation utilisé pour ce projet est **Python**. Python est largement adopté dans le domaine du Data Mining en raison de sa simplicité et de la

richesse de son écosystème de bibliothèques dédiées aux sciences des données. Ce langage offre une excellente flexibilité et est bien adapté pour manipuler et analyser de grandes quantités de données, ce qui en fait un choix optimal pour l'implémentation de l'algorithme de clustering K-Means.

FIGURE II.4 – Python

2.3 Bibliothèques Utilisées

L'implémentation de ce projet a fait appel à plusieurs bibliothèques Python qui facilitent les tâches de manipulation des données, de visualisation et de clustering.

```python
# 1. Setting Environment Variables (Optional)
import os
os.environ["OMP_NUM_THREADS"] = "1"  # Restrict OpenMP to use a single thread to avoid resource overuse
(if needed)

# 2. Libraries for Data Manipulation and Analysis
import pandas as pd  # For handling and processing tabular data (e.g., datasets, dataframes)
import numpy as np   # For numerical computations and array handling

# 3. Libraries for Data Visualization
import seaborn as sns  # For advanced and aesthetic statistical visualizations
import matplotlib.pyplot as plt  # For basic plotting and graph customization

# 4. Libraries for Clustering and Evaluation
from sklearn.cluster import KMeans  # K-Means algorithm for clustering data
from sklearn.preprocessing import StandardScaler  # For scaling data to standardize feature values
from sklearn.decomposition import PCA  # For reducing data dimensions before clustering (if needed)
from sklearn.metrics import silhouette_score, davies_bouldin_score  # For evaluating clustering quality

# 5. Additional Libraries for Enhancements
from matplotlib.patches import Ellipse  # To add ellipses in plots (e.g., visualizing cluster boundaries)
import warnings
warnings.filterwarnings('ignore')  # Suppress warnings for cleaner output
from scipy import stats  # For additional statistical computations (if required)
from yellowbrick.cluster import SilhouetteVisualizer  # For visualizing silhouette scores of clustering
models

# 6. Configure Inline Plotting (Specific to Jupyter Notebooks)
# Ensures that visualizations are displayed directly within the notebook interface
%matplotlib inline

# 7. Setting Default Styles for Visualizations
sns.set_style('whitegrid')  # Apply a clean "whitegrid" style to seaborn visualizations
plt.style.use('fivethirtyeight')  # Use the "fivethirtyeight" theme for matplotlib plots to enhance
appearance
```

FIGURE II.5 – Bibliothèques Utilisées

- **Pandas** (`import pandas as pd`) : Cette bibliothèque est utilisée pour la manipulation et le traitement des données. Elle permet de travailler efficacement avec des ensembles de données structurées (dataframes), facilitant ainsi les opérations de nettoyage, de transformation et d'analyse des données.

FIGURE II.6 – Pandas

- **NumPy** (`import numpy as np`) : NumPy est utilisé pour les opérations numériques avancées, notamment la gestion des tableaux et des matrices. Il est essentiel pour effectuer des calculs sur les données numériques de manière efficace.

FIGURE II.7 – NumPy

- **Seaborn** (`import seaborn as sns`) : Seaborn est une bibliothèque de visualisation avancée qui est utilisée pour créer des graphiques statistiques esthétiques et informatifs, offrant ainsi une meilleure compréhension des tendances et des relations dans les données.

FIGURE II.8 – Seaborn

- **Matplotlib** (`import matplotlib.pyplot as plt`) : Matplotlib est utilisée pour la création de graphiques et de figures personnalisés, permettant une visualisation plus précise des résultats obtenus lors de l'analyse des données.

FIGURE II.9 – Matplotlib

- **Scikit-learn** : Cette bibliothèque est l'une des plus utilisées pour l'apprentissage automatique. Elle offre plusieurs outils pour la mise en œuvre de l'algorithme K-Means :
  - `KMeans` : Implémentation de l'algorithme de clustering K-Means.
  - `StandardScaler` : Pour la normalisation des données, une étape cruciale avant d'appliquer le K-Means.
  - `PCA` : Pour la réduction dimensionnelle des données, afin de mieux visualiser les clusters dans un espace réduit.
  - `silhouette_score, davies_bouldin_score` : Métriques utilisées pour évaluer la qualité des clusters obtenus.

FIGURE II.10 – Scikit-learn

- **Matplotlib Patches** (`from matplotlib.patches import Ellipse`) : Utilisée pour ajouter des ellipses dans les visualisations, ce qui peut être utile pour mettre en évidence les clusters dans les graphiques.

Enfin, des styles de visualisation ont été définis pour améliorer l'aspect graphique des résultats. `Seaborn` a été configuré pour appliquer un style de grille blanche (`whitegrid`), tandis que `Matplotlib` utilise le style "fivethirtyeight" pour offrir un design plus professionnel.

3 Présentation du Projet

FIGURE II.11 – Segmentation des Clients

Ce projet se concentre sur la segmentation des clients à l'aide des algorithmes de clustering K-Means et DBSCAN. L'objectif principal est d'analyser un ensemble de données

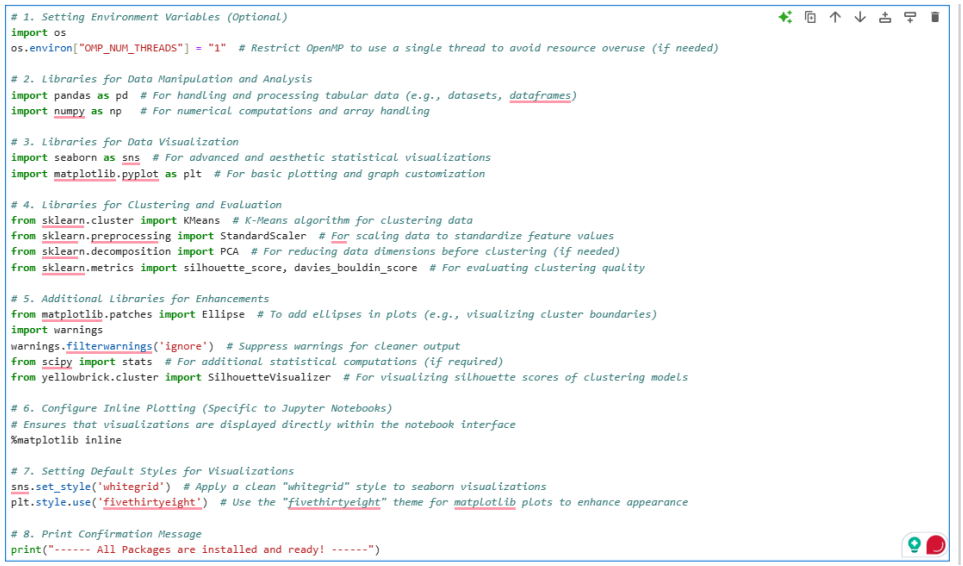
de clients d'un centre commercial afin d'identifier des groupes distincts basés sur des critères démographiques et des comportements d'achat. L'algorithme K-Means, étant basé sur une approche de partitionnement, permet de regrouper les clients en un nombre prédéfini de clusters, tandis que DBSCAN, un algorithme basé sur la densité, identifie des clusters de formes arbitraires et peut également gérer les points de bruit, c'est-à-dire les clients qui ne correspondent à aucun cluster spécifique.

La segmentation des clients est une pratique courante dans le domaine du marketing, car elle permet aux entreprises de mieux comprendre les préférences de leurs clients et d'adapter leurs stratégies en conséquence. En appliquant à la fois K-Means et DBSCAN, ce projet permet de comparer les performances de ces deux algorithmes et d'explorer leurs capacités à segmenter efficacement les données clients selon différents critères.

Le projet se compose des étapes suivantes :

3.1 Importation des Bibliothèques Nécessaires

La première étape consiste à importer les bibliothèques essentielles pour le projet. Cela inclut des bibliothèques telles que `pandas`, `numpy`, `matplotlib`, `seaborn`, ainsi que des bibliothèques spécifiques pour le clustering, comme `scikit-learn`. Cette étape est nécessaire pour préparer l'environnement de travail et s'assurer que toutes les fonctions nécessaires soient disponibles.



```
# 1. Setting Environment Variables (Optional)
import os
os.environ["OMP_NUM_THREADS"] = "1"  # Restrict OpenMP to use a single thread to avoid resource overuse (if needed)

# 2. Libraries for Data Manipulation and Analysis
import pandas as pd  # For handling and processing tabular data (e.g., datasets, dataframes)
import numpy as np   # For numerical computations and array handling

# 3. Libraries for Data Visualization
import seaborn as sns  # For advanced and aesthetic statistical visualizations
import matplotlib.pyplot as plt  # For basic plotting and graph customization

# 4. Libraries for Clustering and Evaluation
from sklearn.cluster import KMeans  # K-Means algorithm for clustering data
from sklearn.preprocessing import StandardScaler  # For scaling data to standardize feature values
from sklearn.decomposition import PCA  # For reducing data dimensions before clustering (if needed)
from sklearn.metrics import silhouette_score, davies_bouldin_score  # For evaluating clustering quality

# 5. Additional Libraries for Enhancements
from matplotlib.patches import Ellipse  # To add ellipses in plots (e.g., visualizing cluster boundaries)
import warnings
warnings.filterwarnings('ignore')  # Suppress warnings for cleaner output
from scipy import stats  # For additional statistical computations (if required)
from yellowbrick.cluster import SilhouetteVisualizer  # For visualizing silhouette scores of clustering models

# 6. Configure Inline Plotting (Specific to Jupyter Notebooks)
# Ensures that visualizations are displayed directly within the notebook interface
%matplotlib inline

# 7. Setting Default Styles for Visualizations
sns.set_style('whitegrid')  # Apply a clean "whitegrid" style to seaborn visualizations
plt.style.use('fivethirtyeight')  # Use the "fivethirtyeight" theme for matplotlib plots to enhance appearance

# 8. Print Confirmation Message
print("------ All Packages are installed and ready! ------")
```

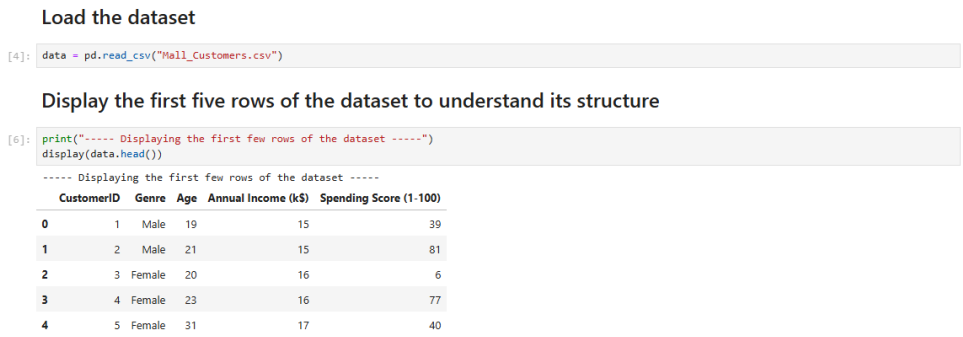
FIGURE II.12 – Importation des Bibliothèques Nécessaires

3.2 Chargement et Prétraitement du Dataset

Dans cette étape, nous chargeons l'ensemble de données et procédons à son prétraitement. Cela inclut la gestion des valeurs manquantes, le nettoyage des données, et la transformation des variables si nécessaire. Ce prétraitement est crucial pour garantir que les données soient adaptées aux algorithmes de clustering et qu'elles n'introduisent pas

d'erreurs ou de biais dans les résultats.

1. Chargeons l'ensemble de données.



Load the dataset

```
[4]: data = pd.read_csv("Mall_Customers.csv")
```

Display the first five rows of the dataset to understand its structure

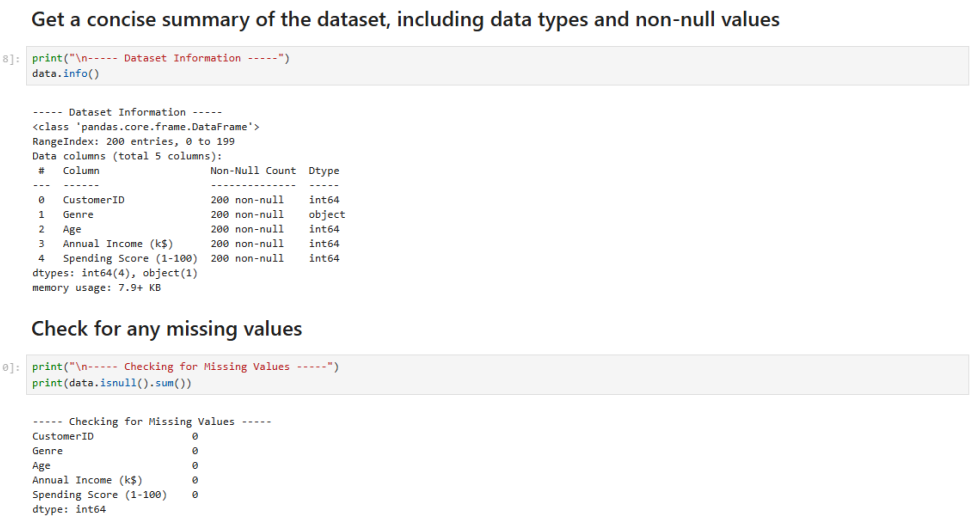
```
[6]: print("----- Displaying the first few rows of the dataset -----")
     display(data.head())
```

```
----- Displaying the first few rows of the dataset -----
```

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

FIGURE II.13 – Chargeons l'ensemble de données.

2. Explorons sa structure et les statistiques de base.



Get a concise summary of the dataset, including data types and non-null values

```
8]: print("\n----- Dataset Information -----")
    data.info()
```

```
----- Dataset Information -----
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Genre                   200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

Check for any missing values

```
0]: print("\n----- Checking for Missing Values -----")
    print(data.isnull().sum())
```

```
----- Checking for Missing Values -----
CustomerID              0
Genre                   0
Age                     0
Annual Income (k$)      0
Spending Score (1-100)  0
dtype: int64
```

FIGURE II.14 – Les statistiques de base.

3. Gérons les données catégorielles en encodant la colonne 'Genre'.

Encode the 'Genre' column: Male -> 0, Female -> 1

```
[12]: print("\n----- Encoding 'Genre' Column -----")
      data['Genre'] = data['Genre'].map({'Male': 0, 'Female': 1})
```

```
----- Encoding 'Genre' Column -----
```

Confirm the encoding process

```
[14]: print("\n----- Confirming Encoded Data -----")
      display(data.head())
```

```
----- Confirming Encoded Data -----
```

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	0	19	15	39
1	2	0	21	15	81
2	3	1	20	16	6
3	4	1	23	16	77
4	5	1	31	17	40

FIGURE II.15 – Les données catégorielles.

4. Supprimons les colonnes non pertinentes pour le clustering (par exemple, 'CustomerID').

Drop the 'CustomerID' column as it is irrelevant for clustering

```
[18]: print("\n----- Dropping 'CustomerID' column as it is irrelevant for clustering----- ")
      data.drop(columns=['CustomerID'], inplace=True)
```

```
----- Dropping 'CustomerID' column as it is irrelevant for clustering-----
```

Confirm changes

```
[20]: print("\n----- Preview of the dataset after preprocessing:-----")
      display(data.head())
```

```
----- Preview of the dataset after preprocessing:-----
```

	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	0	19	15	39
1	0	21	15	81
2	1	20	16	6
3	1	23	16	77
4	1	31	17	40

FIGURE II.16 – les colonnes non pertinentes

3.3 Analyse Exploratoire des Données (EDA)

L'Analyse Exploratoire des Données (EDA) permet de mieux comprendre les caractéristiques du dataset et les relations entre les différentes variables. À cette étape, des visualisations sont réalisées à l'aide de graphiques comme des histogrammes, des boxplots, et des cartes de chaleur des corrélations pour identifier des tendances, anomalies et corrélations pertinentes. Cela aide à mieux comprendre les données avant de procéder à la segmentation.

1. Répartition de l'âge des clients masculins et féminins

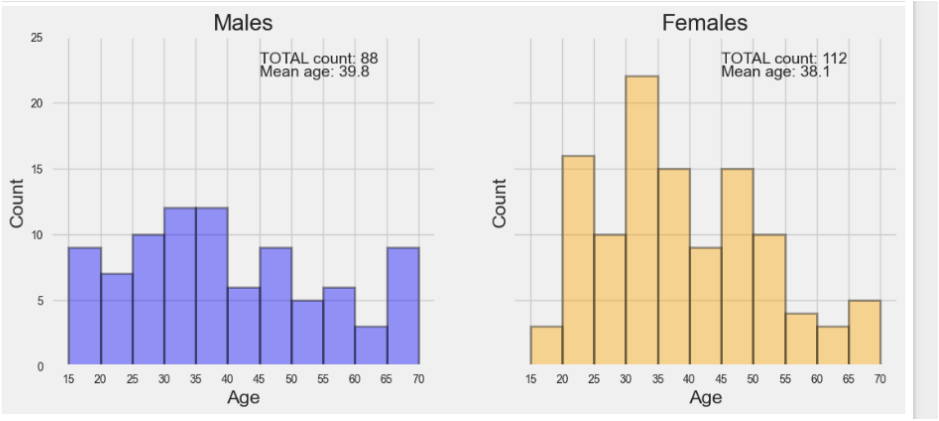


FIGURE II.17 – les colonnes non pertinentes

2. Âge vs Revenu annuel par genre

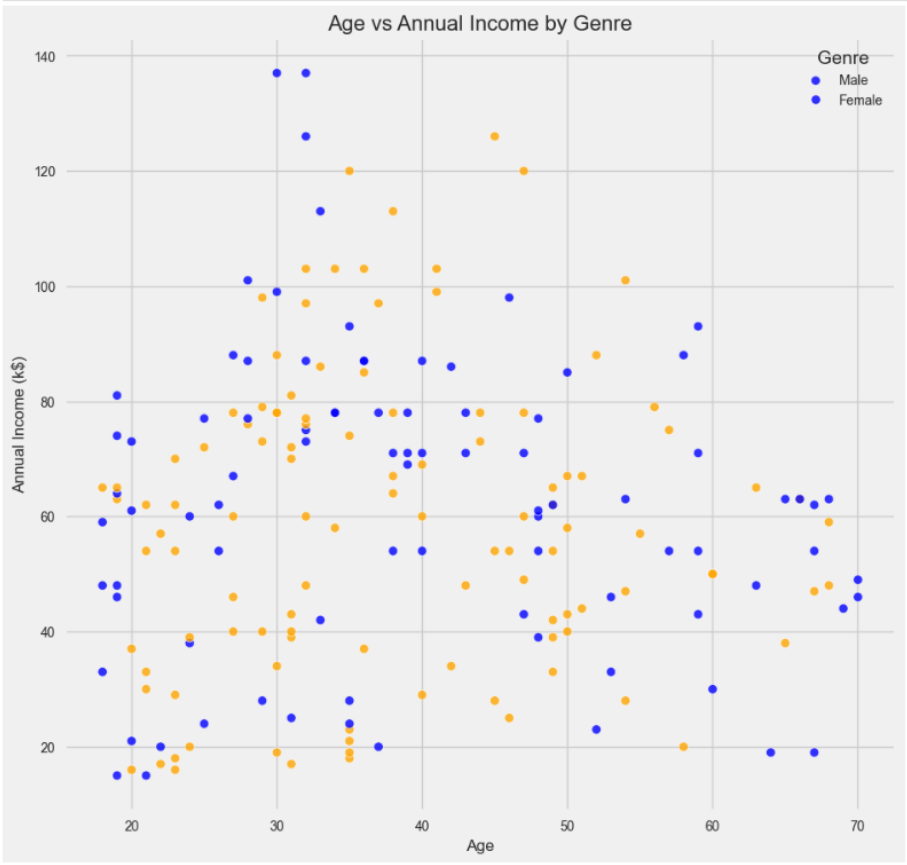


FIGURE II.18 – les colonnes non pertinentes

3. Répartition des scores de dépenses par genre

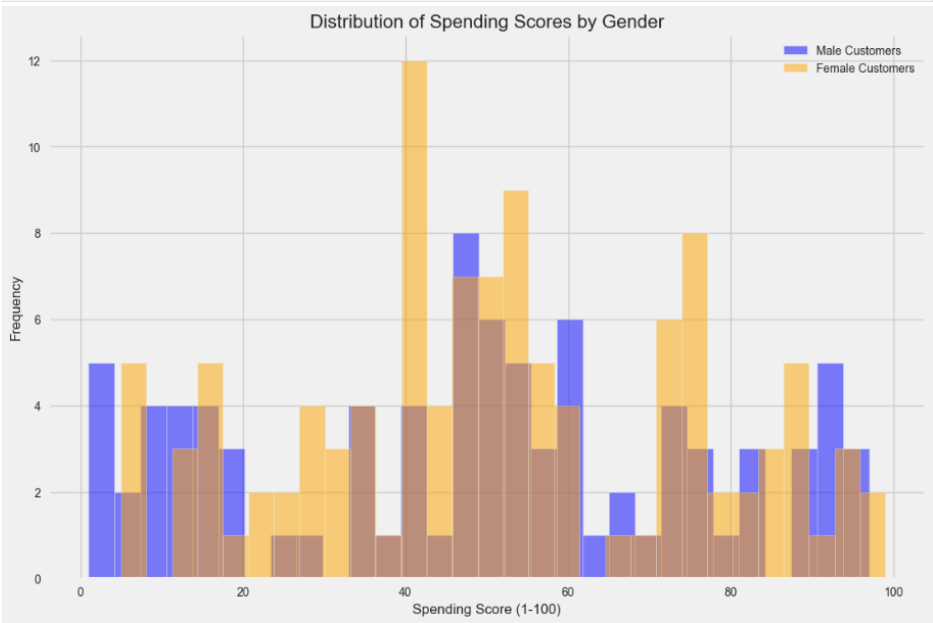


FIGURE II.19 – les colonnes non pertinentes

3.4 Création des Clusters avec K-Means

Dans cette étape, nous normalisons les données afin de garantir que toutes les variables aient la même échelle, ce qui est essentiel pour le bon fonctionnement de l'algorithme K-Means. Ensuite, les caractéristiques les plus pertinentes pour la segmentation sont sélectionnées, permettant de se concentrer sur les variables qui influencent le plus le comportement des clients, comme les dépenses mensuelles ou l'âge.

Dans cette étape, nous appliquons l'algorithme K-Means pour segmenter les clients en groupes distincts. Les sous-étapes sont les suivantes :

- Sélection des caractéristiques pertinentes pour le clustering.

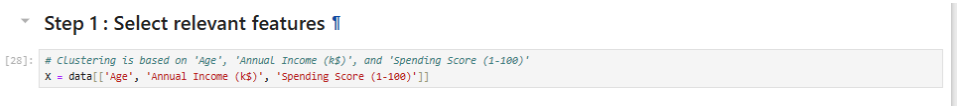


FIGURE II.20 – les caractéristiques pertinentes

- Application de la méthode du coude (Elbow Method) pour déterminer le nombre optimal de clusters.

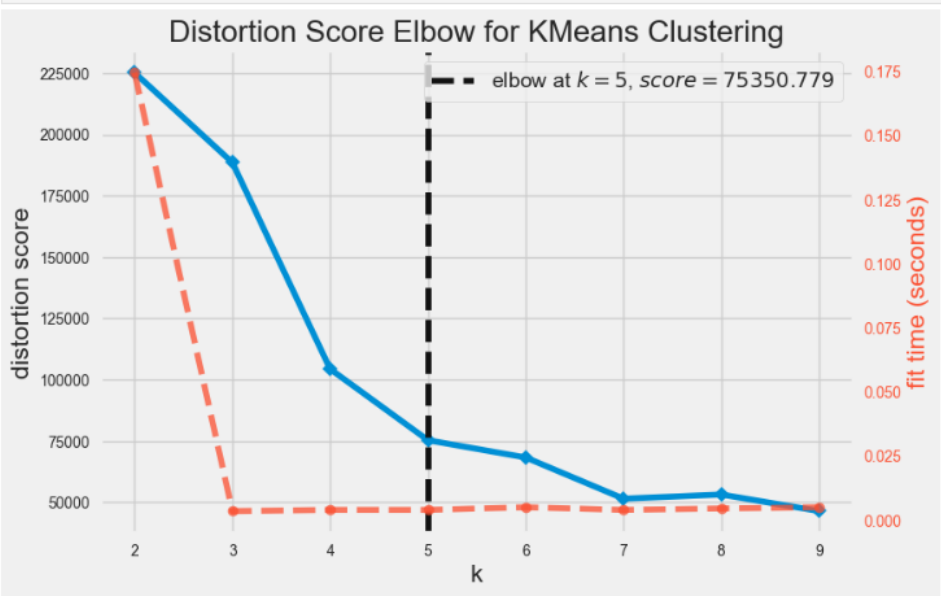
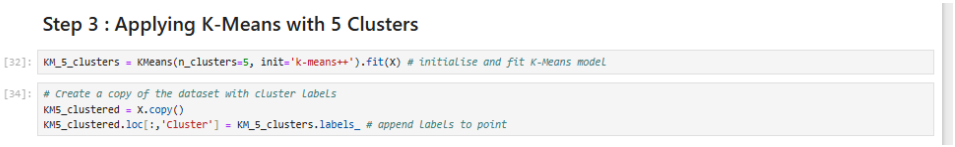


FIGURE II.21 – la méthode du coude

- Application du clustering K-Means pour segmenter les clients.



Step 3 : Applying K-Means with 5 Clusters

```
[32]: KM_5_clusters = KMeans(n_clusters=5, init='k-means++').fit(X) # initialise and fit K-Means model

[34]: # Create a copy of the dataset with cluster labels
KMS_clustered = X.copy()
KMS_clustered.loc[:,'Cluster'] = KM_5_clusters.labels_ # append labels to point
```

FIGURE II.22 – Application du clustering K-Means

- Visualisation des clusters obtenus.

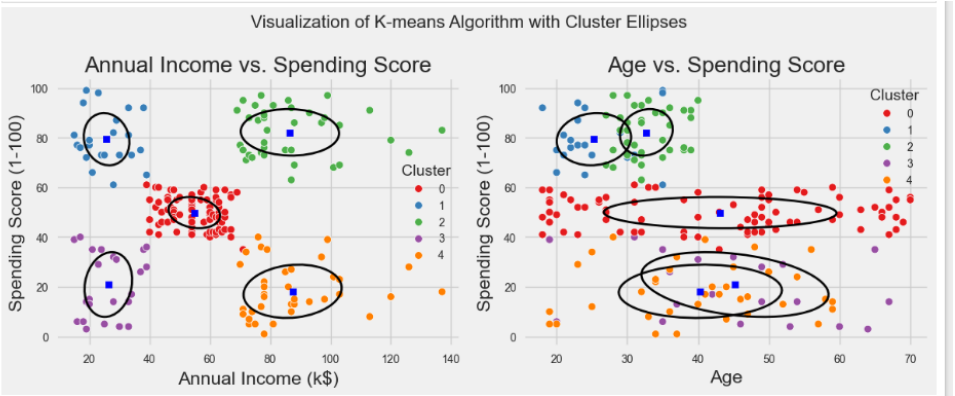
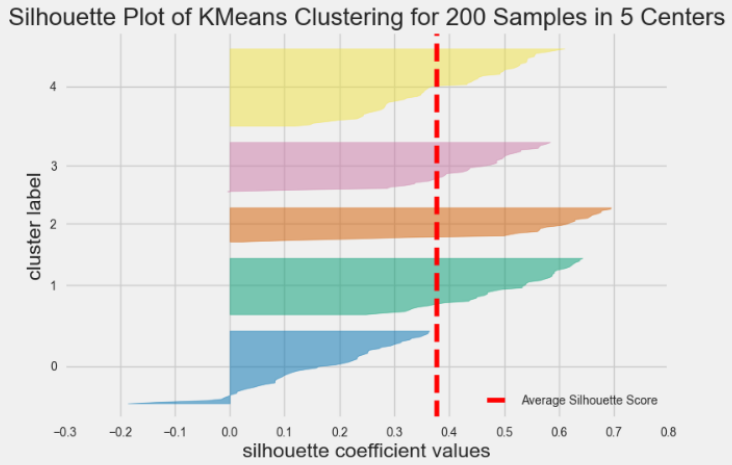


FIGURE II.23 – Visualisation des clusters obtenus

- Évaluation des clusters à l'aide du score de Silhouette.



Metric 1: Silhouette Score

```
# This measures how similar each point is to its own cluster (cohesion) compared to other clusters (separation).
# Values range from -1 (poor clustering) to 1 (perfect clustering).
silhouette_kmeans = silhouette_score(X, KM_5_clusters.labels_)
print(f"Silhouette Score for K-Means: {silhouette_kmeans}")
# A higher silhouette score indicates better-defined clusters.
```

```
Silhouette Score for K-Means: 0.44446409171786105
```

Metric 2: Davies-Bouldin Index

```
# This evaluates the ratio of within-cluster scatter to between-cluster separation.
# Lower values indicate better clustering.
davies_bouldin_kmeans = davies_bouldin_score(X, KM_5_clusters.labels_)
print(f"Davies-Bouldin Index for K-Means: {davies_bouldin_kmeans}")
# A lower Davies-Bouldin Index indicates better clustering quality.
```

```
Davies-Bouldin Index for K-Means: 0.8192608000040172
```

FIGURE II.24 – Évaluation des clusters

Cette phase permet de diviser les clients en groupes significatifs en fonction de leurs caractéristiques.

3.5 Application du Clustering avec DBSCAN

Dans cette étape, nous appliquons l'algorithme DBSCAN pour effectuer un autre type de clustering, basé sur la densité des points. Les sous-étapes sont les suivantes :

- Définition des hyperparamètres pour DBSCAN (comme `eps` et `min_samples`).

Step 1 : Define Hyperparameters for DBSCAN

```
from itertools import product

eps_values = np.arange(8,12.75,0.25) # eps values to be investigated
min_samples = np.arange(3,10) # min_samples values to be investigated

DBSCAN_params = list(product(eps_values, min_samples))
```

Define a grid of hyperparameter values (eps and min_samples) for DBSCAN. These combinations will be tested to find the optimal parameters.

FIGURE II.25 – Évaluation des clusters

- Exécution du clustering DBSCAN et évaluation avec le score de Silhouette.

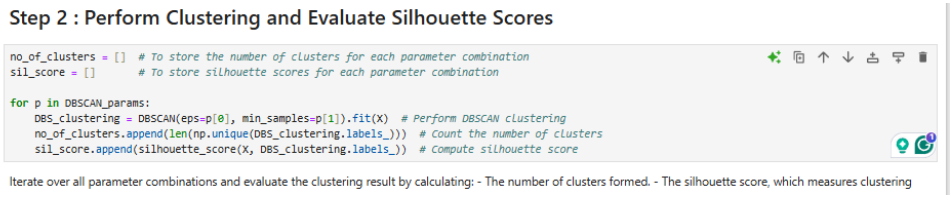


FIGURE II.26 – Évaluation des clusters

- Visualisation du nombre de clusters identifiés.
- Exécution du clustering DBSCAN et évaluation avec le score de Silhouette.

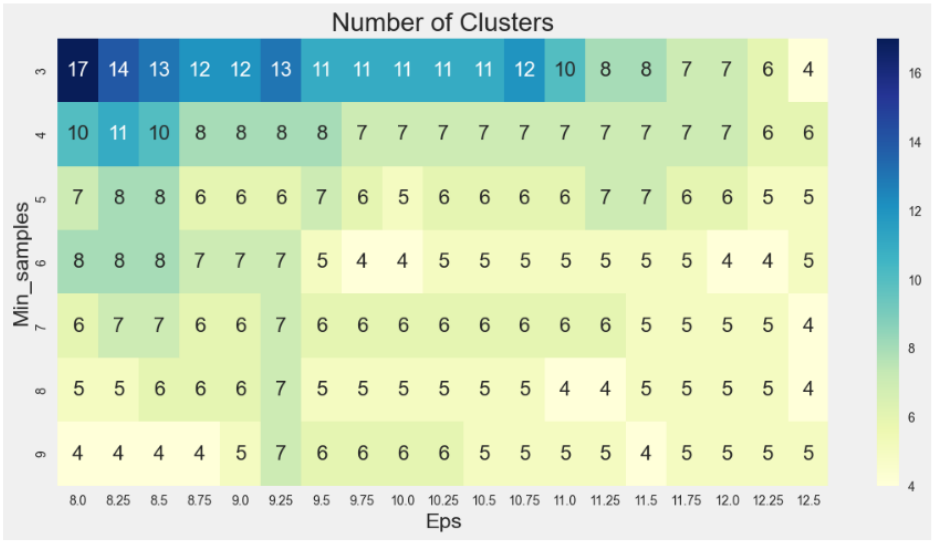


FIGURE II.27 – Évaluation des clusters

- Visualisation des scores de Silhouette pour évaluer la qualité du clustering.

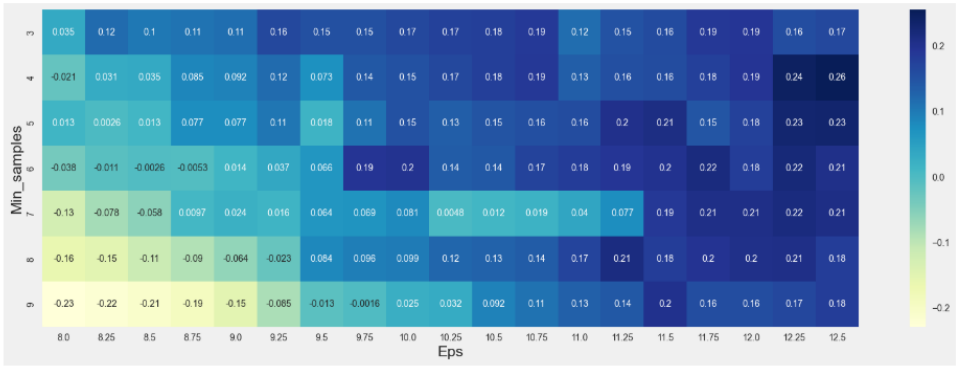


FIGURE II.28 – Évaluation des clusters

- Exécution du clustering DBSCAN avec les paramètres optimaux.

Step 5 : Perform DBSCAN Clustering with Optimal Parameters

```
[92]: DBS_clustering = DBSCAN(eps=12.25, min_samples=6).fit(X)  # Apply DBSCAN with selected parameters

      DBSCAN_clustered = X.copy()
      DBSCAN_clustered.loc[:, 'Cluster'] = DBS_clustering.labels_  # Append cluster labels to the dataset
      outliers = DBSCAN_clustered[DBSCAN_clustered['Cluster'] == -1]  # Identify outliers (label = -1)
```

Perform clustering with the chosen optimal parameters (eps=12.5 and min_samples=4) and identify outliers.

FIGURE II.29 – Évaluation des clusters

- Visualisation des clusters et des points considérés comme des "outliers" (bruit).

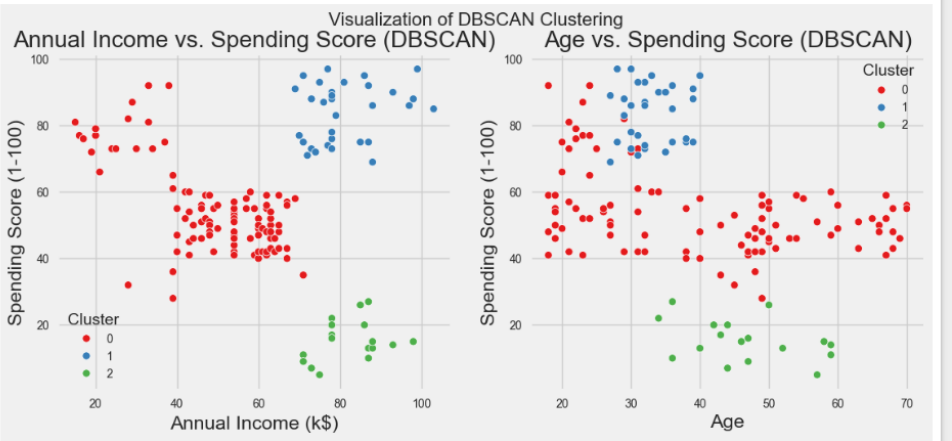


FIGURE II.30 – Évaluation des clusters

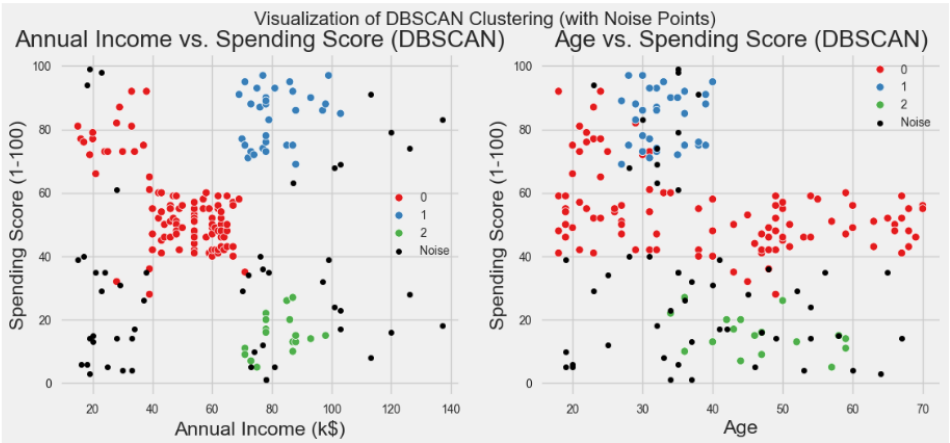


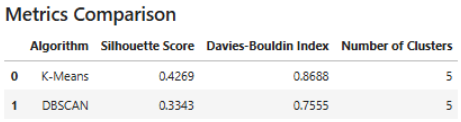
FIGURE II.31 – Évaluation des clusters

- Évaluation de l'algorithme DBSCAN.

FIGURE II.32 – Évaluation des clusters

3.6 Comparaison entre K-Means et DBSCAN

Après avoir appliqué les deux algorithmes de clustering, nous comparons les résultats obtenus par K-Means et DBSCAN. Cette comparaison inclut l'évaluation de la qualité des clusters à l'aide de diverses métriques, telles que le score de Silhouette, et l'examen des différences dans les types de clusters identifiés par chaque algorithme. Cette étape vise à déterminer quel algorithme offre la meilleure segmentation en fonction des données et des objectifs du projet.


Metrics Comparison

	Algorithm	Silhouette Score	Davies-Bouldin Index	Number of Clusters
0	K-Means	0.4269	0.8688	5
1	DBSCAN	0.3343	0.7555	5

FIGURE II.33 – Évaluation des clusters

- **Comparaison des métriques** : Les résultats montrent que K-Means a obtenu un score de silhouette de 0,4269 et un indice de Davies-Bouldin de 0,8688, avec un total de 5 clusters. En comparaison, DBSCAN a obtenu un score de silhouette inférieur, à 0,3343, et un indice de Davies-Bouldin de 0,7555, pour le même nombre de clusters. Ces différences indiquent que K-Means a mieux capturé la structure des données en termes de séparation et de cohésion des clusters.

- **Pourquoi K-Means a mieux fonctionné** : K-Means s'est avéré être l'algorithme le mieux adapté à ce jeu de données, car les clusters présents étaient sphériques, bien séparés et uniformément distribués. Ces caractéristiques correspondent aux forces de K-Means, qui excelle dans la création de clusters distincts lorsque les données ne présentent pas de bruit significatif. Par ailleurs, les dimensions "Revenu annuel vs Score de dépenses" et "Âge vs Score de dépenses" se sont prêtées naturellement à une segmentation précise, rendant K-Means particulièrement efficace.

- **Pourquoi DBSCAN a eu des difficultés** : DBSCAN, en revanche, n'a pas bien performé sur ce jeu de données en raison de la forme sphérique des clusters, qui

ne tire pas parti de sa capacité à gérer des formes de clusters irrégulières. De plus, l'ajustement du paramètre eps s'est révélé complexe, rendant difficile la capture précise de tous les clusters sans fusionner ou diviser excessivement les groupes. Bien que DBSCAN ait identifié 18 points comme bruit, ces derniers n'ont pas eu un impact significatif sur la segmentation globale, mais cela souligne sa capacité à détecter les anomalies.

- **Caractéristiques du jeu de données favorisant chaque algorithme** : Le jeu de données analysé contenait des clusters bien séparés, propres et sans bruit significatif, ce qui a favorisé l'utilisation de K-Means. Cet algorithme fonctionne de manière optimale lorsque les données sont modérément volumineuses et que tous les points appartiennent à des groupes bien définis. En revanche, DBSCAN est plus adapté à des jeux de données présentant des formes de clusters irrégulières, des densités variables ou une proportion importante de points bruités, des caractéristiques absentes dans ce cas.

4 Conclusion

En conclusion, l'implémentation des algorithmes de clustering K-Means et DBSCAN a permis d'explorer différentes approches pour segmenter les clients en groupes distincts. Chaque méthode a ses forces et ses limites, et leur performance dépend fortement des caractéristiques du jeu de données. Tandis que K-Means a démontré son efficacité grâce à des clusters bien séparés et homogènes, DBSCAN reste une alternative puissante pour traiter des données plus complexes ou bruitées. Cette analyse souligne l'importance de sélectionner l'algorithme adapté en fonction du contexte et des objectifs spécifiques de l'étude.

Conclusion Générale

En conclusion, ce rapport a permis d'explorer et d'implémenter des techniques avancées de clustering, notamment les algorithmes K-Means et DBSCAN, pour la segmentation des clients dans un contexte de data mining. Après une introduction théorique approfondie, suivie d'une application pratique sur le dataset *Mall Customers*, les résultats obtenus ont mis en évidence l'importance du choix de l'algorithme en fonction des caractéristiques des données.

K-Means a démontré sa robustesse sur des données bien structurées et homogènes, tandis que DBSCAN a offert des perspectives intéressantes pour des scénarios plus complexes avec des formes de clusters irrégulières ou du bruit. Cette étude illustre également les défis associés à la préparation des données, à l'analyse exploratoire et à l'évaluation des résultats, qui sont des étapes cruciales dans tout projet de data mining.

Ce travail met en lumière l'importance des approches basées sur les données pour répondre à des problématiques concrètes, comme la segmentation des clients, et ouvre des perspectives pour l'application de ces techniques dans d'autres domaines, renforçant ainsi leur rôle central dans l'analyse et la prise de décision stratégique.

Bibliographie

- KMeans Clustering Algorithm Documentation : sklearn.cluster.KMeans.html | Consulté le : 20/11/2024
- DBSCAN Clustering Algorithm Documentation : sklearn.cluster.DBSCAN.html | Consulté le : 20/11/2024
- Principal Component Analysis (PCA) Documentation : sklearn.decomposition.PCA.html | Consulté le : 20/11/2024
- Pandas Documentation : https://pandas.pydata.org/pandas-docs/stable/ | Consulté le : 20/11/2024
- Seaborn Documentation : https://seaborn.pydata.org/ | Consulté le : 20/11/2024
- Matplotlib Documentation : https://matplotlib.org/stable/contents.html | Consulté le : 20/11/2024
- "Data Science for Business" by Foster Provost and Tom Fawcett, O'Reilly Media, 2013.
- "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili, Packt Publishing, 2017.
- "Hands-On Unsupervised Learning with R" by Jason Brownlee, Machine Learning Mastery, 2019.
- Davies, D. L., Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2), 224-227.
- Ester, M., Kriegel, H.-P., Sander, J., Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 226-231.
- Mme. MOUNIR Ilham. (2024). "Outils d'aide à la décision : Fouille de Données" | Consulté le 27/12/2024.
- Mme. MOUNIR Ilham. (2024). "Clustering (regroupement, segmentation)" | Consulté le 29/12/2024.
- "Density-Based Spatial Clustering of Applications with Noise (DBSCAN) : An Advanced Guide" by A. S. Tamang. *Data Science Quarterly*, 2020.