UNIVERSITAIRE CADDI AYYAD
Ecole Supérieure de Technologie -SAFI-
LP Ingénierie des systèmes d'information et réseaux informatiques

Rapport de Datamining

dbscan algorithm

Encadré par :

Mme I. Mounir

Réalisé par :

BOUAMIR Assia

SABKARI Abderrahmane

Juin 2024

Table des matières

1 résumé

Le rapport explore les algorithmes de regroupement (clustering) avec un accent particulier sur l'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise). La nécessité de DBSCAN dans divers contextes est soulignée, mettant en lumière sa capacité à identifier des clusters de densité variable, tout en distinguant les points de bruit.

Le fonctionnement de DBSCAN est détaillé, couvrant les concepts fondamentaux tels que les points centraux, les points frontières et les points de bruit. La méthode de regroupement est expliquée en mettant l'accent sur la formation des groupes en fonction de la densité, avec des critères spécifiques de formation de clusters.

Les avantages de DBSCAN sont exposés, mettant en évidence sa robustesse face à des formes de clusters complexes et sa capacité à gérer les outliers. Cependant, des limitations sont également discutées, indiquant les situations où DBSCAN peut ne pas être approprié.

Une application pratique de DBSCAN sur un exemple de table est présentée, avec une analyse détaillée des clusters identifiés et des points de bruit. Ensuite, une implémentation en Python sur une dataset est abordée, couvrant le choix d'une dataset pertinente, le prétraitement des données, et la mise en œuvre de DBSCAN à l'aide de bibliothèques telles que scikit-learn. La visualisation des résultats à l'aide de graphiques est également démontrée.

L'évaluation des performances est explorée, incluant les métriques d'évaluation le cas échéant, avec une discussion approfondie sur la qualité des clusters obtenus.

En conclusion, le rapport récapitule les points clés, souligne l'importance de DBSCAN dans le contexte du clustering, et suggère des possibilités d'amélioration ou d'extension pour de futures recherches. Ce rapport offre ainsi une compréhension approfondie de l'algorithme DBSCAN et de son application pratique dans l'analyse de données.

2 Introduction générale

Dans le contexte du module sur l'Intelligence Artificielle (AI), ce projet s'inscrit dans une démarche d'exploration et d'application de l'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Les algorithmes de clustering jouent un rôle essentiel dans l'analyse de données en identifiant des structures intrinsèques, facilitant ainsi la prise de décision dans des domaines divers. DBSCAN, en particulier, se distingue par sa capacité à détecter des clusters de densité variable tout en gérant efficacement les points isolés. Ce projet offrira une opportunité pratique d'approfondir la compréhension de cet algorithme, avec une implémentation concrète en Python et une évaluation des performances, renforçant ainsi les compétences des participants en matière d'IA appliquée.

3 l'historique de DBSCAN :

L'algorithme DBSCAN, présenté pour la première fois en 1996 par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaowei Xu dans leur article intitulé "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", a connu une popularité croissante depuis sa publication. Il est largement utilisé en raison de sa capacité à détecter des clusters de tailles et de formes variables dans des données bruyantes ou spatio-temporelles.

Comparé à l'algorithme de classification hiérarchique, un autre algorithme de clustering populaire, le DBSCAN présente des avantages significatifs. Contrairement à la classification hiérarchique, DBSCAN n'exige pas la spécification préalable du nombre de clusters, ce qui le rend plus flexible et adaptatif aux données réelles. De plus, DBSCAN est souvent plus rapide à exécuter pour des ensembles de données volumineux.

Cependant, il est important de noter que l'algorithme de classification hiérarchique offre une représentation visuelle sous forme de dendrogramme, montrant les relations de similarité ou de regroupement entre les éléments d'un ensemble de données. Bien que DBSCAN ne génère pas une telle représentation directe, ses résultats peuvent être évalués et interprétés à travers d'autres métriques de performance de clustering.

En somme, bien que DBSCAN et la classification hiérarchique soient tous deux des choix valables pour le clustering, le choix entre les deux dépend des spécificités des données et des préférences en termes de facilité d'utilisation et de visualisation.

4 Définition

L'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise) est une méthode de clustering utilisée pour regrouper des points de données en clusters en se basant sur la densité des points dans l'espace. Il excelle dans la détection de clusters de formes et de tailles variables dans des données bruyantes, en identifiant des zones denses de points séparées par des zones moins denses. Sa capacité à s'adapter naturellement à la densité locale de l'espace des données en fait un outil puissant, offrant une flexibilité particulière pour l'analyse de données réelles et variées, tout en résistant au bruit et en éliminant la nécessité de spécifier le nombre de clusters à l'avance. .

4.1 Contexte général sur les algorithmes de regroupement (clustering)

Dans le vaste domaine de l'analyse de données, les algorithmes de regroupement, également connus sous le nom d'algorithmes de clustering, occupent une place cruciale en permettant la détection de structures inhérentes au sein de ensembles de données. Leur objectif principal est de diviser un ensemble de données en groupes homogènes, où les éléments au sein d'un groupe sont similaires les uns aux autres tandis que les groupes eux-mêmes sont distincts les uns des autres. Cette approche de classification automatique offre une compréhension approfondie des relations intrinsèques entre les données, facilitant ainsi la prise de décision, la segmentation de marché, la détection d'anomalies, et d'autres applications clés dans des domaines variés tels que la biologie, la finance et l'exploration de données. Dans ce contexte, les algorithmes de clustering constituent un outil fondamental pour extraire des informations significatives à partir de grandes quantités de données, contribuant ainsi de manière significative à l'avancement de l'analyse de données et de l'intelligence artificielle.

4.2 Motivation pour le regroupement basé sur la densité

Deux types courants de méthodes de clustering sont : le partitionnement et les méthodes hiérarchiques.

La méthode de partitionnement partitionne l'ensemble de données en k (l'entrée principale des méthodes) nombre de groupes (clusters). Le processus itératif de partition alloue chaque point ou objet (à partir de maintenant, je l'appellerai point) de l'ensemble de données au groupe auquel il appartient. Une fois les points attribués dans son groupe, la moyenne du

groupe (son centroïde) est calculée en prenant une moyenne pour tous les points du groupe. La méthode de partitionnement la plus connue est K-means . Les méthodes de partition présentent des inconvénients importants : vous devez savoir à l'avance en combien de groupes vous souhaitez diviser la base de données (la valeur K). Un autre inconvénient important est que K-means ne fonctionne pas bien pour trouver des formes non convexes / non sphériques de grappes (figure 1). De plus, K-means est sensible aux données de bruit.

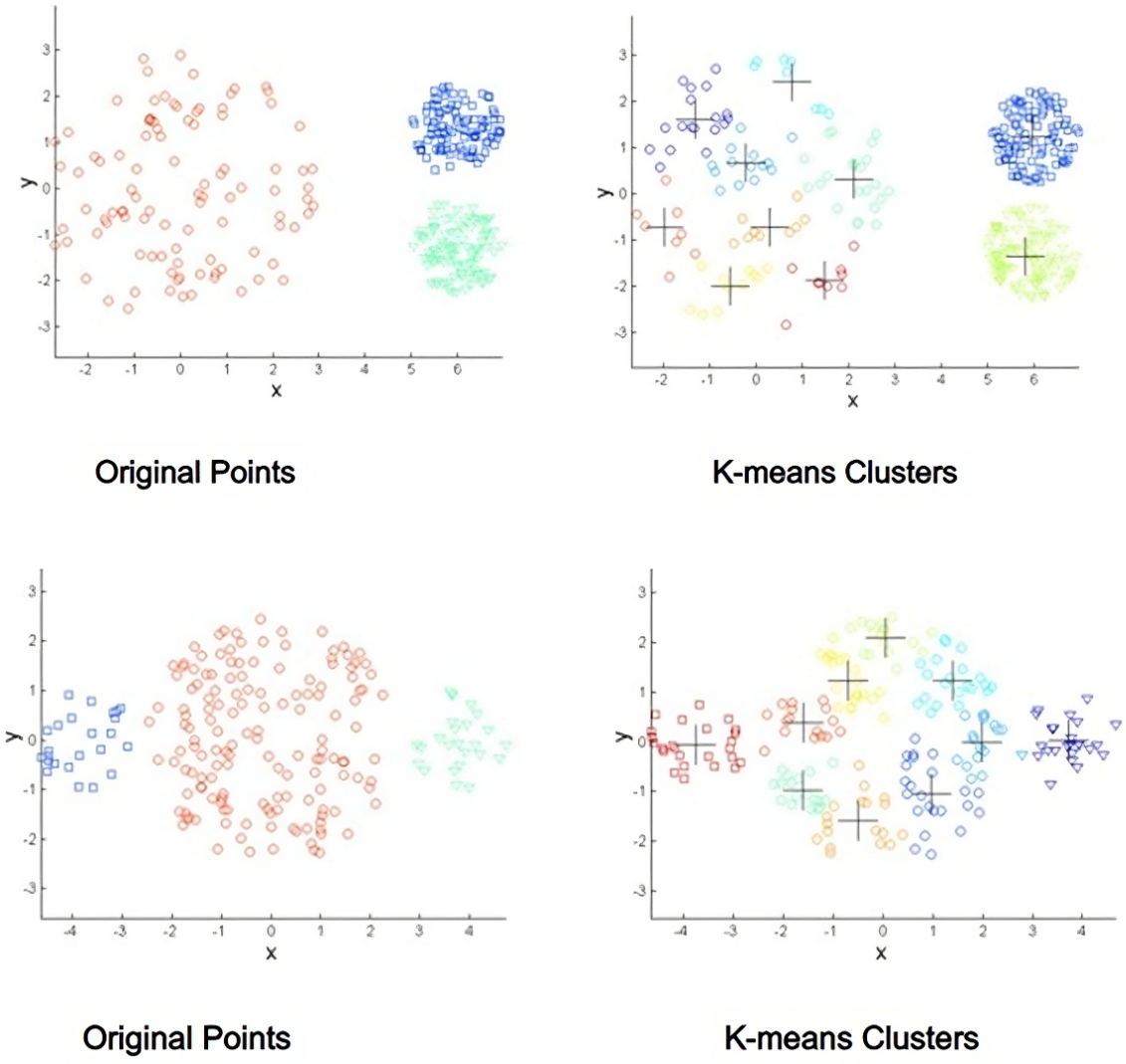


FIGURE 1 – Enter Caption

Une méthode hiérarchique crée une représentation visuelle hiérarchique des données à l'aide d'un arbre spécial (dendrogramme). Une méthode de clustering hiérarchique agglomé-

rative démarre lorsque chaque point de son cluster et à chaque étape, il fusionne des clusters similaires ou proches en un cluster. Dans la dernière étape, tous les points sont dans le même cluster. Contrairement à K-means, vous n'avez pas besoin de décider quel doit être le nombre de clusters, mais cela présente également de sérieux inconvénients : il ne convient pas aux grands ensembles de données, a une complexité de calcul élevée, vous devez choisir la métrique pour fusionner les clusters (liaison) qui affecte le résultat du clustering. De plus, chaque métrique a ses inconvénients : sensibilité au bruit, difficulté à gérer les différences importantes dans la densité des clusters, et sensibilité à la forme et à la taille des clusters.

Comme nous pouvons le voir, les principaux inconvénients du partitionnement et des méthodes hiérarchiques sont : la gestion du bruit et l'obtention de mauvais résultats en trouvant des grappes de forme non sphérique.

La méthode de clustering DBSCAN est capable de représenter des clusters de forme arbitraire et de gérer le bruit.

Clusters of arbitrary shape.

FIGURE 2 – Enter Caption

5 Fonctionnement de l'algorithme DBSCAN

5.1 Principes fondamentaux

Les techniques de clustering hiérarchiques et basées sur les partitions sont très efficaces avec les clusters de forme normale. Cependant, lorsqu'il s'agit de grappes de forme arbitraire ou de détection de valeurs aberrantes, les techniques basées sur la densité sont plus efficaces.

Par exemple, l'ensemble de données de la figure ci-dessous peut facilement être divisé en trois groupes à l'aide de l'algorithme k-means.

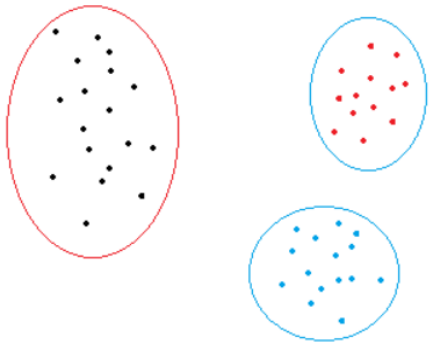
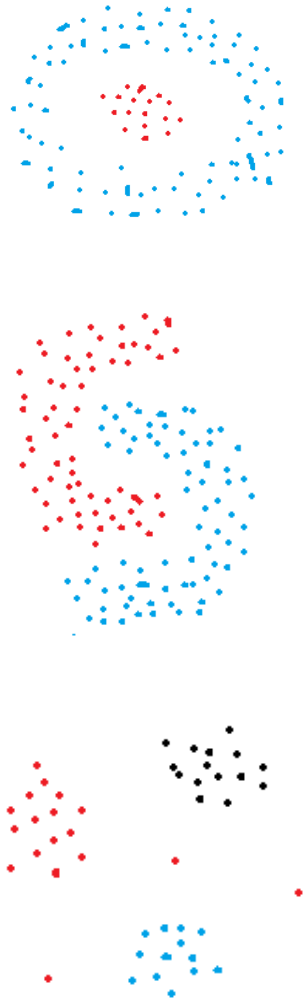


FIGURE 3 – k-signifie clustering

Considérez les chiffres suivants :



Les points de données dans ces figures sont regroupés sous des formes arbitraires ou incluent des valeurs aberrantes. Les algorithmes de clustering basés sur la densité sont très efficaces pour trouver des régions à haute densité et des valeurs aberrantes. Il est très important de détecter les valeurs aberrantes pour certaines tâches, par exemple la détection d'anomalies.

5.1.1 Paramètres requis pour l'algorithme DBSCAN

eps :

Il définit le voisinage autour d'un point de données, c'est-à-dire que si la distance entre deux points est inférieure ou égale à 'eps', ils sont considérés comme voisins. Si la valeur eps est choisie trop petite, une grande partie des données sera considérée comme une valeur aberrante. S'il est choisi très grand, les clusters fusionneront et la majorité des points de

données se trouveront dans les mêmes clusters. Une façon de trouver la valeur eps est basée sur le graphique k-distance.

MinPts :

nombre minimum de voisins (points de données) dans le rayon eps. Plus le jeu de données est grand, plus la valeur MinPts doit être élevée. En règle générale, les MinPts minimaux peuvent être dérivés du nombre de dimensions D dans le jeu de données comme, MinPts >= D+1. La valeur minimale de MinPts doit être choisie au moins 3.

5.1.2 Points Centraux (Core Points) :

Les points centraux sont des éléments de données qui ont un nombre minimum de points dans leur voisinage défini par un rayon spécifique (epsilon). Ces points sont considérés comme le cœur des clusters.

5.1.3 Points Frontières (Border Points) :

Les points frontières sont des éléments de données qui ne sont pas des points centraux eux-mêmes, mais qui se trouvent dans le voisinage d'un point central. Ils contribuent à l'extension du cluster sans être le cœur du groupe.

5.1.4 Points de Bruit (Noise Points) :

Les points de bruit sont des éléments de données qui ne sont pas des points centraux et n'appartiennent à aucun cluster. Ils sont souvent isolés et ne répondent pas aux critères de densité pour être inclus dans un cluster.

5.1.5 Voisinage d'un Point :

Le voisinage d'un point est défini par un rayon (epsilon) autour du point. Les points situés dans ce voisinage sont considérés comme ses voisins.

L'algorithme DBSCAN utilise ces concepts pour former des clusters en se basant sur la densité des points. Il commence par choisir un point arbitraire et explore son voisinage. Si ce point a suffisamment de voisins pour être un point central, un nouveau cluster est formé en incluant ce point central et tous ses voisins. Ce processus se répète jusqu'à ce que tous les points du dataset aient été examinés et attribués à des clusters ou identifiés comme points de bruit. Cette approche basée sur la densité confère à DBSCAN la capacité de détecter des clusters de formes et de tailles variées dans des données complexes.

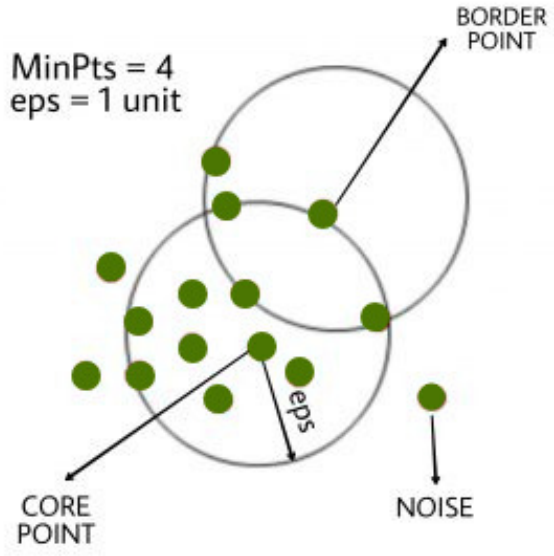


FIGURE 4 – Enter Caption

Dans ce cas, minPts vaut 4. Les points rouges sont des points centraux car il y a au moins 4 points dans leur zone environnante avec un rayon eps. Cette zone est représentée par les cercles sur la figure. Les points jaunes sont des points frontières car ils sont accessibles à partir d'un point central et ont moins de 4 points dans leur voisinage. Accessible signifie être dans la zone environnante d'un point central. Les points B et C ont deux points (y compris le point lui-même) dans leur voisinage (c'est-à-dire la zone environnante avec un rayon de eps). Enfin N est une valeur aberrante car ce n'est pas un point central et ne peut pas être atteint à partir d'un point central.

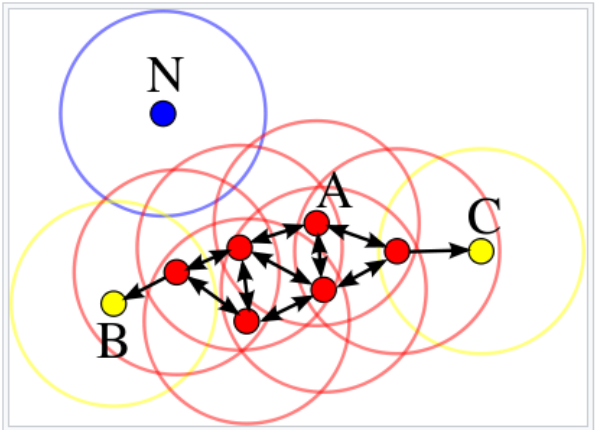


FIGURE 5 – Enter Caption

6 Les étapes de l'algorithme DBSCAN :

Nous avons appris les définitions des paramètres et des différents points de type. Maintenant, nous pouvons parler du fonctionnement de l'algorithme. C'est en fait assez simple :

1. minPts et eps sont déterminés.
2. Un point de départ est sélectionné au hasard à sa zone de voisinage est déterminé en utilisant le rayon eps. S'il y a au moins minPts nombre de points dans le voisinage, le point est marqué comme point central et une formation d'amas commence. Sinon, le point est marqué comme du bruit. Une fois qu'une formation de cluster commence (disons le cluster A), tous les points dans le voisinage du point initial deviennent une partie du cluster A. Si ces nouveaux points sont également des points centraux, les points qui se trouvent à proximité sont également ajoutés à groupe A.
3. L'étape suivante consiste à choisir au hasard un autre point parmi les points qui n'ont pas été visités lors des étapes précédentes. Ensuite, la même procédure s'applique.
4. Ce processus est terminé lorsque tous les points sont visités.

En appliquant ces étapes, l'algorithme DBSCAN est capable de trouver des régions à haute densité et de les séparer des régions à faible densité.

Un cluster comprend des points centraux voisins (c'est-à-dire accessibles les uns des autres) et tous les points frontières de ces points centraux. La condition requise pour former un cluster est d'avoir au moins un point central. Bien que très improbable, nous pouvons avoir un cluster avec un seul point central et ses points de frontière.

7 Implantation de DBSCAN

Dans cette partie, nous allons découvrir comment implémenter et utiliser l'algorithme DBSCAN en utilisant le langage de programmation Python. Cette section se compose de deux parties principales. Dans la première, nous allons découvrir comment implémenter le DBSCAN à partir de zéro, tandis que dans la deuxième, nous allons utiliser la fonction prédéfinie dans Python DBSCAN de la bibliothèque `scikit-learn`.

7.1 DBSCAN à partir de zéro

```python
import numpy

def dbscan(D, eps, MinPts):
    # DBSCAN prend en paramètre un ensemble de données (une liste de vecteurs),
    # une distance seuil 'eps' et un nombre minimum de points 'MinPts'

    # Il retourne une liste de valeurs où 0 indique que le point n'a pas encore été traité,
    # -1 indique que le point est considéré comme du bruit ('noise'),
    # et toute autre valeur indique le numéro du cluster auquel ce point appartient
    labels = [0]*len(D)

    # "C" est l'identifiant du cluster actuel.
    C = 0

    for P in range(0, len(D)):
        # Si le point a déjà été traité, nous passons au point suivant
        if not (labels[P] == 0):
            continue

        # Trouver tous les voisins du point 'P'.
        NeighborPts = region_query(D, P, eps)
        if len(NeighborPts) < MinPts:
            # Si le nombre de voisins est inférieur au minimum requis,
            # le point sera considéré comme du 'bruit' (noise)
            labels[P] = -1
        else:
            # Sinon, nous créons un nouveau cluster
            C += 1
            grow_cluster(D, labels, P, NeighborPts, C, eps, MinPts)
    return labels
```

FIGURE 6 – dbscan - 1

```python
def region_query(D, P, eps):
    # Trouver tous les points dans l'ensemble de données D à une distance eps du point P.
    neighbors = []
    for Pn in range(0, len(D)):
        # Si la distance est en dessous du "eps", ajoutez-la à la liste des voisins
        if numpy.linalg.norm(D[P] - D[Pn]) < eps:
            neighbors.append(Pn)
    return neighbors

def grow_cluster(D, labels, P, NeighborPts, C, eps, MinPts):
    # Faire croître un nouveau cluster avec l'étiquette C à partir du point initial P.
    labels[P] = C
    i = 0
    while i < len(NeighborPts):
        Pn = NeighborPts[i]
        # Si le point est marqué comme bruit, nous allons changer son étiquette.
        if labels[Pn] == -1:
            labels[Pn] = C

        # Si le point est marqué comme n'ayant pas encore été traité,
        # nous allons changer son étiquette
        elif labels[Pn] == 0:
            labels[Pn] = C
            # trouver les voisins de point "Pn"
            PnNeighborPts = region_query(D, Pn, eps)
            if len(PnNeighborPts) >= MinPts:
                # ajouter les voisins de "Pn" au voisins "P"
                NeighborPts = NeighborPts + PnNeighborPts
        i += 1
```

FIGURE 7 – dbscan - 2

7.2 DBSCAN à partir de la bibliothèque scikit-learn

Dans cette partie, nous allons utiliser la fonction DBSCAN de la bibliothèque scikit-learn pour effectuer une implémentation avec des données Extrait de Kaggle.

```
df = pd.read_csv('Mall_Customers.csv')
X_train = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]

clustering = DBSCAN(eps=12.5, min_samples=4).fit(X_train)
DBSCAN_dataset = X_train.copy()
DBSCAN_dataset['Cluster'] = clustering.labels_

outliers = DBSCAN_dataset[DBSCAN_dataset['Cluster'] == -1]

fig2, axes = plt.subplots(1, 2, figsize=(12, 5))

sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
                data=DBSCAN_dataset[DBSCAN_dataset['Cluster'] != -1],
                hue='Cluster', ax=axes[0], palette='Set2', legend='full', s=200)

sns.scatterplot(x='Age', y='Spending Score (1-100)',
                data=DBSCAN_dataset[DBSCAN_dataset['Cluster'] != -1],
                hue='Cluster', palette='Set2', ax=axes[1], legend='full', s=200)

axes[0].scatter(outliers['Annual Income (k$)'], outliers['Spending Score (1-100)'], s=10, label='outliers', c="k")
axes[1].scatter(outliers['Age'], outliers['Spending Score (1-100)'], s=10, label='outliers', c="k")

axes[0].legend()
axes[1].legend()

plt.setp(axes[0].get_legend().get_texts(), fontsize='12')
plt.setp(axes[1].get_legend().get_texts(), fontsize='12')

plt.show()
```

FIGURE 8 – dbscan - 3

7.2.1 resultat d'execution :

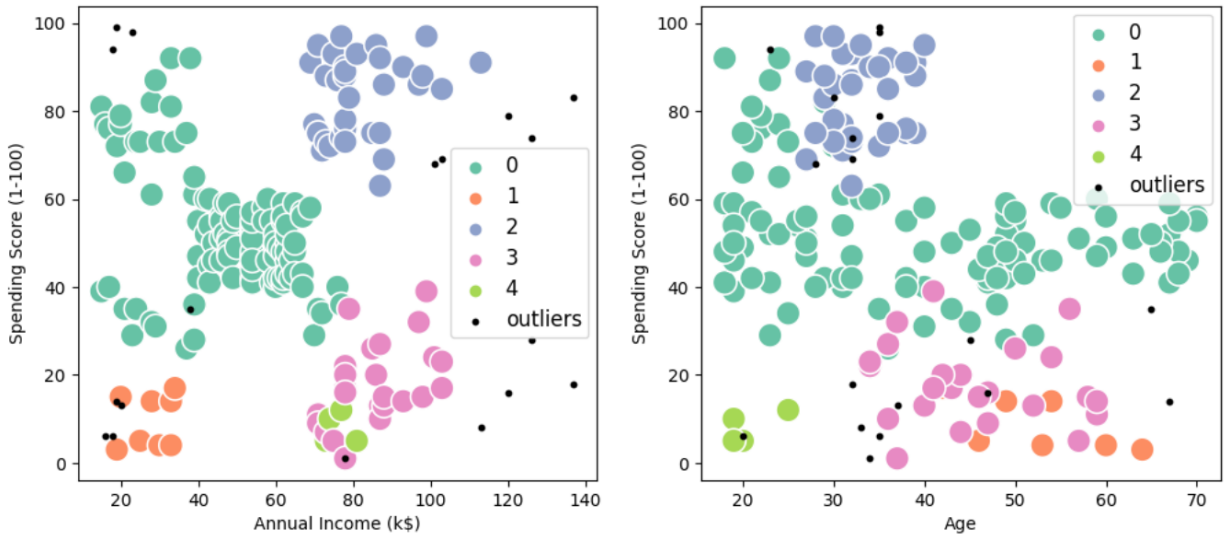


FIGURE 9 – dbscan-4

8 Conclusion

Pour conclure, ce projet représente une excellente opportunité de découvrir le monde du machine learning. Nous travaillons sur l'algorithme DBSCAN et nous le mettons en œuvre en utilisant le langage Python. Cela nous permet d'observer comment cet algorithme traite et regroupe automatiquement les données, ce qui peut grandement faciliter la visualisation et le regroupement des données en vue d'analyses approfondies.