Étude Marketing : Analyse de la clientèle d'un grand magasin et définition de profils clients.

Par Benjamin FOURREAU & François VEYRET

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

INTRODUCTION

A travers ce projet nous avons pour objectif du projet est d'analyser la clientèle d'un grand magasin et de créer des pro- fils clients en effectuant un clustering des clients à partir de données collectées via des questionnaires. Grâce à ces profils, cela permettra à l'entreprise de mieux comprendre les habitudes d'achat de ses clients et d'optimiser ses efforts commerciaux en proposant des produits et des services ciblés en cohérence avec les attentes des différents types de clients.

Nous allons rappeler des méthodes et découvrir de nouvelles méthodes que nous n'avons pas vu en cours, comme les méthodes DBSCAN et Spectral clustering.

Dans un premier temps, nous analyserons les partitions obtenues par les différentes méthodes à l'aide de différentes métriques. Dans un second temps nous proposerons une liste de profils clients et rédigerons une carte d'identité pour chacun des profils types identifiés. Enfin nous conclurons par dire en quoi ce projet nous a permis d'avoir une meilleure compréhension des algorithmes et méthodes et comment ces algorithmes nous serons utiles dans notre carrière en tant qu'ingénieur

.

SOMMAIRE

1) Étude préalable : Comparaison des méthodes de clustering sur des données simulées

1.1 Étude des méthodes

Dans cette partie nous allons expliquer le fonctionnement de méthodes vu en cours : méthode des K-MEANS, CAH, Mélange des gaussiens. Ainsi que 2 méthodes que nous n'avons pas étudier en cours comme la méthode DBSCAN et la méthode Spectral Clustering

Dans cette partie nous allons découvrir deux nouvelles méthodes que nous n'avons pas étudier en cours. Nous allons donc dans un premier temps, faire des recherches sur cette dernière et dans un second temps expliquer comment elles marchent.

1ère méthode : Présentation d'une nouvelle méthode : **DBSCAN**

L'algorithme DBSCAN va étudier les différents niveaux de densité du nuage de point pour en déterminer les clusters.

L'algorithme va prendre deux paramètres :

- Une distance r qui considère que deux points sont voisins.
- Un nombre de point n minimum pour définir un cluster.

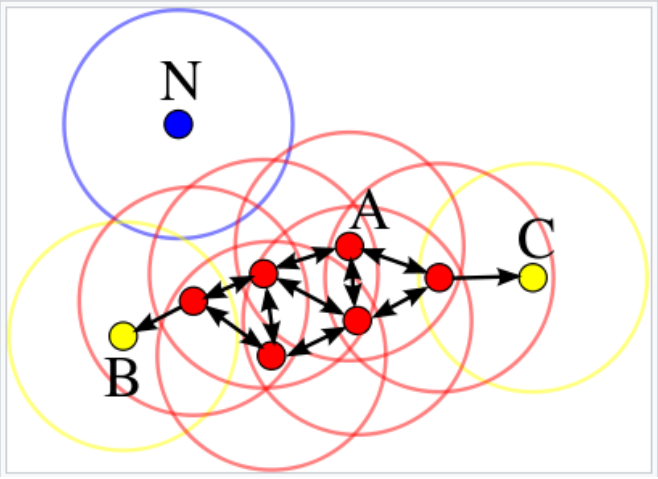


Figure 1 Bscan représentation

On va considérer trois types de points durant l'analyse, des *Core points*, *Border points* et *Outlier points*. Un cluster est formé de Core points et Border points. Un core points doit avoir dans son rayon r au moins n points pour être considéré comme tel et commencer un cluster.

Les Border points pour appartenir dans le cluster doit se trouver dans le rayon r d'un des core points. Les Outlier sont les points en dehors des rayons des Core points, ils ne font donc pas partie du cluster. Lorsqu'on étudie un point soit on le marque comme Core points soit comme un point bruit. Les points bruits sont alors vus comme Outlier ou border lors de l'attribution des points au cluster formé par les Core points.

En rouge les Core points, en jaune les Border points et en bleu les Outlier.

L'algorithme de la méthode DBSCAN se déroule de telle sorte:

- **Initialisation** : pour débuter l'algorithme on déterminer arbitrairement la distance r et le nombre n.
- **Faire** :
  - sélection d'un point aléatoire
  - analyse des points voisins selon son rayon r
  - marqué comme core points si il possède n points sous son rayon r et ajouté au cluster de ces points voisins sinon il est marqué comme un point bruit
- **Fin** : tous les points ont été visités, rassemblement des points voisins aux core points sous cluster

L'avantage de cet algorithme est qu'il est plus efficace pour partitionner des nuages de points de formes spécifiques. Ils forment un cluster un suivant la densité d'un groupe de points.

2ème méthode : Présentation d'une nouvelle méthode **: Spectral Clustering**

Cette méthode représente un graphe de nœuds et d'arêtes (les nœuds sont reliés entre eux) construit par méthode du voisinage ou des plus proches voisins.

Un graph est représenté par plusieurs matrices :

- **Matrice adjacente** : représentant les connections des nœuds (matrice de 1 ou 0 (si pas de poids), 1 pour un couple (i ,j) connectés , diagonal nul)
- **Degree Matrix :** représente le nombre de connections pour chaque nœud (matrice diagonale)

Ensuite, on calcule ce qu'on va appeler la *matrice Laplacienne* en soustrayant la matrice de dégrées par la matrice adjacente. On récupère alors les toutes les informations dans cette matrice.

Pour réaliser les clustering on va analyser les valeurs propres de la Laplacienne.

D'après le cours, on sait que si Pour une matrice A il existe n valeurs propres λ si on a : $\mathbf{Ax = \lambda x}$, avec x les vecteurs propres associés aux valeurs propres

Ainsi dans un jeu de données à n individus (nœuds), il y aura n valeurs propres.

On peut tracer un graphique des valeurs propres.

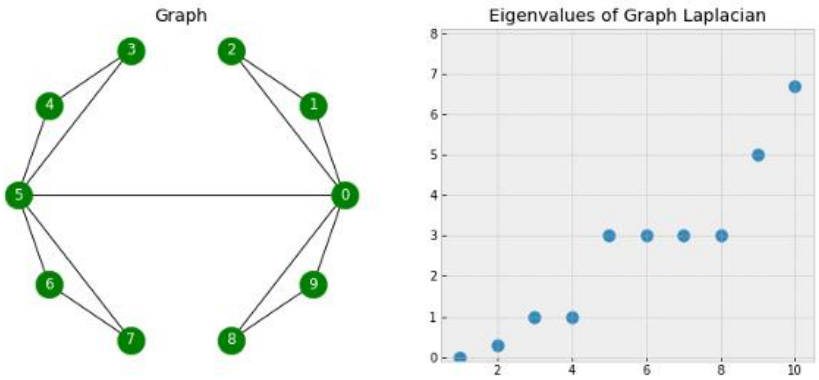


Figure 2 Spectral clustering analyse

On lit alors les informations suivantes :

- la première valeur propre indique le nombre de sous-ensembles connectés (égale à 0 si tous les nœuds sont connectés)

-la première valeur propre non-nulle correspond au gap spectral, nous renseigne sur la densité du graphe.

- la deuxième valeur propre se nomme le fielder value est correspond au nombre minimal de séparation entre nœuds pour avoir deux sous-ensembles. (0 si il y a déjà deux sous-ensembles) Le vecteur associé nous renseigne sur la manière de couper le graphe.

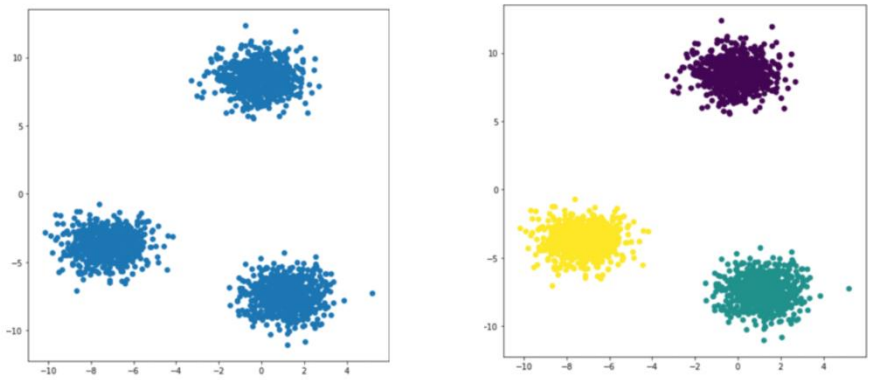
Ensuite plus généralement, les valeurs propres proches de 0 indiquent qu'ils sont proches d'une séparation entre deux sous-ensembles. Ainsi s'il y a k valeurs propres proches de 0 alors on peut s'imaginer avoir k cluster. On peut simplement observer le premier grand écart de valeur entre la valeur propre et le nombre de valeurs propres avant ce gap correspond au nombre de potentiel sous-ensembles ou cluster. Ainsi le vecteur des k-1èmes valeurs propres nous indique où faire les séparations.

Cet algorithme est efficace pour des nuages en formes convexes.

3ème méthode : méthode des **K-MEANS**

Comme son nom l'indique la méthode K-MEANS a pour but d'utiliser la moyenne. C'est un algorithme utilisé dans le clustering et dans le machine learning.

Étant donnés des points et un entier k, l'algorithme vise à diviser les points en k groupes, appelés clusters, homogènes et compacts. Regardons l'exemple ci-dessous :



L'idée de cet algorithme est assez simple. Dans un premier temps, on définit les 3 centroïdes aléatoirement auxquels on associe 3 étiquettes par exemple 0,1,2. Dans un second temps, nous allons pour chaque point regarder leur distance aux 3 centroïdes et nous associons le point au centroïde le plus proche et l'étiquette correspondante. Cela revient à étiqueter nos données.

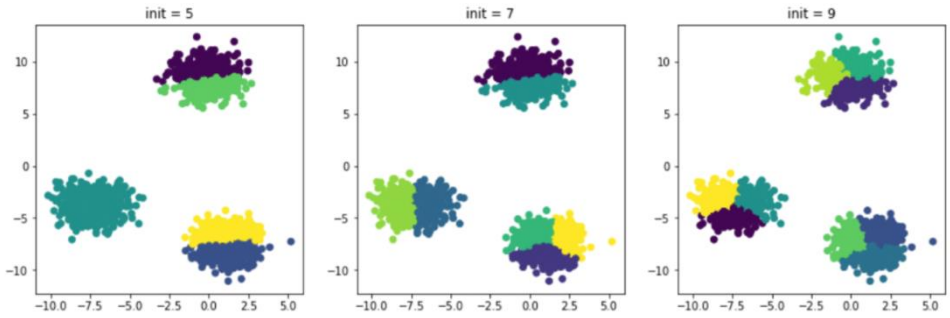
Enfin on recalcule 3 nouveaux centroïdes qui seront les centres de gravité de chaque nuage de points labellisés. On répète ces étapes jusqu'à ce que les nouveaux centroïdes ne bougent plus des précédents. Le résultat final se trouve sur la figure de droite.

Dans cet algorithme, une notion qui est très importante est la notion de distance. En effet, on utilise la distance euclidienne définit comme :

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_i-q_i)^2+\cdots+(p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

Cela nous permet donc d'évaluer la distance entre chaque point et les centroïdes. Pour chaque point on calcule la distance euclidienne entre ce point et chacun des centroïdes puis on l'associe au centroïde le plus proche c'est-à-dire celui avec la plus petite distance.

Généralement les jeux de données ont plus de deux dimensions et il est donc difficile de visualiser le nuage de points et d'identifier rapidement le nombre de clusters optimal. Supposons dans l'exemple précédent que nous n'avons pas visualiser les données avant et décidons de tester différentes fois avec un nombre de clusters initiaux différents. Voici les résultats obtenus :



4ème méthode : Méthode **CAH** (classification ascendante hiérarchique) :

La Classification Ascendante Hiérarchique (CAH) est donc un algorithme de clustering. C'est à dire qu'à partir de vos données, l'algorithme va chercher à créer des groupes d'individus homogènes :

- Des groupes dans lesquels les individus se ressemblent
- Des groupes qui se distinguent le plus possible les uns des autres

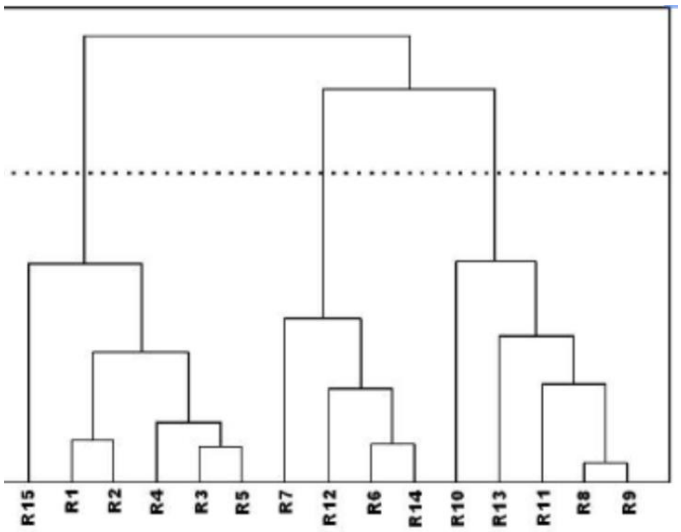
Lorsque les données sont prêtes, et le nombre de dimensions du problème a été réduit, la CAH peut être appliquée. C'est une méthode hiérarchique qui se construit étape par étape en partant du niveau le plus fin où chaque individu est seul dans son groupe jusqu'au niveau le plus agrégé où tous les individus sont dans le même groupe. Pour présenter la manière dont tout cela s'organise on construit un dendrogramme.

L'algorithme CAH se déroule de telle sorte :

- **Initialisation** : On considère que chaque client est seul et isolé dans son group
- **Itération** : On calcule ensuite la distance entre chaque groupe. Comme d'habitude plusieurs distances peuvent être utilisées. Pour la CAH, on utilise la distance de WARD qui permet de pondérer la distance en fonction du nombre d'individus appartenant au groupe. Cela permet d'éviter d'isoler les outliers dans un groupe.

On fusionne les 2 groupes les plus proches et on les relie dans le dendrogramme. Le trait qui nous permet de relier les 2 groupes dans le dendrogramme est d'autant plus long que la distance entre les groupes est élevée.

Exemple d'un dendrogramme :



Et on continue cette étape itérative encore et encore jusqu'à ce qu'il ne reste plus qu'un seul et unique groupe réunissant tous les individus.

5ème méthode : modèle de **mélange des gaussiennes** :

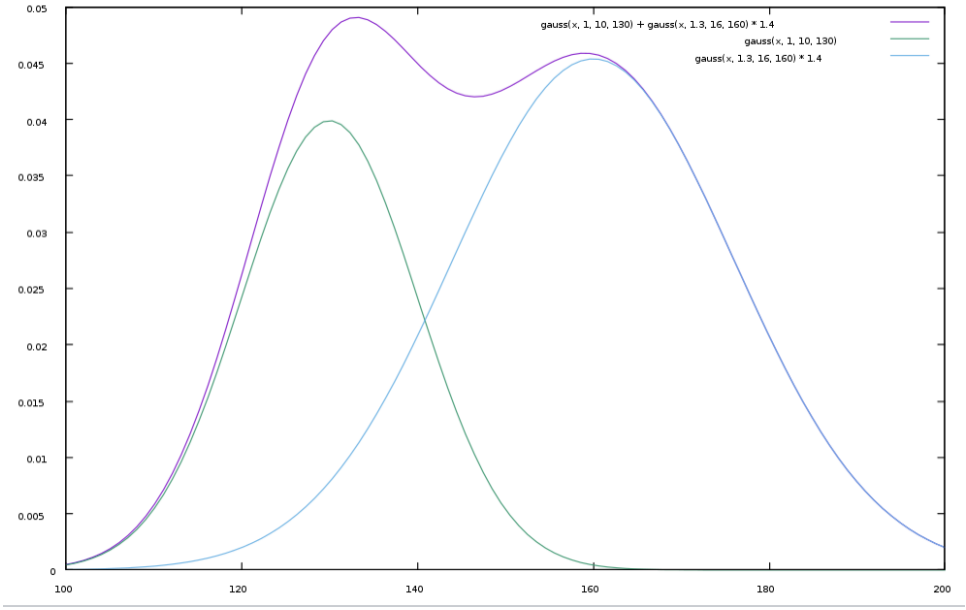
Un modèle de mélange gaussien est un modèle statistique exprimé selon une densité mélange. Il sert à estimer de manière paramétrique la distribution de variables aléatoires en les modélisant comme une somme de plusieurs gaussiennes (appelées noyaux).

Il s'agit alors de déterminer :

- La variance
- La moyenne
- L'amplitude de chaque gaussienne

Ces paramètres sont optimisés selon un critère de maximum de vraisemblance pour approcher le plus possible la distribution recherchée. Cette procédure se fait le plus souvent itérativement via l'algorithme espérance-maximisation (EM).

Les modèles de mélange gaussien permettent de reconstruire de manière efficace les données manquantes dans un jeu de données expérimentales.



1.2 Etude des classes

Dans cette deuxième partie étudier les paramètres d'appel, les attributs, les méthodes et les fonctions de différentes classes et modules. Nous allons étudier les modules suivants : K-MEANS CAH, Modèle de mélange de Gaussiennes, DBSCAN, Spectral Clustering.

1) Classe 1 : Sklearn.cluster.KMeans

Paramètres d'appels :

- N clusters (le nombre de clusters)
- Init (méthode d'initialisation)
- N_init (nombre d'itération aftin de trouver le meilleur centre°

Attributs :

- Labels (retourne l'indice pour chaque point)
- Inertia_ (l'inertie intra)
- Clusters_centers_(coordonnées des centres du cluster)

Méthodes :

- Fit()

2) Classe 2 : Sklearn.cluster.DBSCAN

Paramètres d'appel :

- eps : distance max entre deux points pour qu'ils soient considérés comme voisins.

- min_samples : nombre de voisin minimal d'un point pour qu'il soit considéré comme un core.
- metric : la méthode de calcul de distance entre deux points, par default distance euclidienne (meilleure pour calculer la distance entre deux points).

Attributs :

- core_sample_indices :
- components_ :
- labels_ : retourne la liste des individus marqués selon leurs partitions

Méthodes :

- fit(X,y=None,sample_weight) : exécute Bbscan sur la matrice de donnée X, sample_weight permet d'ajouter du poids à chaque individu.

3) Classe 3 : Sklearn.cluster.SpectralClustering

Paramètres d'appel :

- n_cluster : nombre de sous-ensembles
- n_components : le nombre de vecteur propres pour découper le nuage en sous ensemble (default = n_cluster)
- assign_labels : méthode pour assigner les partitions aux points
- n_inti : le nombre d'initialisation de l'algo k-means pour trouver la meilleure répartition des centres
- affinity : on peut rentrer la matrice adjacente pour qu'il en déduit la laplacienne
- n_neighbors : le nombre de voisin pour la matrice adjacente lors de la methode des plus proches voisins.
- gamma :

Attributs :

- affinity_matrix : retourne la matrice adjacente
- n_features_in_ :
- lables_ : retourne la liste des individus marqués selon leurs partitions

Méthodes :

- fit(X) : exécute le clustering spéctral sur la matrice de donnée X

4) Classe 4 : sklearn.mixture.GaussianMixture

Paramètres d'appel :

- n_components : nombre de cluster (nb de gaussienne différentes)
- covariance_type : type de matrice de covariance, par défaut « full » soit chaque cluster à sa matrice de covariance (vu en cours)

- n_inti : le nombre d'initialisation de l'algo pour trouver les meilleurs premier paramètres des gaussiennes
- Max_iter : nombre max d'itération
- Init_params : la méthode pour l'initialisation , défaut = kmeans

Attributs :

- Covariances_ : renvoi les covariances de chaque cluster

Méthodes :

- fit_predict(X) : exécute le mélange sur la matrice de donnée X et retourne la liste des individus marqués selon leurs partitions

5) Classe 5: scipy.cluster.hierarchy

Méthodes

- linkage(data,methode,optimal_ordering) :
- fcluster(Z_data,t,criterion) : return labels

1.3) Expérimentation

Le but de cette partie est de tester les cinq méthodes sur différents jeux de données. Nous avons trois fichiers textes que nous allons convertir en matrices numpy . Pour faciliter le traitement nous allons centrer puis réduire les données.

Nous utilisons la librairie *pandas*, *numpy* et la classe *StandardScaler* de *sklearn.preporcessing* pour faire cela. Nous voulons connaitre pour les trois jeux de données les meilleures méthodes de clustering. Pour cela on va déterminer pour chaque donnée les meilleurs paramètres de chaque méthode.

La méthodologie mise en place est la suivante :

- On cherche le meilleur score de silhouette pour chaque paramètre et le nombre de cluster k. Le coefficient de silhouette est compris entre -1 et 1, plus on s'approche de 1 plus les points sont correctement classés, plus proche des points de leur cluster que des autres points.
- On affiche les résultats seulement pour un jeu de donnée par méthode pour ne pas surcharger le rapport, mais nous donnons tous nos résultats

Meilleurs paramètres pour k-means :

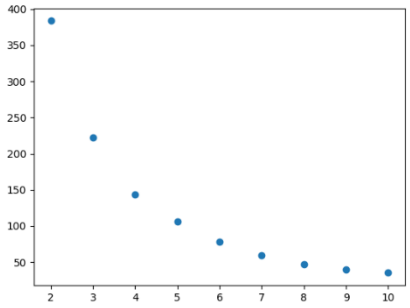
Pour k-means il faut choisir la méthode d'initialisation des centres de gravités initiaux, soit entre « k-means++ » et « random ». Il faut également choisir le nombre de fois (n_inti) que l'algorithme va faire tourner cette méthode d'initialisation pour trouver la meilleure disposition des centres.

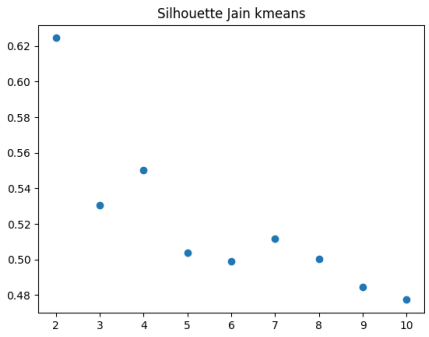
Pour cela on utilise le coefficient ARI, on trouve "k-means++" étant la meilleure méthode avec le plus de stabilité et n_init =10 un nombre suffisant.

Désormais, en gardant ces paramètres on va analyser l'inertie intra et le coefficient de silhouette pour trouver le meilleur K pour chaque type de données.

On affiche les mesures sur un graphe. Pour l'inertie on s'intéresse au coude et pour le coefficient de silhouette la plus grande valeur.

Voici les résultats que nous obtenons pour k=3 et k=2.





Dans le graphique affichant l'inertie intra du nuage pour chaque k, on observe une forte chute à **k=3**. On obtient le meilleur score de silhouette pour **k =2**.

On en déduit que pour le jeu de donnée Jain, **k = 2** semble être adapté avec la méthode k-means.

On fait la même chose les deux autres jeux de données. On trouve k= 5 pour Aggregation et **k = 4** pour Pathbased.

Meilleurs paramètres pour DBSCAN

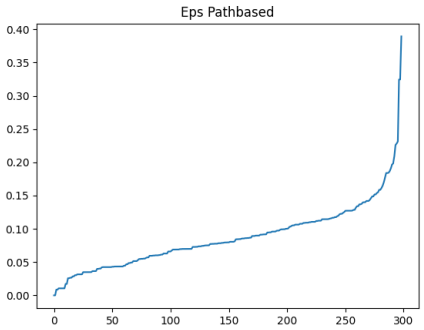
Pour cette méthode il s'agit de trouver le meilleur EPS et le meilleur nombre minimal MIN_SAMPLES.

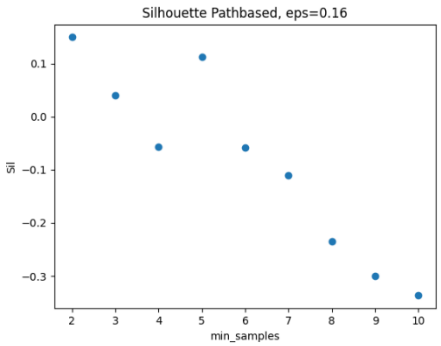
Pour trouver le meilleur EPS, on va se baser sur une théorie basée sur la distance avec la recherche des plus proches voisins pour **n=3** (voir source[1]). On affiche les distances dans l'ordre croissant. On s'intéresse à la valeur de la distance au niveau du coude, nous donnant le meilleur EPS.

Une fois avoir trouver le meilleur EPS, on s'intéresse au score de silhouette pour différentes valeurs de MIN_SAMPLES. On prend le MIN_SAMPLES pour un score le plus proche de **1**.

[1] Microsoft Word - 12.docx (iop.org)

EPS Pathbased





Nous observons un coude pour une valeur en ordonnées de 0.16 environ. Soit **EPS=0.16.**

Nous obtenons également le score de silhouette le plus proche de 1 pour **MIN_SAMPLES = 2**

Nous réalisons la même chose les deux autres jeux de données.

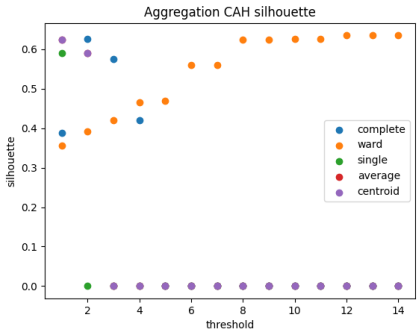
- Aggregation : **EPS= 0.11**, **MIN_SAMPLES= 3**
- Jain : **EPS= 0.20**, **MIN_SAMPLES=3**

Meilleurs paramètres pour CAH:

Nous souhaitons désormais connaitre la meilleure méthode utilisée pour calculer la matrice des distances ainsi que le meilleur threshold (t) pour découper les branches en cluster.

Nous allons tester pour chaque méthode, différentes valeurs de t et calculer le score de silhouette pour connaitre les meilleurs combos.

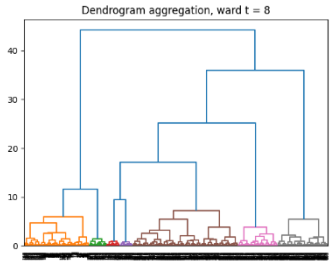
AGGREGATION CAH SILHOUETTE :

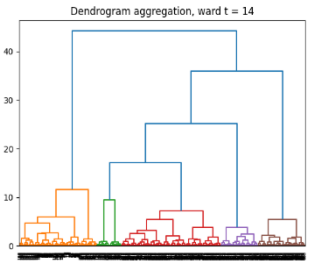


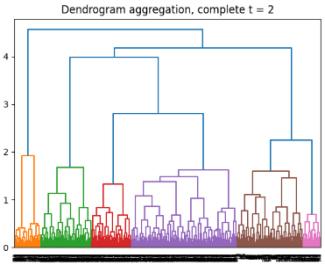
On obtient les meilleurs scores pour la méthode Ward avec t allant de 8 à 14.

Également avec la méthode complète avec **t =2**.

Nous pouvons observer les dendrogrammes pour trouver les meilleurs paramètres en tenant en compte du k résultant :







Nous remarquons que, plus les branches sont longues avant de réaliser une jointure plus les classes sont éloignées. On observe ainsi un meilleur partitionnement pour Ward avec **t=8**, donnant 7 clusters.

Nous décidons donc de garder Ward avec **t=8**, qui correspond au meilleur résultat avec **k = 7**.

Nous faisons la même chose les deux autres jeux de données.

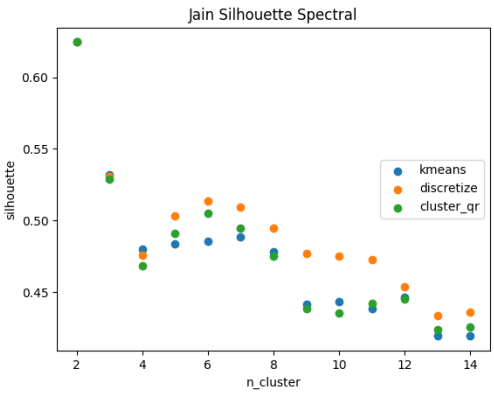
- Jain : complete avec **t=4** donnant **k = 2**
- Pathbased : Ward et **t=14** donnant **k = 4**

Meilleurs paramètres pour Spectral clustering

Pour cette méthode on s'intéresse au choix de la méthode pour assign_labels et au meilleur k.

On va tester pour chaque méthode, différentes valeurs de k et calculer le score de silhouette pour connaitre les meilleurs combos.

Jain Silhouette Spectral :



Nous obtenons le meilleur score pour la méthode cluster_qr avec **k = 2** (n_cluster).

Enfin, nous réalisons la même chose les deux autres jeux de données.

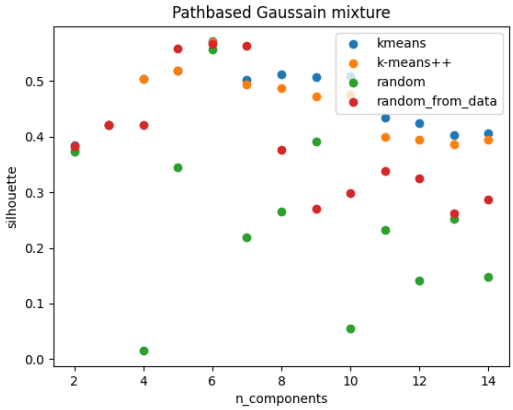
- Aggregation : cluster et **k=5**
- Pathbased : discretize et **k=8**

Meilleurs paramètres pour mélange de gaussiennes

Nous allons desormais nous intéresser cette fois aux méthodes d'initialisation des paramètres des gaussiennes et le nombre de cluster k (n_components).

Nous procédons de la même manière que pour CAH.

Pathbased Gaussian mixture :



Nous obtenons donc le meilleur score pour la méthode K-Means pour **k = 6**. Nous répetons cette méthode avec les deux autres jeux de données.

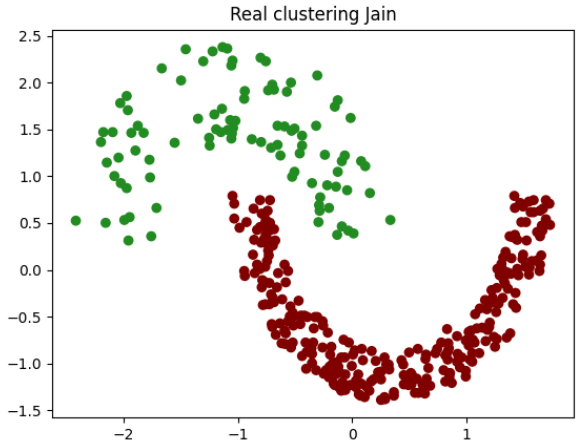
- Aggregation : random_from_data et **k=5**
- Jain : random_from_data et **k=4**

Résultat :

Nous concluons donc que nous afficherons les points selon les couleurs de leur cluster.

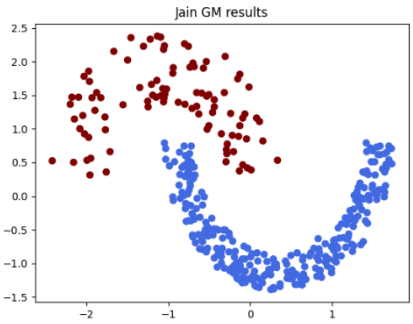
Test quantitatif pour les différentes méthodes :

1) Méthode Jain:



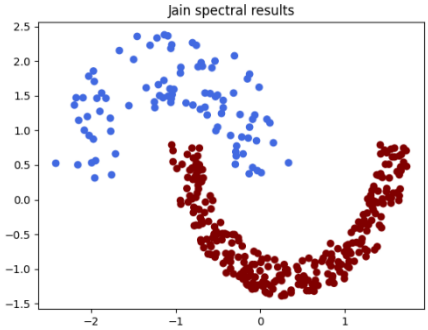
Ainsi, nous observons 2 clusters.

2) Méthode GM:



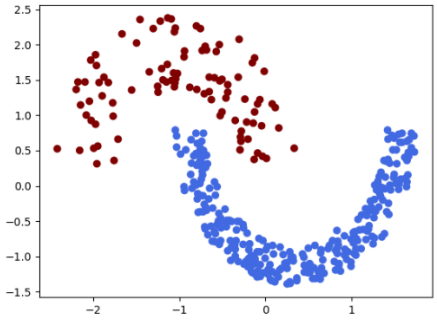
Nous observons des résultats identiques.

3) Méthode Spectral:



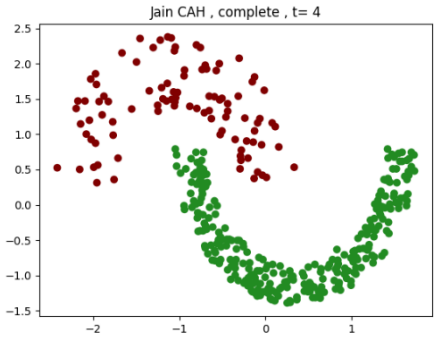
Nous observons également des résultats identiques.

4) Méthode K- Means:



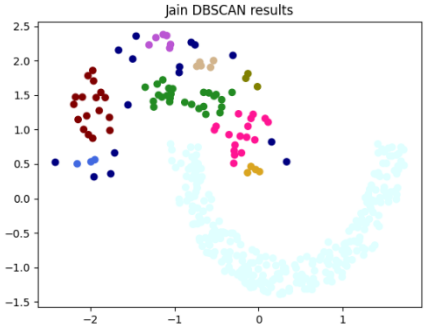
Nous observons des résultats identiques également.

5) Méthode CAH :



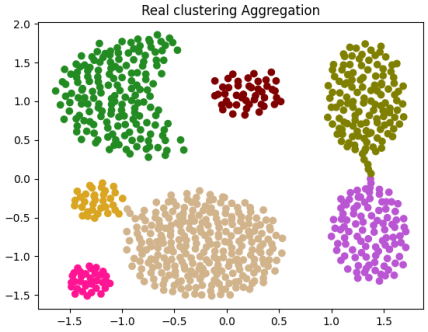
Nous observons des résultats identiques également.

6) Méthode BDSCAN :



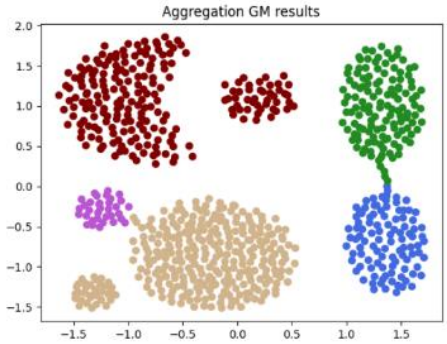
Nous pouvons donc constater que les résultats ici sont très différent.

“Aggregation” pour les différentes méthodes :



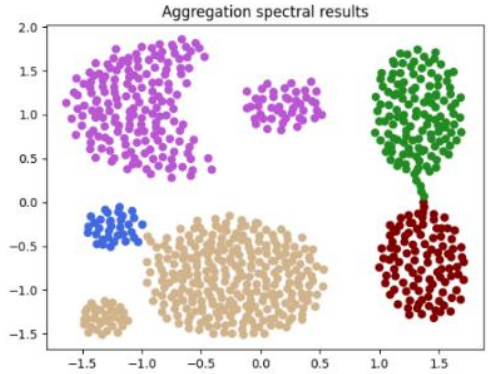
Nous observons 7 clusters.

1) Méthode GM :



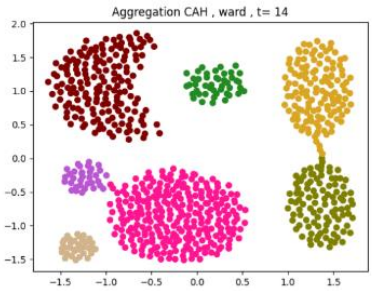
Nous observons des résultats identiques également.

2) Méthode Spectral :



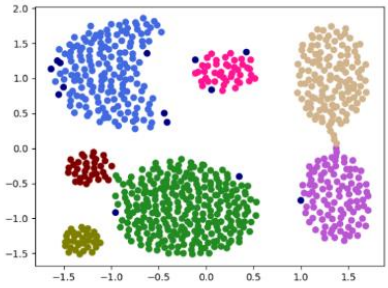
Nous constatons des résultats plutôt identiques, seulement deux clusters aborbés à deux autres.

3) Méthode CAH :



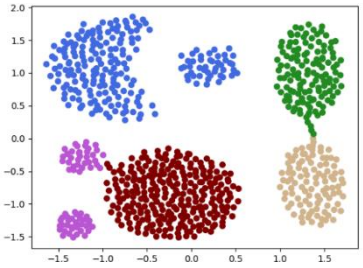
Nous observons des résultats identiques également.

4) Méthode BDSCAN:



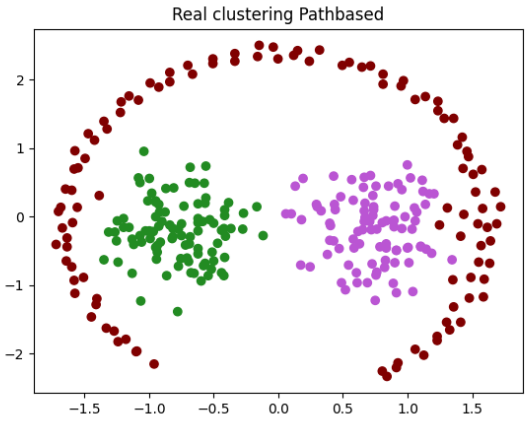
Nous observons des résultats assez identiques, avec simplement quelques anomalies.

5) Méthode K-Means



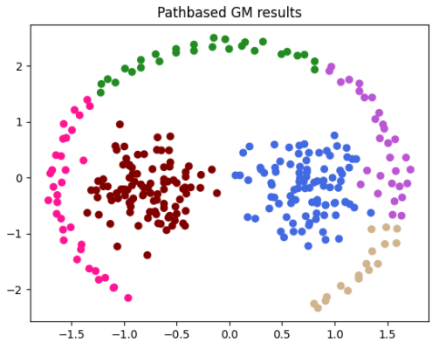
Nous observons des résultats identiques également, avec deux petits clusters différents.

Pathbased pour les différentes méthodes :



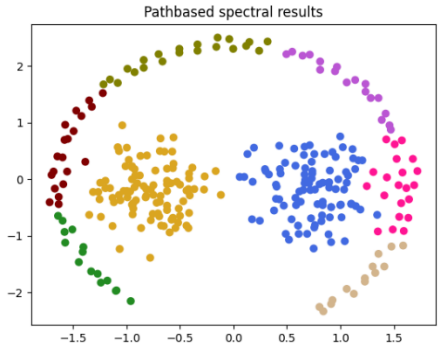
On observe 3 clusters.

1) **Méthode GM :**



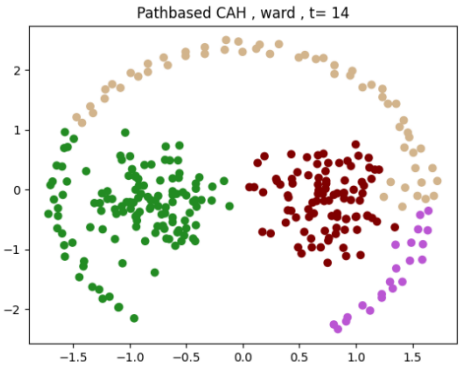
Nous observons des résultats assez différents.

2) **Méthode Spectral :**



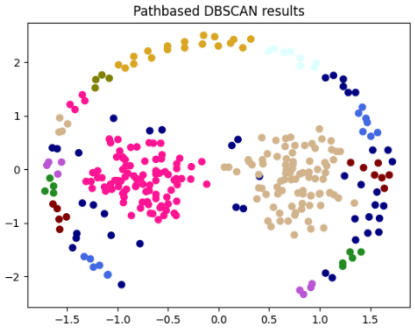
Nous observons des résultats très différents, seuls les les deux centraux sont correct)

3) **Méthode MCAH** :



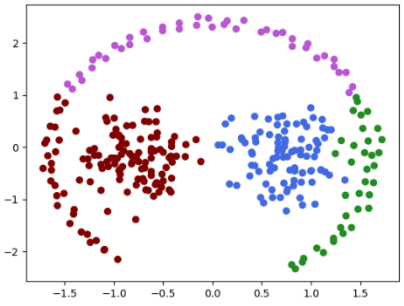
Nous observons des résultats très proches avec un cluster de plus.

4) Méthode BDSCAN :



Nous observons des résultats très différents.

5) Méthode K-Means :



Nous observons des résultats très proches, avec un cluster de plus.

Test ARI, qualitatif :

Pour comparer la similarité entre les partitions obtenues et les partions réels nous allons utiliser l'indice de Rand. Cet indice prend les deux partitions en paramètre et va compter le nombres de pairs

qui se retrouve dans le même cluster sur les deux partitionnement (a), ainsi que les pairs qui se retrouve dans deux cluster différents (b). On additionne ces deux valeurs puis on divise le tout sur le nombre total de paires (c).

Ainsi la valeur de l'indice varie entre 0, partitions totalement différentes et 1, partitions identiques.

On utilise l'indice ajusté ARI avec la classe adjuster_rand_score de sklearn.metrics.cluster.

1) Jain
   - GM = 1
   - Spectral = 1
   - CAH = 1
   - DBSCAN = 0.88
   - Kmeans =1
2) Aggregation
   - GM = 0.85253
   - Spectral = 0.85253
   - CAH = 1
   - DBSCAN = 0.97
   - Kmeans = 0.92
3) Pathbased
   - GM = 0.76
   - Spectral = 0.72
   - CAH = 0.66
   - DBSCAN = 0.61
   - Kmeans = 0.64

Résultat :

Ce test quantitatif et qualitatif nous permettre d'affecter les meilleures méthodes aux trois jeux de données.

--> Jain → best = GM , Spectral, Kmeans, CAH

--> Aggregation → best = CAH, BDSCAN, Kmeans, GM, Spectral

--> Pathbased → best = GM, CAH

Nous pouvons conclure que la majorité des méthodes sont efficaces pour des clusters distants comme dans les jeux de données Aggregation voire jain. La forme du nuage de jeu Pathbased représente plus de difficulté, mais aucune méthode se démarque. On peut noter l'efficacité da la méthode mélange de gaussiennes et kmeans sur l'ensemble des données.

BDSCAN n'est pas la méthode la plus adaptée lorsqu'il y a des différences densité dans le nuage, par exemple pour Jain. A l'inverse très efficace pour des clusters uniformes comme pour le jeu de donnée Aggregation.

2) Analyse et partitionnement de la clientèle d'un grand magasin

Un grand magasin nous fournit une grande quantité de données sur ces clients. Nous devons faire la classification des clients pour guider la stratégie du magasin. Cette classification est non supervisée, nous ne connaissons pas les classes qui existent chez les clients et les données ne sont pas étiqueté pour guider la classification. Nous allons utiliser les méthodes vues dans la partie 1 sur ces nouvelles données et en déduire des partitions.

2.1 Examen des données

Pour débuter nous allons faire une première analyse des données après les avoir importés. Pour faire l'analyse on va convertir les données en DataFrame.

Nous pouvons regarder le nombre de valeurs avec l'attribut size. On observe 64360.

Ensuite, nous analysons la forme de la matrice avec l'attribut shape. On obtient (2240,29), on a donc 2240 clients et 29 variables.

Desormais, nous pouvons constater le type de données des variables. Nous remarquons que la majorité des variables sont des int64 ou float64. Il y a seulement trois variables qui sont des chaines de caractères.

On affiche leur valeur unique avec l'objet set.

- Education = {'2n Cycle', 'Basic', 'Master', 'Graduation', 'PhD'}
- CivilStatus = {'YOLO', 'Single', 'Together', 'Absurd', 'Widow', 'Alone', 'Married', 'Divorced'}
- Registration = '%d-%m-%Y'

On s'intéresse désormais aux valeurs manquantes et aberrantes.

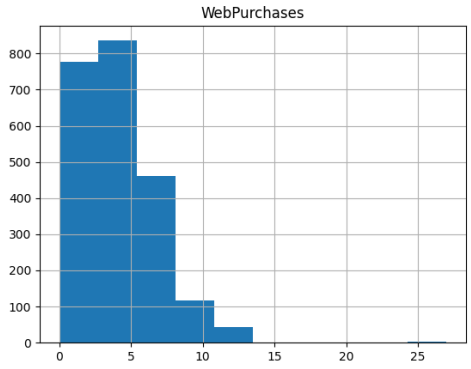
On regarde toutes valeurs manquantes pour toutes les variables avec :

```
for head in data.head() :
   print( head ,"nb of VA :", data[head].isna().sum())
```

On observe que la variables Income possède 24 éléments NA. Ce qui est raisonnable, nous allons pouvoir remplir ces trous dans la partie suivante. Pour les variables aberrantes on analyser les variables en affichant leur histograme avec hist() de Dataframe.

```
data.hist(column='WebPurchases')
plt.show()
```

Exemple pour la variable WebPurchases :



On ne remarque pas de variables aberrantes.

2.2 Pré-traitement des données

Certaines données que nous avons vues plus haut ne sont pas exploitables.

Les valeurs manquantes de Income :

Il faut remplir 24 valeurs, ce qui est très peut sur 2240 valeurs de Income. On peut se permettre de remplacer les valeurs manquantes par la moyenne de la variable. On utilise la méthode fillna de Dataframe avec la fonction mean de Numpy.

```
value = {"Income": data["Income"].mean()}
data = data.fillna(value)
```

Variables qualitatives en variables numériques :

Nous avons vu qu'il y avait deux variables de type chaîne de caractères, correspondant à des valeurs précises. On va les remplaces par des indices.

```
le = preprocessing.LabelEncoder()
education = le.fit_transform(data["Education"])
data["Education"] = education
civilstatus = le.fit_transform(data["CivilStatus"])
data["CivilStatus"] = civilstatus
```

Pour ce qui est de la variable Registration, qui contient des dates de la forme '%d-%m-%Y', on va convertir ces valeurs en nombre d'année. Au lieu d'avoir la date d'inscription on aura le nombre d'année que le client est inscrit.

```
list_registration = data["Registration"]
#transform date of registration into date object
dates_registration = [datetime.strptime(date,'%d-%m-%Y') for date in
list_registration]
#list of number of year
list_registration_nby = [relativedelta.relativedelta(datetime.today(),date
).years for date in dates_registration]
```

```
data["Registration"] = list_registration_nby
```

On peut également transformer les valeurs de la variable BirthYear nous renseignant sur la date de naissance des clients. Pour faciliter la visualisation des données on peut convertir ces années en âges.

```
list_year = data["BirthYear"]
data["BirthYear"] = [ datetime.today().year - y for y in list_year]
```

Suppression de variables :

On décide de supprimer certaines variables que nous jugeons inintéressantes pour la classification.

Nous décidons de retirer les variables suivantes avec la méthode drop de Dataframe :

```
["Accept3","Accept4","Accept5","Accept1","Accept2","Claims","Group","SubGroup"
,"AcceptLast","ID"]
```

Elles nous renseignent sur aucunes informations pertinentes. Les variables Group et SubGroup n'ont aucune signification par rapport à l'individu. Les variables Accept* se résument dans la variable DiscountPurchases.

Normalisation :

Les variables possèdent tous des mesures différentes, il est intéressant d'effectuer une normalisation avec StandardScaler. En utilisant la méthode fit_transform on réalise également une réduction.

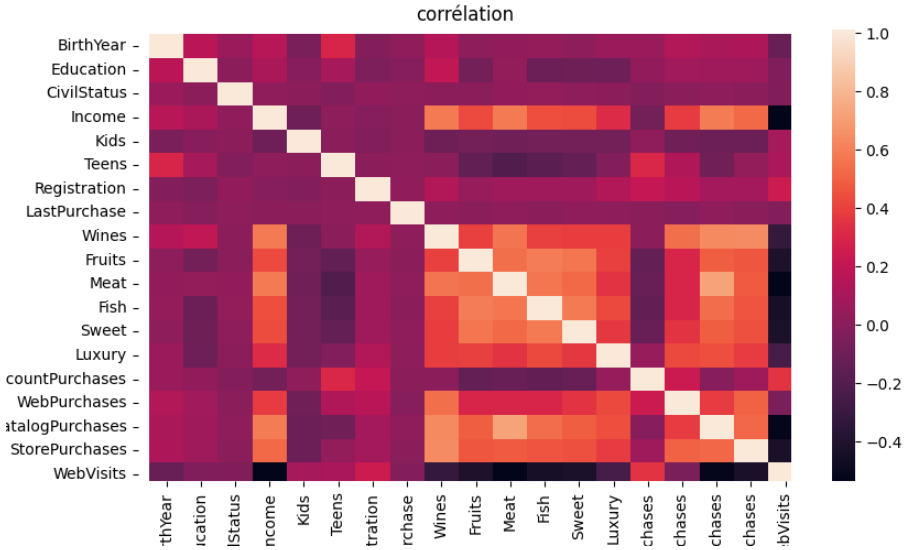
```
scaler = StandardScaler()
data_Z = scaler.fit_transform(data)
```

Les données sont désormais prêtes à être traiter.

2.3 Analyse de la corrélation

On peut commencer pas regarder la corrélation entre chaque couple de variables. Nous donnant une première idée des résultats que nous allons obtenir.

On utilise la méthode corr sur nos données. On affiche les résultats avec la méthode heatmap de la librairie seaborn.



Nous remarquons une forte corrélation entre la revenue et la consommation des différents aliments et les différents achats en physique. Plus faible corrélation avec les achats en ligne.

Il y a également une corrélation entre l'âge et le nombre d'enfants ainsi qu'une corrélation négative entre les achats d'aliments et les visites sur internet

On en conclut donc que : la corrélation est plus forte entre les achats en physiques et les aliments qu'avec les achats en ligne. On observe une corrélation positive entre StorePurchase et les différents aliments, de même avec CataloguePurchases.

2.4 Analyse exploratoire des données

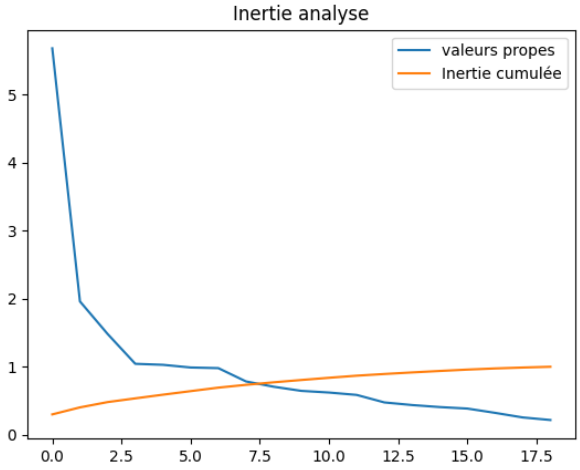
Les données que nous avons possèdes beaucoup de variables, il peut être intéressant de réduire les dimensions en réalisant une ACP normée. On pourra par la suite afficher une représentation des individus et variables selon des certains axes. Ces représentations nous permettrons d'interpréter les résultats du clustering.

Réalisation de l'ACP normée :

Tout d'abord, il faut déterminer le nombre d'axes que nous allons garder pour réaliser la projection.

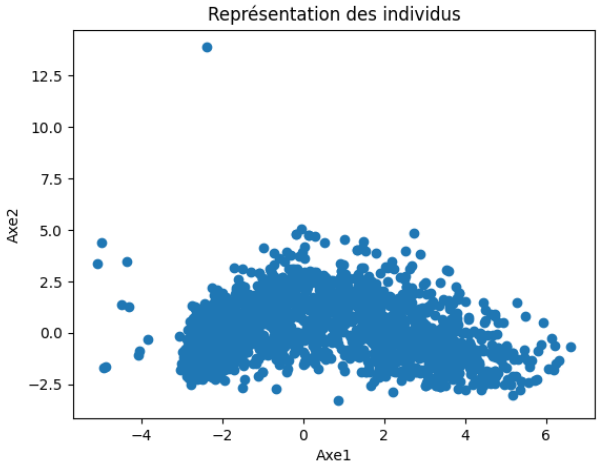
On va utiliser deux critères vus en cours :

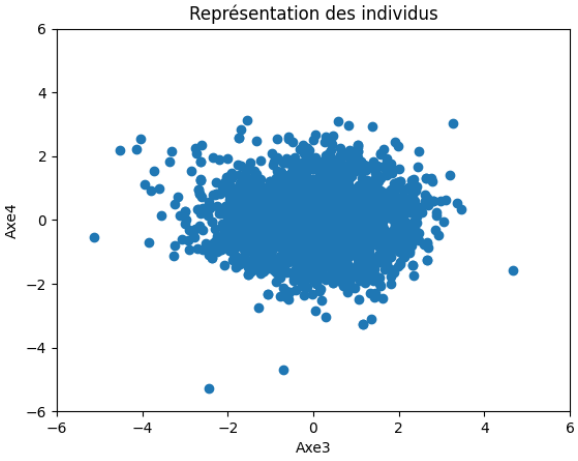
- En bleu la courbe des valeurs propres des axes.
- En orange la courbe du cumul de l'inertie de chaque axe.
- Sur l'axe orange on obtient une qualité globale à 70% pour 6 axes.
- Sur l'axe bleu on voit un coude pour 4 axes.



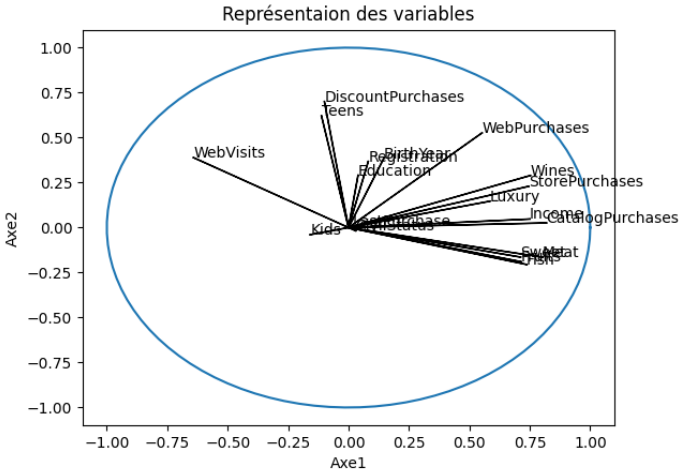
On en déduit que les informations pertinentes se trouves sur les 4 premiers axes. Ce sont les axes que nous allons utiliser pour réaliser l'ACP.

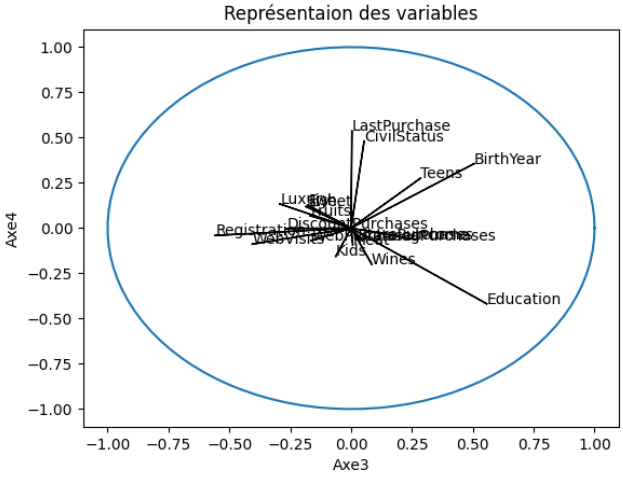
Représentation des individus :





Représentation des variables :





Explications des différentes variables des différents axes :

<u>Variables déterminants l'axe 1 :</u>

-> *Positivement* : income, CatalogPurchase, StorePurchases, wines

-> *Négativement* : Webvisits

<u>Variables déterminants l'axe 2 :</u>

-> *Positivement* : DiscountPurchases, Teens

<u>Variables déterminants les deux axes (1,2) :</u>

-> *Positivement* : WebPurchases

<u>Variables déterminants l'axe 3 :</u>

-> *Négativement* : Registration

<u>Variables déterminants l'axe 4 :</u>

-> *Positivement* : LastPurchase, CivilStatus

<u>Variables déterminants les deux axes (3,4) :</u>

-> *Positivement* : Teens, Birthyear

-> *Négativement* : Education

Nous pouvons en déduire que l'axe 1 représente les personnes aisées et qui préfère aller dans les magasins, nous concluons que ce sont majoritairement des personnes plus âgées.

Nous constatons donc une corrélation entre les couples suivants :

- (income, CatalogPurchase)
- (StorePurchases, wines)
- (income, CatalogPurchase) et (StorePurchases, wines)
- (DiscountPurchases, Teens)
- (LastPurchase, CivilStatus)
- (Teens, Birthyear)

Nous remarquons une décorrélation entre les groupes suivants :

- (DiscountPurchases, Teens) et (Income, CataloguePurchase)
- (DiscountPurchases, Teens) et (StorePurchases, wines)
- WebPurchases et WebVisits
- WebVisits et (income, CatalogPurchase, StorePurchases, wines)

Nous pouvons s'attendre à observer 4 clusters :

- (income, CatalogPurchase, StorePurchases, wines)
- (DiscountPurchases, Teens)
- WebVisits
- WebPurchases

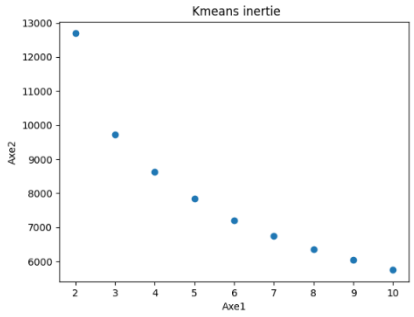
Nous pouvons également s'attendre à observer 3 clusters :

- (income, CatalogPurchase, StorePurchases, wines)
- (DiscountPurchases, Teens)
- WebVisits

2.5 Clustering des données

1) Méthode des K-Means

Pour les paramètres on reprend les résultats de la partie 1, soit n_init = 10 et ini='k-means++'.

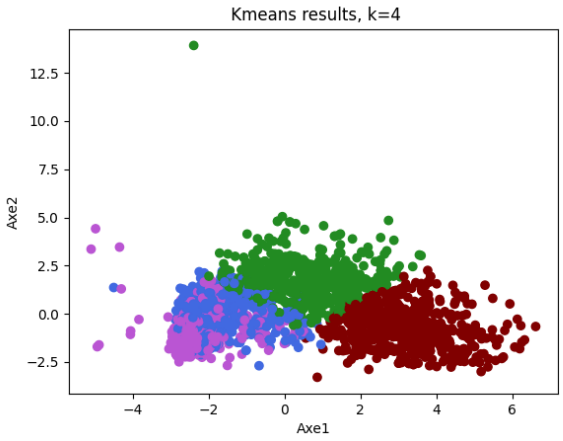


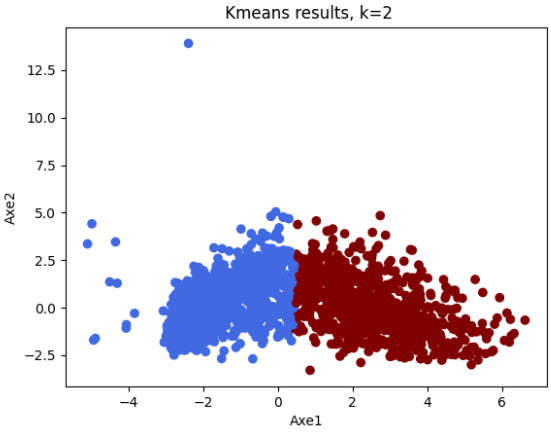
Avec le premier graphique nous remarquons que l'inertie chute rapidement pour des petites valeurs de k. On peut voir un coude pour **k = 3**.

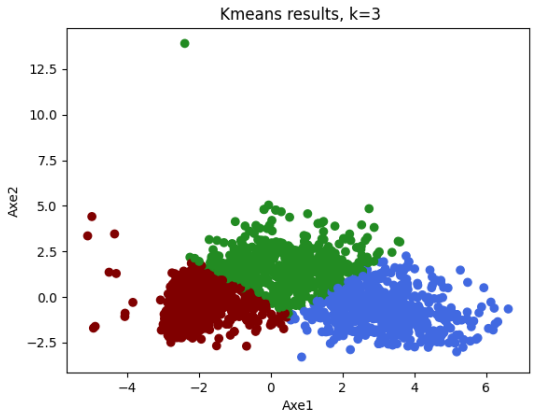
Avec le second graphique nous observons les meilleurs scores de silhouette pour k à 2,3 et 4.

On peut ainsi retenir ces trois valeurs pour essayer l'algorithme k-means.

Résultat :



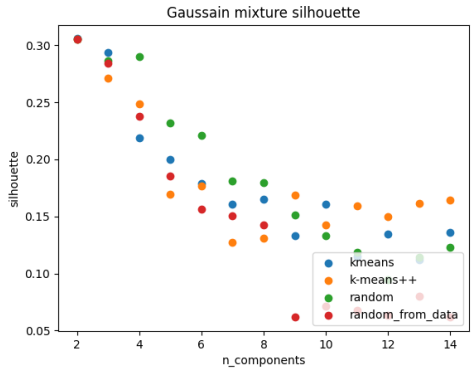




Nous concluons que le résultat le plus intéressant est celui obtenu pour **k=3**.

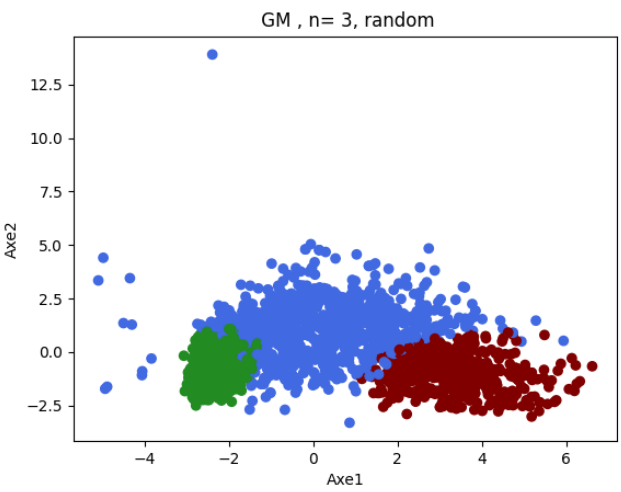
2) Méthode mélange de gaussiennes :

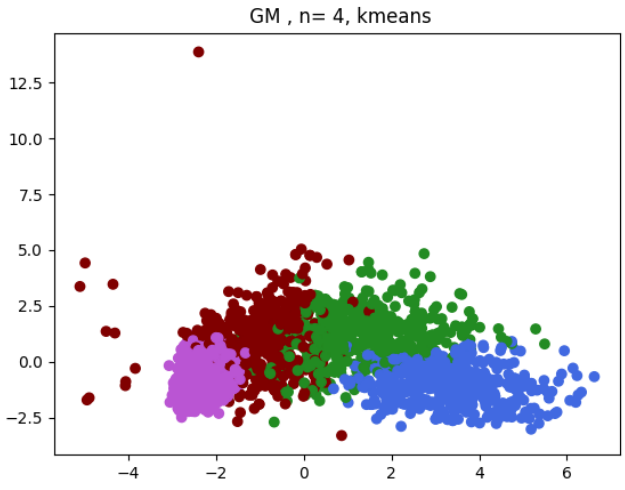
Evaluation des meilleurs paramètres et de k :



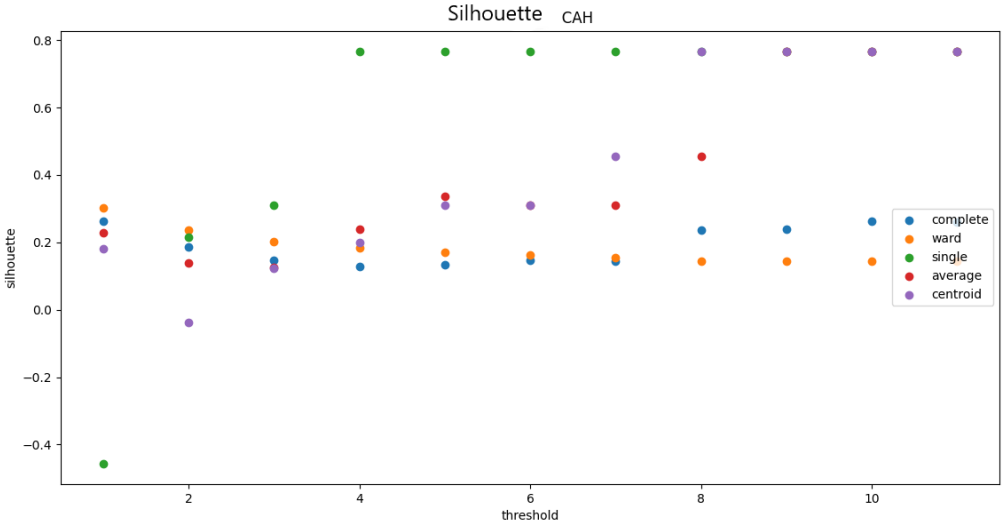
Nous obtenons les meilleurs scores de silhouette pour **n = 2,3** avec kmeans , **n = 3** avec random et **n = 2,3** avec random_from_data. Pour n=4, nous obtenons de même des bons résultats avec kmeans et k-means++.

Nous essayons avec ces paramètres et les résultats les plus intéressant sont les suivants :

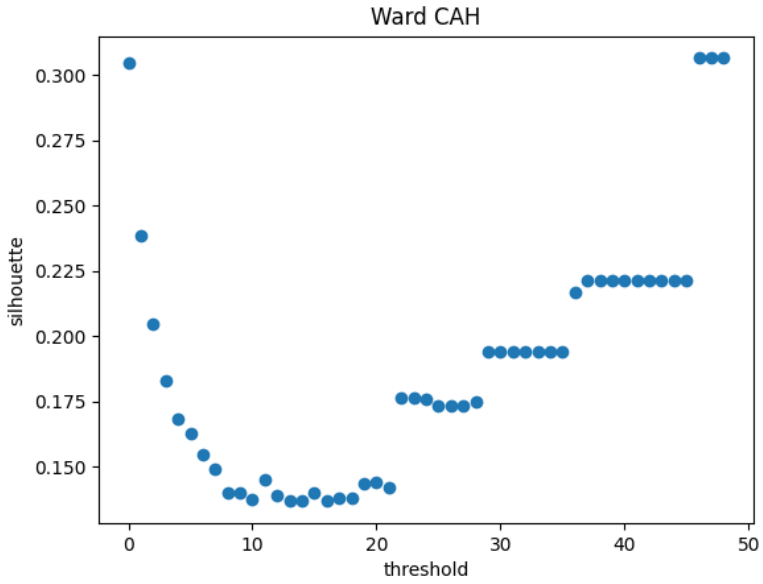




Pour la méthode CAH nous obtenons :



Evaluation des meilleurs paramètres



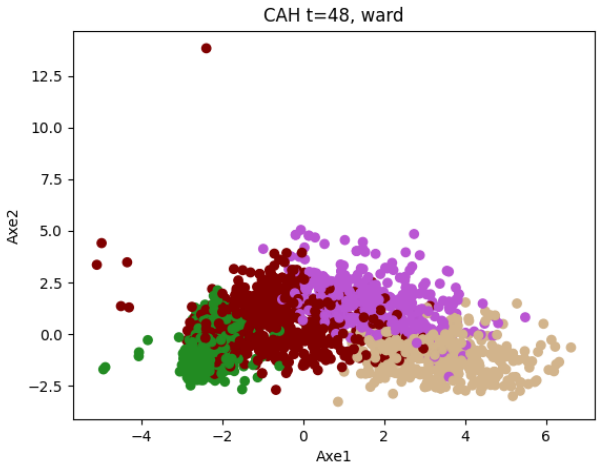
Nous observons sur le premier schéma 8 valeurs de silhouette à 0.8, nous pensons que cela soit une anomalie.

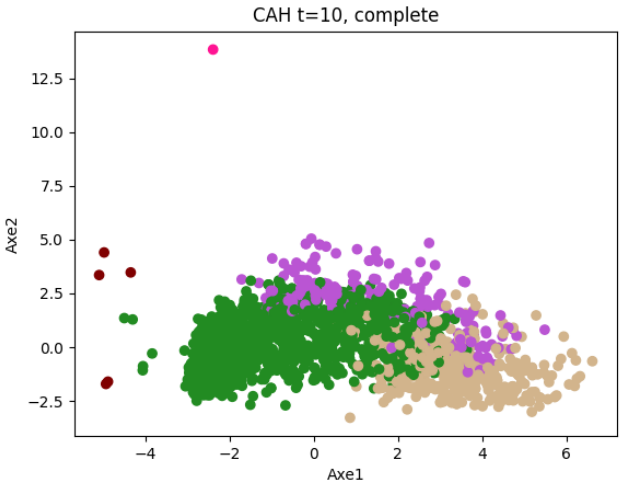
On obtient les meilleurs scores de silhouette pour

- **t=2** avec complete où nous obtenons un **k très élevé**
- **t = 3** avec single où nous obtenons **k =1**
- **t=10** avec complete où nous obtenons **k=5**
- **t=48** avec ward où nous obtenons **k=4**.

On garde les deux derniers groupes de paramètres car elles nous semblent les plus intéressants. Les deux premiers donnent des k inexploitables.

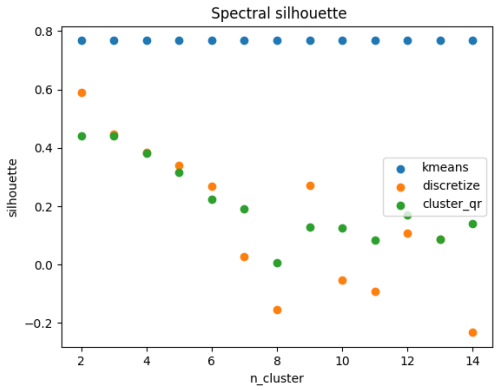
<u>Résultats</u> :





3) *Méthode spectral clustering*

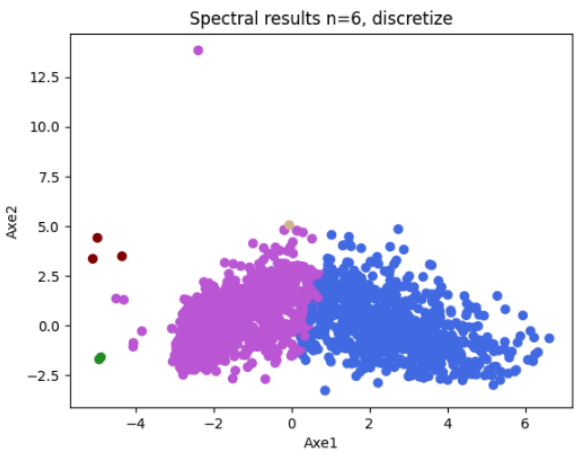
Evaluation des paramètres

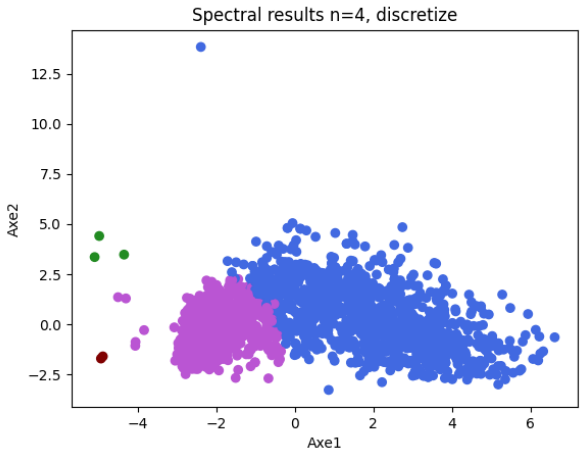


On obtient des valeurs de silhouette intéressantes pour :

- **k = 4,6** avec discretize
- **k = 3** avec cluster_qr

Résultats:

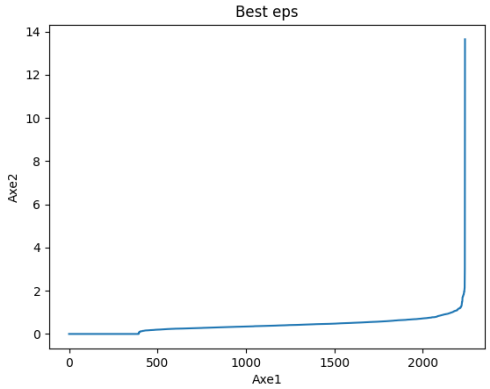




Les résultats sont peu réalistes lorsqu'on compare aux résultats précédents et à l'analyse des corrélations.

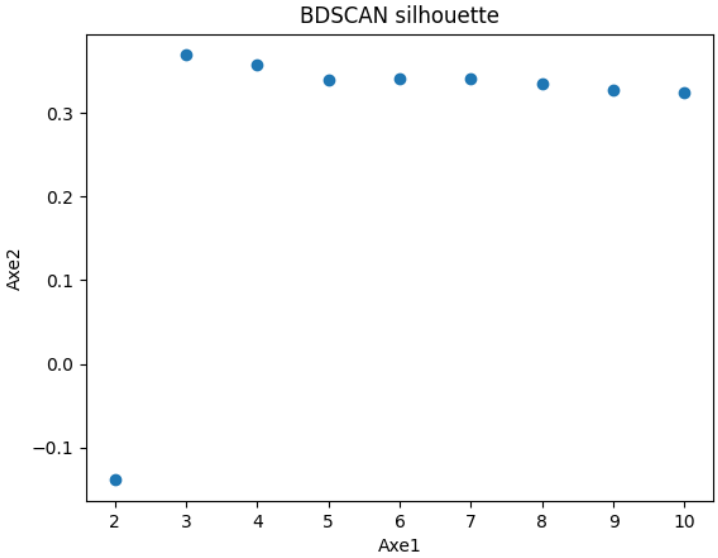
4) Méthode BDSCAN

Evaluation des paramètres :



D'après la méthode vue en partie 1, la meilleure valeur de Zps est 1.3.

Les meilleures valeurs de min_samples :

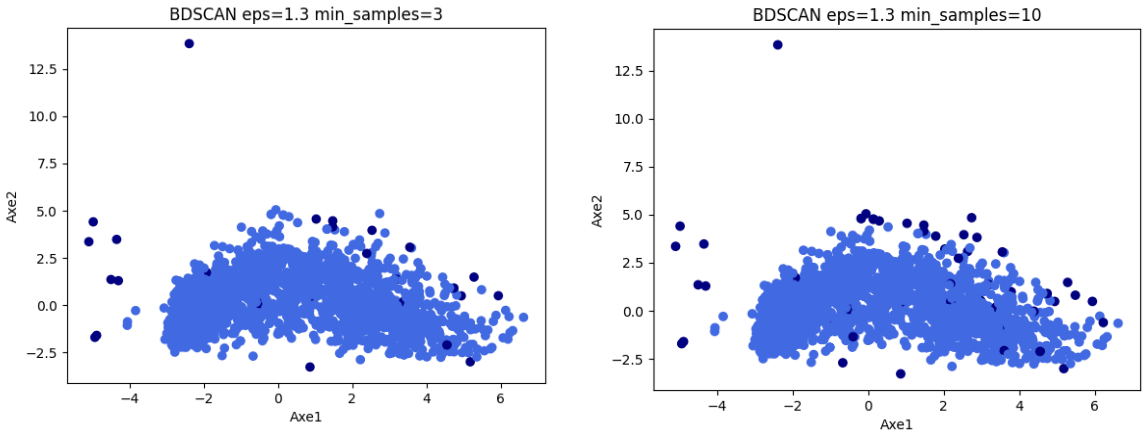


Pour **Eps =1.3** E, on obtient le meilleur score de silhouette pour **min_samples= 3** .

Les scores sont très proches. On peut également prendre **min_samples = 10.**

Résultat :

Pour **min_samples = 3** et **Eps = 1.3**



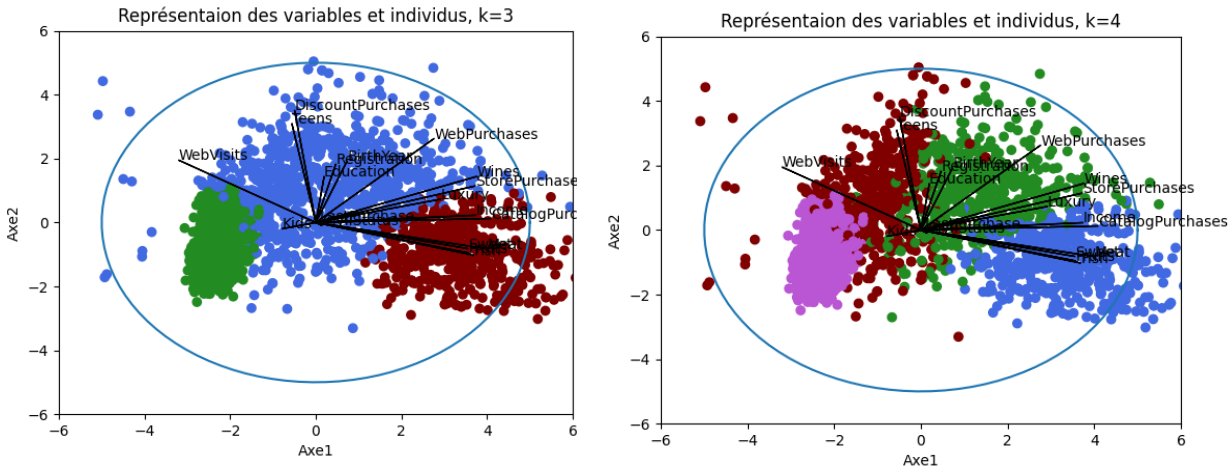
On obtient deux clusters qui ne sont pas exploitables.

Conclusion 2.5 :

Finalement, nous constatons que, graphiquement les meilleurs partions semblent être obtenu pour un nombre de 3 voire 4 clusters pour les méthodes K-Means et mélange de gaussiennes.

Interprétation :

Nous souhaitons désormais savoir quel k choisir pour obtenir la meilleure interprétation possible. Pour cela on peut reprendre le cercle de corrélation des variables et le fusionner avec les représentations des individus pour le mélange des gaussiennes.



On estime la représentation pour trois clusters la plus représentative des profils clients. On réalise la liste des profils clients pour ce résultat.

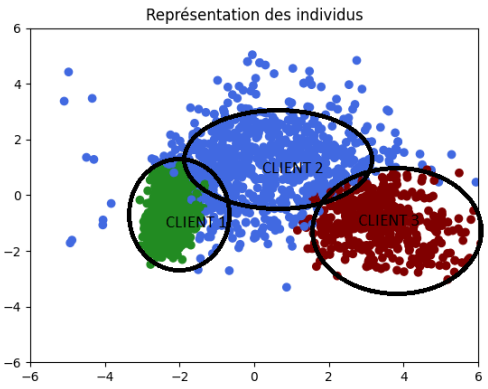
Income étant la variable la plus représentative de l'axe 1 et la variable la plus corrélé avec les autres variables, on peut interpréter nos résultats selon elle. On va catégoriser trois classes selon leur revenu, ce qui explique ensuite certains comportements.

Profil des clients :

1- Première classe au revenu modeste représente les clients visitant le plus de fois le site internet. La variable WebVisits étant corrélé négativement à la variable Income. On retrouve cette classe à gauche du nuage des individus.
2- La deuxième classe représente les revenus moyens possédant de nombreux enfants et à l'affut des réductions. On fait cette déduction comme la variables Teens est davantage décorrélé à Income mais corrélé à DiscountPurchases. On retrouve cette classe au centre du nuage des individus.
3- La dernière classe concerne les revenus aisés avec des achats sur catalogue et en magasin. Ces clients achètent des produits de luxes. On retrouve ce groupe à droite du nuage des individus. On retrouve également les achats des aliments dans ce cluster mais davantage car ces variables sont décorrélés aux visites sur internet et corrélé aux achats en magasin et
4- sur catalogue.

On peut tout de même rajouter dans le cas de la représentation à quatre clusters, une classe de clients effectuant leur achat en ligne sans être corrélé au revenu. On retrouverait ce groupe en haut à droite du nuage comme dans le cluster vert dans le second graphique.

Voici la représentation des profils clients.



Conclusion du projet :

Tout d'abord, ce projet nous a permis de comprendre de manière plus poussée et d'en comprendre leur application à travers de divers exemples. Dans la partie 1.1) nous avons également de nouvelles méthodes ce qui nous permet d'étendre nos connaissances et d'ainsi, avoir conscience qu'il existe un grand nombre de moyen de résoudre un problème.

Ce projet nous a également permis de se rendre compte sur un exemple concret de ce que c'est que de traiter un grand nombre de données et de comment les comprendre, les analyser, et enfin les interpréter. Nous avons dans un premier temps établi une étude préalable pour comparer différentes méthodes ; nous avons donc mis en place une étude des méthodes, une étude des classes ainsi qu'une expérimentation de sorte à voir quelle méthode était la plus convenable. Dans la seconde partie, nous avons établi une analyse dans laquelle nous avons réalisé un examen des données, un pré-traitement des données, ainsi que différentes analyses de données dans le but de répondre au problème demandé.

Ce projet nous sera donc grandement utile dans nos études d'ingénieur, et plus particulièrement dans le domaine de la data science car nous avons désormais une bonne compréhension des différentes méthodes vu en cours, nous pourrons donc appliquer ces dernières lors de notre travail futur.

Sources:

Spectral Clustering. Foundation and Application | by William Fleshman | Towards Data Science

DBSCAN Clustering — Explained. Detailed theorotical explanation and… | by Soner Yıldırım | Towards Data Science

Microsoft Word - 12.docx (iop.org)