

**NOM DU DOSSIER**

**Sous titre**

Date

Service

**RAPPORT DE TP**

**Oriane BERRY, Clémence LEMEILLEUR**

**Promo 56, Année 2022/2023 – 5SDBD-B1**

*« TP Apprentissage Non Supervisé »*

S1 2023

Encadrant : M.Siala

**RAPPORT DE TP**

Oriane Berry, Clémence LEMEILLEUR

Promo 56, Année 2022/2023 – 5SDBD-B1

*“TP Apprentissage Non Supervisé*”

S1 2023

Encadrant: M.SIALA

Table des matières

[Table des matières 3](#_Toc118623503)

[I- Le Clustering 2](#_Toc118623504)

[1. Clustering k-Means et k- Medoids 2](#_Toc118623505)

[2. Clustering agglomératif 4](#_Toc118623506)

[3. Clustering DBSCAN et HDBSCAN 4](#_Toc118623507)

[II- Comparaison des méthodes de clustering 4](#_Toc118623508)

[Table des annexes 6](#_Toc118623509)

***Introduction :***

Dans ce TP nous avons comparé les différents algorithmes de clustering avec plusieurs méthodes qu’elles soient fournies par des outils ou externes. Nous utilisons pour cela des jeux de données en 2 dimensions afin d’avoir une meilleure visualisation de ces dernières.

Tout d’abord nous allons brièvement rappeler :

Qu’est-ce que le clustering ?

Le clustering est une méthode d’apprentissage automatique qui à pour but de regrouper des points de données en fonction de leur similarité ou de leurs distances entre eux. Cette méthode d’apprentissage non supervisée est une technique populaire d’analyse statistique des données.

En prenant un ensemble donné de points, vous pouvez utiliser différents algorithmes de classification pour les regrouper dans des groupes spécifiques. Ils comporteront alors, dans un même groupe de points, des propriétés similaires tout en ayant des caractéristiques les différenciants de ceux des autres groupes.

Le but de la clusterisation est de donner un sens aux données, les faire parler afin de tirer un maximum d’informations qui, à la base, ne sont pas classées.

Cette méthode est utilisée avec des données de secteurs bien différents comme la santé, ou encore les études de profils.

Les manières de classer ses groupes varient en fonction de la forme des données. En effet, le les propriétés de chaque modèle vont influer sur le résultat. Parmi les principaux modèles on peut trouver ceux de groupe, centralisé, graphique, densité, distribué mais encore connectivité.

Nous allons maintenant étudier certaines méthodes afin de constater leurs avantages, inconvénients, et les comparer entre elles.

# Le Clustering

## Clustering k-Means et k- Medoids

Cette méthode est une des plus simple puisque le seul paramètre à choisir est k : le nombre de classes souhaité.

Intérêts de la méthode k-Means

Tout d’abord voici un petit point sur les métriques d’évaluation recommandés :

* Coefficient de silhouette : Différence entre la distance moyenne avec les points du même groupe que lui (cohésion) et la distance moyenne avec les points des autres groupes voisins (séparation). Le coefficient de silhouette varie entre -1 (pire classification) et 1 (meilleure classification).

🡪 Est plus adapté aux données « regroupées » de manière identifiable.

* Indice de Davies-Bouldin : C'est la moyenne du rapport maximal entre la distance d'un point au centre de son groupe et la distance entre deux centres de groupes. Il varie entre 0 (meilleure classification) et +∞ (pire classification).

🡪 Est plus adapté aux données reparties de manière homogène.

* Indice de Calinski-Harabasz : C'est le rapport entre la variance inter-groupes et la variance intra-groupe. Il varie entre 0 (pire classification) et +∞ (meilleure classification)

Nous avons choisi dans notre cas la deuxième méthode.

Nous avons ensuite choisi nos jeux de données avec ceux qui nous semblaient avoir des données bien séparées qui permette l’identification aisée des clusters par la méthode k-Means.

Lors de l’application itérative de la méthode des k-Means sur 2 jeux de données, voici les résultats que nous avons eu au niveau des métriques d’évaluation :

**Paramètre : JDD « aggregation.arff»**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | k=2 | k=3 | k=4 | k=5 | k=6 | k=7 | k=8 | k=9 | k=10 |
| Indice de DB | **0,8964** | 0,6669 | 0,6116 | 0,7042 | 0,6400 | 0,7368 | 0,8270 | 0,7760 | 0,7252 |
| Indice silhouette | 0,4558 | 0,5234 | 0,5236 | 0,4987 | **0,6509** | 0,4804 | 0,4518 | 0,4621 | 0,4793 |
| Temps de calcul | 27,76 ms |  |  |  | 37,31 |  |  |  |  |
| Nb iter | 14 |  |  |  | 5 |  |  |  |  |

Après avoir remarqué que l’indice de DB n’est pas adapté à ce jeu de données. Nous avons testé sur l’indice de silhouette.

Nous allons faire de même avec un jeu de données du même aspect.

**Paramètre : JDD « atom.arff»**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | k=2 | k=3 | k=4 | k=5 | k=6 | k=7 | k=8 | k=9 | k=10 |
| Indice de DB | 1,0942 | 0,9038 | 0,7815 | 0,6566 | 0,6398 | 0,6553 | 0,6978 | 0,0722 | 0,7653 |
| Indice de silhouette | 0,4888 | 0,5358 | 0,5945 | 0,6352 | 0,6464 | **0,6499** | 0,6440 | 0,6372 | 0,5625 |
| Temps de calcul | 23,68ms |  |  |  |  | 39,46 ms |  |  |  |
| Nb iter | 4 |  |  |  |  | 6 |  |  |  |

Limites de la méthode k-Means

**Paramètre : JDD « banana.arff»**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | k=2 | k=3 | k=4 | k=5 | k=6 | k=7 | k=8 | k=9 | k=10 |
| Indice de DB | 0,8930 | 0,8518 | 0,8083 | 0,6698 | 0,6567 | 0,6223 | 0,6141 | 0,5999 | 0,5931 |
| Indice de silhouette | 0,4645 | 0,4544 | 0,4623 | 0,4761 | 0,5054 | 0,5013 | 0,5218 | 0,5214 | 0,5201 |
| Temps de calcul (ms) | 34,33 | 49,57 | 48,83 | 58,49 | 52,26 | 60,69 | 88,74 | 72,34 | 71,49 |
| Nb iter | 7 | 5 | 17 | 14 | 12 | 8 | 15 | 11 | 12 |

**Paramètre: JDD «birch-rg1.arff»**

On remarque que l’exécution de ce code prend énormément de temps, et nous donne un résultat non satisfaisant. En effet il s’agit juste d’un amas de données coupées de manière équitables aléatoirement.

On remarque donc que la méthode k-Means comporte ses limites sur certaines formes de jeux de données.

Méthode k-Medoids

**Paramètre : JDD « aggregation.arff»**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | k=2 | k=3 | k=4 | k=5 | k=6 | k=7 | k=8 | k=9 | k=10 |
| FasterPAM | 6672 | 4692 | 3827 | 3289 | 2929 | 2724 | 2556 | 2361 | 2233 |
| Indice de DB | 0,9208 | 0,7233 | 0,7429 | 0,8396 | 0,7446 | 0,7883 | 0,8570 | 0,8122 | 0,9022 |
| Indice de silhouette | 0,4546 | 0,5144 | 0,4930 | 0,4564 | 0,4828 | 0,4759 | 0,4260 | 0,4457 | 0,4028 |
| Temps de calcul (ms) | 17,01 | 8,07 | 8,75 | 10,62 | 10,79 | 12,27 | 8,89 | 16,66 | 13,65 |
| Nb iter | 3 | 3 | 3 | 4 | 4 | 5 | 3 | 8 | 6 |

On remarque que pour k=3, on a un des pertes FasterPam pas trop élevées et un bon indice de silhouette. Le nombre de clusters le plus pertinent à choisir semble donc être 3.

## Clustering agglomératif

Intérêts de la méthode

Un des inconvénients des méthodes vues jusqu’ici est qu’il nous faut spécifier nous même le nombre de clusters souhaités : k.

Un des avantages de cette nouvelle méthode de clustering est que nous n’avons plus à spécifié ce paramètre.

Le clustering agglomératif va reconstruire les relations entre les clusters de manière arborescente. Une suite de méthodes fusionne au fur et à mesure les deux clusters les plus proches et ainsi de suite pour obtenir un dendrogramme, facile à observer.

Nous avons donc choisi 2 jeux de données de la sorte à ce qu’ils puissent identifier au mieux les clusters :

**Paramètre : JDD «  » en single linkage**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seuil : | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

**Paramètre : JDD «  » en single linkage**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seuil : | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

Nous remarquons donc que pour c’est 2 jeux de données, le seuil de distance optimal est de :

Nous avons ensuite voulu voir l’incidence des différentes manières de combiner les clusters (‘single’ comme précédemment, average, complete, ward linkage) et voici ce que nous avons pu observer :

Nous pouvons donc en déduire que :

Limites de la méthode

Maintenant que nous avons mieux compris comment fonctionnait cette méthode, nous l’avons appliqué à 2 jeux de données sur lesquels elle ne devrait pas fonctionner et voici ce que l’on a obtenu :

**Paramètre : JDD «  » en single linkage**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seuil : | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

**Paramètre : JDD «  » en single linkage**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seuil : | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

Nous avons fait le choix de le faire uniquement en single linkage pour faciliter les comparaisons avec les résultats au-dessus.

Nous pouvons donc voir que dû au …. En effet le clustering agglomératif n’est pas adapté à ce genre de jeu de données.

## Clustering DBSCAN et HDBSCAN

Intérêts de la méthode DBSCAN

Les méthodes DBSCAN et HDBSCAN sont des méthodes de densité. Les classes représentent donc des zones de fortes densités de points en comparaison des autres zones.

Un des avantages de la méthode DBSCAN est qu’elle est capable de détecter ce qu’on pourrait qualifier de « bruit », les points trop éloignés et faussant le reste des données.

Nous avons donc choisi 2 jeux de données de la sorte à ce qu’ils puissent identifier au mieux les clusters. Nous avons laissé la distance en valeur par défaut et choisi des valeurs aléatoires pour les paramètres min-sample et eps :

Min-sample=

Eps=

**Paramètre : JDD «  »**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

**Paramètre : JDD «  »**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

Nous avons ensuite itéré le processus afin de déterminer les paramètres min sample et eps :

Nous pouvons donc dire que les paramètres optimaux sont :

Min-sample=

Eps=

Limites de la méthode DBSCAN

Maintenant que nous avons mieux compris comment fonctionnait cette méthode, nous l’avons appliqué à 2 jeux de données sur lesquels elle ne devrait pas fonctionner et voici ce que l’on a obtenu :

**Paramètre : JDD «  »**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

**Paramètre : JDD «  »**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

Nous pouvons donc voir que dû au …. En effet le clustering agglomératif n’est pas adapté à ce genre de jeu de données.

Comparaison entre DBSCAN et HDBSCAN

Nous allons rapidement faire un petit point sur la comparaison entre ces 2 méthodes avant d’élargir notre vision en comparant, de manière globale, toutes les méthodes de clustering entre elles.

La méthode HDBSCAN est connue pour être insensible à la variabilité de densité dans les données.

Nous avons repris les JDD précédent en les traitants avec HDBSCAN .

**Paramètre : JDD «  »**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |

**Paramètre : JDD «  »**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| FasterPAM |  |  |  |  |  |  |  |  |  |
| Indice de DB |  |  |  |  |  |  |  |  |  |
| Indice de silhouette |  |  |  |  |  |  |  |  |  |
| Temps de calcul (ms) |  |  |  |  |  |  |  |  |  |
| Nb iter |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Les valeurs des paramètres qui ressortent ici sont :

Min-sample=

Eps=

On peut donc, suite à ses observations en déduire les avantages et inconvénients de chacune des méthodes :

De plus, au niveau des performance (temps de calcul) on remarque que

# Comparaison des méthodes de clustering

Après avoir étudié chacune des méthodes de clustering, en mettant en avant leurs avantages et inconvénients respectifs, nous allons maintenant les comparer entre elles.

Pour cela nous allons les tester sur un nouveau jeu de données fournies, s’apparentant à un cas que nous pourrions retrouver dans le cadre d’une étude réelle. Nous avons utilisé pour cela un serveur de calcul.

Nous illustrerons toutes nos observations à l’aide de visualisation pour rendre cette comparaison plus parlante.

A FAIRE AVEC DES ILLUSTRATIONS

On peut faire un tableau avec une colonne méthode, une dataset, une valeur de k, une ou on prend un indice, et une ou on prend le temps total d’exécution.

**Bilan de comparaison des méthodes de clustering :**

***Conclusion :***

Pour conclure nous pouvons dire que ce TP nous a permis de prendre connaissance et de nous familiariser avec différentes méthodes de clustering et de visualiser leurs effets sur des jeux de données en 2 dimensions.

Nous avons pu expérimenter et mieux comprendre les principes de base du clustering de manière globale dans un premier temps, puis plus approfondis via différentes méthodes en comparant leurs avantages et inconvénients.

Cela nous permet donc de nous mettre une fois de plus dans le rôle de l’ingénieur qui est d’utiliser ses connaissances et de les appliquer à des cas réels. D’autant plus que le clustering peut être appliqué à énormément de cas d’usages dans de nombreux domaines et, selon nous, est chaque jour un peu plus dans l’actualité.

# Table des annexes

1. Annexe 1 : Lien GitHub A

**Annexe 1 :** Lien GitHub : <https://github.com/Enario4/ApprentissageNonSupervise>

