



ANALYSE ET TRAITEMENT DE DONNEES



Compte-rendu de TP

SUBMITTED BY

Yacine BENCHEHIDA 5ISS-A2
Hugo LEBELGUET 5ISS-A2
Sami BEYAH MSIoT-A2

UNDER THE GUIDANCE OF

Prof. **Marie-José Huguet**

Academic Year: 2021-2022
January 11, 2021

Introduction

L'objectif de ce compte-rendu est de mettre en œuvre et comparer différents algorithmes de clustering. Pour ce faire, nous nous sommes basés sur les méthodes fournies par le package python *scikit-learn*.

Vous pouvez retrouver notre code sur le dépôt GitHub suivant :

<https://github.com/sami-mkb/BigData>

Table des matières

| | |
|--|-----------|
| Introduction | 0 |
| I- Comparaison des méthodes de clustering | 3 |
| 1. Clustering k-means | 3 |
| 2. Clustering agglomératif | 6 |
| 3. Clustering DBSCAN | 7 |
| 2. Analyse comparative sur les données fournies | 9 |
| 2.1. Visualisation | 9 |
| 2.2. Métriques | 10 |
| 2.3. k-Means | 11 |
| 2.3. DBSCAN | 11 |
| 2.3. Clustering agglomératif | 12 |
| Conclusion..... | 13 |

Table des illustrations

| | |
|---|----|
| FIGURE 1 : 2D-4C-NO4 | 3 |
| FIGURE 2 : BLOBS | 3 |
| FIGURE 3 : LONG1 | 3 |
| FIGURE 4 : SMILE2 | 3 |
| FIGURE 5 : K-MEANS APPLIQUÉ À 2D-4C-NO4 | 4 |
| FIGURE 6 : K-MEANS APPLIQUE A BLOBS | 4 |
| FIGURE 7 : K-MEANS APPLIQUE A SMILE2 | 4 |
| FIGURE 8 : K-MEANS APPLIQUE A LONG1 | 4 |
| FIGURE 9 : APPLICATION DES METRIQUES SUR LE DATASET DIAMOND9 | 5 |
| FIGURE 10 : COEFFICIENT DE SILHOUETTE / INDICE DE DAVIES..... | 7 |
| FIGURE 11: DBSCAN APPLIQUE A SMILE2 (DISTANCE = 5, MIN_PTS = 0.5) | 8 |
| FIGURE 12 : DBSCAN APPLIQUE A BLOBS (DISTANCE = 0.35 MIN_PTS = 14) | 8 |
| FIGURE 13: D64 | 9 |
| FIGURE 14 : D32.TXT | 9 |
| FIGURE 15: N1 | 9 |
| FIGURE 16: N2 | 9 |
| FIGURE 17: K-MEANS APPLIQUE A D32..... | 11 |
| FIGURE 18: K-MEANS APPLIQUE A D64..... | 11 |
| FIGURE 19: DBSCAN APPLIQUE A D32..... | 12 |

Points forts et points faibles identifiés pour les différentes méthodes de clustering étudiées

I- Comparaison des méthodes de clustering

Dans un premier temps, nous avons réalisé une comparaison détaillée des méthodes de clustering suivantes : k-means, clustering agglomératif et DBSCAN. L'objectif étant d'identifier les points faibles et forts de chaque méthode. Pour ce faire, nous avons étudié, entre autres, les résultats et les performances de chaque méthode.

1. Clustering k-means

K-means est une méthode de clustering apparue dans les 50-60, c'est une méthode dite centroïde. L'algorithme nécessite de fixer préalablement le nombre de clusters et le nombre d'itérations, en déterminant dynamiquement chacun des centres de gravité des différents clusters, permet ainsi de dissocier les données en différents groupes. Tous les points du dataset sont inclus dans un cluster.

Pour étudier le fonctionnement de l'algorithme k-Means, nous avons commencé par sélectionner des jeux de données qui nous paraissent pertinents à étudier avec cette méthode. Nous avons sélectionné les quatre jeux de données suivants : *blobs*, *2d-4c-no4*, *smile2* et *long1*.

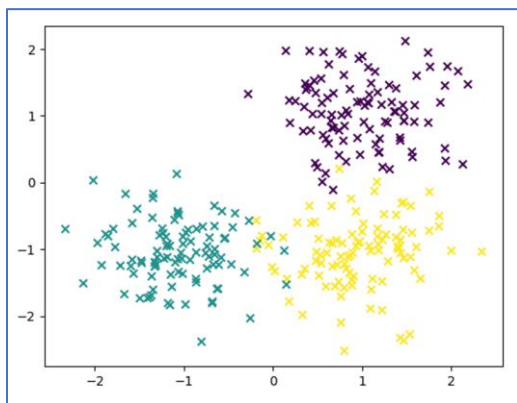


Figure 2 : blobs

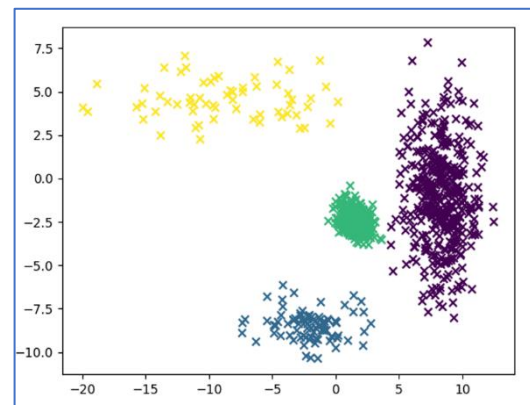


Figure 1 : 2d-4c-no4

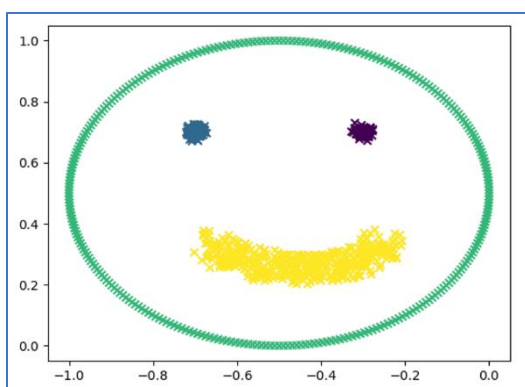


Figure 4 : Smile2

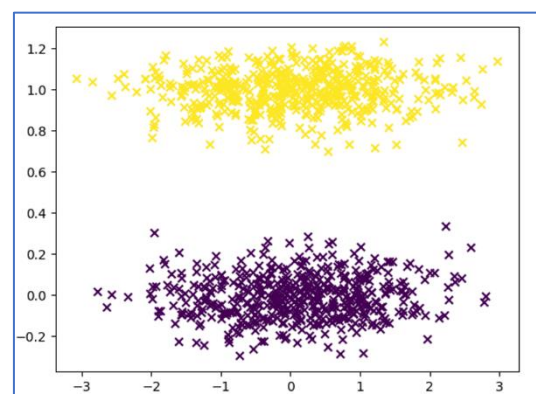


Figure 3: long1

D'abord, nous avons appliqué la méthode k-Means aux datasets blobs et 2d-4c-no4. Ces deux datasets ont l'avantage d'avoir un nombre de clusters finis, respectivement 3 et 4. Nous obtenons les résultats suivants :

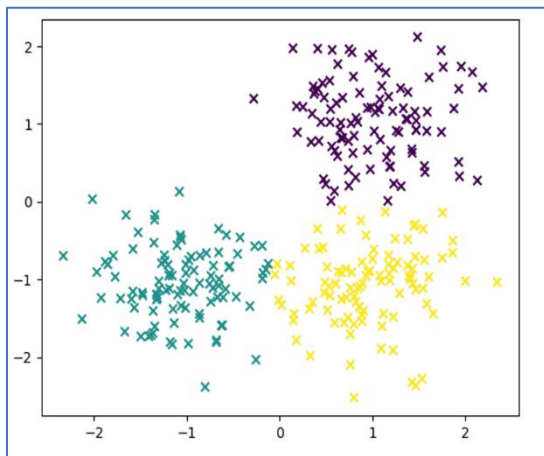


Figure 6 : K-means appliqué à blobs

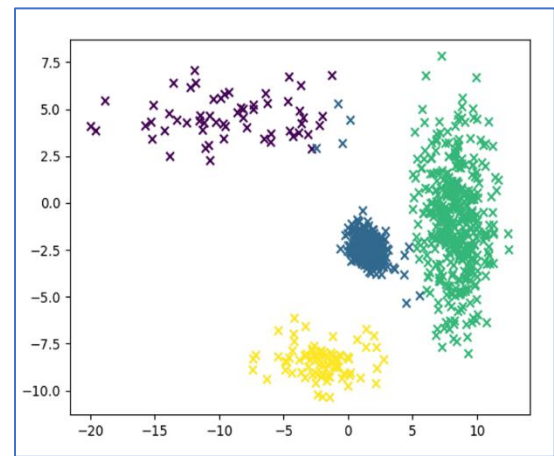


Figure 5 : K-means appliqué à 2d-4c-no4

Les résultats obtenus pour ces deux datasets sont plutôt impressionnants. La méthode K-Means arrive à détecter et classer les différents points très rapidement. Néanmoins, on peut remarquer que certains points ne sont pas attribués aux bons clusters dans les deux datasets. Ceci est principalement dû au fait que k-Means est très sensible à la densité. Si un point d'un cluster est proche d'un autre cluster plus dense, il peut lui être attribuer. Nous avons relevé le temps d'exécution pour ces deux datasets, il est très court car la méthode k-means est très simple à mettre en œuvre. Vous pouvez trouver le temps d'exécution pour les différents datasets ici : [BigData/partie 1/timing analysis/kmeans clustering](#)

KMeans

Temps d'execution Kmeans- 2d-4c-no4 [2] = 0.047443

Temps d'execution Kmeans- long1 [2] = 0.052730

Dans un second temps, nous avons appliqué la méthode k-means aux datasets long1 (Figure 3) et smile2 (Figure 4). Nous obtenons les résultats suivants :

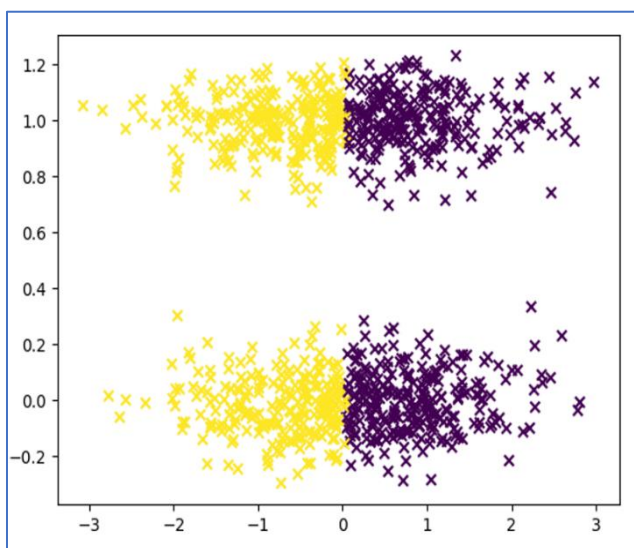


Figure 8 : K-Means appliqué à long1

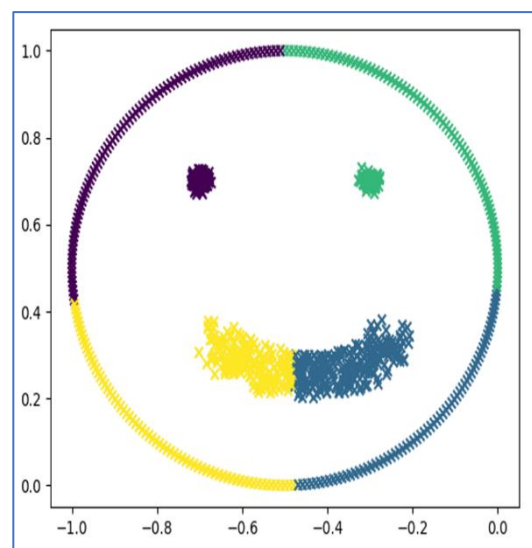


Figure 7 : K-Means appliqué à Smile2

Les résultats obtenus pour long1 et smile2 sont beaucoup plus décevants. On peut clairement observer sur les figure 7 et 8 que l'algorithme n'arrive pas à dissocier les différents clusters. K-means semble bloqué dans des optimums locaux. La méthode K-Means semble uniquement adapté aux formes convexes.

Afin d'approfondir notre évaluation de cette méthode, nous avons cherché à déterminer automatiquement le bon nombre de clusters. Jusqu'alors, le nombre de clusters à identifier été passé en paramètre de l'algorithme.

Pour ce faire, nous avons utiliser deux procédures dans k-means, l'indice de Davies Bouldin et le coefficient de silhouette.

- **L'indice de Davies Bouldin** est la moyenne du rapport maximal entre la distance d'un point au centre de son propre groupe et la distance entre deux centres d'autres groupes. Il doit par conséquent être minimisé, plus la valeur est proche de 0, plus les clusters sont homogènes et bien séparés.
- **Le coefficient de silhouette** est la moyenne du coefficient de silhouette pour tous les points. Le coefficient de silhouette d'un point est la différence entre la distance moyenne avec les points du même groupe que lui-même et la distance moyenne avec les points des autres groupes voisins. L'objectif est de le maximiser, d'être au plus proche de 1.

D'abord, nous avons implémenté une fonction itérative qui fait varier le nombre de cluster fixés dans la méthode k-Means de 2 à 12. Pour visualiser le coefficient de silhouette et l'indice de Davies Bouldin pour chaque valeur de cluster, nous avons réaliser le graphique suivant :

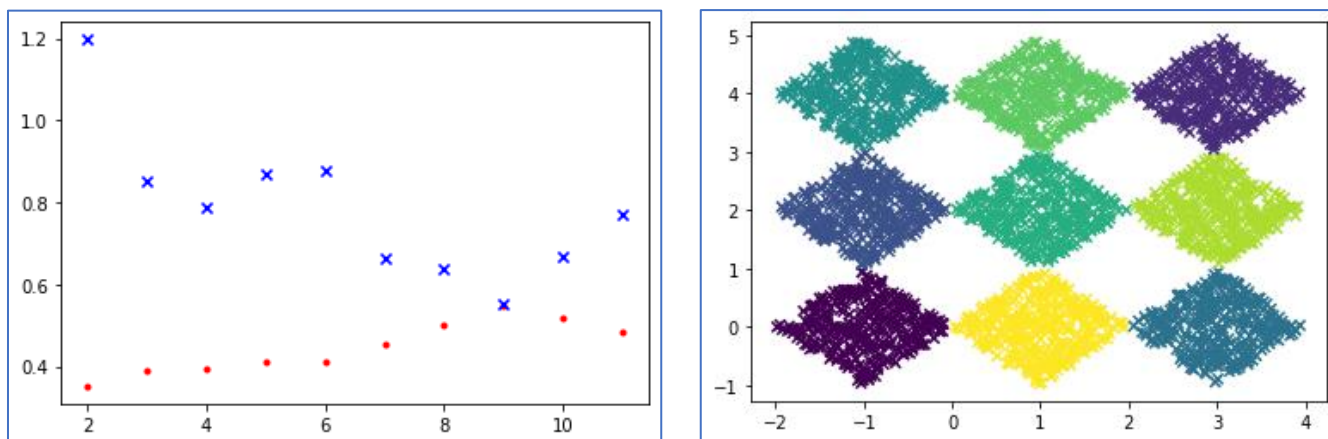


Figure 9 : Application des métriques sur le dataset Diamond9

Nous pouvons voir à gauche sur la figure 9, les résultats des deux métriques pour les différentes valeurs de clusters. En rouge, le coefficient de silhouette et en Bleu l'indice de Davies. Le coefficient de silhouette est maximal et l'indice de Davies est minimale à 9 clusters pour le dataset *Diamond9*. On peut donc vérifier le bon fonctionnement de ces métriques pour ce dataset.

La plus grande faiblesse de l'algorithme K-Means est qu'il faut passer le bon nombre de clusters en entrée de l'algorithme. Certes, on peut utiliser des métriques comme on a vu ci-dessus (figure 9), mais cela rester une approximation et les métriques ne sont pas toujours fiables.

2. Clustering agglomératif

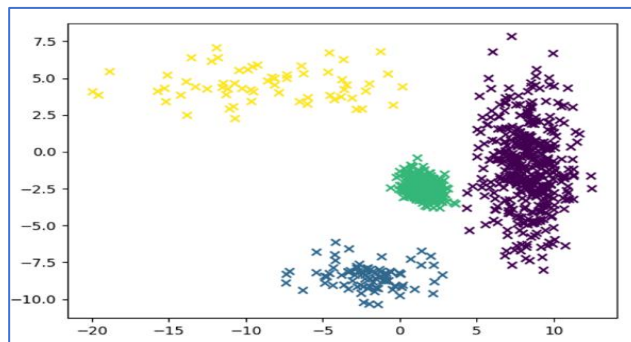
Il existe deux types de clustering dans la méthode hiérarchique : ascendant, dit agglomératif et descendant, dit divisif. On va s'intéresser au clustering agglomératif.

Dans la méthode agglomératif, on considère initialement chaque point comme un cluster. Ensuite, on réitère l'algorithme en fusionnant les points proches en fonction de la similarité. Le nombre de clusters finale doit être défini dans l'algorithme en tant que paramètre. Comme pour DBSCAN, la distance entre les points est un paramètre crucial. Aussi, les résultats peuvent significativement varier selon le calcul de similarité des clusters. Il existe différentes méthodes pour le calcul des similarités, nous avons implémentés les méthodes suivantes :

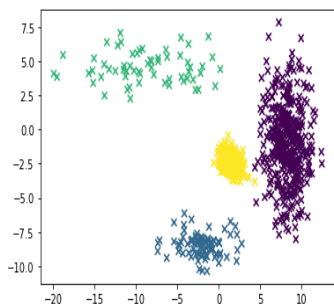
- Ward : l'augmentation de la variance intra-cluster est minimale.
- Complete linkage : minimise la distance maximale entre les observations de paires de clusters.
- Average linkage : minimise la moyenne des distances entre toutes les observations des paires de clusters.
- Single linkage : minimise la distance entre les observations les plus proches des paires de clusters.

Nous avons appliqué ces méthodes aux datasets suivants : 2d-4c-no4 et blobs. Le nombre de clusters a été fixé dans le l'algorithme.

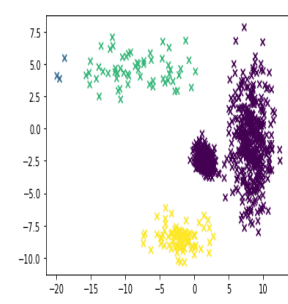
2d-4c-no4.arff



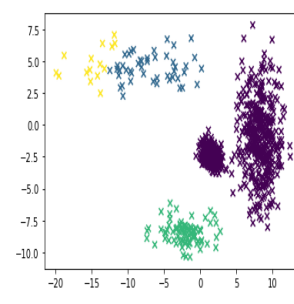
Méthode Ward



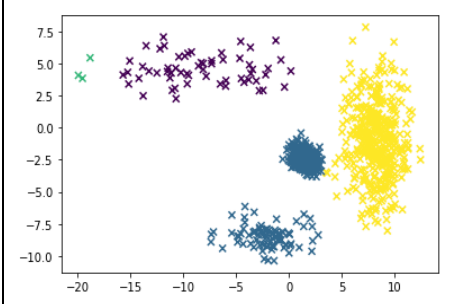
Méthode Single



Méthode complete



Méthode Average



On peut constater que la méthode la plus optimale pour obtenir le bon nombre de cluster est la méthode ward sur ce dataset. Par rapport au dataset d'origine, on arrive à obtenir des données très similaire. On peut clairement différencier les 4 clusters.

On peut vérifier cela en utilisant les métrique utilisés précédemment pour k-Means, le coefficient de silhouette et l'indice de Davies. En gardant ce linkage "ward", si nous faisons varier le nombre de clusters passé en paramètre, en se fiant aux deux critères d'évaluation choisis, nous obtenons bien un nombre optimal de 4 clusters. On peut voir cela sur la figure 10 :

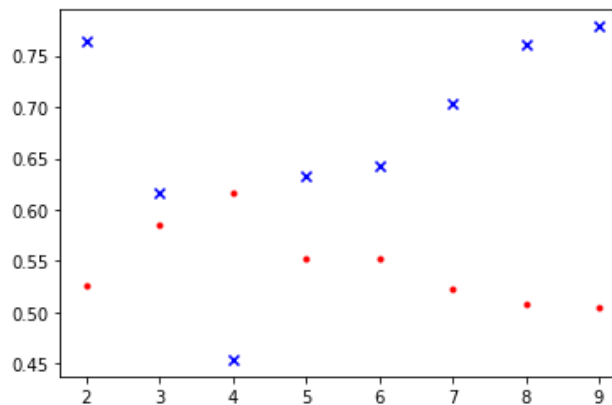


Figure 10 : Coefficient de silhouette / indice de Davies

En termes de temps d'exécution, il est en général plus long pour la méthode agglomératifs par rapport aux autres méthodes qu'on a étudié. La complexité de l'algorithme est plus grande. Elle dépend du calcul de similarité.

3. Clustering DBSCAN

L'algorithme DBSCAN considère les clusters comme des zones de haute densité séparées par des zones de faible densité. En raison de cette vision générique, les clusters trouvés par DBSCAN peuvent avoir n'importe quelle forme, contrairement aux k-means qui supposent que les clusters ont une forme convexe. Concrètement, DBSCAN fonctionne par voisinage. Tous les point atteignables, c'est-à-dire dans un certain rayon du point observé, appartiennent au même cluster. Ainsi, on construit nos clusters en visitant chaque point et en observant son voisinage.

Pour ce faire, l'algorithme comporte deux paramètres, `mini_pts` et `distance`, qui définissent formellement ce que nous entendons par densité. Des `mini_pts` plus élevés ou des `distance` plus faibles indiquent une plus grande densité nécessaire pour former un cluster.

Nous avons appliqué la méthode DBSCAN sur une forme non-convexe : `smile2`. Nous avons obtenu les résultats suivants :



Figure 11: DBSCAN appliqué à smile2 (distance = 5, min_pts = 0.5)

Nous pouvons clairement constater sur la figure 11 que en paramétrant correctement les deux paramètres de DBSCAN, nous pouvons réaliser un clustering sur les datasets aux forme non-convexes comme smile2 ci-dessus. De plus, nous pouvons constater que certains points ne sont assimilés à aucun cluster contrairement à dans la méthode k-means.

Les points non-assimilés à un cluster sont considérés comme du bruit. Nous pouvons visualiser cela, en violet, ci-dessous en appliquant DBSCAN au dataset blobs. On obtiens de meilleurs résultats qu'avec le méthode k-means.

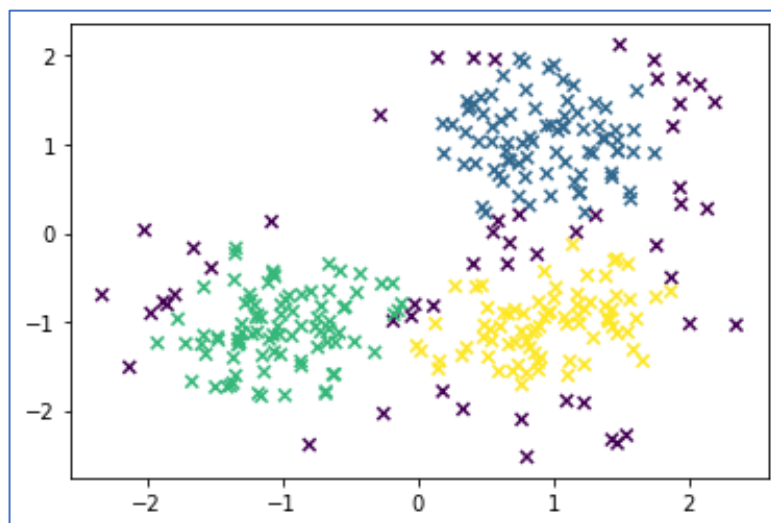


Figure 12 : DBSCAN appliqué à blobs (distance = 0.35 min_pts = 14)

Le point fort de DBSCAN par rapport à K-Means est qu'on a pas besoin de définir le nombre de clusters en entrée de l'algorithme. DBSCAN visite chaque point pour lui attribuer un cluster par la notion de voisinage, cela n'implique en rien la connaissance du nombre de clusters recherché. Aussi, on donne un nombre de voisin nécessaire pour appartenir au cluster, ainsi on élimine le bruit.

2. Analyse comparative sur les données fournies

Dans cette partie, nous avons appliqués les différentes méthodes de clustering sur de nouveaux jeux de données. Les données correspondent à des données artificielles de petites dimensions et à un jeu de données "réel" ayant un nombre d'attributs plus important.

2.1. Visualisation

Dans un premier temps, parmi les datasets qui nous ont été fournis, nous avons étudiés les datasets suivants :

- d32.txt
- d64.txt
- n1.txt
- n2.txt
- w2.txt

On peut voir ci-dessous la représentation de ces datasets :

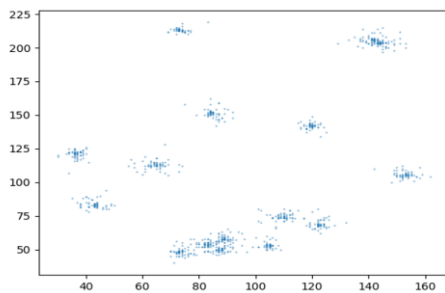


Figure 14 : d32.txt

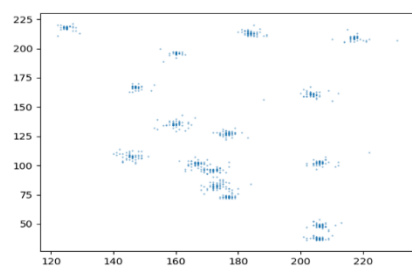


Figure 13: d64

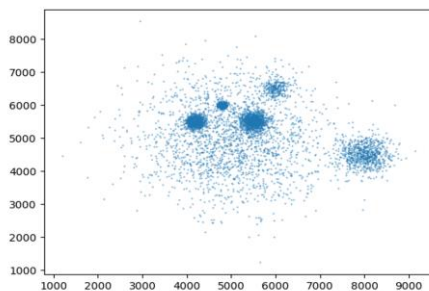


Figure 16: n2

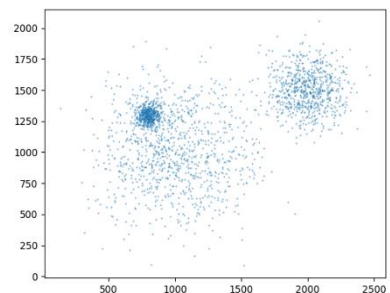
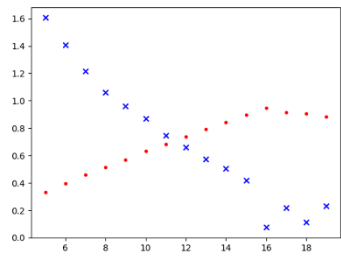
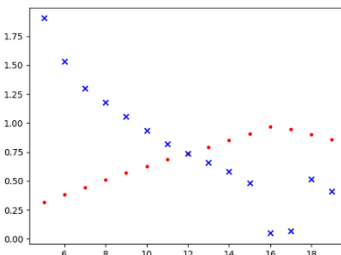
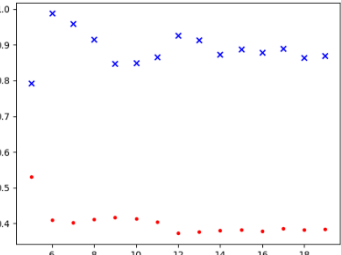
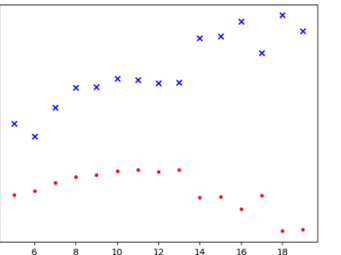


Figure 15: n1

2.2. Mètriques

Nous avons appliqué les métriques que nous avons vues précédemment : coefficient de silhouette et indices de Davies à ces datasets. L'objectif étant d'observer si on arrive à obtenir le bon nombre de clusters. Au vu de la forme des clusters et de la rapidité d'exécution de k-means, nous avons opté pour cette méthode en premier lieu. Nous avons appliqués ces métriques pour k-Means.

| Dataset | Métriques | Cohérences |
|---------|---|--|
| D32 | <p><u>Nombre de clusters détectés : 16</u></p>  | <p>Pour le dataset D32, on visualise le même nombre de clusters. Les données sont cohérentes.</p> |
| D64 | <p><u>Nombre de clusters détectés : 16</u></p>  | <p>Pour le dataset D64, on visualise le même nombre de clusters. Les données sont cohérentes.</p> |
| N1 | <p><u>Nombre de clusters détectés : 5</u></p>  | <p>En visualisation le dataset, on ne compte pas le même nombre de clusters. Les métriques ne sont pas cohérentes.</p> |
| N2 | <p><u>Nombre de clusters détectés :</u></p>  | <p>En visualisation le dataset, on ne compte pas le même nombre de clusters. Les métriques ne sont pas cohérentes.</p> |

2.3. k-Means

On a appliqué k-Means à ces datasets. Nous avons obtenu les résultats suivants.

Pour d32, les clusters distants des autres clusters et de forme convexes ont bien été identifiés. Les clusters proches de clusters denses n'ont pas été distingués.

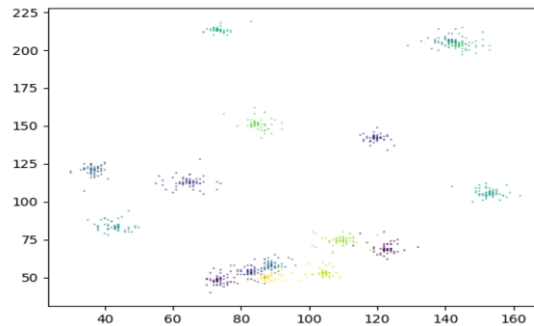


Figure 17: k-means appliqué à d32

Pour d64, nous obtenons de meilleurs résultats. Les clusters sont bien distincts, de forme convexes, donc bien déterminés par k-means. Ce résultat est cohérent.

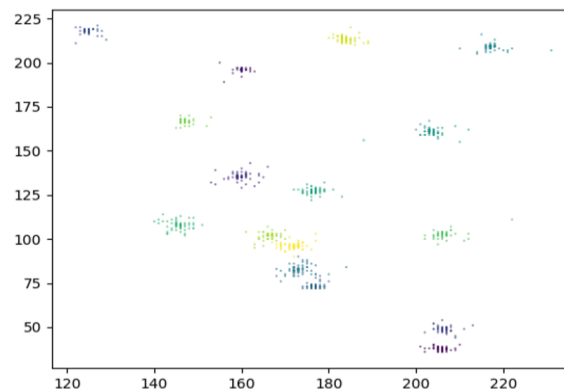


Figure 18: k-means appliqué à d64

Nous n'avons pas appliqué k-means pour les datasets N1 et N2 puisque ce sont des formes non-convexes donc particulièrement peu adapté à cette méthode. De plus, nous n'avons pas pu déterminer le bon nombre de clusters via les métriques

2.3. DBSCAN

Nous avons appliqué DBSCAN pour les mêmes datasets. Nous n'avons pas à fixer le nombre de clusters mais nous devons faire varier la distance et le nombre de points afin d'optimiser le clustering.

Pour d32, nous avons obtenus la figure suivantes.

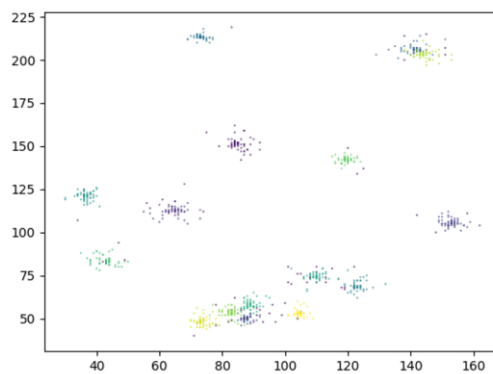


Figure 19: DBSCAN appliqué à d32

Nous avons fixé les valeurs suivantes :

- Distance : 0.5
- mini_pts : 14

Nous arrivons à obtenir des résultats similaires que pour k-Means. DBSCAN arrive à distinguer les différents clusters. Néanmoins, certains points ne sont pas affectés à des clusters. L'algorithme considère qu'ils sont du bruit.

2.3. Clustering agglomératif

Nous avons appliqué le clustering agglomératifs aux datasets suivants : d32 et n2. Le nombre de clusters est 15 selon les métriques et notre visualisation.

| Méthodes | Ward | Average | Complete | Single |
|----------|------|---------|----------|--------|
| Dataset | | | | |
| D32 | | | | |
| N2 | | | | |

Conclusion

Durant ces TP, nous avons pu découvrir en détails le fonctionnement des méthodes de clustering communément utilisés dans le domaine des Big Data et de l'intelligence artificielle.

Nous n'avons pas de connaissances préalables sur ce domaine donc la prise en main a été difficile. Cependant, nous avons pu mettre en œuvre différentes méthodes et surtout nous avons pu prendre en main la librairie scikit-learn. Cette dernière est très puissante et nous sera très utile dans d'éventuelles projets sur l'analyse et le traitement de données.