CLUSTERING
BENAZZOUZ Abir - HEBRAS Grégoire - 5ISS

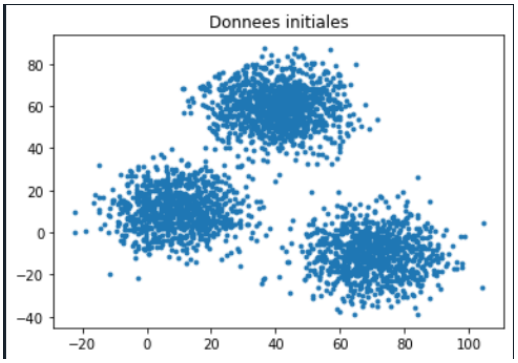
Dépot Git : https://github.com/abirbenazzouz/tp_clustering

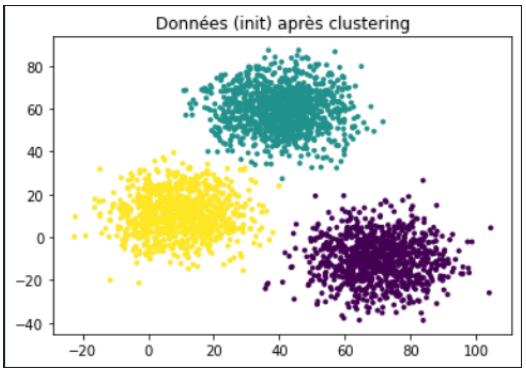
Clustering K-means:

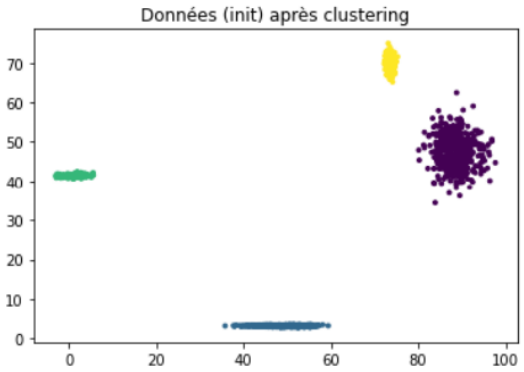
Cette méthode de clustering fait appel à l'algorithme K-means, qui est un algorithme non supervisé de clustering non hiérarchique. Il permet de trouver K centres de gravité (K clusters), mais ce nombre K doit être fixé au préalable.
En suivant cette méthode, il y a exclusivité d'appartenance : chaque observation ne peut appartenir qu'à un seul cluster, jamais plus. Cependant plusieurs clusterings sont possibles pour un même jeu de données. Il convient donc de choisir K afin qu'il mette en lumière des patterns le plus pertinemment possible.

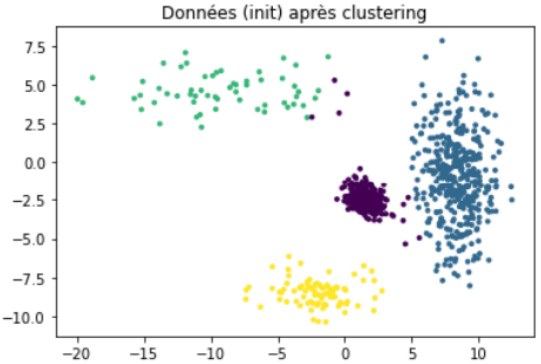
Pour créer des regroupements entre les données, cette méthode mesure la distance entre les points. Il existe plusieurs manières de définir la distance, il convient donc de choisir une définition de la distance la plus adaptée possible aux données puisque cette distance constitue la référence pour déterminer la dissimilarité entre deux points : plus elle est petite plus ils sont similaires selon le critère choisi.

Quelques exemples de clustering via K-means :



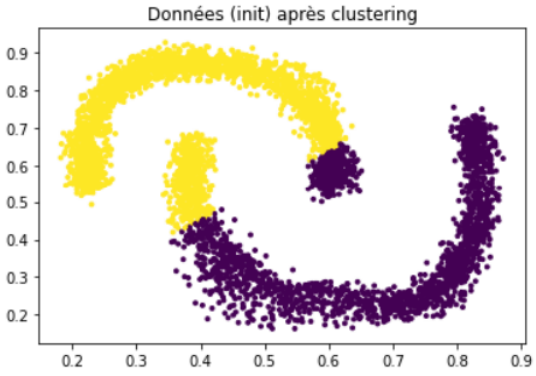


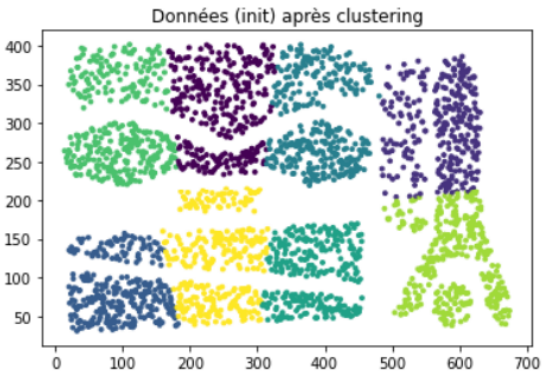




Cette méthode mesurant la distance entre les points, elle ne peut globalement s'appliquer qu'à des formes centrées autour d'un "barycentre" et fonctionne alors particulièrement bien

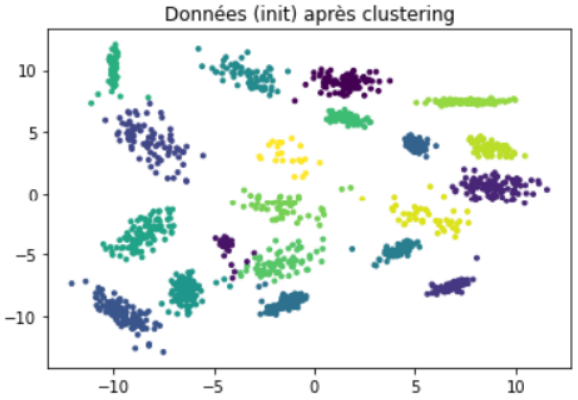
avec les clusters globulaires. Dès que les formes deviennent trop complexes, ou que le nombre de clusters à détecter devient trop grand, la méthode ne fonctionne plus aussi bien comme on peut le constater ci-dessous :

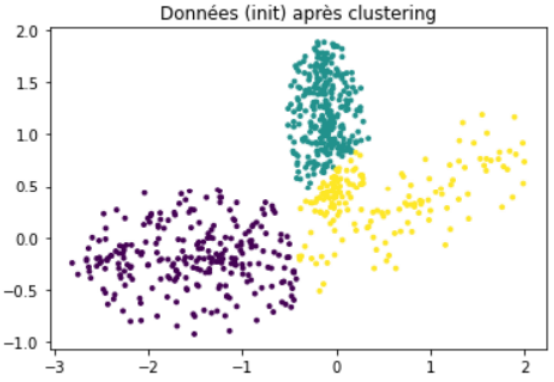




Dans les cas où les formes de clusters sont complexes, ou comme ci-dessus "entremêlées", la méthode k-means va vouloir regrouper les points proches plutôt que ceux appartenant au même groupement

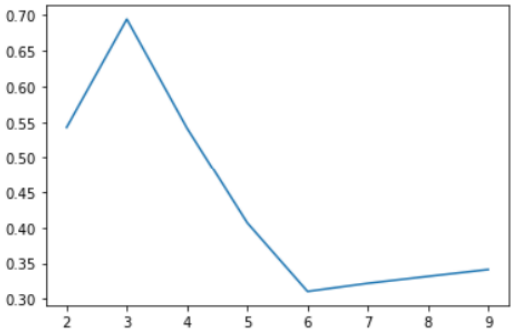
De la même manière si les clusters se confondent les uns avec les autres certains points peuvent être mal attribués :

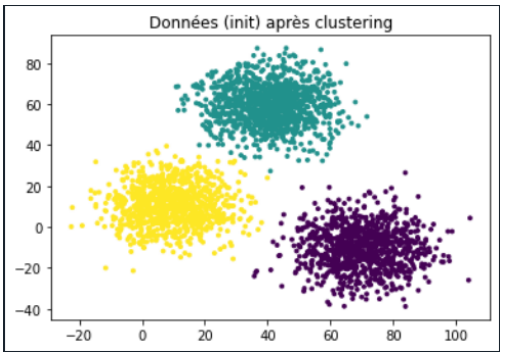




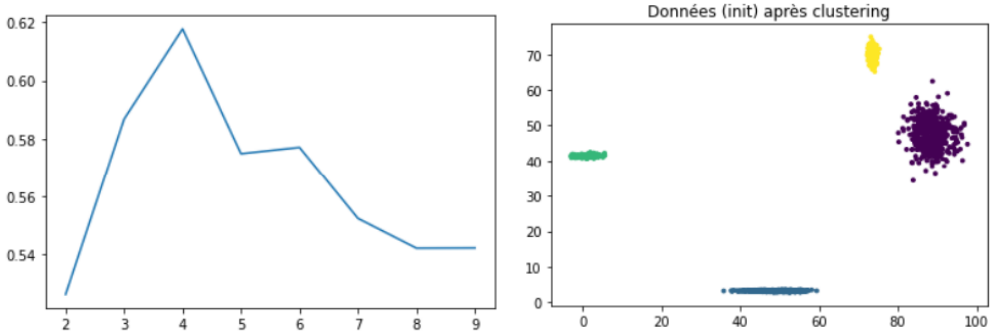
Pour les cas où le nombre de clusters est difficile à déterminer, on peut tester le coefficient de silhouette qui évalue la qualité du clustering. Dans nos exemples simples, on a pu le faire grâce à une simple boucle “for”. En traçant ce score de silhouette on peut voir quelle valeur de k est la plus adaptée.

k=3





k=4



Cependant,déterminer le k de cette manière avec la méthode K means prend trop de temps de calcul si le nombre de clusters est trop important, on ne peut donc utiliser cette méthode que pour un nombre de clusters raisonnable et pas trop élevé. C'est là une limite de cette méthode.

Points + :

- Simple à implémenter
- Rapide : Il convient donc également aux gros data sets
- Flexible : K-means d'adapte aux changements des données
- Produit des clusters proches
- Faible coût de calcul

Points - :

- Ne s'applique qu'à "certaines formes" de clusters : son efficacité dépend de la forme des clusters et il ne fonctionne pas avec des clusters de taille et de forme inhabituelles.
- Il faut choisir K au préalable : ce choix n'est pas toujours évident, particulièrement quand le jeu de données est grand et qu'on a peu d'informations dessus au départ.
- Les résultats peuvent varier entre différentes exécutions de l'algorithme : l'algorithme n'est pas toujours très fiable.

En définitive, K-means convient pour une utilisation très générale, pour un nombre de clusters peu élevé et des clusters de taille relativement uniformes et pour des formes plutôt sphériques.

Clustering agglomératif :

Le clustering agglomératif (=clustering ascendant) est une méthode hiérarchique. Cette méthode contrairement à K-means ne nous contraint pas à pré-spécifier le nombre de clusters. Cet algorithme va commencer par traiter chaque point comme un cluster singleton puis effectuer des regroupements successifs (et donc agglomérer, d'où le nom de la

méthode) de paires de clusters “proches” (en effectuant une mesure de similarité) jusqu’à ce que tous les clusters aient été fusionnés en un seul cluster contenant toutes les données. Cette méthode a pour principaux paramètres le nombre de clusters k (qui peut valoir un *integer* ou *None*) ; l’affinity (métrique utilisée pour calculer la connectivité(linkage)) : peut être la *distance euclidienne, l1, l2,* la *distance de manhattan,* la *distance cosinus* ou *precomputed* ; la mémoire (*memory*) : un string ou un objet ; la connectivity (array-like ou callable, None par défaut) ; le linkage ; la distance-treshold, etc.

Il existe plusieurs types de connectivités (linkage):

- **Ward**: minimise la somme des carrés des différences au sein de tous les clusters. Cette approche minimise la variance, et en ce point ressemble à la fonction objective de k-means mais avec une approche hiérarchique agglomérative
- **Maximum** / **complete linkage**: minimise la distance maximale entre les observations de paires de clusters
- **Average linkage:** minimise la moyenne entre les distances entre toutes les observations de paires de cluster
- **Single linkage:** minimise la distance entre les observations les plus proches de paires de clusters.

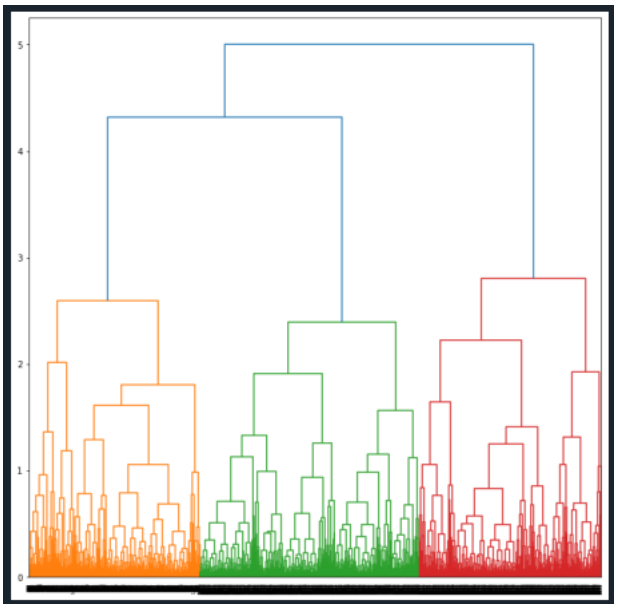
On remarque directement que le temps d’exécution de cette méthode est assez important.

```
Dendrogramme 'complete' données standardisées
-------------------------------------------------------------
Appel Aglo Clustering 'complete' pour une valeur de k fixée
nb clusters = 3 , runtime =  164.19 ms
Coefficient de silhouette :  0.6908368385665083
```

COMPARER A K MEANS

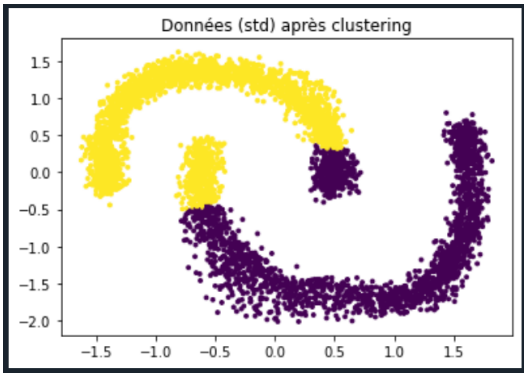
Cette méthode permet de visualiser le résultat sous forme de dendrogramme (arbre où les feuilles sont des échantillons et les nœuds des clusters).

Nous obtenons le dendrogramme suivant avec l’échantillon xclara (À VÉRIFIER):

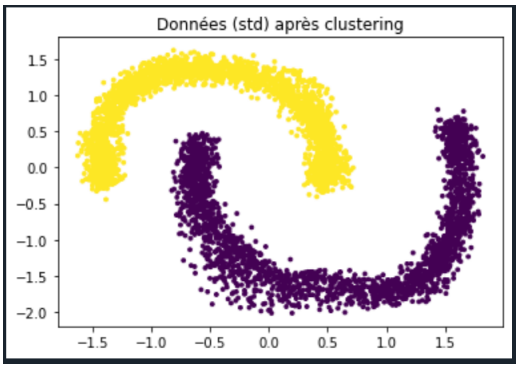


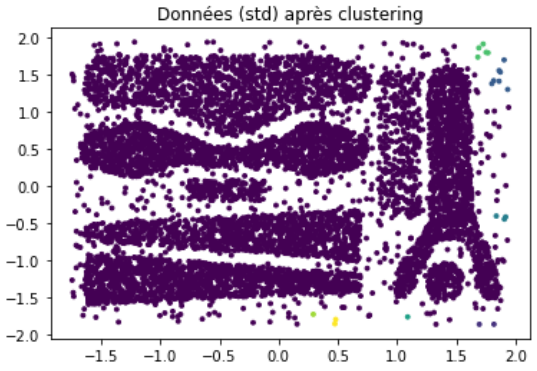
On peut ensuite également faire varier le type de linkage:

Avec le **complete linkage** nous obtenons les résultats suivants (respectivement avec les données xclara et banana) :



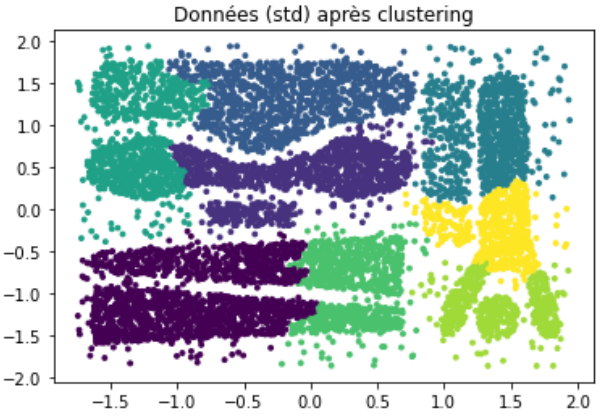
Avec le **single linkage** nous obtenons les résultats suivants (respectivement avec les données banana et cluto-t8-8k) :



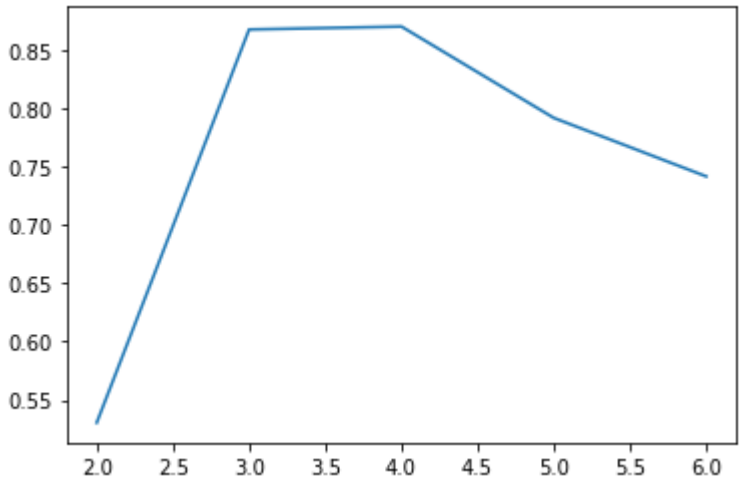


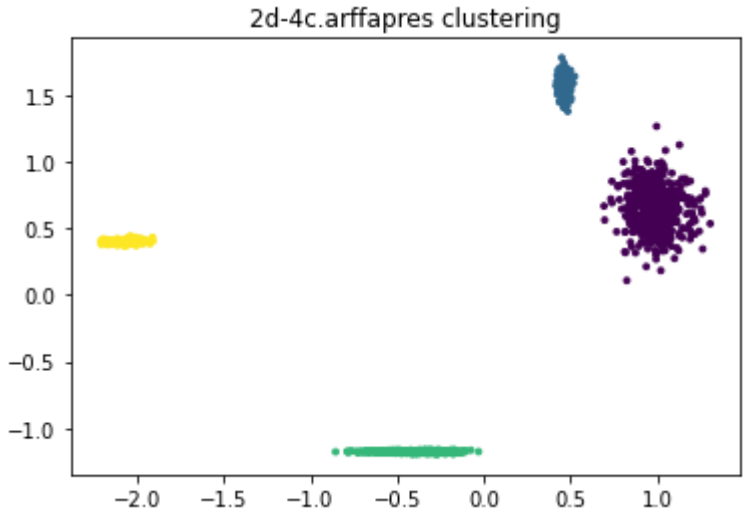
On peut en conclure que le single linkage peut bien fonctionner lorsque les clusters sont bien définis et sans “bruits” ou points isolés, mais dans les autres cas cette méthode de linkage a tendance à lier les clusters dès qu’un point s’en trouve trop proche. Par conséquent dans la plupart des cas, single linkage ne sera pas la meilleure méthode (sera la pire ?).

Enfin, avec le **ward linkage** nous obtenons comme résultat avec les datas cluto-t8-8k :



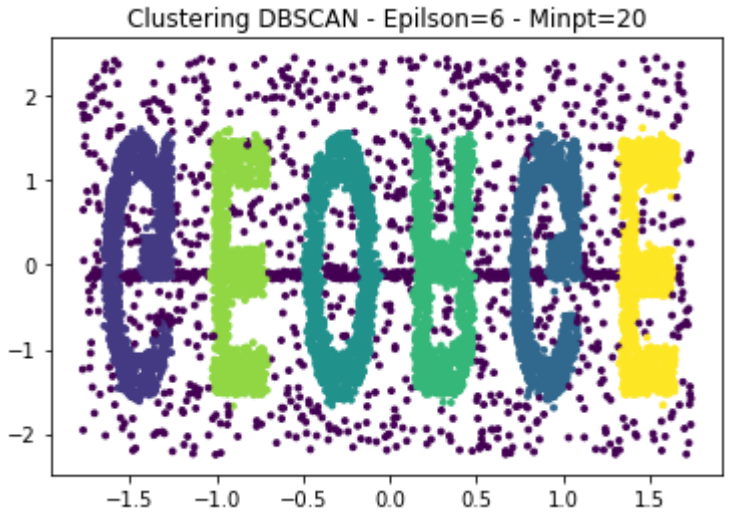
De la même manière que précédemment on essaye de déterminer le meilleure k en utilisant le coefficient de silhouette:
(On fait varier le k)





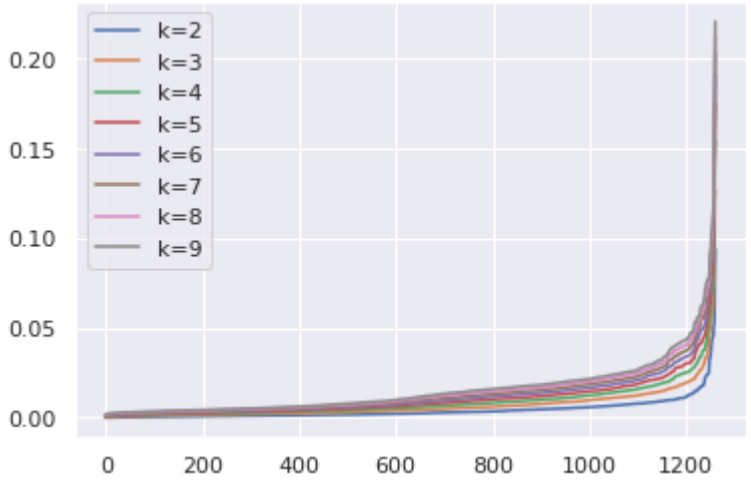
Clustering DBSCAN :

En appliquant les bon paramètres (distance=6, nd de point=20), dbscan est capable de trouver les clusters grâce à leur densité même dans des situations très complexes et d'écarter le bruit.
Plus grand est le dataset, plus grande devrait être la valeur de MinPts

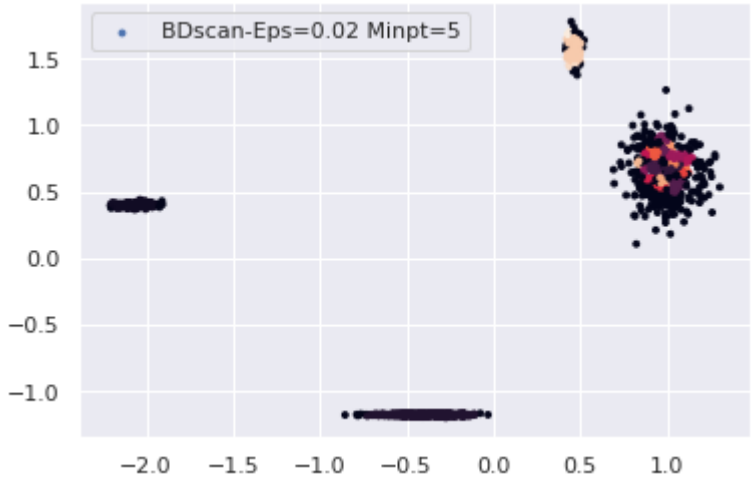



```
Affichage données standardisées
Estimated number of clusters: 6
Estimated number of noise points: 1122
runtime =  43.47 ms
```

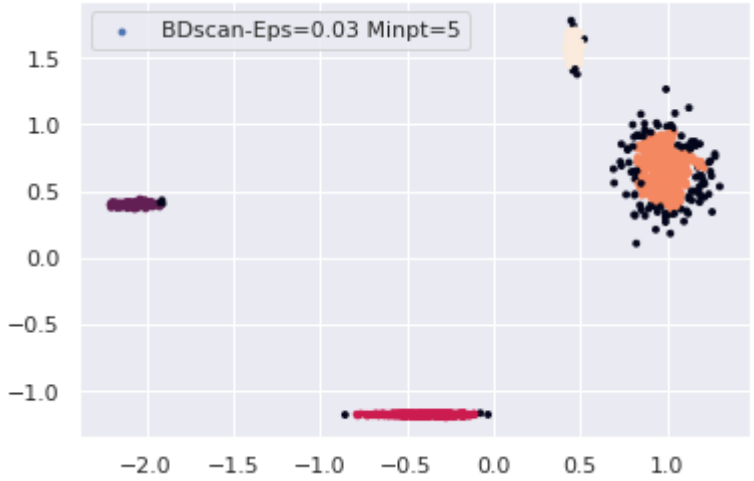
On trace la distance moyenne d'un point à ses plus proches voisin:



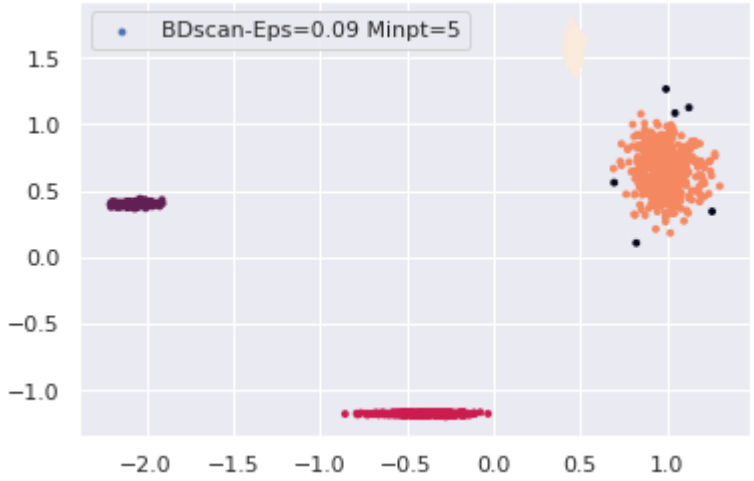
On en déduit la valeur estimée optimale de epsilon au point d'inflexion de la courbe, ici environ entre 0.02 et 0.05. On a tracé ici les courbes pour un nombre de voisins considéré
En parcourant la plage de epsilon qui semble convenir et en faisant varier le nombre de points min, on a un bon aperçu de l'impact de chaque paramètre sur le clustering.
Par exemple si epsilon est trop faible on détecte trop de clusters

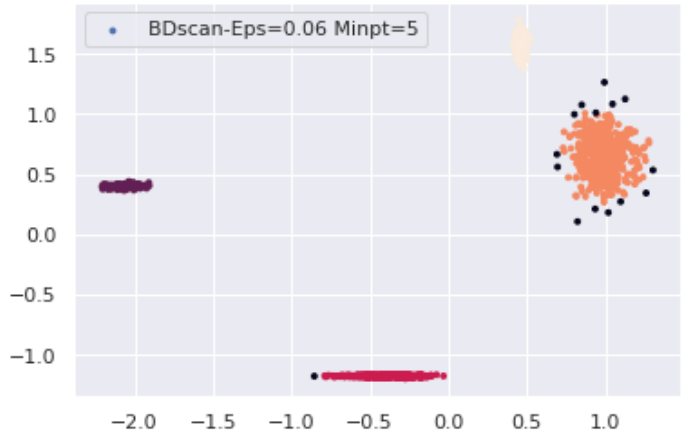


De plus, on remarque qu'avec un epsilon trop faible on peut aussi détecter trop de points bruits même avec le bon nombre de cluster

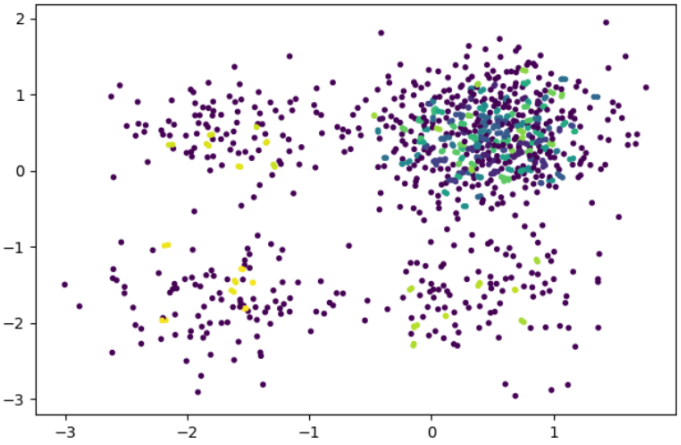


A partir d'une certaine valeur de epsilon on n'observe quasiment plus d'évolution du clustering sur ce set de donnée, mais on suppose que pour un set de données avec des clusters plus regroupés on risque de rassembler des clusters qui ne doivent pas l'être.



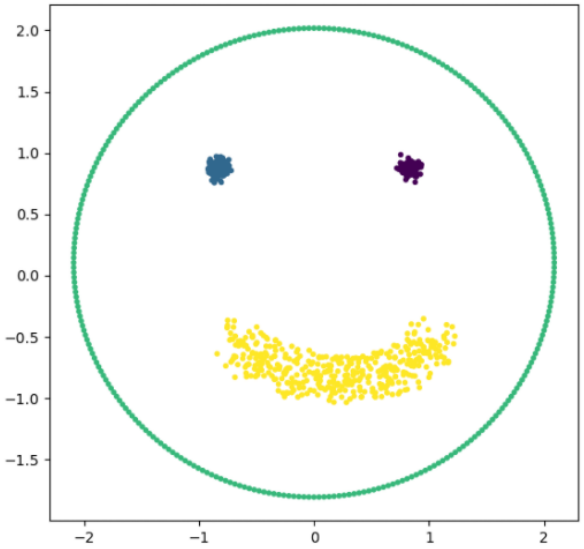


de la même manière, on essaie d'appliquer DBSCAN sur d'autres jeux de données, dans l'exemple suivant, la méthode DBSCAN n'est pas applicable:



il y a beaucoup de bruit et un trop grand nombre de cluster même en utilisant les méthode de détermination des paramètres optimaux

A l'inverse sur des jeux de données ou les clusters sont mieux définie (comme smile2.arff) DBSCAN marche très bien:



On utilise ici la méthode k-mean sur le set de donnée “d32.txt”
l’analyse de la silhouette nous indique que le bon nombre de cluster est 16, on obtient le clustering suivant: