TP Clustering

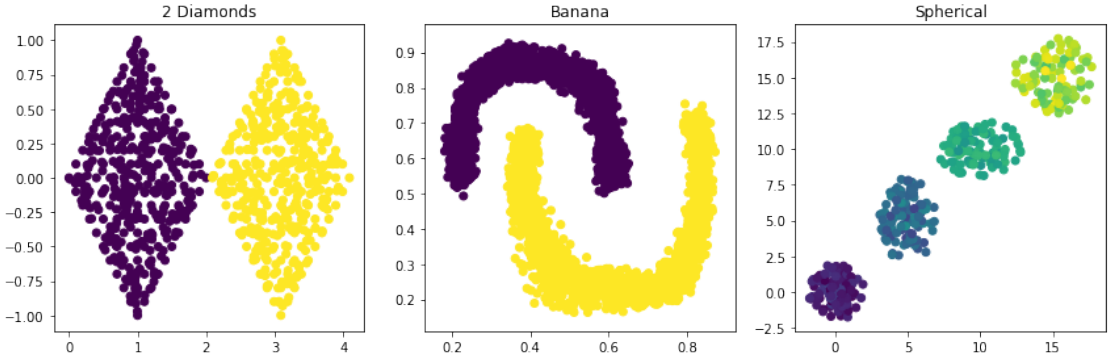
December 20, 2019

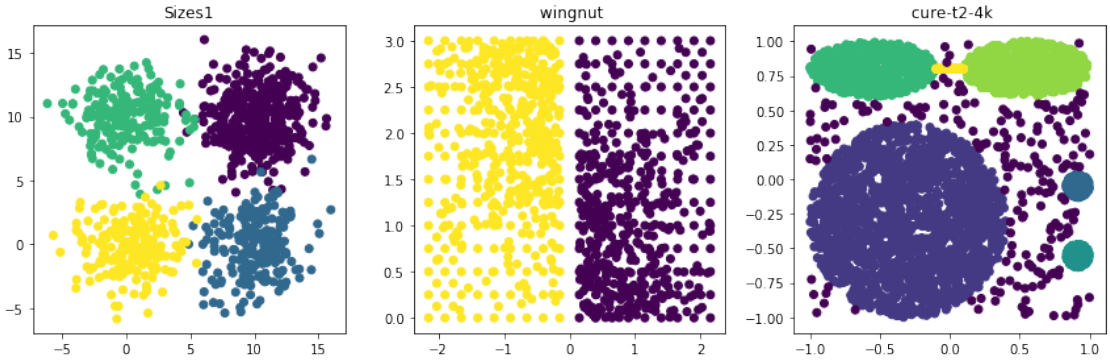
Amine LAKHAL & Mohamed Khalil JAMAI

1 Jeux de données

Nous utilisons 6 Datasets pour la suite des TPs. Chaque Dataset répond à un ou plusieurs critères:

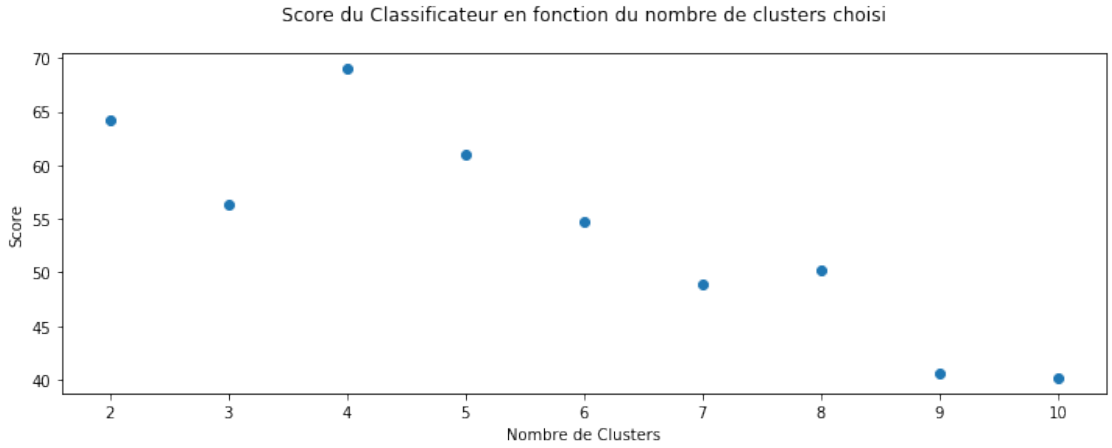
- `Formes convexes et Données non bruités : 2 Diamonds`
- `Formes non convexes avec des densités égales : Banana`
- `Formes convexes et bien separées : Spherical_4_3`
- `Formes mal separées : Sizes1`
- `Formes avec des densités différentes : wingnut et Disk Dense 3000`
- `Données bruités : wingnut (et xclara)`



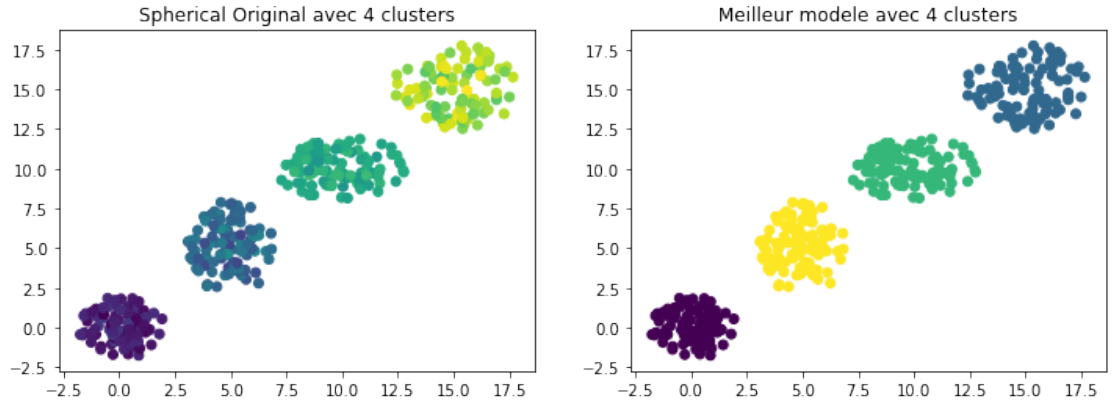


2 Clustering k-Means

Parmi les jeux de données retenus, soit des formes convexes, formes convexes mal séparées, des formes non convexes et des formes de densité variable, etc., nous allons appliquer itérativement la méthode précédente pour déterminer le bon nombre de clusters à l'aide de critères d'évaluation.



`Le meilleur modele pour Spherical contient 4 clusters. Il a un score de 68.98, et a un temps de calcul de 0.03 secondes.`

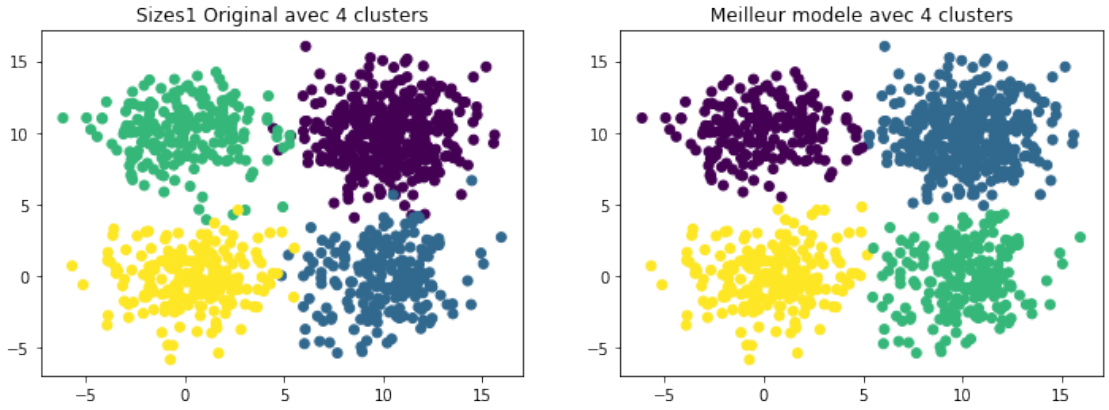


En effet, nous nous attendions à obtenir 4 clusters tout comme la dataset, car elle est composée de 4 formes bien distinctes. Cela parait cohérent car dans notre cas, les formes sont assez petites et bien separées. C'est pour cela que ces formes ont du coincider parfaitement avec les centres de clusters choisis par K-means. On remarque également que les scores obtenus avec d'autres nombres de clusters restent plutot bons.

NB: Nous n'afficherons plus tous les graphes de scores par soucis de place. Il est possible de l'afficher en passant le paramètre 'okScore' à True.

- Formes convexes mal séparées : Sizes1

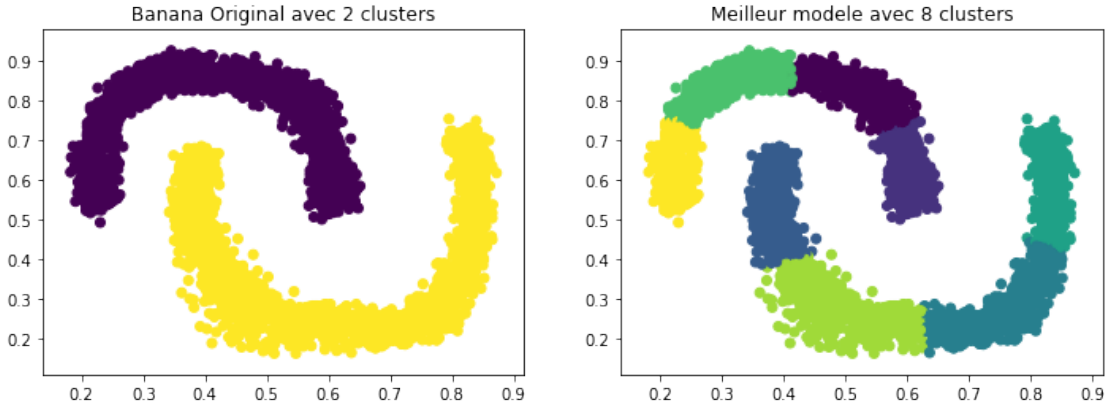
```
Le meilleur modele pour Sizes1 contient 4 clusters. Il a un score de 59.34, et a
un temps de calcul de 0.11 secondes.
```



A l'image des formes convexes bien séparées, les formes convexes un peu separées s'en sortent très bien avec K-means. Il y a seulement quelques points se situants aux extremités des clusters que K-means n'arrive pas à bien placer, mais cela se comprend parfaitement comme ce sont des points qui se situent assez loins de leurs voisins initiaux. Cela explique pourquoi le score est légérement plus bas que précedement (60 contre 70).

- Formes non convexes: Banana

```
Le meilleur modele pour Banana contient 8 clusters. Il a un score de 52.19, et a
un temps de calcul de 0.35 secondes.
```

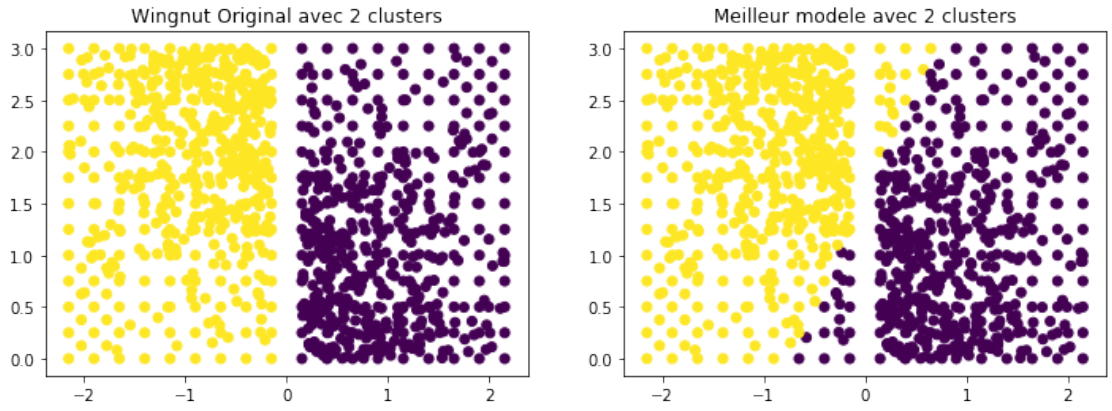


Pour les formes non convexes, K-means affiche un mauvais resultat en proposant 8 clusters au

lieu de 2. K-means crée des 'centres de clusters' un peu partout dans la figure (en fonction du nombre de clusters qu'on lui demande). C'est pour cela qu'on se retrouve souvent avec des clusters assez symétriques. Dans cette exemple, on peut déterminer les centres de clusters qui devraient correspondre au centre des formes qu'il propose.

Cependant, nous nous étonnons de voir que K-means affiche de bons scores (selon lui en tout cas) à partir de 6 clusters.

```
 - Formes de densité variable : Wingnut
Le meilleur modele pour Wingnut contient 2 clusters. Il a un score de 46.02, et
a un temps de calcul de 0.13 secondes.
```



K-means a également du mal à distinger des formes à densité variable. Encore une fois, on peut distinguer les centres de clusters dont il s'est basé pour commencer. Ce sont 2 clusters symétriques qui ne sont pas verticaux mais en diagonale.

Conclusion • Retrouvez-vous une sensibilité de l'algorithme à l'initialisation ?

Non, car kmeans++ utilise un certain algorithme à l'initialisation qui utilise toujours les memes points de départs en fonction de plusieurs dépendants de la dataset. Par contre, Random rend kmeans plus sensible car les points centraux de clustering à l'initialisation sont aléatoires, et plus symétriques comme avant. Ainsi, après plusieurs éxecutions, on peut obtenir des scores meilleurs que avec l'algorithme par défaut. Nous n'affichons pas les tests avec Random par manque de place.

• La méthode est-elle sensible à la nature des formes (cercle, rectangle, losange, non connexes, ...) et à la densité des données ?

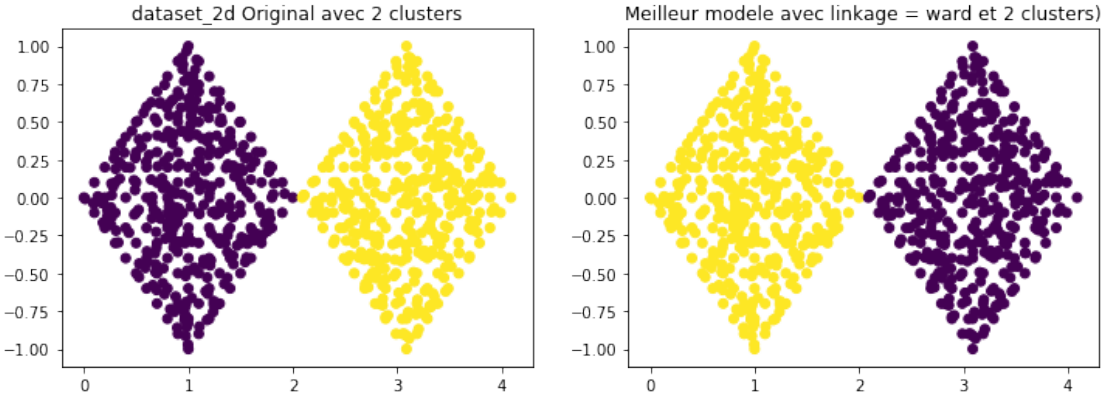
La méthode est sensible aux formes circulaires, ou du moins celles qui s'en rapprochent. C'est le cas des formes convexes plus généralement. On obtient des scores de 70 et 60 pour respectivement des formes convexes bien separées et mal separées. En ce qui concerne les formes non convexes,les résultats eux sont catastrophiques et on est très loin de la classification de la dataset, bien que les scores soient corrects. Par exemple Banana est autour de 50 en score mais propose 8 clusters au lieu de 2. K-means est également sensible à la densité dans le sens où il ne prend pas en compte ce

paramètre. Du moment que les formes sont convexes, comme pour Wingnut, les résultats restent assez bons.

3 Clustering agglomératif

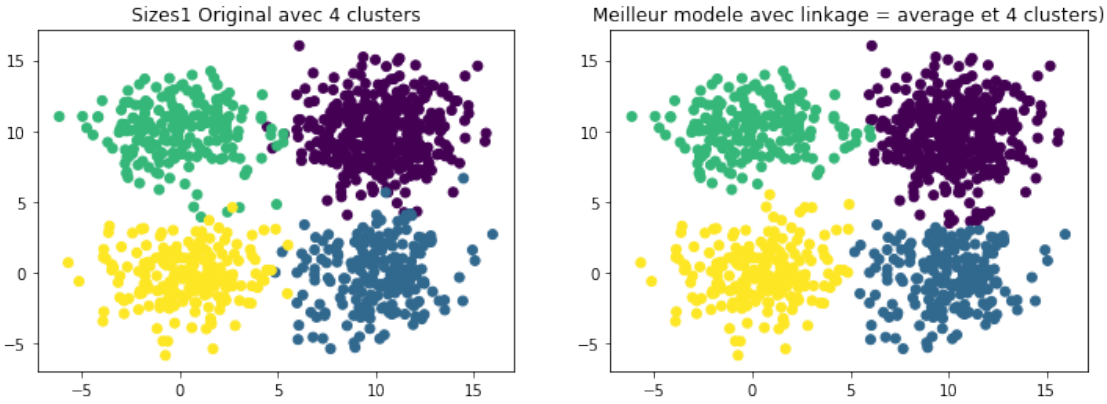
Tout d'abord, nous avons itéré sur les 4 métrics possibles, grâce à la fonction agglo_linkage, afin de trouver la meilleure manière de combiner des clusters selon le dataset. Ensuite, on garde cette donnée pour trouver le nombre de clusters idéal (en itérant dessus), et ainsi le meilleur score. Nous appliquons ceci sur tout type de formes: convexes bien et mal séparées, des formes non convexes et des formes de densité variable.

```
 - Formes convexes assez bien séparées: 2Diamonds
Le meilleur modele pour 2 Diamonds contient 2 clusters. Il a un score de 63.06,
et a un temps de calcul de 0.02 secondes.
```



Le clustering agglomératif distingue parfaitement les formes convexes bien séparées comme on peut le remarquer sur la figure ci-dessus.

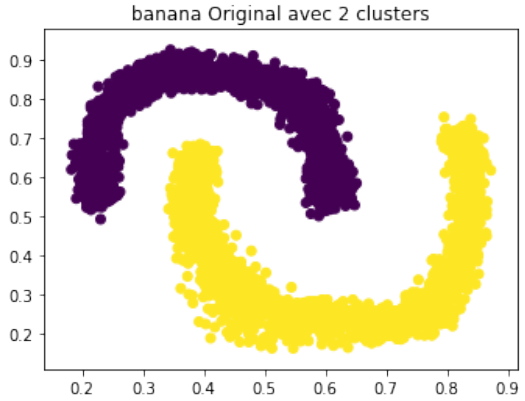
```
 - Formes convexes mal séparées: Sizes1
Le meilleur modele pour Sizes1 contient 2 clusters. Il a un score de 58.54, et a
un temps de calcul de 0.03 secondes.
```

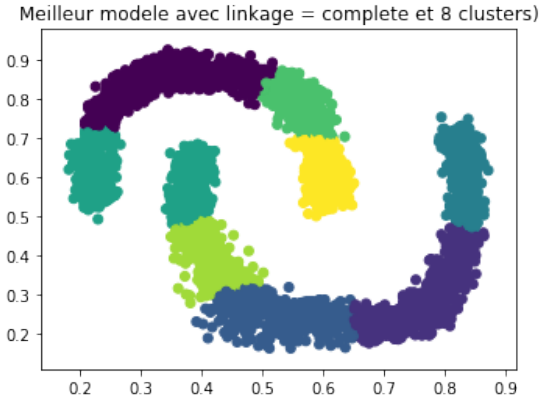


Pour les formes convexes qui sont mal séparées, cette méthode a renvoyé un bon résultat malgré le fait que certains points sont mal classés. Cela n'empêche pas que nous retrouvons quand même le bon nombre de clusters.

- Formes non convexes: Banana

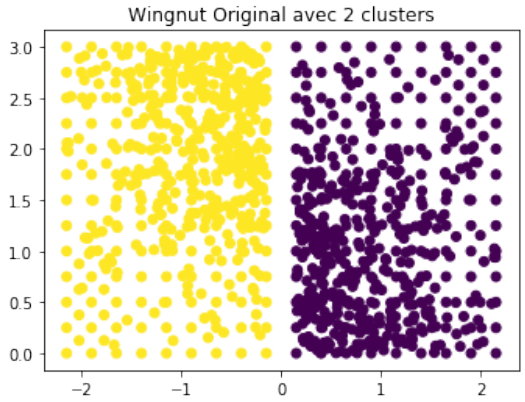
```
Le meilleur modele pour Banana contient 2 clusters. Il a un score de 47.02, et a
un temps de calcul de 0.28 secondes.
```

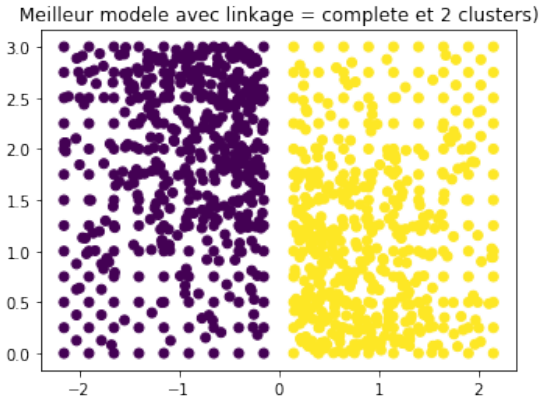




Cette fois-ci, le clustering agglomératif a du mal à différencier les classes de formes non convexes. En effet, pour Banana, on retrouve un grand nombre de clusters ainsi que certains points de classes différentes qui sont considérés d'une même classe.

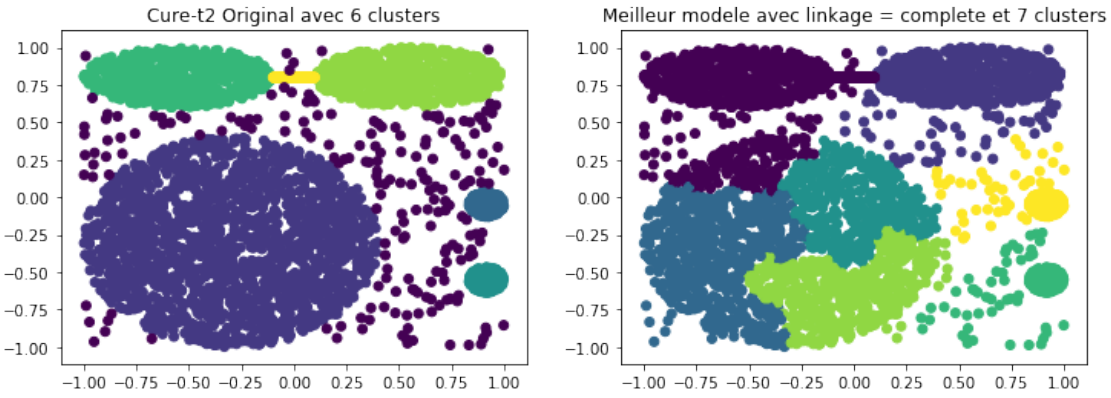
```
Le meilleur modele pour Wingnut contient 2 clusters. Il a un score de 44.82, et
a un temps de calcul de 0.02 secondes.
```





Pour les formes de densités variables, on a un très bon résultat avec 2 clusters bien distincts pour ce cas du wingnut ainsi d'une bonne répartition des points dans chaque classe.

Formes avec du bruit: Cure-t2
Le meilleur modele pour Cure-t2 contient 2 clusters. Il a un score de 46.48, et a un temps de calcul de 0.24 secondes.



Dans ce cas, on a obtenu un résultat faible. En effet, cette méthode ne distingue pas le bruit des autres classes. On remarque que les points sont répartis par région et que cela ne correspond pas à l'original.

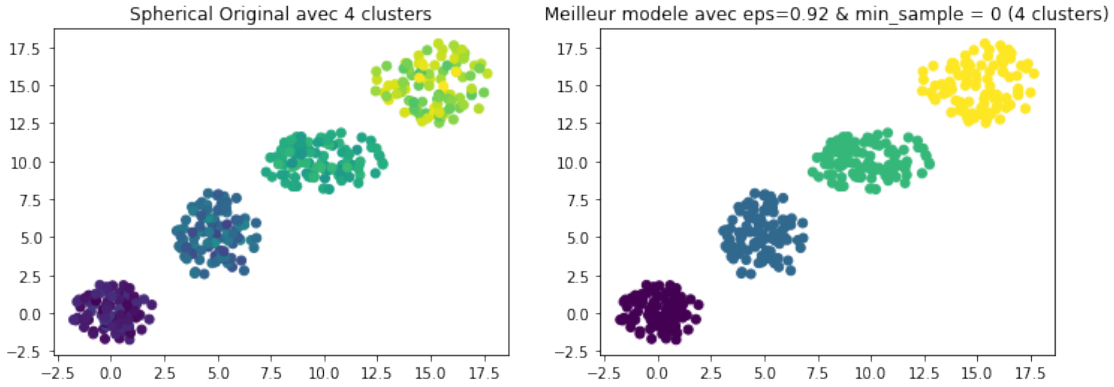
Conclusion Le clustering agglomératif renvoie les meilleurs résultats dans le cas des formes convexes. Nous avons constaté cela à travers les deux exemples avec les formes qui sont bien et mal séparées. Il est aussi insensible aux densités des données comme ce qu'on a remarqué avec wingnut. Enfin, cette méthode a un point faible qui est les formes non convexes vu qu'elle renvoie des résultats qui sont très loins de ce qui est attendu.

4 Clustering DBSCAN

Après avoir choisi des valeurs aléatoires pour les paramètres epsilon et min-sample, nous allons essayer de détérminer les meilleurs valeurs possibles, pour chacune des datasets suivantes: formes convexes bien identifiées, des formes convexes mal séparées, des formes non convexes, des formes de densité variable et la présence de bruit. On essaie d'abord de déterminer le meilleur, ensuite on le garde et on cherche le meilleur min_sample, le tout, pour chaque forme.

- Formes convexes bien séparées : Spherical

En gardant min_sample par défault, la meilleur valeur de Epsilon pour Spherical est de 0.92. Elle correspond à un score de 68.98, et a un temps de calcul de 0.006 sec.
En variant min_sample, on obtient comme meilleur valeur 0. Il correspond à un score de 68.98, et a un temps de calcul de 0.0057 sec.

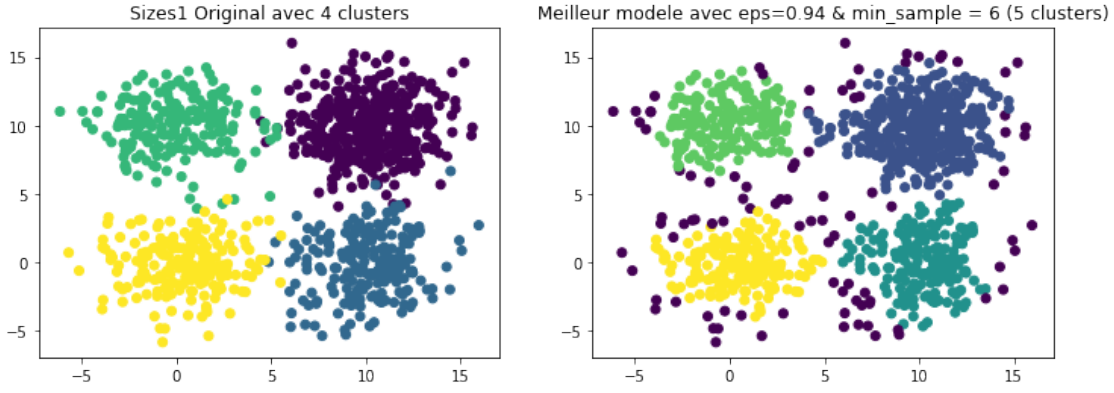


Le valeur d'Epsilon qui rend le meilleur score possible pour cette dataset se situe entre 0.7 et 1.0. Les scores sont quasiment les mêmes pour tous les espilon dans cet intervale-là. DBSCAN affiche de bons résultats pour les formes convexes, que ce soit en terme de score ou de clustering.

NB: Cette dataset est assez particulière car comme on peut l'apercevoir dans son affichage, il y énormement de nuances de couleurs, et donc beaucoup de classes. Il y en a plus de 300 en realité. Cependant, on se doute bien que ce sont 4 grandes classes, distinctes, et avec des couleurs très proches. DBSCAN arrive à faire abstraction de ces nuances et à bien rassembler tous les points en fonction de leurs localisation. Si ces nuances de couleurs avaient été un défault initialement, DBSCAN aurait pu le corriger.

- Formes convexes mal séparées : Sizes1

En gardant min_sample par défault, la meilleur valeur de Epsilon pour Sizes1 est de 0.94. Elle correspond à un score de 44.66, et a un temps de calcul de 0.0142 sec.
En variant min_sample, on obtient comme meilleur valeur 6. Il correspond à un score de 51.92, et a un temps de calcul de 0.0141 sec.

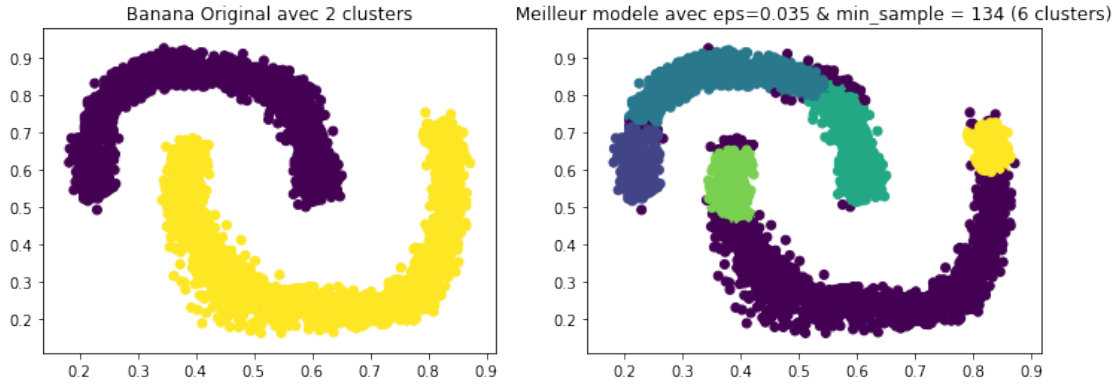


Pour les formes convexes mais mal separées, les résultas sont assez bons (45 pour epsilon entre 0.90

et 0.95). Visuellement aussi c'est plutot bon puisqu'on retrouve les 4 grandes classes. Seulement, DBSCAN est sensible aux points assez éloignés de leurs voisins initiaux et estime qu'ils appartiennent tous à un meme 5e cluster (couleur violet, à droite). DBSCAN a un peu du mal avec les formes non compacts, bien que convexes.

- Formes non convexes et de densités égales : Banana

En gardant min_sample par défault, la meilleur valeur de Epsilon pour Banana est de 0.035. Elle correspond à un score de 36.73, et a un temps de calcul de 0.098 sec.
En variant min_sample, on obtient comme meilleur valeur 134. Il correspond à un score de 38.07, et a un temps de calcul de 0.0642 sec.

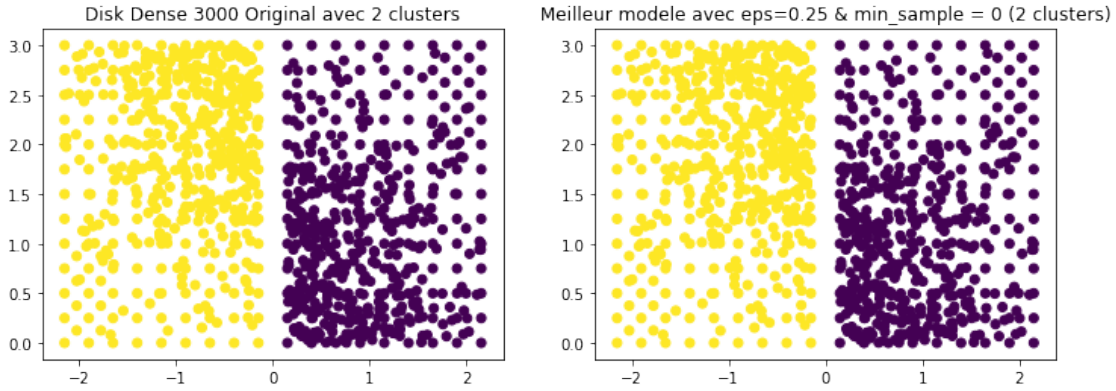


Le fait que la dataset Banana soit composée de formes non convexes n'a pas trop l'air de géner DBSCAN pour la banane du bas, mais pas pour celle du haut. Dans ce cas-ci, le meilleur Epsilon retenu est très petit, 0.035. En effet, il y a beaucoup de points dans cette dataset qui ne sont pas très proches de leurs voisins et DBSCAN n'a pas l'air de remarquer que ce sont des voisins.

Si faire varier epsilon (la distance maximale pour considérer 2 points comme voisins) avait un réel effet positif, la variation de la taille minimale des voisinages, elle, n'arrange rien pour les formes non convexes et bien séparées.

- Formes de densités variables : Wingnut

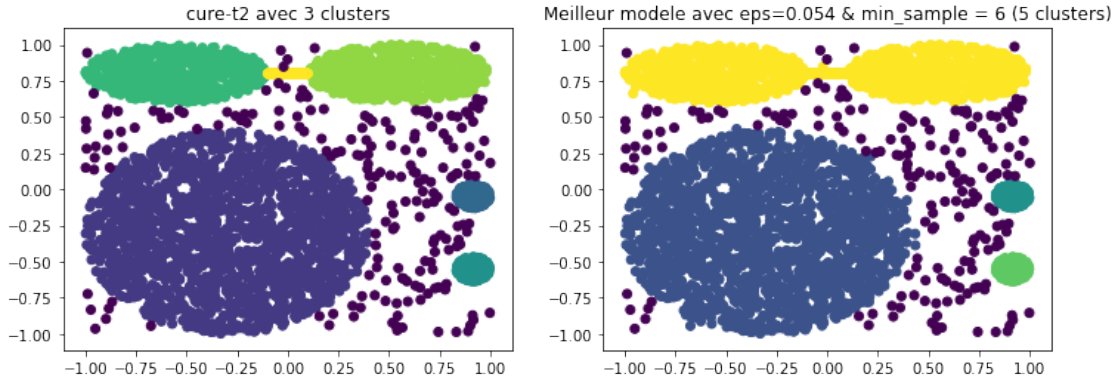
En gardant min_sample par défault, la meilleur valeur de Epsilon pour Wingnut est de 0.25. Elle correspond à un score de 44.82, et a un temps de calcul de 0.0154 sec.
En variant min_sample, on obtient comme meilleur valeur 0. Il correspond à un score de 44.82, et a un temps de calcul de 0.016 sec.



Les résultats sont très bons pour les formes avec des densités variables. DBSCAN est donc insensible à la densité. Le résultat est parfait. Encore une fois, DSBCAN gère bien la densité variable pour une forme donnée.

- Formes avec du bruit : Cure-t2

En gardant min_sample par défault, la meilleur valeur de Epsilon pour Cure-t2 est de 0.054. Elle correspond à un score de 41.62, et a un temps de calcul de 0.0411 sec.
En variant min_sample, on obtient comme meilleur valeur 6. Il correspond à un score de 41.59, et a un temps de calcul de 0.04 sec.



Comme préssenti tout à l'heure, DBSCAN est légérement sensible aux formes qui se chevauchent, ne serait-ce que légérement. La valeur de epsilon est très petite pour cette dataset. Comme les formes sont collées et que epsilon est petit, DBSCAN pense que les formes collées ne sont en réalité qu'une seule forme. Cependant, DBSCAN arrive à distinguer la quasi totalité du bruit et le mettre dans un cluster unique, à l'exception de quelques points collés aux autres formes.

La variation de min_sample n'arrange absolument pas DBSCAN pour les formes avec du bruit. Les scores restent néanmoins assez bons mais ce n'est pas représentatif.

Conclusion Epsilon La valeur d'epsilon dépend de la dataset, et surtout de son échelle, à savoir la distance entre les différents points. Par exemple, pour la dataset 2 Diamonds ayant des formes convexes denses avec des points très proche: Le meilleur score est obtenu pour un epsilon bien précis et petit se situant autour de 0.12.

Que ce soit pour les formes convexes on non convexes, du moment que les formes sont distinctes, denses et sans bruits, les résultats sont au moins assez bons. C'est le cas pour Spherical par exemple. En ce qui concerne la séparation des formes, on voit bien que pour 2Diamonds, DBSCAN a du mal à gérer les points situés entre les 2 losanges, mais le résultat reste assez bon. En parlant de 2Diamonds, on apercoit aussi l'éffet de la densité. Puisque DBSCAN a retenu un epsilon petit et les points légérement détachés sont considérés comme appartenant à d'autres clusters.

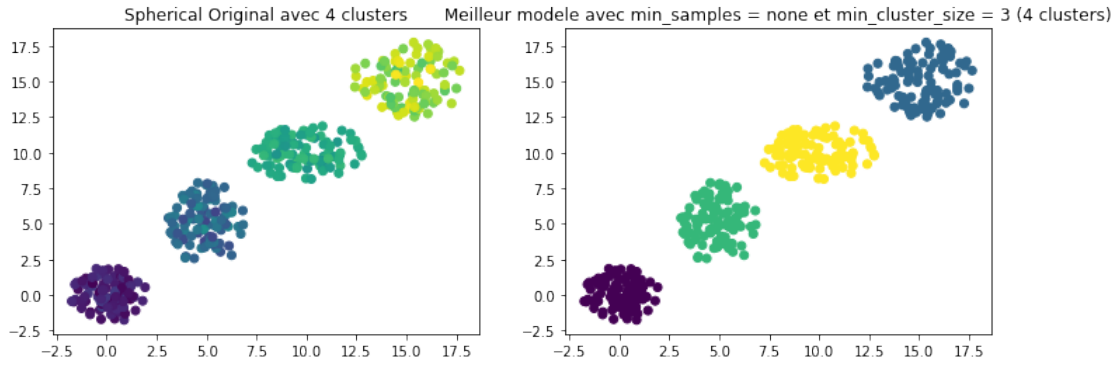
Conclusion Min-sample On remarque que en variant min_sample, DBSCAN affiche les meilleurs résultats possibles sur des formes très denses. DBSCAN arrive à s'adapter en cas de variation de densité mais reste un peu sensible au bruit par contre.

Nous pensons aussi que DBSCAN affiche les meilleurs résultats lorsque les différentes formes sont très proches, voire collées. Celà explique pourquoi la dataset Banana bien que très dense en points, se retouve avec un seul cluster. Et cela explique pourquoi les autres sont plutot bien traités. La dataset Sizes1 est un bon exemple car elle se situe à mi chemin: elle est très dense en points mais les formes sont légérements éloignées. Le résutat est donc lui aussi mitigé avec la reconnaissance des formes mais pas des points autour.

5 Clustering HDBSCAN

Pour cette partie, les paramètres les plus importants à utiliser sont min_cluster_size et min_samples. Afin d'avoir les meilleures performances, nous avons décider de chercher d'abord le min_cluster_size qui renvoie le meilleur score. Ensuite, nous rajoutons le paramètre min_samples qu'on va faire varier entre 1 et la meilleure valeur de min_cluster_size. Cela nous permettrait de choisir le bon nombre de points par cluster et d'éviter de considérer certains points comme bruit alors qu'ils appartiennent à des clusters de petites tailles.

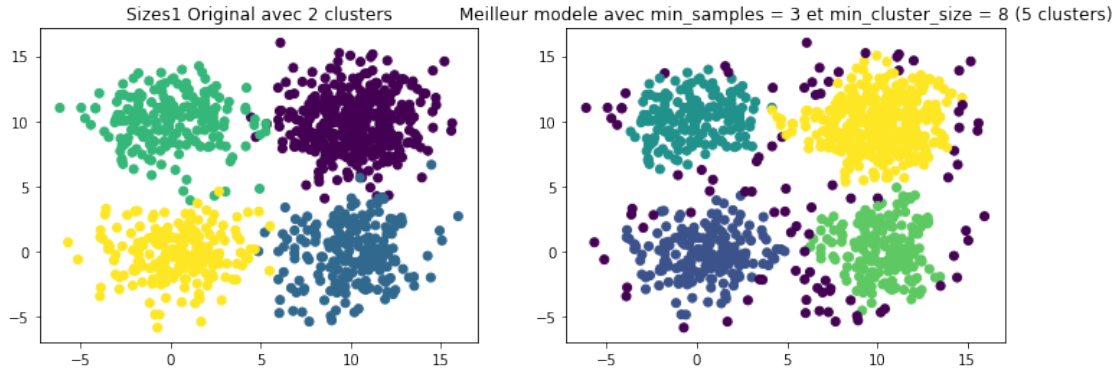
```
Formes convexes bien séparées : Spherical

Le meilleur score est de 68.98 pour min_cluster_size = 3
Pour min_cluster_size = 3 et min_samples = none le meilleur score devient : 52.9
```



Nous remarquons que HDBSCAN arrive bien à distinguer chaque cluster. En effet, pour min_cluster_size à 3, on arrive à avoir un très bon résultat avec 4 clusters bien distincts. Par contre, en rajoutant le paramètre min_samples, on obtient un résultat moins et c'est pour cela que nous le gardons à nul.

`Formes convexes mal séparées: Sizes1`

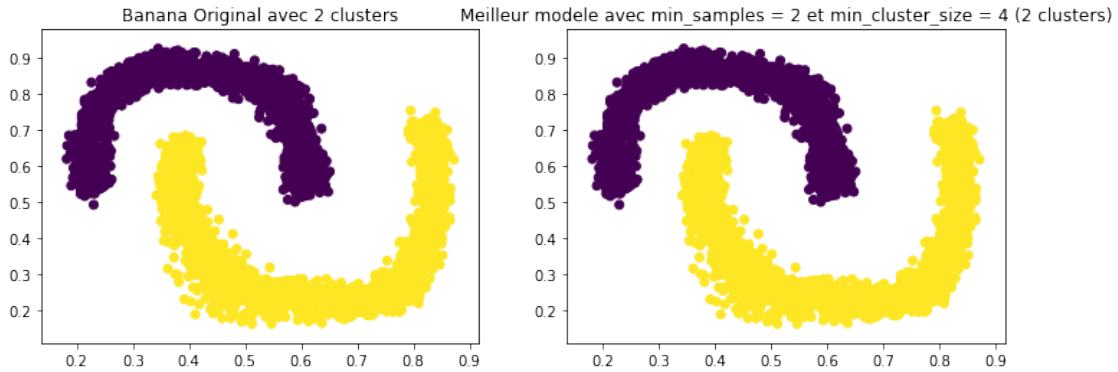
```
Le meilleur score est de 51.74 pour min_cluster_size = 8
Pour min_cluster_size = 8 et min_samples = 3 le meilleur score devient : 52.18
```



Pour les formes convexes mal séparées, cette méthode a plus de mal à bien différencier les clusters. On remarque qu'il y a une distinction entre ces derniers, sauf que pour les points plus éloignés, cette méthode n'arrive pas bien à les attribuer à leurs formes. Cette fois-ci, on a utilisé le paramètre min_samples pour améliorer le résultat final.

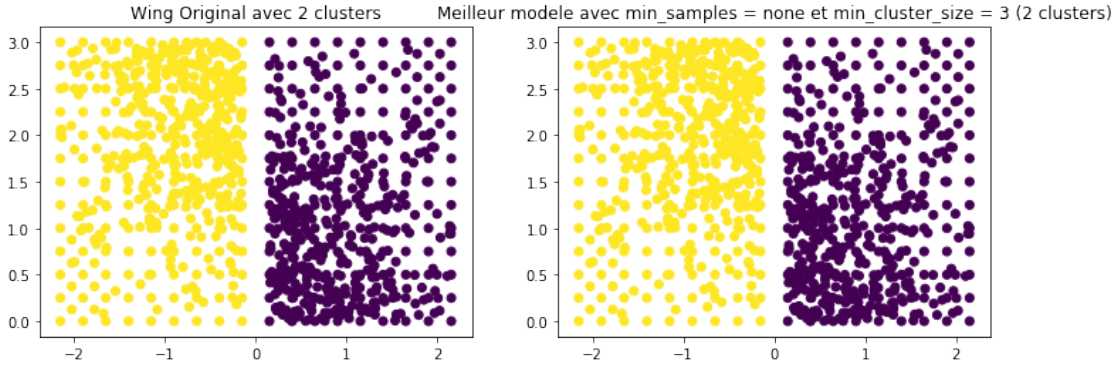
`Formes non convexes et de densités égales : Banana`

```
Le meilleur score est de 36.73 pour min_cluster_size = 4
Pour min_cluster_size = 4 et min_samples = 2 le meilleur score devient : 36.73
```



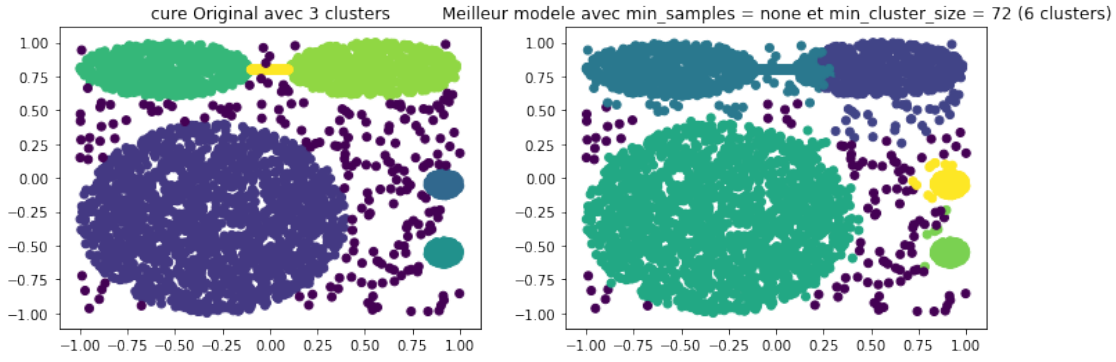
Pareil que le cas précédent, cette méthode n'a pas de mal avec les formes non convexes et de densités égales comme dans le cas de Banana où on remarque une bonne séparation entre les classes.

```
Formes de densités variables : wing

Le meilleur score est de 44.82 pour min_cluster_size = 3
Pour min_cluster_size = 3 et min_samples = none le meilleur score devient :
19.65
```



Dans le cas des densités variables, HDBSCAN a réussi à bien séparer les clusters. On a obtenu un excellent résultat malgré le fait que le score soit faible. D'ailleurs, le fait de bien distinguer les densités est considéré parmi les points forts de cette méthode.

```
Formes avec du bruit : cure-t2-4k

Le meilleur score est de 45.67 pour min_cluster_size = 72
Pour min_cluster_size = 72 et min_samples = none le meilleur score devient :
42.96
```



On remarque que HDBSCAN arrive à bien distinguer le bruit bien qu'il considère certains points comme du bruit et que certains points bruit soient classés dans des clusters. Néanmoins le résultat en globalité est correct. On constate aussi qu'il a aussi du mal à séparer des clusters qui sont liés.

Conclusion: D'après les résultats, nous remarquons que HDBSCAN arrive à bien identifier les formes convexes bien séparées comme spherical. C'est aussi le cas avec les formes convexes mal séparées de Sizes1 avec une performance moins importante. Pour les formes qui contiennent du bruit, HDBSCAN distingue le bruit des autres clusters même s'il considère quelques points comme bruits alors qu'ils ne le sont pas, ou l'inverse. Enfin, on remarque qu'il renvoie de très bon résultat avec les formes de densités variées comme le cas du dataset wingnut et même avec les formes non convexe comme par exemple pour banana

6 Synthèse

KMeans et DBScan sont considérées comme les méthodes les plus utilisées et les plus évidentes à implémenter. Chacune dispose de caractéristiques et de performances différentes.

KMeans est plus rapide que DBScan. Tandis que ce dernier n'a pas besoin du nombre de cluster. Parmi les inconvénients de KMeans on constate qu'elle essaie de créer un cluster de même taille, peu importe la façon dont les données sont dispersées. Cette méthode ne se soucie pas aussi de la densité des données.

DBSCAN résout certains des problèmes des KMeans en travaillant avec la densité des points. Il s'agit d'une méthode basée sur la densité. L'hypothèse principale de DBSCAN est que deux régions denses sont séparées par une région peu dense. DBScan ne fonctionne pas bien sur des clusters de densités différentes. Elle a besoin d'une sélection rigoureuse de ses paramètres.

Pour le clustering agglomératif, ses points forts sont qu'il est facile à comprendre et à implémenter, à l'image de K-means d'ailleurs. Par contre, il fournit rarement la meilleure solution. Il implique beaucoup de décisions arbitraires vu que nous n'avons pas manipulé beaucoup de paramètres. En plus, il fonctionne mal avec des types de données mixtes et sur de très grands ensembles de données. C'est pourquoi des clusters plus petits sont générés, mais cela peut être utile pour la découverte par exemple. Le clustering agglomératif est d'une complexité temporelle élevée. L'utilisation de différentes métriques de distance pour mesurer les distances entre les clusters peut générer des résultats différents. Le mieux est d'effectuer plusieurs expériences et de comparer les résultats pour confirmer la véracité des résultats originaux.

Pour HDBScan, l'avantage important est que nous pouvons apercevoir concerne les formes de densité variable. Puis, le deuxième avantage est que le paramètre epsilon est éliminé. Au lieu de cela, nous avons un nouveau paramètre min_cluster_size qui est utilisé pour déterminer le nombre de points qu'on veut avoir au minimum dans un cluster. Cela échange un paramètre peu intuitif par un paramètre qui n'est pas si difficile à choisir. Cela s'applique également sur DBSCAN. Le réel atout de HDBSCAN comparé à DBSCAN reste la gestion des densités variables et dans une moindre mesure, le bruit