

TP Clustering

Pierre BINET

Constance GAY

5SDBD - 2019/2020

Git : https://github.com/ConstanceGay/Apprentissage_Cluster

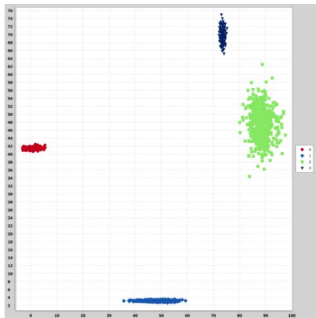
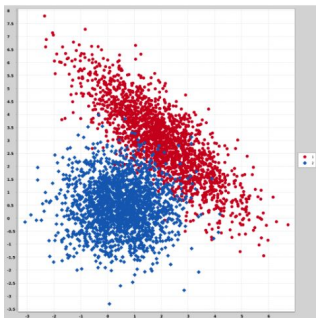
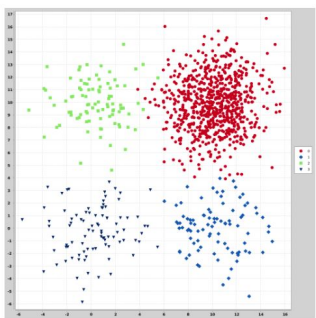
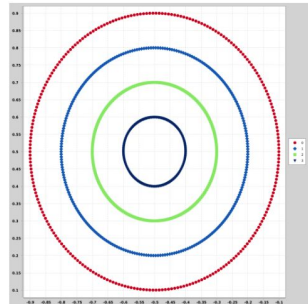
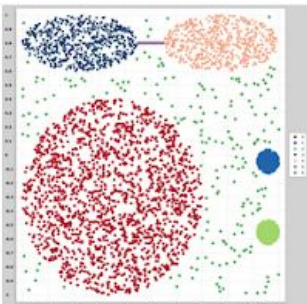
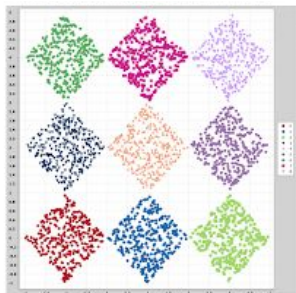
1. Jeux de données

- formes convexes / non convexes
- formes bien séparées / mal séparées
- densité similaire / variable
- présence ou non de données bruitées

Travail à réaliser:

- **Sélectionner quelques jeux de données (il faut pouvoir justifier les choix)**

Nous avons choisi les jeux de données suivants:

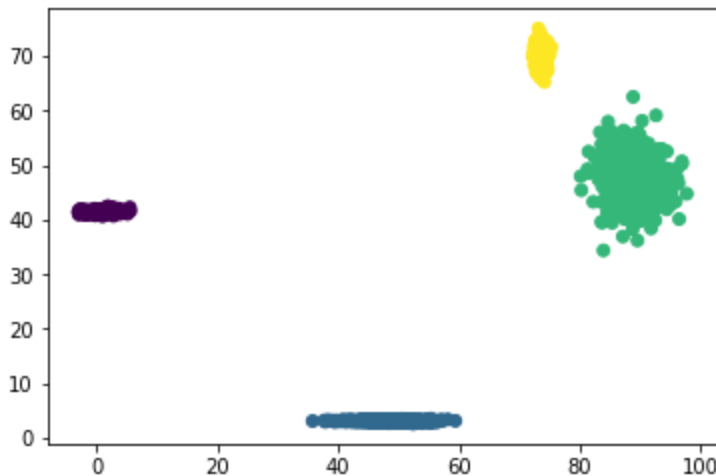
<p>Le plus simple: 2d-4c: convexe + formes bien séparées + densité variable</p> 	<p>Engytime: mal séparées + densité variable</p> 	<p>Sizes4: densité variable</p> 
<p>Dartboard1: non convexe + formes bien séparé + densité similaire + non bruité</p> 	<p>Cure-t2-4k: très bruité, introduit avec DBSCAN pour tester la détection du bruit</p> 	<p>Diamond9: introduit avec HDBSCAN pour tester la sensibilité à la nature des formes</p> 

2. Clustering K-means

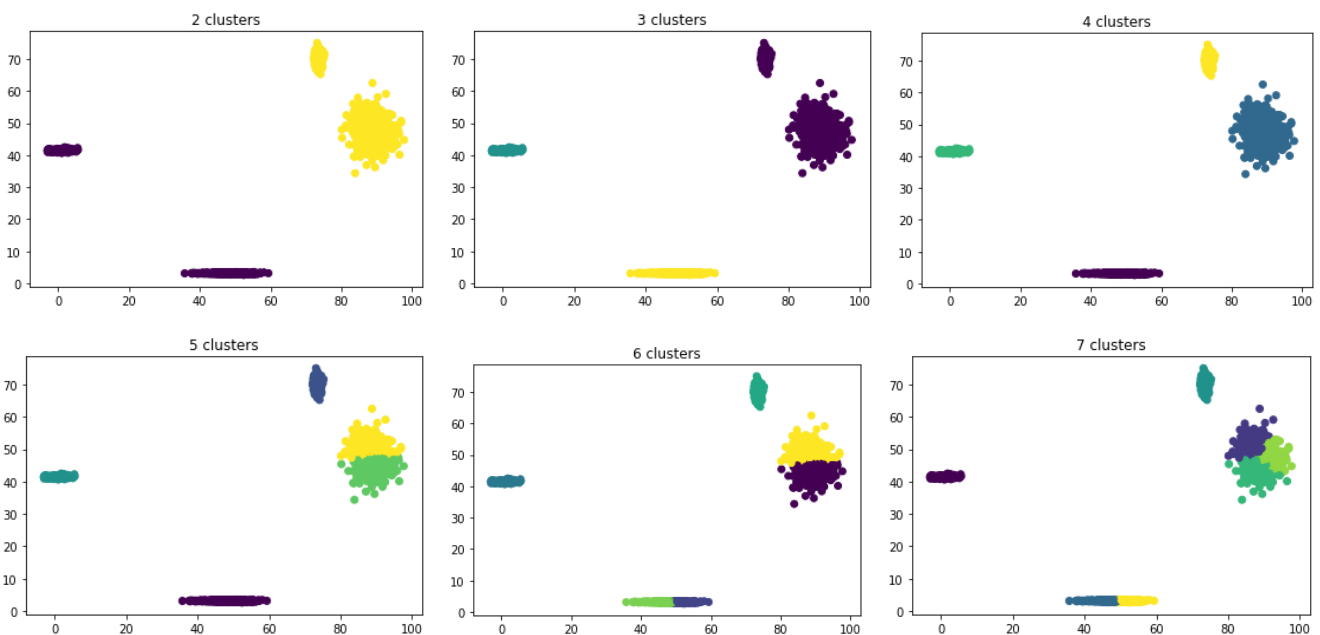
```
class sklearn.cluster.KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001,
precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=None,
algorithm='auto')
```

Nous nous sommes tout d'abord concentré sur le dataset **2d-4c**, qui présente des formes convexes et bien séparées.

- Appliquez la méthode k-means **en lui donnant directement le nombre de clusters attendus** (utilisez l'initialisation k-means++)

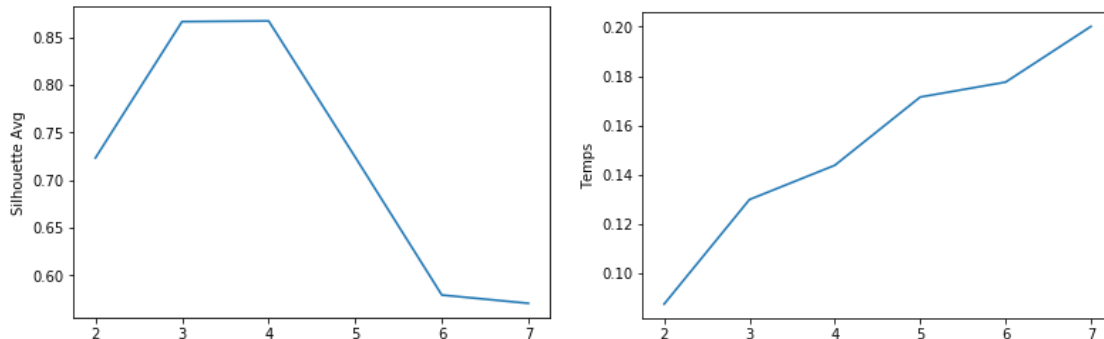


- Appliquez itérativement la méthode précédente pour déterminer le bon nombre de clusters à l'aide de critères d'évaluation fournis par Scikitlearn :

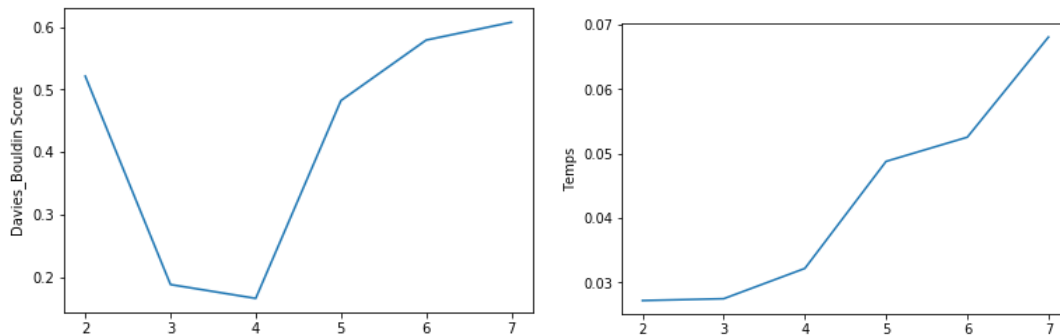


Choisissez pour cela un ou des critères d'évaluation appropriés:

Nous avons choisi la méthode du **coefficient de silhouette** (coefficient combinant une mesure de cohésion et une de séparation, que l'on cherche à maximiser):



Et nous avons dans un esprit de comparaison également testé l'**indice de Davies Bouldin** (lui aussi également le fruit de la combinaison d'une mesure de cohésion et d'une mesure de séparation, coefficient que l'on cherche cette fois-ci à minimiser).

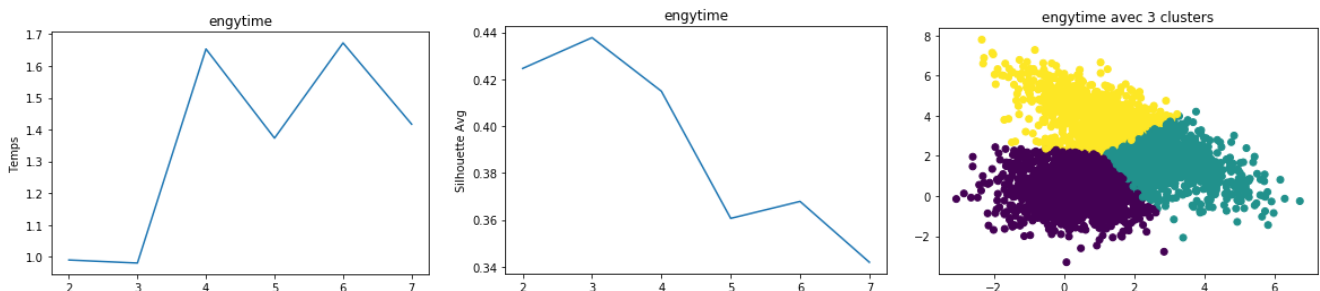


Arrivez-vous à retrouver le résultat attendu à l'aide de ces critères d'évaluation ?

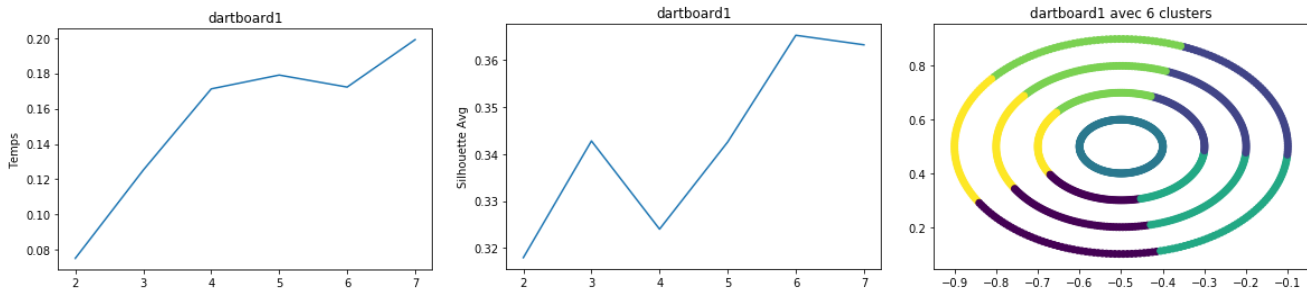
Oui! Les deux méthode indiquent que 4 est bien le nombre de clusters le plus approprié.

Reprenez les expérimentations (méthode k-Means itérative sur le nombre de clusters) en considérant cette fois :

- des formes convexes mal séparées: **Engytime**



- **des formes non convexes: Dartboard1**



- **des formes de densité variable: Sizes4**



- **Arrivez-vous à retrouver le résultat attendu à l'aide de ces critères d'évaluation ?**

K-means a beaucoup de mal à trouver le bon nombre de clusters dans le cas de formes non-convexes ou mal séparées, comme on l'a vu avec **Dartboard1 et Engytime**. En revanche avec **Sizes4**, le bon nombre de cluster est trouvé, on en déduit que la variabilité de la densité impacte moins k-means.

Questions générales sur les expérimentations menées :

- **Pouvez-vous éviter de tester trop de valeurs différentes de k ?**

Une bonne façon de ne pas avoir à tester trop de valeurs de k serait de s'arrêter lorsqu'on attend un pic dans la courbe montrant le Silhouette average. Le risque cependant serait de s'arrêter à un maximum local et ne pas chercher plus loin.

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

K-means ne semble pas être trop sensible à l'initialisation: les résultats ont été sensiblement les mêmes à chaque fois que nous avons fait tourner notre code.

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

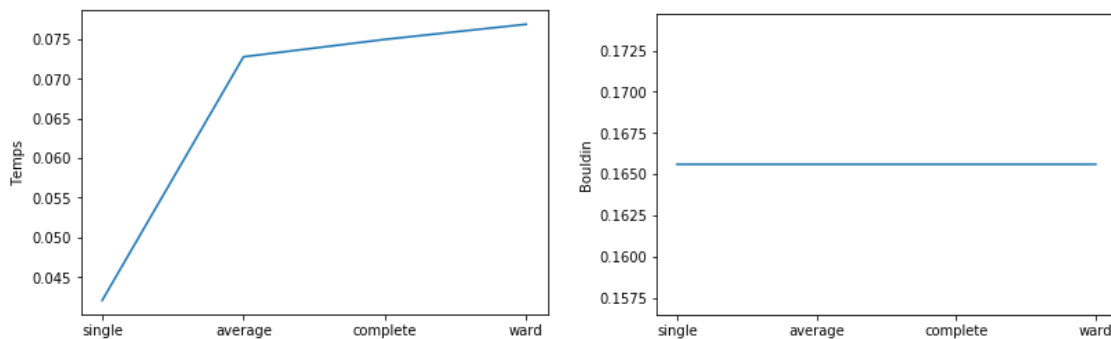
Oui, K-means ne semblent pas détecter le bon nombre de cluster pour les formes non convexes. En revanche il semble ne pas être trop influencé par la densité variable.

3. Clustering Agglomératif :

`class sklearn.cluster.AgglomerativeClustering(n_clusters=2, affinity='euclidean', memory=None, connectivity=None, compute_full_tree='auto', linkage='ward', pooling_func='deprecated', distance_threshold=None)`

Dans un premier temps, nous nous sommes comme pour k-means concentré sur le dataset **2d-4c**, qui présente des formes convexes et bien séparées.

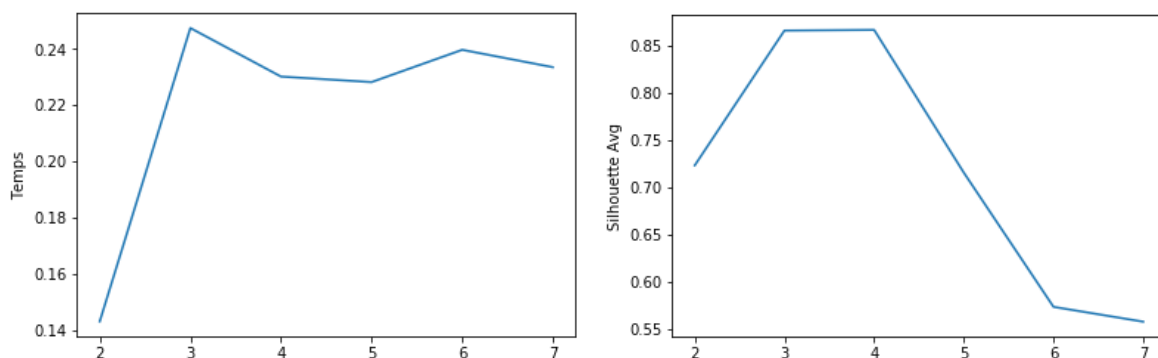
- **Appliquez une méthode de clustering agglomératif en lui donnant le nombre de clusters attendus. Considérez différentes manières de combiner des clusters (single, average, complete, ward linkage), uniquement pour la distance euclidienne.**



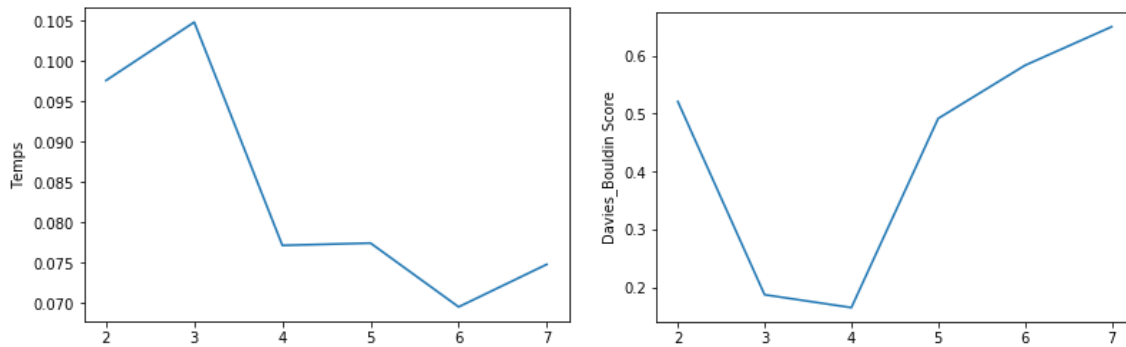
Le résultat est le même (exactement le même résultat avec la méthode d'évaluation DB) mais le temps pris pour trouver le résultat semble bien plus court avec la méthode single.

- **Appliquez itérativement la méthode précédente pour déterminer le bon nombre de clusters à l'aide des critères d'évaluation fournies par Scikitlearn :**
 - Reprenez le ou les critères d'évaluation précédents ;
 - Mesurez le temps de calcul

Silhouette:



Davies Bouldin :

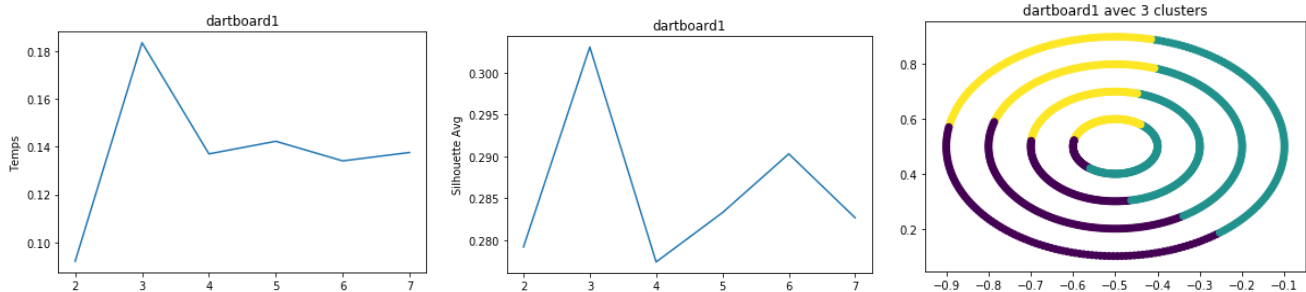


– Arrivez-vous à retrouver le résultat attendu à l'aide de ces critères d'évaluation ?

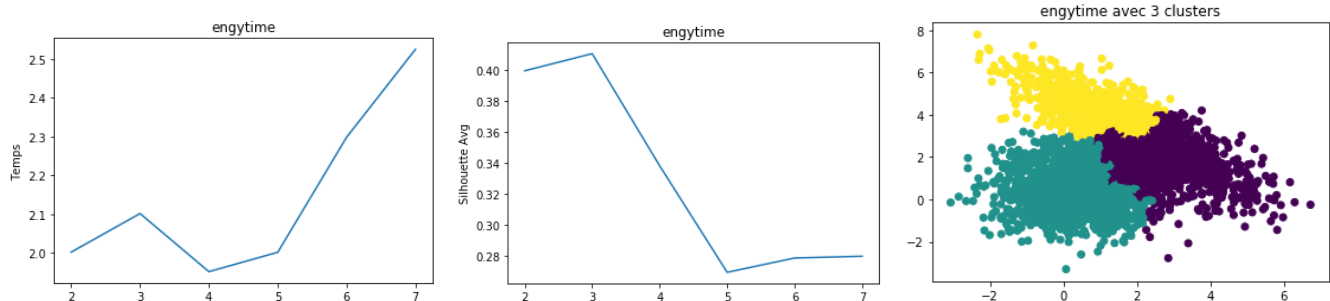
Oui, on retrouve que le nombre de clusters le plus approprié est 4, pour les 2 méthodes d'évaluations.

Reprenez les expérimentations (méthode agglomérative itérative sur le nombre de clusters) en considérant cette fois :

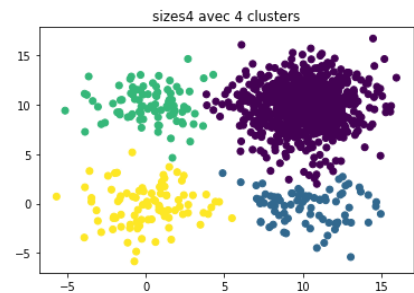
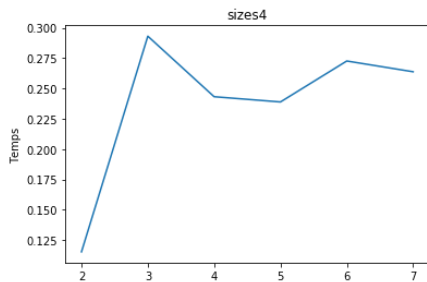
- des formes non convexes: **Dartboard1**



- des formes convexes mal séparées: **Engytime**



- **des formes de densité variable: Sizes4**



- **Arrivez-vous à retrouver le résultat attendu à l'aide de ces critères d'évaluation ?**

Tout comme k-means, le clustering agglomératif trouve le bon résultat pour **sizes4**, mais ne détecte pas le bon nombre de cluster dans le cas de **dartboard1** et **engytime**.

Questions générales sur les expérimentations menées :

- **Pouvez-vous éviter de tester trop de valeurs différentes de k ?**

On peut répondre la même chose que pour Kmeans, qu'on pourrait s'arrêter à un optimum trouvé mais avec le risque de n'avoir qu'un optimum local.

- **Quel est l'impact des différentes combinaisons des clusters ?**

Les différents types de combinaisons des clusters permettent de choisir quel critère sera minimisé afin de faire fusionner deux clusters. **Ward** permet de minimiser la variance des clusters que l'on fusionne, alors que **average** minimise la distance moyenne entre deux clusters, **complete** se base sur la plus grande distance entre les points de deux clusters et **single** prend la distance minimum entre les points des deux clusters. Dans le TP nous avons pu constater que le type de combinaison n'avait pas d'impact sur le score mais que certains comme **single** étaient beaucoup plus rapides que **ward** qui était le plus lent.

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

Tout comme k-means, le clustering agglomératif semble être influencé par la nature des formes, leur séparation et plus particulièrement par la convexité des données, tout en étant moins influencé par la densité variable des données.

4. Clustering DBSCAN

`class sklearn.cluster.DBSCAN(eps=0.5, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None)`

Considérez de nouveau les exemples comportant des formes convexes bien identifiées. La présence de données bruitées n'est pas à prendre en compte.

On reprend donc **2d-4c**.

- **Appliquez la méthode DBSCAN en lui donnant des valeurs "au hasard" pour les paramètres min-sample et eps et en laissant la métrique de distance à sa valeur par défaut**

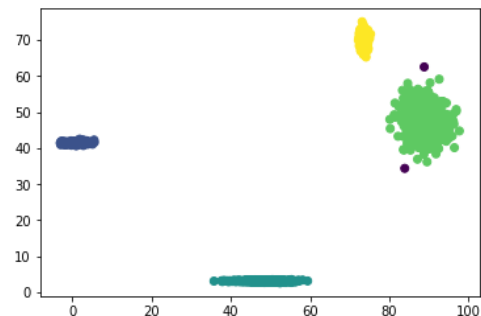
Valeurs données au hasard:

eps=4.06 et min_smp=8

Résultat:

Temps: 0.04412741400000186

Score: 3.3094878518458257

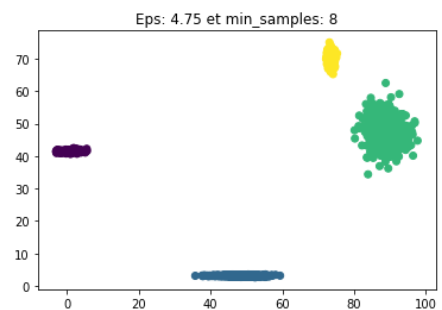
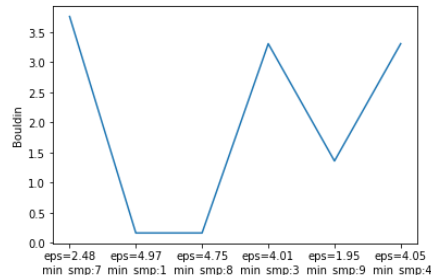
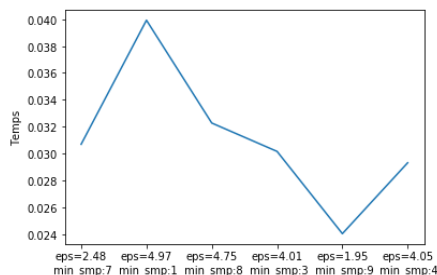


eps: Two points are considered neighbors if the distance between the two points is below the threshold epsilon.

min_samples: The minimum number of neighbors a given point should have in order to be classified as a core point. *It's important to note that the point itself is included in the minimum number of samples.*

- **Appliquez itérativement la méthode précédente pour déterminer des bonnes valeurs pour les paramètres min-sample et eps à l'aide des critères d'évaluation fournis par Scikitlearn :**

- **Reprenez le ou les critères d'évaluation précédents**
- **Mesurez le temps de calcul**



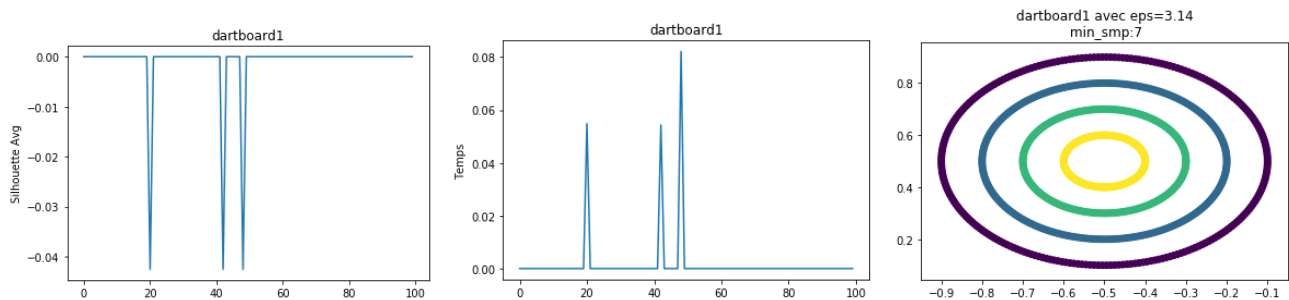
- **Arrivez-vous à retrouver le résultat attendu à l'aide de ces critères d'évaluation ?**

Oui, il arrive parfois que la meilleure combinaison de valeur d'eps et de min_samples soit sélectionnée au hasard et on peut afficher le résultat: on retrouve 4 clusters et un score de Davies-Bouldin très bas. Pour cela, nous devons donc avoir un nombre d'itérations suffisant pour que cette combinaison apparaisse.

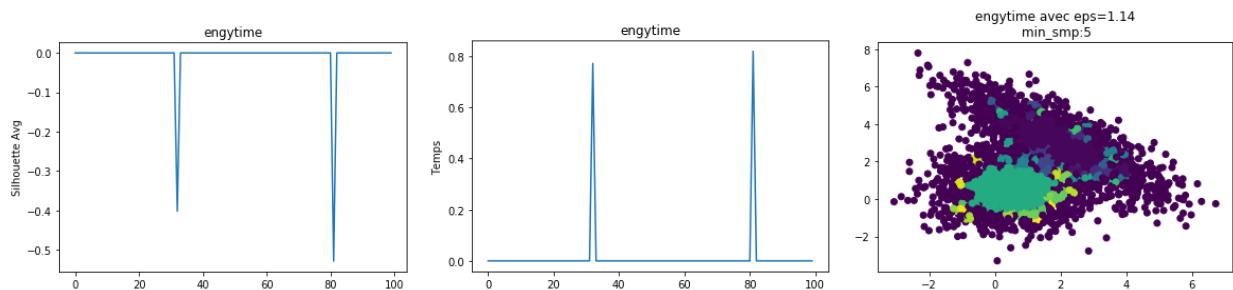
Dans l'exemple en fin de page précédente, avec ϵ compris entre 0.1 et 5 et min_samples entre 1 et 10, on tombe donc sur le bon nombre de cluster.

Reprenez les expérimentations en considérant cette fois :

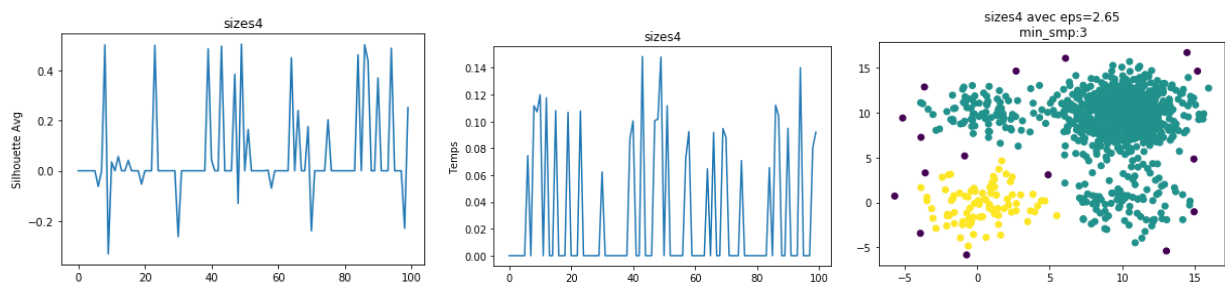
- **des formes non convexes**



- **des formes convexes mal séparées**

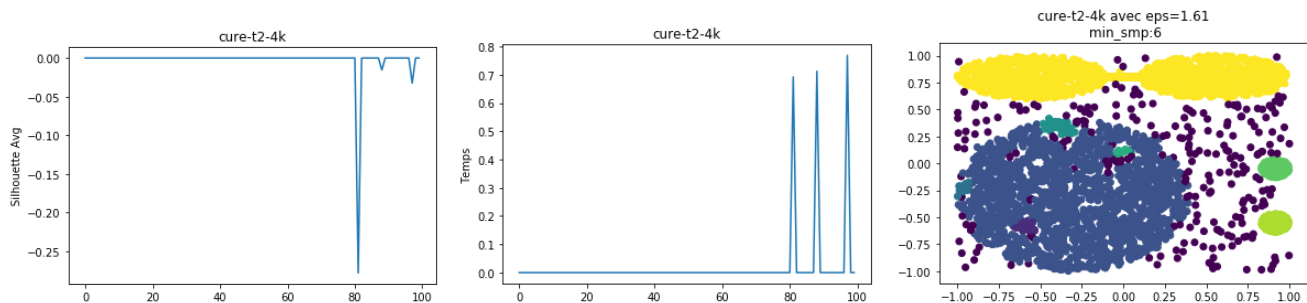


- **des formes de densité variable**



- **la présence de bruit**

Nous avons pour cela choisi un nouveau jeu de donnée, très bruité, **cure-t2-4k**.



- **Arrivez-vous à retrouver le résultat attendu à l'aide de ces critères d'évaluation ?**

DBSCAN semble être adapté pour l'étude des formes non-convexes et non bruitées: c'est jusque-là la seule méthode qui a su parfaitement reconnaître le **dartboard1**. En revanche il a beaucoup de mal avec les jeux de données bruités, de densité variables ou mal séparées.

Questions générales sur les expérimentations menées :

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

A l'inverse de k-means et du clustering agglomératif, DBSCAN a tendance mieux détecter le bon nombre de cluster pour les formes non convexes, mais est très influencé par la variabilité de la densité, la séparation, ainsi que par le bruit dans les données.

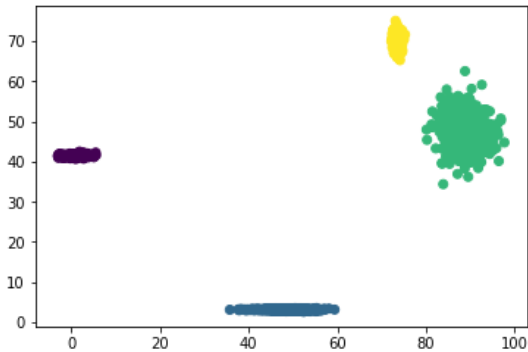
- **Le bruit dans les données est-il bien identifié ?**

Le bruit dans les données semble mieux identifié que pour les autres méthodes, mais cela a une limite car on voit que dans le cas de **cure-t2-4k**, tout le bruit n'est pas reconnu, et certaines parties du cercle principal au centre du jeu sont reconnues comme étant du bruit.

5. Clustering HDBSCAN

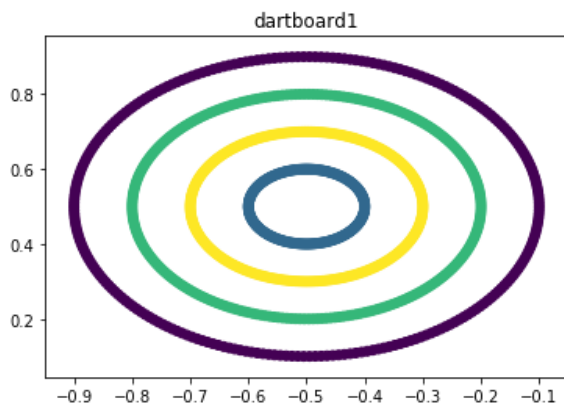
Reprenez les expérimentations effectuées avec DBSCAN sur des données bruitées et des données de densité variables. Comparez les résultats de ces deux méthodes:

- **Test initial sur 2d-4c :**



Bonne detection des 4 clusters.

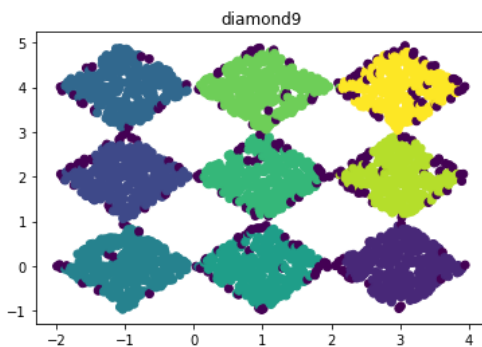
- **Sensible à la nature des formes: Dartboard1:**



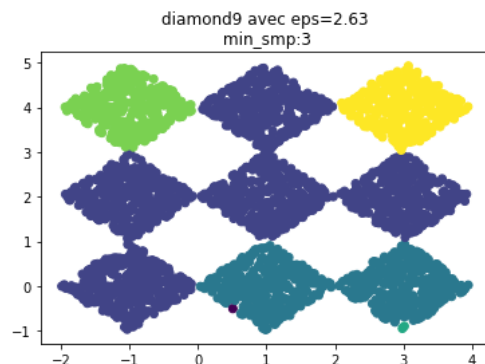
HDBSCAN arrive aussi à détecter les cercles concentriques de ce jeu de données, tout comme DBSCAN.

Nous ajoutons également le jeu de données **diamond9** pour tester avec une nouvelle forme. Les losanges sont eux aussi assez bien détectés. En comparaison, nous avons affiché le résultat sur le même jeu de donnée avec DBSCAN:

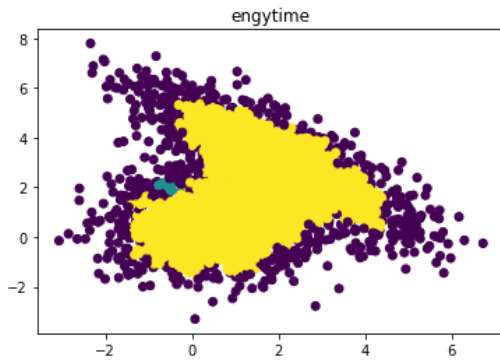
Résultat avec HDBSCAN



Résultat avec DBSCAN

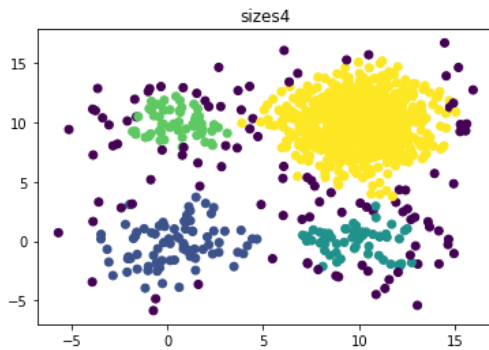


Ici, le résultat pour **engytime** (formes convexes mal séparées):



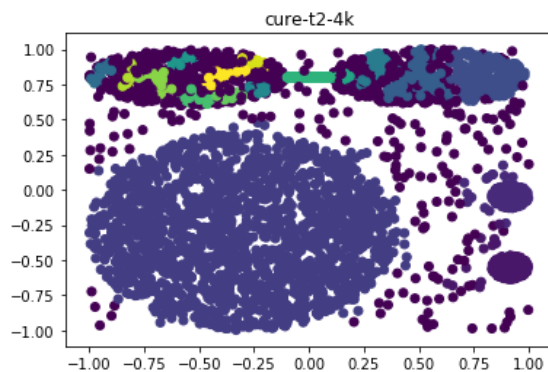
On voit que HDBSCAN ne délivre pas le résultat attendu car il ne détecte qu'un seul gros cluster. DBSCAN avait lui aussi beaucoup de mal avec ce jeu de donnée.

- **Sensibilité à la densité des données ? Sizes4**



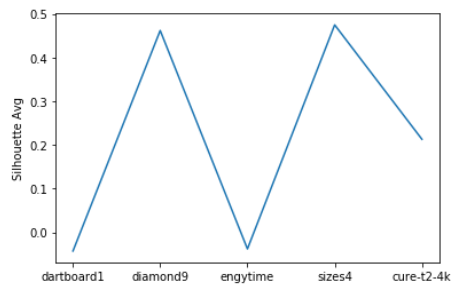
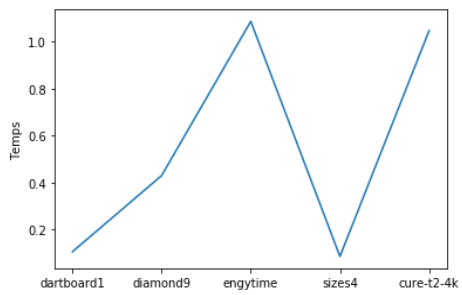
On voit ici aussi que HDBSCAN ne délivre pas le résultat attendu: les formes sont à densité variable et les parties peu denses des 4 clusters centraux sont considérées comme du bruit. En revanche, DBSCAN n'avait trouvé que deux clusters et du bruit.

- **Le bruit dans les données est-il bien identifié ? cure-t2-4k**

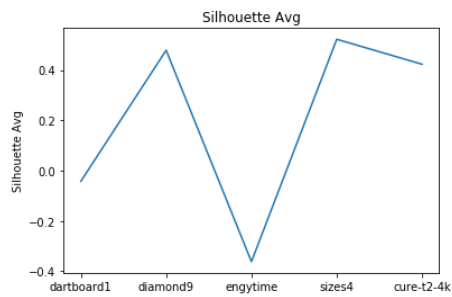
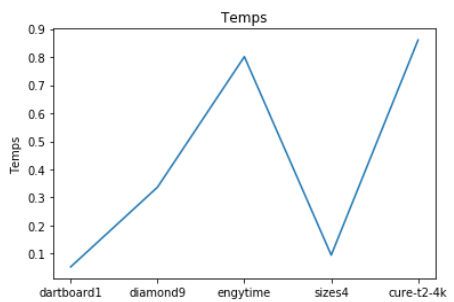


On voit que les formes sont très mal détectées. En comparaison, DBSCAN avait ici légèrement mieux détecté les formes en ne classant ni des parties deux ovales du haut ni les deux cercles de droites comme du bruit.

- **Les temps de calcul sont-ils différents ?**



Ci-contre les résultats pour HDBSCAN



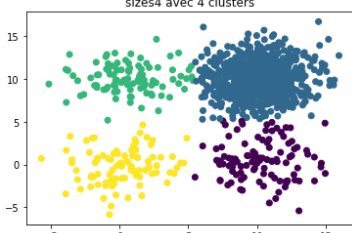
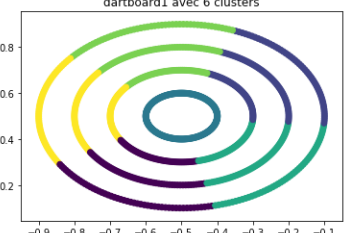
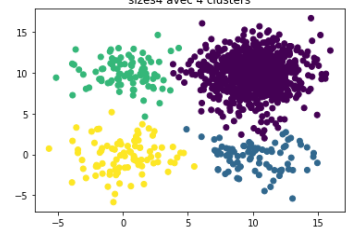
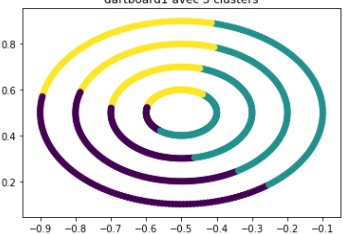
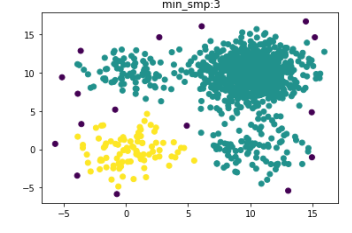
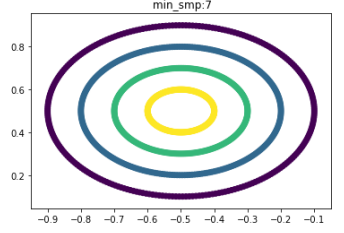
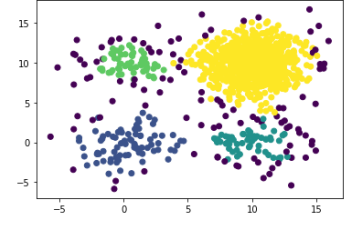
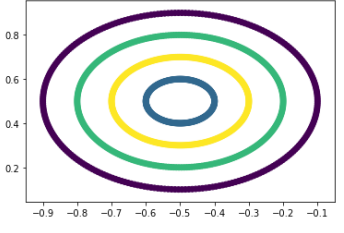
Ci-contre les résultats pour DBSCAN

DBSCAN est donc légèrement plus rapide que HDBSCAN

Synthèse

Durant cette étude, nous avons mis en évidence les sensibilités des différentes manières de clustering en les faisant chacune travailler sur 6 datasets différents. Ces datasets présentent un panel de formes convexes ou non, bien séparées ou non, à densité variable ou non, et présentant ou non du bruit.

Nous avons affiché ici nos principaux résultats sur **Sizes4** et **Dartboard1**, représentatifs des différences entre les algorithmes exploités.

Sizes4	Dartboard1	
		K-means: réussi facilement à séparer les densités variables mais pas les formes non convexes.
		Algomerative: même commentaire que pour KMeans.
		DBSCAN: pratique pour détecter les formes non convexes mais ne peut pas discerner les formes mal séparées
		HDBSCAN: écarte très facilement le bruit, et lui aussi détecte les formes non convexes

Nous avons fait un tableau récapitulatif pour représenter quelles types de données peuvent être classées avec les différents algorithmes:

	2d-4c: convexe + séparées + densité variable	Engytime: mal séparées + densité variable	Sizes4: densité variable	Dartboard1 : non convexe + formes bien séparé + densité similaire + non bruité	Cure-t2-4k: très bruité	Diamond9: sensibilité à la nature des formes
KMeans	✓	✗	✓	✗		
Agglomerative clustering	✓	✗	✓	✗		
DBSCAN	✓	✗	✗	✓	~	✗
HDBSCAN	✓	✗	~	✓	✗	✓

Ici on constate que K-means et Agglomerative clustering fonctionnent mieux sur des formes convexes, séparées et à densité variable. En revanche DBSCAN fonctionne aussi sur ce genre de formes mais également sur des formes non convexes et il réussit à peu près à discerner le bruit. Quand à HDBSCAN il fonctionne mieux que DBSCAN sur les formes à densité variable et il est également plus sensible à la nature des formes.

Ces différentes méthodes ont toutes des forces et faiblesses mais par exemple aucune d'entre elles n'a su correctement détecter les deux formes du dataset **Engytime**. Le clustering reste donc une méthode qu'il faut utiliser avec précaution et sans garantie de résultats. Le paramétrage est très important ainsi que la forme des données et il sera donc essentiel d'analyser et de prévisualiser ces données pour paramétrer en conséquence l'algorithme choisi.