

Rapport

Apprentissage non supervisé

Git repository: https://github.com/ColorPalmy/TP_Apprentissage_Non_Supervise.git

1. Jeux de données

En annexe 1 se trouvent les jeux de données que nous avons sélectionnés ainsi que leurs caractéristiques.

2. Clustering k-means

Nous avons commencé par exécuter k-means en lui donnant le bon nombre de clusters sur le dataset R15 (dans lequel il y a 15 clusters distincts). Le résultat est conforme à nos attentes, nous obtenons les mêmes clusters que ceux représentés sur le graphique.

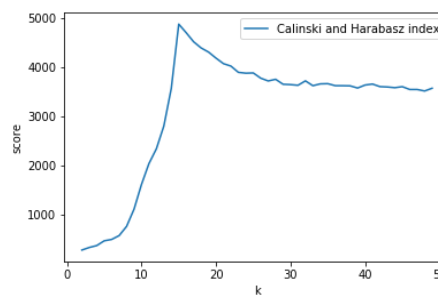
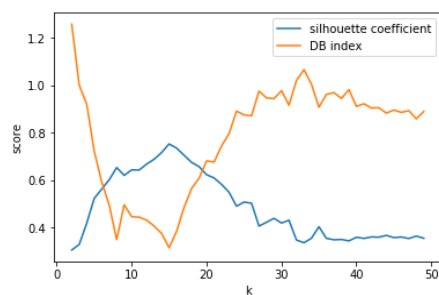
Afin de déterminer le meilleur k, on utilise une méthode itérative afin de trouver ce nombre. Pour savoir quelle est l'itération qui donne le meilleur résultat, on se base sur les critères suivants :

- l'indice de Davies Bouldin (DB index) : Il s'agit de moyenne du rapport maximal entre la distance d'un point au centre de son groupe et la distance entre deux centres de groupes. Il permet donc d'estimer la qualité des clusters en fonction de leur séparation et de leur homogénéité. Plus l'indice de Davies Bouldin est petit, mieux les clusters sont formés (séparés et homogènes). On cherche donc à minimiser cet indice.
- le coefficient de silhouette : calcul de la 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). Ce coefficient permet donc également de mesurer la qualité du partitionnement des données en clusters. Cet indice varie entre -1 (pire résultat) et 1 (meilleur résultat). On cherche donc à maximiser cette valeur.
- l'indice Calinski-Harabasz : c'est le rapport entre la variance inter-groupes et la variance intra-groupe, soit le ratio entre la dispersion des données entre les clusters et la dispersion des données dans les clusters. Vu que dans le cas idéal, les clusters sont bien séparés mais qu'au sein de ces clusters les valeurs doivent être les plus proches

possibles (clusters denses), ce sont les valeurs élevées de cet indice qui caractérisent un bon clustering. On cherche donc à maximiser cet indice.

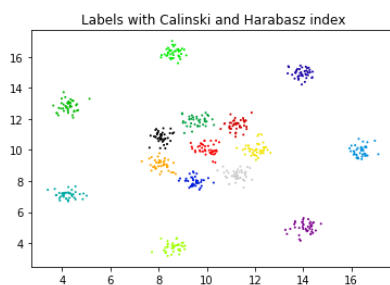
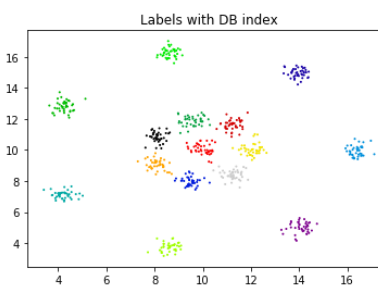
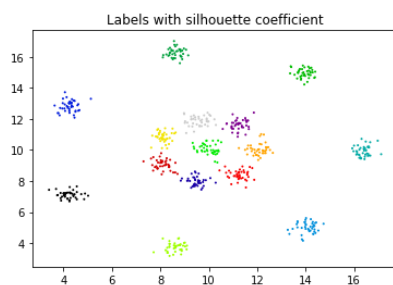
On calcule ces trois critères à chaque itération afin de trouver le meilleur partitionnement de données et donc la meilleure valeur de k , qui correspond au nombre de clusters.

Nous avons effectué une fonction itérative qui applique k-means au dataset R15 en faisant varier k entre 2 et 50. Pour chaque itération nous avons calculé les critères ci-dessus afin de déterminer la valeur idéale de k .



Pour chaque critère le nombre idéal est $k=15$, ce qui signifie que chacun des critères permet d'identifier le nombre de clusters attendu pour ce dataset.

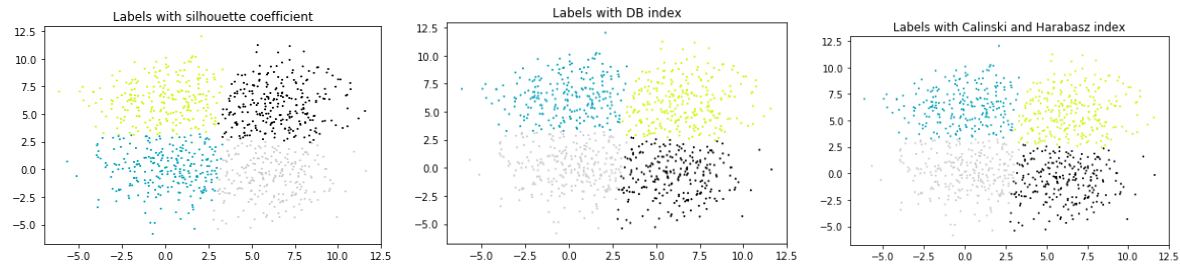
Nous avons représenté les clusters détectés grâce à chacun des critères en utilisant la fonction labels. Les trois critères permettent d'identifier correctement les 15 clusters attendus :



Nous avons ensuite effectué le même type d'expérimentations sur d'autres datasets ayant des caractéristiques différentes :

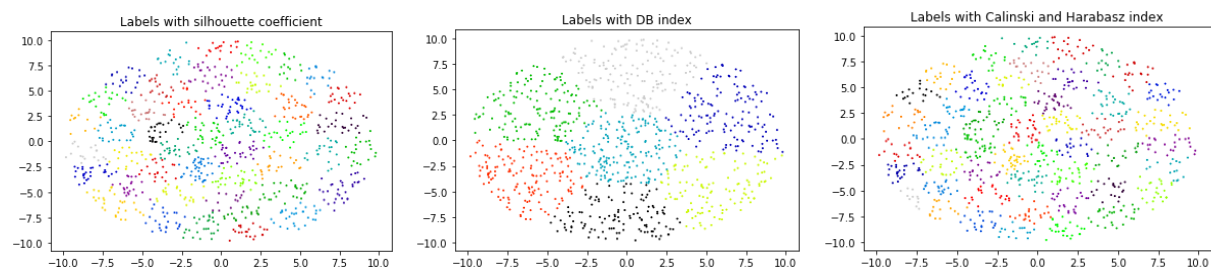
- Formes convexes mal séparées :

Nous avons utilisé le dataset square5 dont les formes (4 clusters) sont convexes, de taille équivalentes, mais mal séparées. Voici nos résultats :

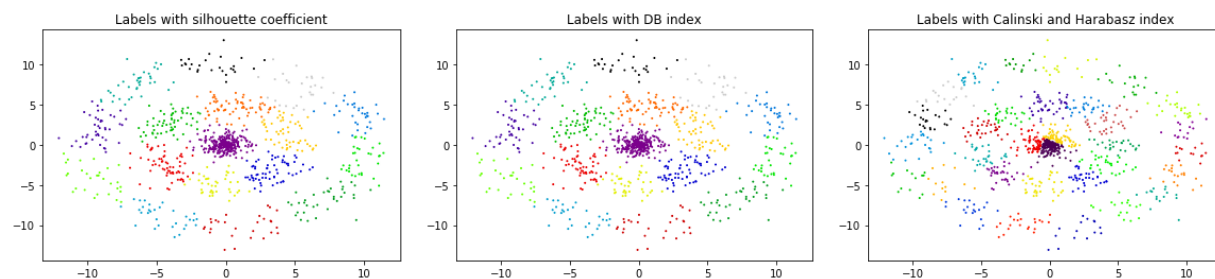


Les trois critères permettent de détecter correctement les 4 clusters attendus.

- Formes non convexes : Tests des clusters disk-1000n et rings

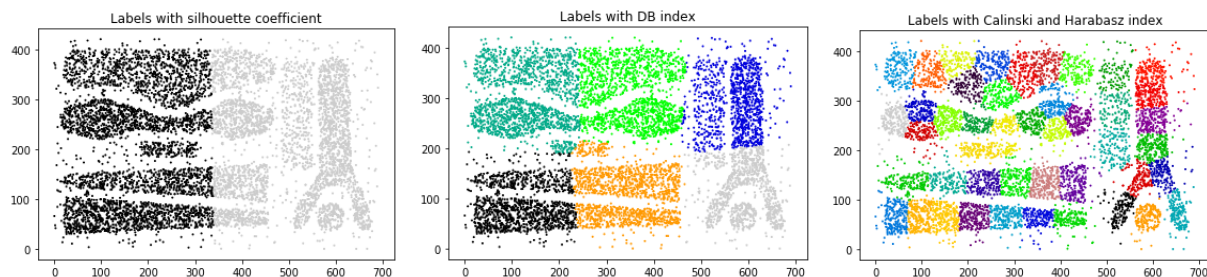


Le critère DB index détecte 7 clusters au lieu de 2. On constate que tous les points du cluster central convexe original sont dans le même cluster détecté mais que celui-ci est étendu sur certains points du cluster périphérique original, ce qui tend à montrer que k-means est plus efficace sur des clusters de tailles équivalentes, en particulier lorsqu'il sont mal séparés. Le cluster non convexe est scindé en plusieurs clusters convexes, ce qui nous laisse penser que la caractéristique non convexe rend l'identification des clusters difficile. Par ailleurs, on constate que les critères silhouette coefficient et CH index détectent respectivement 45 et 48 clusters. Ces résultats sont dus à la fois à la mauvaise séparation et à la densité équivalente des deux clusters.



On obtient 17 clusters avec les critères silhouette coefficient et DB index, et 32 clusters avec le critère CH index. Ces résultats sont très éloignés du résultat attendu, en particulier celui obtenu par le critère CH index, à cause de la propriété non convexe de deux des clusters originaux, de la différence de taille des clusters et de leur manque d'homogénéité.

- Formes de densité variable :




On obtient $k=2$ pour le critère silhouette coefficient, $k=6$ pour le critère DB index et $k=45$ pour le critère CH index. Aucun des critères ne permet de retrouver les clusters originaux. Ils forment tous des clusters dont on peut définir un centroïde. Aucun n'a une forme allongée. Les clusters détectés ne sont pas homogènes.

- Questions générales sur nos tests de K-means :

Nous n'arrivons pas systématiquement à retrouver le résultat attendu à l'aide de ces critères d'évaluation. Nous y sommes parvenus sur les datasets R15 et Square5, mais sur aucun des autres datasets testés.

A moins d'avoir une idée du nombre de clusters qu'on cherche, on est obligé de tester de nombreuses valeurs de k . En effet, si on étudie les courbes des différents critères dans le but de s'arrêter dès qu'on trouve le minimum (ou le maximum dans le cas du CH index), on risque de tomber sur un minimum ou un maximum local et donc de ne pas trouver la bonne valeur.

Il nous est arrivé de devoir relancer K-means à cause d'un minimum local faussant les résultats, ce qui peut arriver car les centroïdes initiaux sont choisis aléatoirement. Il s'agit bien d'une sensibilité à l'initialisation.



L'algorithme a de mauvais résultats sur les clusters non convexes ou bien de forme très allongée, qui n'ont pas vraiment de centre. Il a de meilleurs résultats sur les formes convexes basées sur un centroïde comme des disques ou des carrés, de préférence de tailles équivalentes ou très bien séparées. L'algorithme est également sensible à la densité et fonctionne mieux sur des clusters homogènes.

3. Clustering agglomératif

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

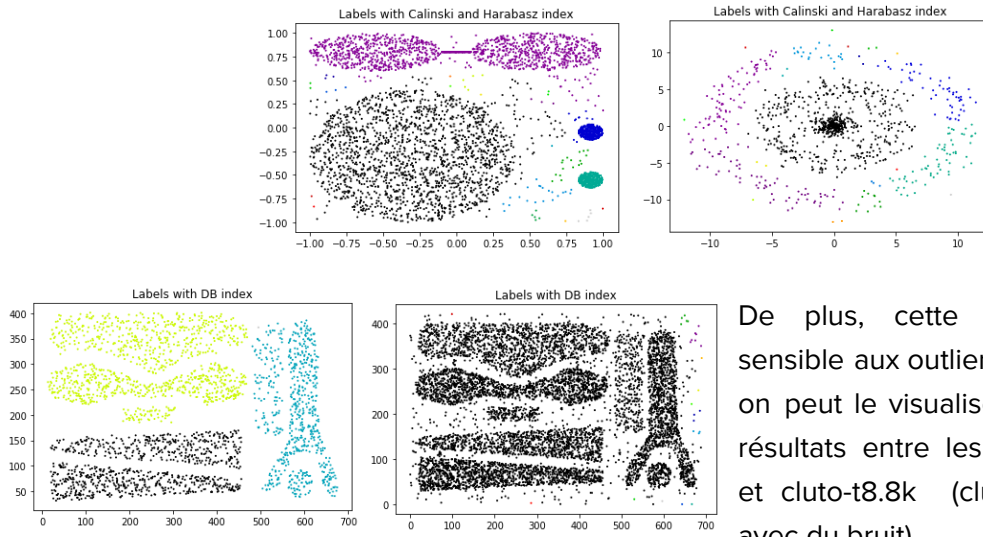
Il est difficile de limiter le nombre de valeurs différentes de k à tester car pour ce faire il faudrait s'appuyer sur les courbes d'évolution des scores obtenus avec nos 3 critères d'évaluation et arrêter l'exploration lorsque les scores cessent de s'améliorer. Or les scores ne varient pas de façon linéaire mais fluctuent, on ne peut pas distinguer un seuil à partir duquel le score ne s'améliore plus.

Une stratégie d'exploration des valeurs de k pourrait être de tester des valeurs réparties sur l'intervalle, sélectionner la valeur qui produit le meilleur score et tester des valeurs contiguës dans une plage autour de cette valeur sélectionnée. On pourrait ainsi espérer que la valeur optimale de k se trouve dans cet intervalle. Par exemple, tester des valeurs entre 0 et 50 avec un pas de 10 et calculer le score pour chaque métrique, et si le meilleur score est pour k=10 on pourrait évaluer les valeurs de k entre [1;19]. Cette approche devrait être adaptée en itérant le processus de sélection et d'évaluation pour essayer de couvrir globalement tout l'intervalle initial (sans tester toutes les possibilités). Ainsi, la recherche serait centrée sur des points "prometteurs" tout en ayant n'omettant pas des valeurs différentes.

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

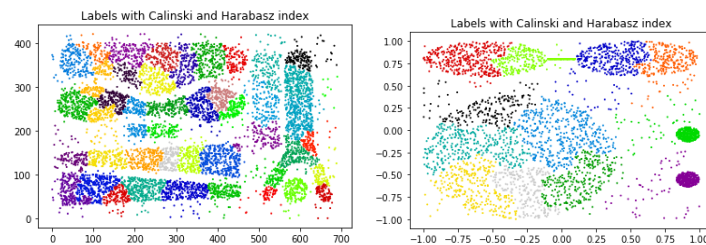
Selon la méthode de combinaison des clusters utilisée, on obtient des nombres et formes de clusters très différents. On remarque aussi que le bruit a un impacte sur la méthode utilisée.

De façon générale, on observe que la méthode **Single** ne permet pas d'obtenir les résultats attendus. Les clusters obtenus ont des densités variables, certains clusters ne comportent que quelques points alors que d'autres sont très denses, comme c'est le cas sur le dataset rings. Cette méthode a tendance à former des clusters en forme de chaîne car elle considère la distance minimale entre les points des clusters, donc les extrémités les plus proches des clusters. Ce phénomène est visible pour le dataset cure-t2-4k (avec le cluster violet) et pour le dataset rings pour lequel les clusters ont une forme allongée qui suit la forme de l'ellipse.

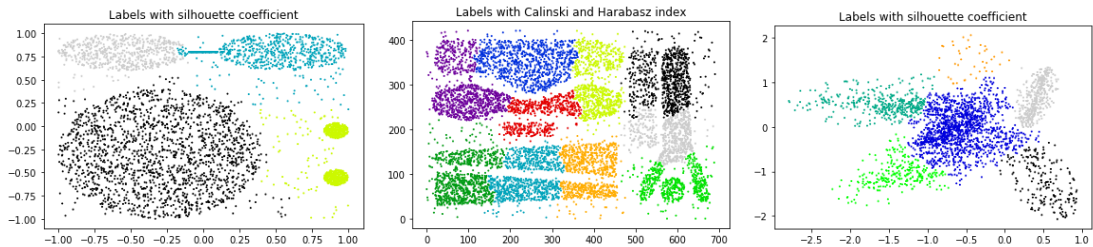


De plus, cette méthode est très sensible aux outliers et au bruit, comme on peut le visualiser en comparant les résultats entre les datasets complex8 et cluto-t8.8k (clusters de complex8 avec du bruit).

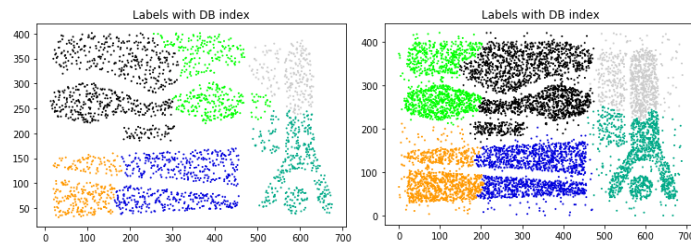
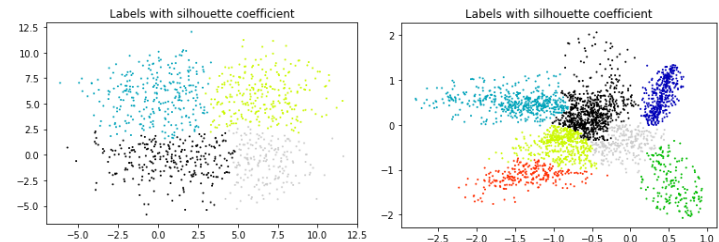
La méthode **Complete** permet d'obtenir de bons résultats lorsque les clusters sont bien séparés et de forme circulaire. Ainsi, elle sépare des clusters de taille importante en petits clusters, ce qui n'est pas toujours voulu, par exemple cluto-t8.8k ou cure-t2-4K. Les clusters formés sont compacts bien qu'ils regroupent du bruit.



La méthode **Average** permet de former des clusters de formes et de tailles différentes ce qui est pertinent pour plusieurs de nos datasets qui ne présentent pas des clusters de formes et tailles similaires. Ainsi, on obtient des résultats cohérents (bien que différents de ceux espérés) pour les datasets cure-t2-4K et cluto-t8.8k avec des clusters de formes variables (formes plutôt rectangulaires). Pour le dataset elly-2d10c13s, les clusters attendus se superposent alors que cela est contraire à la définition des clusters. On obtient avec cette méthode un partitionnement des données que l'on peut juger satisfaisant (il faudrait avoir une interprétation des données).



Avec la méthode **Ward**, les résultats obtenus sont proches de l'attendu pour les datasets qui ont des clusters de forme ronde tels que elly-2d10c13s (résultats que l'on peut juger satisfaisant) et square5 (résultat espéré).

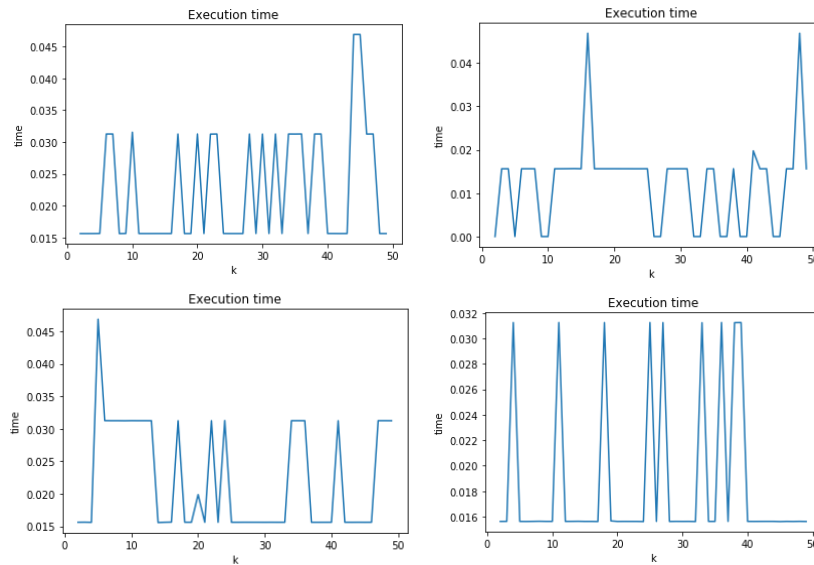


On remarque que cette méthode n'est pas sensible au bruit, comme on peut l'observer avec la formation de clusters similaires pour les datasets complex8 et cluto-t8.8k.

Cette méthode n'est pas vraiment adaptée pour des formes différentes de cercles ou amas (par exemple sur le dataset complex8).

Single	Complete	Average	Ward
Clusters de formes chaînées: rectangles, arcs de cercle, ... Identifie des formes convexes et non convexes Clusters de formes et tailles différentes Très sensible au bruit et aux outliers	Clusters de forme ronde/amas et tailles proches Pas pour les formes non convexes Tend à séparer des larges groupes de points en plusieurs clusters Peu sensible au bruit	Clusters de formes de natures et tailles différentes et de densités variables Pas pour les formes non convexes Assez sensible au bruit	Clusters de forme ronde/amas Pas pour les formes non convexes Clusters de taille assez proche Peu sensible au bruit

Le clustering agglomératif est efficace avec des temps d'exécution peu élevés mais donne des résultats plus ou moins pertinents selon la méthode de combinaison et le critère d'évaluation choisis. Les clusters peuvent avoir des densités variables (comme avec le dataset elly-2d10c13s), ce qui peut engendrer la prise en compte du bruit dans les clusters avec certaines combinaisons de clusters. Selon la méthode de combinaison, on peut former des clusters de toutes les formes, tailles et densités, et en particulier des formes non convexes avec la méthode Single.



Ci-contre les temps d'exécution avec les méthodes de combinaison Ward, Single, Complete et Average pour le dataset R15. On remarque qu'ils sont plus élevés que pour d'autres méthodes (tel que DBSCAN) mais restent raisonnables.

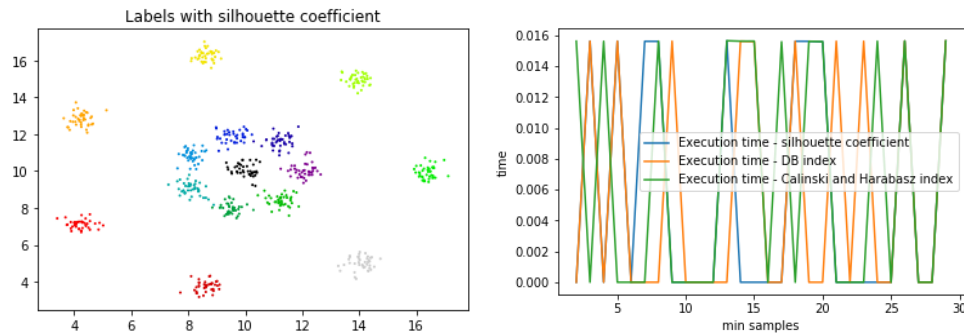
4. Clustering DBSCAN

Pour déterminer les bonnes valeurs de ϵ et min-samples, nous sélectionnons d'abord un intervalle de valeurs à tester pour chaque paramètre. Pour cela, nous affichons les données sur un graphique et estimons des minima et maxima pour ϵ et min-samples selon la forme des données et l'échelle. Nous itérons ensuite sur ces plages de données: on fixe une valeur de min-samples et on teste toutes les valeurs de l'intervalle pour ϵ pour trouver celle qui donne le meilleur résultat avec les 3 métriques considérées. Nous obtenons ainsi pour chaque métrique des couples (ϵ , min-samples) et nous conservons celui produisant le meilleur clustering.

Nous avons adopté la stratégie de faire varier ϵ et min-samples simultanément car ces deux paramètres ne sont pas indépendants (c'est un couple de valeur qui est optimal, une valeur d'épsilon est optimale pour une certaine valeur de min-samples et ne le sera pas pour d'autres). L'intervalle de valeurs testées pour min-samples est difficile à estimer et impacte fortement les résultats (meilleurs résultats en prenant des petits intervalles).

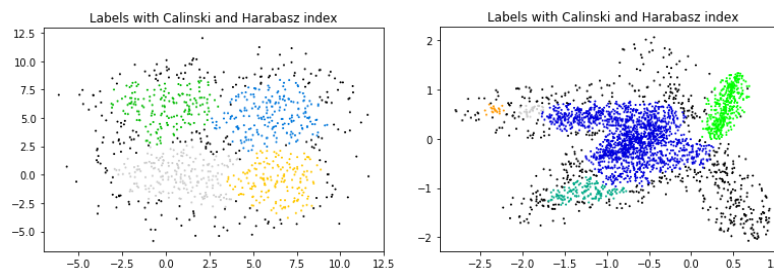
- Formes convexes bien séparées

On remarque que DBSCAN donne de très bon résultats sur des données convexes de densités similaires mais bien séparées (dataset R15). Le temps d'exécution est relativement faible sans présenter une évolution monotone selon les valeurs de min-samples.



- Formes convexes mal séparées

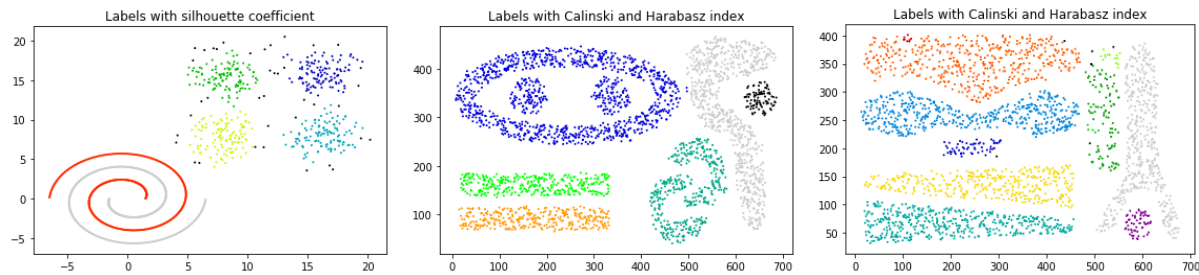
DBSCAN permet de former des clusters de formes convexes mais mal séparées telles que pour le dataset square5, bien que les points aux extrémités de ces clusters sont labellisé à tort comme des outliers. Cela peut s'expliquer par le fait qu'ajouter ces points extrêmes dans les clusters amènerait à la formation de clusters avec des densités variables (plus dense au centre qu'en bordure). Cependant, lorsque les densités des clusters sont variables et les données ne sont pas bien séparées, il n'arrive pas à identifier correctement les clusters, comme c'est le cas avec le dataset elly-2d5c13s.



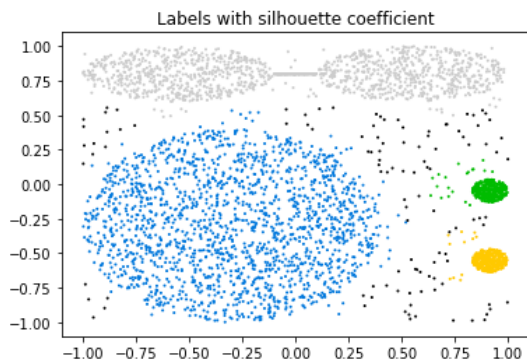
On peut donc conclure que DBSCAN distingue des clusters de forme convexe s'ils sont bien séparés ou s'ils sont mal séparés mais ont des densités homogènes.

- Formes non convexes

On note aussi que c'est une méthode efficace pour des données non convexes, comme c'est le cas avec les spirales dans le dataset spiralsquare ou avec les formes complexes (quelconques) dans le dataset complex8 et complex9. Par rapport aux résultats ci-dessus pour des clusters de densités différentes, on remarque que pour le dataset spiralsquare, les clusters sont bien identifiés malgré leurs densités très éloignées. On peut justifier ce phénomène par la séparation bien nette des clusters.



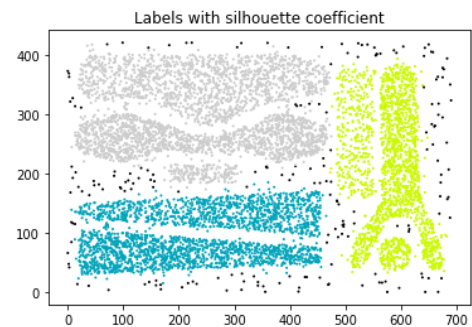
- Formes de densité variable



Les clusters de densités différentes sont bien identifiés avec cette méthode (dataset spiralsquare ci-dessus et cure-t2-4k). Cependant, les clusters comportant des densités variables ne sont pas correctement distingués car DBSCAN forme des clusters de densités similaires, comme on peut le voir au-dessus sur le dataset elly-2d5c13s.

- Présence de bruit

Une des particularité de DBSCAN est sa capacité à identifier des outliers et donc du bruit par rapport au clusters. Cela peut se visualiser dans la densité des clusters formés qui est assez uniforme. On remarque ainsi que pour les datasets cure-t2-4k (ci-dessus) et cluto-t8.8k le bruit est identifié comme outlier.



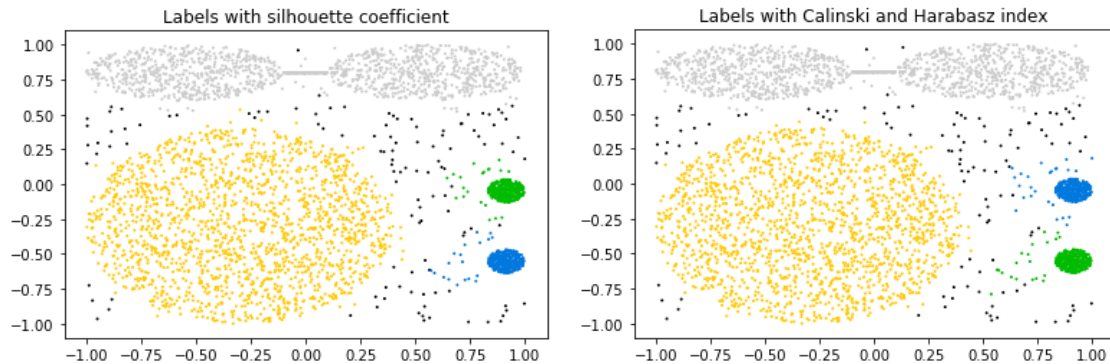
Un inconvénient de la méthode DBSCAN réside dans le choix délicat des paramètres *eps* et *min-samples* qui ont un impact important sur la performance de l'algorithme. En effet, selon le critère d'évaluation utilisé et l'intervalle de valeurs à tester pour *eps* et *min-samples*, les clusters formés varient beaucoup en nombre, forme et densité.

5. Clustering HDBSCAN

Nous avons repris les expérimentations effectuées avec l'algorithme DBSCAN sur les données bruitées (cluto-t8.8k) et les données à densité variable (elly-2d5c13s, spiralsquare et cure-t2-4k).

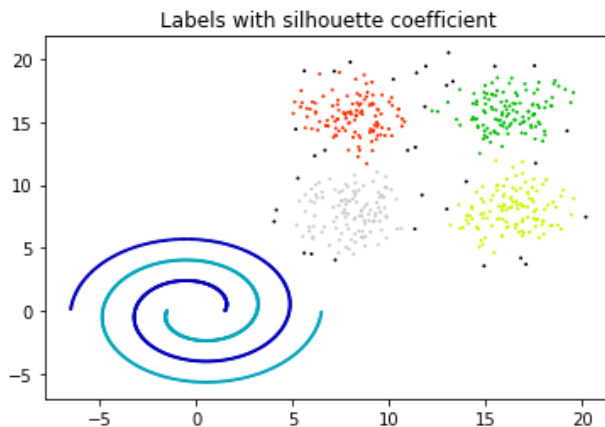
- Données à densité variable

Nous avons utilisé les datasets spiralsquare et cure-t2-4k. Voici les résultats pour le dataset cure-t2-4k :



Le résultat est analogue à celui obtenu par DBSCAN (sauf dans le cas du critère DB index qui ne permet pas de traiter correctement ce dataset). Les clusters de densité variable qui ne sont pas séparés ne sont pas correctement identifiés. Les autres clusters sont en revanche bien délimités.

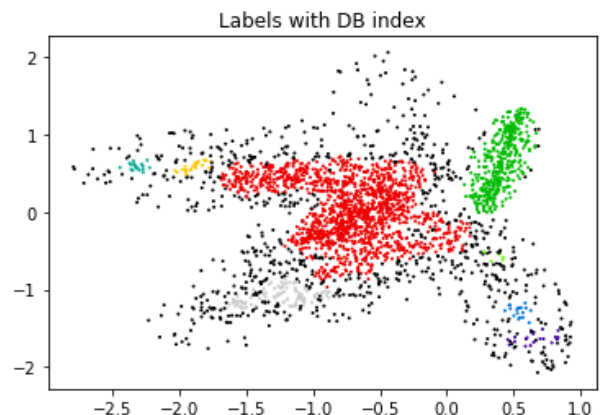
Voici nos résultats en ce qui concerne le dataset spiralsquare :



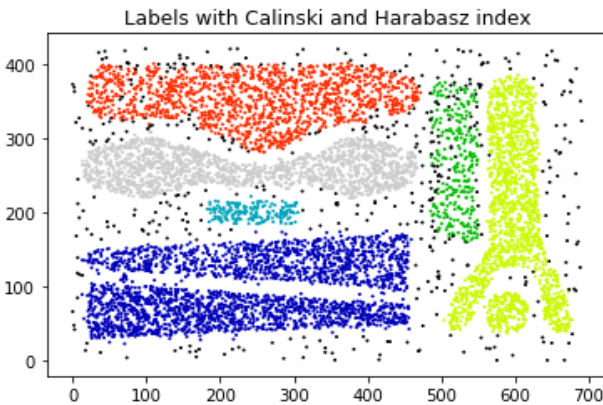
Dans le cas de ce dataset la meilleure métrique est le coefficient de silhouette . Les clusters sont bien identifiés. En revanche les résultats avec les deux autres critères sont moins bons. Le résultat est très similaire à celui obtenu avec DBSCAN, on constate que malgré leur différence de densité les clusters sont bien identifiés, autant convexes que non convexes.

Enfin, nous avons testé le dataset elly-2d5c13s :

On constate que de façon analogue à DBSCAN, HDBSCAN ne permet pas de différencier ces clusters de densité variable et mal séparés.



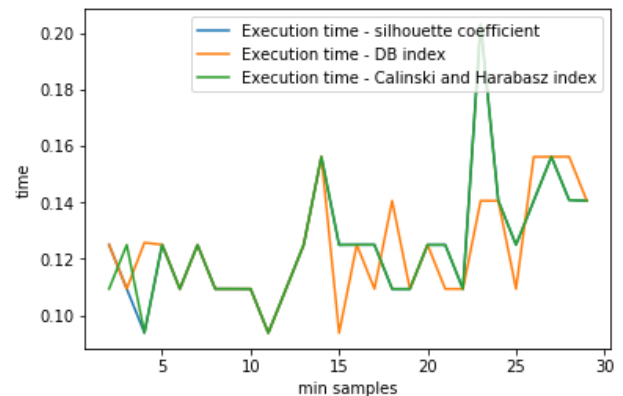
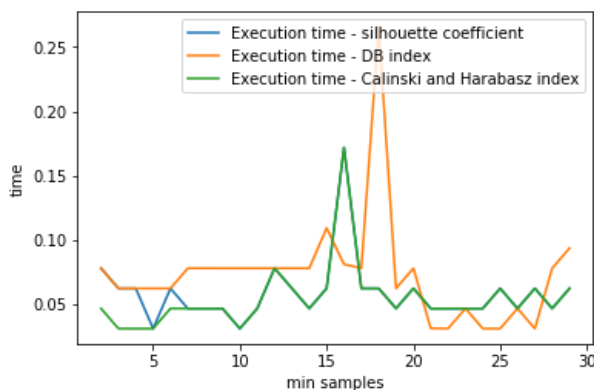
- Données bruitées



Sur le dataset cluto-t8.8k, on obtient de très bons résultats avec le critère CH index. Comme pour DBSCAN, HDBSCAN identifie correctement les clusters convexes et non convexes. On identifie et on délimite correctement 6 clusters, les deux restants étant bien délimités mais confondus. Pour les deux autres métriques les résultats sont nettement moins bons. On remarque que le bruit est bien identifié comme tel. Le résultat

rendu par le critère CH index est meilleur que celui de DBSCAN, à cela près que certaines données à l'intérieur des clusters sont identifiées comme du bruit. HDBSCAN semble moins sensible que DBSCAN aux formes très allongées et proches d'autres clusters.

Pour finir, nous avons comparé les temps de calcul, voici l'exemple du dataset cure-t2-4k (à gauche DBSCAN et à droite HDBSCAN) :



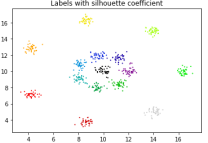
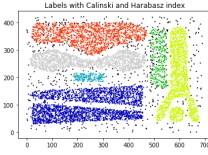
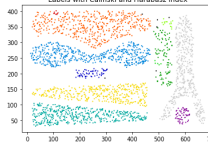
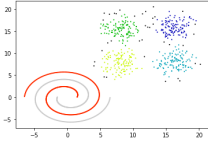
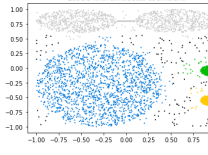
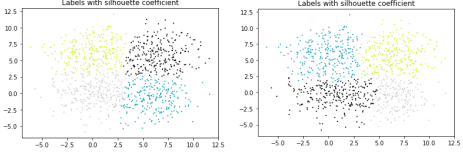
On constate tout d'abord que les échelles sont très proches donc aux variations liées aux métriques près, les temps d'exécution sont très similaires. On constate également que les temps d'exécution des trois métriques sont très proches sauf pour certaines valeurs de min-samples pour lesquelles une variation plus importante du temps d'exécution est enregistrée.

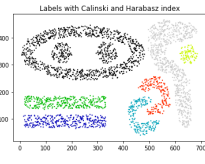
6. Synthèse

Méthode	Formes convexes / non convexes	Formes bien / mal séparées	Densités similaires / variables	Présence / Absence de bruit
Kmeans	+ : formes convexes basées sur un centroïde - : ne fonctionne pas sur les formes non convexes ou trop allongées	+ : fonctionne sur clusters mal séparés si leur centroïde est très clairement identifiable (formes rondes ou carrées) et si les clusters sont de la même taille	- : fonctionne mieux avec des clusters homogènes et de taille équivalente, il est sensible à la variation de densité et de taille des clusters	- : sensible au bruit et aux outliers qui perturbent la délimitations des clusters et déplacent le centroïde
Méthode Agglomérative	+ : formes convexes (méthodes Complete, Average et Ward) + : formes non convexes avec la méthode Single + : clusters de formes, tailles et densités différentes avec la méthode Average	+ : identifie des clusters mal séparés (avec Ward) si les densités sont différentes - : si densités similaires et clusters proches, si des clusters sont formés ils ne sont pas ceux attendus	+ : clusters de densités variables	- : assez sensible au bruit (surtout Single) et particulièrement aux outliers (qui sont mis dans des clusters)
DBSCAN	+ : formes convexes et non convexes + : clusters de formes et tailles différentes	- : clusters qui se chevauchent ou pas de séparation distincte entre eux et densité différente	+ : clusters de densité similaire ou différente - : pas pour des clusters non homogènes	+ : détection d'outliers - : certaines extrémités de clusters sont identifiés comme du bruit
HDBSCAN	similaire à DBSCAN	similaire à DBSCAN	+ : aussi bon voire meilleur que DBSCAN dans le cas d'une variation de densité intra-cluster	+ : détection des outliers et du bruit - : certaines valeurs intra-clusters sont identifiées comme du bruit.

Il faut aussi noter qu'un inconvénient des méthodes Kmeans et de clustering Agglomératif est la nécessité de donner un nombre de clusters "attendus", ce qui n'est pas le cas avec DBSCAN et HDBSCAN. En revanche, ces 2 algorithmes imposent le choix des paramètres *eps* et *min-samples* qui sont difficiles à ajuster. On peut aussi considérer les temps d'exécutions et de paramétrage qui varient selon les méthodes et les formes des datasets.

Meilleures méthodes pour nos datasets:

Datasets	Meilleures méthodes	Clusters obtenus
R15 $k^* = 15$	Kmeans Agglomératif avec méthodes Average, Ward et Complete DBSCAN (eps= 0.7 et min-samples= 29)	 $k = 15$
cluto-t8.8k $k^* = 8$	HDBSCAN avec le coefficient de silhouette ou l'indice de Calinski et Harabasz (eps= 0,01 et min-samples= 9)	 $k = 7$
complex8 $k^* = 8$	DBSCAN avec l'indice de Calinski et Harabasz (eps= 14,9 et min-samples= 5)	 $k = 12$
spiralsquare $k^* = 6$	DBSCAN avec le coefficient de silhouette ou l'indice de Calinski et Harabasz (eps= 0.95 et min-samples= 5)	 $k = 7$ (6 clusters + outliers)
cure-t2-4k $k^* = 6$	DBSCAN avec le coefficient de silhouette ou l'indice de Calinski et Harabasz (eps= 0.1 min-samples= 9) HDBSCAN avec l'indice de Calinski et Harabasz (eps= 0.1 et min-samples= 15)	 $k = 5$ similaire avec HDBSCAN
square5 $k^* = 4$	Kmeans Agglomératif avec la méthode Ward	 Kmeans - Agglomeratif $k = 4$

<p>complex9 k* = 9</p>	<p>Agglomératif avec méthode Single et indice de Calinski et Harabasz</p>	<div data-bbox="974 226 1177 388">  </div> <p>k = 7</p>
----------------------------	---	---

Pour les datasets elly-2d10c13s, disk-1000n et rings, aucun des algorithmes avec les paramètres testés ne donne un clustering proche de l'attendu (ci-dessous les résultats pour les datasets disk-1000n avec Kmeans et elly-2d10c13s avec Agglomératif avec Ward). Il faudrait appliquer des algorithmes optimisés pour déterminer les paramètres à fournir aux méthodes de clustering pour obtenir les clusters voulu.

