Analyse descriptive et predictive
Apprentissage non supervisé

Compte rendu - TP Clustering

Étudiant :
Ralaimady SETA

Enseignant :
Marie-José HUGUET

Table des matières

1 Introduction **2**

2 Clustering K-Means **3**
2.1 Scores de regroupement et de séparation . . . 3
2.1.1 square1.arff . . . 3
2.1.2 xclara.arff . . . 4
2.2 Application itérative de la méthode k-Means basée sur l'inertie . . . 5
2.3 Application itérative de la méthode k-Means basée sur des métriques d'évaluation externes 6
2.4 Intérêts et Limites de la méthode k-Means . . . 7
2.4.1 Jeux de données pour lesquels k-Means fonctionne correctement . . . 7
2.4.2 Jeux de données pour lesquels k-Means ne fonctionne pas correctement . . . 8
2.4.3 Analyse de la méthode k-Means . . . 8
2.5 Version mini-batch de la méthode k-Means . . . 9

3 Clustering agglomératif **10**
3.1 Impacts des différentes variantes de linkage . . . 10
3.1.1 Jeux de données initiales . . . 10
3.1.2 Application du clustering agglomératif avec différentes variantes de linkage . . . 11
3.2 Application itérative de la méthode agglomérative pour la recherche du meilleur nombre de clusters . . . 13
3.3 Intérêts et Limites du clustering agglomératif . . . 14

4 Clustering DBSCAN et HDBSCAN **15**
4.1 Clustering DBSCAN . . . 15
4.1.1 Tests de différentes valeurs de paramètres . . . 15
4.1.2 Application de la méthode DBSCAN . . . 16
4.1.3 Intérêts et Limites de la méthode DBSCAN . . . 17
4.2 Clustering HDBSCAN . . . 19
4.2.1 Comparaison HDBSCAN et DBSCAN . . . 19
4.2.2 Limites du clustering HDBSCAN . . . 19

5 Conclusion **20**

1 Introduction

Dans le cadre du cours d'Analyse descriptive et prédictive de ce premier semestre de cinquième année Systèmes Distribués et Big Data, je me suis penché sur l'étude de plusieurs jeux de données afin de tester et comparer différents algorithmes de clustering.

L'idée était d'appréhender les différents algorithmes sur des jeux de données tests connus afin d'ensuite appliquer les mêmes algorithmes sur des jeux de données à analyser.

Mon projet peut se retrouver sur un dépôt git au lien suivant :
https ://github.com/MadySeta/TP_Clustering.git

2 Clustering K-Means

2.1 Scores de regroupement et de séparation

2.1.1 square1.arff

Voici le résultat de la méthode *k-Means* sur le jeu de données *square1.arff* avec k le nombre de clusters égal à 4.

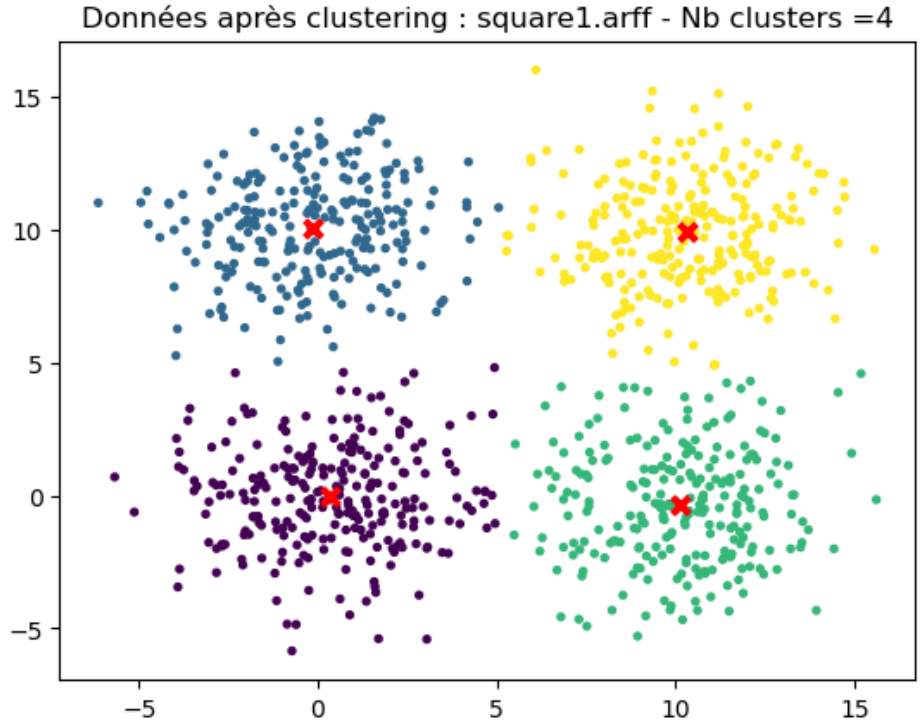


FIGURE 1 – Résultat de k-Means avec k=3 sur le jeu de données square.arff

	Score de regroupement	Score de séparation
Distance minimale	0.131	1.874
Distance maximale	6.833	22.421
Distance centre	2.571	11.571

TABLE 1 – Scores de regroupement et de séparation du clustering du jeu de données square.arff

2.1.2 xclara.arff

Voici le résultat de la méthode *k-Means* sur le jeu de données *xclara.arff* avec k le nombre de clusters égal à 3.

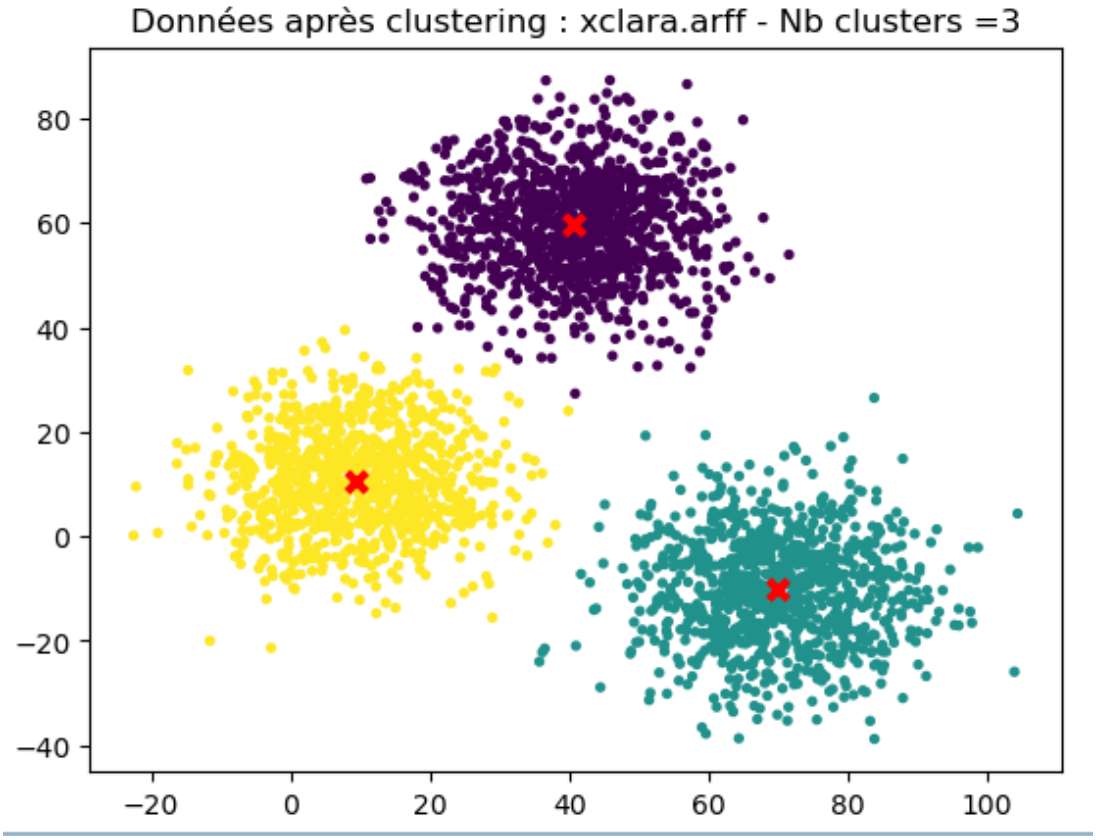


FIGURE 2 – Résultat de k-Means avec k=3 sur le jeu de données xclara.arff

	Score de regroupement	Score de séparation
Distance minimale	0.558	7.543
Distance maximale	36.255	131.065
Distance centre	12.685	65.918

TABLE 2 – Scores de regroupement et de séparation du clustering du jeu de données xclara.arff

2.2 Application itérative de la méthode k-Means basée sur l'inertie

Afin de déterminer le meilleur nombre de clusters k, on va appliquer itérativement la méthode K-Means sur le jeu de donnée *xclara.arff* pour différentes valeurs de k pour comparer leurs inerties.

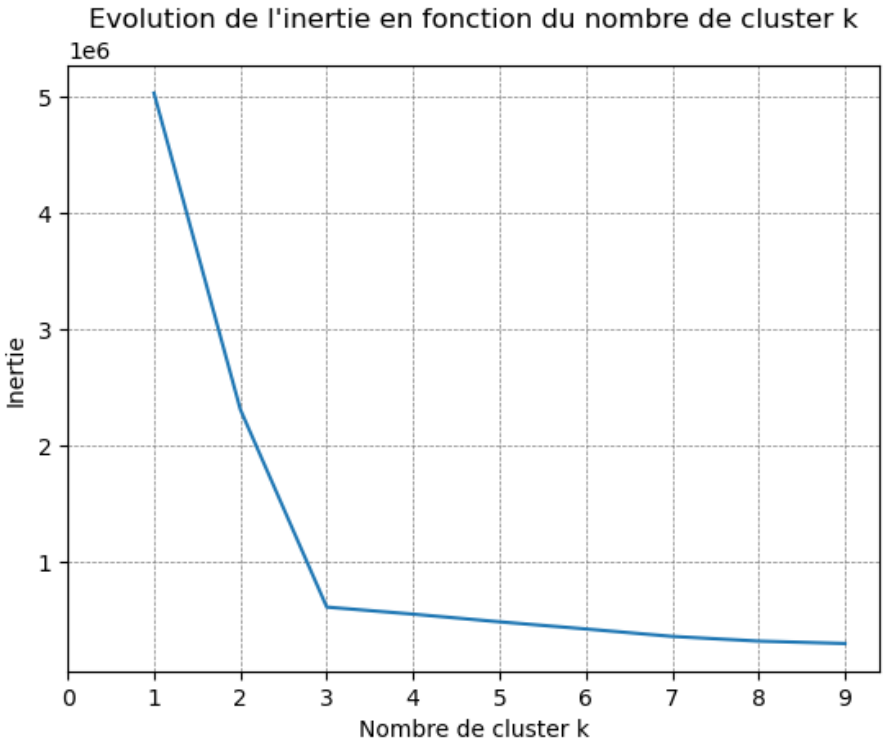


FIGURE 3 – Application itérative de k-Means sur le jeu de données xclara.arff

Pour déterminer une bonne solution de clustering, il faut considérer le point d'inflexion. Pour cela, on peut utiliser la méthode du coude. Ici, le point d'inflexion est lorsqu'on a un k=3. Ce résultat correspond à ce qu'on peut identifier visuellement.

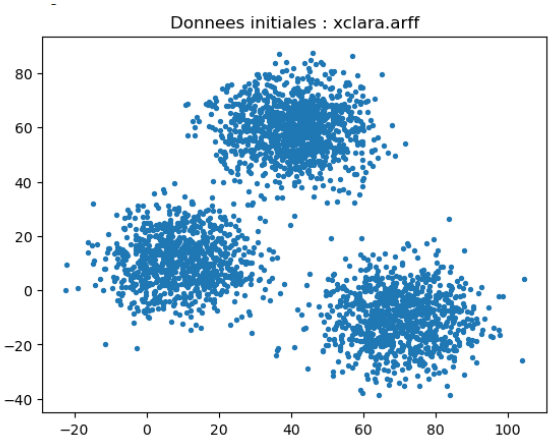


FIGURE 4 – Jeu de données xclara.arff avant clustering

2.3 Application itérative de la méthode k-Means basée sur des métriques d'évaluation externes

Comme métrique d'évaluation, j'ai choisi le coefficient de silhouette. Compris entre [-1,1], ce coefficient combine 2 mesures : la distance moyenne entre chaque élément et le reste des éléments de son cluster ainsi que la distance moyenne minimale de l'élément aux éléments des autres clusters.

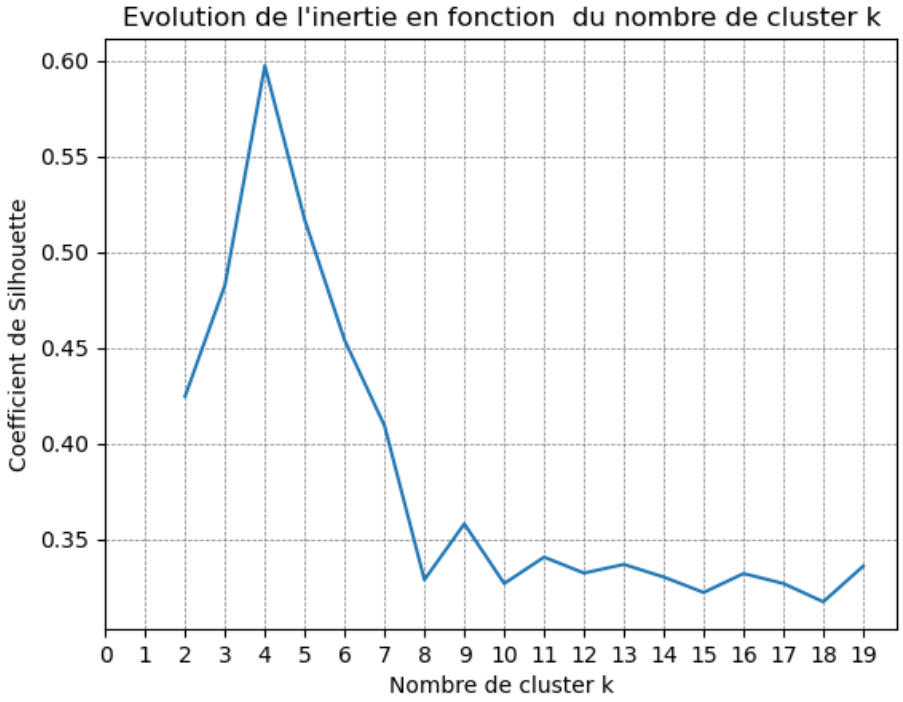


FIGURE 5 – Enter Caption

On peut clairement voir sur ce graphique que le meilleur nombre de clusters pour le jeu de données *square1.arff* est égale à 4. Le temps de calcul total est de 39.02 ms. Ce résultat correspond à ce qu'on peut identifier visuellement.

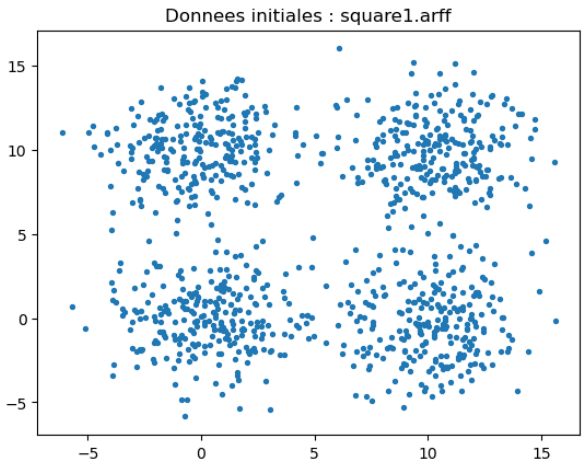


FIGURE 6 – Enter Caption

2.4 Intérêts et Limites de la méthode k-Means

Afin de mettre en évidence les intérêts et limites de la méthode k-Means, je vais l'appliquer sur 4 jeux de données dont deux d'entre eux pour lesquelles k-Means identifie correctement les clusters et deux autres pour les lesquelles k-Means n'arrive pas à identifier correctement les clusters.

Afin de déterminer le nombre de clusters, j'ai appliqué la même méthode itérative que dans la section précédente avec le coefficient de silhouette.

2.4.1 Jeux de données pour lesquels k-Means fonctionne correctement

— s-set1 avec k=15

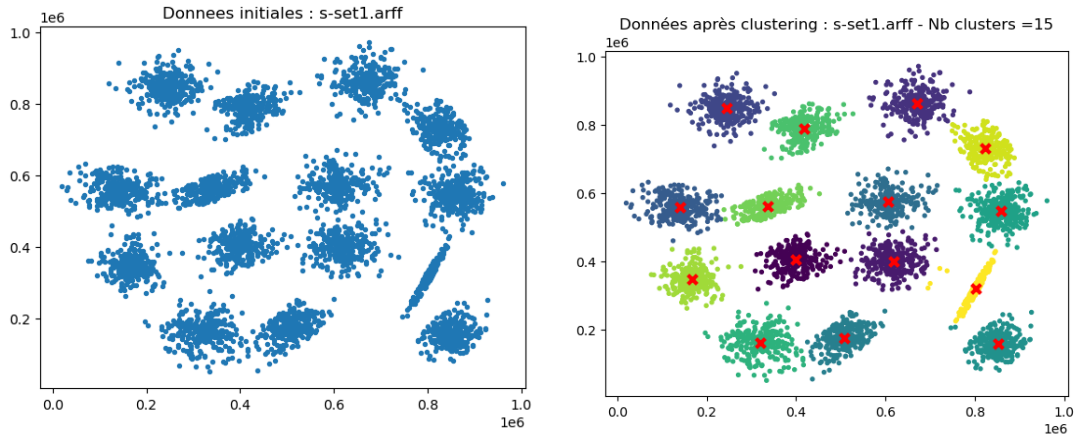


FIGURE 7 – K-Means appliquée à s-set1.arff avec k=15

— diamond9 avec k=9

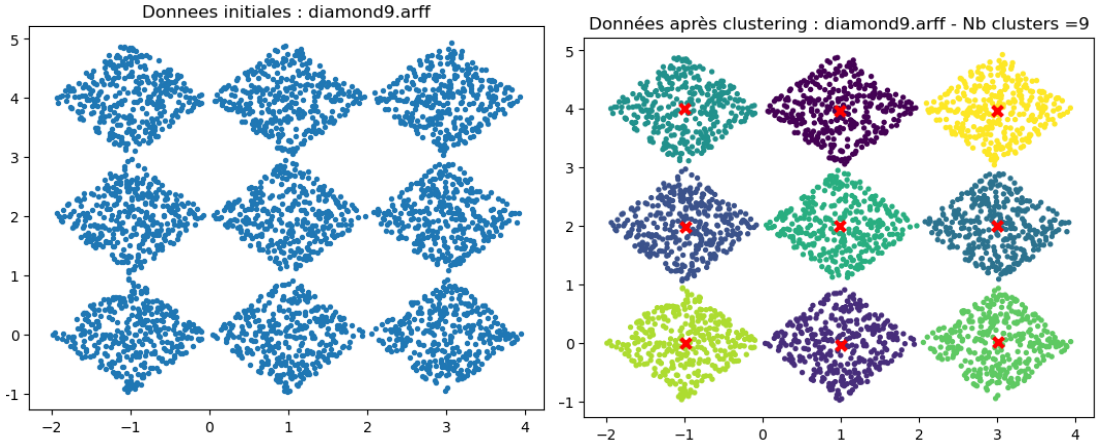


FIGURE 8 – K-Means appliquée à diamond9.arff avec k=9

2.4.2 Jeux de données pour lesquels k-Means ne fonctionne pas correctement

— spiral

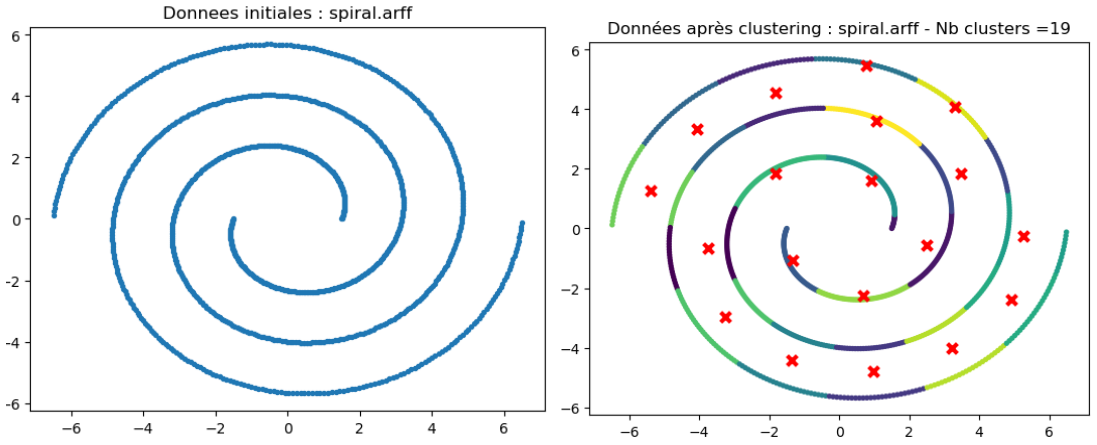


FIGURE 9 – K-Means appliquée à spiral.arff avec k=15

— dartboard

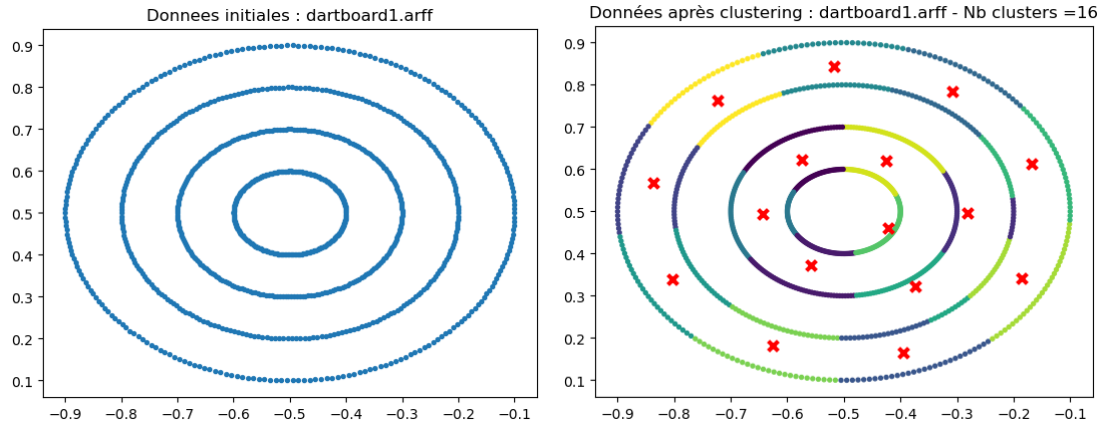


FIGURE 10 – K-Means appliquée à dartboard.arff avec k=9

2.4.3 Analyse de la méthode k-Means

On peut donc conclure que la méthode k-Means ne fonctionne pas correctement quand les clusters n'ont pas une forme convexe. Comme pour les jeux de données *dartboard* et *spiral*, les données partageant le même centre de gravité empêchent la méthode k-Means de faire un clustering correct.

En plus de ce frein, l'initialisation a un impact sur le développement de l'algorithme. Les centres des clusters choisis avant le lancement de l'algorithme auront un impact sur les clusters définis par la suite. De plus, l'initialisation requiert la définition du nombre de clusters a recherché au préalable ou une itération sur le nombre de clusters afin d'avoir les meilleurs résultats (ce qui est du temps de calcul en plus).

2.5 Version mini-batch de la méthode k-Means

Pour analyser l'impact du batchsize sur le clustering, j'ai pris en approche itérative en regardant l'évolution du coefficient de silhouette.

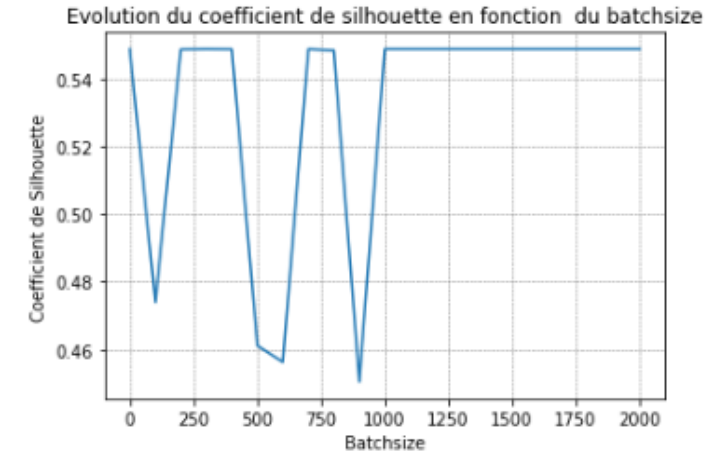


FIGURE 11 – Evolution du coefficient de silhouette en fonction du batchsize sur le jeu de données diamond9.arff

On peut remarquer que le coefficient est à son maximum pour plusieurs valeurs du batchsize. Il faut cependant faire attention à prendre une de ces valeurs à coefficient de silhouette maximal sinon les résultats du clustering sont complètement faussés même avec le bon nombre de clusters.

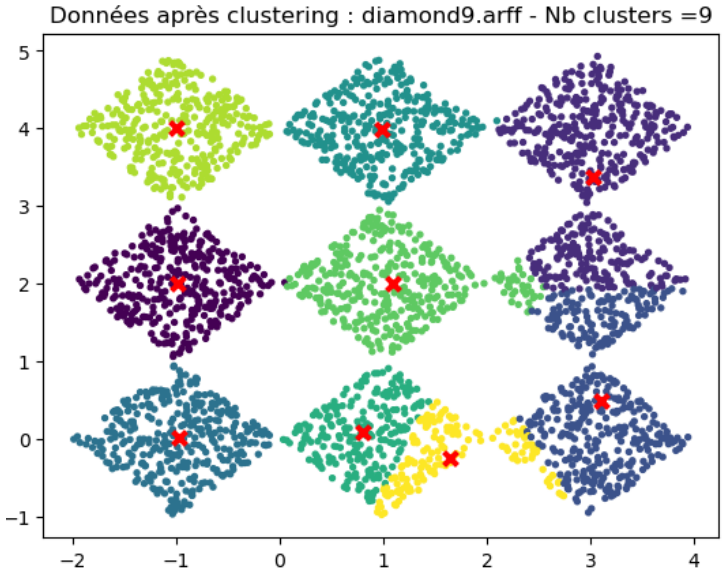


FIGURE 12 – Version Mini-batch appliquées sur le jeu de données diamond9.arff avec un batchsize égal à 2000

Après avoir tester la version Mini-batch de la méthode k-Means sur les jeux de données *spiral et dartboard1*, j'ai conclut que cette version ne résout pas le problèmes de non-convexité des données de la version classique de K-Means.

3 Clustering agglomératif

3.1 Impacts des différentes variantes de linkage

Pour analyser l'impact du linkage sur le clustering, nous allons tester les différents linkage (single, complete, average et ward) sur deux jeux de données (square1.arff et 3-spiral.arff). Pour mes analyses, je n'ai pas fixé le nombre de clusters pour permettre au paramètre *distance_ threshold* de prendre la valeur à 0 et donc de faire le clustering sur tout l'arbre.

3.1.1 Jeux de données initiales

— square1.arff

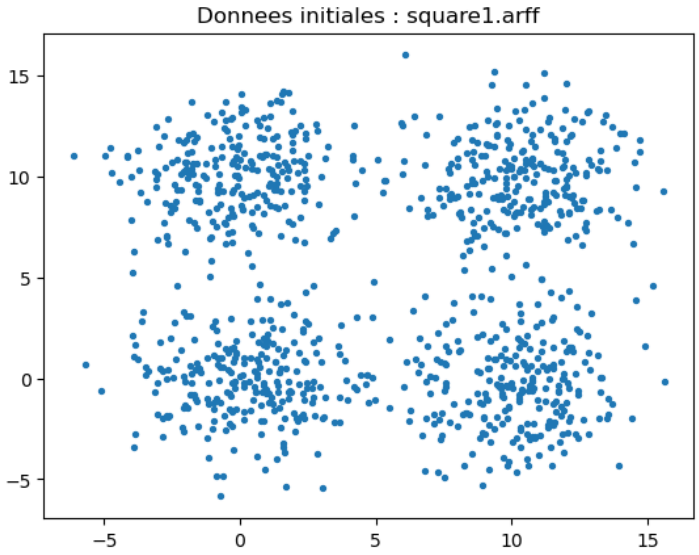


FIGURE 13 – square1.arff

— 3-spiral.arff

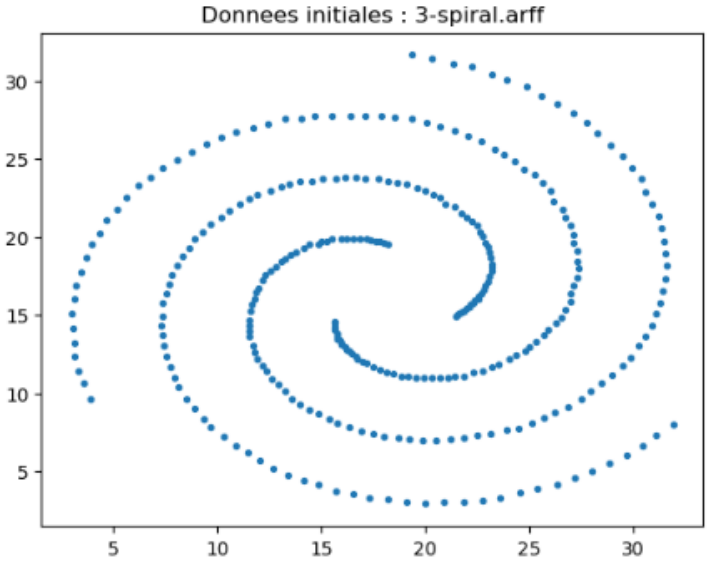
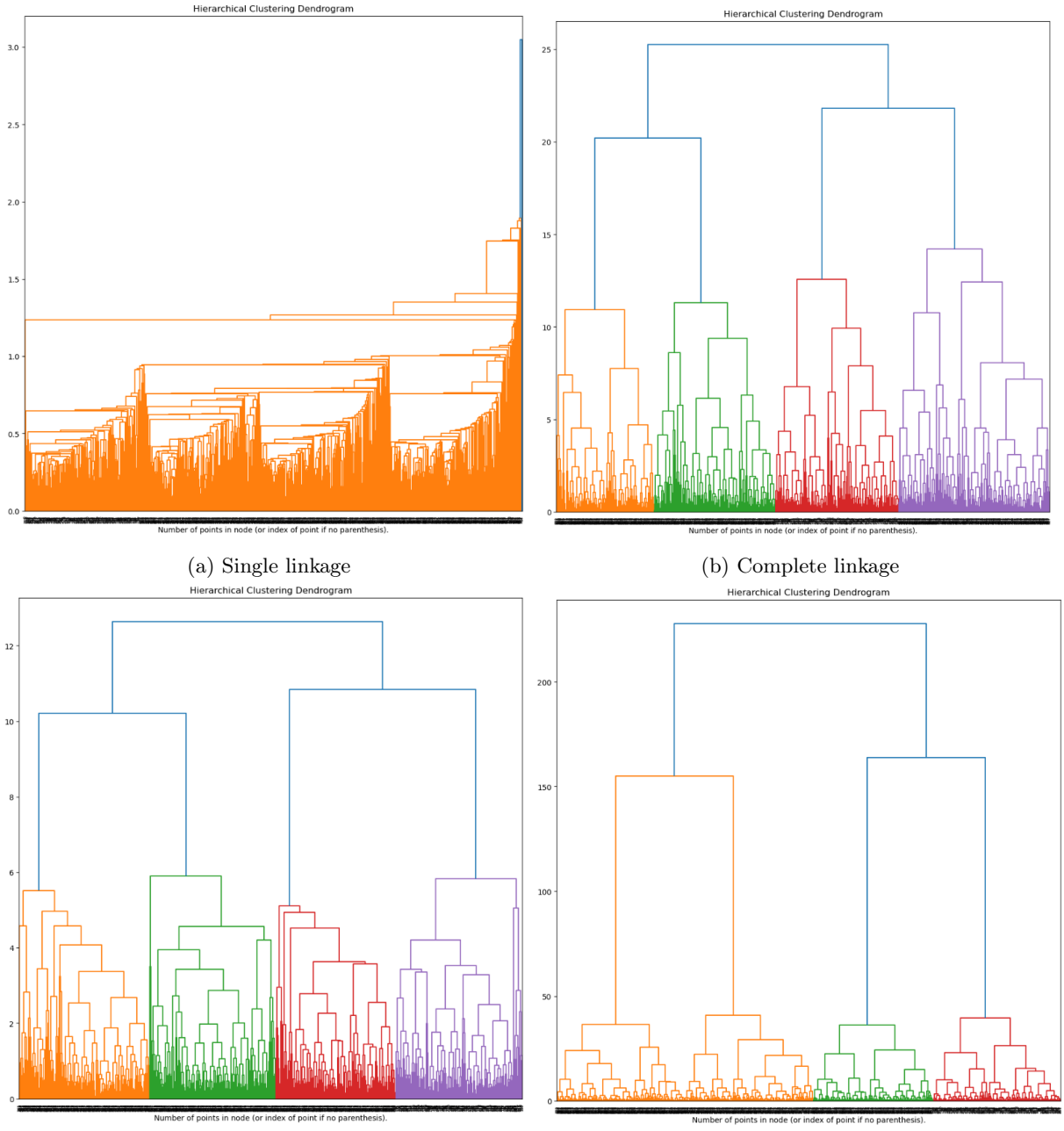


FIGURE 14 – 3-spiral.arff

3.1.2 Application du clustering agglomératif avec différentes variantes de linkage

— square1.arff (k=4)



(a) Single linkage

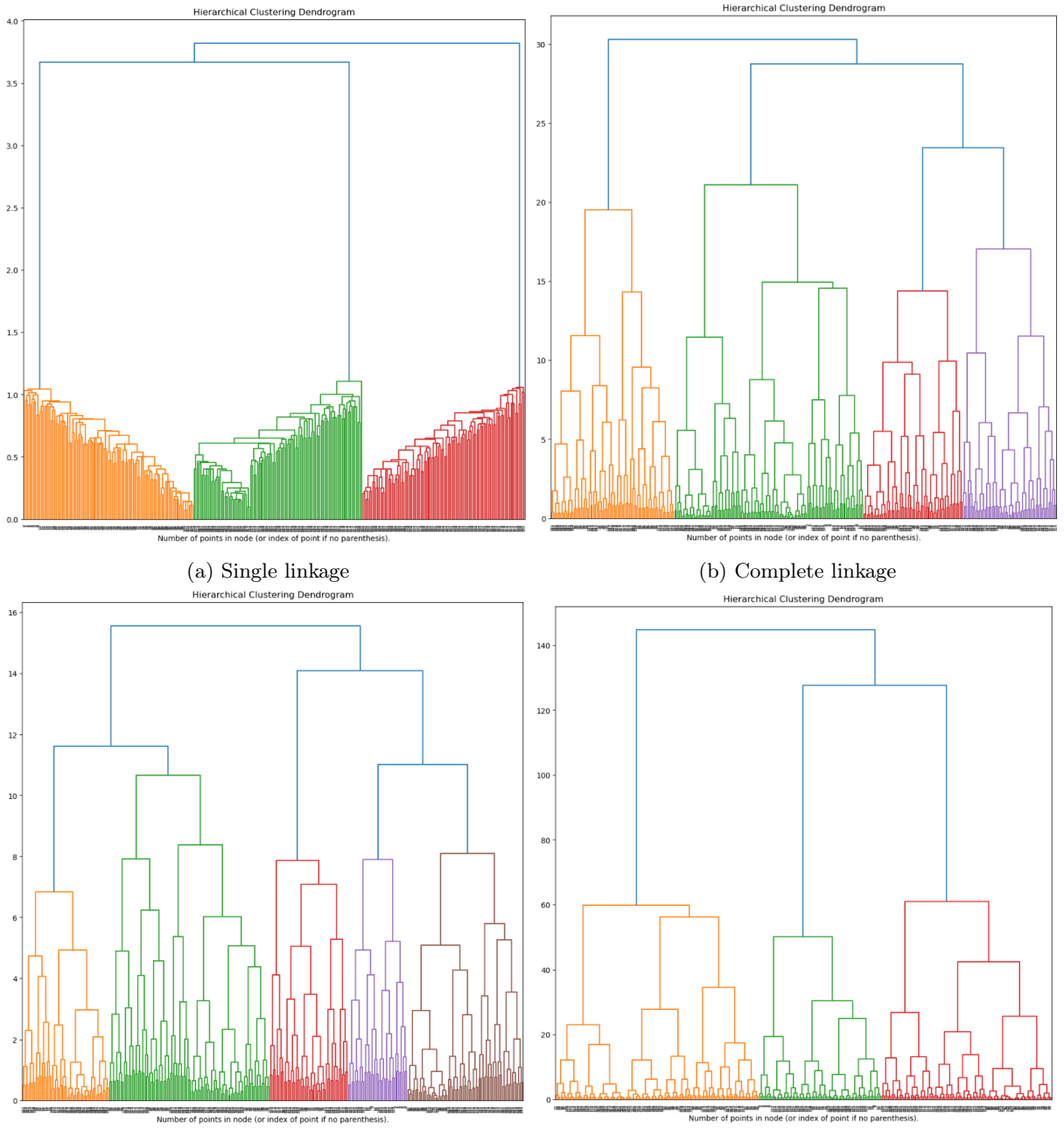
(b) Complete linkage

(c) Average linkage

(d) Ward linkage

FIGURE 15 – Clustering agglomératif sur square1.arff

— 3-spiral.arff (k=3)



(a) Single linkage

(b) Complete linkage

(c) Average linkage

(d) Ward linkage

FIGURE 16 – Clustering agglomératif sur 3-spiral.arff

— Analyse des dendrogrammes

Comme on peut le remarquer sur les dendrogrammes, les résultats et performances des différentes variantes de linkage ne sont pas absolues. Par exemple, en ne fixant pas le nombre de clusters k, la variante single linkage n'arrive pas instinctivement à trouver 4 clusters pour le jeu de données *square.arf* mais arrive bien à trouver 3 clusters pour le *3-spiral.arff*.
La méthode du clustering agglomératif nous donne une réelle flexibilité, car les résultats sont drastiquement différents selon le linkage choisi.

— Temps de calculs

	Single linkage	Complete linkage	Average linkage	Ward linkage
xclara.arff	65.12 ms	216.04 ms	175.81 ms	218.91 ms
3-spinal.arff	2.97 ms	7.34 ms	18.61 ms	10.31 ms

TABLE 3 – Temps de calcul

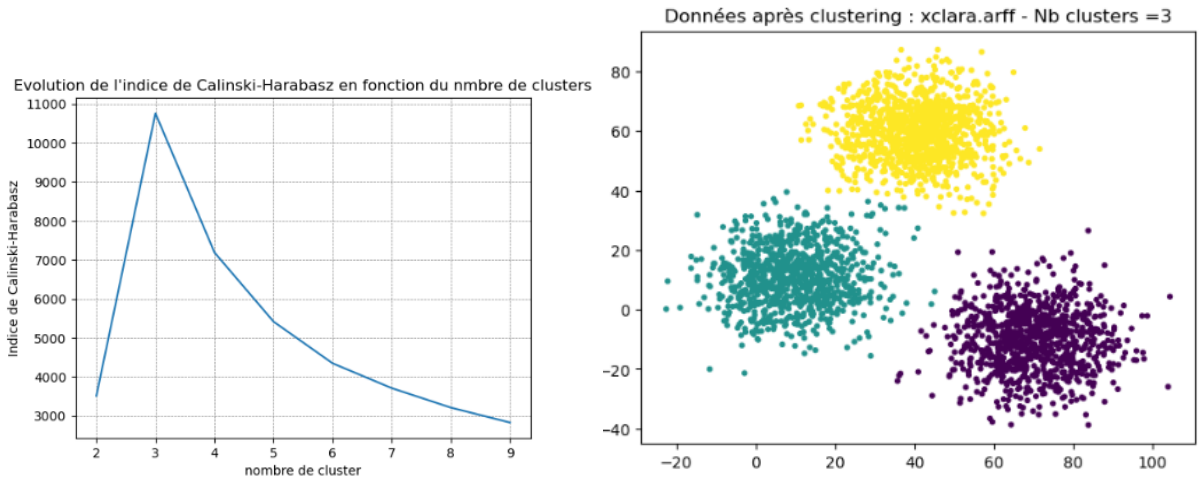
— Analyse des temps de calculs

Les temps de calcul sont tout aussi variables. Les performances des variantes de linkage dépendent fortement du jeu de données.

3.2 Application itérative de la méthode agglomérative pour la recherche du meilleur nombre de clusters

Afin de trouver le meilleur nombre de clusters, je vais appliquer itérativement la méthode de clustering agglomératif sur les deux jeux de données *xclara.arff* et *3-spiral.arff* avec un linkage "average". Pour la comparaison des différents clusterings, je vais utiliser l'indice de Calinski-Harabasz. Compris entre [0, +inf], ce coefficient correspond au rapport entre la somme de la dispersion entre éléments du même cluster et la somme de la dispersion des éléments de clusters différents. Ce coefficient est particulièrement sensible à la taille du jeu de données. On souhaite ici maximiser au maximum ce coefficient.

— xclara.arff



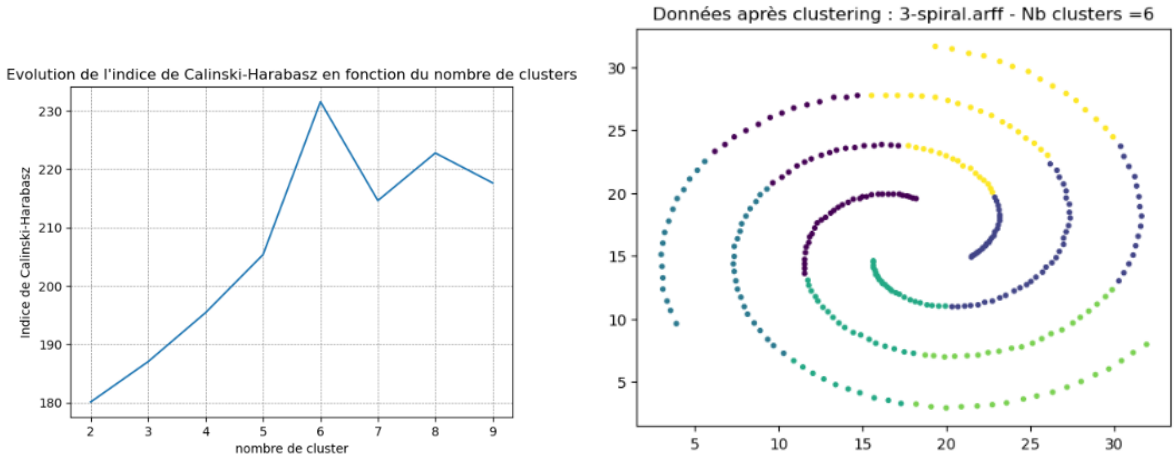
(a) Evolution de l'indice de Calinski Harabasz en fonction du nombre de clusters

(b) Clustering agglomeratif sur xclara.arff avec k=3

FIGURE 17 – xclara.arff

Comme on le remarque ici, pour le jeu de données *xclara.arff*, la recherche du nombre de clusters a bien été fait. On a un indice de Calinski-Harabasz élevé pour un nombre de clusters égal à 3 qui correspond à ce qu'on peut identifier visuellement.

— 3-spiral



(a) Evolution de l'indice de Calinski Harabasz en fonction du nombre de clusters

(b) Clustering agglomeratif sur xclara.arff avec k=6

FIGURE 18 – 3-spiral

Cependant pour *3-spiral.arff*, espérant 3 clusters (qu'on peut déterminer visuellement), la recherche itérative nous donne un indice de Calinski-Harabasz élevé pour un nombre de clusters égal à 6.

3.3 Intérêts et Limites du clustering agglomératif

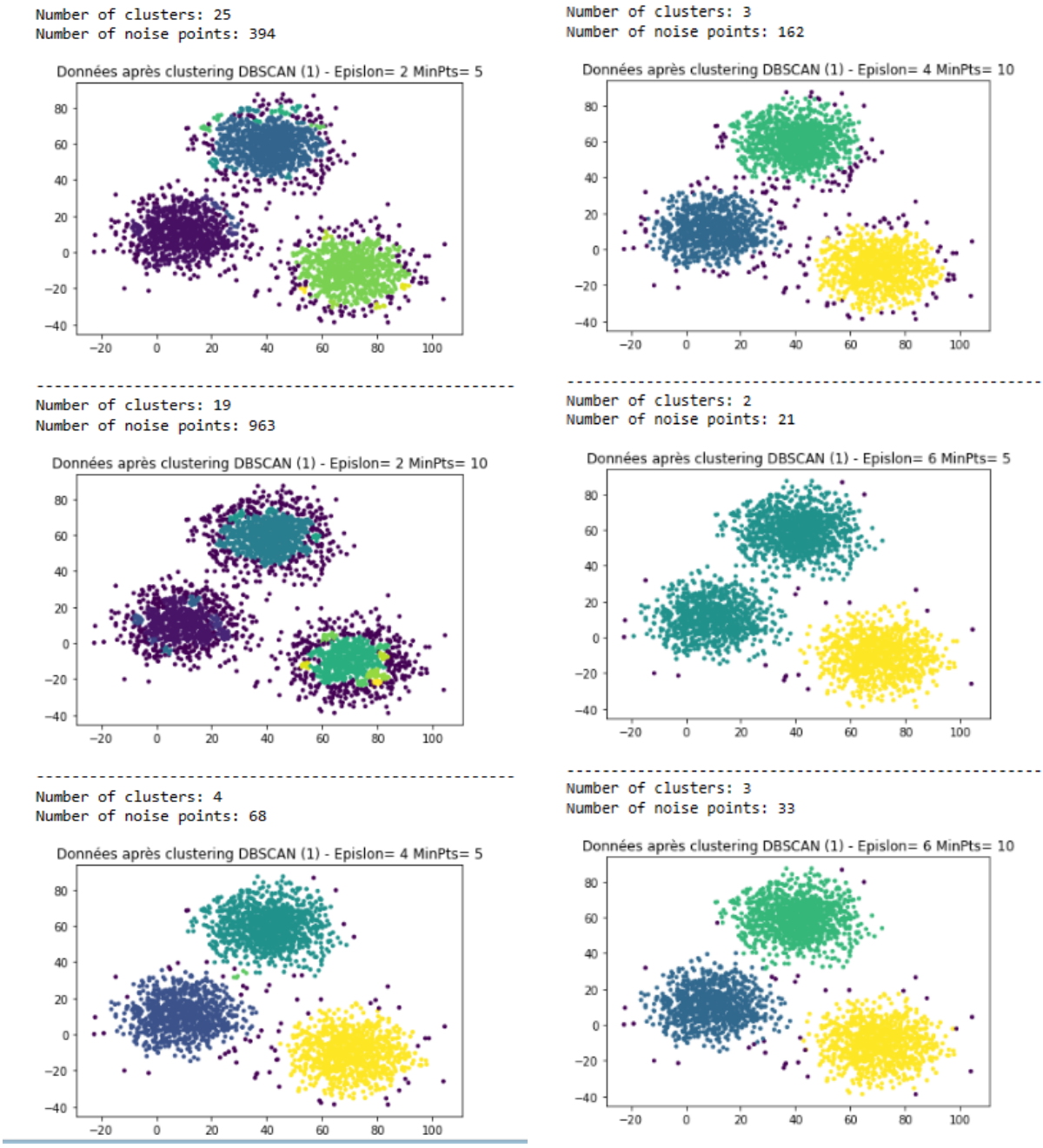
La méthode du clustering agglomératif nous donne une réelle flexibilité, car les résultats sont drastiquement différents selon le linkage choisi. Par exemple, en limitant le nombre de clusters, 'average' et 'ward' nous donnent des résultats similaires aux méthodes précédentes (même si les proportions peuvent changer). En revanche, 'single' linkage change totalement les résultats, permettant par exemple de séparer correctement les clusters de smile2, mais se retrouve en échec sur des clusters jusqu'ici bien traités. Ce 'single linkage' est très intéressant dans les datasets où les clusters sont très denses, même s'ils prennent beaucoup d'espace avec des formes sphériques ou convexes. Cependant, il est inefficace dans les datasets où la densité d'un même cluster est moins élevée, ou dans ceux dont les clusters sont trop proches les uns des autres. Les temps de calcul dépendent de la limite imposée, mais restent raisonnables si celle-ci est choisie convenablement.

4 Clustering DBSCAN et HDBSCAN

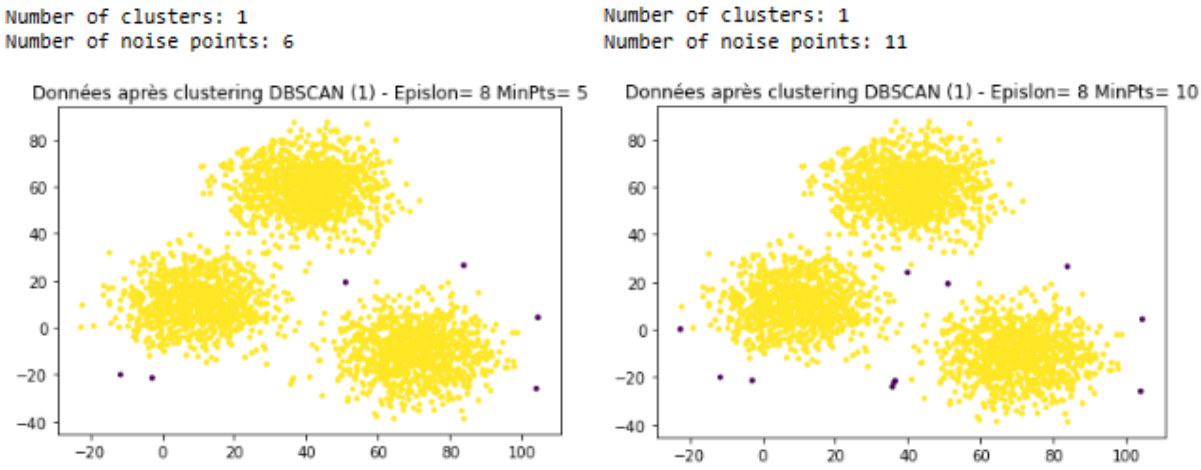
4.1 Clustering DBSCAN

4.1.1 Tests de différentes valeurs de paramètres

Dans cette partie, je vais tester le clustering DBSCAN pour différentes valeurs de epsilon et de min_samples.



(a) (b)



(a) (b)

FIGURE 20 – Clustering du jeu de données xclara.arff avec différentes valeurs de paramètres

Comme on peut le voir, les paramètres epsilon et min_samples influent énomement sur le résultat du clustering. On a par exemple les deux configurations {epsilon= 4, min_samples = 10} et {epsilon= 6, min_samples = 10} qui permettent d'obtenir 3 clusters, mais avec un nombre de "noise points" différent.

4.1.2 Application de la méthode DBSCAN

Afin de rechercher des paramètres pertinents pour les paramètres *eps* et *min_samples* je vais calculer les distances moyennes aux k plus proches voisins pour chaque exemple du jeu de données xclara.arff et déterminer le "coude" de la courbe.

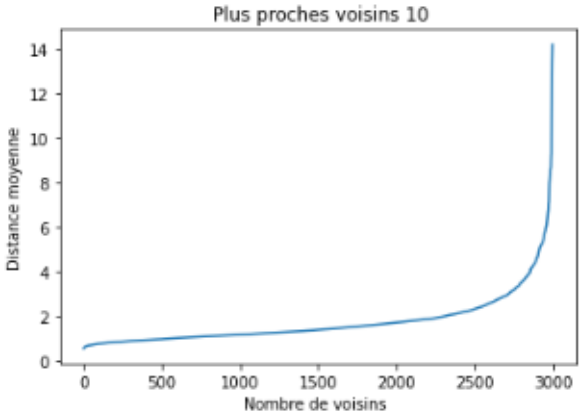


FIGURE 21 – Distance moyenne en fonction de k sur le jeu de données xclara.arff

Ce graphique nous permet de trouver un intervalle de epsilon, nous permettons d'avoir un bon clustering. Ici, le "coude" correspond à l'intervalle [3,4].

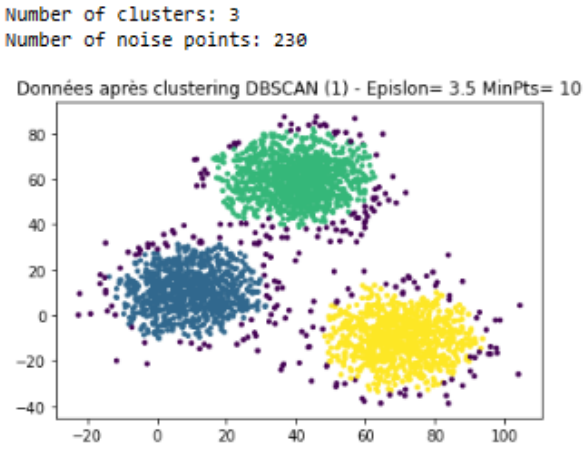


FIGURE 22 – Clustering DBSCAN de xclara.arff après recherche de bons paramètres epsilon et min_samples

4.1.3 Intérêts et Limites de la méthode DBSCAN

1. Jeu de données pour lesquels DBSCAN fonctionne correctement

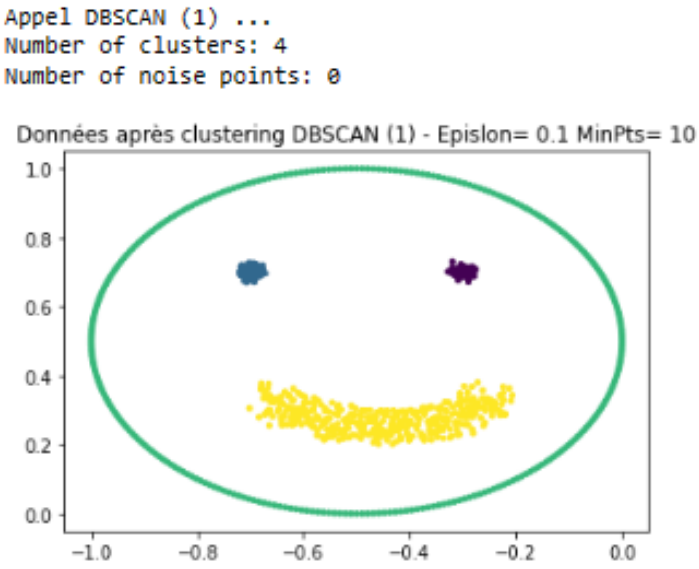


FIGURE 23 – Clustering DBSCAN du jeu de données smile2.arff

2. Jeux de données pour lesquels DBSCAN a des difficultés.

```
Appel DBSCAN (1) ...
Number of clusters: 100
Number of noise points: 3894
```

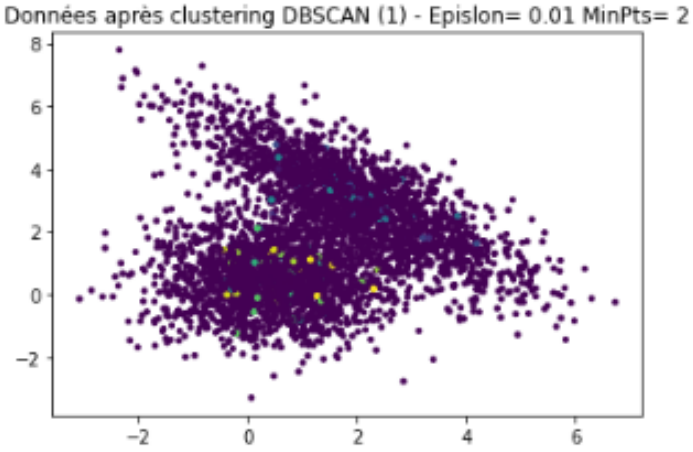


FIGURE 24 – Clustering DBSCAN du jeu de données engytime.arff

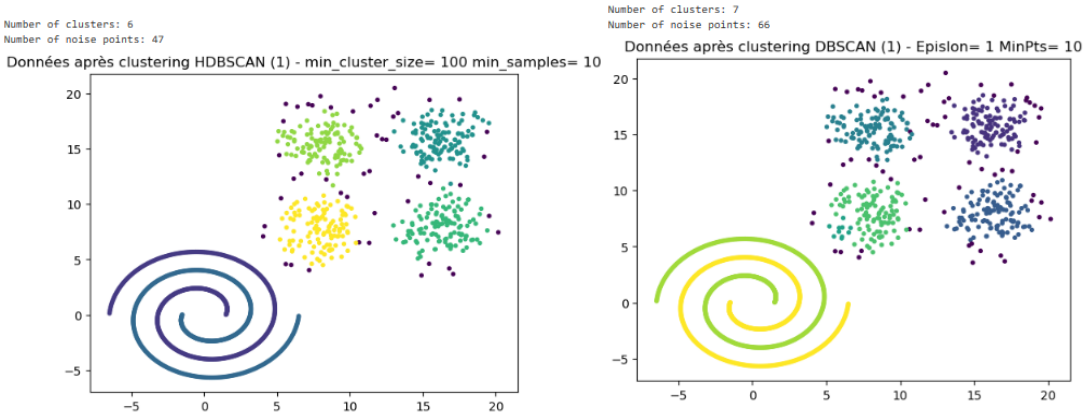
DBSCAN est très efficace pour séparer les *datasets* dont les clusters sont denses, qu'ils soient convexes ou pas. À ce titre, les *datasets* comme *smile2* sont idéaux, mais même ceux comme *xclara* fournissent de bons résultats une fois la bonne valeur d'epsilon choisie. Cependant, lorsque les clusters sont proches les uns des autres, il semble inaltérable que les points les plus éloignés de leur cluster respectif soient isolés par DBSCAN, car si on augmente trop la valeur d'epsilon, tous les points se retrouvent dans le même cluster. Au même titre, si les clusters ont des densités différentes, DBSCAN ne peut les gérer correctement. Les temps de calculs sont très faibles, mais peuvent augmenter drastiquement avec la taille du dataset.

Un avantage considérable de cette méthode est la recherche automatique du bon nombre de clusters, ainsi que la détection de bruit.

4.2 Clustering HDBSCAN

Le grand avantage de cette solution est de ne pas avoir à la guider avec des variables : en effet, la seule nécessité est d'imposer le nombre de points minimal d'un cluster, ce qui est souvent bien plus aisé que déterminer le nombre de clusters ou la distance minimale entre eux. D'autant plus que HDBSCAN traite le problème de diversité de densité que DBSCAN avait.

4.2.1 Comparaison HDBSCAN et DBSCAN



(a) HDBSCAN de spiralsquare.arff (b) DBSCAN de spiralsquare.arff

FIGURE 25 – Comparaison HDBSCAN et DBSCAN sur le jeu de données spiralsquare.arff

4.2.2 Limites du clustering HDBSCAN

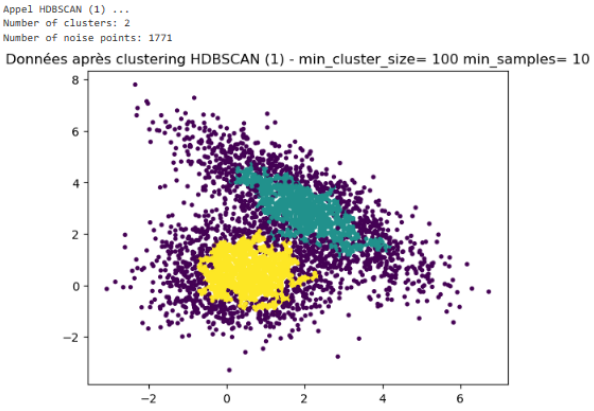


FIGURE 26 – Clustering HDBSCAN sur le jeu de données engytime.arff

De même que la méthode DBSCAN, HDBSCAN aura plus de mal avec des jeux dans lesquels les clusters sont rapprochés. Nous utiliserons donc ici le jeu de données *engytime.arff*. Aussi, HDBSCAN parvient à identifier des « îles » comme DBSCAN, mais considère tout le reste comme du bruit. Cela est particulièrement visible sur le jeu engytime.

Le fait de fixer le nombre de points minimal d'un cluster nous permet également de ne pas considérer les clusters de taille inférieure à ce nombre. Cependant, si cette taille minimale est mal fixée, les résultats du clustering beaucoup s'éloigner du clustering attendu.

5 Conclusion

Dans ce TP, j'ai ainsi pu découvrir et comparer différentes méthodes de clustering notamment des méthodes fournies par scikitlearn. L'étude des méthodes k-means, clustering agglomératif, dbscan et hdbscan à travers leurs applications sur multiples datasets m'a permis de me familiariser avec un concept du machine learning, l'apprentissage non supervisé. Cette étude m'a notamment montré que les performances de chacune des méthodes de partitionnement sont entièrement dépendantes du dataset étudié et que le choix de la méthode doit être effectué en fonction de l'utilisation voulue des données.