TP clustering

TESSIER Raphael et SIARD Nicolas

Introduction

Ce rapport explore deux approches fondamentales de l'apprentissage non supervisé, à savoir le clustering agglomératif et l'algorithme K-means. L'objectif principal est de réaliser une comparaison critique entre ces deux méthodes en analysant à la fois leurs avantages et leurs limitations. Il est également souligné que l'efficacité de chaque algorithme est étroitement liée aux caractéristiques spécifiques de chaque ensemble de données examiné. Différentes métriques pour l'analyse de la pertinence des modèles seront également présentées.

La première section de notre analyse se focalise sur un examen approfondi des caractéristiques de chaque méthode. Nous aborderons leur aptitude à traiter différents types de jeux de données. Cette évaluation permettra une meilleure compréhension des contextes dans lesquels chaque méthode peut être utilisée.

La deuxième section de notre étude consiste en une évaluation pratique sur plusieurs datasets proches mais présentant certaines variations. Nous permettant d'étudier la performance effective des algorithmes dans des conditions diverses.

L'ensemble du code élaboré dans le cadre de ce travail pratique peut être consulté sur le git suivant : https://github.com/NicoSiard/Machine_Learning/tree/master/Unsupervised

Les scripts *k-Means.py* et *hierarchical.py* permettent l'analyse des jeux de données des datasets du dossier *artificial*, par appel respectif des fonctions *computeKmeans* et *computeAglomerative*, en passant les arguments comme explicité dans la description de la fonction.

Le choix du dataset s'effectue dans le *main* au niveau de l'affectation de databrut.

Le script *Script_Part2.py* permet l'analyse des jeux de données de *dataset-rapport* en réutilisant les fonctions définies dans *k-Means.py* et *hierarchical.py.*

Partie 1 : Analyse des méthodes de clustering

I. Métriques d'évaluation

a. Coefficient de silhouette

Le coefficient de silhouette montre la proximité des points d'un cluster avec ceux des autres clusters. Il est intéressant à utiliser sur des clusters de forme convexe, et ne sera pas très performant sur des clusters de forme irrégulière. Le coefficient varie entre -1 (mauvais partitionnement) à 1 (meilleur regroupement)

b. Indice de Davies-Bouldin

Cet indice exprime la similarité entre les clusters en comparant la distance entre les clusters avec la taille des clusters. La valeur de l'indice peut fortement varier d'un dataset à un autre mais restera positive, plus l'indice est proche de zéro, meilleur est le clustering. Il est intéressant à utiliser lorsque l'on souhaite obtenir des clusters homogènes.

c. Indice de Calinski-Harabasz

Il s'agit du rapport entre la séparation entre cluster et la dispersion au sein du cluster. Une valeur élevée correspond à un bon clustering. Il permet d'évaluer la compacité et la cohésion des clusters.

II. Clustering k-Means

K-Means est une méthode de clustering ayant pour but de regrouper *k* clusters en cherchant à minimiser la distance entre les points et le centre du cluster auxquels ils appartiennent. Pour cela l'algorithme place *k* centre aléatoirement puis classe les points en fonction de leurs distances à ce point. Il calcule le nouveau centre des clusters puis recalcule les points lui appartenant. L'algorithme se termine lorsque les points n'évoluent plus. Cette méthode de classification ne permet d'obtenir que des clusters de forme convexe, comme nous allons le voir par la suite.

Afin d'évaluer nos clustering et de trouver le meilleur cas, nous allons ici utiliser le coefficient de silhouette, celui-ci étant idéal pour les clusters de forme convexe. Nous implémentons donc un algorithme permettant de rechercher le meilleur nombre de clusters en regardant le score de silhouette. (*script k-means.py*)

Nous avons donc appliqué cette méthode sur 4 différents jeux de données, les résultats étant trouvable en *Annexe 1, figure 1 à 4.*

Nous pouvons voir que l'algorithme fonctionne parfaitement pour les deux datasets convexes, *figure 1 et 2*, où le coefficient de silhouette est bien meilleur lorsque le bon nombre est utilisé.

Pour les datasets non-convexes, l'algorithme ne parvient pas à trouver un clustering permettant d'obtenir un score intéressant comme pour la *figure 3*, où le score ne dépasse pas 0.4.

Cependant l'algorithme peut également trouver des clusters correspondant à un score très correct (0.7) même lorsque le dataset est non convexe. On peut ainsi obtenir le clustering en *figure 4,* qui même si celui-ci ne donne pas le regroupement qui serait attendu, sera considéré comme un bon clustering selon le score de silhouette. Il faut donc bien veiller à n'utiliser l'algorithme k-Means uniquement lorsque l'on est certain de rechercher des formes convexes dans le dataset.

III. Clustering agglomératif

Nous allons nous intéresser ici au clustering hiérarchique ascendant. Celui-ci démarre en plaçant chaque point dans un cluster et regroupe à chaque itération les clusters les plus proches, jusqu'à atteindre un seuil exprimé en nombre de cluster ou bien en distance entre ces derniers (il est alors possible de ne pas imposer de nombre de cluster à l'algorithme).

Afin de rechercher les clusters les plus proches, plusieurs méthodes pour calculer la distance (similarité) entre deux clusters peuvent être utilisé :

- Single-linkage : prend en compte la similarité entre les points les plus proches entre les deux clusters. Cela permet d'obtenir de longues chaînes et des clusters non-convexes.
- Complete-linkage : prend en compte la similarité entre les points les plus éloignés entre les deux clusters. Cela permet d'obtenir des clusters sphériques et convexes.
- Average-linkage : prend en compte la moyenne des distances points à points. Cette méthode améliore la résistance de l'algorithme aux outliers.
- Ward's method : permet d'obtenir des résultats proches de k-means

Cet algorithme permet donc d'obtenir différente forme de clusters en fonction de la fonction de similarité utilisé.

Les trois métriques présenter en *I* peuvent être utilisé avec cette classification en fonction de ce que l'on recherche comme cluster dans le dataset.

Il faut donc adapter les fonctions de similarité et les métriques.

De la même manière que pour k-means, nous allons vérifier ces éléments en appliquant l'algorithme à différents jeux de données, les résultats étant visible en *Annexe 2*.

Dans un premier temps, nous pouvons comparer les résultats de l'algorithme en fonction du type de la condition d'arrêt sur un dataset simple (*figure 5 et 6*). On peut voir que le meilleur modèle obtenu est le même dans les deux cas. Cependant, comme il est plus difficile d'estimer un intervalle de distance dans laquelle se trouverais celle optimal que d'estimer un intervalle de nombre de cluster en voyant les données en 2 dimensions, nous garderons l'intervalle en nombre de cluster pour la suite des tests. Il faut cependant noter que dans le cas où il est difficile de visualiser les données, il peut être intéressant d'utilisé un seuil en distance, l'algorithme trouvant alors de lui-même le *k* optimal.

Intéressons-nous maintenant aux avantages et inconvénients des différentes fonctions de similarité. Nous pouvons voir sur les *figures 7 et 8* que les bonnes et mauvaises utilisations de single et complete linkage : Single étant efficace pour des clusters sous forme de chaîne avec les points à distance régulière, alors que complete permet des regroupements sphériques, même avec des distances variables entre les points des clusters.

Partie 2 : Etude et analyse comparative sur un nouveau jeu de données

Nous allons désormais réaliser l'analyse des résultats de différentes méthodes de clustering pour les datasets x1, x2, x3, x4. Cela nous permettra d'étudier l'évolution des algorithmes pour des datasets assez similaires mais où les clusters à identifier se distingue de moins en moins.

Nous n'étudierons pas y1, celui-ci présentant plus de 20 fois plus de point que les autres, le temps de calcul aurait été multiplié par 20 dans le cadre de k-means, et par jusqu'à 8000 pour certain des algorithmes hiérarchiques, d'après les complexités de ces algorithmes (O(n) à $O(n^3)$).

Nous n'analyserons pas non plus les datasets zz1 et zz2, ceux-ci présentant des clusters bien isolé mais variant en intensité. Ils auraient uniquement été intéressant à étudier avec l'algorithme DBSCAN, celui-ci se basant sur une densité de point fixe pour déterminer l'appartenance ou non à un cluster.

a. Dataset x1

Pour ce dataset, présentant des groupements convexes et denses, nous allons utiliser le coefficient de silhouette afin de sélectionner le meilleur nombre de cluster pour chacune des méthodes de clustering analysées.

Nous allons comparer les résultats pour l'algorithme k-means, et l'algorithme hiérarchique en single-linkage, complete-linkage et average-linkage. Les visualisations des résultats sont visibles en *Annexe 3-a, figure 9 à 12*.

Nous obtenons un nombre de cluster *k* égal à 15 pour les trois premiers algorithmes, avec un score d'approximativement 0.7 pour chacun. La courbe d'évolution du score en fonction du nombre de cluster forme un pic, nous pouvons donc en déduire que la valeur *k=15* est la plus optimal et de loin.

Cependant, en regardant dans le détail les clusters produit, on se rend compte que ceux-ci sont différents : les deux groupements en bas à droite sont découpés de manière différente.

On peut voir que en complete-linkage (*figure 10*), le modèle ne parvient pas à identifier le groupe allongé comme un cluster (entouré en noir sur la figure), il le sépare en deux et en fusionne une partie avec un autre cluster à proximité afin de former des groupements plus sphérique. Cela s'explique par sa façon de calculer les distances, comme expliqué en *Partie1-III*.

L'algorithme K-means produit des clusters quasi parfait, à l'exception de 6 points, que nous aurions aimé voir placer dans un cluster différent (*figure 9, entouré dans la couleur du cluster désiré*). En effet, le cluster vert étant assez allongé, son centre de gravité se trouve assez éloigné de ses extrémités pouvant exclure certain point (cercle vert), mais pouvant inclure d'autre point moins loin du centre de gravité mais ne se positionnant pas dans l'axe du cluster (cercle jaune) et se rapprochant plus d'un autre cluster selon un point de vue humain.

L'algorithme donnant les meilleurs résultats est alors l'average linkage, celui-ci prenant en compte chaque point du cluster et non uniquement sont centre ou ces extrémités, permettant de bien identifié des formes convexes plus complexes.

Au contraire en single-linkage le clustering ne fonctionne absolument pas. L'algorithme permettant de réaliser des chaînes de points, en les regroupant de proche en proche, il va dès le début regrouper des zones de points avec une faible séparation avant de s'intéresser aux autres clusters. Comme le dataset présente quelque point reliant les clusters les uns aux autres, l'algorithme n'arrive pas à les séparer, car il groupe les points trop tôt et par sa construction, ne peut plus les reséparer. Le coefficient maximum de silhouette ainsi obtenu est de seulement 0.2 environ pour 8 clusters identifiés.

b. Dataset x2

Pour ce dataset, présentant des groupements convexes et denses, nous allons utiliser le coefficient de silhouette afin de sélectionner le meilleur nombre de cluster pour chacune des méthodes de clustering analysées.

Nous allons comparer les résultats pour l'algorithme k-means, et l'algorithme hiérarchique en single-linkage, complete-linkage et average-linkage. Les visualisations des résultats sont visibles en *Annexe 3-b, figure 13 à 16*.

Nous obtenons un nombre de cluster k égal à 15 pour les trois premiers algorithmes, avec un score d'approximativement 0.65 pour k-means et average. La courbe d'évolution du score en fonction du nombre de cluster forme un pic, nous pouvons donc en déduire que la valeur k=15 est la plus optimal.

Pour complete-linkage (*figure 14*) le score est cette fois ci de 0.5, en effet, les observations faites sur le dataset x1 précédemment sur les clusters allongés sont encore plus perceptible ici, et réduisent le coefficient de silhouette du modèle. On peut également voir à la fonction score que plusieurs nombres de cluster (13, 15 et 22) produisent un score proche, indiquant une incertitude sur la séparation.

Il est encore une fois difficile de distinguer lequel de k-means et de average-linkage est le plus performant, leurs scores étant pratiquement identique et la réelle appartenance au cluster difficile à identifier d'un point de vue humain.

En single-linkage le clustering ne fonctionne toujours pas. Une fois de plus, l'algorithme permettant de réaliser des chaînes de points, en les regroupant de proche en proche, il va dès le début regrouper des zones de points avec une faible séparation avant de s'intéresser aux autres clusters. Ce comportement est d'autant plus visible car les clusters sont encore moins séparés que précédemment. Le coefficient maximum de silhouette ainsi obtenu est de seulement 0.2 environ pour seulement 2 clusters identifiés. Soit le minimum possible avec notre script (1 seul cluster ne pouvant pas être évalué en score)

Ce dataset met encore plus en exergue les spécificités des algorithmes observés sur le dataset x1.

c. Dataset x3

Ce dataset est encore plus dense que les précédent, les méthodes hiérarchiques en complete-linkage et single-linkage produisant les mêmes types de résultats que précédemment, nous ne nous attarderons pas plus dessus.

Nous allons donc comparer les résultats pour l'algorithme k-means, et l'algorithme hiérarchique en average-linkage. Les visualisations des résultats sont visibles en *Annexe 3-c, figure 17 et 18*.

Ici nous atteignons les limites du clustering agglomératif. En effet les points étant rapproché et les séparations entre les clusters très peu marqué, celui-ci n'arrive pas à regrouper les points de façon à constituer des clusters cohérents, son coefficient de silhouette peine à atteindre les 0.4 et aucune valeur de nombre de cluster ne se dégage réellement.

Au contraire, k-means continus de fonctionner en indiquant un k plus performant, même si le score de 0.5 nous indique que la répartition n'est pas optimale.

d. Dataset x4

Nous arrivons aux mêmes conclusions que pour le dataset x3 concernant le clustering hiérarchique avec les trois fonctions de similarités, nous ne développerons donc pas sur ces algorithmes pour ce dataset.

En ce qui concerne k-means (*Annexe 3-c, figure 19*), nous pouvons remarquer que le score reste environ le même que précédemment, ce qui pourrait indiquer des performances identiques. Cependant nous observons que la majorité des séparations se font uniquement selon des droites, sans réel logique derrière. Aucune distinction n'étant réellement visible à ces endroits là sur le dataset brut (*Annexe 3-c, figure 20*).

On peut alors se demander si le clustering est encore pertinent.

Conclusion

En résumé, notre analyse comparative entre le clustering agglomératif et l'algorithme K-means a révélé les avantages et les limitations de chaque approche. Le clustering agglomératif offre une flexibilité notable, mais peut être confronté à des défis de scalabilité et à une sensibilité à la densité des données au sein des clusters. À l'inverse, K-means présente une efficacité computationnelle supérieure, bien qu'il soit plus à même d'être influencé négativement par des formes de clusters non circulaires. L'utilisation d'une boucle et de recherche d'optimum de score permet pour ces méthodes de s'affranchir en partie du besoin de connaissance des hyperparamètres comme le nombre de cluster des datasets.

Le choix entre ces deux approches dépend des exigences spécifiques du projet et des caractéristiques inhérentes aux données. Cette comparaison approfondie apporte une compréhension plus fine de la manière d'appliquer efficacement ces méthodes de clustering dans divers contextes d'application.

Annexe

I. K-Means

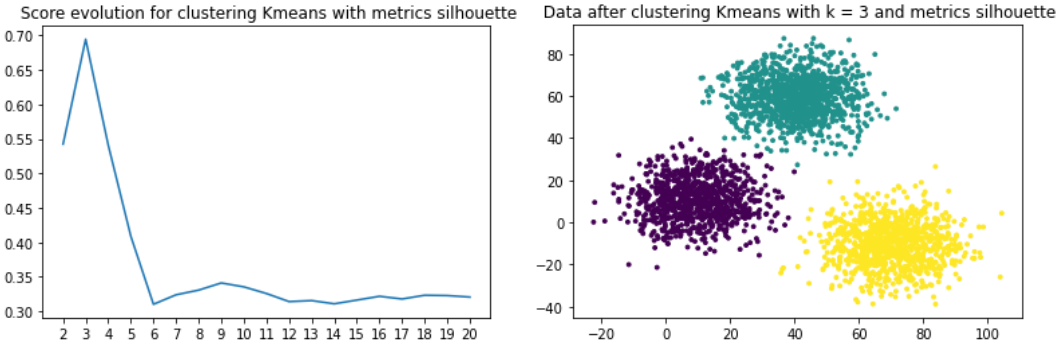

Figure 1 : Algorithme k-means sur le dataset xclara

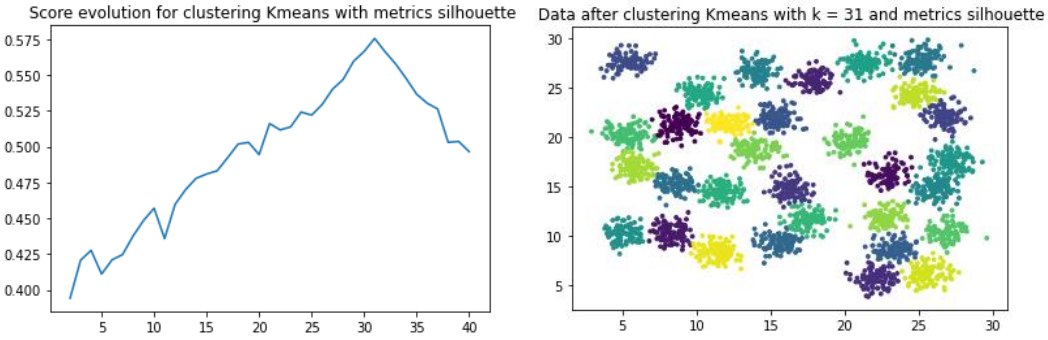

Figure 2 : Algorithme k-means sur le dataset D31

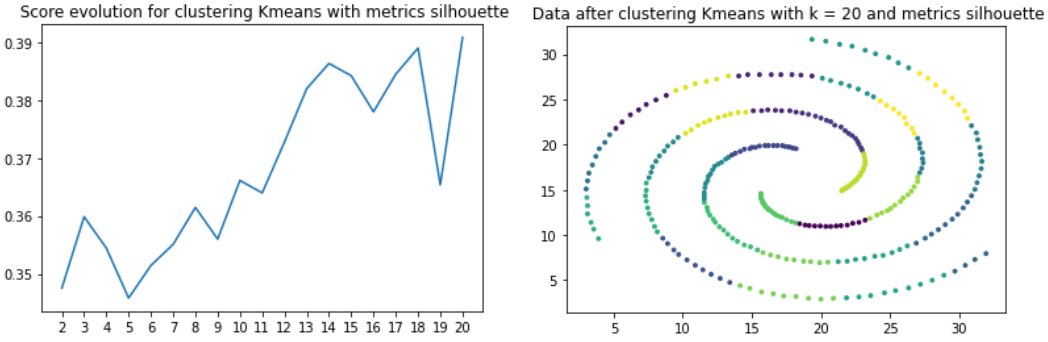

Figure 3 : Algorithme k-means sur le dataset spiral-3

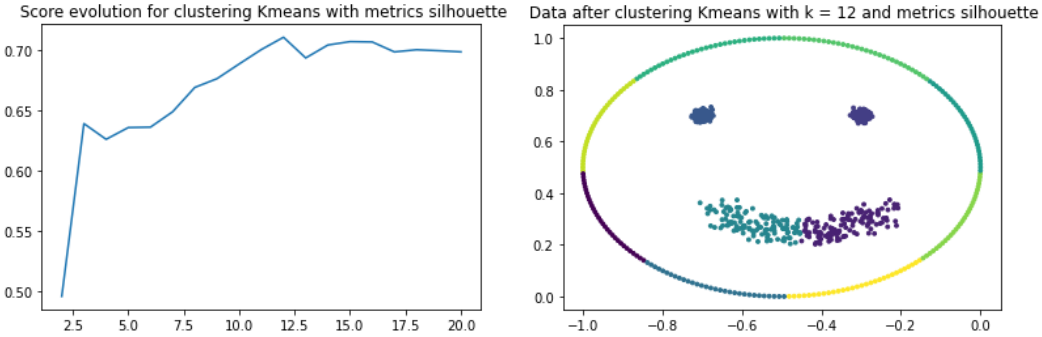

Figure 4 : Algorithme k-means sur le dataset smile1

II. Agglomératif

a. Comparaison nombre de cluster et threshold

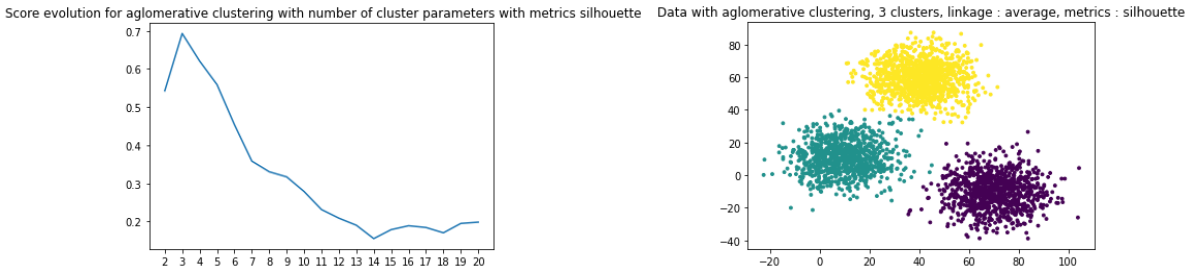

Figure 5 : Algorithme hiérarchique avec seuil en nombre de cluster sur le dataset xclara

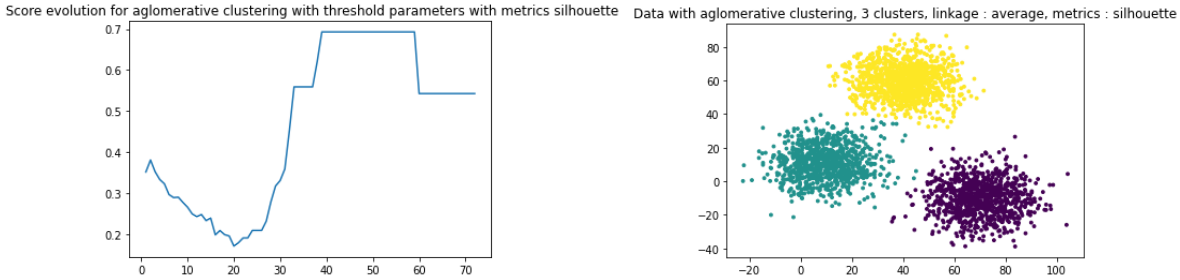

Figure 6 : Algorithme hiérarchique avec seuil en distance sur le dataset xclara

b. Comparaison des fonctions de similarité

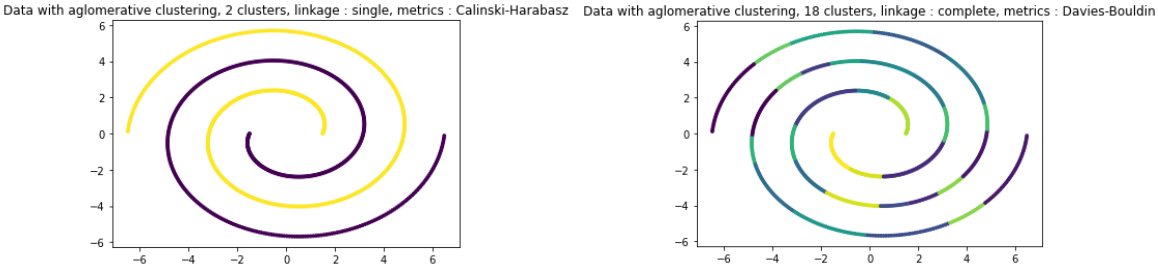

Figure 7 : Single-linkage fonctionnant sur forme non convexe contrairement à complete-linkage, dataset spiral

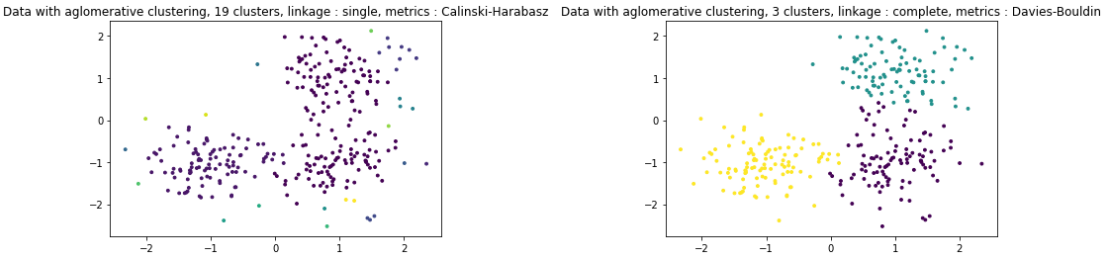

Figure 8 Complete-linkage fonctionnant pour des regroupements sphérique, contrairement à single-linkage, dataset blobs

III. Nouveau jeu de données

a. Dataset x1

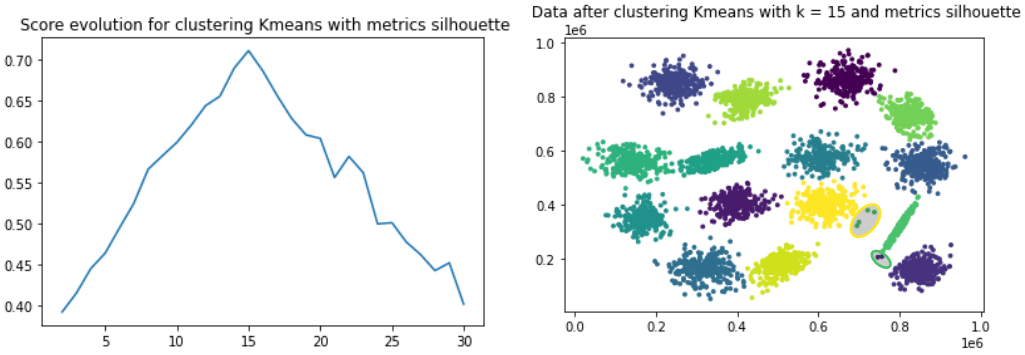

Figure 9 : k-means appliqué au dataset x1

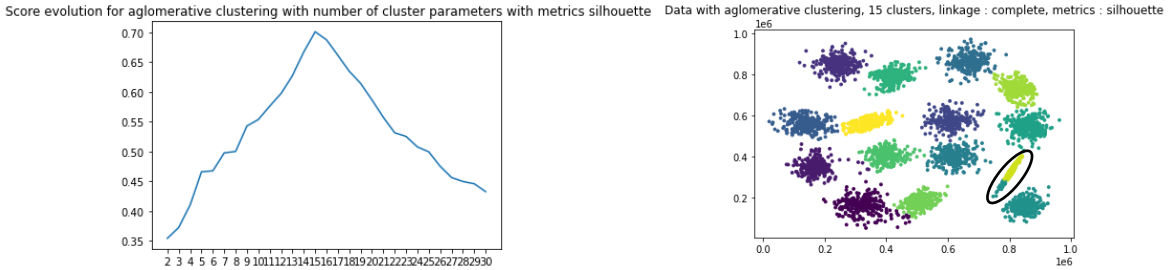

Figure 10 : Hiérarchique, complete-linkage, appliqué à x1

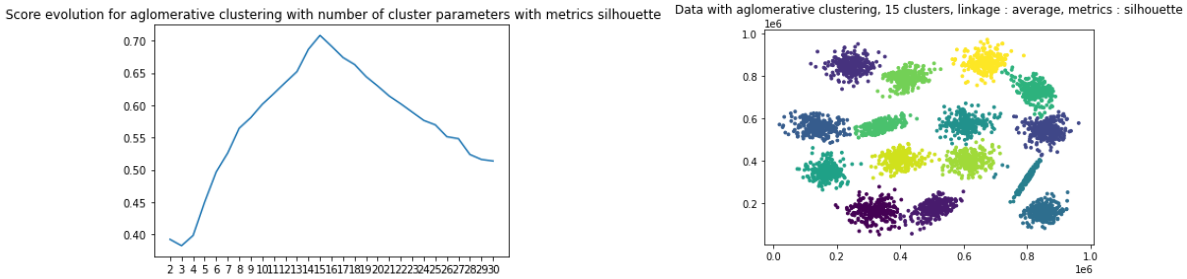

Figure 11 : Hiérarchique, average-linkage, appliqué à x1

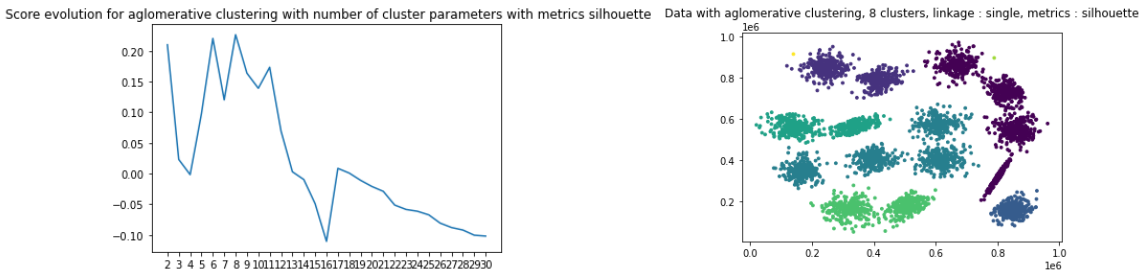

Figure 12 : Hiérarchique, single-linkage, appliqué à x1

b. Dataset x2

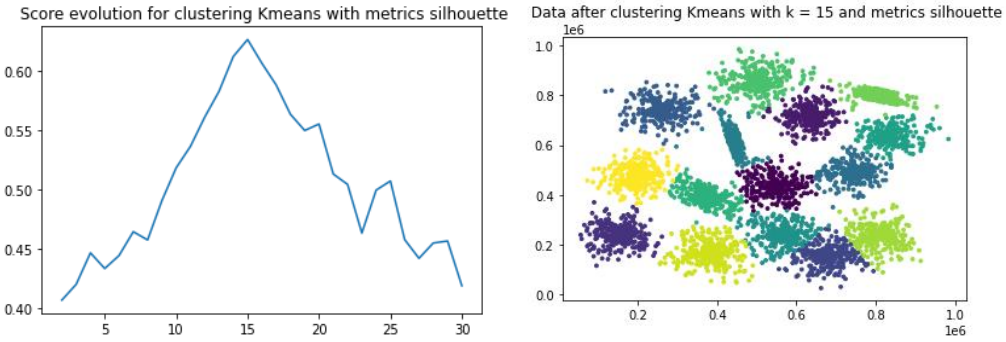

Figure 13 : k-means appliqué au dataset x2

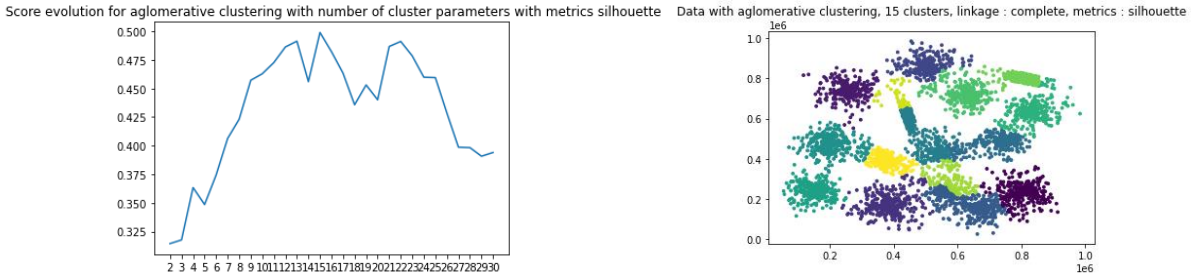

Figure 14 : Hiérarchique, complete-linkage, appliqué à x2

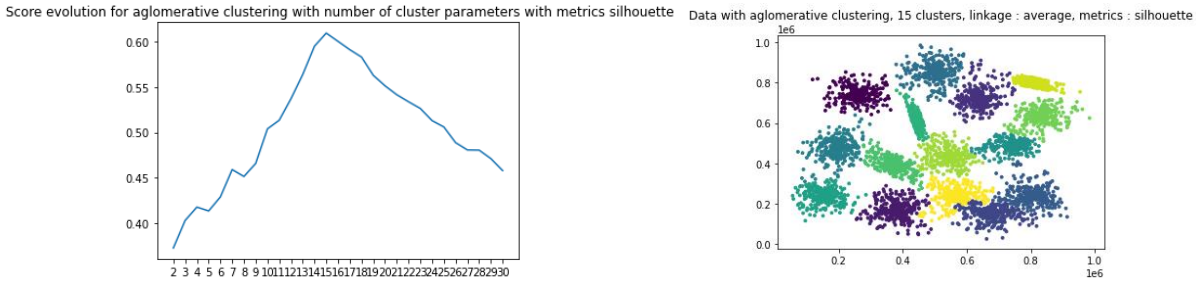

Figure 15 : Hiérarchique, average-linkage, appliqué à x2

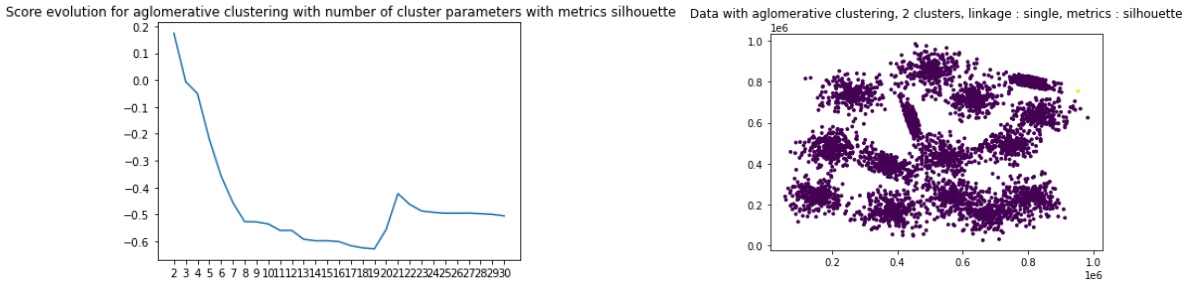

Figure 16 : Hiérarchique, single-linkage, appliqué à x2

c. Dataset x3

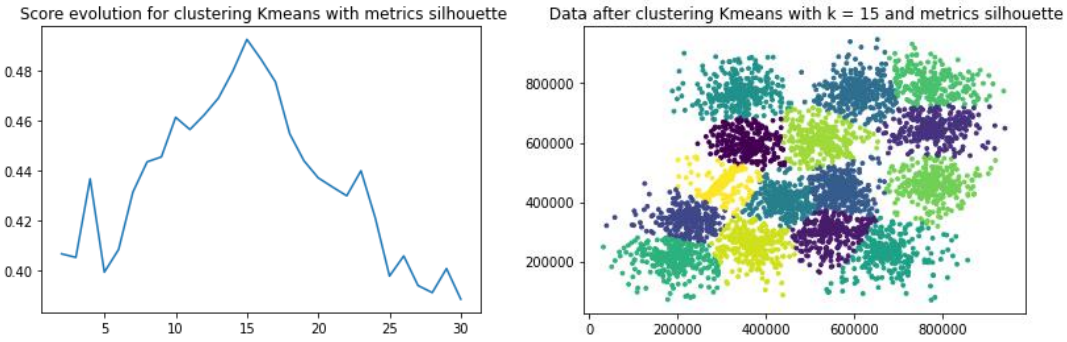

Figure 17 : : k-means appliqué au dataset x3

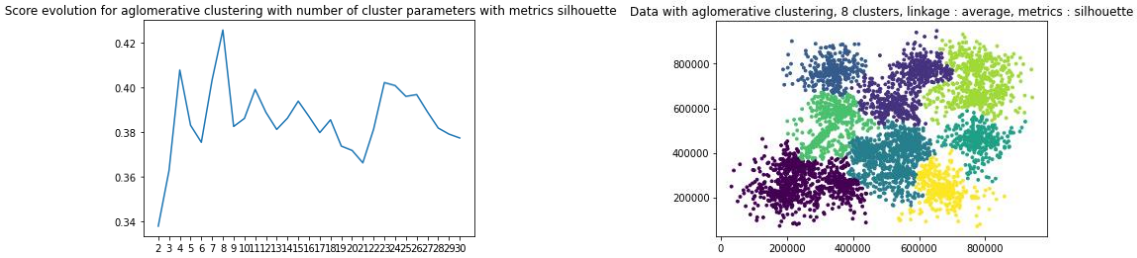

Figure 18 : Hiérarchique, average-linkage, appliqué à x3

d. Dataset x4

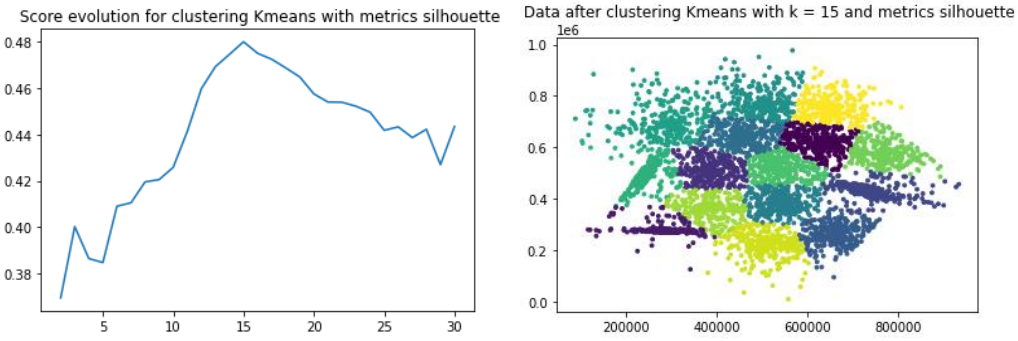

Figure 19 : k-means appliqué au dataset x4

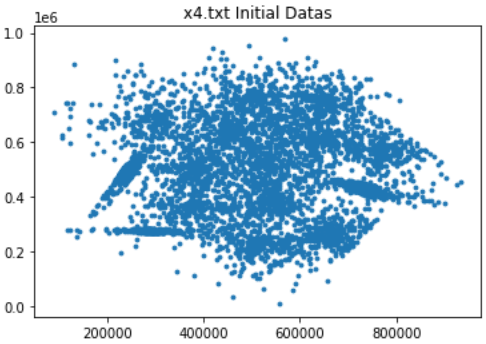

Figure 20 : dataset x4 brut