Rapport TP d'apprentissage non supervisé

UF : Analyse descriptive et prédictive
Responsable d'uf : MJ. Huguet

Bravais Jérôme - Genoud Quentin - Wowk Timothée

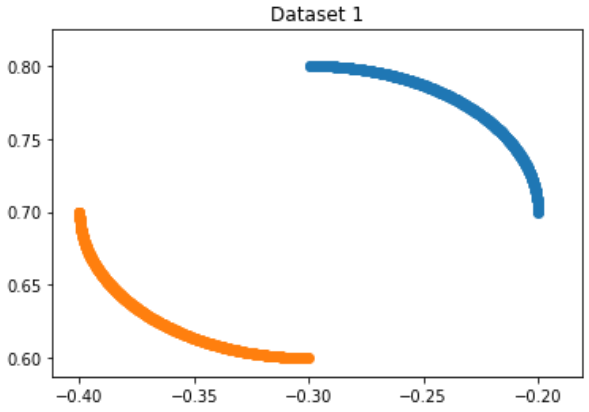
5-IR-SDBD 03/12/2019

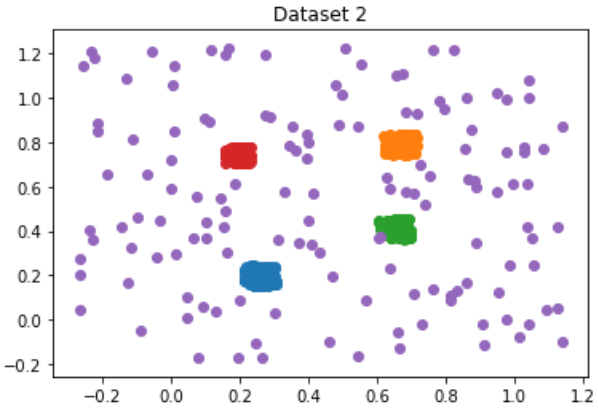
Lien vers le github :
https://github.com/QuentinG66/tps_app_non_supervise

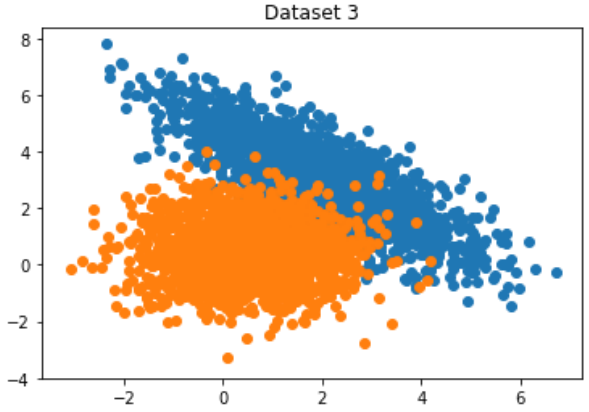
1-Introduction

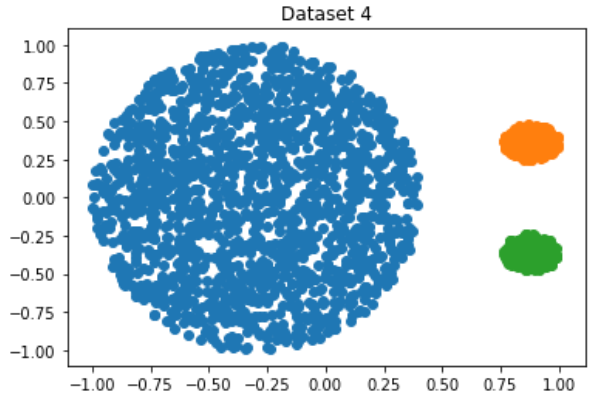
A la différence de l'apprentissage supervisé, l'apprentissage non supervisé ne fournit pas les labels des données d'apprentissage. Dans le cas du clustering, nous avons un ensemble de points (de coordonnées x et y). On se limitera ici à des données en deux dimensions pour faciliter l'affichage. Le but est de différencier les clusters c'est à dire les différents groupes de points. Dans ce TP nous allons utiliser plusieurs méthodes pour réaliser du clustering sur des datasets variés. Puis nous allons utiliser et comparer différents critères d'évaluations , comme le score de Davies Bouldin ou le score de Calinski Harabasz.

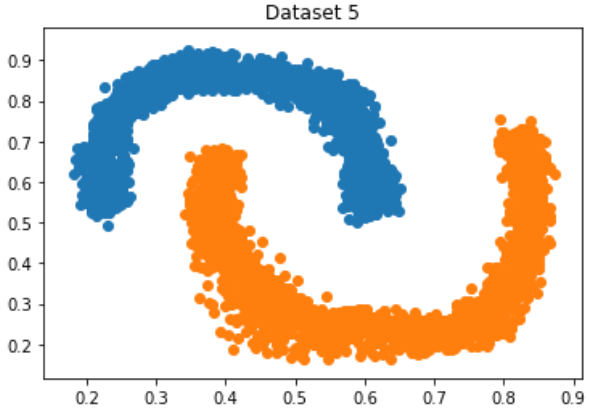
Davies Bouldin : [0: +inf[plus le score est faible, plus la classification est de qualité.
Calinski Harabasz : [0: +inf[plus le score est grand, plus la classification est de qualité.
Pour réaliser nos expérimentations, nous utiliserons 6 jeux de données :

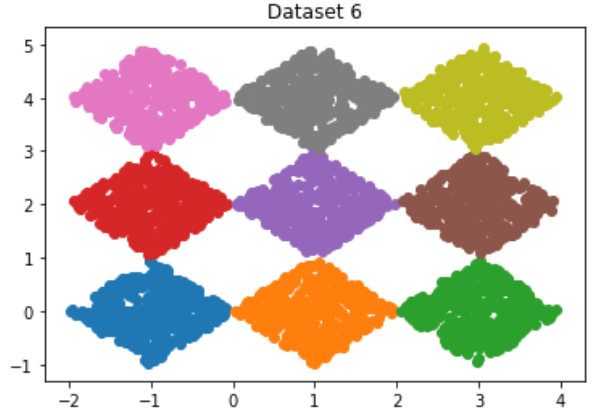












Nous avons choisis ces jeux de données car ils ont des caractéristiques différentes. Notre dataset référence est le 6 car dans ce dataset les formes sont convexes et moyennement séparés, il est aussi non bruité et constitué de 9 cluster différents de densités similaires. Les datasets 1 et 5 ont été choisis car les formes des clusters sont concaves (surtout le dataset 5). Le dataset 2 car il est bruité et le dataset 4 car les densités des clusters sont différentes. Le dataset 3 a été choisi car les 2 clusters sont mal séparés.

2-K-Means

Nous allons commencer par donner à K-Means le nombre exact de clusters attendus sur un dataset avec des formes convexes bien identifiées.

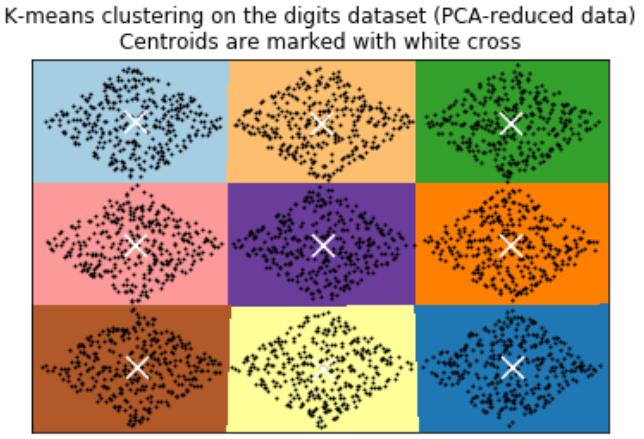


Figure 1

Ici on a donné le bon nombre de cluster (k=9) à l'algorithme et le résultat semble concluant. Les clusters sont bien définis et les centroides sont au centre des clusters.
Davies Bouldin's score : 0.5531
Calinski Harabaz's score : 5855.1522

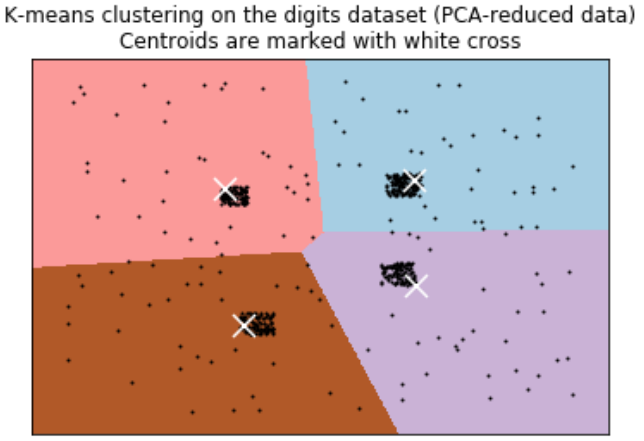


Figure 2

Pour le dataset n°2, on a donné 4 centres à l'algorithme, on remarque que même avec du bruit les centres sont assez bien placés.
Davies Bouldin's score : 0.4627
Calinski Harabaz's score : 996.1731

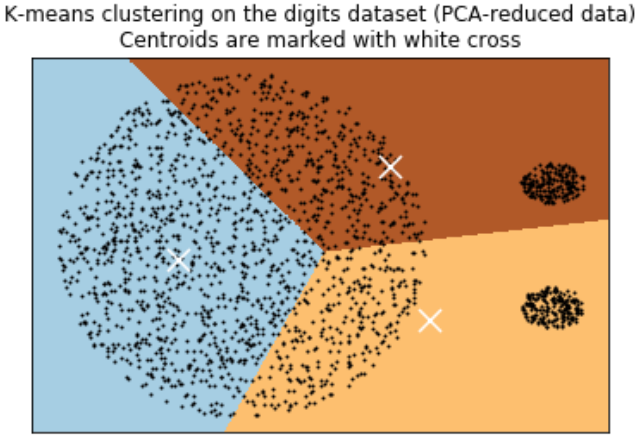


Figure 3

Pour le dataset n°3, on a donné 3 centres à l'algorithme. Contrairement à ce qu'on pourrait attendre, les centres sont mal placés. Cela est dû à la différences de densité des groupes de points.
Davies Bouldin's score : 1.0028Calinski Harabaz's score : 1242.1562

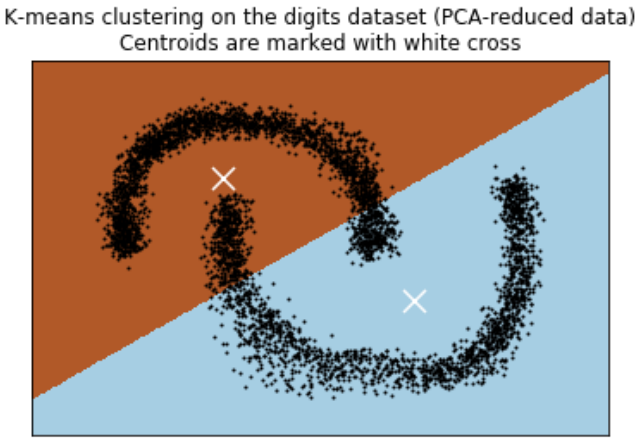


Figure 4

Pour le dataset 5, on a donné 2 centres à l'algorithme. Le résultat n'est pas satisfaisant car les clusters déterminés ne suivent pas bien les vraies formes des groupes de points. Cela est dû à la nature concave des clusters.
Davies Bouldin's score : 0.8930
Calinski Harabaz's score : 5689.2740

Nous avons réalisé des tests supplémentaires sur le dataset 6 (celui des 9 diamants). Nous avons faire varier le nombre de centres donné à l'algorithme. On a testé de 2 clusters à 15 clusters. Puis à l'aide des métriques on a déterminé le meilleur k.

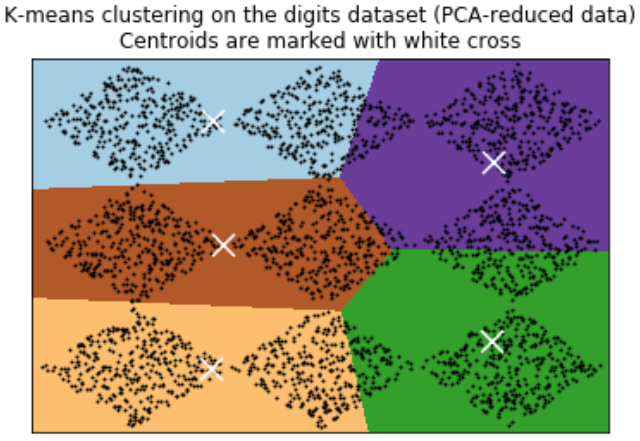


Figure 5

Dans cette figure, nous avons donné 5 centres à l'algorithme. Le clustering et les scores sont moins bons qu'avec 9 centres.
Davies Bouldin's score : 0.8692 > 0.5531
Calinski Harabaz's score : 2948.4647 < 5855.1522

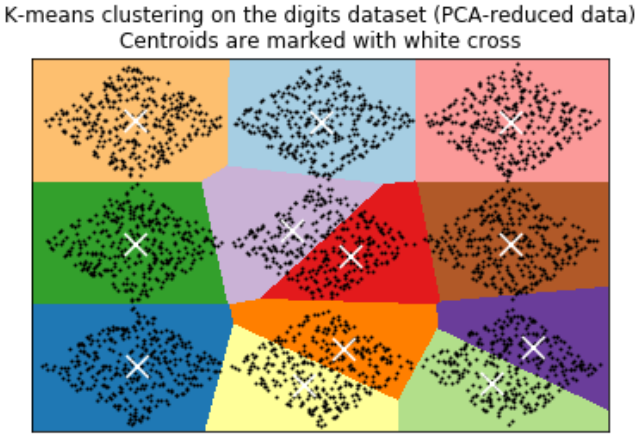


Figure 6

Ici nous avons donné 12 centres à l'algorithme. Le clustering et les scores sont moins bons qu'avec 9 centres.
Davies Bouldin's score : 0.8492 > 0.5531
Calinski Harabaz's score : 4945.9173 < 5855.1522

Le tableau ci dessous récapitule la méthode itérative pour déterminer le bon nombre de clusters (de 2 à 15 clusters) pour nos 6 datasets. Quand on parle du nombre de cluster "déterminé" (avec Davies et Calinski), c'est le nombre de cluster associé au meilleur score obtenu avec la méthode itérative. Concernant les temps de calculs, ils ont été mesurés sur la méthode de détermination entière (c'est à dire sur les 14 itérations).

Tableau récapitulatif de k-Means pour nos datasets

dataset	n°1	n°2	n°3	n°4	n°5	n°6
nb cluster	2	4	2	3	2	9
nb cluster déterminé Davies	2	4	3	6	10	9
meilleur score Davies	0.4324	0.4627	0.8049	0.6687	0.5933	0.5531
temps de calculs Davies (en s)	0.5360	0.3491	1.9183	0.9603	1.3261	0.8012
nb cluster déterminé Calinski	15	15	3	15	15	9
meilleur score Calinski	18921	1116	3981	3172	14636	5855
temps de calcul Calinski (en s)	0.5223	0.3372	1.9672	0.9513	1.3837	0.7987

On constate que l'indice de Davies Bouldin sur ces datasets avec la méthode k-Means est plus pertinent que le score de Calinski. En effet, à 2 reprises le score de Calinski détermine un mauvais nombre de clusters alors que le score de Davies a son meilleur score pour le bon nombre de cluster (pour les dataset 1 et 2). De plus quand les 2 scores ne déterminent pas le bon nombre de clusters, Davies est toujours le plus proche du vrai nombre de cluster (pour les datasets 3,4 et 5).

D'ailleurs pour ces datasets (3,4,5), la méthode k-Means ne semble pas correctement partitionner les clusters. Pour le dataset 3 c'est probablement car les 2 cluster sont très mal séparés. Pour le 4 c'est la différence de densité qui explique probablement le manque d'efficacité de la méthode (voir figure 3 au-dessus). Et pour le dataset 5 c'est sûrement car les formes des clusters sont concaves (voir figure 4 au-dessus). En revanche un peu de bruit ne semble pas avoir trop d'impact sur cette méthode (en tout cas avec le score de Davies Bouldin) comme le montre bien la figure 2. Par contre le bruit n'est pas clairement identifié, ni isolé, et il est classé dans différents clusters.

On peut éviter de tester trop de valeurs différentes de k, aussi bien en utilisant l'indice de Davies Bouldin que celui de Calinski. En effet, dès qu'un maximum local (ou minimum local pour Davies Boulvin) est atteint, c'est à tous les coups le meilleur k possible (en tout cas sur nos datasets). Donc dès que le score pour la valeur n+1 est moins bon que celui de la valeur n, on peut s'arrêter car on sait qu'on aura pas de meilleur score que pour la valeur n.

Le choix des points initiaux a parfois un impact sur le résultat. En effet, lorsque l'on utilise init='random' en paramètre cela change parfois le résultat (le meilleur k déterminé). C'est par exemple le cas sur le dataset n°5, le meilleur k déterminé avec Davies Bouldin est égal à 10 pour l'initialisation "k-means ++" et parfois avec l'initialisation random il prend la valeur 14. K-means a une certaine sensibilité à l'initialisation, ce qui est cohérent avec la stratégie gloutonne de cet algorithme..

La méthode semble sensible à la nature des formes, en effet on constate que sur les formes concaves la partition des clusters n 'est pas parfaite (figure 3). De plus k-means détermine des centres de gravité et chaque point d'un cluster est plus

proche de son centre de gravités que des autres centre, ce qui explique notamment que les formes concaves ne soient pas adaptés pour cette méthode.

3-Agglomératif

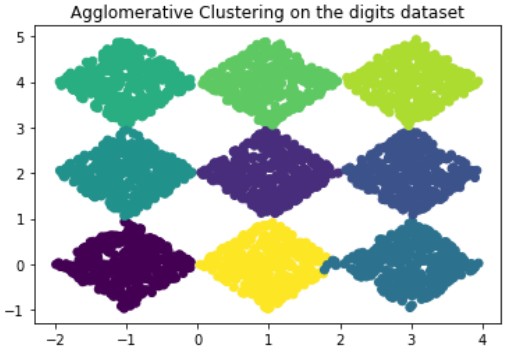
Voici un tableau récapitulant la méthode de détermination du nombre de cluster d'un dataset. Il faut savoir que cette méthode a été réalisé avec 2 différents critères d'évaluation (Davies Bouldin et Calinski Harabasz) mais surtout sur 4 manières de combiner des cluster différentes (single, average, complete et ward). On a donc effectué 8 recherches itératives avec des paramètres différents sur chaque dataset. Il faut noter que dans ce tableau, on retrouve seulement les meilleures (et les pires) combinaisons/scores. Dans ce tableau, il n'y a pas le détail complet de chaque score pour toutes les combinaisons. Pour avoir les détails, il suffit d'utiliser la méthode itérative de détermination du nombre de cluster dans la partie associée du code.

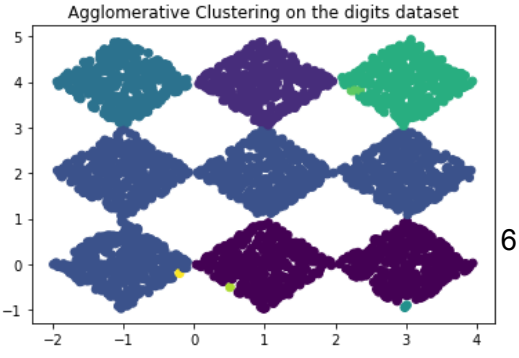
Tableau récapitulatif de l'Agglomerative Clustering pour nos datasets

dataset	n°1	n°2	n°3	n°4	n°5	n°6
nb cluster	2	4	2	3	2	9
meilleure prédiction nb cluster	2	4	3	3	2	9
Les meilleures combinaison de cluster et score utilisé (Davies et Calinski)	Toutes avec le score de Davies	Ward (Davies) et Complete avec les 2 scores	Average (Calinski) et Ward pour les 2 scores	Single pour les 2 scores	Single pour les 2 scores	Toutes les méthodes sauf Single
Pires prédiction nb cluster	15	15 et 10	12, 7 et 6	14,13,12	15,13,12,11	12 et 2
Les pires combinaison de cluster et score utilisé	Toutes avec le score de Calinski (Single pas trop mauvaise)	Ward (Calinski) et Average (Calinski)	Single (les 2 scores) et Average (Calinski)	Toutes les autres (Ward Calinski pas trop mauvaise)	Toutes les autres	Single avec les 2 score différents

Tout d'abord, on constate qu'il n'y a jamais de datasets où toutes les combinaisons et tous les scores sont bons. De plus, il n'y a pas vraiment de combinaisons plus efficaces que d'autre dans toutes les situations. On constate par exemple que la méthode "single" est mauvaise pour le dataset n°6 et pour le dataset n°3 mais qu'elle est la plus efficace sur les datasets n°4 et n°5.

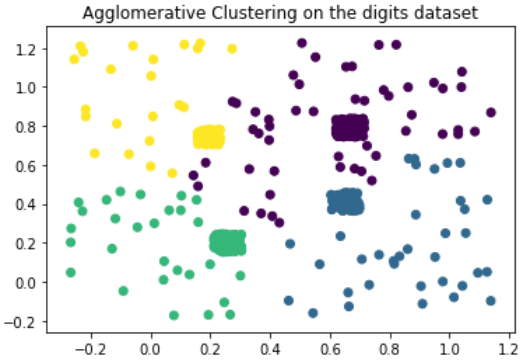
Pour le dataset de base (le dataset n°6), on constate que toutes les combinaisons (hormis "single") permettent de déterminer correctement le bon nombre de clusters.

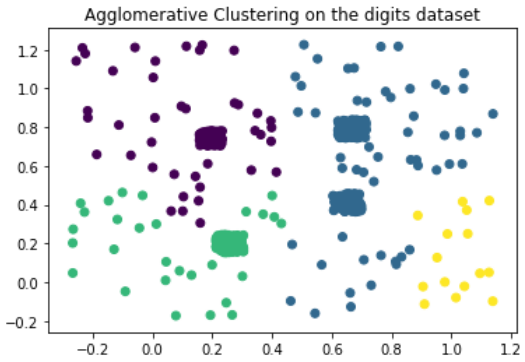




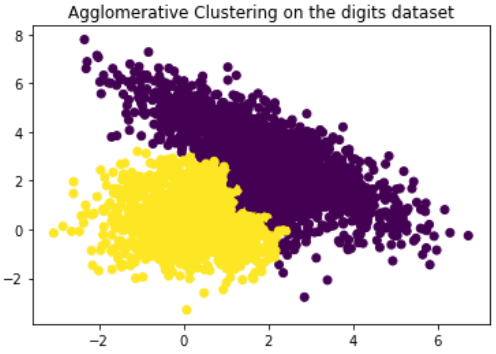
On voit au dessus que pour k = 9 la méthode par défaut “ward” (à gauche) les clusters sont plutôt bien partitionnés ce qui n’est pas du tout le cas pour la combinaison “single” (à droite).

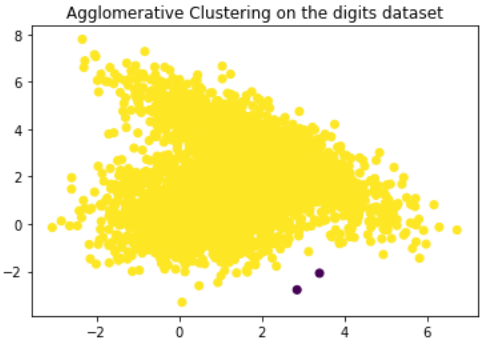
Pour le dataset n°2 qui est bruité, on constate que les combinaisons “complete” et “ward” (Davies) déterminent correctement le bon nombre de clusters. Les autres combinaisons semblent affectées par le bruit. Pour k = 4, on a à gauche “complete” et à droite “average”. Cependant, le bruit n’est pas isolé du reste (comme pour k-Means).



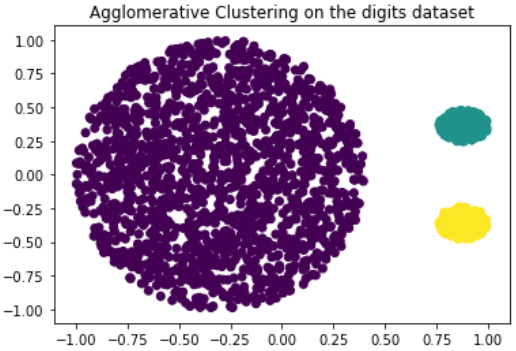


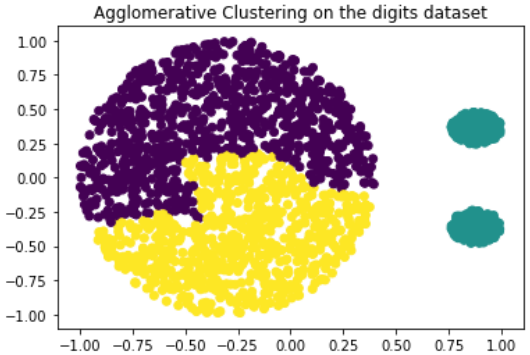
Pour le dataset n°3, on arrive pas à déterminer correctement le bon nombre de cluster car ils sont mal séparés. Cependant les combinaisons “ward” et “average” (Calinski) trouvent tout de même un résultat proche de la vérité avec 3 clusters (au lieu de 2). Voici d’ailleurs ce que trouve la méthode “ward” pour k = 2 (à gauche). A droite avec les même k, on retrouve la “pire” combinaison pour ce dataset (“single”).

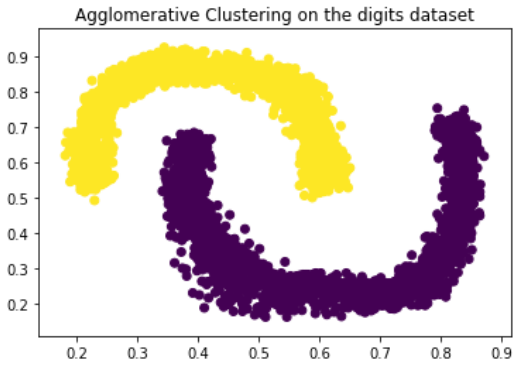


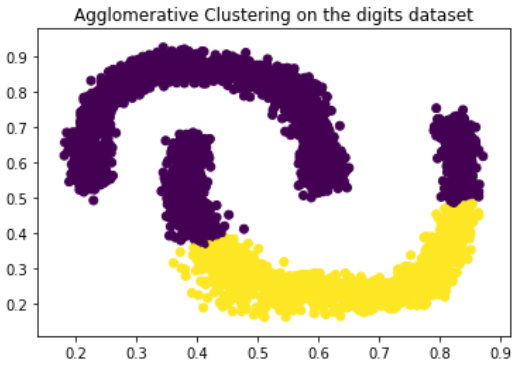


Pour les datasets n°4 et n°5, la combinaison “single” est la seule qui arrive à déterminer correctement le bon nombre de clusters. Cela s’explique à cause de la différence de densité pour le dataset n°4. Pour le dataset n°5 cela s’explique car les formes des clusters dans le dataset sont concaves. A gauche on retrouve “single” et à droite “ward” (k = 3 pour le dataset 4 et k=2 pour le dataset 5).



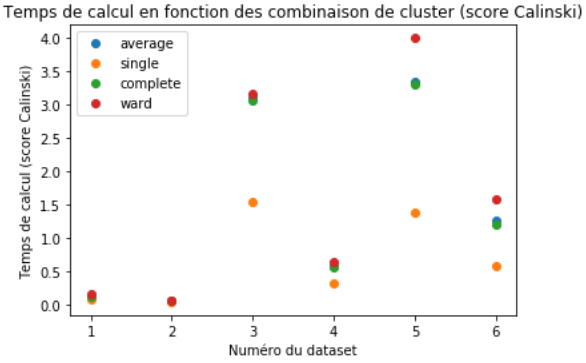


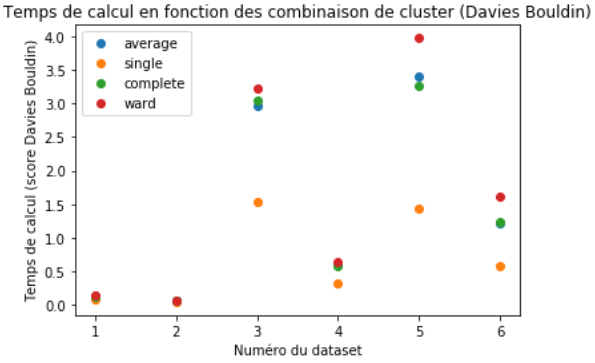




Les différentes combinaisons de clusters ne sont pas toutes adaptées en fonction du type de cluster. En effet, "single" semble efficace pour les clusters avec des formes concaves ou de densités variables contrairement aux autres combinaisons. Mais dès que les formes sont bruitées, mal séparées (ou pas trop nettement séparées comme le dataset des 9 diamants) le partitionnage à l'aide de cette méthode est beaucoup moins bon que les autres combinaisons. Cela s'explique car pour "single", le calcul de similarité entre cluster est la distance entre les 2 points les plus proches. Pour les datasets avec des formes convexes mais qui sont bruités ou pas très bien séparés, on préférera utiliser la combinaison par défaut ("ward").

De plus, en ce qui concerne les temps d'exécution des différentes combinaisons, on constate que la méthode single est toujours la plus rapide peu importe le dataset, et que par contre la méthode "ward" est toujours la plus longue (voir les graphes ci-dessous). Cela parait logique vu que "ward" est une combinaison un peu plus complexe que "single". "Ward" cherche à maximiser l'homogénéité des clusters en prenant la moyenne des distances au carré entre chaque point et le centre des clusters.

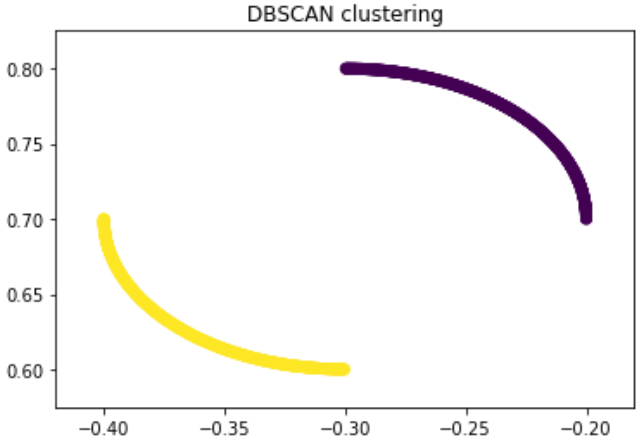




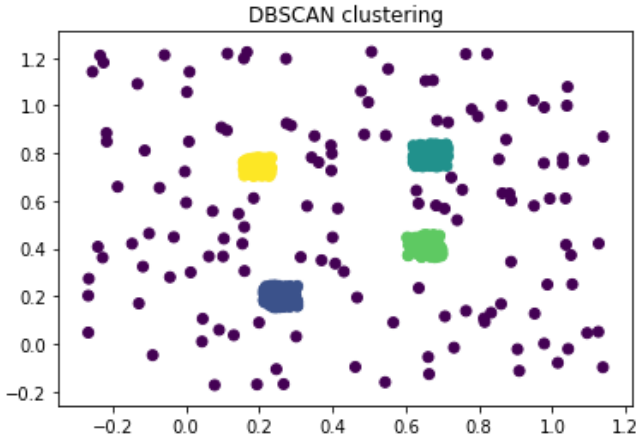
Concernant les différentes métriques de distance, déjà il faut savoir que "ward" ne fonctionne qu'avec la distance euclidienne. De plus, je n'ai pas vu beaucoup de différence de résultats en utilisant différentes métriques de distance (en ce qui concerne la détermination du bon nombre de clusters). La seule différence que l'on a observé, et qui permet d'avoir un meilleur résultat, c'est sur le dataset 2 où au lieu de trouver k=5 (sachant qu'ici le 5ème cluster identifié ne correspond pas du tout à l'isolement du bruit) avec la combinaison single et la distance euclidienne, on trouve le bon nombre de cluster (k=4) en utilisant la distance de manhattan ("affinity " = "manhattan').

4-DBSCAN

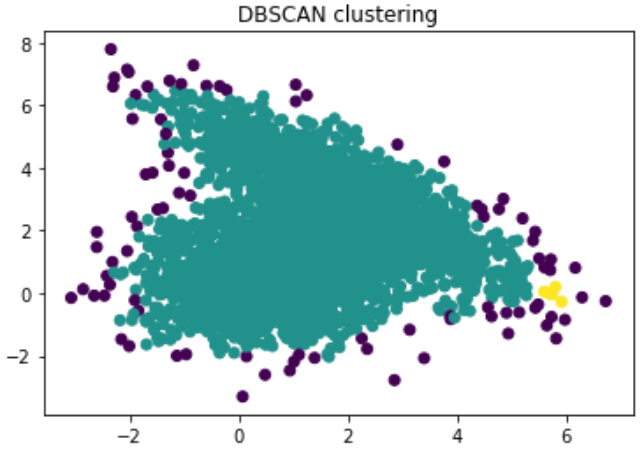
L'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise) sert à partitionner les données. Nous avons voulu tester cet algorithme sur les différentes données présentées en introduction.



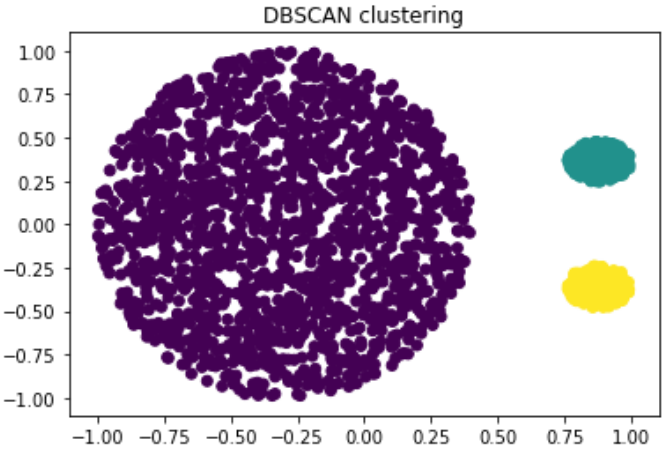
Sur le dataset 1 les données sont bien séparées et l'algorithme fonctionne très bien.
Davies Bouldin's score : 0.4324
Calinski Harabaz's score : 4270.4427
Epsilon = 0.05
Samples = 5



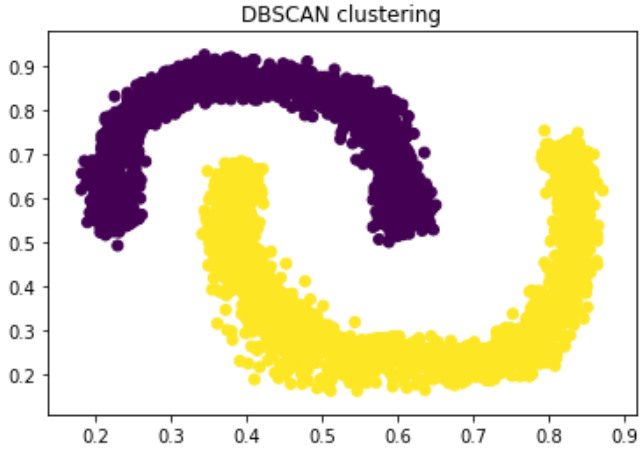
Sur le dataset 2, on peut remarquer que même avec du bruit l'algorithme fonctionne bien. On note même que le bruit est bien identifié.
Davies Bouldin's score : 1.9898
Calinski Harabaz's score : 168.6166
Epsilon = 0.05
Samples = 5



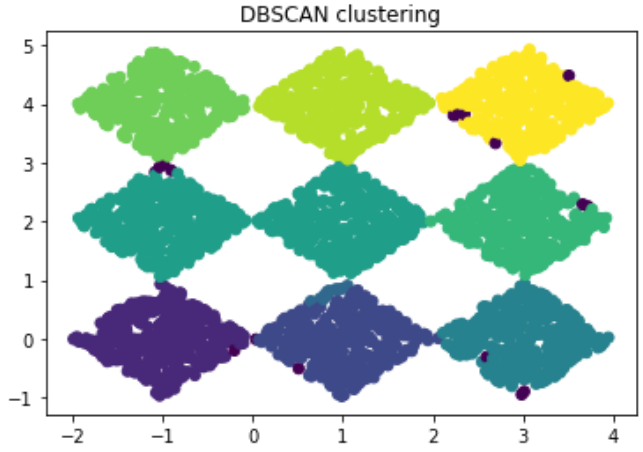
Sur le dataset 3, l'algorithme ne parvient pas à différencier les clusters. Cela est dû au fait que les clusters sont mal séparés, que les frontières entre les deux sont mal délimités.
Davies Bouldin's score : 31.0794
Calinski Harabaz's score : 11.4550
Epsilon = 0.3
Samples = 5



Sur le dataset 4, les clusters sont bien mis en évidence (contrairement à ce même dataset avec k-means). Les cercles étant de densités différentes, on peut dire que l'algorithme n'est pas sensible aux différences de densité.
Davies Bouldin's score : 0.5255
Calinski Harabaz's score : 840.1500
Epsilon = 0.1
Samples = 5



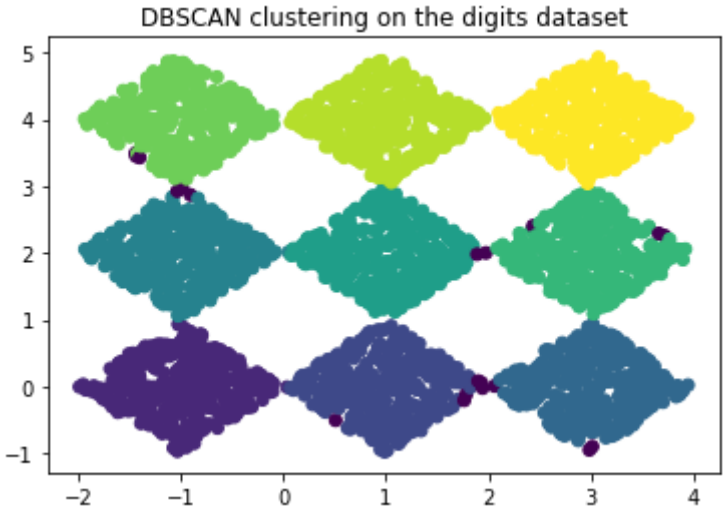
Sur le dataset 5, l'algorithme fonctionne parfaitement malgré les formes concaves (K-means ne marchait pas avec ce dataset).
Davies Bouldin's score : 1.0988
Calinski Harabaz's score : 3751.7304
Epsilon = 0.05
Samples = 5



Sur le dataset 6, l'algorithme a des difficultés pour différencier les clusters. Cela est dû au fait que les clusters se touchent les uns les autres.
Davies Bouldin's score : 1.6318
Calinski Harabaz's score : 2865.6652
Epsilon = 0.13
Samples = 5

Les scores de Davies Bouldin's et Calinski Harabaz ne semblent pas suffisamment pertinents. On a remarqué qu'un score moins bon donnait parfois un clustering semblant meilleur lors de l'affichage. Nous avons réalisé une fonction appelée calc_score prenant en compte les vrais labels des points. Grâce à cette fonction on peut donc savoir exactement le nombre de points bien classés par l'algorithme. (L'apprentissage est toujours non supervisé car on n'utilise pas les labels lors de l'apprentissage). Avec cette fonction on sait que sur la figure précédente **77%** des points sont bien classifiés.

Le problème de DBSCAN est la recherche des paramètres optimaux. Pour les trouver, nous avons réalisé une fonction comportant deux boucles "for" imbriquées, itérants sur epsilon et le nombre de samples.

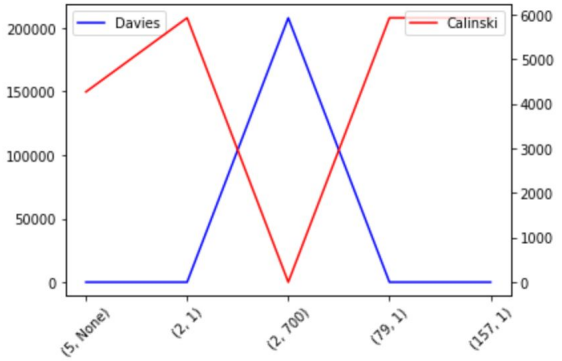


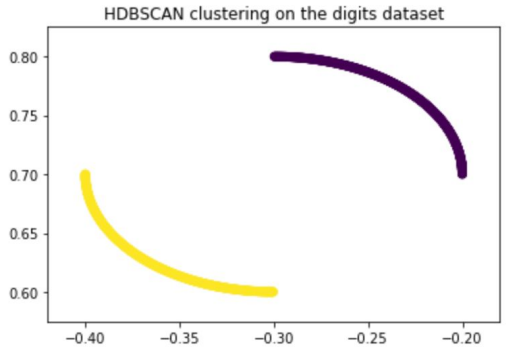
Pour le dataset 6 nous avons trouvé comme paramètres optimaux :
Epsilon = 0.1459
Samples = 7
Avec :
Davies Bouldin's score : 1.7951 (Moins bon qu'avant ce qui est étonnant car le clustering est quand même meilleur)
Calinski Harabaz's score : 4728.2332
(Ici le score est meilleur qu'avec les paramètres précédents)
Score vrai : **99.3%** de points bien classifiés contre **77%** avec les paramètres précédents.

5-HDBSCAN

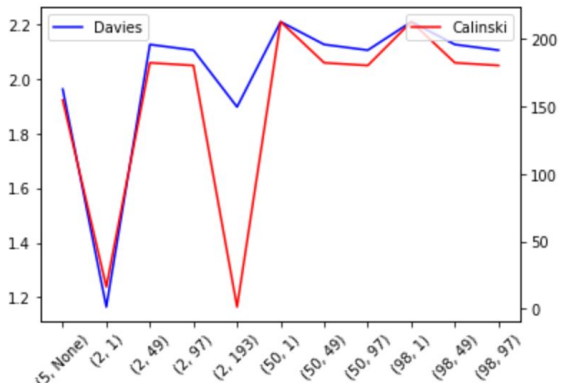
L'algorithme HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) sert à partitionner les données, comme HDBSCAN, mais prend des paramètres différents : au lieu de lui donner epsilon, distance max entre 2 point du même cluster, nous donnons le nombre minimum de points pour un cluster ainsi que le nombre minimum d'échantillons pour qu'un point soit considéré comme un "core point". Cela va permettre de faciliter la recherche des paramètres optimaux. Nous avons voulu tester cet algorithme sur les même données que précédemment de manière à pouvoir comparer les deux.

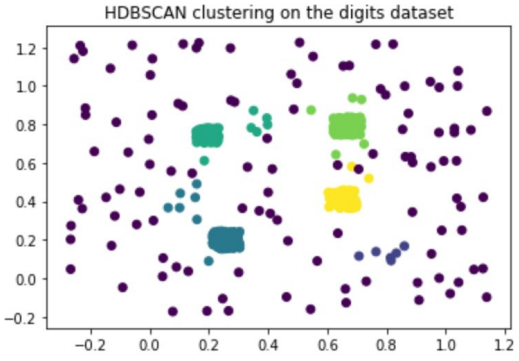
100 combinaisons de min_cluster_size et min_sample vont être testées afin de voir les différences de score selon les paramètres (en ne gardant que les classifications à plus de 2 clusters). Nous testerons aussi les paramètres par défaut (premier point de chaque plot sur la suite).



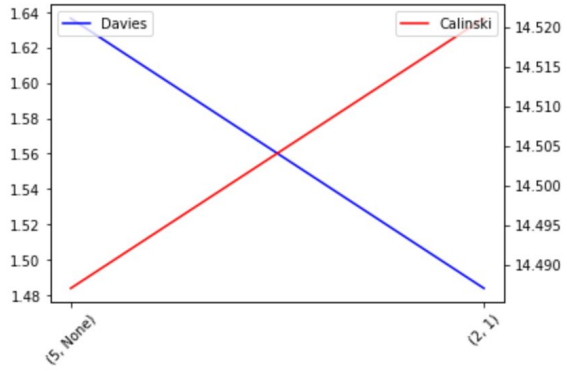


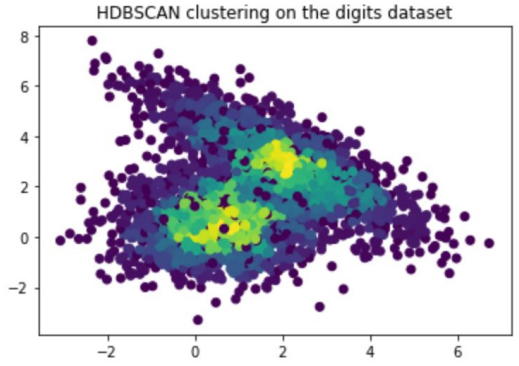
Pour le premier dataset, on remarque clairement que 3 couples de paramètres sortent du lot, mais après affichage des clusters, seul le 1er couple, celui par défaut, nous prédit le bon nombre de cluster. Pour les autres, les scores sont bon uniquement car les données ne sont pas bruitées et bien séparées, mais trop de clusters sont classifiés.



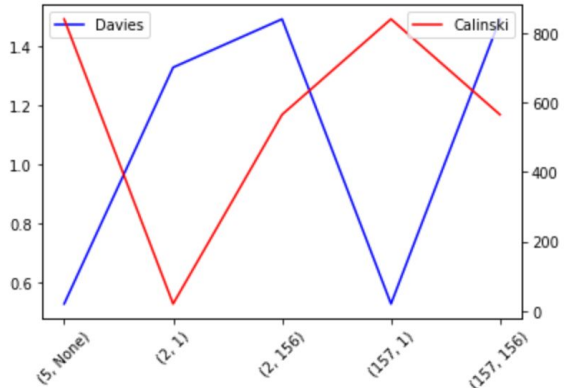


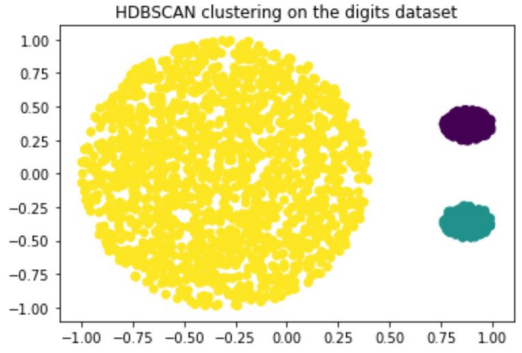
Pour le dataset 2, on remarque que toutes les combinaisons (hormis 2) ont des scores plutôt proches les unes des autres, mais à l'affichage, aucunes d'entre elles n'arrive à séparer correctement les clusters. DBSCAN avec son paramètre epsilon, permet d'avoir une séparation plus fine et d'obtenir un meilleur résultat.



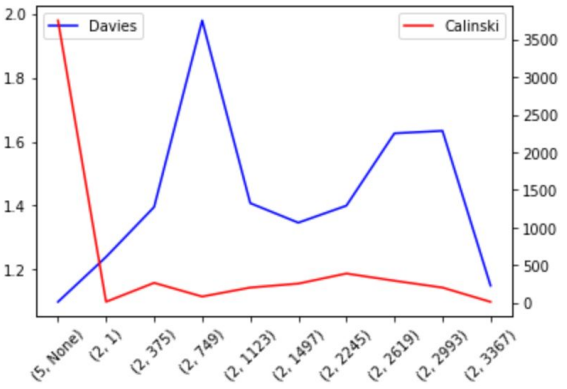


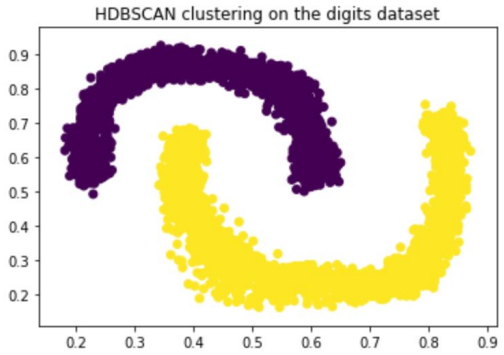
Pour le dataset 3, seul 2 combinaisons ont réussi à prédire plus de 2 cluster, mais comme pour DBSCAN, la conclusion est la même: des cluster mal, voir non séparés sont extrêmement difficiles à interpréter et à classifier.



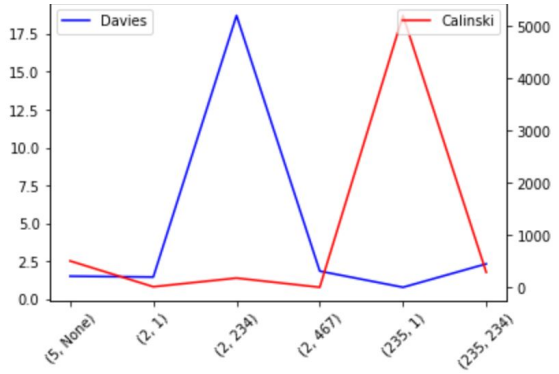


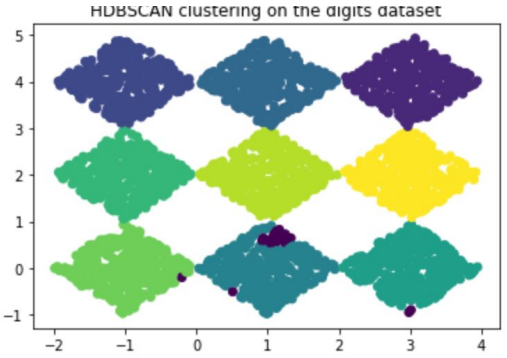
Pour le dataset 4, aucun problème pour prédire le bon nombre de clusters et classifier chaque point. Les données étant séparées et non bruitées, c'est facile et rapide. On peut remarquer que les paramètres par défaut fonctionnent ici très bien.





Pour le 5eme dataset, seul les paramètres par défaut fonctionnent correctement et permettent d'avoir des scores corrects, et on voit que tous les points sont bien classifiés dans le bon cluster.



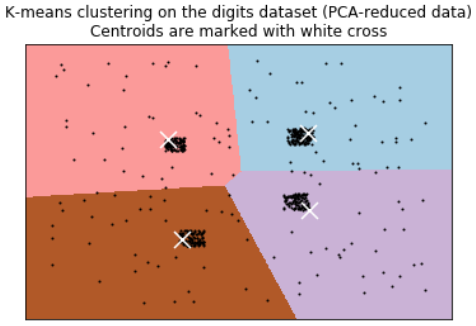


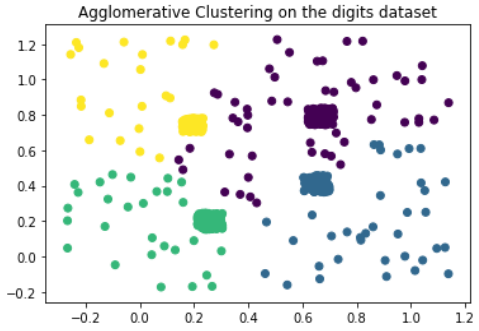
Pour le dernier dataset, à l'inverse du 5eme, les paramètres par défaut ne marchent pas du tout, et les points sont classifiés n'importe comment. En revanche, une combinaison sort du lot (235,1) avec un très bon score nettement supérieur aux autres. A l'affichage, on voit bien la séparation entre les clusters, mais HDBSCAN en prédit un de trop. Un ajustement plus fin avec des pas plus petits dans notre boucle aurait probablement permis de trouver les paramètres parfaits de manière à résoudre ces mauvaises classifications.

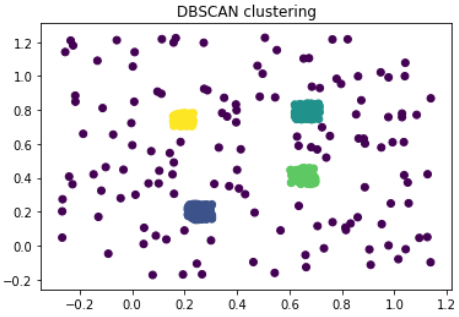
En conclusion, avec des paramètres optimaux, DBSCAN et HDBSCAN ont des résultats très similaires, avec des bonne classifications, mais aussi les mêmes défauts, avec des difficultés sur les données très peu séparées. En revanche du point de vue de la facilité d'utilisation et la facilité à trouver ces mêmes paramètres optimaux, une grande différence peut être remarquée. En effet pour DBSCAN, une longue boucle itérant sur beaucoup d'epsilons et de samples est nécessaire, d'autant plus que epsilon est sensible, intrinsèquement, aux échelles des données, alors que pour HDBSCAN, les paramètres par défaut fonctionnent très bien et sont très souvent les meilleurs. Cela permet de gagner beaucoup de temps sur beaucoup de datasets, et au cas où ça ne marche pas, il suffit de faire une boucle comme avec DBSCAN.

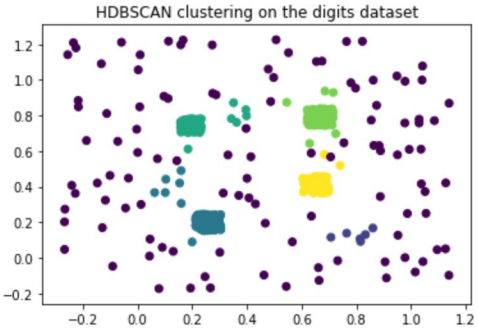
6-Synthèse

Dans cette partie on va comparer les différentes méthodes de clustering en observant les résultats sur les mêmes jeux de données.

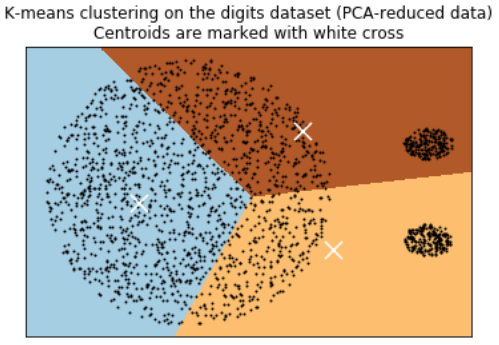


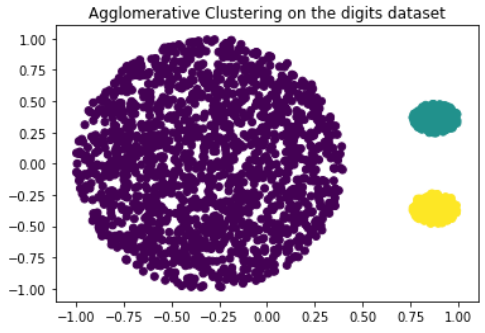


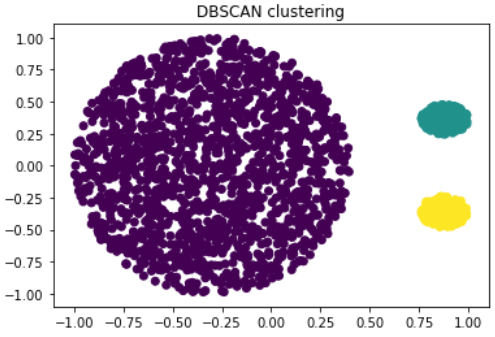


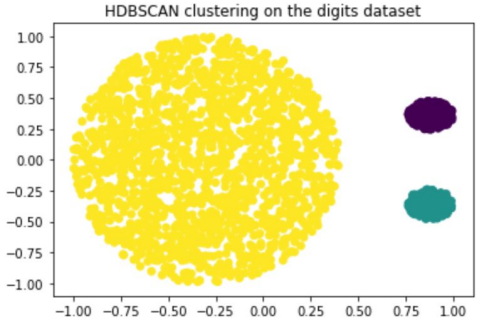


Sur le jeu de données avec du bruit, on remarque que K-means et l'agglomerative déterminent le bon nombre de clusters mais n'arrivent pas correctement à isoler le bruit. Par contre, DBSCAN et HDBSCAN sont capables de déterminer les bons clusters tout en isolant le bruit correctement (cluster violets sur les graphes). On remarque que DBSCAN a, avec un peu de chance sur les paramètres, parfaitement isolé le bruit ce qui n'est pas tout à fait le cas pour HDSBCAN.

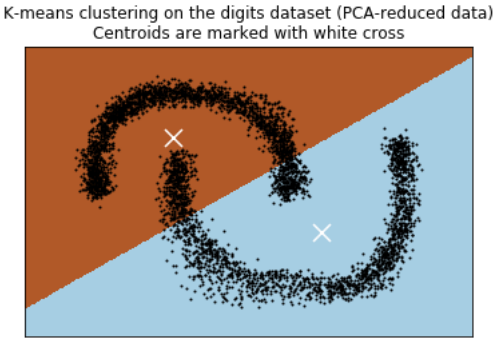


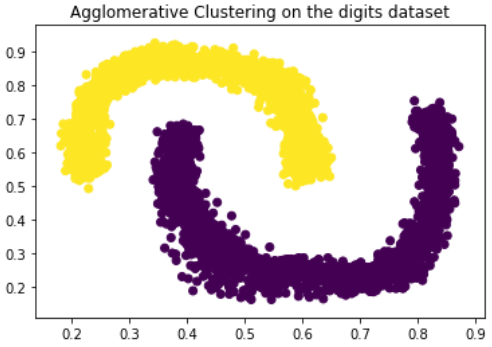


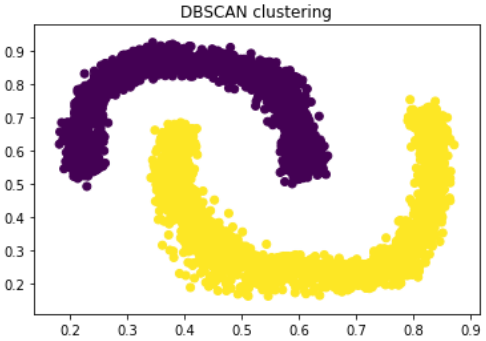


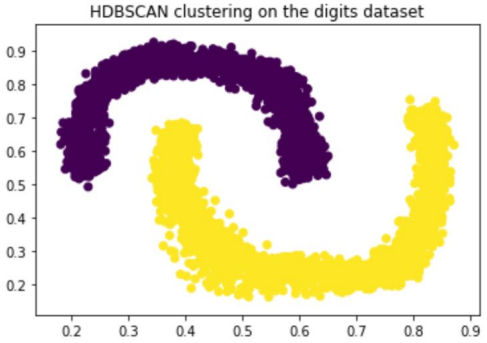


Pour ce dataset où les formes sont de densités variables, on constate que seul K-Means n'arrive pas du tout à déterminer correctement les 3 clusters. Il faut cependant noter que pour l'agglomerative clustering on a utilisé le paramètre "single" (les autres combinaisons ne marchaient pas du tout pour ce dataset là).

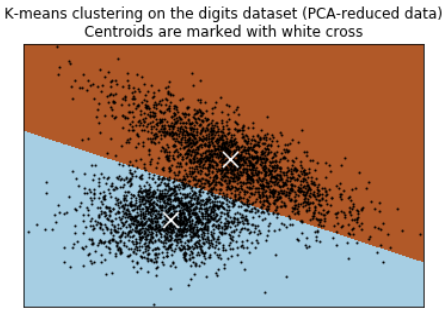


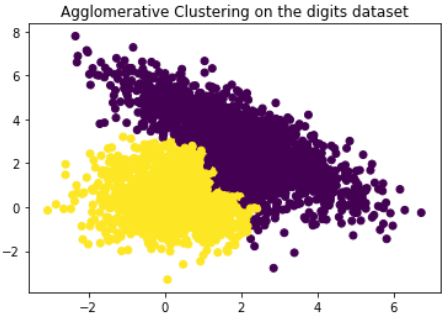


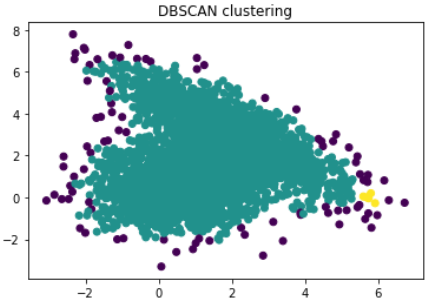


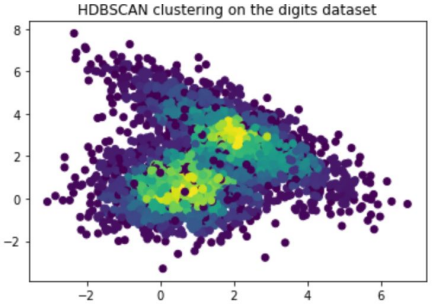


Même constat pour le dataset avec des formes concaves que pour le dataset précédent.









Pour ce dataset où les 2 clusters sont mal séparés, on se rend compte que K-means et Agglomerative ne se débrouillent pas trop mal pour différencier les 2 clusters (la frontière entre les 2 clusters est quand même assez différente que celle du dataset original). En revanche pour DBSCAN et HDBSCAN ils ne s'en sortent pas bien du tout comme on peut le voir sur les 2 graphes juste au-dessus.