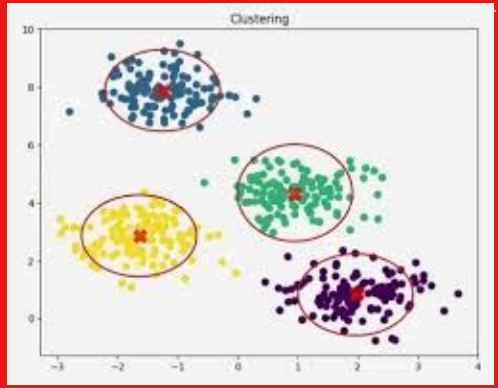
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Clustering

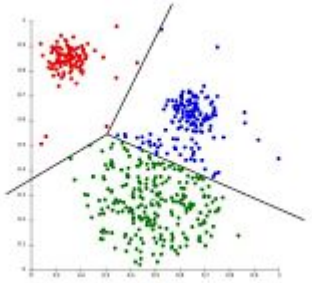


Hassouna Mohamed Amine

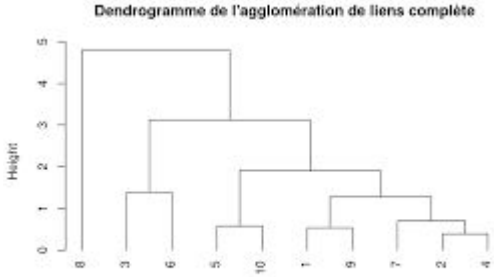
Laforge Mateo

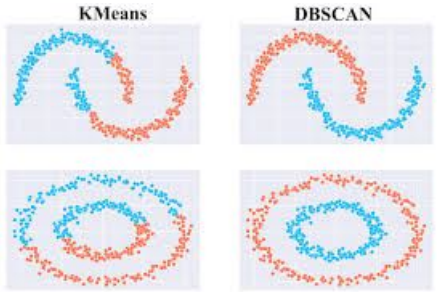
PLAN :

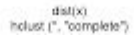
- Présentation du jeux de données
- Méthode des K-Means
- Méthode Aggléomératif
- Méthode DBSCAN
- Méthode HDBSCAN



  - Principe de fonctionnement et caractéristiques principales
  - Résultat Visuels
  - Automatisation de choix des paramètres
  - Avantages et limites de la méthode

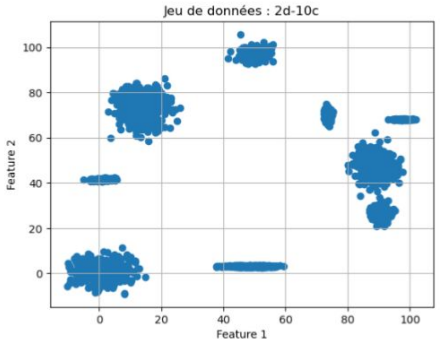






Présentation du jeux de données

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import arff
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

dataset_name = 'smile3.arff'

def load_arff_data(file_path):
    """Load numeric data from a .arff file and return as NumPy array."""
    data, meta = arff.loadarff(file_path)
    X = np.array([list(row)[:-1] for row in data], dtype=float)
    return X

file_path = os.path.join('../dataset', 'artificial', dataset_name)
X = load_arff_data(file_path)
print("Shape du dataset :", X.shape)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

plt.scatter(X_scaled[:, 0], X_scaled[:, 1], s=10, c='blue')
plt.title(f"Visualisation du dataset {dataset_name}")
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")
plt.show()
```



❌ mauvais choix de centres
❌ clusters mal séparés
❌ silhouette score plus faible
❌ interprétation incorrecte des résultats

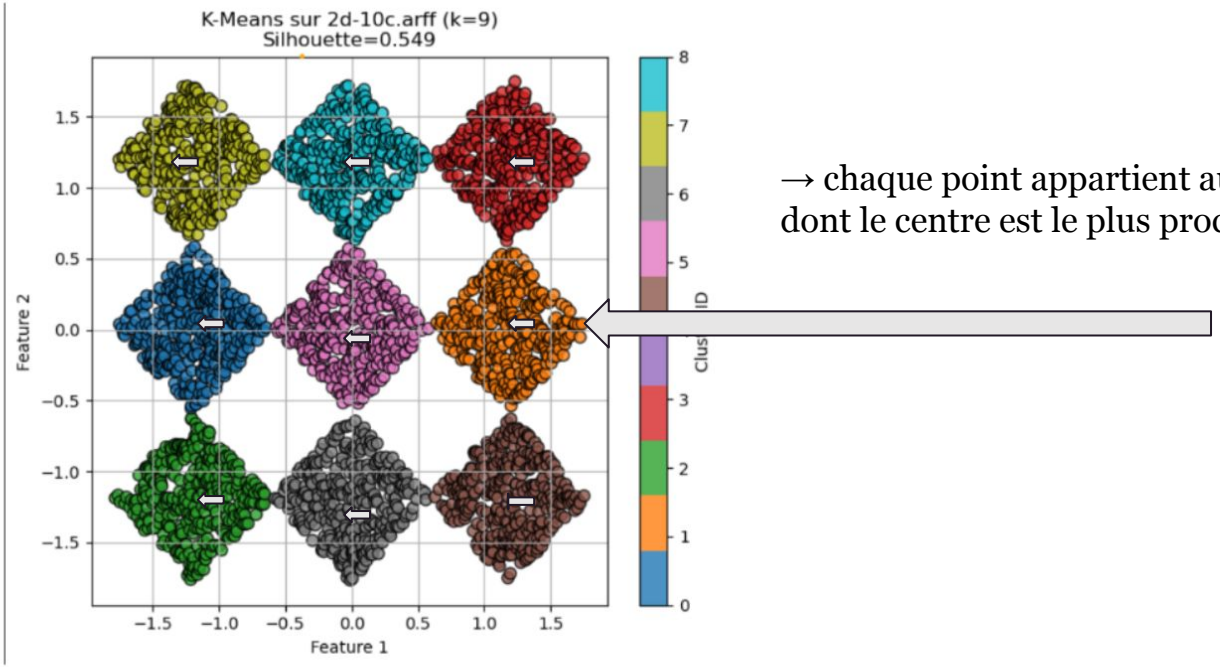
Méthode des K-Means :

Principe : une méthode itérative de clustering partionnel

Objectif : minimiser la distance intra-cluster : distance moyennes des exemples au centre de leur cluster

Entrée : nombre de clusters fixé en Avance

Forme de clusters uniquement convexes et sphériques

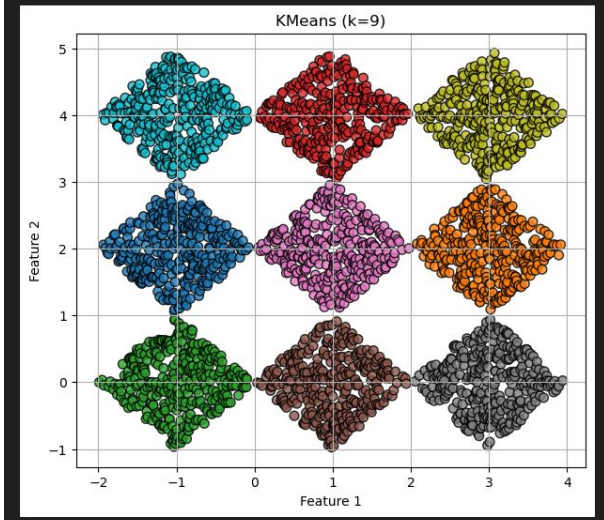


→ chaque point appartient au cluster dont le centre est le plus proche

FIGURE 2.1 – Résultat du K-Means sur le dataset `diamond9.arff` avec $K = 9$

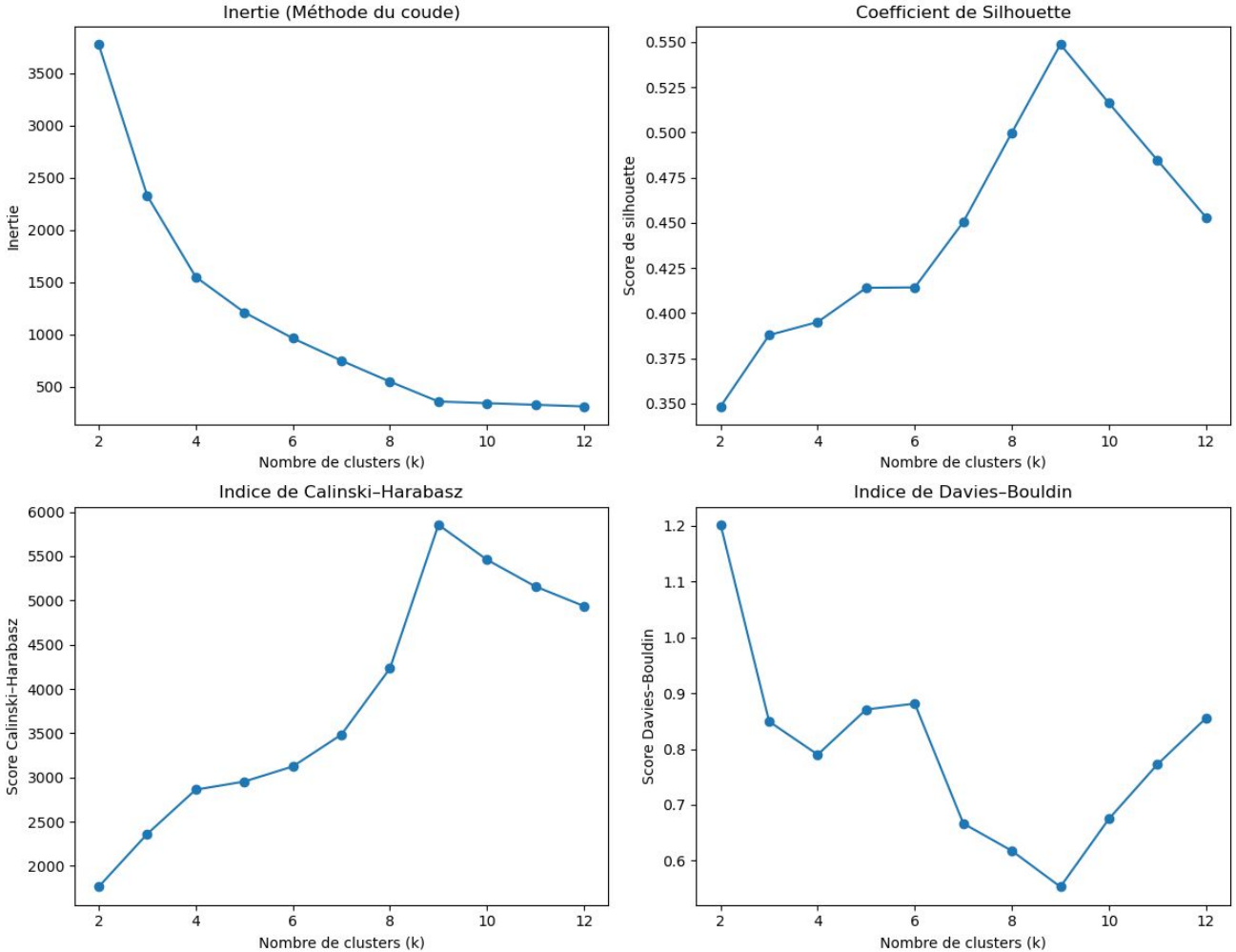
Automatisation de choix des paramètres

Shape du dataset : (3000, 2)

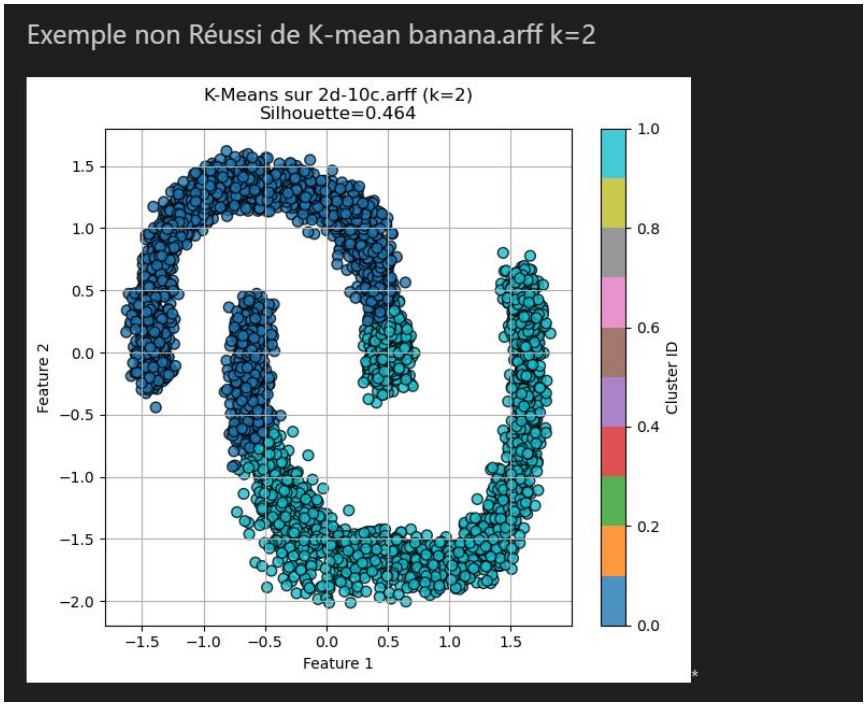



```python
import numpy as np

from sklearn.cluster import KMeans
from Method import Method

def elbow_kmeans(data, k_max=10):
    """
    Trouve le meilleur k pour k-means en utilisant la méthode du coude.

    Paramètres:
    - data: numpy array ou pandas DataFrame, les données à clusteriser.
    - k_max: int, le nombre maximum de clusters à tester.

    Retourne:
    - kmeans: KMeans, instance d'object KMeans entraînée
    - meilleur_k: int, le k suggéré par la méthode du coude.
    - inerties: list, les inerties pour chaque k (pour tracer le graphique).
    """
    inerties = []
    kmeans_liste: list[KMeans] = []
    for k in range(1, k_max + 1):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data)
        kmeans_liste.append(kmeans)
        inerties.append(kmeans.inertia_)

    # Calcul des différences pour trouver le "coude"
    diffs = np.diff(inerties)
    diff_ratios = diffs[:-1] / diffs[1:]
    meilleur_k = np.argmax(diff_ratios) + 2  # +2 car diff commence à k=2
    kmeans: KMeans = kmeans_liste[meilleur_k-1]

    return kmeans, meilleur_k

class KMeansMethod(Method):

    def predict(self, X):
        kmean_inst, _ = elbow_kmeans(X, 10)
        labels = kmean_inst.predict(X)
        return labels

    def __repr__(self):
        return "k-means"
```

Comparaison des métriques de clustering sur diamond9.arff

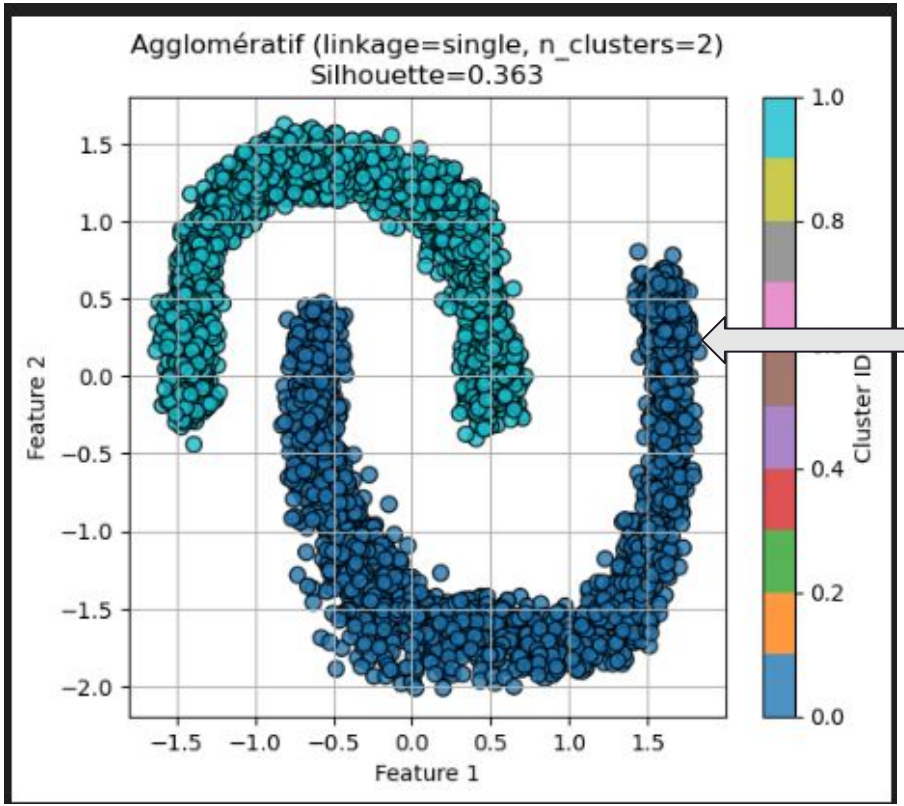


Limites : Forme de clusters uniquement convexes et sphériques



⇨ Les barycentres des solutions ne font pas parties des clusters eux mêmes

Méthode Aggléomératifs:

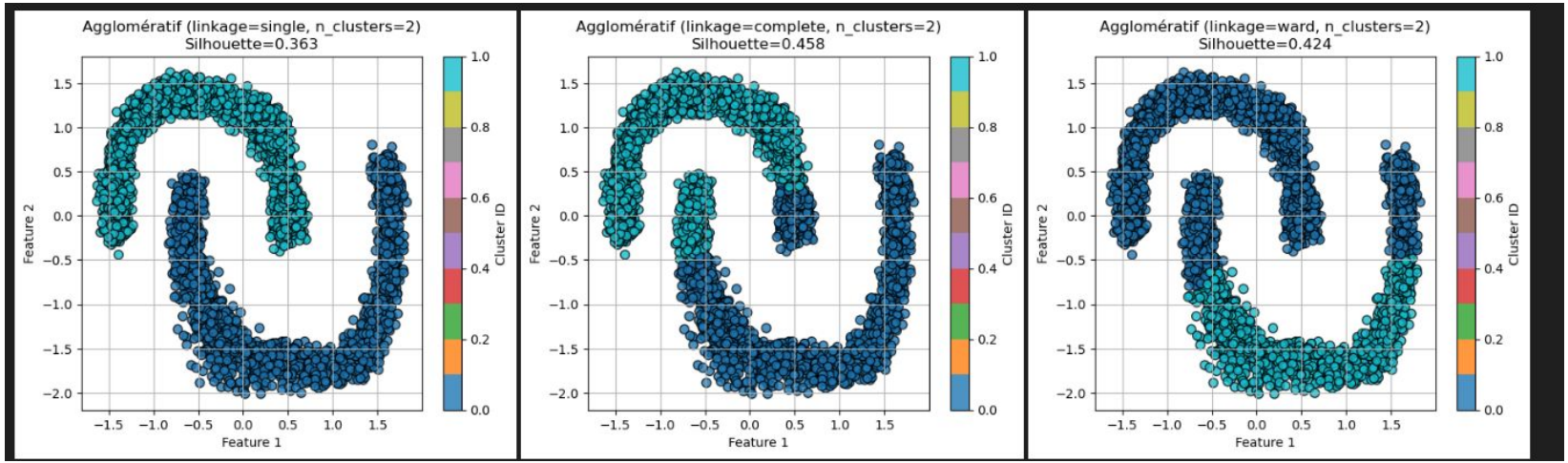


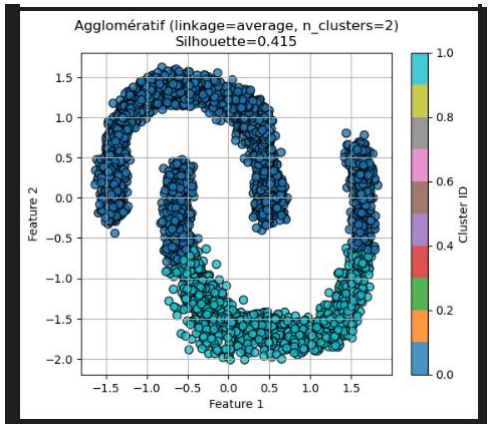
Contrairement à K-Means :

- pas besoin de centres de clusters
- pas besoin de clusters convexes ou sphériques
- pas de notion d'inertie à minimiser

→ On part de **chaque point comme un cluster individuel**,
→ Puis on **fusionne progressivement** les clusters les plus proches,
Jusqu'à n'obtenir que le nombre de clusters souhaité.

Les Linkages (règles de fusions):
comment mesurer la distance entre deux clusters ?





➢ **Single linkage :** Distance = **distance minimale** entre un point du cluster A et un point du cluster B

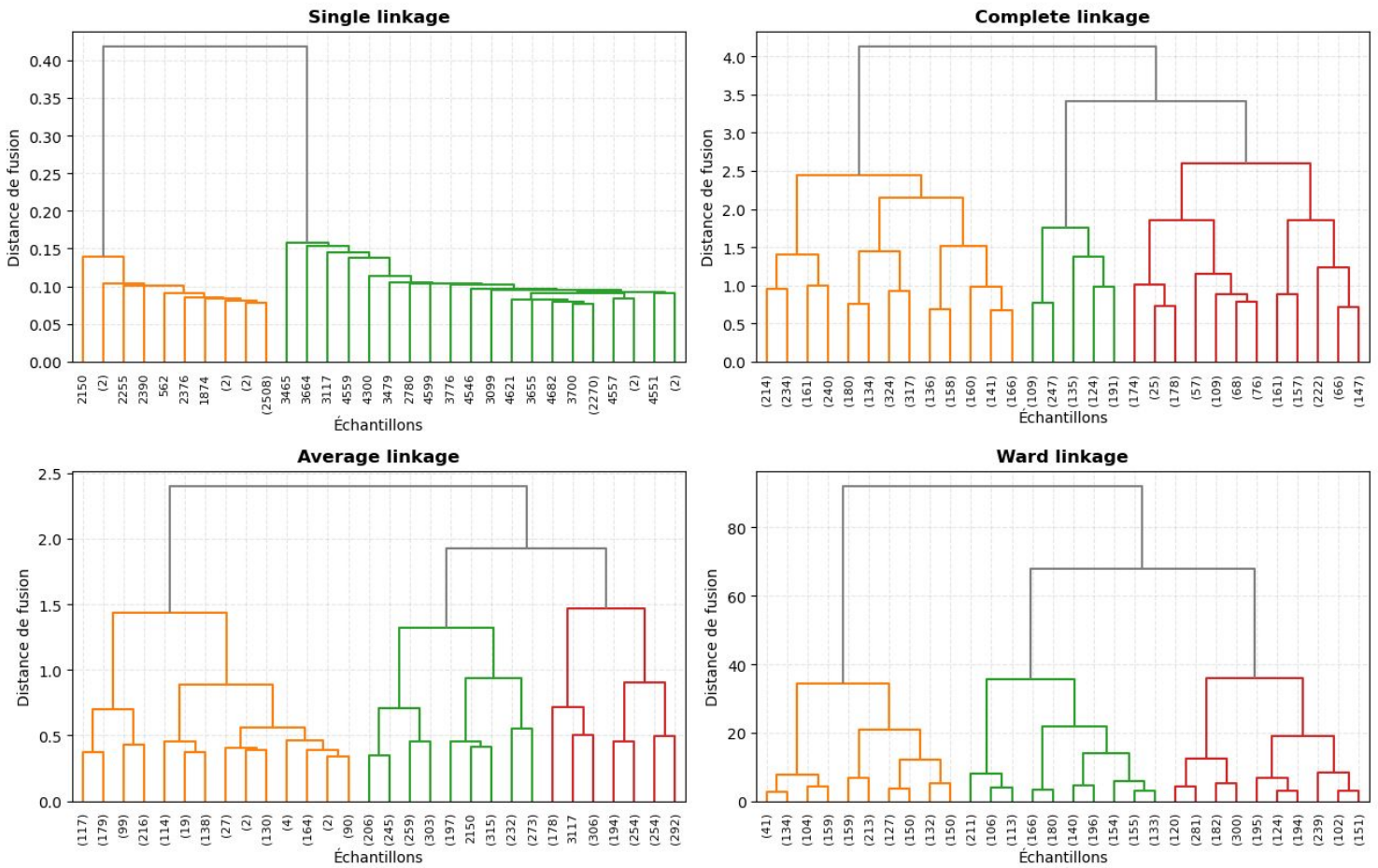
➢ **Complete linkage :** Distance = **distance maximale** entre les points des deux clusters

➢ **Average linkage :** Distance = moyenne des distances entre tous les couples de points

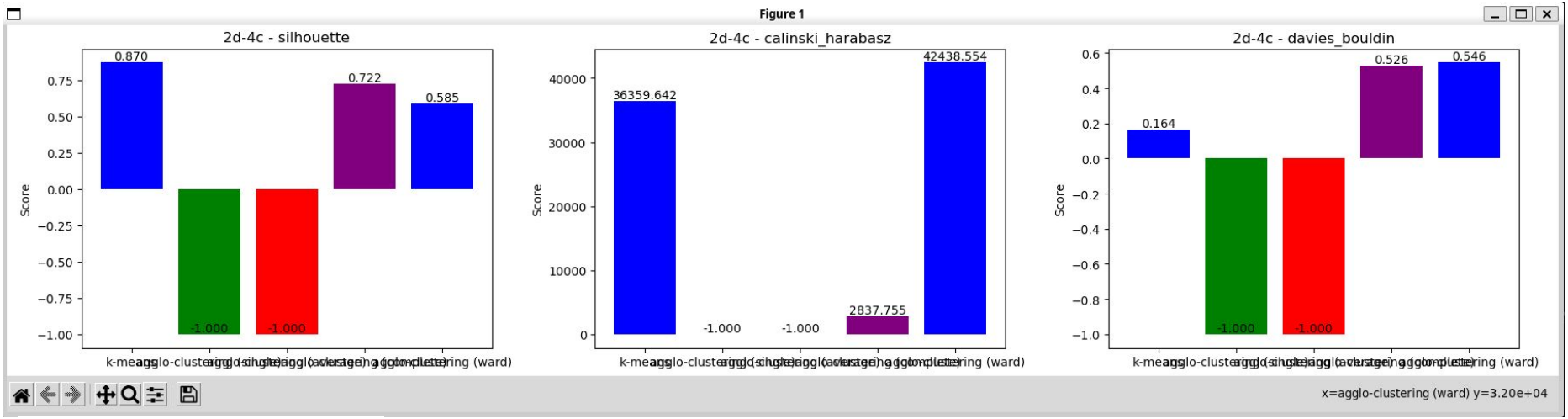
➢ **Ward linkage :** Distance = **augmentation de l'inertie** si on fusionne les 2 clusters

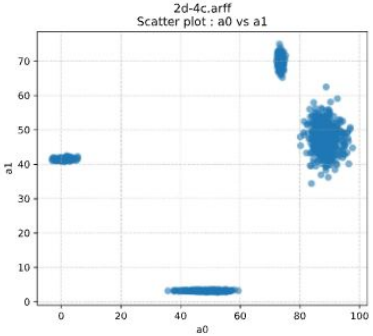
Le Dendrogramme:

L'**arbre** qui visualise les fusions successives des clusters.

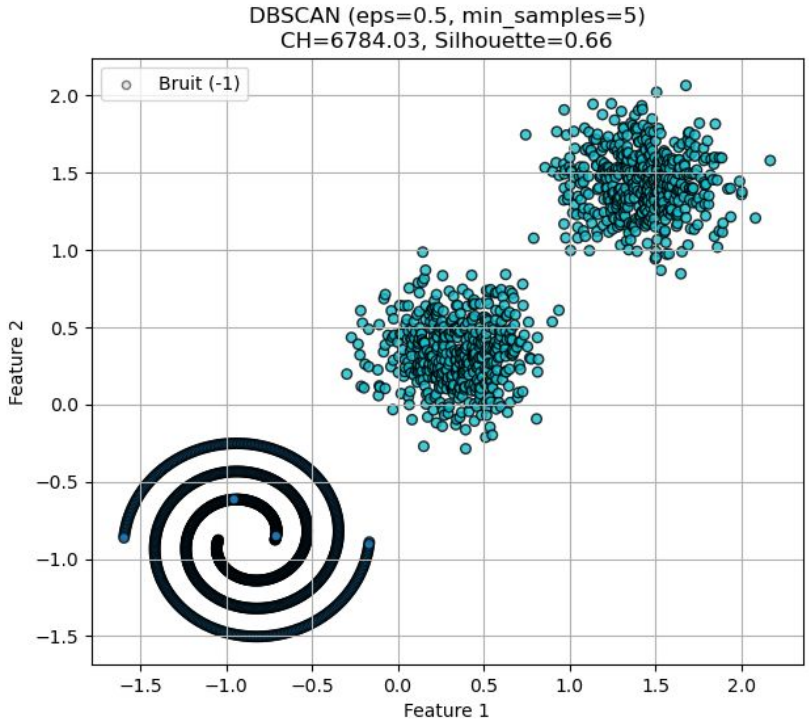


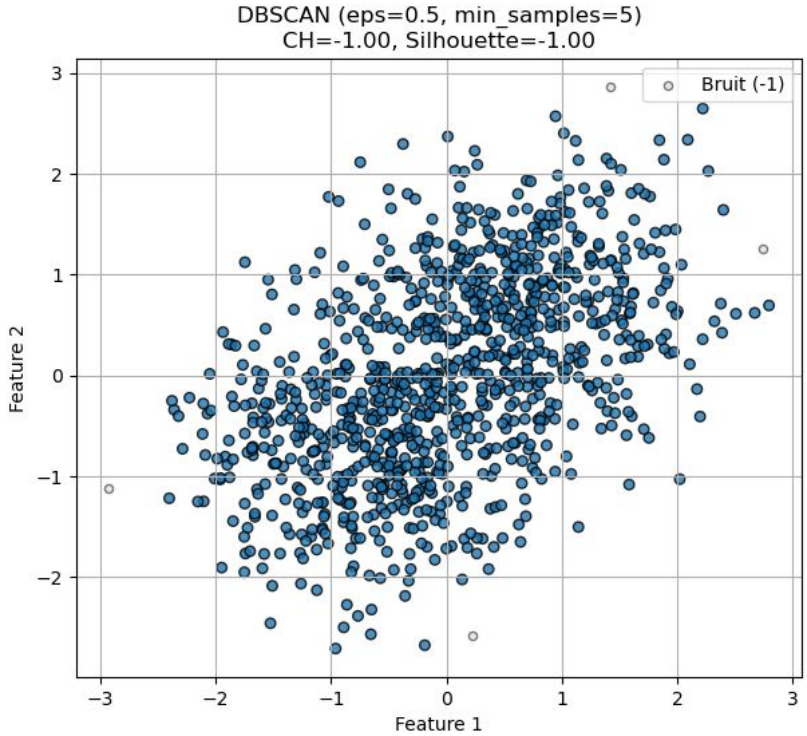
Etude Comparative:



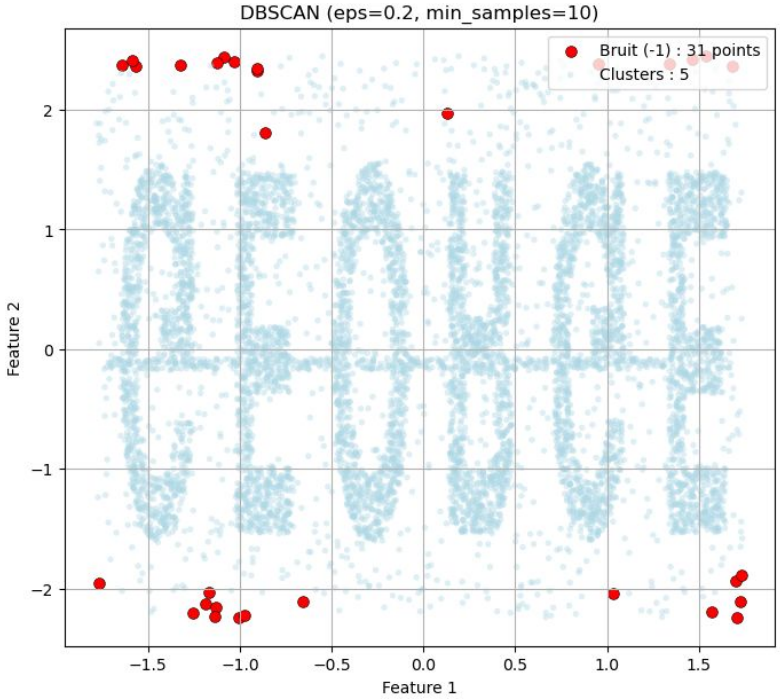


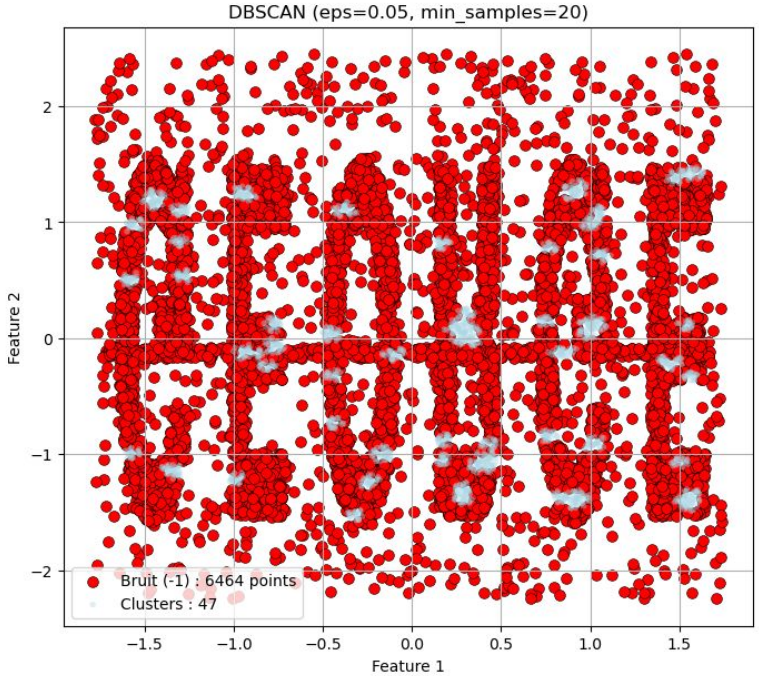
Méthode DBSCAN:





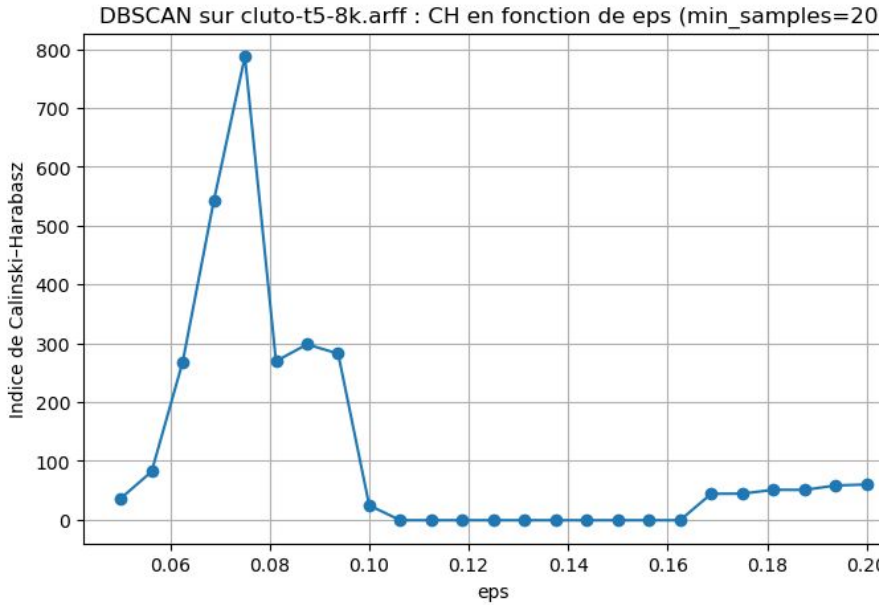
Choix de paramètres:

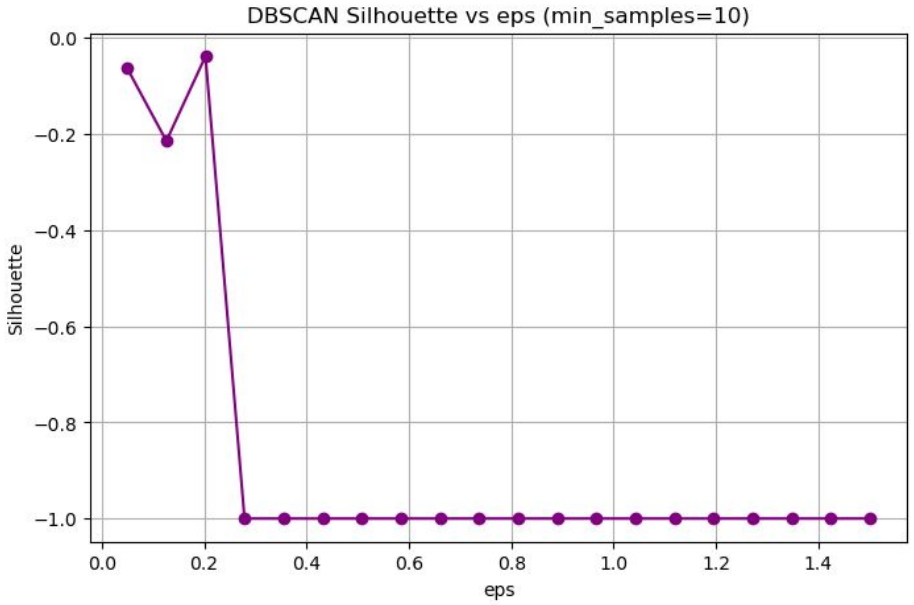




La distance Epsilone:

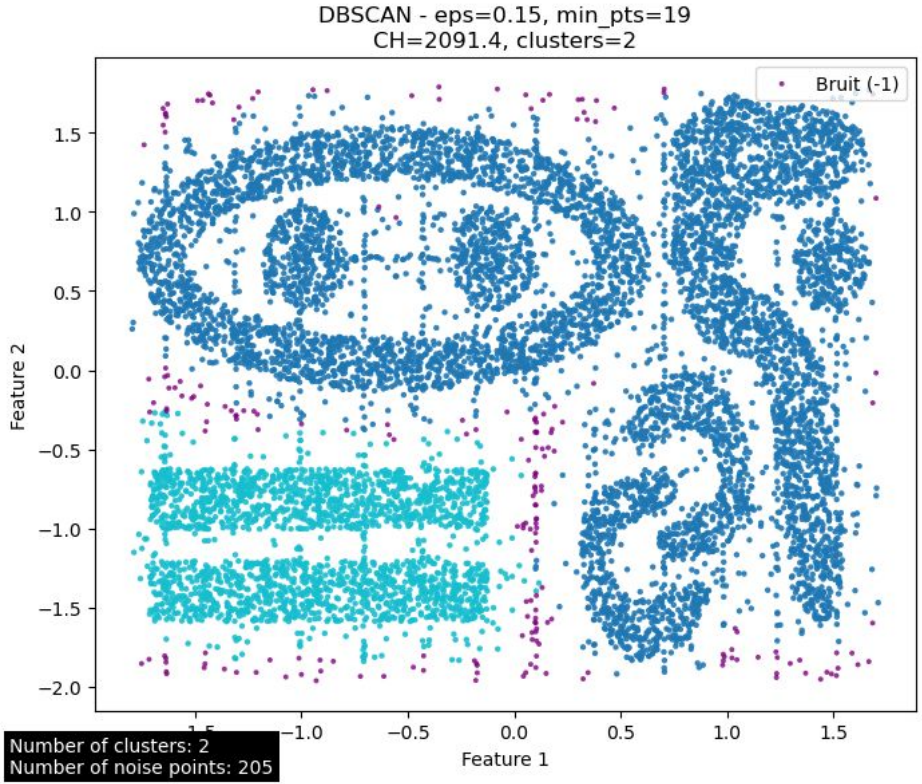
le rayon qui définit la distance maximale pour considérer qu'un point est voisin d'un autre. Plus ε est grand, plus DBSCAN fusionne de régions ensemble et forme des clusters plus larges.

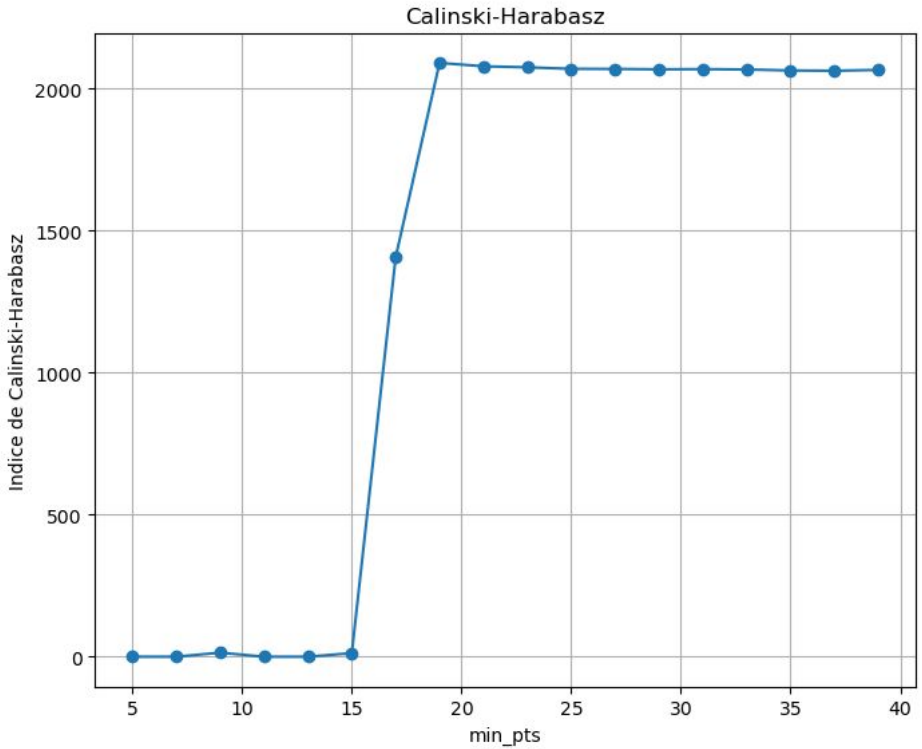




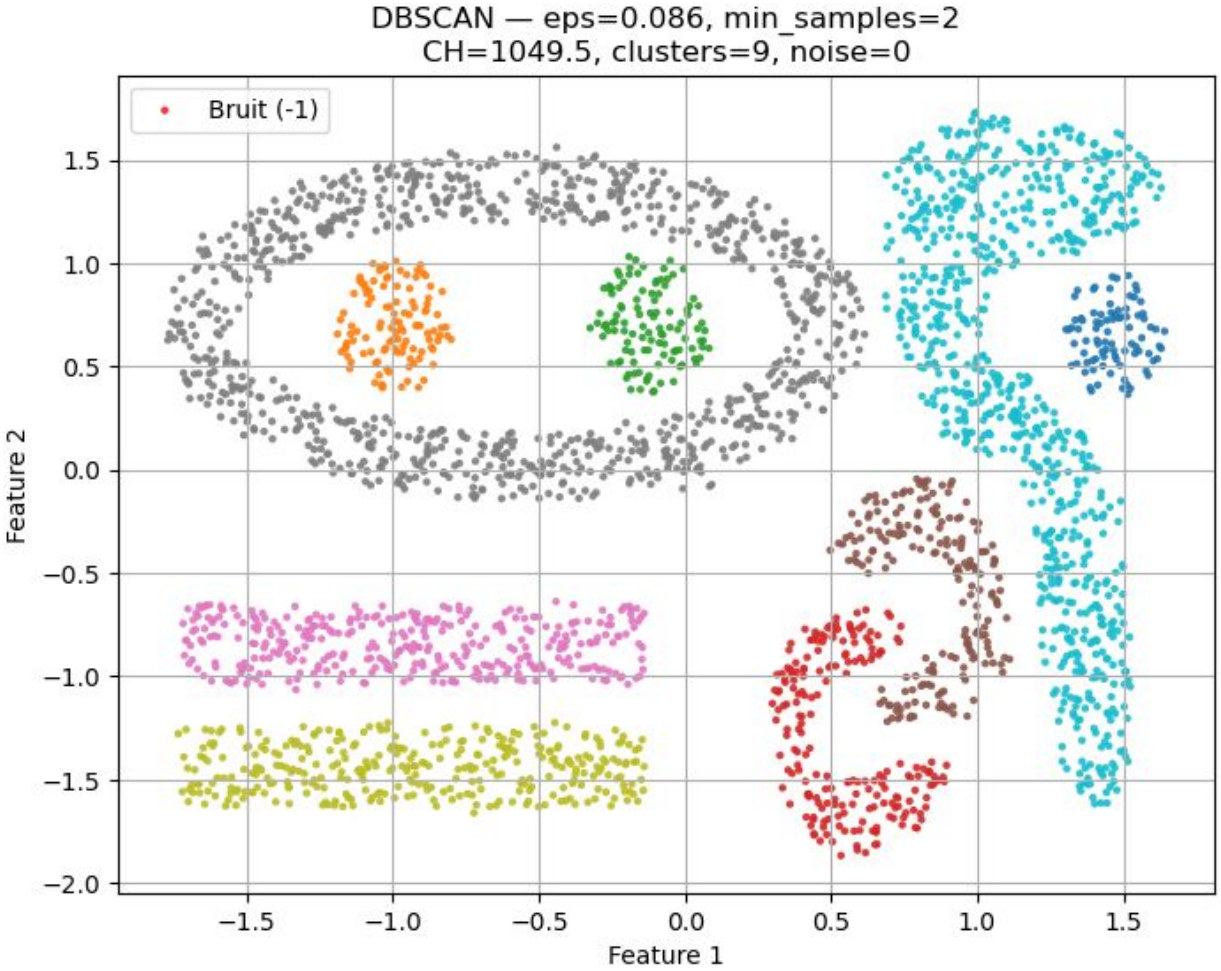
min_samples :

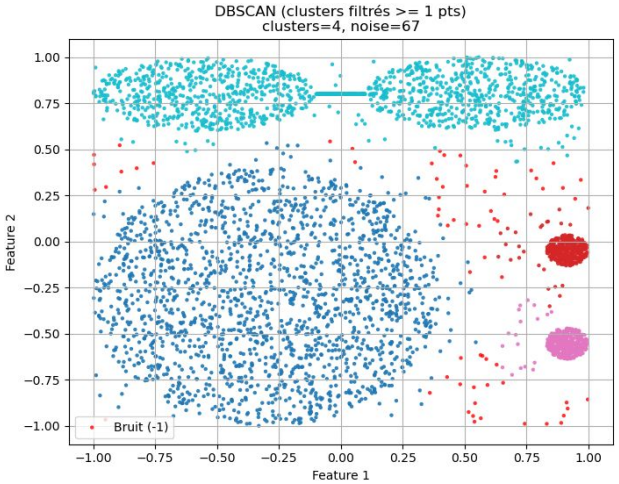
le nombre minimum de points nécessaires dans le voisinage ε pour qu'un point soit considéré comme un "core point". Plus min_samples est grand, plus DBSCAN exige une densité élevée pour former un cluster.

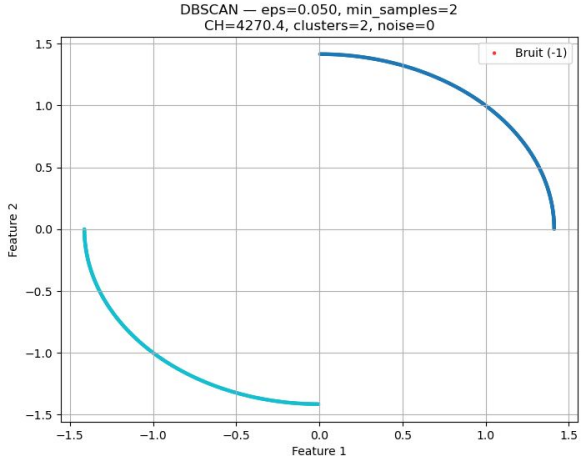


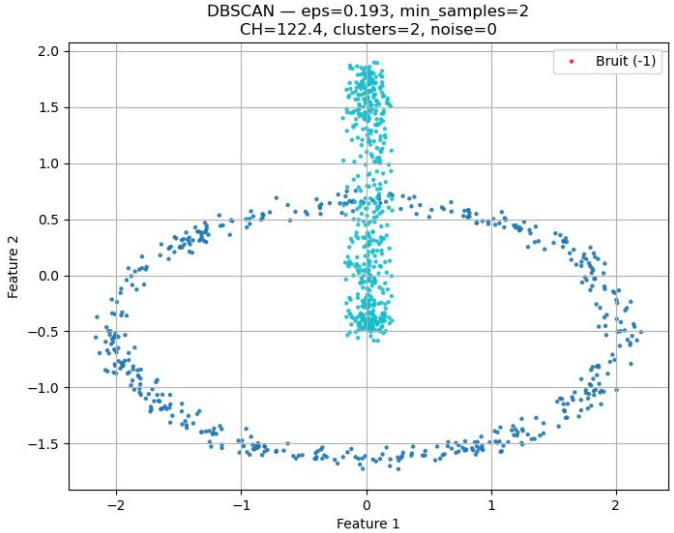


Automatisation des choix de paramètres:

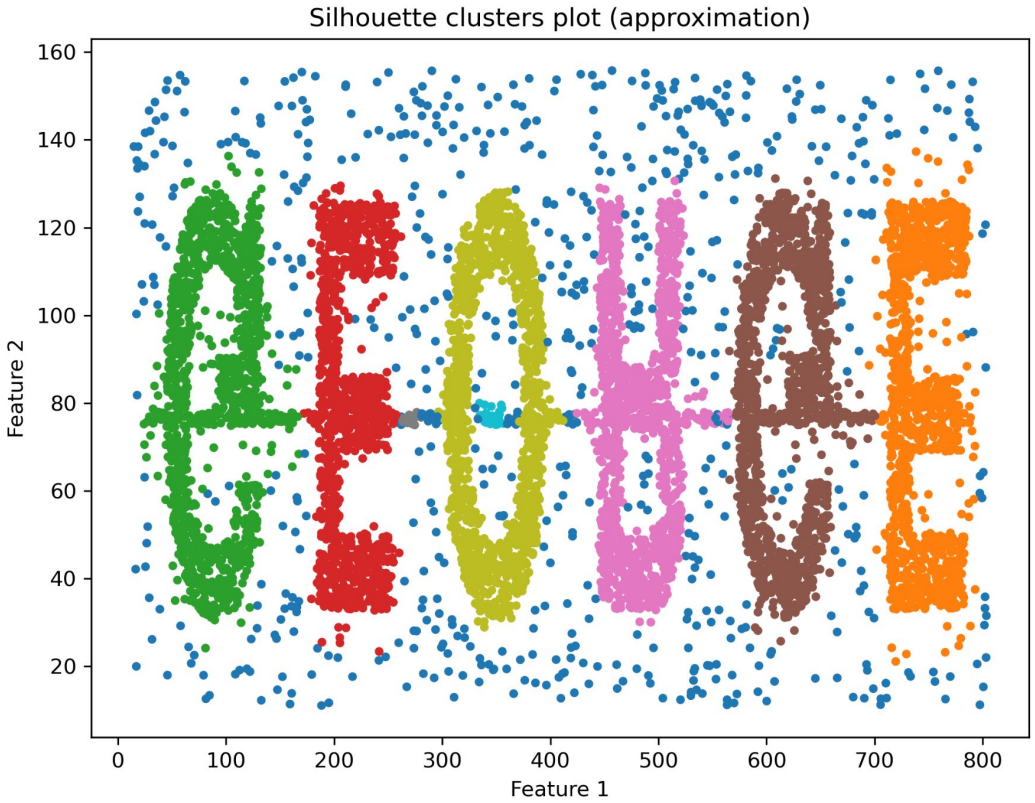



DBSCAN (clusters filtrés >= 1 pts)
clusters=4, noise=67
Bruit (-1)
Feature 1
Feature 2

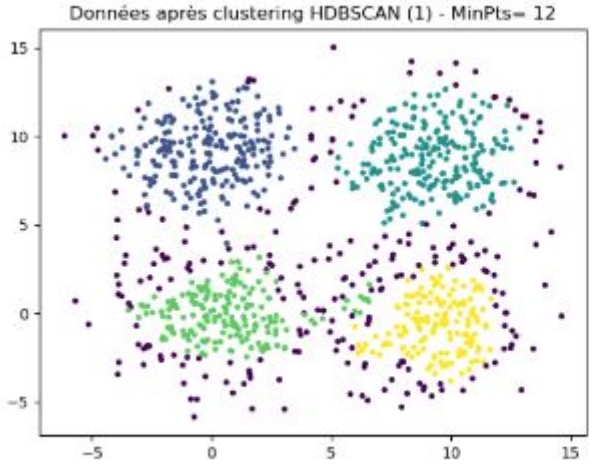

DBSCAN — eps=0.050, min_samples=2
CH=4270.4, clusters=2, noise=0
Bruit (-1)
Feature 1
Feature 2

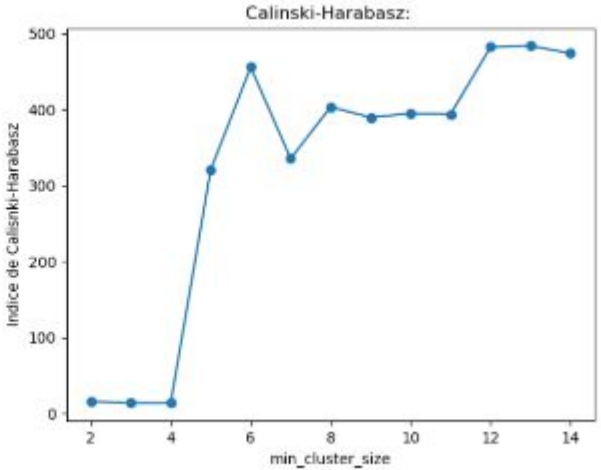

DBSCAN — eps=0.193, min_samples=2
CH=122.4, clusters=2, noise=0
Bruit (-1)
Feature 1
Feature 2

Méthode HDBSCAN:



Paramètres:

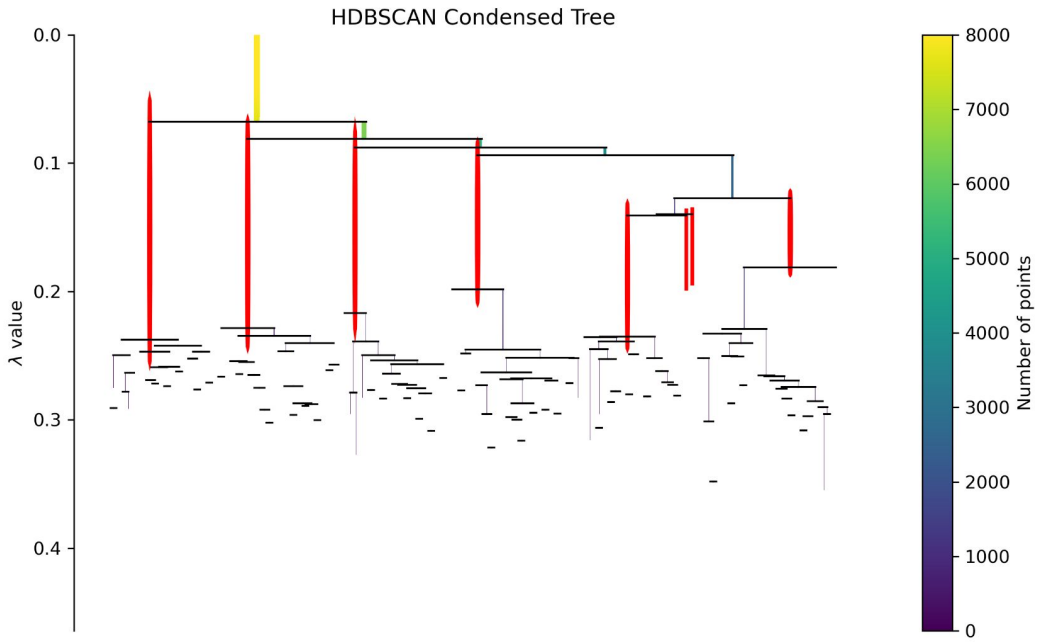




HDBSCAN utilise principalement le paramètre ***min_cluster_size***, qui contrôle la taille minimale des clusters et influence directement leur granularité.

Il utilise aussi ***min_samples***, qui ajuste la sensibilité à la densité : plus il est grand, plus l'algorithme devient conservateur et détecte davantage de bruit.

Condensed Tree de HDBSCAN cluto-t5-8k



Comparaison des Méthodes HDBSCAN et DBSCAN:

Méthode	Davies-Bouldin	Calinski-Harabasz	Silhouette	Temps (ms)	Jeu de Données
Dbscan 6 eps, 8 min pts	1.76	1750	0.44	1750 ms	xclara
Hdbscan 6	1.57	6412	0.65	3702.18 ms	xclara
Dbscan 1 eps, 20 min	1.25	514.57	0.42	599.71 ms	sizes3
Hdbscan 10	1.56	461.86	0.48	447.11 ms	sizes3
Dbscan	0.31	125	0.51	2033.09 ms	long2
Hdbscan 7	1.99	119	0.29	2562.36 ms	long2
Dbscan 1.2 eps pour 8	1.49	381.46	0.47	1006.2 ms	sizes4
Hdbscan 8	1.43	433.62	0.49	2602.85 ms	sizes4

Par rapport à DBSCAN,
Plus robuste à la variation de paramètres
Gère bien les densités variables

• **DBSCAN** est simple à implémenter et fonctionne bien sur des clusters de densité uniforme, mais il est limité lorsque les données ont des structures de densité variable.

• **HDBSCAN** est plus flexible et performant pour gérer des clusters de densités variables, mais il est plus complexe à paramétrer et peut être plus lent.

Code source :

NB : le code implémenté fait voir plusieurs testes fait surtout ceux qui sont dans Rapport.ipynb dans le dossier TP1

projet Globale : https://github.com/Clustering-Laforge-Hassouna/TP-Clustering

partie Clustering:

https://github.com/Clustering-Laforge-Hassouna/TP-Clustering/tree/main/TP1