Master 1 SD

Nom :
Prénom :

Analyse et visualisation de données

Séance de TP 4
Density Based Clustering (DBSCAN)

1- Codage de DBSCAN

a. my_DBSCAN( )

```
def my_DBSCAN(X, eps, minpts, Visualisation=False):
    N, pp = np.shape(X)
    no_cluster = 0
    Dist = np.linalg.norm(X[:, np.newaxis] - X, axis=2)
    # eps = estime_EPS(Dist)
    # minpts = estime_MINPTS(X,Dist,eps)


    Visite = [False] * N
    y = -np.ones(N)  # tableau des labels des données, initialisé bruit (-1)
    Clusters = []

    for p in range(N):
        if not Visite[p]:
            Visite[p] = True
            Voisins = EpsilonVoisinage(p, X, Dist, eps)
            if len(Voisins) >= minpts:
                no_cluster += 1
                cluster = [p]
                y[p] = no_cluster
                cluster, y, Visite = etendre_cluster(X, y, Dist, cluster,
no_cluster, Voisins, Visite, eps, minpts)
                Clusters.append(cluster)

    return y
```

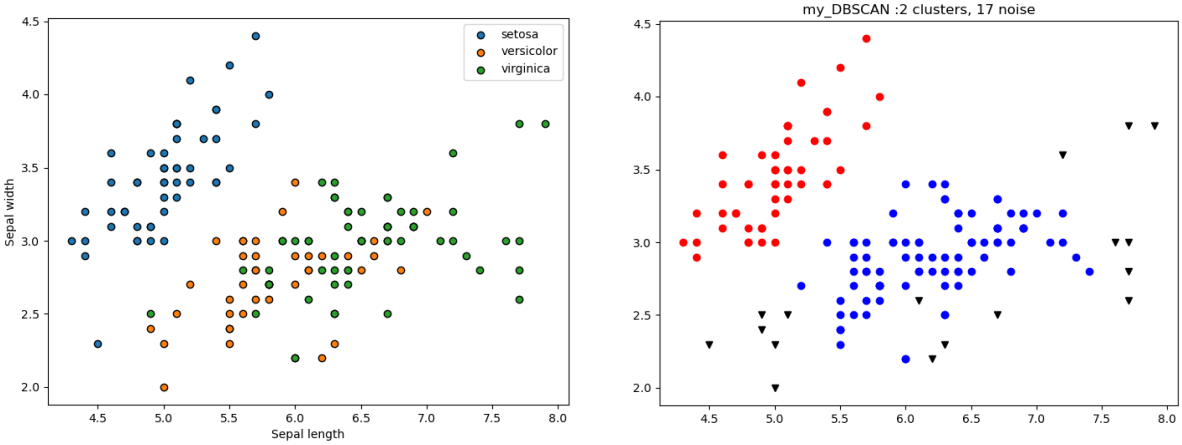
b. etendre_clusters( )

```
def etendre_cluster(X, y, Dist, Cluster, no_cluster, Voisins, Visite, eps,
minpts):
    for v in Voisins:
        if not Visite[v]:
            Visite[v] = True
            if y[v] == -1:
                y[v] = no_cluster
                Cluster.append(v)

            Voisins2 = EpsilonVoisinage(v, X, Dist, eps)
            if len(Voisins2) >= minpts:
                for vv in Voisins2:
                    if vv not in Voisins:
                        Voisins.append(vv)

    return Cluster, y, Visite
```

2- Analyse de DBSCAN

a. *Analyser le résultat de votre programme sur les données IRIS. Combien de Clusters sont déterminés pour les valeurs par défaut des deux hyper-paramètres?*

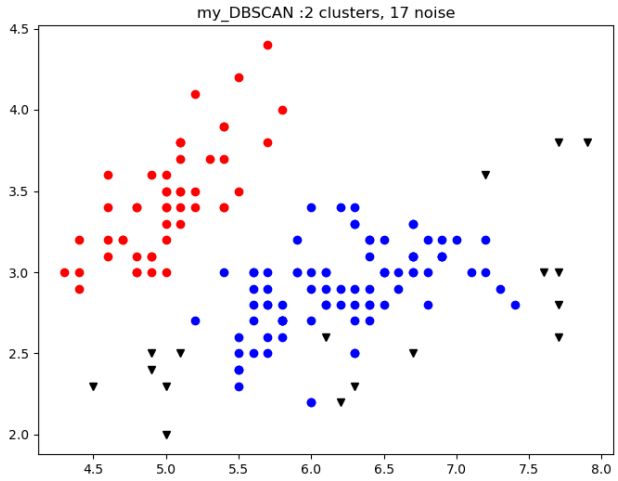
En utilisant les hyperparamètres par défaut (minPts=5 et eps=0,5), 2 clusters ont été détectés, ce qui signifie qu'aucune distinction n'a été trouvée entre les clusters Virginica et Versicolor, car ces clusters se chevauchent. Le cluster Setosa a été correctement identifié.



De plus, 17 points ont été classés comme du bruit, car les frontières des clusters sont des zones de faible densité et le nombre minimum de voisins pour la valeur de eps n'a pas été trouvé.

b. Comparer à la méthode DBSCAN de scikitlearn

La classification des points en clusters réalisée par scikit-learn présente le même nombre de clusters et de points indéterminés, et la distribution dans le plan à deux dimensions semble également être exactement la même. Cela indique que les deux méthodes sont en accord entre elles.

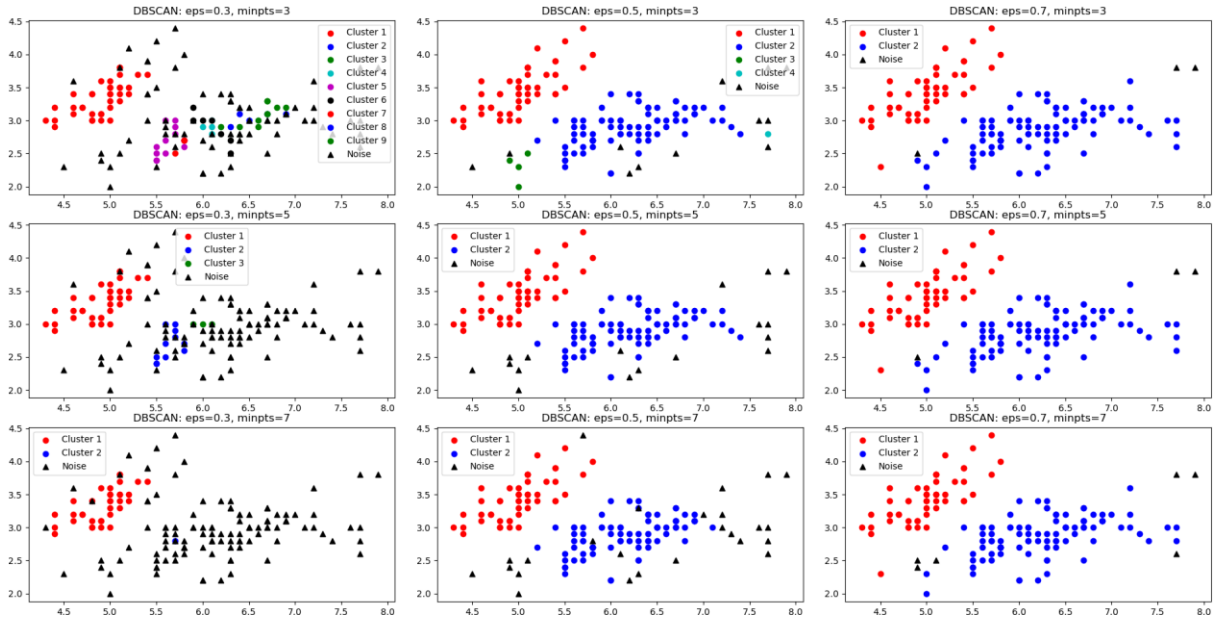


3- Influence des hyperparamètres eps et minpts

a) faire varier ces deux paramètres dans les plages suivantes et noter le nombre de clusters trouvés pour chaque configuration ep in [0.3,0.5,0.7] minpts $\in [3, 5, 7]$

Bruit		eps		
		0,3	0,5	0,7
minpts	3	71	12	3
	5	99	17	3
	7	114	25	6

K		eps		
		0,3	0,5	0,7
minpts	3	9	4	2
	5	3	2	2
	7	2	2	2



b) quel est le paramètre le plus influent ?

Parmi les deux hyperparamètres, eps est celui qui a l'influence prédominante sur la classification des points. Les résultats montrent que plus l'hyperparamètre eps est grand, moins il y a de bruit et moins il y a de clusters. Cela se produit parce qu'un rayon plus grand facilite la satisfaction de la condition minPts, donc l'algorithme continue plus longtemps dans le processus de classification d'un même cluster et parvient à classifier même les points aux extrémités, où la densité est faible.

En revanche, l'augmentation du paramètre minPts a une forte influence sur la quantité de bruit, car toutes les régions n'ont pas une densité suffisamment élevée pour satisfaire des valeurs élevées de minPts. La quantité de clusters diminue avec l'augmentation de minPts, mais de manière moins marquée, car le paramètre eps a une influence plus prédominante. Cependant, une valeur plus élevée de minPts inhibe la propagation des clusters et la création de nouveaux clusters dans des zones isolées de plus grande densité.

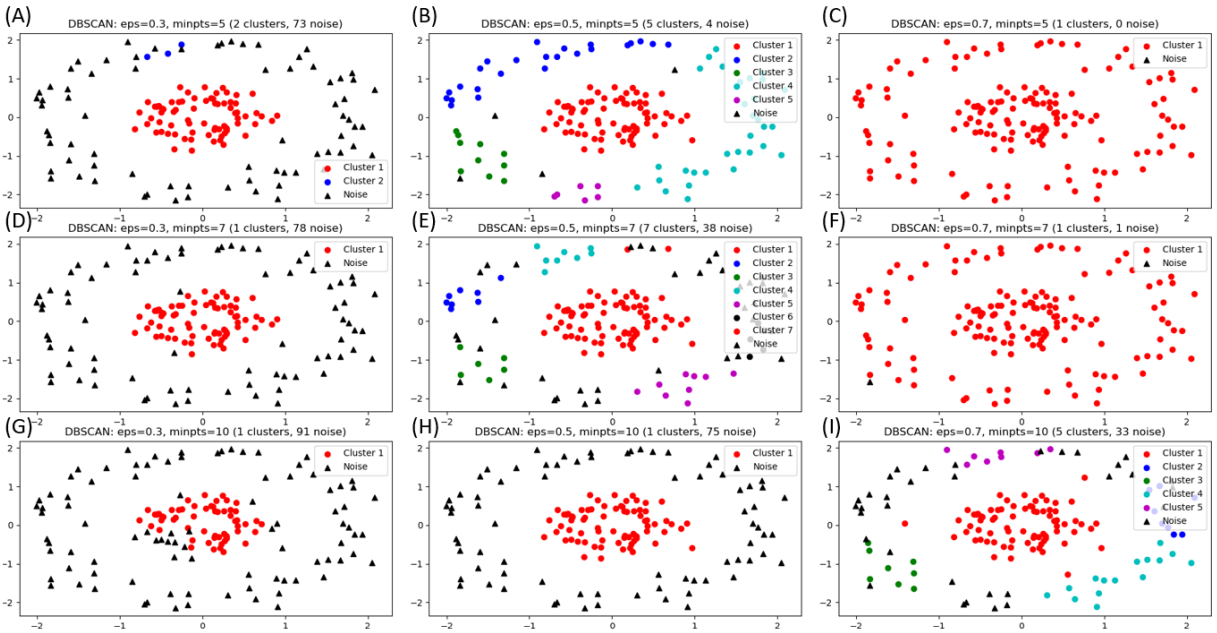
4- Analyse des données circulaires

a. faire varier les deux hyperparamètres dans les plages suivantes et noter le nombre de clusters trouvés pour chaque configuration

$ep \in [0.3, 0.5, 0.7]$ $minpts \in [5, 7, 10]$

Bruit		eps		
		0,3	0,5	0,7
minpts	5	73	4	0
	7	78	38	1
	10	91	75	33

K		eps		
		0,3	0,5	0,7
minpts	5	2	5	0
	7	1	7	1
	10	1	1	5



b. quel est le paramètre le plus influent ?

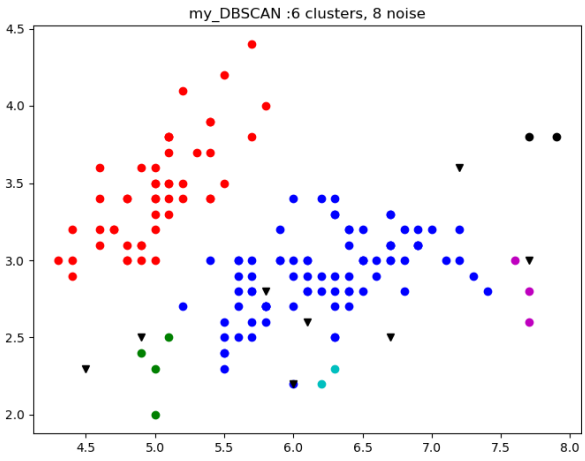
Pour cette distribution de données, on observe que le cercle central possède une densité élevée et, en conséquence, il obtient une classification satisfaisante dans tous les cas. En revanche, le cercle extérieur présente une densité plus faible, ce qui entraîne des problèmes de classification lorsque les valeurs des hyperparamètres ne sont pas bien définies, avec eps continuant à avoir une influence prédominante.

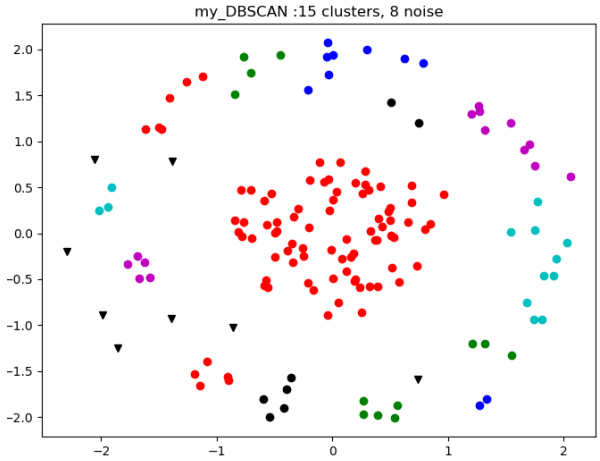
Pour les combinaisons d'hyperparamètres (A, D, G, H), le rayon n'est pas suffisant pour englober un nombre satisfaisant de voisins, et le cercle présente une grande zone d'indécision. Dans les résultats de B, E et I, le cercle extérieur a été divisé en plusieurs clusters qui reflètent les différentes interruptions du parcours de l'algorithme, causées par les zones de faible densité. Enfin, C et F montrent qu'un grand rayon et une faible condition minimale ont permis à l'algorithme de passer d'un cercle à l'autre, créant ainsi un seul cluster.

5- Programmer les méthodes d'estimation des deux hyperparamètres, en respectant les interfaces ci-dessous.

```
def estime_EPS(Dist):
    # estimation du rayon du epsilon voisinage
    N = Dist.shape[0]
    Diag = np.eye(N)*1000
    EPS = np.percentile(np.min(Dist+Diag,axis=0),95)

    return EPS

def estime_MINPTS(X,Dist,eps):
    # estimation de minpts dans le epsilon voisinage
    NVoisins = []
    N,pp =np.shape(X)
    for p in range(N):
        NVoisins = NVoisins+[len(EpsilonVoisinage(p,X,Dist,eps))]
        MINPTS = math.ceil(np.percentile(np.asarray(NVoi-
sins,dtype=np.float64),5))

    return MINPTS
```

6- Tester les deux estimateurs les deux jeux de données. Combiens de clusters sont obtenus ? Que peut-on conclure ?





Les estimateurs des hyperparamètres éliminent les outliers de la distance entre les points et la quantité de voisins afin de définir une valeur qui puisse mieux représenter les données. Pour le jeu de données Iris, eps=0,475 et minPts=2 ont été calculés, tandis que pour les cercles générés, nous avons eps=0,333 et minPts=2.

Les deux combinaisons de paramètres se sont révélées efficaces pour réduire les zones d'indécision, et très peu de points ont été classés comme du bruit. Cependant, la définition des clusters n'a été satisfaisante dans aucun des cas. Dans le jeu de données Iris, des zones de faible densité à la périphérie des données ont été classées comme de petits clusters distincts. Nous pouvons affirmer que DBSCAN n'est pas une bonne technique pour séparer correctement les clusters du jeu de données Iris, indépendamment des hyperparamètres, en raison du chevauchement entre Virginica et Versicolor.

Dans l'autre jeu de données, la grande différence de densité entre les deux cercles rend impossible l'existence d'une seule valeur d'eps et de minPts idéale pour les deux ensembles. Les paramètres estimés ne conviennent pas bien au cercle extérieur, ce qui entraîne sa division en plusieurs clusters.

7- Quels sont les problèmes non résolus par DBSCAN?

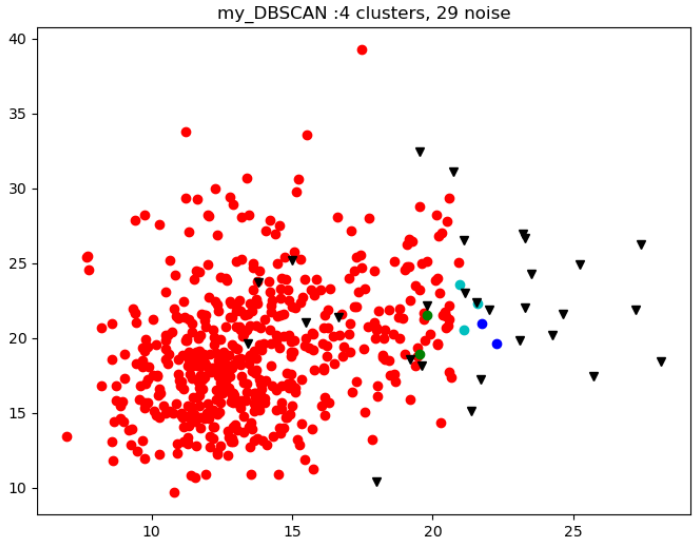
Le succès de l'algorithme dépend fortement de la bonne sélection des paramètres eps et minPts. Si ces paramètres ne sont pas choisis correctement, DBSCAN peut générer des clusters trop petits (comme on le voit dans les images B, E et I), ou au contraire

fusionner des clusters qui ne devraient pas être fusionnés (comme c'est le cas dans C et F).

De plus, DBSCAN fonctionne bien lorsque les clusters sont séparés par des régions de faible densité, mais il échoue lorsque les clusters ont des densités très différentes, comme les deux cercles générés par la fonction make_circles(). L'algorithme ne peut pas s'adapter à plusieurs densités, car les paramètres eps et minPts sont fixes.

Enfin, DBSCAN est incapable de différencier deux clusters qui se chevauchent, comme Virginica et Versicolor dans le jeu de données Iris, car il n'y a pas de différence de densité entre ces deux clusters pour créer une interruption dans le cheminement de l'algorithme.

Un autre exemple d'échec dans la détermination des clusters est l'ensemble de données sur le cancer du sein, où même les hyperparamètres estimés (eps=85.35 et minpts=2) ne sont pas capables de distinguer les clusters qui se mélangent.



your_DBSCAN.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Oct  8 09:38:57 2022

@author: Thierry Paquet
"""

#!/usr/bin/env python
# -*- coding: utf-8 -*-


import numpy as np
from sklearn import datasets
from sklearn.cluster import DBSCAN
from numpy.linalg import norm
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from sklearn.preprocessing import StandardScaler
import sklearn.cluster
import math

colors =['k','r','b','g','c','m',]
n_colors = 6

#############################################################################
def EpsilonVoisinage(i,X,Dist,eps):
    N,p =np.shape(X)
    #Voisins = [v for v in range(N) if ( i != v and Dist[v,i] < eps)]
    Voisins = [v for v in range(N) if ( Dist[v,i] < eps)] # Devemos considerar o proprio ponto no calcul
    return Voisins

#############################################################################
def etendre_cluster(X, y, Dist, Cluster, no_cluster, Voisins, Visite, eps, minpts):

    for v in Voisins:
        if not Visite[v]:
            Visite[v] = True
            if y[v] == -1:
                y[v] = no_cluster
                Cluster = Cluster + [v]

            Voisins2 = EpsilonVoisinage(v, X, Dist, eps )
            if len(Voisins2) >= minpts:
                for vv in Voisins2:
                    if vv not in Voisins:
                        Voisins.append(vv)

    return Cluster, y, Visite

#############################################################################
```

```
52 def estime_EPS(Dist):
53     # estimation du rayon du epsilon voisinage
54     N = Dist.shape[0]
55     Diag = np.eye(N)*1000
56     EPS = np.percentile(np.min(Dist+Diag,axis=0),95)
57 
58     return EPS
59 
60 def estime_MINPTS(X,Dist,eps):
61     # estimation de minpts dans le epsilon voisinage
62     NVoisins = []
63     N,pp =np.shape(X)
64     for p in range(N):
65         NVoisins = NVoisins+[len(EpsilonVoisinage(p,X,Dist,eps))]
66         MINPTS = math.ceil(np.percentile(np.asarray(NVoisins,dtype=np.float64),5))
67 
68     return MINPTS
69 
70 ###########################################################################
71 #              MY DBSCAN
72 def my_DBSCAN(X, eps, minpts, Visualisation = False):
73     N,pp =np.shape(X)
74     no_cluster = 0
75 
76     # on pré-calcule toutes les distances entre points
77     Dist = np.reshape(norm(X - X[0,:],axis=1),(N,1))
78     for n in range(1,N):
79         D = np.reshape(norm(X - X[n,:],axis=1),(N,1))
80         Dist = np.concatenate((Dist,D),axis=1)
81 
82     eps = estime_EPS(Dist)
83     print("eps = ", eps)
84     minpts = estime_MINPTS(X,Dist,eps)
85     print("minpts = ", minpts)
86 
87     Visite = [False for _ in range(N)]
88 
89     y = - np.ones(N)  # tableau des labels des données, initialisé bruit (-1)
90     Clusters = []
91 
92     for p in range(N):
93         if not Visite[p]:
94             Visite[p] = True
95             Voisins = EpsilonVoisinage(p, X, Dist, eps)
96             if len(Voisins) >= minpts:
97                 no_cluster = no_cluster+1
98                 cluster = [p]
99                 y[p] = no_cluster
100                cluster, y, Visite = etendre_cluster(X, y, Dist, cluster, no_cluster,
    Voisins, Visite, eps, minpts)
101                Clusters.append(cluster)
102            # else:
103            #     continue # p est du bruit
104
```

```
105     if Visualisation :
106         print(len(Clusters),' clusters trouvés', no_cluster)
107         print("Clusters =",Clusters)
108         for cluster in Clusters:
109             print('effectif cluster ',len(cluster))
110 
111         Bruit = [n for n in range(N) if y[n] == -1]
112         print('effectif  bruit',len(Bruit))
113 
114     return y
115 
116 if __name__ == '__main__':
117 
118 ############################################################
119 
120     ########### DATASE IRIS ############################
121     iris = datasets.load_iris()
122     X = iris.data
123     y = iris.target
124 
125     fig = plt.figure(2, figsize=(8, 6))
126     plt.clf()
127     plt.scatter(X[0:50, 0], X[0:50, 1],
    cmap=plt.cm.Set1,edgecolor='k',label=iris.target_names[0])
128     plt.scatter(X[50:100, 0], X[50:100, 1],
    cmap=plt.cm.Set1,edgecolor='k',label=iris.target_names[1])
129     plt.scatter(X[100:150, 0], X[100:150, 1],
    cmap=plt.cm.Set1,edgecolor='k',label=iris.target_names[2])
130     plt.xlabel('Sepal length')
131     plt.ylabel('Sepal width')
132     plt.legend(scatterpoints=1)
133 
134     eps = 0.7
135     minpts = 5
136 
137     my_y = my_DBSCAN(X,eps,minpts)
138     statistiques = np.unique(my_y,return_counts=True)
139     K = len(statistiques[0])-(1 if -1 in statistiques[0] else 0)
140     Bruit = [p for p in range(len(my_y)) if my_y[p]==-1]
141 
142     fig = plt.figure(figsize=(8, 6))
143 
144     for k in range(1,K+1):
145         plt.plot(X[my_y==k, 0], X[my_y==k, 1], colors[k%n_colors]+'o')
146     plt.plot(X[my_y==-1, 0], X[my_y==-1, 1], 'kv')
147 
148     plt.title('my_DBSCAN :'+str(K)+' clusters, '+str(len(Bruit))+' noise')
149     plt.show()
150 
151 
152     ############ comparaison avec DBSCAN de scikit learn ############################
153     yy = DBSCAN(eps=eps,min_samples=minpts).fit_predict(X)
154     statistiques = np.unique(yy,return_counts=True)
155     Noise = [p for p in range(len(yy)) if yy[p]==-1]
```

```
    K = len(statistiques[0])-(1 if -1 in statistiques[0] else 0)

    print('yy len:',len(yy))
    fig = plt.figure(figsize=(8, 6))
    for k in range(K):
        plt.plot(X[yy==k, 0], X[yy==k, 1], colors[(k+1)%n_colors]+'o')
    plt.plot(X[yy==-1, 0], X[yy==-1, 1], 'kv')

    plt.title('scikitlearn DBSCAN :'+str(K)+' clusters, '+str(len(Noise))+' noise')
    plt.show()


    ############ CIRCLES GENERES ########################################
    X, y = make_circles(n_samples=150, factor=0.3, noise=0.1)
    X = StandardScaler().fit_transform(X)

    eps = 0.7
    minpts = 7

    my_y = my_DBSCAN(X,eps,minpts)
    statistiques = np.unique(my_y,return_counts=True)
    K = len(statistiques[0])-(1 if -1 in statistiques[0] else 0)
    Bruit = [p for p in range(len(my_y)) if my_y[p]==-1]

    fig = plt.figure(figsize=(8, 6))

    for k in range(1,K+1):
        plt.plot(X[my_y==k, 0], X[my_y==k, 1], colors[k%n_colors]+'o')
    plt.plot(X[my_y==-1, 0], X[my_y==-1, 1], 'kv')

    plt.title('my_DBSCAN :'+str(K)+' clusters, '+str(len(Bruit))+' noise')
    plt.show()


    ############ BREAST CANCER ############################
    breast = datasets.load_breast_cancer()
    X = breast.data
    y = breast.target

    eps = 0.5
    minpts = 5

    my_y = my_DBSCAN(X,eps,minpts)
    statistiques = np.unique(my_y,return_counts=True)
    K = len(statistiques[0])-(1 if -1 in statistiques[0] else 0)
    Bruit = [p for p in range(len(my_y)) if my_y[p]==-1]

    fig = plt.figure(figsize=(8, 6))

    for k in range(1,K+1):
        plt.plot(X[my_y==k, 0], X[my_y==k, 1], colors[k%n_colors]+'o')
    plt.plot(X[my_y==-1, 0], X[my_y==-1, 1], 'kv')

    plt.title('my_DBSCAN :'+str(K)+' clusters, '+str(len(Bruit))+' noise')
```

```
210     plt.show()
```