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()
```