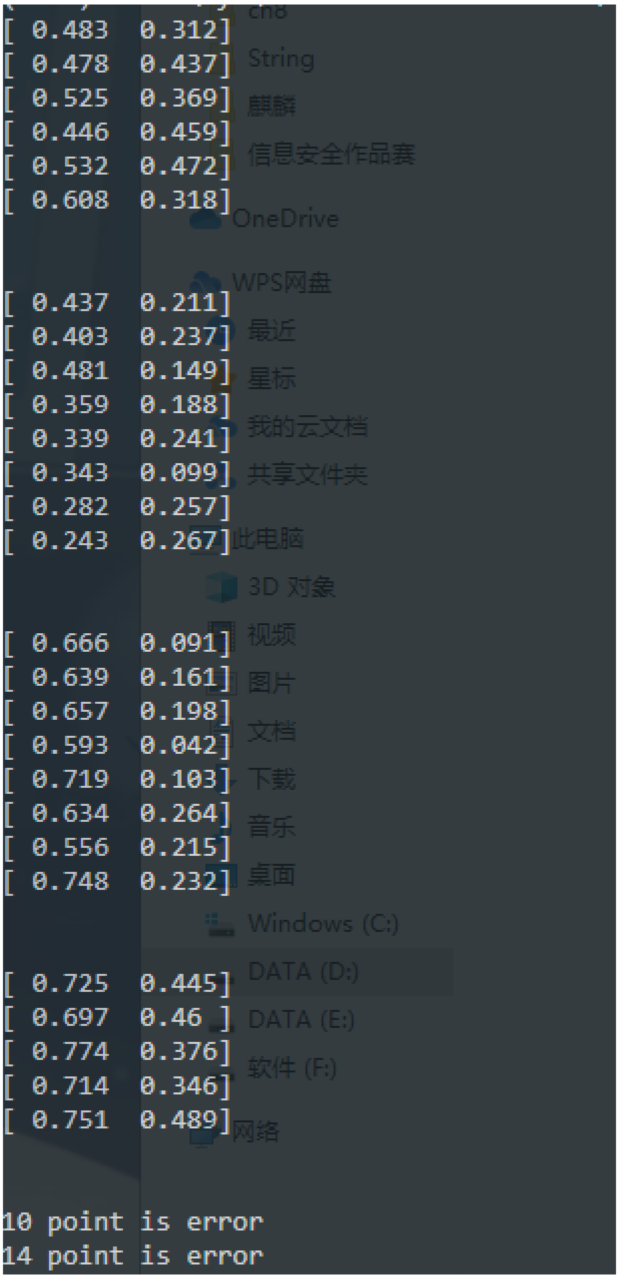
DBSCAN

```python
import numpy as np

Train_data = np.array([
    [0.697, 0.460], [0.774, 0.376], [0.634, 0.264], [0.608, 0.318], [0.556, 0.215],
    [0.403, 0.237], [0.481, 0.149], [0.437, 0.211], [0.666, 0.091], [0.243, 0.267],
    [0.245, 0.057], [0.343, 0.099], [0.639, 0.161], [0.657, 0.198], [0.360, 0.370],
    [0.593, 0.042], [0.719, 0.103], [0.359, 0.188], [0.339, 0.241], [0.282, 0.257],
    [0.748, 0.232], [0.714, 0.346], [0.483, 0.312], [0.478, 0.437], [0.525, 0.369],
    [0.751, 0.489], [0.532, 0.472], [0.473, 0.376], [0.725, 0.445], [0.446, 0.459]])
Train_size = Train_data.shape[0]



epsilon = 0.11
MinPts = 5
used = [False for i in range(Train_size)]
Kernel = []
isKernel = [False for i in range(Train_size)]

def distCmp(x, y):
    return np.linalg.norm(x - y) <= epsilon

def Func1(*args):
    global cnt
    cnt += 1

def Func2(x):
    # print(args)
    if not used[x]:
        if isKernel[x]:
            Queue.append(x)
            Kernel.remove(x)
        C[-1].append(Train_data[x])
        used[x] = True

def Judge(i, Func):
    for j in range(Train_size):
        if distCmp(Train_data[i], Train_data[j]):
            Func(j)

for i in range(Train_size):
    cnt = 0
    Judge(i, Func1)
    if cnt >= MinPts:
        Kernel.append(i)
        isKernel[i] = True
C = []
while Kernel:
    Start = np.random.choice(Kernel, 1)[0]
    used[Start] = True
    Kernel.remove(Start)
    Queue = [Start]
    C.append([])
    C[-1].append(Train_data[Start])
    while Queue:
        now = Queue[0]
        Queue.pop(0)
        Judge(now, Func2)

for x in C:
    for y in x:
        print(y)
    print('\n')

for idx in range(Train_size):
    if not used[idx]:
        print('%d point is error' % idx)
```

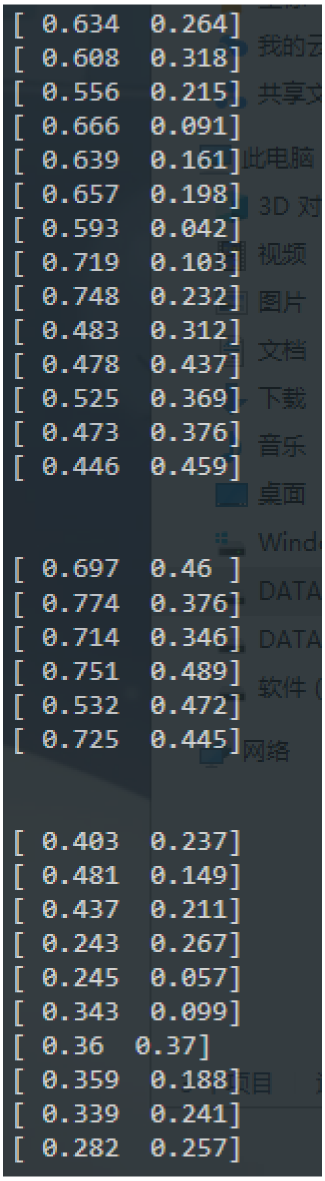
运行结果



```
[ 0.483  0.312]
[ 0.478  0.437]
[ 0.525  0.369]
[ 0.446  0.459]
[ 0.532  0.472]
[ 0.608  0.318]

[ 0.437  0.211]
[ 0.403  0.237]
[ 0.481  0.149]
[ 0.359  0.188]
[ 0.339  0.241]
[ 0.343  0.099]
[ 0.282  0.257]
[ 0.243  0.267]

[ 0.666  0.091]
[ 0.639  0.161]
[ 0.657  0.198]
[ 0.593  0.042]
[ 0.719  0.103]
[ 0.634  0.264]
[ 0.556  0.215]
[ 0.748  0.232]

[ 0.725  0.445]
[ 0.697  0.46 ]
[ 0.774  0.376]
[ 0.714  0.346]
[ 0.751  0.489]


10 point is error
14 point is error
```

kmeans

```python
import numpy as np

k = 3

def rand_row(dataset, size):
    n_size = dataset.shape[0]
    row = np.random.choice(n_size, size, replace=False)
    TrainMatrix = dataset[row,:]
    return TrainMatrix



Train_data = np.array([
    [0.697, 0.460], [0.774, 0.376], [0.634, 0.264], [0.608, 0.318], [0.556, 0.215],
    [0.403, 0.237], [0.481, 0.149], [0.437, 0.211], [0.666, 0.091], [0.243, 0.267],
    [0.245, 0.057], [0.343, 0.099], [0.639, 0.161], [0.657, 0.198], [0.360, 0.370],
    [0.593, 0.042], [0.719, 0.103], [0.359, 0.188], [0.339, 0.241], [0.282, 0.257],
    [0.748, 0.232], [0.714, 0.346], [0.483, 0.312], [0.478, 0.437], [0.525, 0.369],
    [0.751, 0.489], [0.532, 0.472], [0.473, 0.376], [0.725, 0.445], [0.446, 0.459]])

Train_Size = Train_data.shape[0]
mu = rand_row(Train_data, k)
belong = np.zeros(Train_Size, dtype=int)
size = np.zeros(k, dtype=int)

while True:
    flag = False
    size = np.zeros(k)
    for i in range(Train_Size):
        Last_Belong = belong[i]
        minDis = 0x7fffffff
        for j in range(k):
            dis = np.linalg.norm(Train_data[i] - mu[j])
            if dis < minDis:
                belong[i] = j
                minDis = dis
        if belong[i] != Last_Belong:
            flag = True
        size[belong[i]] += 1
    if not flag:
        break
    mu = [0.0, 0.0] * k
    for i in range(Train_Size):
        mu[belong[i]] += Train_data[i] / size[belong[i]]

C = [[] for i in range(k)]
for i in range(Train_Size):
    C[belong[i]].append(Train_data[i])

for i in range(k):
    for x in C[i]:
        print(x)
    print('\n')
```



```
[ 0.634  0.264]
[ 0.608  0.318]
[ 0.556  0.215]
[ 0.666  0.091]
[ 0.639  0.161]
[ 0.657  0.198]
[ 0.593  0.042]
[ 0.719  0.103]
[ 0.748  0.232]
[ 0.483  0.312]
[ 0.478  0.437]
[ 0.525  0.369]
[ 0.473  0.376]
[ 0.446  0.459]


[ 0.697  0.46 ]
[ 0.774  0.376]
[ 0.714  0.346]
[ 0.751  0.489]
[ 0.532  0.472]
[ 0.725  0.445]


[ 0.403  0.237]
[ 0.481  0.149]
[ 0.437  0.211]
[ 0.243  0.267]
[ 0.245  0.057]
[ 0.343  0.099]
[ 0.36  0.37]
[ 0.359  0.188]
[ 0.339  0.241]
[ 0.282  0.257]
```

kmeans可以确认分为3个聚类, 而DBSCAN不可, DBSCAN可以辨别出异常点, 而kmeans不行