11/17/2020 DBSCAN.py

```
1 from hw4code.KMeans import KMeans,compute_purity,compute_NMI,getEuclideanDist
2 from hw4code.DataPoints import DataPoints
3 import random
4
5
6 class DBSCAN:
7
      # -----
      def __init__(self):
8
         self.e = 0.0
9
         self.minPts = 3
10
         self.noOfLabels = 0
11
12
      # ------
      def main(self, dataname):
13
14
         seed = 71
15
         self.dataname = dataname[5:-4]
16
         print("\nFor " + self.dataname)
17
         self.dataSet = KMeans.readDataSet(dataname)
18
19
         random.Random(seed).shuffle(self.dataSet)
         self.noOfLabels = DataPoints.getNoOFLabels(self.dataSet)
20
         self.e = self.getEpsilon(self.dataSet)
21
22
         print("Esp :" + str(self.e))
23
         self.dbscan(self.dataSet)
24
25
      26
27
      def getEpsilon(self, dataSet):
28
         distances = []
         sumOfDist = 0.0
29
30
         for i in range(len(dataSet)):
31
             point = dataSet[i]
             for j in range(len(dataSet)):
32
                if i == j:
33
34
                    continue
35
                pt = dataSet[j]
                dist = getEuclideanDist(point.x, point.y, pt.x, pt.y)
36
37
                distances.append(dist)
38
39
             distances.sort()
             sumOfDist += distances[7]
40
             distances = []
41
42
         return sumOfDist/len(dataSet)
43
      # ------
44
      def dbscan(self, dataSet):
45
         clusters = []
         visited = set()
46
47
         noise = set()
48
         # Iterate over data points
49
50
         for i in range(len(dataSet)):
             point = dataSet[i]
51
             if point in visited:
52
53
                continue
54
             visited.add(point)
55
             N = []
56
             minPtsNeighbours = 0
57
             # check which point satisfies minPts condition
58
59
             for j in range(len(dataSet)):
                if i==j:
```

```
11/17/2020
                                              DBSCAN.py
                        continue
 61
 62
                    pt = dataSet[j]
                    dist = getEuclideanDist(point.x, point.y, pt.x, pt.y)
 63
 64
                    if dist <= self.e:</pre>
 65
                        minPtsNeighbours += 1
 66
                        N.append(pt)
 67
                if minPtsNeighbours >= self.minPts:
 68
 69
                    cluster = set()
 70
                    cluster.add(point)
                    point.isAssignedToCluster = True
 71
 72
 73
                    j = 0
 74
                    while j < len(N):
 75
                        point1 = N[j]
 76
                        minPtsNeighbours1 = 0
 77
                        N1 = []
 78
                        if not point1 in visited:
 79
                           visited.add(point1)
 80
                            for 1 in range(len(dataSet)):
 81
                               pt = dataSet[1]
                               dist = getEuclideanDist(point1.x, point1.y, pt.x, pt.y)
 82
                               if dist <= self.e:</pre>
 83
 84
                                   minPtsNeighbours1 += 1
 85
                                   N1.append(pt)
 86
                           if minPtsNeighbours1 >= self.minPts:
 87
                               self.removeDuplicates(N, N1)
 88
 89
                        # Add point1 is not yet member of any other cluster then add it
    to cluster
 90
                        # Hint: use self.isAssignedToCluster function to check if a point
    is assigned to any clusters
                        # ========#
 91
 92
                        # STRART YOUR CODE HERE #
 93
                        # ========#
 94
                        if not point1.isAssignedToCluster:
 95
                           cluster.add(point1)
 96
                           point1.isAssignedToCluster = True
 97
                        # =======#
 98
                          END YOUR CODE HERE
 99
                        100
                        j += 1
101
102
                    # add cluster to the list of clusters
                    clusters.append(cluster)
103
104
105
                else:
106
                    noise.add(point)
107
108
            # List clusters
109
110
            print("Number of clusters formed :" + str(len(clusters)))
            print("Noise points :" + str(len(noise)))
111
112
            # Calculate purity
113
            compute_purity(clusters,len(self.dataSet))
114
115
            compute_NMI(clusters, self.noOfLabels)
            DataPoints.writeToFile(noise, clusters, "DBSCAN_"+ self.dataname + ".csv")
116
        # -----
117
118
        def removeDuplicates(self, n, n1):
```

localhost:4649/?mode=python 2/3

11/17/2020 DBSCAN.py for point in n1: 119 isDup = False 120 121 for point1 in n: if point1 == point: 122 123 isDup = True break 124 if not isDup: 125 126 n.append(point) 127

128