

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

```

61         continue
62     pt = dataSet[j]
63     dist = getEuclideanDist(point.x, point.y, pt.x, pt.y)
64     if dist <= self.e:
65         minPtsNeighbours += 1
66         N.append(pt)
67
68     if minPtsNeighbours >= self.minPts:
69         cluster = set()
70         cluster.add(point)
71         point.isAssignedToCluster = True
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 l in range(len(dataSet)):
81                     pt = dataSet[l]
82                     dist = getEuclideanDist(point1.x, point1.y, pt.x,
pt.y)
83                     if dist <= self.e:
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                 noise.discard(point1)
97             # =====#
98             # END YOUR CODE HERE #
99             # =====#
100             j += 1
101
102             # add cluster to the list of clusters
103             clusters.append(cluster)
104
105         else:
106             noise.add(point)
107
108
109     # List clusters
110     print("Number of clusters formed :" + str(len(clusters)))
111     print("Noise points :" + str(len(noise)))
112
113     # Calculate purity
114     compute_purity(clusters, len(self.dataSet))
115     compute_NMI(clusters, self.noOfLabels)
116     DataPoints.writeToFile(noise, clusters, "DBSCAN_" + self.dataname +

```

```
117 # -----
118 def removeDuplicates(self, n, n1):
119     for point in n1:
120         isDup = False
121         for point1 in n:
122             if point1 == point:
123                 isDup = True
124                 break
125         if not isDup:
126             n.append(point)
127
128
```