Data Mining Assignment 5 - Aradhya Mathur

10.2

In [34]:
```
import numpy as np
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')
```

a)

In [35]:
```
points = np.array([[2,10],[2,5],[8,4],[5,8],[7,5],[6,4],[1,2],[4,9]])
initial_centroid = np.array([[2,10],[5,8],[1,2]])
```

In [36]:
```
#Using max_iter we can check for each iteration.
kmeans = KMeans(n_clusters=3, init=initial_centroid, max_iter=1)
kmeans.fit(points)
```

Out[36]:
```
KMeans(init=array([[ 2, 10],
       [ 5,  8],
       [ 1,  2]]), max_iter=1,
       n_clusters=3)
```

In [37]:
```
kmeans.cluster_centers_
```

Out[37]:
```
array([[ 2. , 10. ],
       [ 6. ,  6. ],
       [ 1.5,  3.5]])
```

b)

In [38]:
```
kmeans = KMeans(n_clusters=3, init=initial_centroid, max_iter=2)
kmeans.fit(points)
```

Out[38]:
```
KMeans(init=array([[ 2, 10],
       [ 5,  8],
       [ 1,  2]]), max_iter=2,
       n_clusters=3)
```

In [39]:
```
kmeans.cluster_centers_
```

Out[39]:
```
array([[3.  , 9.5 ],
       [6.5 , 5.25],
       [1.5 , 3.5 ]])
```

In [40]:
```
kmeans = KMeans(n_clusters=3, init=initial_centroid, max_iter=3)
kmeans.fit(points)
```

Out[40]:
```
KMeans(init=array([[ 2, 10],
       [ 5,  8],
       [ 1,  2]]), max_iter=3,
       n_clusters=3)
```

In [41]:
```
kmeans.cluster_centers_
```

Out[41]:
```
array([[3.66666667, 9.        ],
       [7.        , 4.33333333],
       [1.5       , 3.5       ]])
```

In [42]:
```
kmeans = KMeans(n_clusters=3, init=initial_centroid, max_iter=4)
kmeans.fit(points)
```

Out[42]:
```
KMeans(init=array([[ 2, 10],
       [ 5,  8],
       [ 1,  2]]), max_iter=4,
       n_clusters=3)
```

In [43]:
```
kmeans.cluster_centers_
```

Out[43]:
```
array([[3.66666667, 9.        ],
       [7.        , 4.33333333],
       [1.5       , 3.5       ]])
```

As centers at iteration 3 and iteration 4 were same (remain unchanged), these are the final centers of the cluster.

In [44]:
```
kmeans.labels_
```

Out[44]:
```
array([0, 2, 1, 0, 1, 1, 2, 0])
```

This signifies that point[2,10] lies in cluster 0 , point [2,5] lies in cluster 2 similarly for all the points.

All points: [2,10],[2,5],[8,4],[5,8],[7,5],[6,4],[1,2],[4,9]
Cluster 0: [2,10] , [5,8] , [4,9]
Cluster 1: [8,4] , [7,5] , [6,4]
Cluster 2: [2,5] , [1,2]

11.2

In [45]:
```
from scipy.special import comb
import pandas as pd
import numpy as np
```

In [46]:
```
#Ada,Bob
AB_Probability = []
AB_Euclidean = []
AB_Jaccard = []
AB_df = pd.DataFrame()
def probability_AB(i):
    probability = (comb(N=7,k=i-3)*comb(N=990,k=10-i))/(comb(N=997,k=7))
    return(probability)

for i in range(3,11):
    AB_Probability.append(probability_AB(i))
    AB_Euclidean.append(np.sqrt(i))
    AB_Jaccard.append(i/(20-i))
    AB_df = AB_df.append({'I': i ,'Probability': probability_AB(i) , 'Euclidean Distance': 2*(10-i) , 'Jaccard': i/(20-i)},ignore_index=True)
```

In [47]:
```
AB_df #Ada,Bob
```

Out[47]:

	I	Probability	Euclidean Distance	Jaccard
0	3.0	9.517334e-01	14.0	0.176471
1	4.0	4.739323e-02	12.0	0.250000
2	5.0	8.660691e-04	10.0	0.333333
3	6.0	7.319719e-06	8.0	0.428571
4	7.0	2.966451e-08	6.0	0.538462
5	8.0	5.404466e-11	4.0	0.666667
6	9.0	3.643051e-14	2.0	0.818182
7	10.0	5.256928e-18	0.0	1.000000

In [48]:
```
#Ada,Cathy
AC_Probability = []
AC_Euclidean = []
AC_Jaccard = []
AC_df = pd.DataFrame()
def probability_AC(j):
    pro_AC = (comb(N=10,k=j)*comb(N=990,k=10-j))/(comb(N=1000,k=10))
    return(pro_AC)

for j in range(1,11):
    AC_Probability.append(probability_AC(j))
    AC_Euclidean.append(np.sqrt(j))
    AC_Jaccard.append(j/(20-j))
    AC_df = AC_df.append({'J': j ,'Probability': probability_AC(j) , 'Euclidean Distance': 2*(10-j) , 'Jaccard': j/(20-j)},ignore_index=True)
```

In [49]:
```
#Ada,Cathy
AC_df
```

Out[49]:

	J	Probability	Euclidean Distance	Jaccard
0	1.0	9.214765e-02	18.0	0.052632
1	2.0	3.800387e-03	16.0	0.111111
2	3.0	8.247703e-05	14.0	0.176471
3	4.0	1.026772e-06	12.0	0.250000
4	5.0	7.505338e-09	10.0	0.333333
5	6.0	3.171627e-11	8.0	0.428571
6	7.0	7.344917e-14	6.0	0.538462
7	8.0	8.363392e-17	4.0	0.666667
8	9.0	3.758406e-20	2.0	0.818182
9	10.0	3.796369e-24	0.0	1.000000

To find the probability that dist(Ada, Bob) > dist(Ada, Cathy)

In [50]:
```
pr_i = list(AB_df['Probability'])
pr_i.insert(0, 0)
pr_i.insert(1, 0)
pr_i
```

Out[50]:
```
[0,
 0,
 0.9517333554252858,
 0.047393226032356704,
 0.0008660691051559818,
 7.319718603446738e-06,
 2.9664513083877356e-08,
 5.404465946455389e-11,
 3.643050857064637e-14,
 5.2569276437191 5e-18]
```

In [51]:
```
pr_j = list(AC_df['Probability'])
pr_j
```

Out[51]:
```
[0.09214765049540127,
 0.0038003868076005626,
 8.247702803805026e-05,
 1.0267719445887 82e-06,
 7.505338072121046e-09,
 3.1716269743581164e-11,
 7.344917005316234e-14,
 8.363392339049355e-17,
 3.7584057247772208e-20,
 3.7963694189618 26e-24]
```

Using Euclidean

In [52]:
```
sum_euc = 0
for i in range(2, 9):
    for j in range(i+1,10):
        sum_euc += pr_i[i]*pr_j[j]
sum_euc
```

Out[52]:
```
9.847436804201952e-07
```

9.847436804201952e-07 is the probability that dist(Ada, Bob) > dist(Ada, Cathy) using Euclidean Distance

Using Jaccard

In [53]:
```
sum_jac = 0
for i in range(2, 10):
    for j in range(0,i-1):
        sum_jac += pr_i[i]*pr_j[j]
sum_jac
```

Out[53]:
```
0.09233115445920217
```

0.09233115445920217 is the probability that dist(Ada, Bob) > dist(Ada, Cathy) using Jaccard Similarity

In [ ]:
```

```