CS 271 Spring 2020
Assignment 8

Student ID: 014498887
Name: Yash Sahasrabuddhe

CHAPTER 6

Q.2.

a)

The correlation coefficient is used to measure the cluster quality. The adjacency matrix A, of n x n dimensions is defined in such a way that the value of the elements is 1 if two data points belong to the same cluster, or else is 0. There is also a distance matrix D of n x n dimensions, with the elements being the distance between the two data points.

Ideally, if the adjacency matrix element value is 1, then the corresponding distance between the data points must be small and otherwise large. We can define this as, there must be strong inverse correlation between these matrices. This means that the value of $r_{AD} = -1$. This leads to the clusters being easily differentiable.

b)

As we have seen earlier, when $r_{AD} = -1$, then the clusters are easily differentiable. If $r_{AD} \rightarrow 1$, then this means that the data points are not differentiable and hence clustering will not be possible.

Q.4.

Q.4 (A) In the first clustering,

$n_1 = 8$, $n_2 = 5$, $n_3 = 9$.

probability of data, p_{ij} is given by-

$$p_{ij} = M_{ij} / M_j$$

Entropy of cluster is given by-

$$E_j = -1 \sum_{i=1}^{l} p_{ij} \cdot \log(p_{ij})$$

Also, total entropy is given by-

$$E = \frac{1}{n} \sum_{j=1}^{K} M_j E_j$$

Hence, entropies of each cluster are-

$$E_1 = -\left[\left(\frac{6}{8}\right)\cdot\log\left(\frac{6}{8}\right)+\left(\frac{1}{8}\right)\cdot\log\left(\frac{1}{8}\right)+\left(\frac{1}{8}\right)\cdot\log\left(\frac{1}{8}\right)\right]$$

$$\boxed{E_1 = 0.7355}$$

$$E_2 = -\left[\left(\frac{3}{5}\right)\cdot\log\left(\frac{3}{5}\right)+\left(\frac{1}{5}\right)\cdot\log\left(\frac{1}{5}\right)+\left(\frac{1}{5}\right)\cdot\log\left(\frac{1}{5}\right)\right]$$

$$\boxed{E_2 = 0.9502}$$

$$E_3 = -\left[\left(\frac{7}{9}\right)\cdot\log\left(\frac{7}{9}\right) + \left(\frac{1}{9}\right)\cdot\log\left(\frac{1}{9}\right) + \left(\frac{1}{9}\right)\cdot\log\left(\frac{1}{9}\right)\right]$$

$$\boxed{E_3 = 0.6836}$$

Therefore, total entropy is -

$$E = \frac{1}{22}\left[(0.7355\times 8) + (0.9502\times 5) + (0.6836\times 9)\right]$$

$$\boxed{E = 0.7631}$$

Purity is given by -

$$U = \frac{1}{n}\sum_{j=1}^{k} M_j U_j$$

$$\Rightarrow U = \frac{1}{22}(6+3+7)$$

$$= \frac{1}{22}[16]$$

$$\Rightarrow \boxed{U = 0.7272}$$

(B) Similarly, for second cluster,

$n_1 = 8$, $n_2 = 5$, $n_3 = 9$.

Applying the entropy formulas & the purity formula -

$$E_1 = -\left[\left(\frac{3}{8}\right)\cdot\log\left(\frac{3}{8}\right) + \left(\frac{3}{8}\right)\cdot\log\left(\frac{3}{8}\right) + \left(\frac{2}{8}\right)\cdot\log\left(\frac{2}{8}\right)\right]$$

$$\boxed{E_1 = 1.0822}$$

$$E_2 = -\left[\left(\frac{2}{5}\right)\cdot\log\left(\frac{2}{5}\right) + \left(\frac{2}{5}\right)\cdot\log\left(\frac{2}{5}\right) + \left(\frac{1}{5}\right)\cdot\log\left(\frac{1}{5}\right)\right]$$

$$\boxed{E_2 = 1.0548}$$

$$E_3 = -\left[\left(\frac{4}{9}\right)\log\left(\frac{4}{9}\right) + \left(\frac{3}{9}\right)\cdot\log\left(\frac{3}{9}\right) + \left(\frac{2}{9}\right)\cdot\log\left(\frac{2}{9}\right)\right]$$

$$\boxed{E_3 = 1.0608}$$

Therefore,

$$E = \left(\frac{1}{22}\right)\left[(8\times1.0822) + (5\times1.0548) + (9\times1.0608)\right]$$

$$\Rightarrow \boxed{E = 1.0672}$$

Purity will be given by -

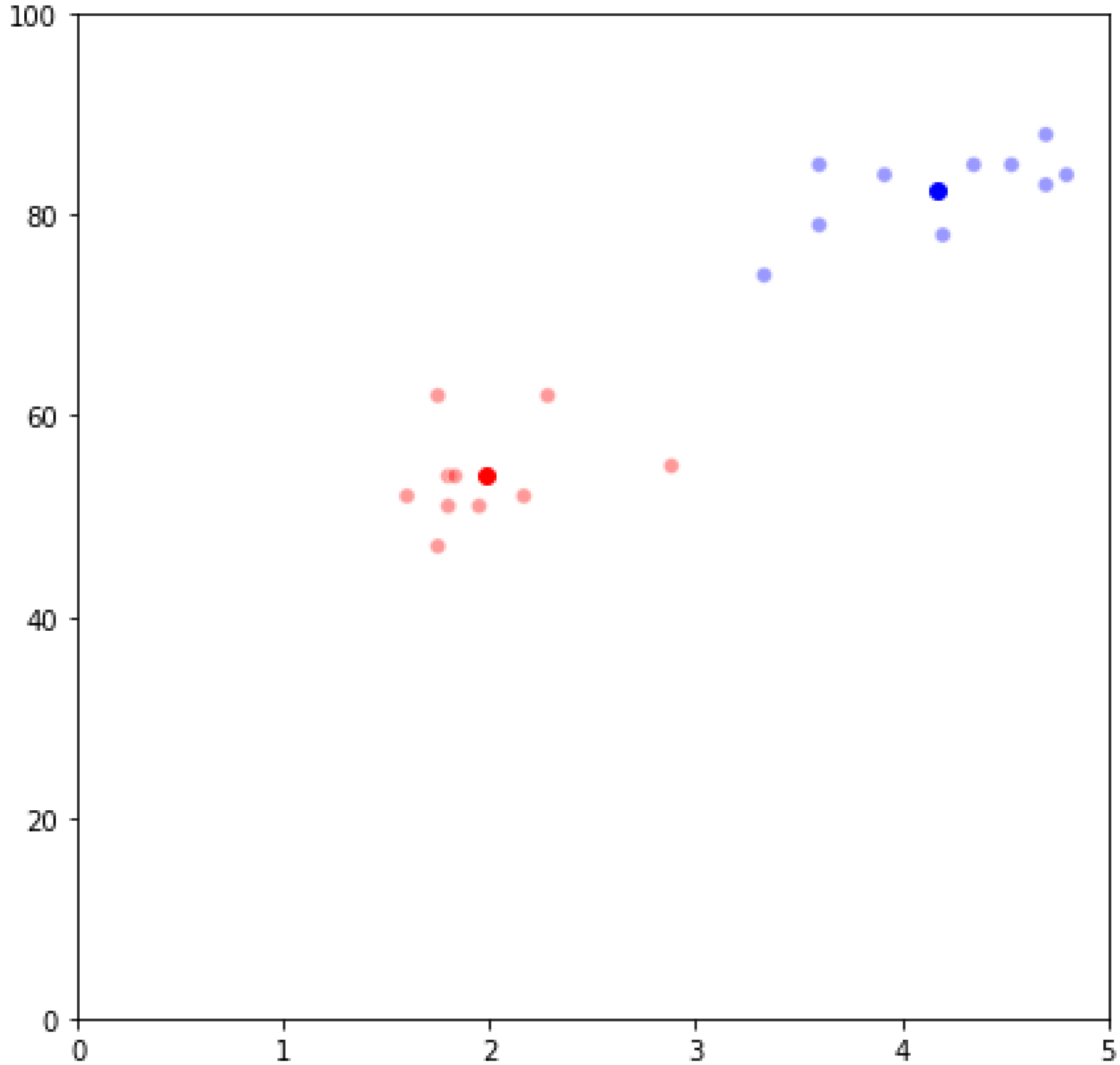
$$U = \left(\frac{1}{22}\right)[3+2+4]$$

$$= \frac{9}{22}$$

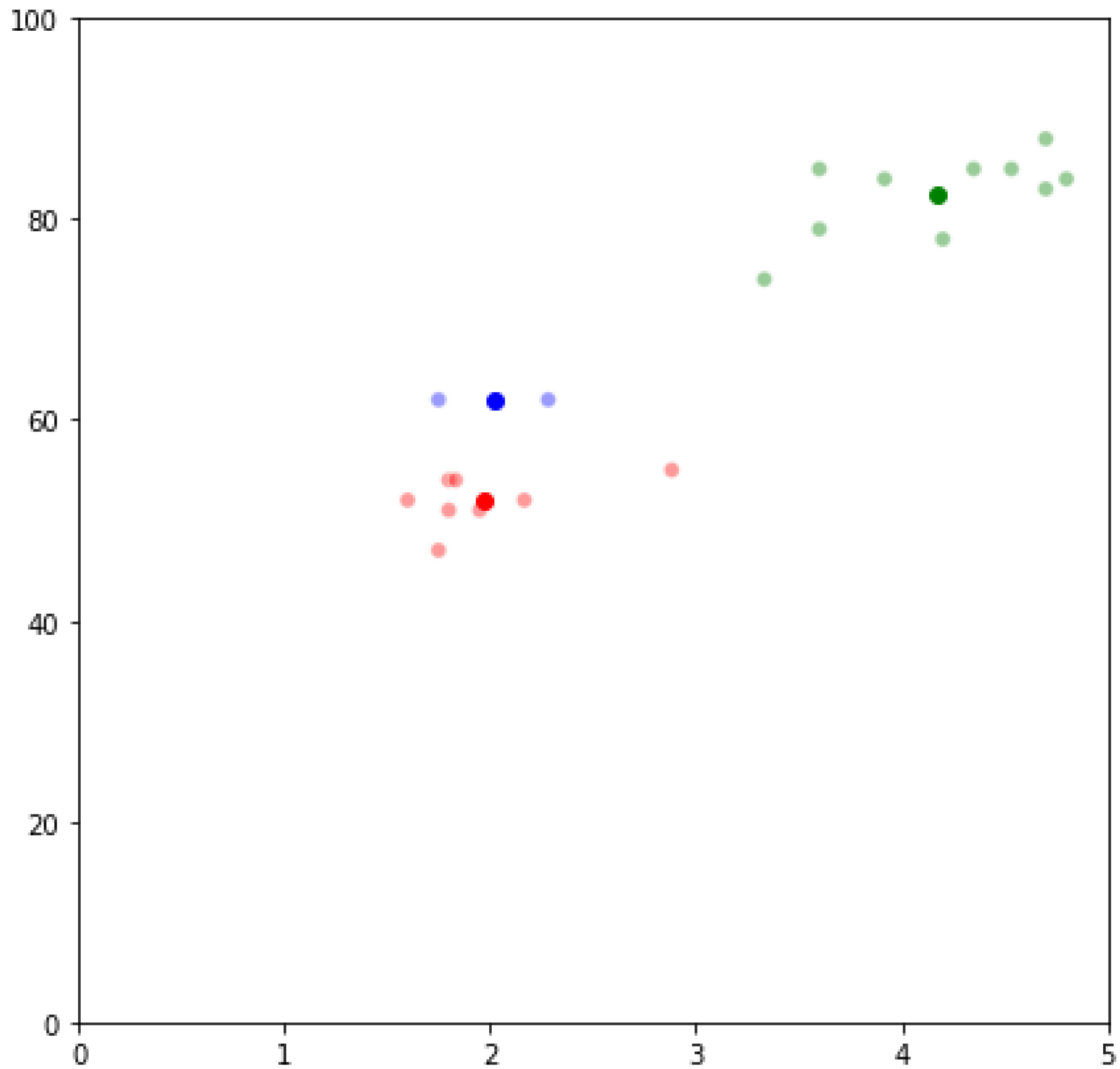
$$\boxed{U = 0.4090}$$

Q.5.

a) The source code for this part where the number of clusters, K = 2 is attached with the submission as Q5a.py. The result of the code is a plotted graph where points are assigned to cluster. The output is –



b) The source code for this part where the number of clusters, K = 3 is attached with the submission as Q5b.py. The result of the code is a plotted graph where points are assigned to cluster. The output is –



Q.7.

The probabilities for the next E step in the coin flip example is calculated using the program Q7.py which is included in the submission. The output of the code is –

```
P-values:
p1 1      0.893847439551
p2 1      0.106152560449
p1 2      0.604605923711
p2 2      0.395394076289
p1 3      0.936986059202
p2 3      0.0630139407976
p1 4      0.464071861944
p2 4      0.535928138056
p1 5      0.826640609138
p2 5      0.173359390862

Process finished with exit code 0
```

Q.8.

a) Using the same initializations for values of $\boldsymbol{\theta}$ & $\boldsymbol{\tau}$, the EM algorithms converges to the same values of $\boldsymbol{\theta}$, that is 0.7934 & 0.5139 respectively. The source code for the EM algorithm is included in the submission as Q8a.py. The output of the code is as below –

```
/usr/bin/python2.7 "/Users/yash/Documents/CS271 ML/Assignments/Assignment8/Q8a.py"


Value of theta1 is:  0.793367649613
Value of theta2 is:  0.513916591214


Value of tau1 is:  0.522751316897
Value of tau2 is:  0.477248683103

Process finished with exit code 0
```

b) Using 3 different initializes, after executing the codes for multiple iterations, **each initialization results converge to the same values** of $\boldsymbol{\theta}$ & $\boldsymbol{\tau}$, that is, to the same values of $\boldsymbol{\theta}$ - 0.7934 & 0.5139 and $\boldsymbol{\tau}$ - 0.5227 & 0.4772. However, there is an interesting catch to the convergence of values of $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$ that the dominating probability of $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$ converges to the higher value. This means that if in one case, $\boldsymbol{\tau}_1 > \boldsymbol{\tau}_2$, or $\boldsymbol{\theta}_1 > \boldsymbol{\theta}_2$ then the converged value of $\boldsymbol{\tau}_1$ and $\boldsymbol{\theta}_1$ will be greater and if $\boldsymbol{\tau}_2 > \boldsymbol{\tau}_1$ or $\boldsymbol{\theta}_2 > \boldsymbol{\theta}_1$, then the converged value of $\boldsymbol{\tau}_2$ and $\boldsymbol{\theta}_2$ will be greater. The code for the same is attached with the submission as Q8b.py and the output is also attached below –

```
/usr/bin/python2.7 "/Users/yash/Documents/CS271 ML/Assignments/Assignment8/Q8b.py"


Original Initialization of theta and tau:
Theta1:  0.6
Theta2:  0.5
tau1:  0.7
tau2:  0.3


After applying EM algorithm:
Theta1:  0.793367649613
Theta2:  0.513916591214
tau1:  0.522751316897
tau2:  0.477248683103


First random initialization of theta and tau:
Theta1:  0.4
Theta2:  0.7
tau1:  0.6
tau2:  0.4


After applying EM algorithm:
Theta1:  0.513916591214
Theta2:  0.793367649613
tau1:  0.477248683103
tau2:  0.522751316897
```

```
Second random initialization of theta and tau:
Theta1:  0.8
Theta2:  0.6
tau1:  0.9
tau2:  0.1


After applying EM algorithm:
Theta1:  0.793367649613
Theta2:  0.513916591214
tau1:  0.522751316897
tau2:  0.477248683103


Third random initialization of theta and tau:
Theta1:  0.6
Theta2:  0.8
tau1:  0.6
tau2:  0.4


After applying EM algorithm:
Theta1:  0.513916591214
Theta2:  0.793367649613
tau1:  0.477248683103
tau2:  0.522751316897

Process finished with exit code 0
```

Q.13.

a) & **b)** The source code for these sub-parts are included in the submission as Q13ab.py in the Chapter 6 sub-folder. The output of the code is-

```
Converged values of Tau, Theta and S for (a) part using given
initializations:
Tau1 0.5000002084482039
Tau2 0.49999979155179614
Mew1 [[ 1.98160168]
 [54.00001977]]
Mew2 [[ 4.17329924]
 [82.49999211]]
S1
[[ 0.12796941  0.41133545]
 [ 0.41133545 20.40047833]]
S2
[[ 0.25380684  1.26527997]
 [ 1.26527997 15.85030796]]
Converged values of Tau, Theta and S for (b) part using random
initializations:
Tau1 0.5000002084482039
Tau2 0.49999979155179614
Mew1 [[ 1.98160168]
 [54.00001977]]
Mew2 [[ 4.17329924]
 [82.49999211]]
S1
[[ 0.12796941  0.41133545]
 [ 0.41133545 20.40047833]]
S2
[[ 0.25380684  1.26527997]
 [ 1.26527997 15.85030796]]

In [6]:
```

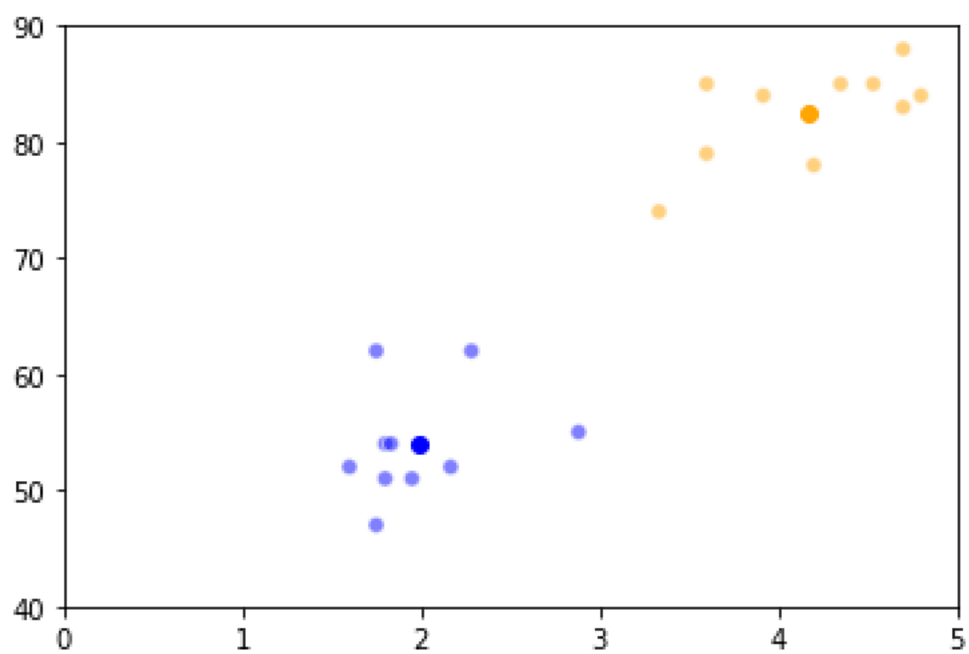
c) The source code for this is included in the submission as Q13c.py in the Chapter 6 sub-folder. The output of the code is-

```
Converged values of Tau, Theta and S for (c) part for three
clusters:
Tau1 0.4500000181596757
Tau2 0.40020812595847977
Tau3 0.14979185588184452
Mew1 [[ 1.88144448]
 [53.88888893]]
Mew2 [[ 4.34956889]
 [83.2492176 ]]
Mew3 [[ 3.27165407]
 [71.31886827]]
S1
[[ 0.04187429  0.34571608]
 [ 0.34571608 22.54320902]]
S2
[[0.1569682  0.73595903]
 [0.73595903 9.44687769]]
S3
[[8.75567217e-02 3.66696963e+00]
 [3.66696963e+00 1.53583030e+02]]

In [22]:
```

d) The graph achieved in Q5 part (a) is very similar to the graph achieved using EM clustering. The centroids are not exactly same, but the points assigned to the clusters in both the cases is same. The source code for this is included in the submission as Q13c.py in the Chapter 6 sub-folder. The output of the code is-

```
Converged values of Tau, Theta and S for (d) part using given
initializations:
Tau1 0.5000002084482039
Tau2 0.49999979155179614
Mew1 [[ 1.98160168]
 [54.00001977]]
Mew2 [[ 4.17329924]
 [82.49999211]]
S1
[[ 0.12796941  0.41133545]
 [ 0.41133545 20.40047833]]
S2
[[ 0.25380684  1.26527997]
 [ 1.26527997 15.85030796]]
The centroids of the data will be the converged values of mew
The centroids are:  [[ 1.98160168 54.00001977]
 [ 4.17329924 82.49999211]]
```



In [31]:

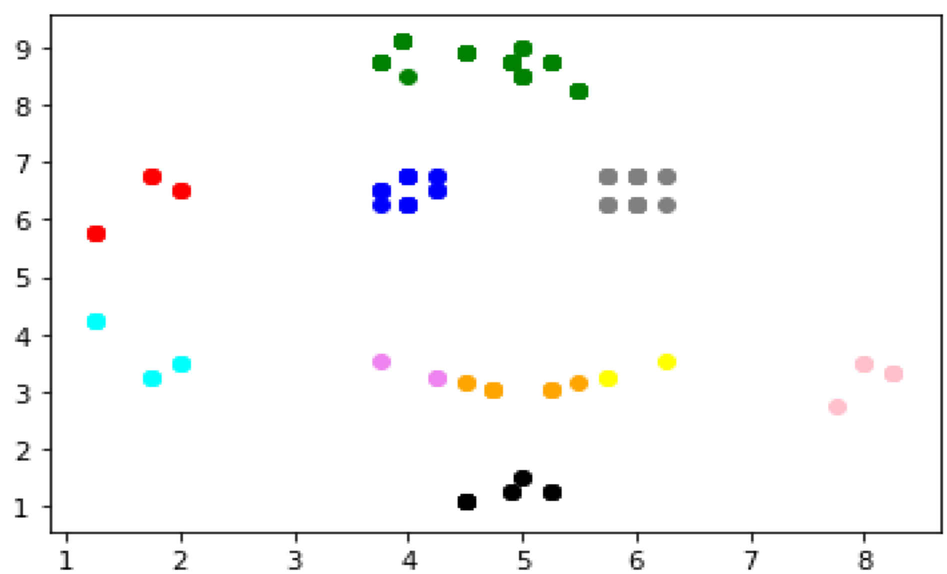
Q.16.

a) A point is marked as visited first at the time of cluster formation. These visited points can be added to the clusters if the distance between the visited point and a core point is less than the epsilon value defined.

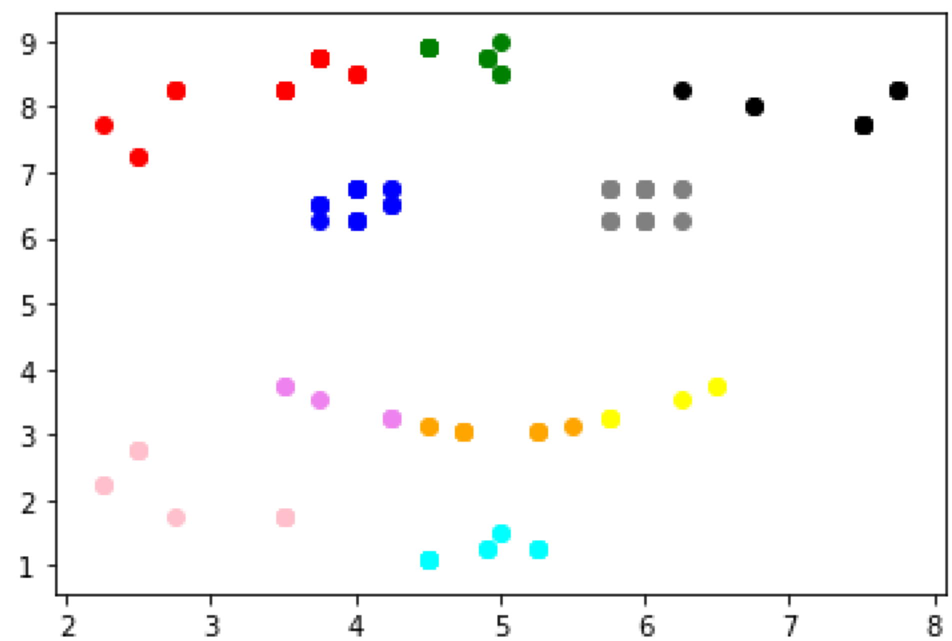
b) If the point X is at an equal distance from multiple core points, then the point will be added to the last cluster in the iteration. Hence, if the order of cluster iteration changes, then the point X can be assigned to different clusters.

c) The source code for this part is added with the submission in the Chapter 6 codes folder as Q16.py. The graphs for different initializations of m and epsilon are the outputs as below: -

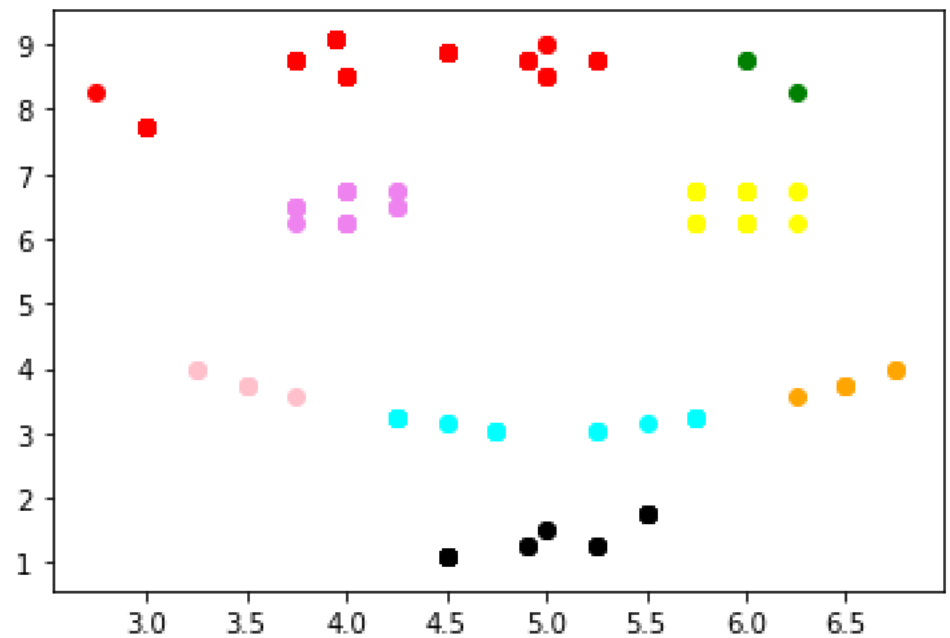
```
Elements in Cluster 1: 4
Elements in Cluster 2: 11
Elements in Cluster 3: 5
Elements in Cluster 4: 4
Elements in Cluster 5: 3
Elements in Cluster 6: 4
Elements in Cluster 7: 2
Elements in Cluster 8: 2
Elements in Cluster 9: 6
Elements in Cluster 10: 6
Number of Outliers: 56
m: 3
epsilon: 0.6
```



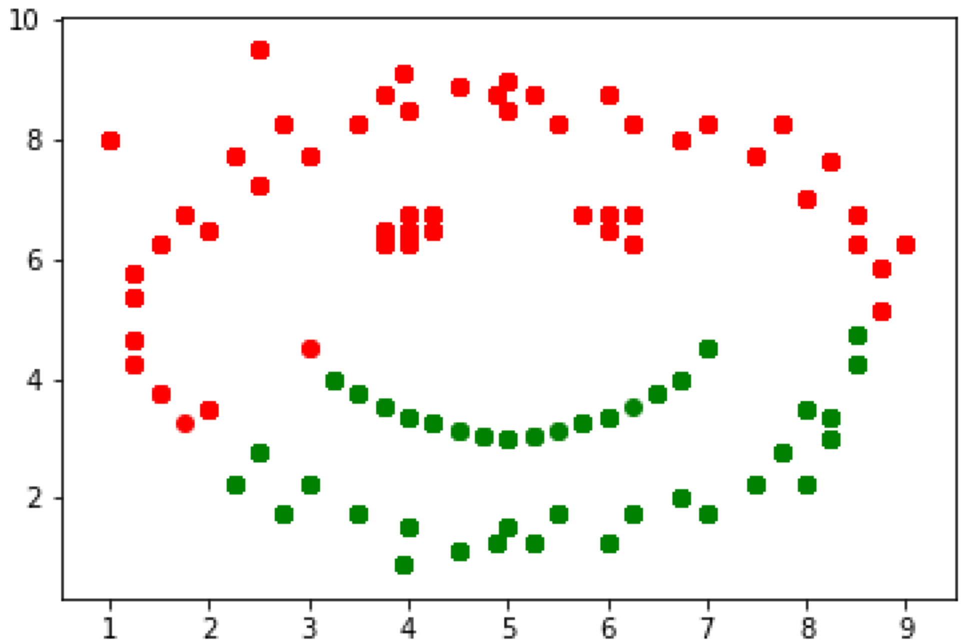
```
plt.show()
Elements in Cluster 1: 7
Elements in Cluster 2: 5
Elements in Cluster 3: 4
Elements in Cluster 4: 5
Elements in Cluster 5: 4
Elements in Cluster 6: 4
Elements in Cluster 7: 3
Elements in Cluster 8: 3
Elements in Cluster 9: 6
Elements in Cluster 10: 6
Number of Outliers: 56
m: 4
epsilon: 0.75
```



```
Elements in Cluster 1: 12
Elements in Cluster 2: 2
Elements in Cluster 3: 6
Elements in Cluster 4: 6
Elements in Cluster 5: 3
Elements in Cluster 6: 3
Elements in Cluster 7: 6
Elements in Cluster 8: 6
Number of Outliers: 59
m: 5
epsilon: 1
```



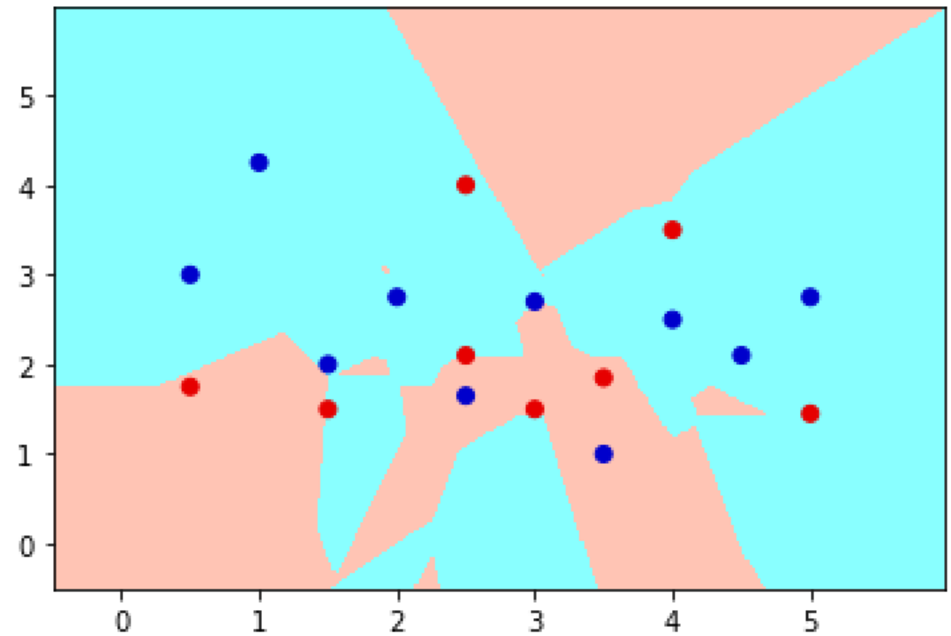
```
Elements in Cluster 1: 56
Elements in Cluster 2: 41
Number of Outliers: 6
m: 10
epsilon: 2
```



CHAPTER 7

Q.1.

The source code for this question has been added with the submission in the respective folder of chapter 7 with file name as Q1.py. The plotted output graph of the code is-



Q.4.

Q.4. Given –

The datasubsets according to the equation –

$$A = \{1, 2, 3, 5, 10\}$$

i	Malware		Benign	
i	$S(X_i)$	$H(X_i)$	$S(Y_i)$	$H(Y_i)$
1	120	7	120	4
2	120	7	130	5
3	100	6	140	5
5	100	6	110	6
10	110	6	120	7
Averages	110	6.4	124	5.4

$\therefore$ Threshold of $S(X_i) = 117$

& threshold of $H = 5.9$

Using these thresholds,

first using S,

$$T_m = \{X_3, X_5, X_{10}, Y_5\}$$

$$T_b = \{X_1, X_2, Y_1, Y_2, Y_3, Y_{10}\}$$

Calculating the entropies,

$$H(T_m) = -\frac{3}{4} \cdot \log\left(\frac{3}{4}\right) - \frac{1}{4} \cdot \log\left(\frac{1}{4}\right)$$

$$\boxed{H(T_m) = 0.8113}$$

$$H(T_b) = \frac{-2}{6} \cdot \log\left(\frac{2}{6}\right) - \left(\frac{4}{6}\right) \log\left(\frac{4}{6}\right)$$

$$\Rightarrow \boxed{H(T_b) = 0.9183}$$

Now,

Information Gain (G_s) is

$$G(s) = 1 - \left[\frac{4}{10} \times 0.8113 + \frac{6}{10} \times 0.9183\right]$$

$$\boxed{G_s = 0.1245}$$

Now, using H as threshold,

$$T_m = \{X_1, X_2, X_3, X_5, X_{10}, Y_5, Y_{10}\}$$

$$T_b = \{Y_1, Y_2, Y_3\}$$

Now,

$$H(T_m) = \frac{-5}{7} \cdot \log\left(\frac{5}{7}\right) - \frac{2}{7} \log\left(\frac{2}{7}\right)$$

$$\boxed{H(T_m) = 0.8631}$$

$$H(T_b) = -\frac{3}{3}\log\left(\frac{3}{3}\right)$$

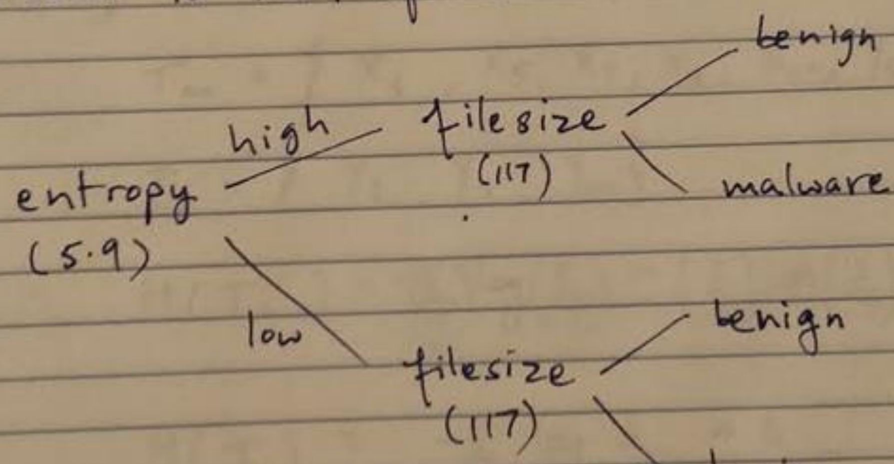
$$\boxed{H(T_b) = 0}$$

Therefore,

$$G_H = 1 - \left[\frac{7}{10} \times 0.8631 + 0\right]$$

$$\Rightarrow \boxed{G_H = 0.3958}$$

Since, $G_H > G_S$, we will keep entropy closer to root of tree.



(Decision tree 1)

(ii) Data subsets using B

$$B = \{3, 5, 7, 9, 10\}$$

i	Malware		Benign	
	$S(X_i)$	$D(X_i)$	$S(Y_i)$	$D(Y_i)$
3	100	34	140	26
5	100	35	110	20
7	100	32	140	28
9	100	32	100	24
10	110	34	120	25
Avg.	102	33.4	122	24.6

$\therefore$ Threshold for $S = 112$

& threshold for $D = 29$

Using S as threshold,

$$T_m = \{X_3, X_5, X_7, X_9, X_{10}, Y_5, Y_9\}$$

$$T_b = \{Y_3, Y_7, Y_{10}\}$$

$$\therefore \quad H(T_m) = -\left(\frac{5}{7}\right)\log\left(\frac{5}{7}\right) - \left(\frac{2}{7}\right)\log\left(\frac{2}{7}\right) = 0.8631$$

$$H(T_b) = -\frac{3}{3}\log\frac{3}{3} = 0$$

$$\therefore \quad G_S = 1 - \left[\frac{7}{10} \times 0.8631 + 0\right]$$

$$\boxed{G_S = 0.39583}$$

Using D as threshold,

$T_m = \{X_3, X_5, X_7, X_9, X_{10}\}$

$T_b = \{Y_3, Y_5, Y_7, Y_9, Y_{10}\}$

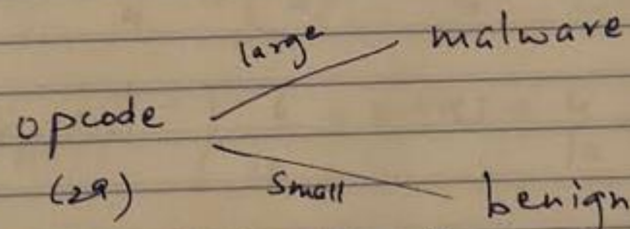
$\therefore \; H(T_m) = H(T_b) = -\frac{5}{5} \log\left(\frac{5}{5}\right) = 0$

$(\text{As } n(T_m) = n(T_b))$

$\therefore \; \boxed{G_D = 1 - 0 = 1}$

Since, $G_D > G_S$,

$\therefore$ Decision Tree is —



Decision Tree 2

(iii) Now, using C as subset.

$C = \{1, 2, 6, 8, 10\}$

i	Malware $H(X_i)$	Malware $D(X_i)$	Benign $H(Y_i)$	Benign $D(Y_i)$
1	7	32	4	22
2	7	28	5	23
6	5	27	7	26
8	6	33	4	21
10	6	34	7	25

Averages   6.2   30.8   5.4   22.2

$\therefore$ Threshold of $H = 5.8$

& threshold of $D = 26.5$

Using H as threshold,

$$T_m = \{X_1, X_2, X_8, X_{10}, Y_6, Y_{10}\}$$

$$T_b = \{X_6, Y_1, Y_2, Y_8\}$$

$$\therefore \quad H(T_m) = -\left(\frac{4}{6}\right)\log\left(\frac{4}{6}\right) - \left(\frac{2}{6}\right)\log\left(\frac{2}{6}\right) = 0.9183$$

$$H(T_b) = -\frac{1}{4}\log\left(\frac{1}{4}\right) - \frac{3}{4}\log\left(\frac{3}{4}\right) = 0.8113$$

$$G_H = 1 - \left[\frac{6}{10} \times 0.9183 + \frac{4}{10} \times 0.8113\right]$$

$$\boxed{G_H = 0.1245}$$

Using D as threshold,

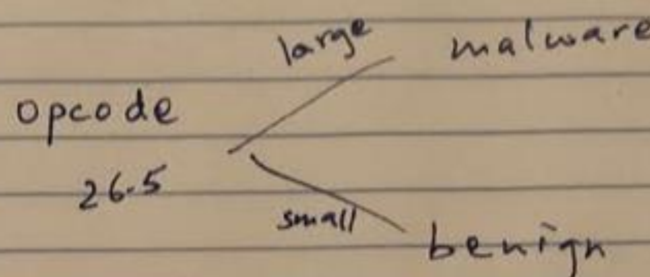
$$T_m = \{X_3, X_5, X_7, X_9, X_{10}\}$$

$$T_b = \{Y_3, Y_5, Y_7, Y_9, Y_{10}\}$$

$$\therefore\ H(T_m) = H(T_b) = -\frac{5}{5}\log\left(\frac{5}{5}\right) = 0 \quad \left(\because n(T_m) = n(T_b)\right)$$

$$\therefore\ \boxed{G_D = 1}$$

Since, $G_D > G_H$, the decision tree is



Decision tree 3

Samples to classify —

Sample	$S(V_i)$	$H(V_i)$	$D(V_i)$
V_1	100	7	27
V_2	130	7	28
V_3	115	4	30
V_4	105	4	35
V_5	140	6	20

Results from Decision Tree 1 → (M, B, B, B, B)
Results from Decision Tree 2 → (B, B, M, M, B)
Results from Decision Tree 3 → (M, M, M, M, B)

∴ Final results of classification is

$(V_1, V_2, V_3, V_4, V_5) = (M, B, M, M, B)$

Q.7.

Q.7. (a) Given in equation 7.12,

$$L(w, \lambda) = -\frac{1}{2} w^t S_B w + \frac{1}{2} \lambda (w^t S_w w - 1)$$

To maximize, we will take partial derivatives w.r.t w & λ,

We obtain,

$$-\frac{1}{2}(2 S_B) w + \frac{1}{2}\lambda (2 S_w W)$$

$$\Rightarrow -S_B W + \lambda S_w W = 0$$

Therefore,

$$S_B W = \lambda S_w w$$

(b) We already know that,

$$S_B W = \lambda S_w W$$

If S^{-1} exists, we can write the above equation as -

$$SW = \lambda W \quad , \quad (S = S_W^{-1} S_B)$$

Here, a large value of λ must be chosen as for classification the S_B values must be large and S_w values should be small. This can be acheived using a large eigen vector, corresspording to which ~~t~~the eigen value will also be dominant.

Q.11.

Q.11

Given -

$$J(w) = \frac{M(w)}{C(w)}$$

$$\Rightarrow J(w) = \frac{(\hat{\mu}_x - \hat{\mu}_y)^2}{\hat{s}_x^2 + \hat{s}_y^2}$$

Simplifying as $w^T = (w_1 \; w_2)$,

$$J(w) = \frac{\left(w^T(\mu_x) - w^T(\mu_y)\right)^2}{\sum_{i=1}^{m} w^T (x_i - \mu_x) . w (x_i - \mu_x)^T + \sum_{i=1}^{n} (y_i - \mu_y) w (y_i - \mu_y)^T}$$

The scatter matrices are,

$$S_x = \sum_{i=1}^{m} (x_i - \mu_x)(x_i - \mu_x)^T$$

$$S_y = \sum_{i=1}^{n} (y_i - \mu_y)(y_i - \mu_y)^T$$

Substituting S_x & S_y in $J(w)$'s eq^n,

we get,

$$J(w) = \frac{\left(w^T(\mu_x - \mu_y)\right)^2}{w^T S_x w + w^T S_y w}$$

The between class scatter matrix is given by -

$$S_B = (\mu_x - \mu_y)(\mu_x - \mu_y)^T$$

Therefore, substituting it in $J(w)$, we get -

$$J(w) = \frac{w^T S_B w}{w_T S_w w} \quad - \text{I}$$

To maximize this eq^n, we will take partial derivatives w.r.t. w_1 & w_2. and equate them to 0.

$$\frac{d\,J(w)}{d\,w_1} = \frac{(2 S_B . w)([1\ 0]^T).\ w^T S_w w - (2 S_w w)([1\ 0]^T).w^T S_B w}{(w^T S_w w)^2}$$

equating to 0, we get,

$$(2 S_B w).(w^T S_w . w) - (2 S_w . w).(w^T S_B . w) = 0 \quad - 1$$

similarly, taking derivative w.r.t to 'w_2', we get,

$$(2 S_B w)(w_T S_w . w) - (2 S_w w)(w^T S_B w) = 0. \quad - 2$$

Adding eq^n 1 & eq^n 2,

$$(4 S_B w) \cdot (w^T S_w w) - (4 S_w w) \cdot (w^T S_B w) = 0$$

Dividing by $4 w^T S_w w$,

$$S_B w - \left(\frac{w^T S_B w}{w^T S_w w} \right) S_w w = 0$$

$$\Rightarrow \quad S_B w - J(w) S_w w = 0 \text{ (from I)}$$

Assuming that S_w^{-1} exists, we get –

$$S_w^{-1} S_B w = J(w) w$$

$$\Rightarrow \quad \boxed{S w = \lambda w} \quad \left(\lambda = \text{scalar quantity } J(w) \right)$$

ADABOOST

Q.2.

The source code for this question is included in the submissions in the AdaBoost sub-folder as adaboost.py. The output of the source code is-

```
Accuracy for classifier C250:  90.0


The iterations for which accuracy is 90% or greater are:
[167, 247, 250]
```

Note- Discussed the homework questions with Aditi Walia, Akshay Kajale, Hardik Trehan, Harshit Trehan.