Question 1)

Blood Transfusion Set:Supervised, Semi-Supervised, and Unsupervised Learning

a) Data Download:

I separated the data based on the 0s and 1s in the dataset and then combined **20%** of each class; hereby using it as Test set.

The rest of the data was used as Train set.

b) Supervised Learning:

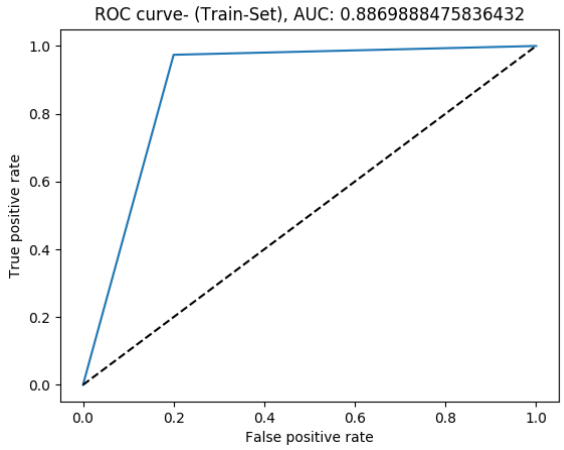
I used 5-fold cross-validation to determine the best value of the L1-penalty parameter from a grid of different values.

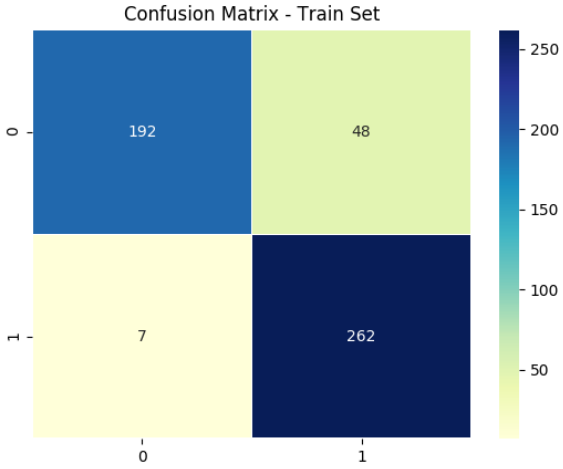
Using the **sklearn.preprocessing** package I was able to normalize the data. Also, I accounted for the imbalance using SMOTE to balance the data by over-sampling the minority class.

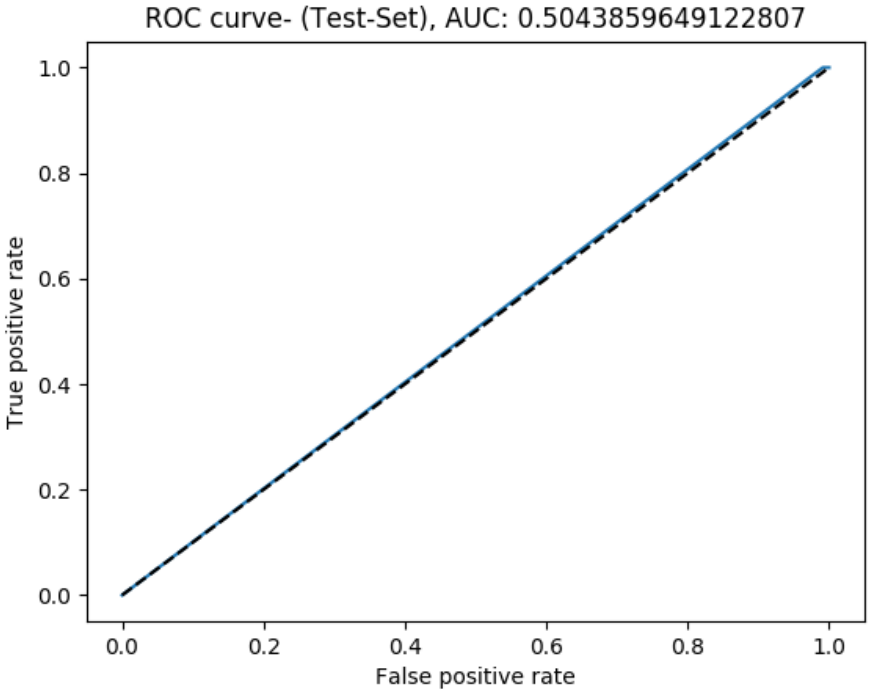
I utilized **GridSearchCV** for this task. I experimented with *250* different values for penalty parameter and determined the best for the model. This value was used as the best parameter.

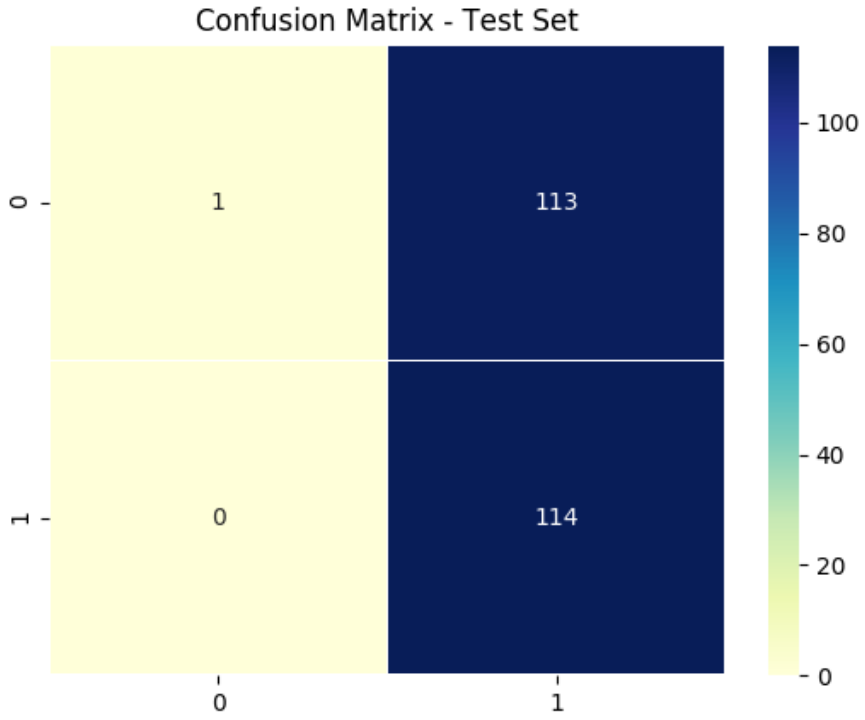
I obtained the following results:

Linear SVC - Supervised Learning- Train set
Fitting...
Best SVM-penalty parameter is {'C': 0.9558277108433734}
Best Score with this parameter is 0.8664047151277013
Predicting...
Accuracy: 0.8919449901768173
AUC: 0.8869888475836432
Linear SVC - Supervised Learning- Test set
Accuracy: 0.5043859649122807
AUC: 0.5043859649122807






ROC curve- (Test-Set), AUC: 0.5043859649122807
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


Confusion Matrix - Test Set
1
113
0
114
0
1
100
80
60
40
20
0

Clearly the model performed badly and failed to classify class-0 well. That is evident by the confusion matrix and the ROC above.

c) Semi-Supervised Learning

i. After separating the data into labeled and unlabeled an L-1 penalized Linear SVM was fit on the labeled data. Much like the previous part of the question I utilized the GridSearchCV to perform 5-fold cross-validation on the model.

ii. I then used the **decision function** in the **LinearSVC** class to determine the data-point in the unlabeled buffer closest to the decision boundary.

I then allowed the model to predict the label for this data-point and added it to the labeled buffer whilst removing it from the unlabeled set at the same time.

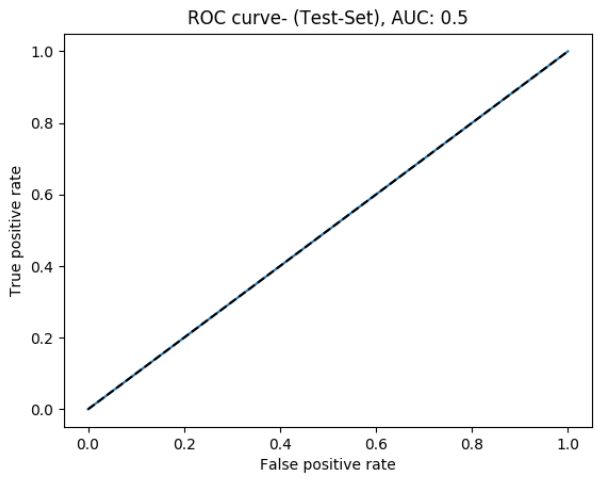
I also printed the index of the closest data-point. After obtaining a final model I used that model to predict the test set data. I determined the misclassification rate and used it gauge the accuracy of the model.

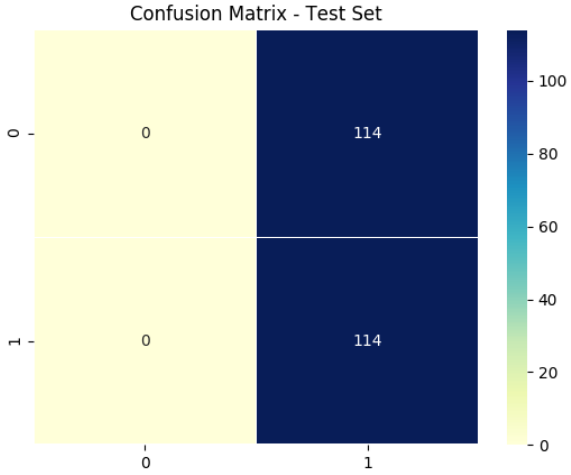
The following results were obtained:

```
Linear SVC - Semi-Supervised Learning- Train set-Labeled iteration: 1
Fitting...
        Best SVM-penalty parameter is  {'C': 6.12248775510204}
        Best Score with this parameter is 0.8427947598253275
index: 216


Linear SVC - Semi-Supervised Learning- Train set-Labeled iteration: 2
Fitting...
        Best SVM-penalty parameter is  {'C': 7.142885714285714}
        Best Score with this parameter is 0.8391304347826087
index: 114


 Linear SVC - Semi-Supervised Learning- Train set-Labeled iteration: 3
Fitting...
        Best SVM-penalty parameter is  {'C': 6.53064693877551}
        Best Score with this parameter is 0.8398268398268398
index: 68


 Linear SVC - Semi-Supervised Learning- Train set-Labeled iteration: 4
Fitting...
        Best SVM-penalty parameter is  {'C': 7.142885714285714}
        Best Score with this parameter is 0.8362068965517241
```

index: 212
.
.
.
.
.
.
.
.
Linear SVC - Semi-Supervised Learning- Train set-Labeled iteration: 297
Fitting...
Best SVM-penalty parameter is {'C': 8.979602040816326}
Best Score with this parameter is 0.9428571428571428
index: 0

Linear SVC - Supervised Learning- Test set
Accuracy: 0.5
AUC: 0.5





As observed above the Semi-Supervised Approach poorly predicts the Test data. It wrongly classifies an entire class of the dataset.

d. Unsupervised Learning

i. I ran the k-means algorithm 100 times by triggering the hyper-parameter **n_init.**

Very often k-means will work fine and come up with very good clusterings. But as we are worried about getting stuck in a bad local minima, one common thing to do is run k-means many times (using different random initial values for the cluster centroids μj). Then, out of all the different clusterings found, pick the one that gives the lowest distortion J(c, μ)

This can be done by triggering the hyper-parameter **random_state**. I assigned a random instance to it.

ii. The parameter **Kmeans.cluster_centers_** gives the co-ordinates of the center for each of the clusters.

I then used **Kmeans.transform()** method this transformed by dataset into a [N X k] matrix, where N is the number of samples and k is the number of clusters.

This matrix represents the euclidean distance of each of the data-points in the dataset from the center of each cluster. This means that the closer a data-point is to a cluster, than another, the more the chances of it being in the former cluster than the later.

I sorted the matrix based on this distance and picked the 30 closest data-points to each cluster.

Now as I have obtained their index, to determine their true label all I needed to do was to use that index and check Y_train (training set-true labels).

After performing this task I used majority polling to label every cluster with a class(0/1).

Now, I have obtained 2 clusters with two separate labels. I then used the predict() method in Kmeans to allow the model to predict labels for the data and compared it to their true labels. This process yielded several miss-classifications, this miss-classification rate contributed to determine the accuracy of the model.

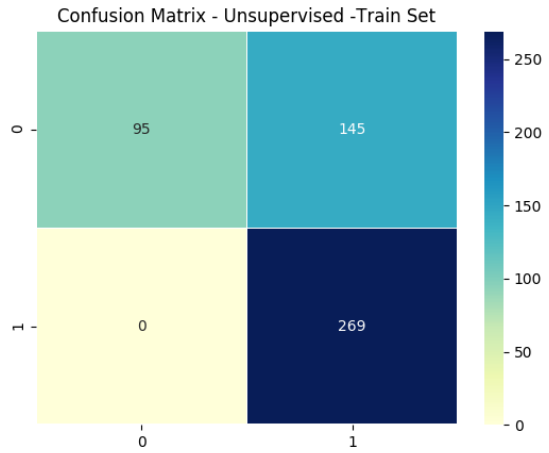
Here are the results obtained:

Centroids:

Cluster-0: [0.06910311 0.00396899 0.99224684 0.08623242]
Cluster-1:[0.00615781 0.00399807 0.99951781 0.02551618]

Confusion Matrix:
[[95 145]
 [0 269]]

Accuracy: 0.7151277013752455



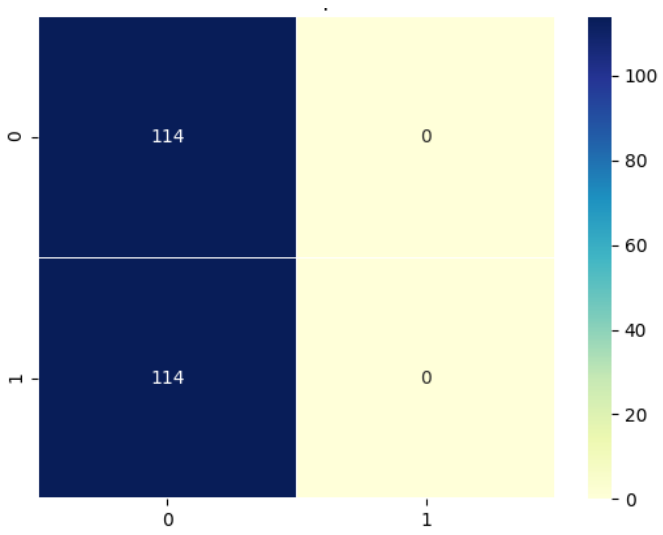
iii. The model obtained in the previous part was used to predict the labels on the Test set.
Here are the results obtained:

k-means Clustering- Unsupervised learning

Predicting…

Confusion Matrix-test-set:
[[114 0]
[114 0]]

Accuracy: 0.5



Much like the Semi-Supervised approach this method has also suffered in predicting labels for 1 of the classes.

f. Comparison of all Approaches

As observed the Supervised learning is the best method that can be used in this scenario where the sample of data is very less.

Due to scarcity of the data the semi-supervised and unsupervised learning approaches do not perform well at all.

Ideally semi-supervised performs well on the train set when compared to the unsupervised learning.
I believe the true powers of the unsupervised learning are realized when we have a large amount of data to deal with, on contrary to the problem-dataset.

For the given data-set Supervised learning method is the winner hands-down.

Question 2)

K-Means Clustering on a Multi-Class and Multi-Label Data Set

a) Determination of the optimal k-value

To determine the number of clusters to be used in the clustering process, I used a technique well-known as Elbow method.

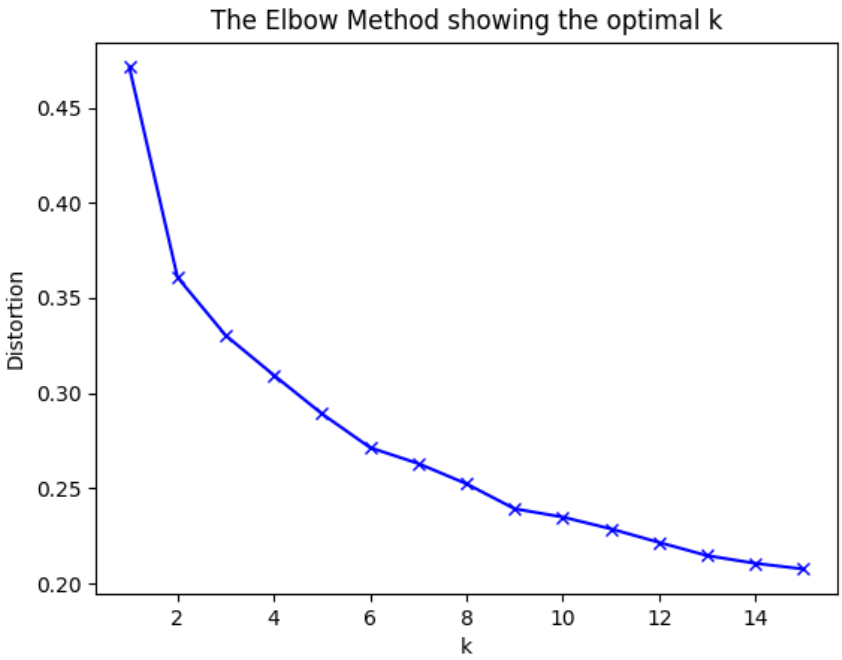
This method looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion". This "elbow" cannot always be unambiguously identified.

After performing the elbow criterion test with 15 different values of k, I obtained the optimal value of **k to be 2.**

Here are the results obtained:

Distortions:
[0.6746431644135847, 0.5190084106643769, 0.4788931840662131, 0.4501682018094673, 0.4140918899382861, 0.3913976428644167, 0.3800351118686867, 0.36289188653734666, 0.34939280078645, 0.3417315625359848, 0.32726085355587753, 0.3232530771702386, 0.3121239227558629, 0.30649315447967757, 0.30424784594256005]

Elbow at k: 2



b. Performing k-means Clustering

After determining the value of k to be 2 for our problem, I proceeded by running the k-means clustering for the dataset.
Post-execution I was able to cluster the data into 2 different clusters, namely Cluster-0 and Cluster-1.

I then determined the label-family for each cluster by reading the true labels of the data-points in the corresponding cluster and then took a majority poll.

In a similar fashion I was able to perform this task of determining the label-genus and label-species for each cluster.

Here are the results I obtained:

```
#######   k-Means Clustering for Anuran Calls   #######

Fitting...

Centers of Centroids:

Cluster- 0 : [ 9.81912190e-01  3.68580302e-01  4.11499956e-01  3.38745161e- 01
  6.24981505e-02  1.66989514e-01  1.12450307e-01 -4.54103922e-02
 -3.26806081e-03  7.66570254e-02  3.29197462e-02 -9.09333134e-03
 -1.41984527e-02  3.32633353e-02  4.57955472e-02 -6.01663237e-03
 -1.87769057e-02  4.60654835e-03 -4.62719786e-04  1.21530071e-02
  1.62062920e-02 -1.24504200e-02]

Cluster- 1 : [ 0.99785469  0.27859975  0.21097633  0.55321929  0.19157641     0.02890766
-0.11521227  0.04465765  0.25965825  0.03534465 -0.26424321          0.09582125         0.31604205
-0.11173041 -0.24924994  0.090127    0.19610765        0.01090299     -0.09847145     -0.11862293
0.05841477  0.18755761]
```

Clustering for label-Family...

Cluster-0-label by Majority Polling: **Hylidae**
Cluster-1-label by Majority Polling: **Leptodactylidae**

Clustering for class-Genus...

Genus Cluster-0-label by Majority Polling: **Hypsiboas**
Genus Cluster-1-label by Majority Polling: **Adenomera**

Clustering for class-Species...

Species Cluster-0-label by Majority Polling: **HypsiboasCordobae**
Species Cluster-1-label by Majority Polling: **AdenomeraHylaedactylus**

Evidently, for *Cluster-0* **Hylidae**, **Hypsiboas** and **HypsiboasCordobae** are the classes for the labels family, genus and species respectively.

And **Leptodactylidae**, **Adenomera** and **AdenomeraHylaedactylus** correspond to the classes for labels family, genus and species respectively in *Cluster-1*.

c. Hamming Loss of the Majoirty Triplet

In the previous part of this question we determined a class for labels family, genus and species for every cluster.

These correspond to a majority triplet for that cluster.

Cluster 0:**Hylidae**, **Hypsiboas** and **HypsiboasCordobae**

Cluster 1: **Leptodactylidae**, **Adenomera** and **AdenomeraHylaedactylus**

I then read the true labels of the data-points and compared them to labels determined by the algorithm. I then determined the hamming loss for each label and also an average hamming loss for the dataset.

Out of curiosity I even calculated the Zero-One Loss for the dataset.

The following results were obtained:

```
#######  k-Means Clustering for Anuran Calls  #######

Fitting...

Clustering for label-Family...

        Cluster-0-label by Majority Polling: Hylidae
        Cluster-1-label by Majority Polling: Leptodactylidae
        Hamming Loss: 0.2334954829742877

Clustering for class-Genus...

        Genus Cluster-0-label by Majority Polling: Hypsiboas
        Genus Cluster-1-label by Majority Polling: Adenomera
        Hamming Loss: 0.2982626824183461

Clustering for class-Species...

        Species Cluster-0-label by Majority Polling: HypsiboasCordobae
        Species Cluster-1-label by Majority Polling: AdenomeraHylaedactylus
        Hamming Loss: 0.3638637943015983
```

Cluster-0 Majority Triplet: Hylidae Hypsiboas HypsiboasCordobae
Cluster-1 Majority Triplet: Leptodactylidae Adenomera AdenomeraHylaedactylus

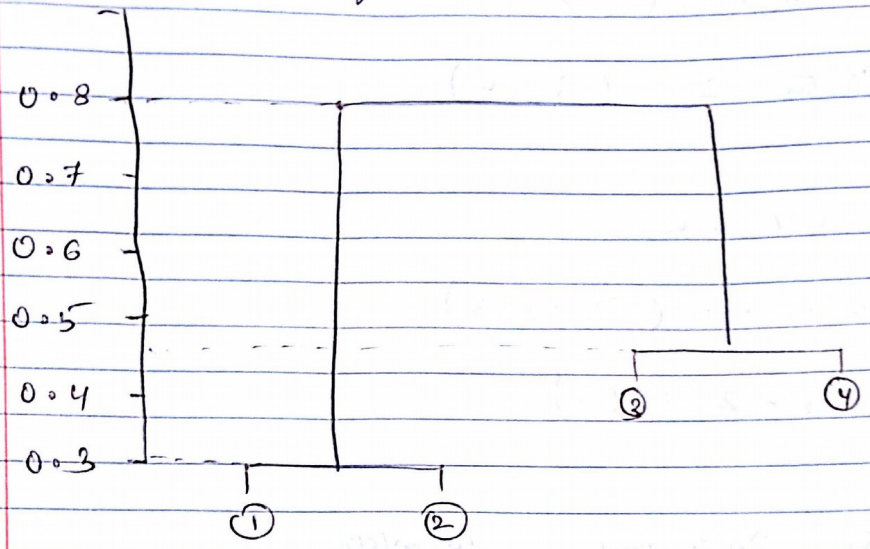
Average Hamming Loss: 0.2985406532314107
Average Zero-one Loss: 0.3638637943015983

The algorithm performs fairly, this is a tribute to the higher number of samples in the data-set.

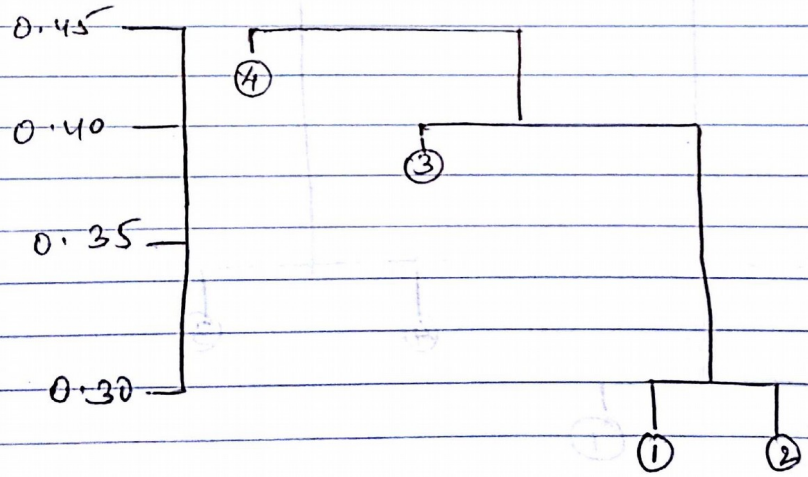
Question 3)

ISLR – 10.7.2.

(a). Cluster Dendogram: Complete Linkage.



(b) Cluster Dendogram: Single Linkage



c) We will have 2 clusters :

Cluster - 1 : (1, 2)

Cluster - 2 : (3, 4)

d) We will have :

Cluster - 1 : ((1, 2), 3)

Cluster - 2 : (4)

e) Cluster Dendogram : Complete.