Assignment1: Report

Aim:

This report aims to provide a brief description for the main functions used in the source codes of the assignment, which are: **myPreparation.r, myClustering.r, myClassification.r**

Functions used:

The various functions used are listed as follows:

myPreparation.r:

names(): This function refers to the column names in the dataset. It accepts a dataset as an argument and denotes the features of that dataset.

str(): This function accepts an object as input and displays the structure of it. We use this function in our code to display the structure of our data frame and find out the data types for each variable.

factor(): This function is used to create a factor. Factors are vectors with particular values. A variable of factor type can only contain values from the factor vector. These values are denoted by “levels”.

integer(): Creates objects of type “integer”; as.integer() is used to assign the datatype integer to a variable.

character(): Creates objects of type “character” using as.character(). Used for string values.

levels(): It denotes the levels attribute of a variable. If the variable is a factor variable, it denotes the factor vector containing the possible values for the factor variable.

saveRDS(): This is used to save a file in the RDS format.

readRDS(): This is used to read a file of RDS format.

myClustering.r:

seed(): This function is used in our code to produce reproducible results. Using the

set.seed(numbervalue) function will use the same workspace that was earlier used for that numbervalue.

jpeg(): This function is used to save an image in the jpeg format. It accepts the file name as attribute, which refers to the location where the image is to be saved. Its other arguments include height and width which specify the height and width of the image respectively.

plot(): This function is used to plot the object given in its arguments.

dev.off(): It is used to shut down the active device. This function is used in the code, to close the filename mentioned in the jpeg() function, once the required data is written to it.

kmeans(): This function is used to perform the k-means clustering on the given data. In the code we mention the data and the number of clusters in its arguments.

hclust(): This function is used to perform hierarchical clustering on data.

cutree(): This function is used to produce the mentioned number of cuts in the dendogram.

dist():

myClassification.r:

nrow(): This function measures the number of rows in the data frame.

sample(): This function takes a sample of the size mentioned, from the elements present in the argument.

predict(): This function is used in our code, to predict the class label from the ctree model. The first attribute is the model based on which the prediction should be made. The second attribute we used is newdata, which is the test features which are used to make the prediction.

as.matrix(): It displays the given object in a matrix format.

table(): We use this function in our code to display the actual labels vs the predicted labels in the table format, which are the values for True positive, False positive, True Negative and False Negative.

sum(): This function is used to find the sum of values present in the arguments.

diag(): It is used to refer to the diagonal elements of a matrix. In our code, we choose this to find the values for True positive and True negative.

apply(): It is used to collectively perform some operation on data from an array or matrix. We use this function in our code to sum the rows and columns to find the number of instances per class and the number of predictions per class

ctree(): It is used to generate a classification tree based on the specified formula and the mentioned data.

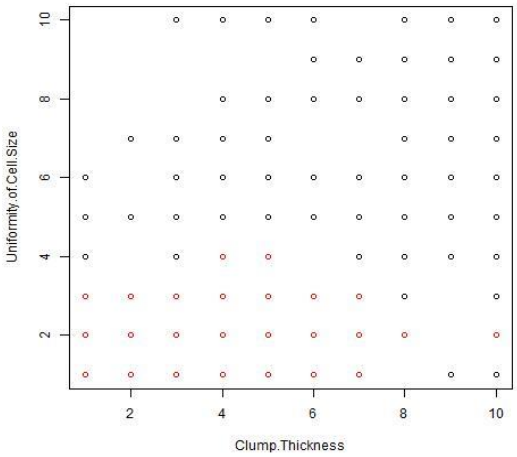
control_ctree(): This function provides more parameters to alter the ctree() function.

Plots:

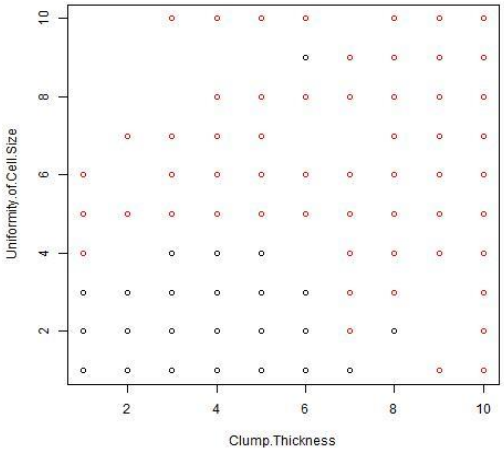
The following are the clusters generated in our code.

K-Means Clustering

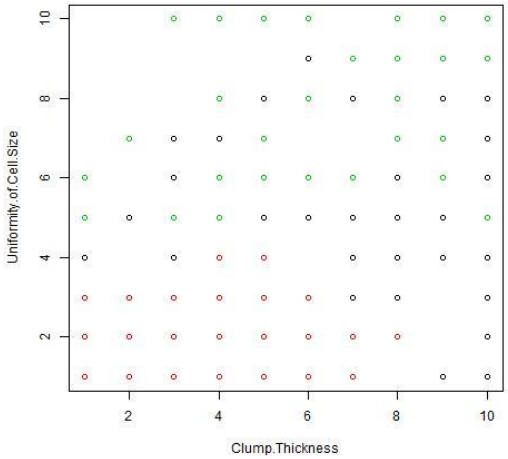
(i)When k=2(default parameters)



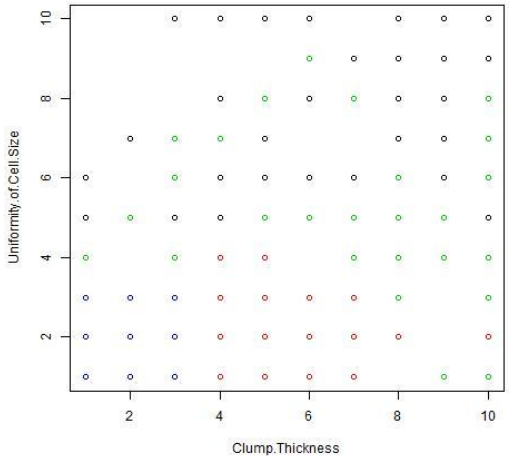
(ii) When k=2, coloured points according to class



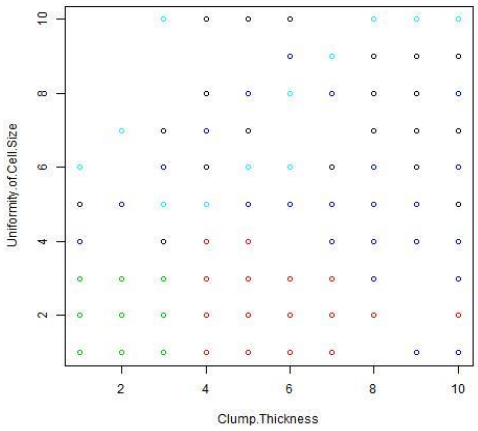
(iii)When k=3



(iv) When k=4

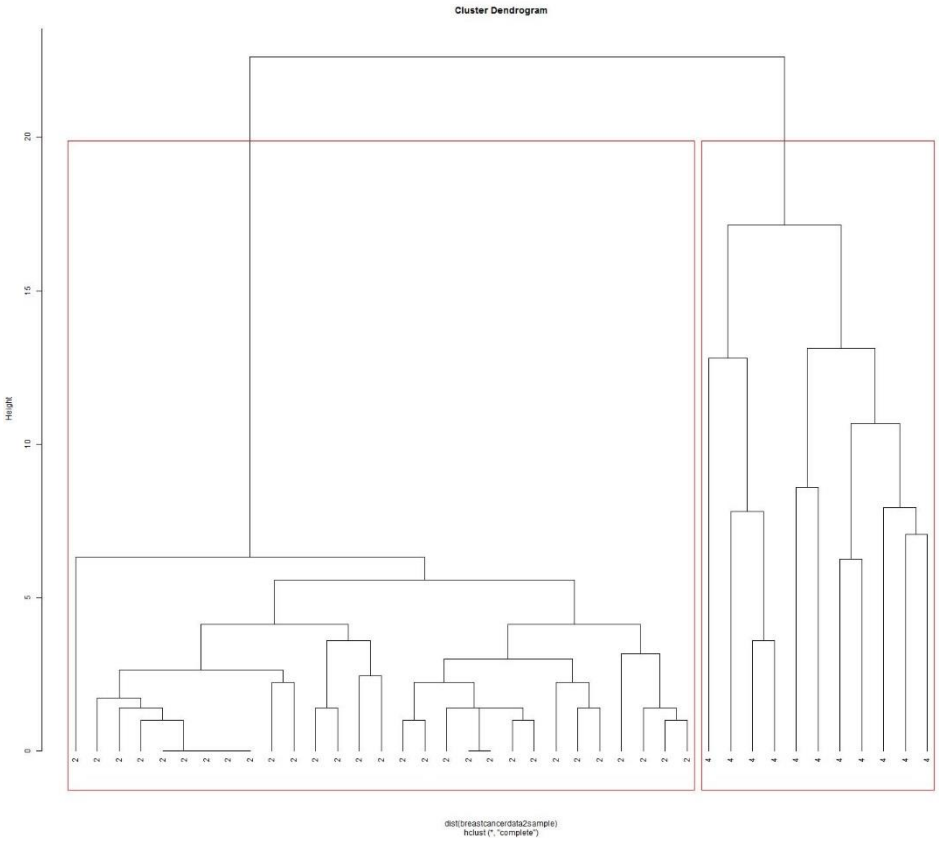


(v)When k=5

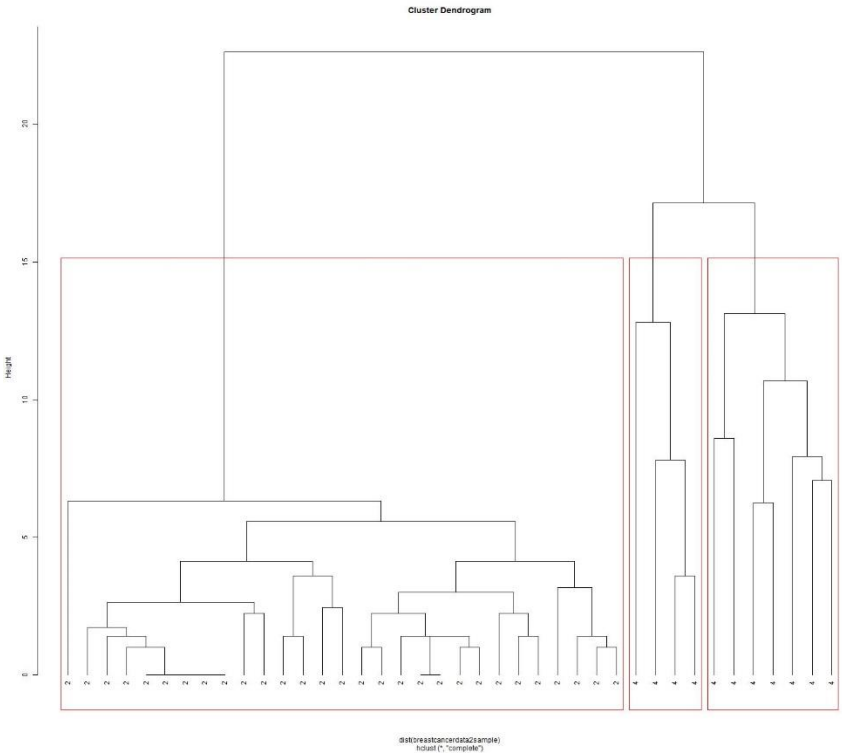


Hierarchial Clustering

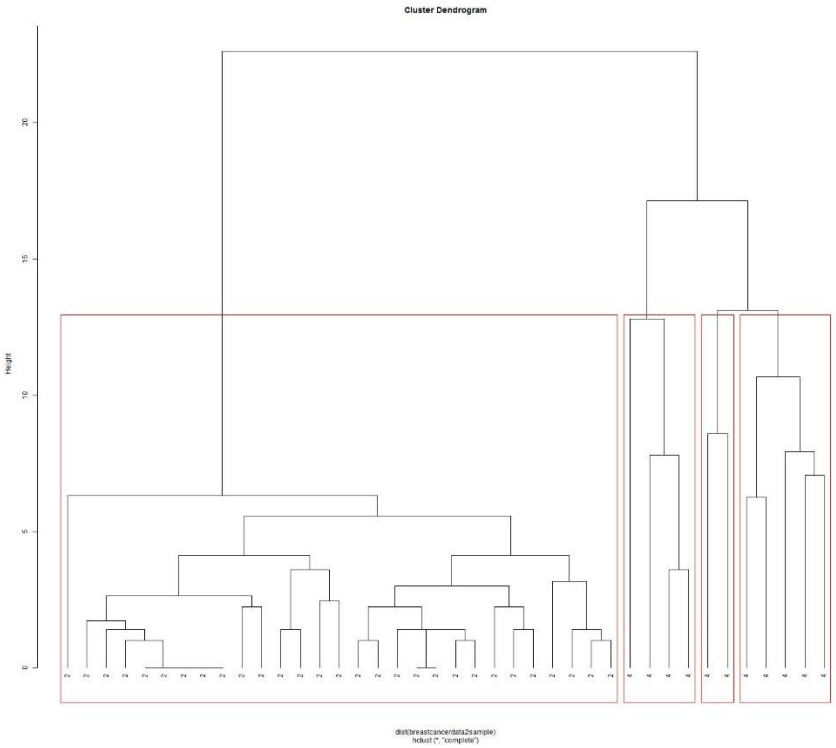
(i)2 clusters, default parameters



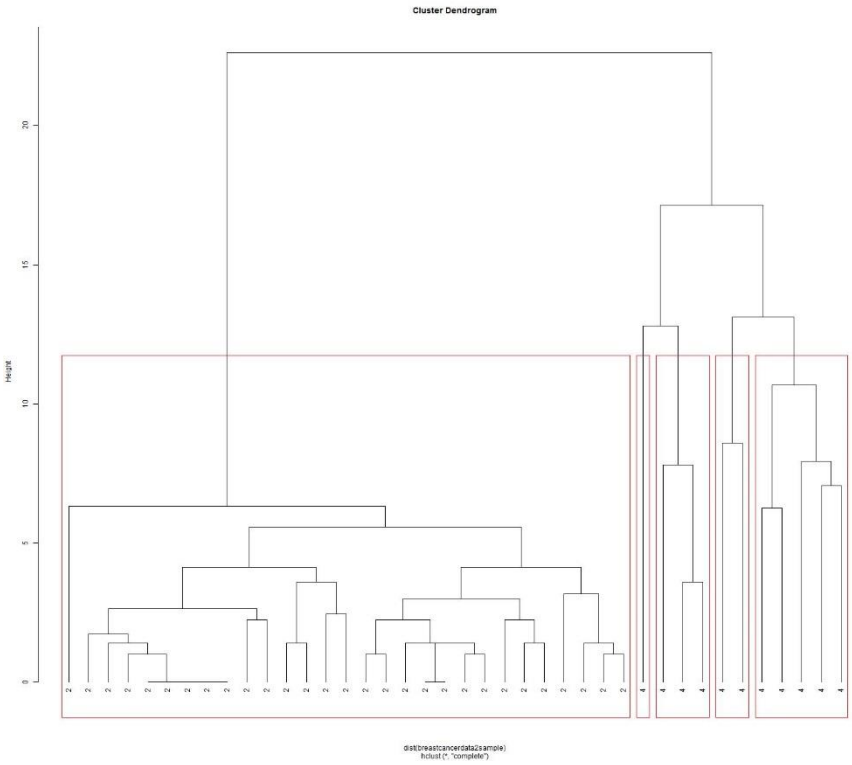
(ii)3 clusters, default parameters



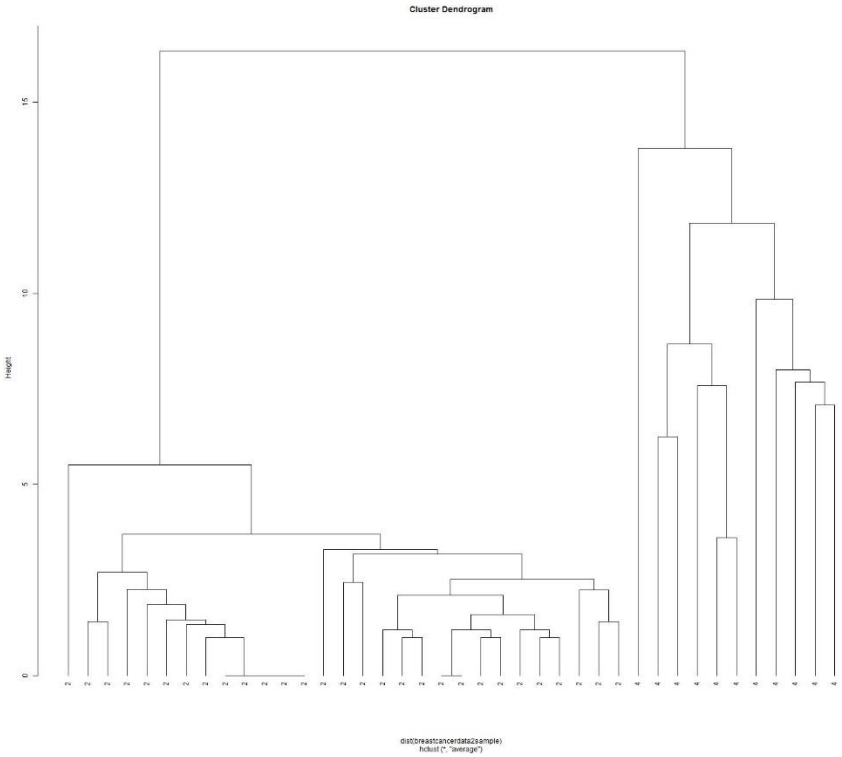
(iii)4 clusters, default parameters



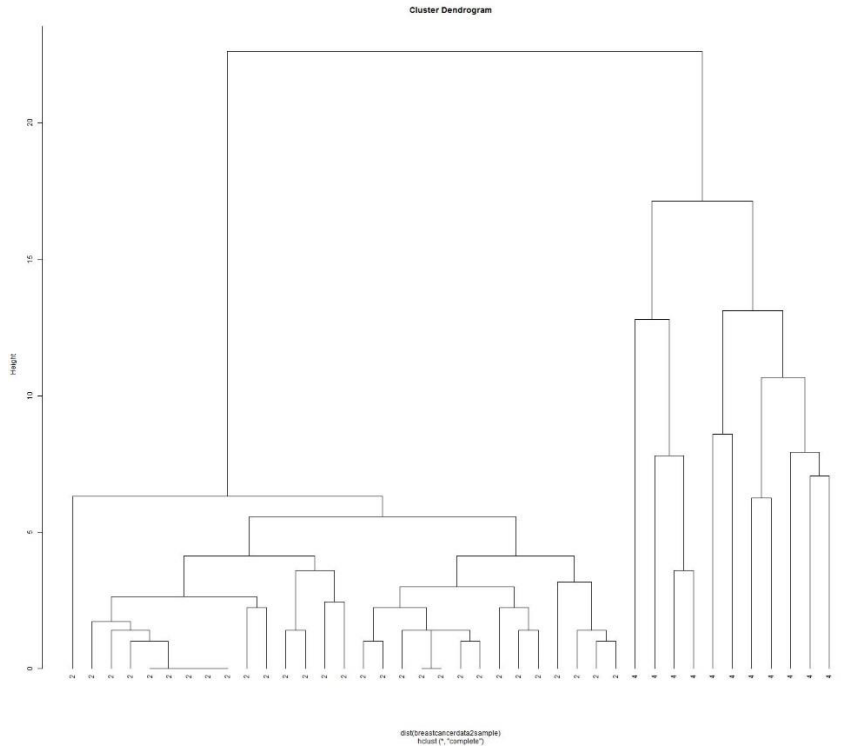
(iv)5 clusters, default parameters



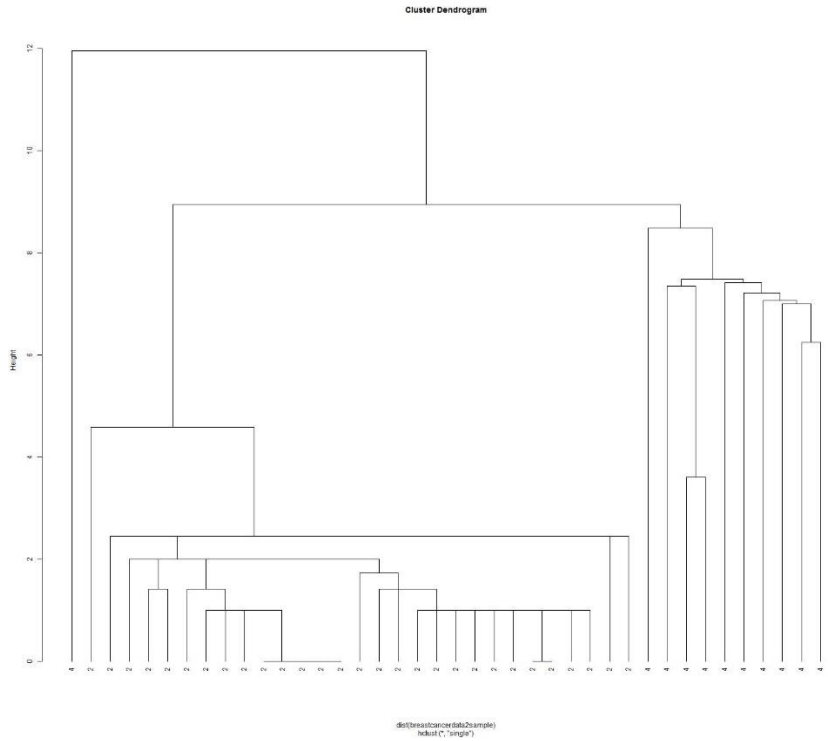
(v)method=average



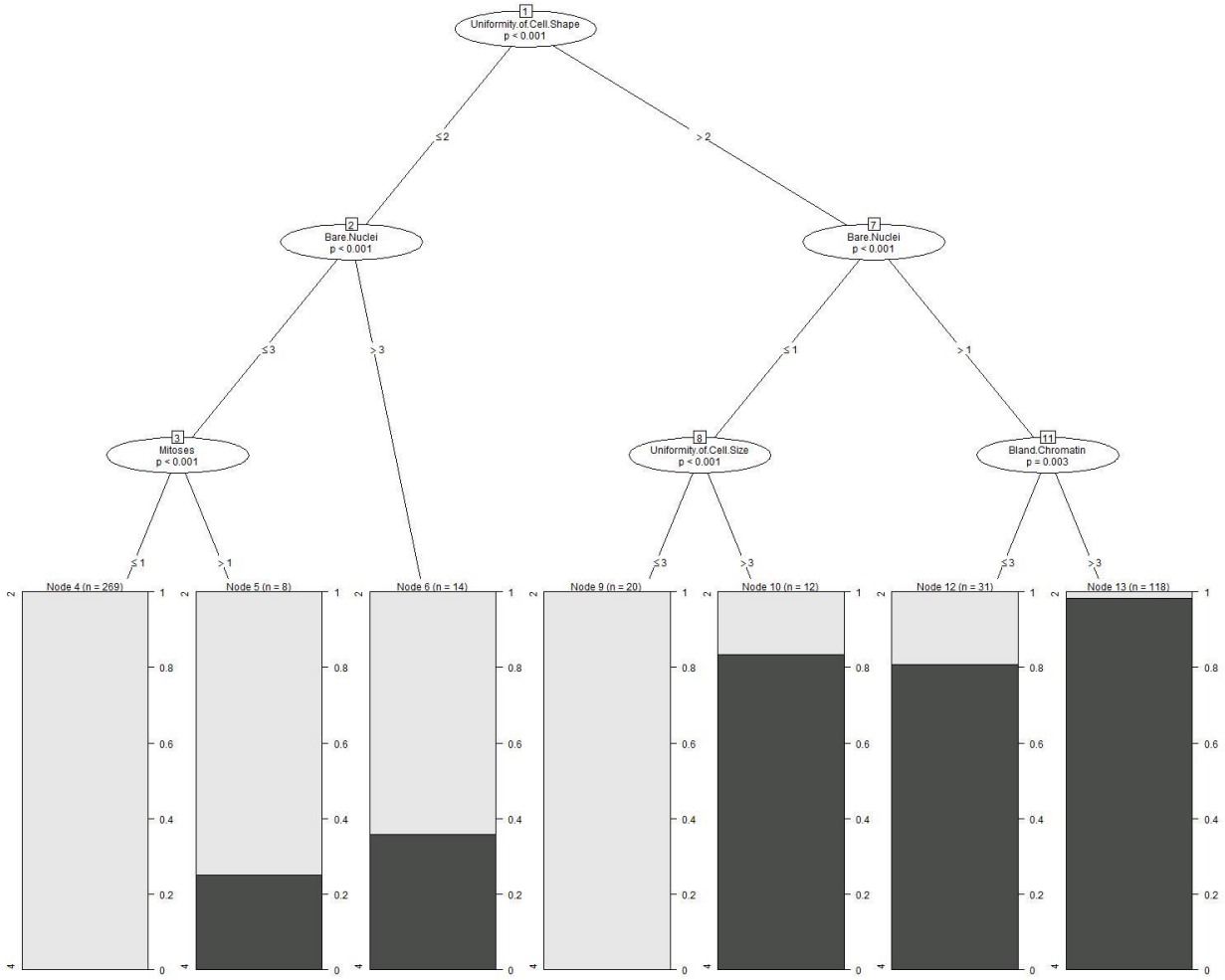
(vi)method= complete



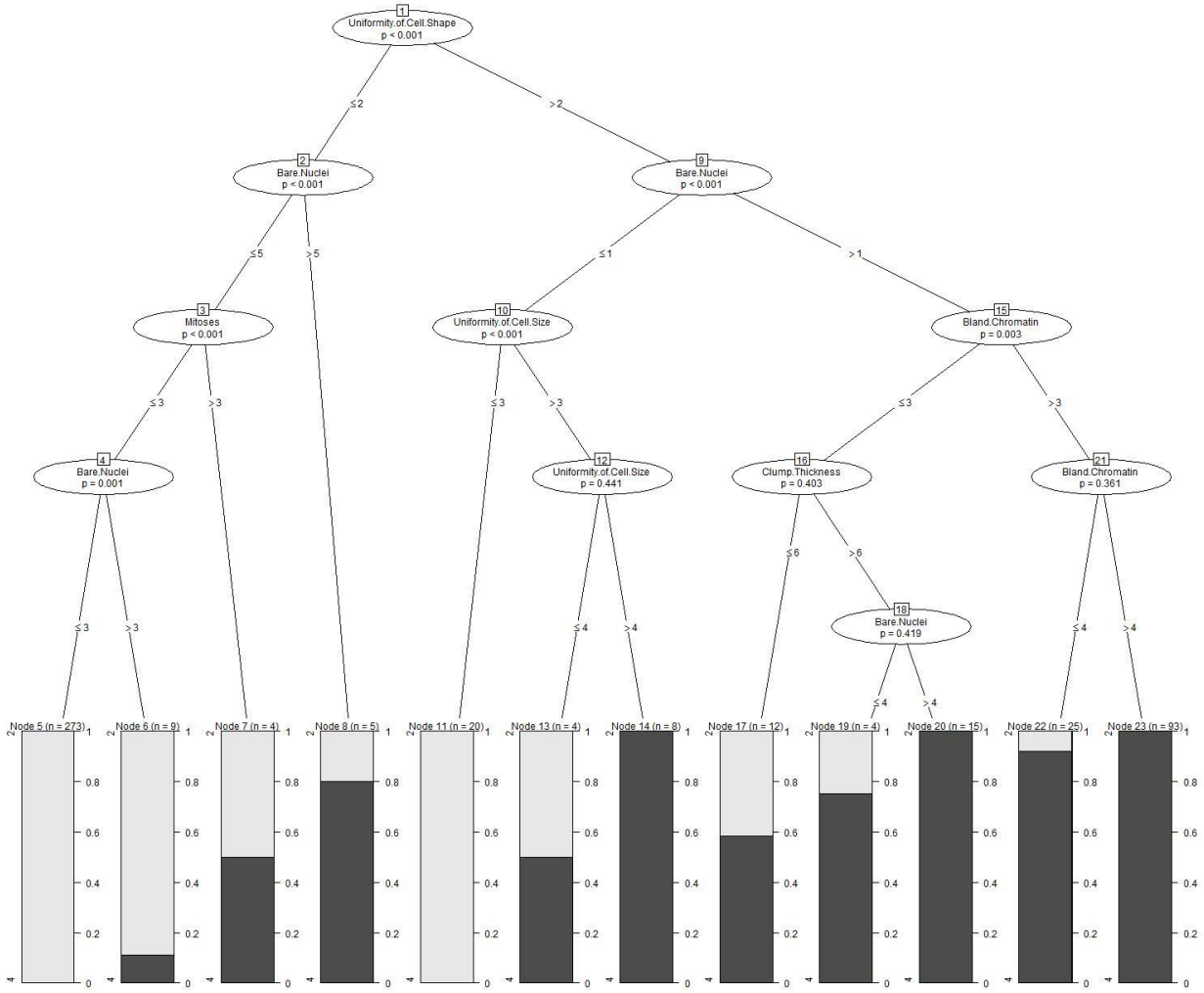
(vii)method=single



(viii)Classification tree with default parameters



(ix)Classification tree with modified control tree parameters



Evaluation:

Task 2.4

The cluster obtained in 2.2, uses the default parameters for kmeans() and produces a cluster that is already colored with respect to the result of the clustering.

In task 2.3 we notice that most points are colored similar to task 2.2, excepting a few anomalies. We can thus conclude that the clusters represent the benign vs malignant classes

Task 2.6

The SSE values obtained for k=2,3,4 and 5 are respectively 19323.17, 16255.51, 15173.81, 13808.94

We notice that as the value of k increases the clusters become too close to each other and there are many small clusters. When k=2, SSE value is too high, which means that the distance between the points in a cluster is too much. On the other hand when k=5, the clusters are very close to each other.

Task2.8

On comparing the plots obtained in task 2.8, we can conclude that we don't need a new subtype of diseases, because this would mean a larger k value and the clusters would become too close to each other.

Task2.9

Yes, the data is sensitive to the agglomeration method used for clustering. The default agglomeration method is "complete".

Task3.2

The important variables are: Uniformity of cell size, bare nuclei, mitoses and uniformity of cell size. From the classification tree we observe that the split on uniformity of cell shape is good since its result is more homogeneous.

Task3.4

For the nearest neighbor classifier, we can see that the accuracy is the highest when k=5. We notice that the model is prone to noisy data and easily classifies new data.