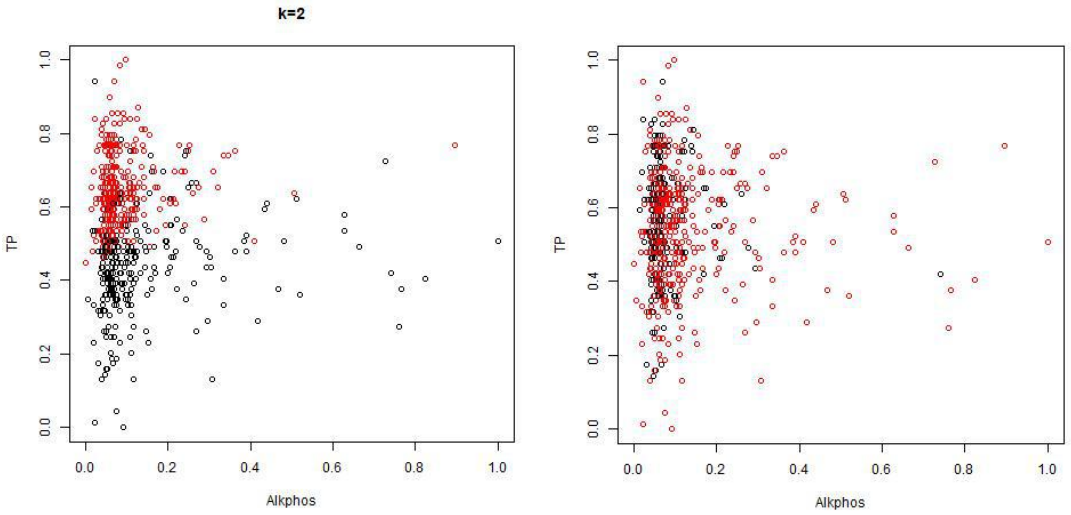
Data mining report

Student name: Yaxin Yu

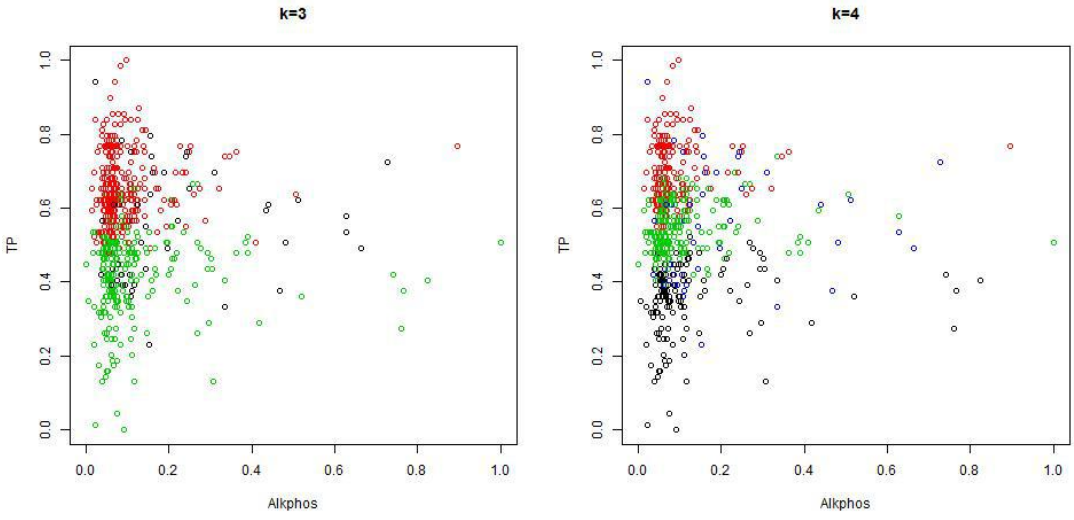
Student number:45120486

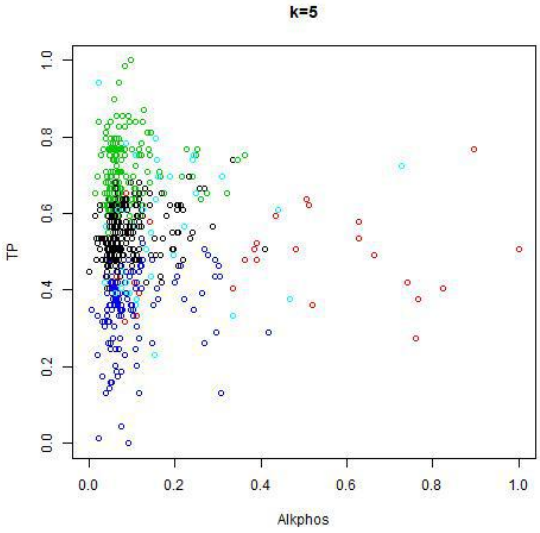
2.5 Compare the 2 plots obtained in the previous two tasks (tasks 2.3. and 2.4.) – do the clusters visually represent non-patient the patient vs classes?

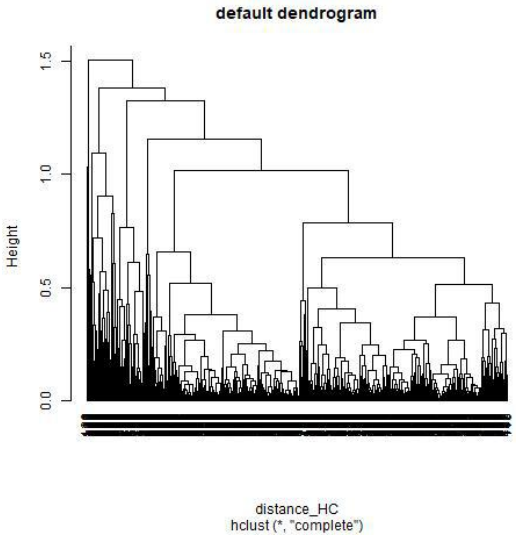


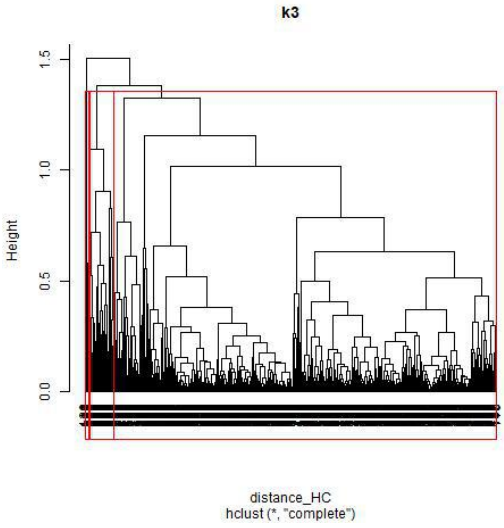
The first plot is to cluster the data into 2 clusters using K-Means algorithm and the second one is to plot another 2D plot with the same dimensionsform and the color of "Class" attribute. The first one with 2 clusters K-Means algorithm did a greater job to divide data into two groups since it has more obvious dividing line to show the difference of each clusters clearly compared with the right-hand side one. However, these cluters still can not represent non-patient vs patient classes.

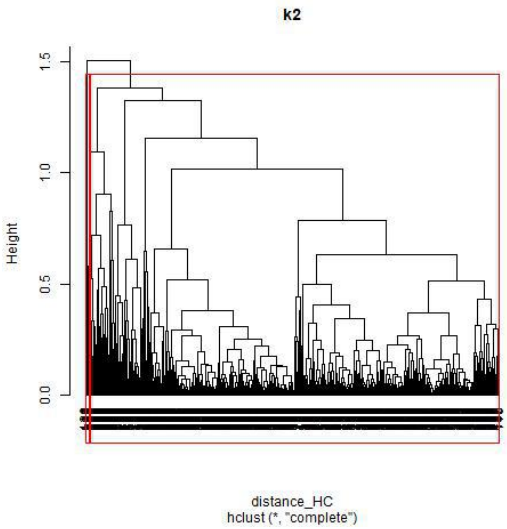
2.7 Compare the plots and Sum of Squared Error (SSE) obtained in the previous task and provide your comments on the quality of clustering.

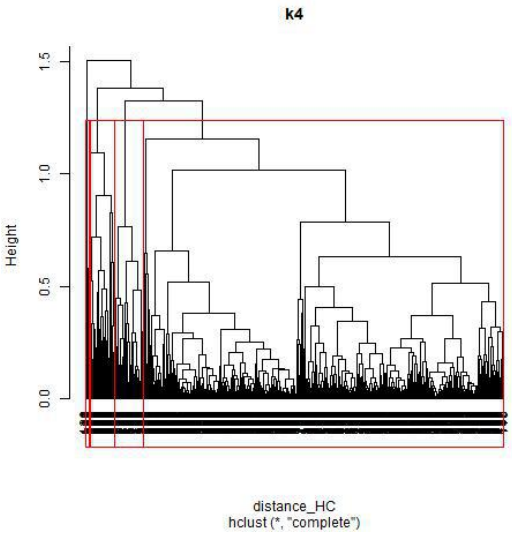


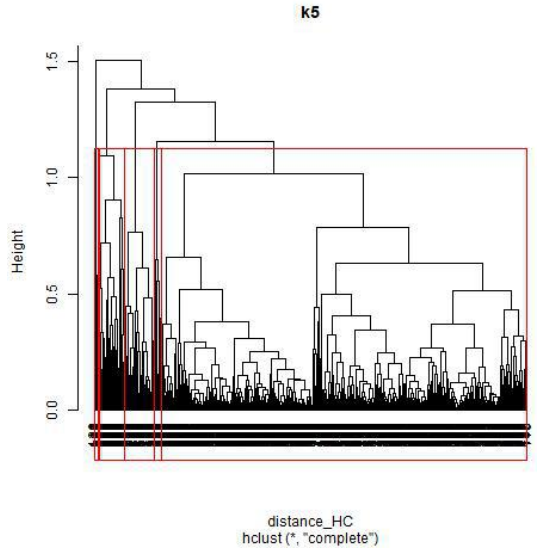

k=5
TP
Alkphos
0.0
0.2
0.4
0.6
0.8
1.0


default dendrogram
Height
distance_HC
hclust (*, "complete")
0.0
0.5
1.0
1.5


k3
Height
distance_HC
hclust (*, "complete")
0.0
0.5
1.0
1.5


k2
Height
distance_HC
hclust (*, "complete")
0.0
0.5
1.0
1.5


k4
Height
distance_HC
hclust (*, "complete")
0.0
0.5
1.0
1.5



(1) SSE：

```
a <- "when k=2,SSE="
b <- "when k=3,SSE="
c <- "when k=4,SSE="
d <- "when k=5,SSE="
print(paste(a,ILPD_clusters$tot.withinss,sep=""))
"when  k=2,SSE=43.4730095656656"
print(paste(b,ILPD_clusters_3$tot.withinss,sep=""))
"when  k=3,SSE=31.967152118246"
print(paste(c,ILPD_clusters_4$tot.withinss,sep=""))
"when  k=4,SSE=27.2021755913873"
print(paste(d,ILPD_clusters_5$tot.withinss,sep=""))
"when  k=5,SSE=23.4518872567465"
```

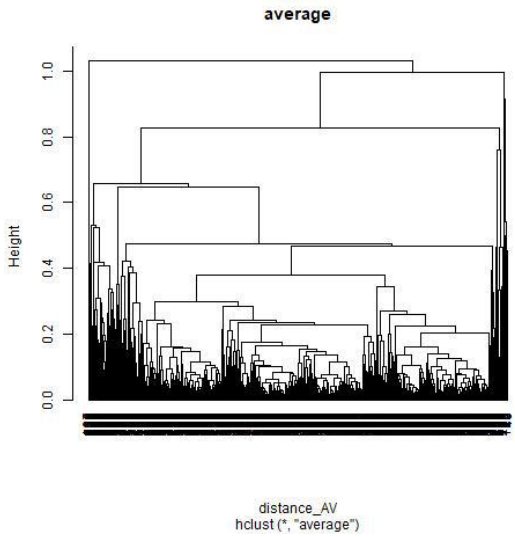
(2)

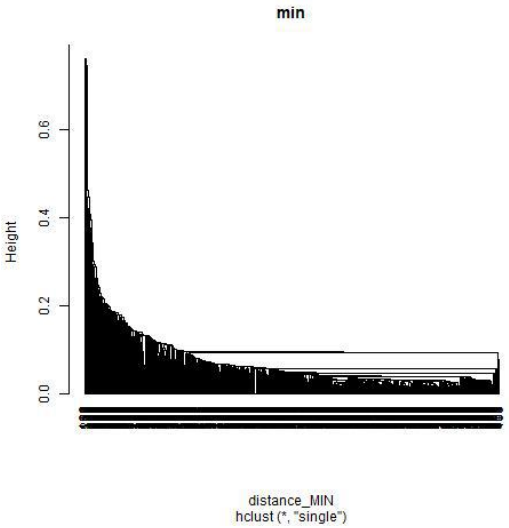
For SSE results, we can see that SSE is declining when K is incresing and it satifys the nornal situation. Besides, As k incresing, the decreasing speed of SSE is getting slower, which means that k=5 is greater than others.

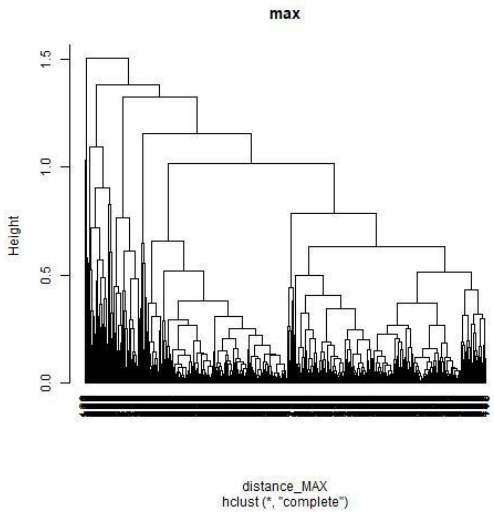
It is soft clusters since some items can belong to multiple groups. Besides, although this kind of method divide the entire group into seperate clusters, there is still no clear distinction between patient and non-patients.

2.9 Compare the plots obtained in the tasks 2.3., 2.4., 2.6. and 2.8. and provide your observations on the achieved clusters - should we have a new subtype of diseases?
(1) Yes.It can be seen from kmeans clusters that when TP is lower than 0.4, points are always belong to the same separate cluter,which indicates that we have new subtype of diseases.
(2) For plots in 2.8,it is easy to see that more than half subjects are always in one cluster with k changing, which also indicates that we have new subtype of diseases.

2.10 Try different agglomeration methods in hierarchical clustering. Plot the resulting dendrograms and provide your comments on the quality of clustering - is the data sensitive to the used agglomeration method? Based on your results, what do you think is the default agglomeration method used in Task 2.8.?

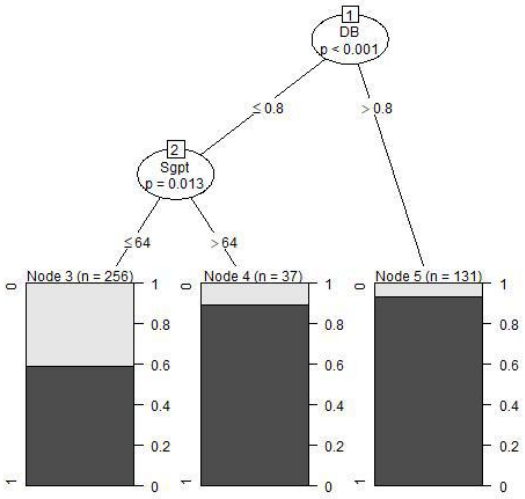






The data is more sensitive to the new agglomeration methods.
Default agglomeration method used in Task 2.8 is “MAX”since its dendrograms default setting is same as dendrograms with “MAX” setting.

3.2 Learn a classification tree from the training data using the default parameters of the function from the library. Plot that classification tree and provide your comments on its structure (e.g., what are the important/unimportant variables? Is there any knowledge we can infer from the tree representation that helps in differentiating between the classes?). Using the learned tree, predict the class labels of the test data. Calculate the accuracy, precision, and recall.



(1)ILPD_prediction <- predict(ILPD_ctree, newdata=testing_data[,-11])
confusionMatrix <- as.matrix(table(testing_data[,11],ILPD_prediction))
print(confusionMatrix)
accuracy_result <- sum(diag(confusionMatrix))/sum(confusionMatrix)
row_sum <- apply(confusionMatrix, 1, sum)
column_sum <- apply(confusionMatrix, 2, sum)
precision_result <- confusionMatrix[2,2]/column_sum
recall_result <- confusionMatrix[2,2]/row_sum
print(accuracy_result)
[1] 0.6918239
print(recall_result)
0 1
2.244898 1.000000
print(precision_result)
0 1
Inf 0.6918239

(2) Since no repeat labels in the decision tree, all variables are important.And from the defalut desicion tree,we can see that for each node(node3,nide4,node5), the proportion of class"1" is bigger than that of class"0",which means that wh atever the inputed data is , the output will always be "1". Hence, the class la bel of the test data is 1(patient).

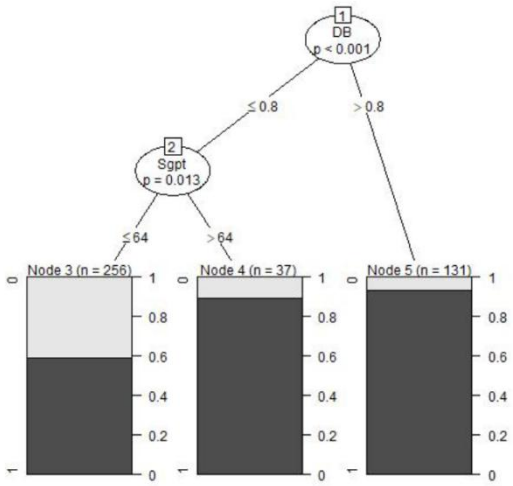
3.3 Try building your classification tree again via the function but using parameters that are different from the default settings. Can you achieve better accuracy or more meaningful representation by tuning some parameters? (Note that in the function from library, you can modifiy parameters. Execute form RStudio Console for the detailed documentation.)

Attempt:

(1)First attempt:

```
ILPD_control <- ctree_control(teststat = c("quad", "max"),
                              testtype = c("Bonferroni", "MonteCarlo", "Univariate", "Teststatistic"),
                               mincriterion = 0.8, minsplit = 30, minbucket = 5,
                              nresample = 9999, stump = FALSE, maxdepth = 4)
```

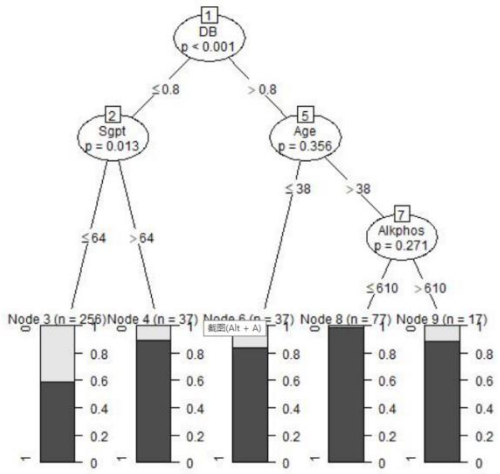
```
print(accuracy_result2)
[1] 0.6918239
 print(recall_result2)
0           1
2.244898 1.000000
print(precision_result2)
0           1
Inf 0.6918239
```



(2)Second attempt:

```
ILPD_control <- ctree_control(teststat = c("quad", "max"),
                              testtype = c("Bonferroni", "MonteCarlo", "Univariate", "Teststatistic"),
                               mincriterion = 0.6, minsplit = 30, minbucket = 7,
                              nresample = 9999, stump = FALSE, maxdepth = 4)
```

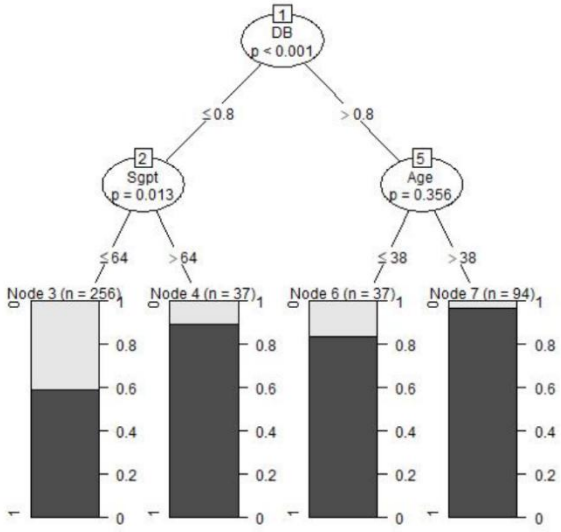
print(accuracy_result2)
[1]0.6918239
print(recall_result2)
0 1
2.244898 1.000000
print(precision_result2)
0 1
Inf 0.6918239



(3)Third attempt:

```
ILPD_control <- ctree_control(teststat = c("quad", "max"),
                              testtype = c("Bonferroni", "MonteCarlo", "Univariate", "Teststatistic"),
                               mincriterion = 0.6, minsplit = 30, minbucket = 7,
                              nresample = 9999, stump = FALSE, maxdepth = 2)
```

print(accuracy_result2)
[1]0.6918239
print(recall_result2)
0 1
2.244898 1.000000
print(precision_result2)
0 1
Inf 0.6918239



(4)Final result:

```
ILPD_control <- ctree_control(teststat = c("quad", "max"),
                              testtype = c("Bonferroni", "MonteCarlo", "Univariate", "Teststatistic"),
                              mincriterion = 0.6, minsplit = 30, minbucket = 5,
                              nresample = 9999, stump = FALSE, maxdepth = 4)
```

print(accuracy_result2)

[1]0.6918239

print(recall_result2)

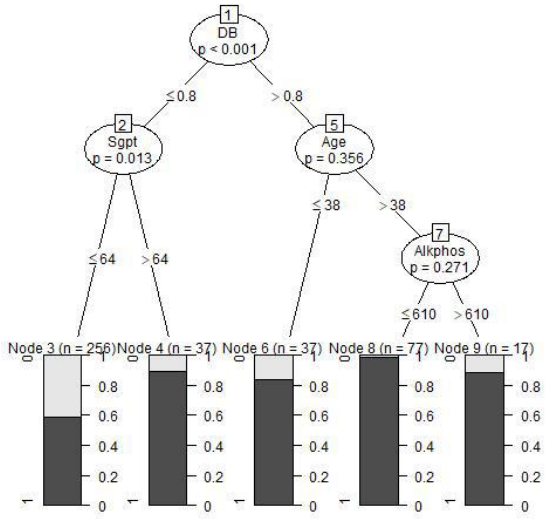
0 1

2.244898 1.000000

print(precision_result2)

0 1

Inf 0.6918239



After several attempts, it is easy to find that the maximum depth of the decision tree is 3, and when the mincriterion=0.6, the decision tree is more distinct. The final decision

tree is a better structured classification. But no matter how many times I change the data of ctree_control, there is no obvious change in the decision tree's accuracy, precision and recall.

3.4 Apply K-NN classification to predict the labels in the test subset and calculate the accuracy, precision and recall. Particularly, try different values of K (e.g. K = 1, 2, 3, 4, 5), and report your observations on the achieved classification

(1)k=1

```
print(knn_accuracy)
[1]  0.6289308
print(knn_precision_result)
0             1
2.2631579  0.7107438
print(knn_recall_result)
0             1
1.7551020  0.7818182
```

(2)k=2

```
print(knn_accuracy2)
[1]  0.6540881
print(knn_precision_result2)
0             1
2.8437500  0.7165354
print(knn_recall_result2)
0             1
1.8571429  0.8272727
```

(3)k=3

```
print(knn_accuracy3)
[1]  0.6603774
print(knn_precision_result3)
0             1
3.2068966  0.7153846
print(knn_recall_result3)
0             1
1.8979592  0.8454545
```

(4)k=4

```
print(knn_accuracy4)
[1]  0.6666667
print(knn_precision_result4)
0             1
```

3.1000000 0.7209302

```
print(knn_recall_result4)
```

0 1

1.8979592 0.8454545

(5) k=5

```
print(knn_accuracy5)
```

[1] 0.6603774

```
print(knn_precision_result5)
```

0 1

3.4814815 0.7121212

```
print(knn_recall_result5)
```

0 1

1.9183673 0.8545455

Observations:

As shown in the above datas,when k<=4, the accuracy is incresing. And when k>5, the accuracy is delicing.Hence, the best k is 4 and labels in the test subset is “1”(patient).