STAT 318 Assignment 3

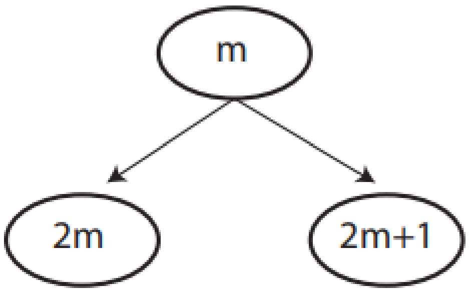
Student Name : Xiao Meng

Student ID : 88211337

1.

(a)

As known,



The best split should have maximal reduction in impurity

$$\Delta(I) = I(m) - \frac{N_{2m}}{N_m} I(2m) - \frac{N_{2m+1}}{N_m} I(2m+1)$$

For split with X_1, we can get the reduction in impurity,

$$\Delta(I) = \frac{1}{2} \times (1 - \frac{1}{2}) + \frac{1}{2}(1 - \frac{1}{2}) - \frac{3}{4} \times (\frac{1}{3} \times \frac{2}{3} + \frac{2}{3} \times \frac{1}{3}) - \frac{1}{4} \times 0 = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}$$

For split with X_2, we can get the reduction in impurity,

$$\Delta(I) = \frac{1}{2} - \frac{1}{2} \times 2 \times \frac{2}{5} \times \frac{3}{5} - \frac{1}{2} \times 2 \times \frac{3}{5} \times \frac{2}{5} = \frac{1}{50}$$

For split with X_3, we can get the reduction in impurity,

$$\Delta(I) = \frac{1}{2} - \frac{1}{2} \times 2 \times \frac{1}{2} \times \frac{1}{2} - \frac{1}{2} \times 2 \times \frac{1}{2} \times \frac{1}{2} = 0$$

Therefore, the best split with maximal $\Delta(I)$ is X_1.

(b)

(c)

As (b) shown, only the left daughter node is impure, to find the best split of left daughter node, calculate the impure reduction of X_2, X_3.

For split with X_2, we can get the reduction in impurity,

$$\Delta(I) = 2 \times \frac{1}{3} \times \frac{2}{3} - \frac{2}{5} \times 0 - \frac{3}{5} \times 2 \times \frac{4}{9} \times \frac{5}{9} = \frac{4}{27}$$

For split with X_3, we can get the reduction in impurity,

$$\Delta(I) = 2 \times \frac{1}{3} \times \frac{2}{3} - \frac{2}{3} \times 2 \times \frac{1}{2} \times \frac{1}{2} - \frac{1}{3} \times 0 = \frac{1}{9}$$

Therefore the best split of the left daughter node with the maximal $\Delta(I)$ is X_2

(d)

As the tree in (c) shown, the right daughter node of node $X_2 < 0.5$ should be classifed as High. So there are 20 observations to be misclassified.

(e)

After splitting with X_3, for its left daughter node we can get the reduction impurity:

For X_1:

$$\Delta(I) = 2 \times \frac{1}{2} \times \frac{1}{2} - 2 \times \frac{1}{2} \times \frac{1}{2} = 0$$

For X_2 :

$$\Delta(I) = 2 \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{2}$$

With the maximal $\Delta(I)$, the best split for the left daughter node is X_2.

For the right daughter node we can get:

For X_1 :

$$\Delta(I) = 2 \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{2}$$

For X_2 :

$$\Delta(I) = 2 \times \frac{1}{2} \times \frac{1}{2} - 2 \times \frac{1}{2} \times 2 \times \frac{4}{5} \times \frac{1}{5} = \frac{9}{50}$$

With the maximal $\Delta(I)$, the best split for the left daughter node is X_1.

(f)

From (e) we can get a tree with no misclassifed while we got 20 misclassied in tree of (c), which means tree in (e) is more accurancy than tree in (c). Then, we can understand the greedy natural in CART is that each iteration of split just choose a locally optimise just like choosing the first best split X_1 in (c), rather than globally which could make a better tree in some future steps.

2

(a)

Fit and plot a regression tree to the training set.

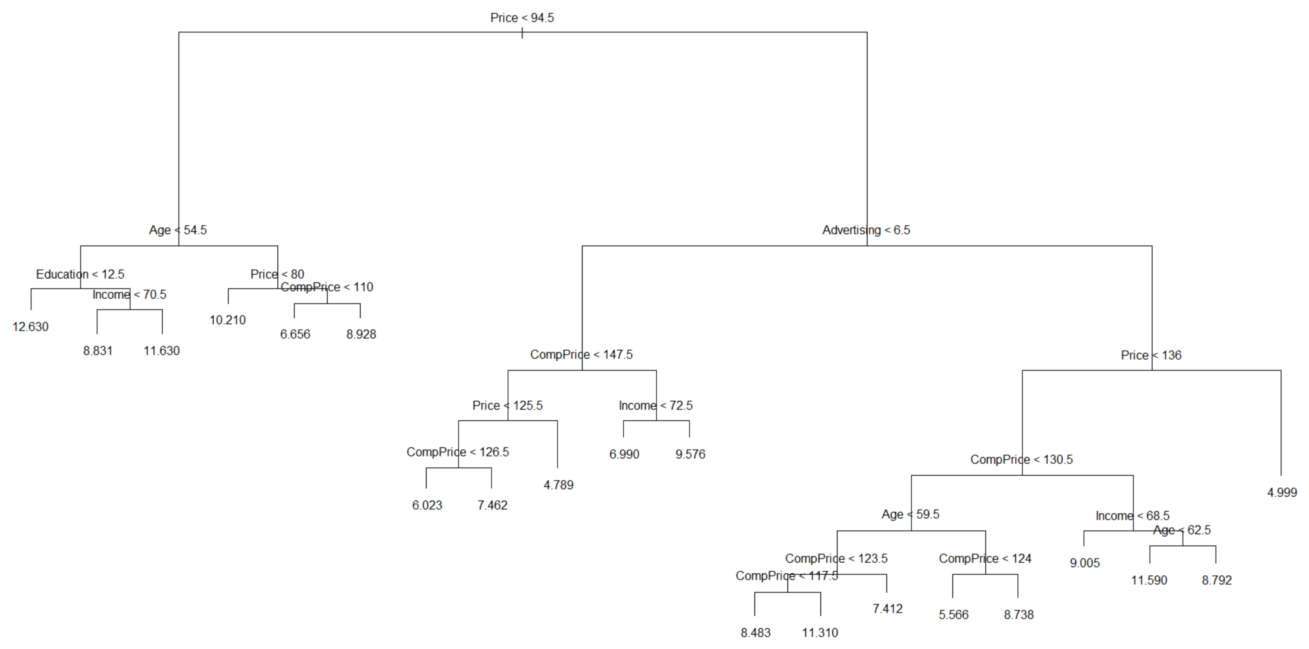


Figure 2.1 : regression tree with the traing set

For this regression tree, the training MSE is 3.0411(4dp) and the testing MSE is 6.1262(4dp).

Based on the regression tree we can see, there are 20 terminal nodes and the residual mean deviance is 3.275. For all 9 variables, there are only 6 variables to be used to create the regression tree. The Price is the most significant factor, besides Age and Advertising also played important roles for sale.

(b)

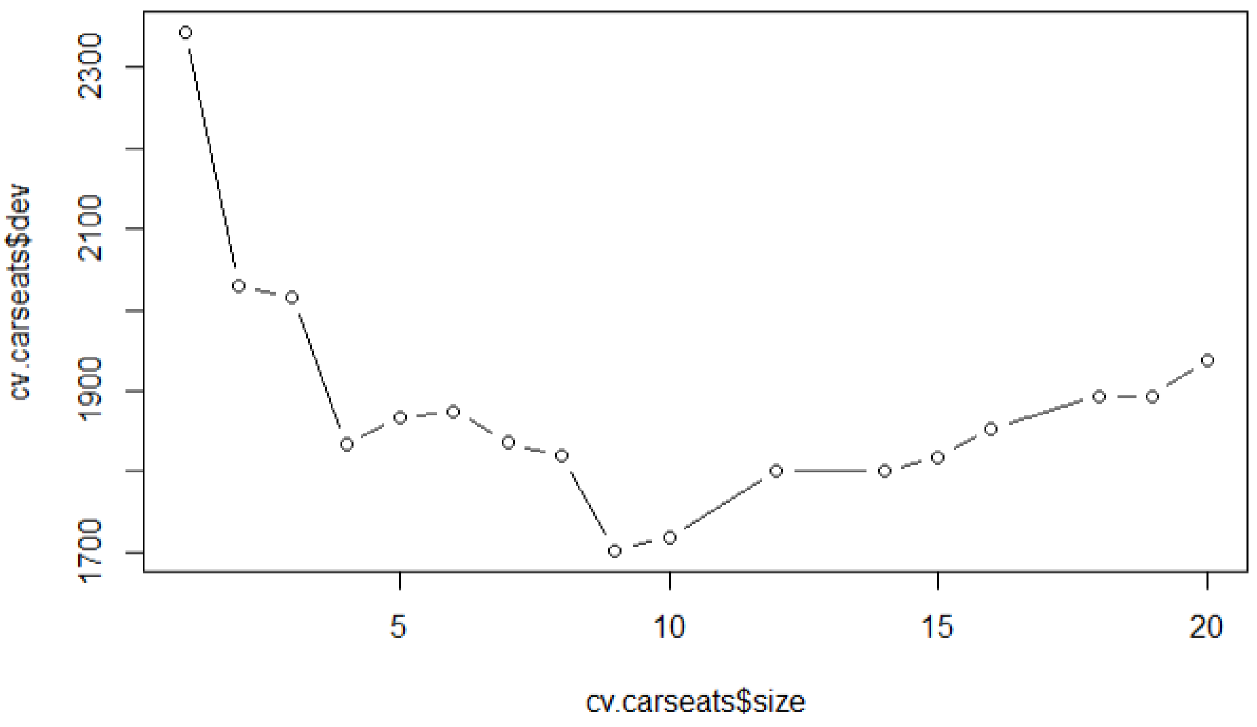


Figure 2.2 : Error rate of each size of tree

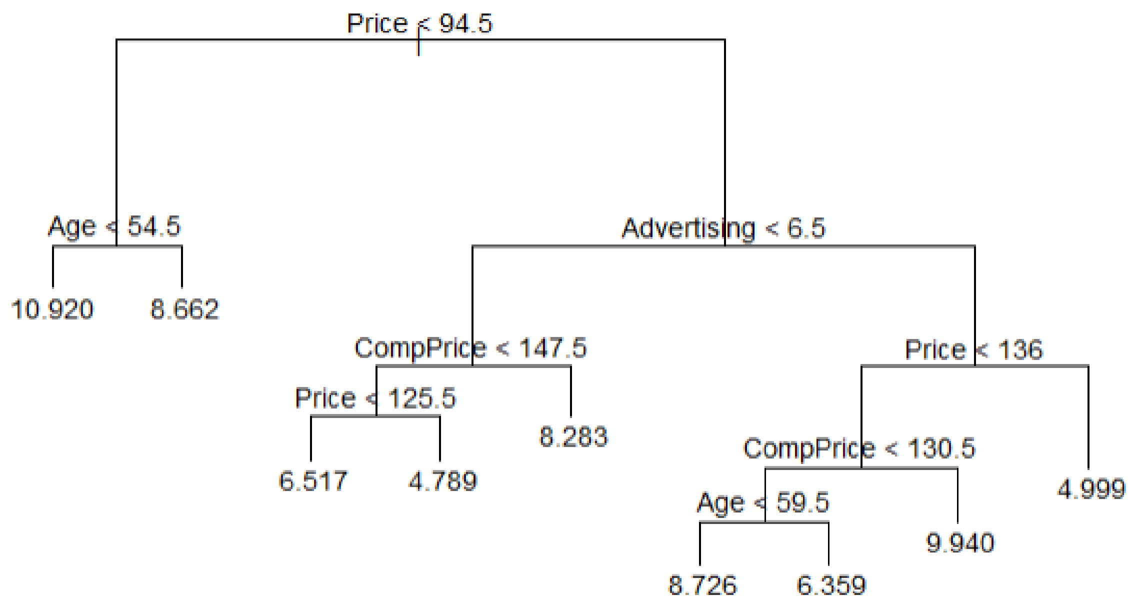


Figure 2.3 : regression tree with 9 nodes

From figure 2.2 we can see, compare with 20 terminal nodes, the error rate of many nodes below 20 is lower and regreesion tree is simpier. So pruning improve the tree's performance. When the number of nodes is 9, the regression tree has the lowest cross-validation error rate. Also, this regression tree with 9 nodes is not too complex and all simpier trees with less nodes have higher error rate. Therefore, I chosed 9 nodes to do the pruning.

After pruning to 9 nodes, the test MSE 5.6903(4dp) which is lower than 6.1262(4dp) of regression tree with 20 nodes in (a). That means pruning improve the test MSE.

(c)

When fit a bagged regression tree with all predictor in this problem, which means parameter mtry = 9, and fit a random forrest with mtry = p/3 = 3, we can get the test and training MSEs shown as table below:

	test MSE	training MSE
bagged regression tree	4.5547	0.8447
random forest	4.9128	1.045

From this table, we can know both test and traning MSEs of bagged regression tree are lower, which means bagged regression tree performed better than random forest in this problem. While one feature of random forest is known as decorrelating, therefore decorrelating trees does not have an effective strategy for this problem.

(d)

Fit the boosted regression tree with the training data for tree depths from 1 to 5, shrinkage parameters in (0.1, 0.01, 0.001) and number of trees in (1000, 2500, 5000), the best tree with minimal test MSEs has tree depth of 1, shrinkage parameter of 0.01 and 1000 trees. The traning and test MSE for the best tree is 3.5571 and 4.1067.

(e)

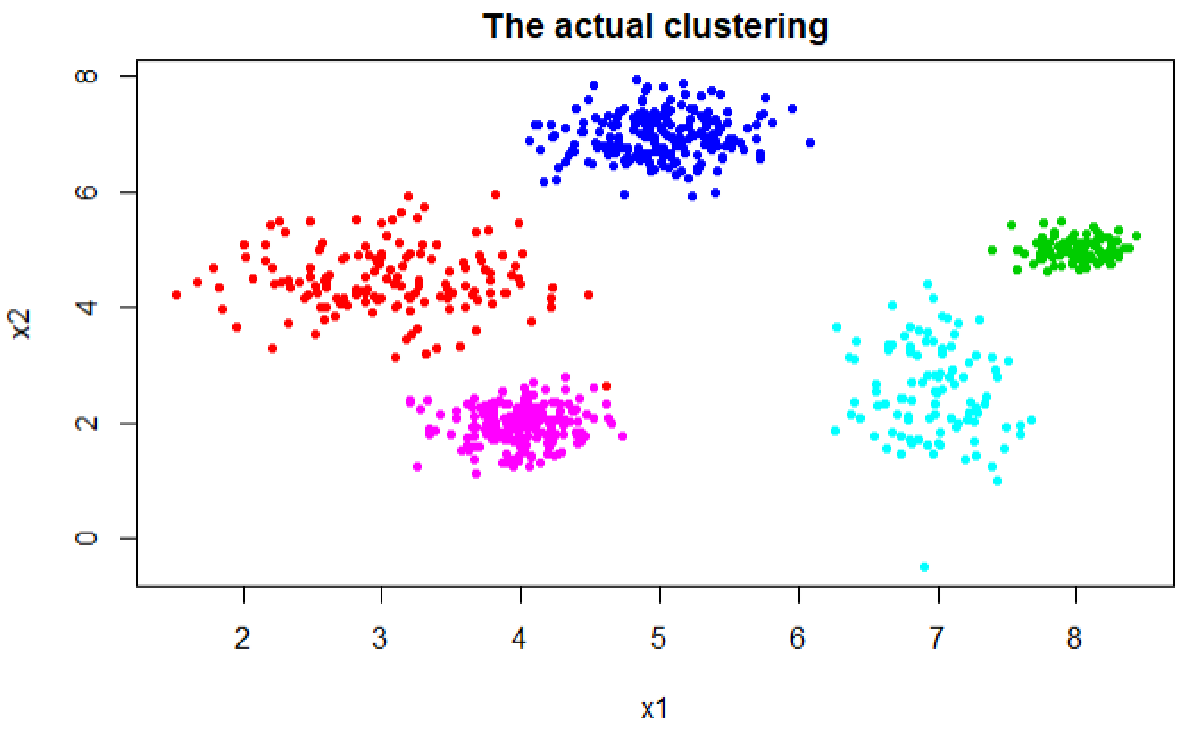
With the test MSE of each regression tree, the boosted regression tree with lowest test MSE performed best in this problem.

As the importance shown, the most important predictors are Price, CompPrice, Advertising, and age.

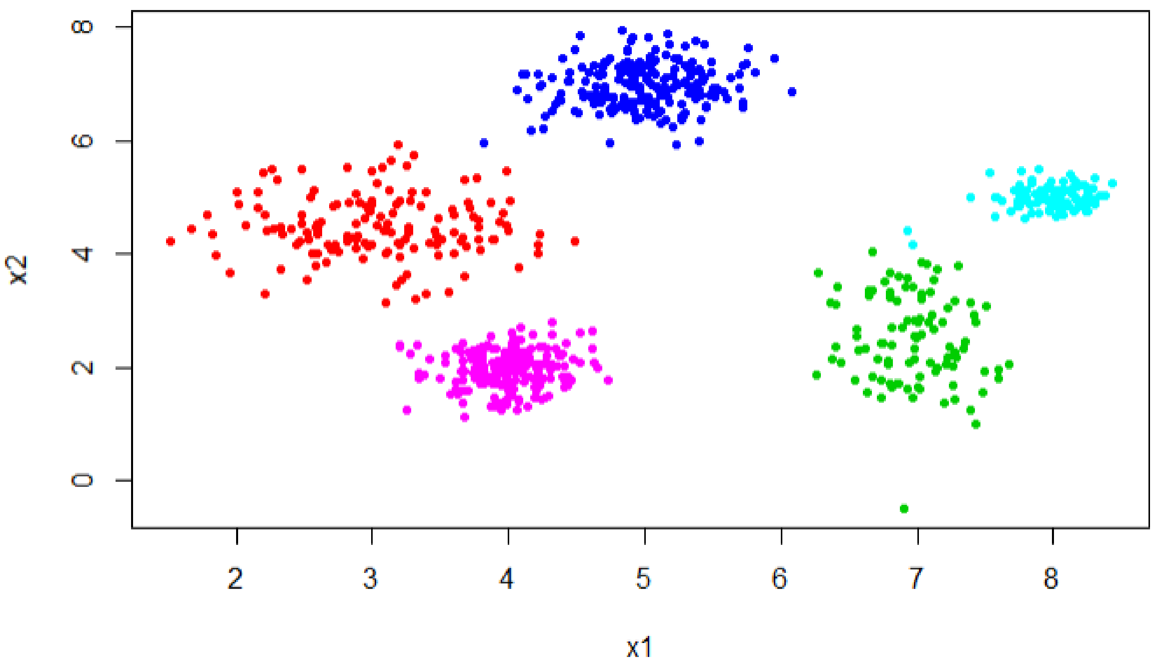
	var <fctr>	rel.inf <dbl>
Price	Price	42.90767396
CompPrice	CompPrice	16.48200129
Advertising	Advertising	14.89501501
Age	Age	14.37095979
Income	Income	8.93440890
Education	Education	1.63060259
Population	Population	0.70124045
Urban	Urban	0.04750439
US	US	0.03059361

3.

(a)



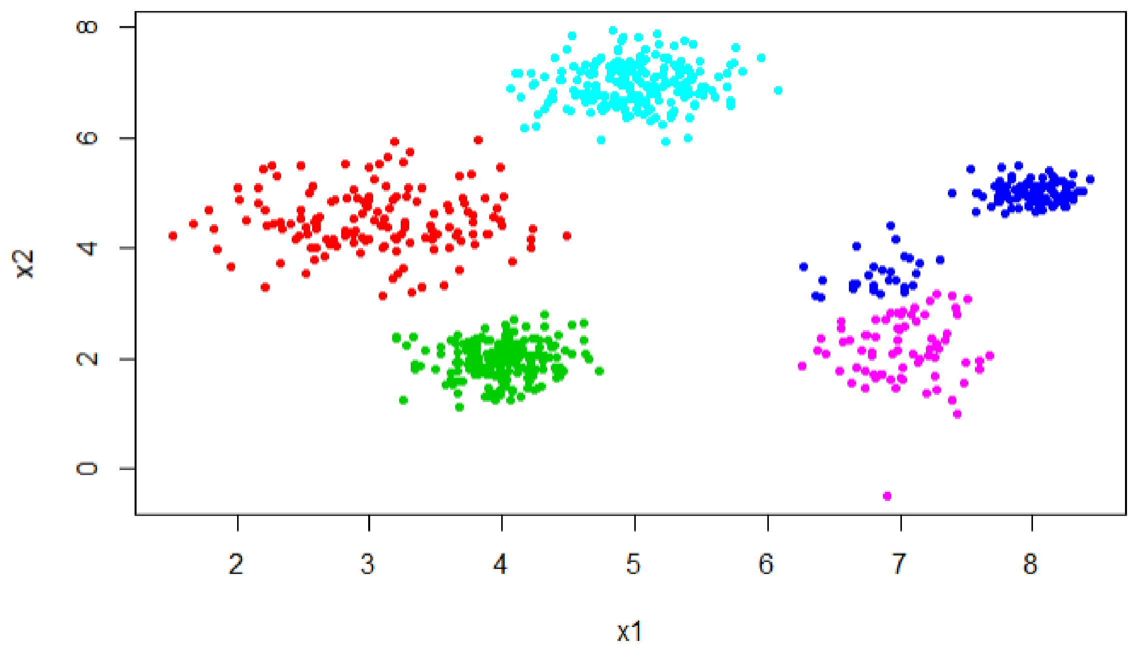
K-Means Clustering Result with K = 5



```
km1.clusters   1   2   3   4   5
           1 148   0   0   0   0
           2   0 100   0   2   0
           3   1   0 200   0   0
           4   1   0   0   0 200
           5   0   0   0  98   0
```

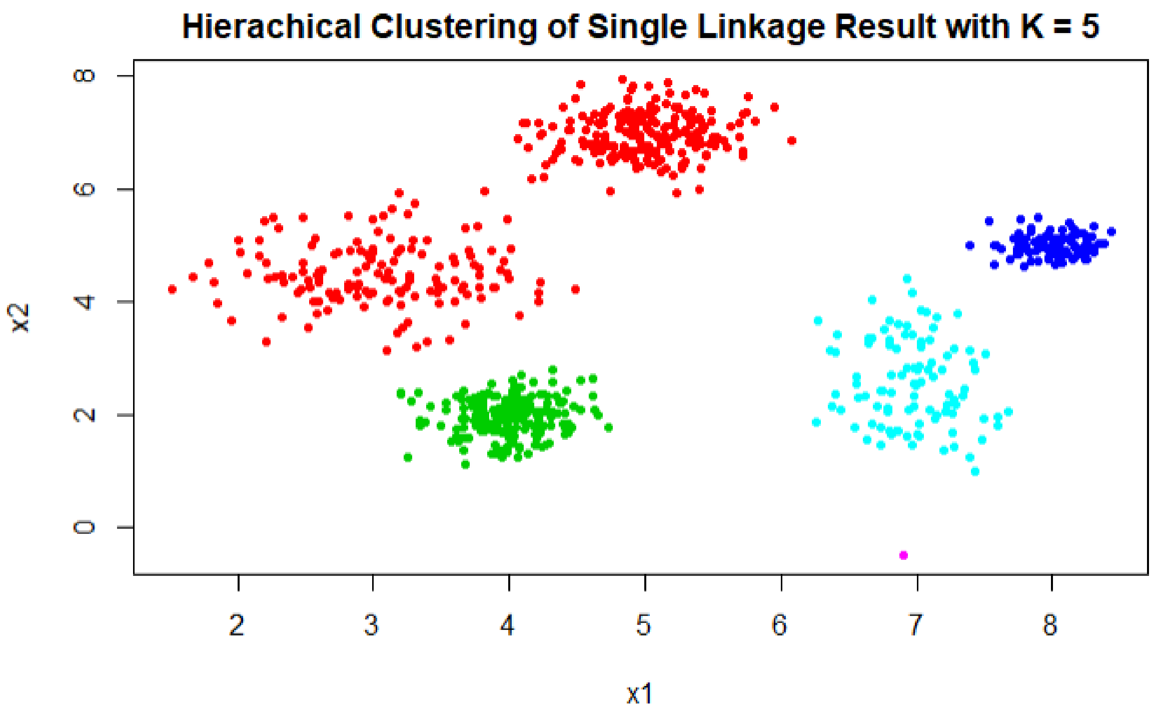
Shown as the table, compare with the actual cluster labels, cluster 1 and 4 are identical to cluster 1 and 5 in k-means. Meanwhile the other clusters are almost the same. In total, there are only 4 points assigned to other clusters by k-means clustering. The error rate of k-means in this data set is 4/750 = 0.53%.

Hierachical Clustering of Complete Linkage Result with K = 5

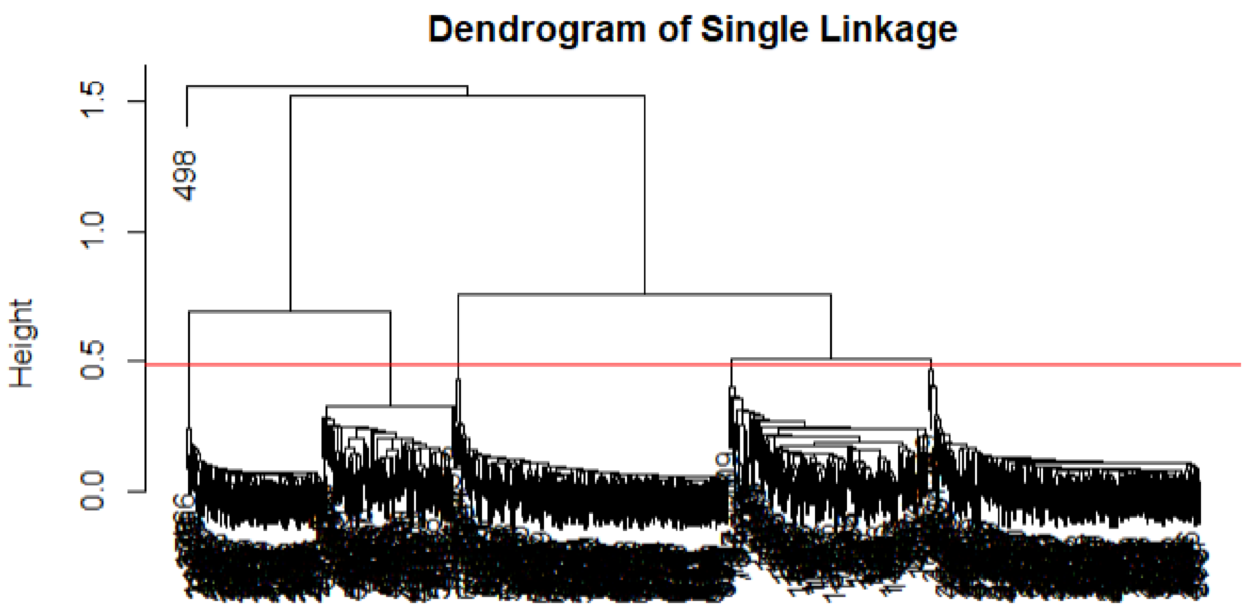


```
cluster1.complete   1   2   3   4   5
                1 149   0   0   0   0
                2   1   0   0   0 200
                3   0 100   0  27   0
                4   0   0 200   0   0
                5   0   0   0  73   0
```

Using Hierachical Clustering with Complete Linkage for this data set, observations in cluster 2, 3, 5 are significantly assigned as three individual clusters correctly. However, an observation is depart from cluster 1 to cluster 2 in method of complete linkage. Futhermore, this method of clustering assigned 27 observations of cluster 4 to cluster 3. The error rate of Complete Linkage in this data set is 28/750 = 3.73%.



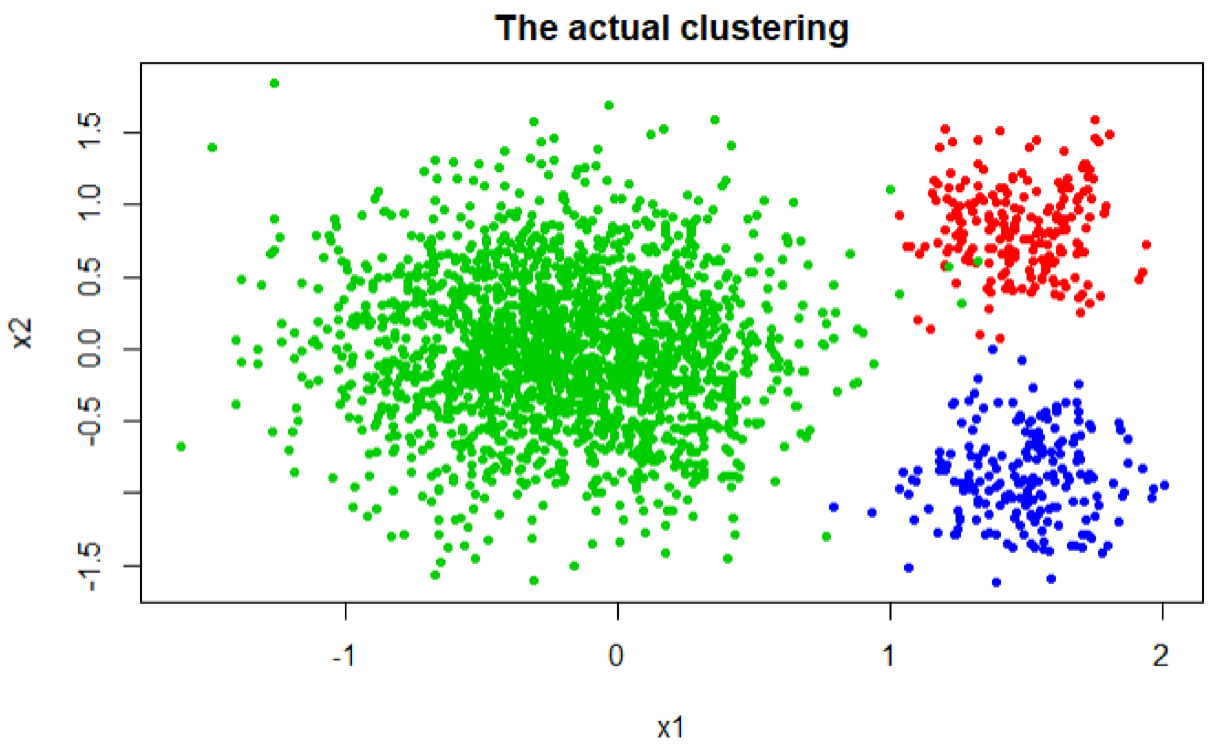
```
cluster1.single   1   2   3   4   5
              1 149   0 200   0   0
              2   1   0   0   0 200
              3   0 100   0   0   0
              4   0   0   0  99   0
              5   0   0   0   1   0
```

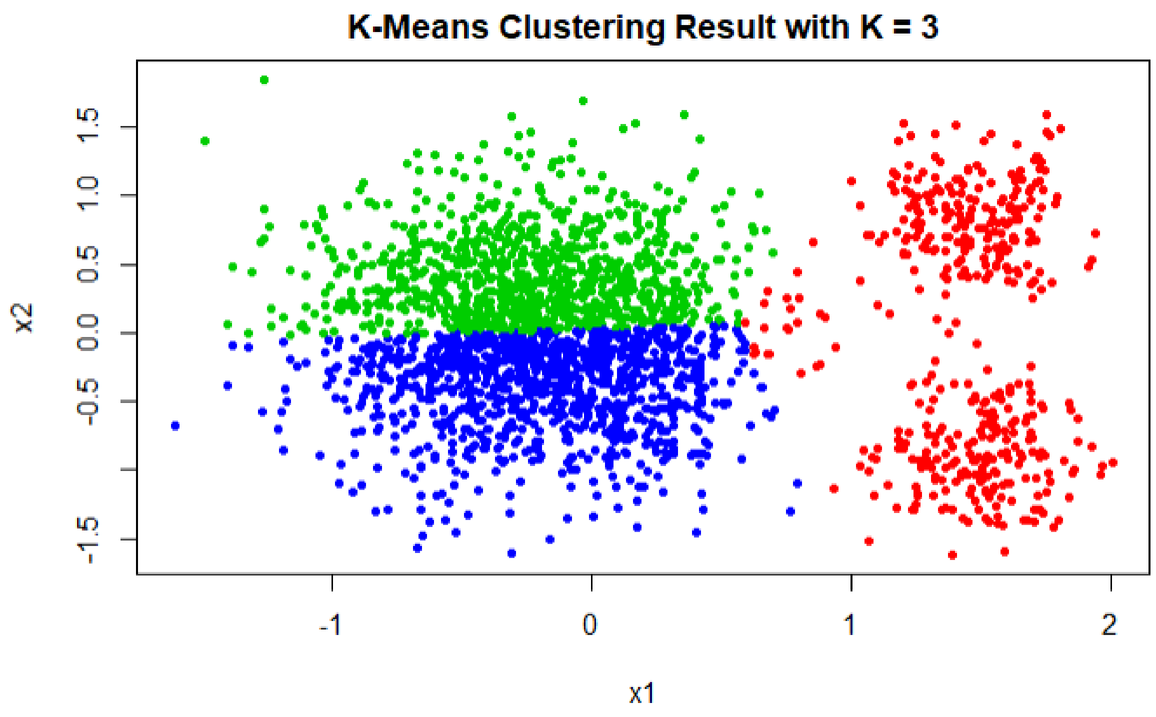


For Hierachical Clustering with method Single Linkage, we can see the actual cluster 2, 4, 5 are still almost assigned to individual clusters. But because of the algorithm of Single Linkage fuses single observation at a time, algorithm assigns only one observation into cluster 5 which is far from each other clusters. From Dendrogram we can see, if exclude that observation, this algorithm still can makes 5 clusters. While with a chain between cluster 1 and 3, this algorithm merges these two clusters into one. The error rate is about 202/750 = 26.93%.

After performing these three motheds clustering we can get, with the minimal error rate of clusters, K-means could give the best performance for this data set. While Hierachical Clustering with Complete Linkage performed better than Single Linkage.

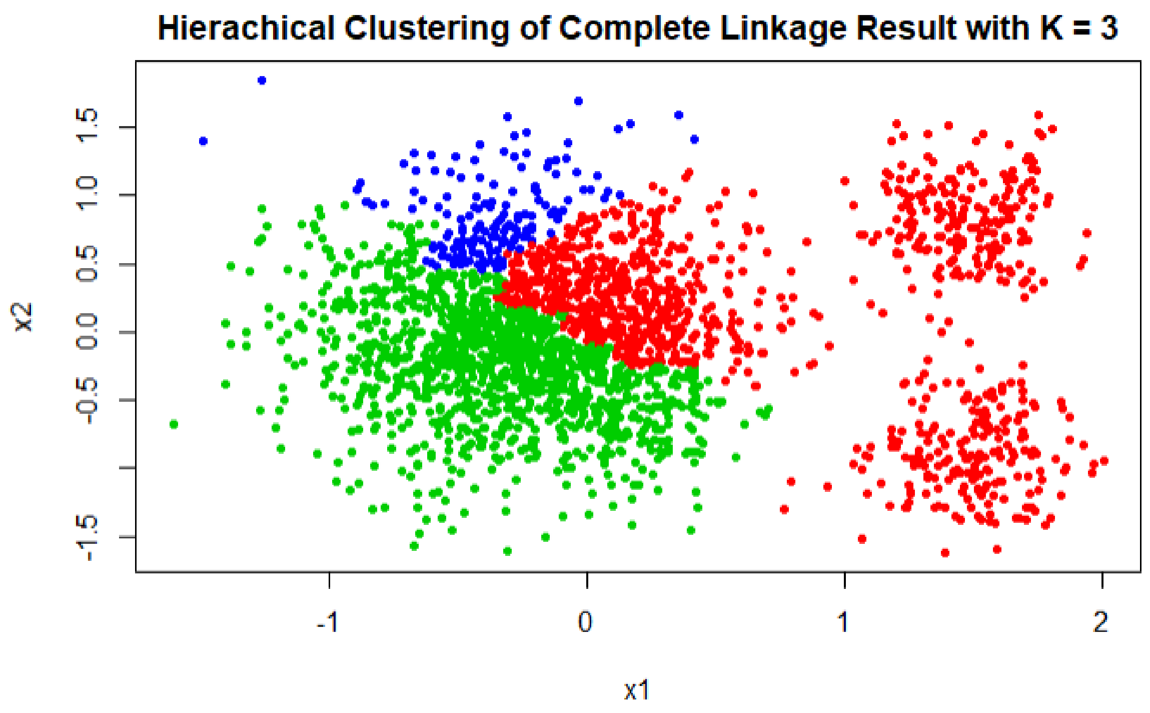
(b)





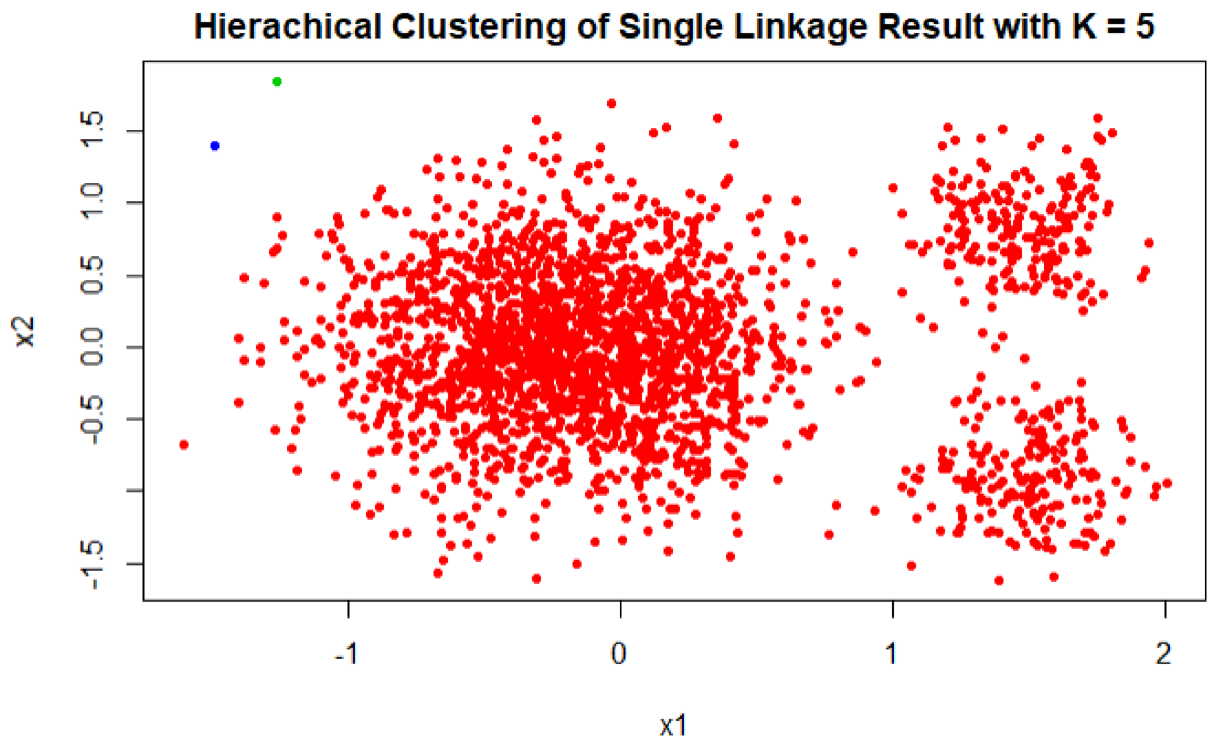
```
km2.clusters    1    2    3
           1  200   28  199
           2    0  935    0
           3    0 1037    1
```

Compare with the actual cluster labels, almost all of observations of actual cluster 1 and 3 are assigned to one cluster in k-means. The actual cluster 2 is divided into 3 clusters: small parts in cluster 1 in k-means, almost 47% of the rest are assigned to cluster 2 and other parts are into cluster 3.

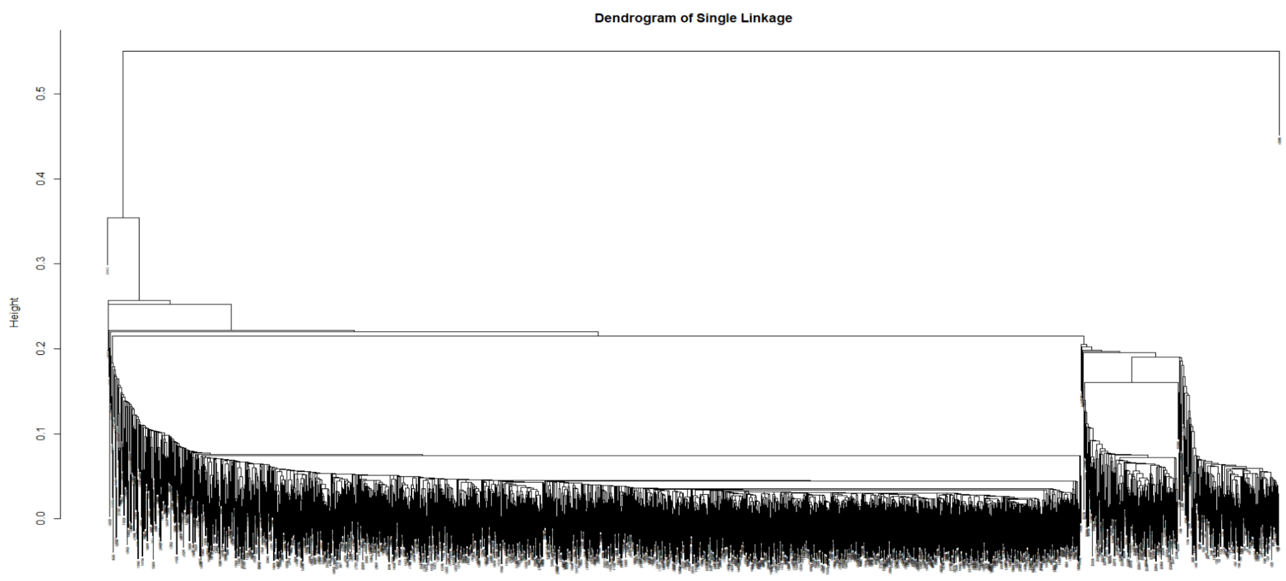


```
cluster2.complete    1    2    3
                1  200  609  200
                2    0 1233    0
                3    0  158    0
```

The Hierachical Clustering of Complete Linkage algorithm, the actual cluster 1 and 3 are still assigned into one cluster. For actual cluster 2, near 30% assigns to cluster 1, 7.5% observations are assigned to cluster, all the rest are assigned to cluster2.



```
cluster2.single    1    2    3
              1  200 1998  200
              2    0    1    0
              3    0    1    0
```



In this data set, Hierachical Clustering with Single Linkage almost lose its function. As the chain among every cluster, except for 2 observations which are far from the others, all the observations are assigned to one cluster. From the Dendrogram we can see, even through exclude the unusual observations, it still has many tailing clusters

Compared with these three methods in this data set, we can get Hierachical clustering with complete linkage performed the best, which more observations in the actual cluster 2 are assigned as an individual cluster while the actual cluster 1 and 3 are both assigned to one cluster in these three algorithms. Meanwhile k-mean clustering performed a little worse. With a trailing clusters in algorithm of single linkage, single linkage method played the worst in this data set.

Appendix

```r
#======================
#2 fit regression tree
#======================

#(a)Fit a regression tree to the training set.
library(tree)
cartrain = read.csv("E:/文档/UC/318/Assignment/A3/carTrain.csv")
cartest = read.csv("E:/文档/UC/318/Assignment/A3/carTest.csv")
tree.carseats = tree(Sales~., cartrain)
summary(tree.carseats)

#plot the tree
plot(tree.carseats)
text(tree.carseats, pretty = 0)

#MSE of training and test data
```

```r
pred.train = predict(tree.carseats, cartrain)
plot(pred.train, cartrain$Sales)
abline(0, 1)
mean((pred.train - cartrain$Sales)^2)

pred.test = predict(tree.carseats, cartest)
plot(pred.test, cartest$Sales)
abline(0, 1)
mean((pred.test - cartest$Sales)^2)

#(b) Pruning
#Find the best depth of tree
cv.carseats = cv.tree(tree.carseats)
plot(cv.carseats$size, cv.carseats$dev, type = "b", xlab = "Tree Size", ylab =
"Mean Squared Error")

#Pruning with the best depth
tree.carseats.prune = prune.tree(tree.carseats, best = 9)
plot(tree.carseats.prune)
text(tree.carseats.prune, pretty = 0, cex = 0.9)
pred.test.prune = predict(tree.carseats.prune, cartest)
mean((pred.test.prune - cartest$Sales) ^ 2)

#(c) Fit a bagged regression tree and a random forest
#fit a bagged regression tree(Using random forest with the number of all the
#predict variables : 9)
library(randomForest)
bag.carseats = randomForest(Sales~., cartrain, mtry = 9, importance = TRUE)
bag.carseats

#test and training MSE for bagged
pred.train.bag = predict(bag.carseats, cartrain)
mean((pred.train.bag - cartrain$Sales)^2)
pred.test.bag = predict(bag.carseats, cartest)
mean((pred.test.bag - cartest$Sales)^2)

#fit a random forest with p/3 = 3 variables
rf.carseats = randomForest(Sales~., cartrain, mtry = 3, importance = TRUE)
rf.carseats

#test and traning MSEs for random forest
pred.train.rf = predict(rf.carseats, cartrain)
mean((pred.train.rf - cartrain$Sales)^2)
pred.test.rf = predict(rf.carseats, cartest)
mean((pred.test.rf - cartest$Sales)^2)

#(d)Fit a boosted regression tree with training data and calculate the test and
traning MSEs
#fit a boosted regression tree with different depth, number of trees and
shrinkages to find the best one(with smallest test MSE)
library(gbm)
mse.boost = c()
min_mse = 50
```

```r
for (n in c(1000, 2500, 5000)){
     for (d in seq(1:5)){
         for (s in c(0.1, 0.01, 0.001)){
             boost.carseats = gbm(Sales~., cartrain, distribution = "gaussian", n.trees = n, interaction.depth = d, shrinkage = s, verbose = F)
             pred.test.boost = predict(boost.carseats, cartest, n.trees = n)
             mse = mean((pred.test.boost - cartest$Sales)^2)
             mse.boost = c(mse.boost, mse)
             if (mse < min_mse){
                min_mse = mse
                min_par = c(s, d, n)
             }
         }
     }
}
options(scipen = 2)
min_mse
min_par

#calculate the test and training MSEs with depth = 1, number of tree = 1000, shrinkage = 0.01
boost.carseats.best = gbm(Sales~., cartrain, distribution = "gaussian", n.tree = 1000, interaction.depth = 1, shrinkage = 0.01, verbose = F)
pred.train.boost.best = predict(boost.carseats.best, cartrain, n.trees = 1000)
mean((pred.train.boost.best - cartrain$Sales)^2)
pred.test.boost.best = predict(boost.carseats.best, cartest, n.trees = 1000)
mean((pred.test.boost.best - cartest$Sales)^2)

#(e)the most important predictors in this problem
#With the smallest test MSE, boost regression tree model performed the best.
#Find the most important predictors in boost regression tree model
summary(boost.carseats.best)

#=======================
#3 Cluster
#=======================
#(a)k-means with k = 5 plot with different colours and show the total within-cluster sum of squares
data1 = read.csv("E:/文档/UC/318/Assignment/A3/A3data1.csv")
km1.out = kmeans(data1[1:2], 5, nstart = 50)
plot(data1[1:2], col = (km1.out$cluster + 1), main = "K-Means Clustering Result with K = 5", pch = 20, cex = 1)
km1.out$tot.withinss

#(b)hierarchical clustering with k = 5 plot with different colours with complete linkage
hc1.complete = hclust(dist(data1[1:2]), method = "complete")
cluster1.complete = cutree(hc1.complete, 5)
plot(data1[1:2], col = (cluster1.complete + 1), main = "Hierachical Clustering of Complete Linkage Result with K = 5", pch = 20)

#(b)hierarchical clustering with k = 5 plot with different colours with single linkage
```

```r
hc1.single = hclust(dist(data1[1:2]), method = "single")
cluster1.single = cutree(hc1.single, 5)
plot(data1[1:2], col = (cluster1.single + 1), main = "Hierachical Clustering of Complete Linkage Result with K = 5", pch = 20)

#(C)compare the clustering methods of complete and single with the actual cluster
km1.clusters = km1.out$cluster
table(km1.clusters, data1$Cluster)
mean(km1.clusters != data1$Cluster)

table(cluster1.complete, data1$Cluster)
mean(cluster1.complete != data1$Cluster)

table(cluster1.single, data1$Cluster)
mean(cluster1.single != data1$Cluster)

#Repeat (a) - (c) with K = 3
data2 = read.csv("E:/文档/UC/318/Assignment/A3/A3data2.csv")
km2.out = kmeans(data2[1:2], 3, nstart = 50)
plot(data2[1:2], col = (km2.out$cluster + 1), main = "K-Means Clustering Result with K = 3", pch = 20, cex = 1)

hc2.complete = hclust(dist(data2[1:2]), method = "complete")
cluster2.complete = cutree(hc2.complete, 3)
plot(data2[1:2], col = (cluster2.complete + 1), main = "Hierachical Clustering of Complete Linkage Result with K = 3", pch = 20)

hc2.single = hclust(dist(data2[1:2]), method = "single")
cluster2.single = cutree(hc2.single, 3)
plot(data2[1:2], col = (cluster2.single + 1), main = "Hierachical Clustering of Complete Linkage Result with K = 5", pch = 20)

km2.clusters = km2.out$cluster
table(km2.clusters, data2$Cluster)
mean(km2.out$cluster != data2$Cluster)

table(cluster2.complete, data2$Cluster)
mean(cluster2.complete != data2$Cluster)

table(cluster2.single, data2$Cluster)
mean(cluster2.single != data2$Cluster)


```