HW6_Q8-9

Q8

Q8a. Split the data set into a training set and a test set.

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.1.3
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.2
```

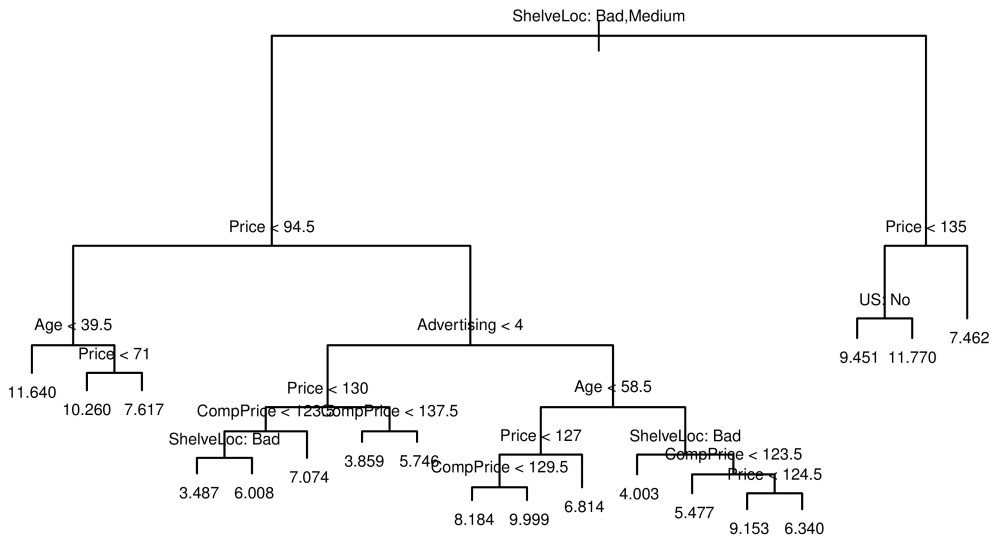
```
attach(Carseats)
# for same results every time
set.seed(1)

train <- sample(1:nrow(Carseats), nrow(Carseats)/2)
Cars.train <- Carseats[train , ] #200 obs 11 vars
Cars.test <- Carseats[-train , ] # ^       ^
```

Q8b. Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

```
# summary values
Cars.tree <- tree(Sales~.,Cars.train)
summary(Cars.tree)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Cars.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "CompPrice"
## [6] "US"
## Number of terminal nodes:  18
## Residual mean deviance:  2.167 = 394.3 / 182
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.88200 -0.88200 -0.08712  0.00000  0.89590  4.09900
```

```
# plot
plot(Cars.tree)
# cex = amt text should be scaled
text(Cars.tree, pretty = 0, cex = 0.5)
```



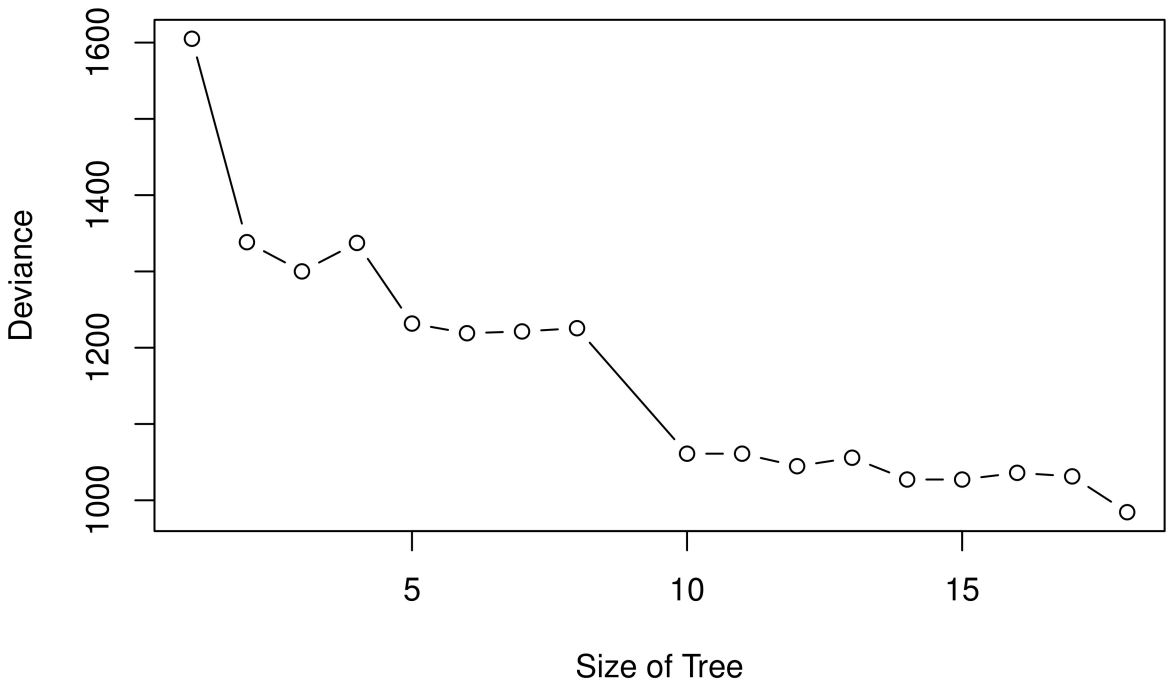

```
#MSE pred
pred.Cars = predict(Cars.tree, Cars.test)
mean((Cars.test$Sales - pred.Cars)^2)
```

```
## [1] 4.922039
```

The test MSE is 4.92. From the tree plot we can interpret that the variables ShelveLoc and Price must be the top two most important factors when predicting car seat sales from this dataset as they allow for the best split.

8c. Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```
set.seed(1)
cv.Cars <- cv.tree(Cars.tree)
plot(cv.Cars$size,cv.Cars$dev,xlab = "Size of Tree",ylab = "Deviance",type = "b")
```




```
# best size = 18 -- full tree
pruned.Cars.tree <- prune.tree(Cars.tree, best = 18)

test.pred <- predict(pruned.Cars.tree, Cars.test)
mean((test.pred - Cars.test$Sales)^2)
```

```
## [1] 4.922039
```

The optimal tree is the fully grown tree without any pruning because the plot shows that the test MSE is the same as in the previous part.

8d.Use the bagging approach in order to analyze this data

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.3
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
set.seed(1)
bag.car<-randomForest(Sales~.,Cars.train,importance=TRUE,mtry=10)
importance(bag.car) %>% as.data.frame() %>%arrange(desc(IncNodePurity))
```

```
##                %IncMSE IncNodePurity
## Price       56.3326252    502.903407
## ShelveLoc   48.8886689    380.032715
## CompPrice   24.8888481    170.182937
## Age         17.7275460    157.846774
## Advertising 12.7692401     97.164338
## Income       4.7121131     91.264880
## Population  -1.8074075     58.244596
## Education    0.5962186     44.598731
## US           4.2172102     18.073863
## Urban        0.1728373      9.822082
```

```
bag.car.predict<-predict(bag.car,Cars.test)
mean((bag.car.predict-Cars.test$Sales)^2)
```

```
## [1] 2.605253
```

Used randomForest package with mtry = 10 for bagging. The resulting test MSE was 2.574, which is reduced comapred to the pruned model, thus proving that bagging helped in reduced the MSE value. This also shows that Price and ShelveLoc remain the 2 most important variables in predicting carseat sales in this dataset.

8e.Use random forests to analyze this data. What test MSE do you obtain?

```
set.seed(1)
rf.car<-randomForest(Sales~.,Cars.train,
                     importance=TRUE,
                     mtry=sqrt(10))
importance(rf.car) %>% as.data.frame() %>%arrange(desc(IncNodePurity))
```

```
##                %IncMSE IncNodePurity
## Price       34.9793386     385.93142
## ShelveLoc   34.9248499     298.54210
## Age         14.3055912     178.42061
## CompPrice   14.8840765     158.82956
## Income       4.3293950     125.64850
## Advertising  8.2215192     107.51700
## Population  -0.9488134      97.06024
## Education    1.3117842      70.49202
## US           6.1139696      33.98963
## Urban       -1.2680807      17.39986
```

```
rf.car.predict <- predict(rf.car,Cars.test)
mean((rf.car.predict-Cars.test$Sales)^2)
```

```
## [1] 2.960559
```

The random forest function caused the MSE to increase compared to the bagging method. Important factors remain the same, Price and ShelveLoc.

8f. Now use BART

```
library(BART)
```

```
## Warning: package 'BART' was built under R version 4.1.3
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## Loading required package: nnet
```

```
## Loading required package: survival
```

```
x <- Carseats[,1:11]
y <- Carseats[, "Sales"]
xtrain <- x[train , ]
ytrain <- y[train]
xtest <- x[-train , ]
ytest <- y[-train]
set.seed (1)
bartfit <- gbart (xtrain , ytrain , x.test = xtest)
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 200, 15, 200
## y1,yn: 2.781850, 1.091850
## x1,x[n*p]: 10.360000, 1.000000
## xp1,xp[np*p]: 11.220000, 1.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 1
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.273474,3,4.01382e-30,7.57815
## *****sigma: 0.000000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,15,0
## *****printevery: 100
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 5s
## trcnt,tecnt: 1000,1000
```

```
yhat.bart <- bartfit$yhat.test.mean
mean ((ytest - yhat.bart)^2)
```

```
## [1] 0.1603478
```

Q9

9a.Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
library(tidyverse)
```

```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v stringr 1.4.0
## v tidyr   1.1.4     v forcats 0.5.1
## v readr   2.0.2
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x nlme::collapse()        masks dplyr::collapse()
## x randomForest::combine() masks dplyr::combine()
## x dplyr::filter()         masks stats::filter()
## x dplyr::lag()            masks stats::lag()
## x ggplot2::margin()       masks randomForest::margin()
```

```
summary(OJ)
```

```
##  Purchase WeekofPurchase     StoreID        PriceCH         PriceMM
##  CH:653   Min.   :227.0   Min.   :1.00   Min.   :1.690   Min.   :1.690
##  MM:417   1st Qu.:240.0   1st Qu.:2.00   1st Qu.:1.790   1st Qu.:1.990
##           Median :257.0   Median :3.00   Median :1.860   Median :2.090
##           Mean   :254.4   Mean   :3.96   Mean   :1.867   Mean   :2.085
##           3rd Qu.:268.0   3rd Qu.:7.00   3rd Qu.:1.990   3rd Qu.:2.180
##           Max.   :278.0   Max.   :7.00   Max.   :2.090   Max.   :2.290
##      DiscCH            DiscMM         SpecialCH        SpecialMM
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.00000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.05186   Mean   :0.1234   Mean   :0.1477   Mean   :0.1617
##  3rd Qu.:0.00000   3rd Qu.:0.2300   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :0.50000   Max.   :0.8000   Max.   :1.0000   Max.   :1.0000
##     LoyalCH          SalePriceMM     SalePriceCH      PriceDiff       Store7
##  Min.   :0.000011   Min.   :1.190   Min.   :1.390   Min.   :-0.6700   No :714
##  1st Qu.:0.325257   1st Qu.:1.690   1st Qu.:1.750   1st Qu.: 0.0000   Yes:356
##  Median :0.600000   Median :2.090   Median :1.860   Median : 0.2300
##  Mean   :0.565782   Mean   :1.962   Mean   :1.816   Mean   : 0.1465
##  3rd Qu.:0.850873   3rd Qu.:2.130   3rd Qu.:1.890   3rd Qu.: 0.3200
##  Max.   :0.999947   Max.   :2.290   Max.   :2.090   Max.   : 0.6400
##    PctDiscMM        PctDiscCH       ListPriceDiff       STORE
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.140   1st Qu.:0.000
```

```
##  Median :0.0000   Median :0.00000   Median :0.240   Median :2.000
##  Mean   :0.0593   Mean   :0.02731   Mean   :0.218   Mean   :1.631
##  3rd Qu.:0.1127   3rd Qu.:0.00000   3rd Qu.:0.300   3rd Qu.:3.000
##  Max.   :0.4020   Max.   :0.25269   Max.   :0.440   Max.   :4.000
```

```
set.seed(1)
train_index <- sample(1:nrow(OJ), 800)
train <- OJ[train_index, ]
test <- OJ[-train_index, ]
```

9b.Fit a tree to the training data, with Purchase as the response and the other variables as predictors.What is the training error rate? How many terminal nodes does the tree have?

```
OJ_tree <- tree(Purchase ~ ., train)
summary(OJ_tree)
```

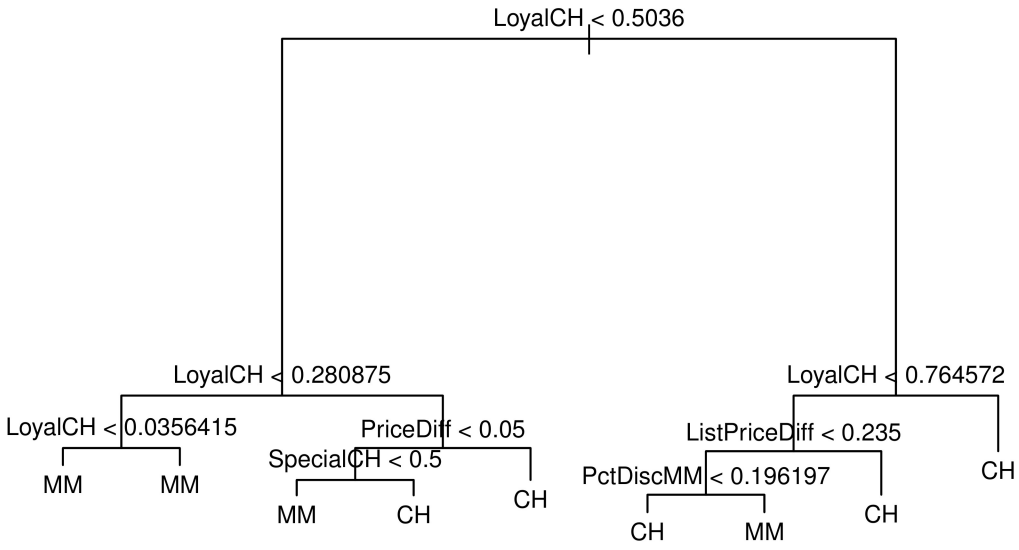
```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
## [5] "PctDiscMM"
## Number of terminal nodes:  9
## Residual mean deviance:  0.7432 = 587.8 / 791
## Misclassification error rate: 0.1588 = 127 / 800
```

There are 9 terminal nodes and there is a training error rate of 15.88%. 5 of the 17 predictors were used in splits.

9c.Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

9d.Create a plot of the tree, and interpret the results.

```
plot(OJ_tree)
text(OJ_tree, pretty=0, cex=.75)
```



Based on the tree plot above the mose important splitting variabls is LoyalCH, followed by PriceDiff. Those that had LoyalCH < 0.281 were predicted to by MM OJ, with those that lean towads CH OJ still choosing MM if it was below SpecialCH < 0.5.

9e. Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

```
# confusion matrix
test_pred <- predict(OJ_tree, test, type = "class")
table(test_pred, test_outcome = test$Purchase)
```

```
##          test_outcome
## test_pred  CH  MM
##        CH 160  38
##        MM   8  64
```

```
# test error rate
1 - mean(test_pred == test$Purchase)
```

```
## [1] 0.1703704
```

The test error rate is 0.17037.

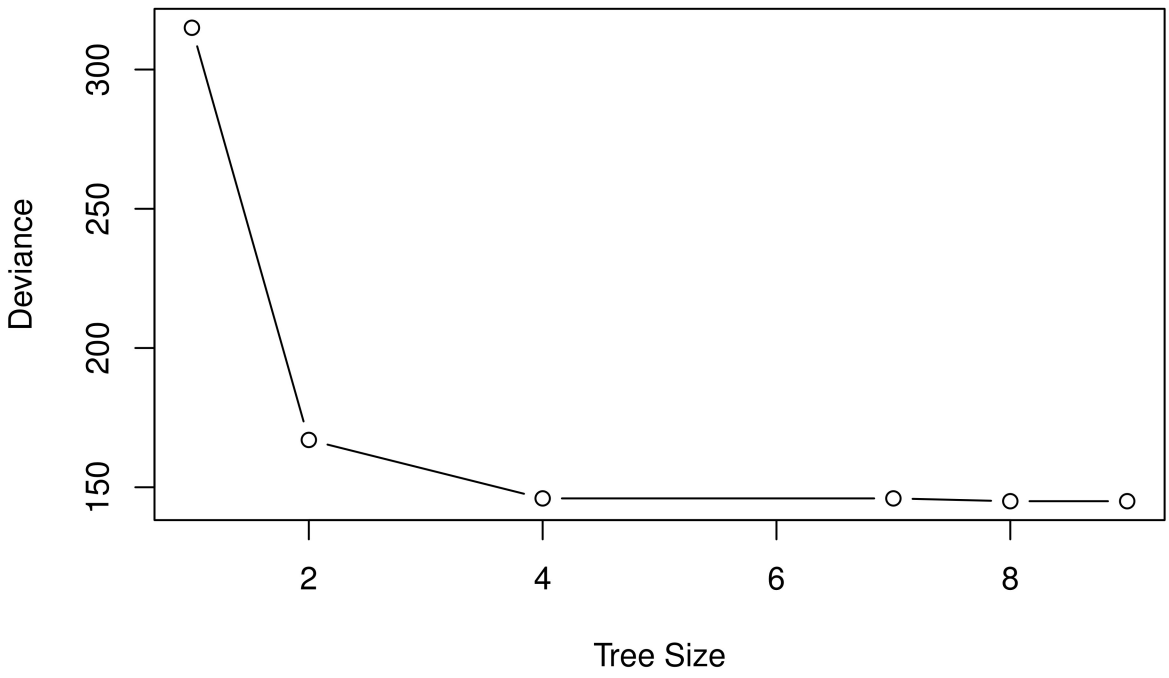
9f.Apply the cv.tree() function to the training set in order to determine the optimal tree size

```
set.seed(1)

CV_OJ_tree <- cv.tree(OJ_tree, K = 10, FUN = prune.misclass)
CV_OJ_tree
```

```
## $size
## [1] 9 8 7 4 2 1
##
## $dev
## [1] 145 145 146 146 167 315
##
## $k
## [1]       -Inf   0.000000   3.000000   4.333333  10.500000 151.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

9g.Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

```
# tree size = # of nodes (1 = root node w no splits)
plot(CV_OJ_tree$size, CV_OJ_tree$dev, type = "b", xlab = "Tree Size", ylab = "Deviance")
```

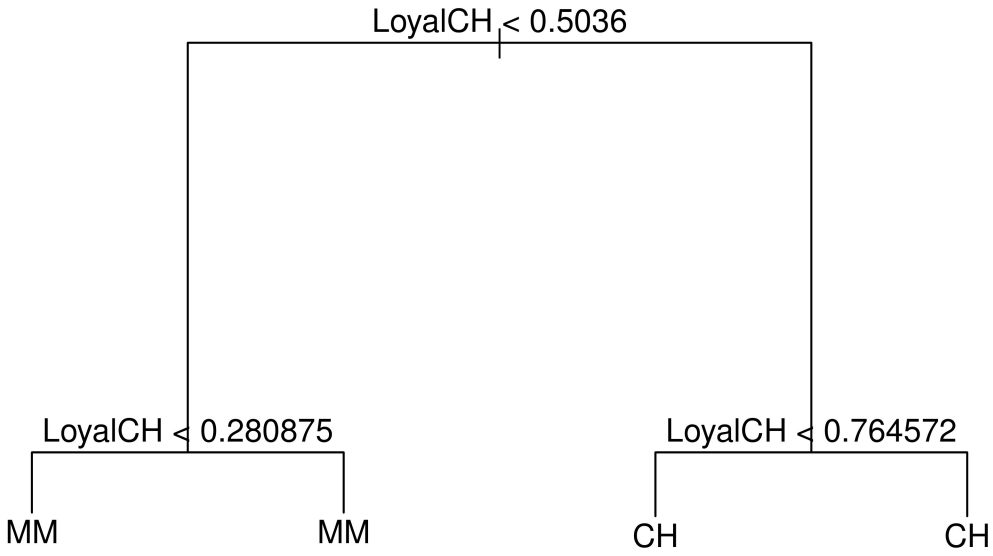


9h. Which tree size corresponds to the lowest cross-validated classification error rate?

From both 9f. and 9g. we are able to see and conclude that both tree size 4, 7, 8 & 9 have very similar AND lowest cross-validation errors, therefore we will choose the model with 4.

9i.Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation.

```
pruned_OJ_tree <- prune.tree(OJ_tree, best = 4)
plot(pruned_OJ_tree)
text(pruned_OJ_tree, pretty = 0.75)
```



9j.Compare the training error rates between the pruned and unpruned trees. Which is higher?

```
# unpruned training error rate
mean(predict(OJ_tree, type = "class") != train$Purchase)
```

```
## [1] 0.15875
```

```
#pruned training error rated
mean(predict(pruned_OJ_tree, type = "class") != train$Purchase)
```

```
## [1] 0.205
```

The training error rate for the pruned tree > unpruned tree.

9k.Compare the test error rates between the pruned and unprunedtrees. Which is higher?

```
# unpruned
mean(predict(OJ_tree, type = "class", newdata = test) != test$Purchase)
```

```
## [1] 0.1703704
```

```
# pruned
mean(predict(pruned_OJ_tree, type = "class", newdata = test) != test$Purchase)
```

```
## [1] 0.1851852
```

The pruned test error rate > unpruned.