ISE 599 Midterm

Shaoqian Chen 8831737894 Wednesday

3/4/2020

1. The OJ data set from the ISLR library contains 1070 purchases where the customer either purchased

Citrus Hill or Minute Maid Orange Juice (Use ?OJ for more details). It is of interest to predict

Purchase using all other variables. Use set.seed(1) to create a training set containing a random

sample of 800 observations, and a test set containing the remaining observations.

Fit a classification tree to the training data to answer questions (a) to (c).

a

Plot the tree. What is the training error rate? What is the test error rate?

```
library(ISLR)
library(tree)
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```r
str(OJ)
```

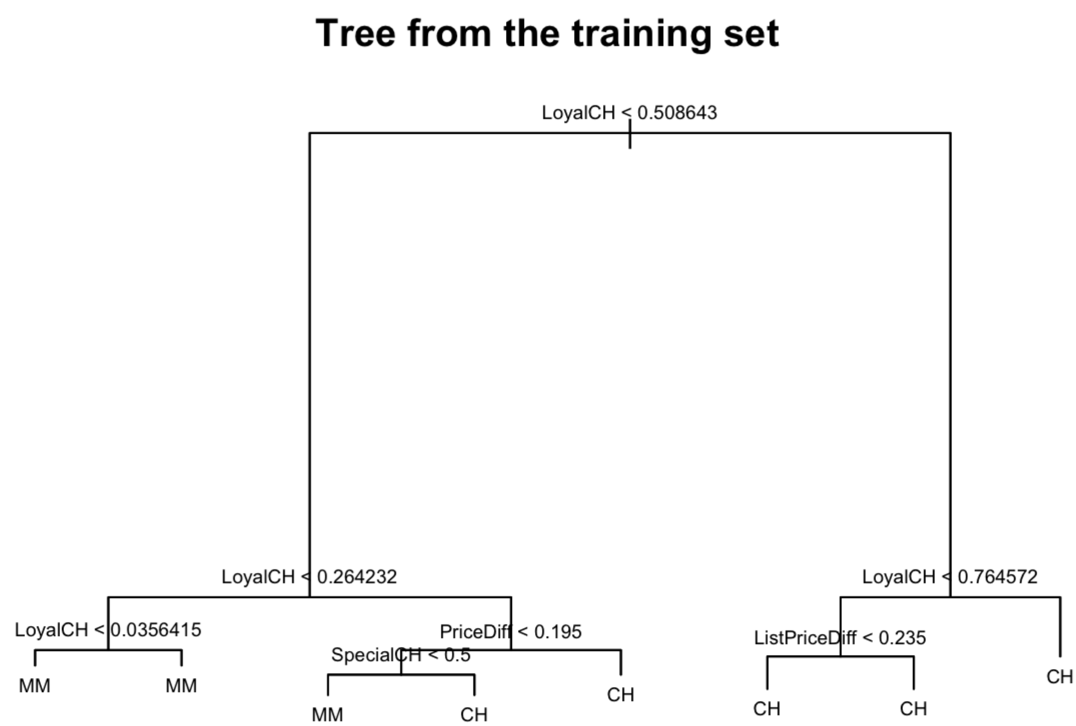
```
## 'data.frame':    1070 obs. of  18 variables:
##  $ Purchase      : Factor w/ 2 levels "CH","MM": 1 1 1 2 1 1 1 1 1 1 ...
##  $ WeekofPurchase: num  237 239 245 227 228 230 232 234 235 238 ...
##  $ StoreID       : num  1 1 1 1 7 7 7 7 7 7 ...
##  $ PriceCH       : num  1.75 1.75 1.86 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceMM       : num  1.99 1.99 2.09 1.69 1.69 1.99 1.99 1.99 1.99 1.99 ...
##  $ DiscCH        : num  0 0 0.17 0 0 0 0 0 0 0 ...
##  $ DiscMM        : num  0 0.3 0 0 0 0 0.4 0.4 0.4 0.4 ...
##  $ SpecialCH     : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ SpecialMM     : num  0 1 0 0 0 1 1 0 0 0 ...
##  $ LoyalCH       : num  0.5 0.6 0.68 0.4 0.957 ...
##  $ SalePriceMM   : num  1.99 1.69 2.09 1.69 1.69 1.99 1.59 1.59 1.59 1.59 ...
##  $ SalePriceCH   : num  1.75 1.75 1.69 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceDiff     : num  0.24 -0.06 0.4 0 0 0.3 -0.1 -0.16 -0.16 -0.16 ...
##  $ Store7        : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 2 2 2 2 2 ...
##  $ PctDiscMM     : num  0 0.151 0 0 0 ...
##  $ PctDiscCH     : num  0 0 0.0914 0 0 ...
##  $ ListPriceDiff : num  0.24 0.24 0.23 0 0 0.3 0.3 0.24 0.24 0.24 ...
##  $ STORE         : num  1 1 1 1 0 0 0 0 0 0 ...
```

```r
#RNGkind(sample.kind = 'Rounding')
set.seed(1)
train = sample(nrow(OJ),800)
OJ.train = OJ[train, ]
OJ.test = OJ[-train, ]
```

```r
tree.OJ = tree(Purchase ~ . , data = OJ, subset = train)
summary(tree.OJ)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ, subset = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7305 = 578.6 / 792
## Misclassification error rate: 0.165 = 132 / 800
```

```
plot(tree.OJ)
text(tree.OJ,cex=0.6,pretty=0)
title("Tree from the training set")
```




```
#test error rate
OJ.pred= predict(tree.OJ, OJ.test, type = "class")
table(OJ.test$Purchase, OJ.pred)
```

```
##     OJ.pred
##       CH  MM
##   CH 147  12
##   MM  49  62
```

```
mean(OJ.test$Purchase != OJ.pred)
```

```
## [1] 0.2259259
```

Training error rate = 16.5%;

Test error rate = 22.59259%

b

Use set.seed(2) and cross-validation to find the best number of terminal nodes.

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

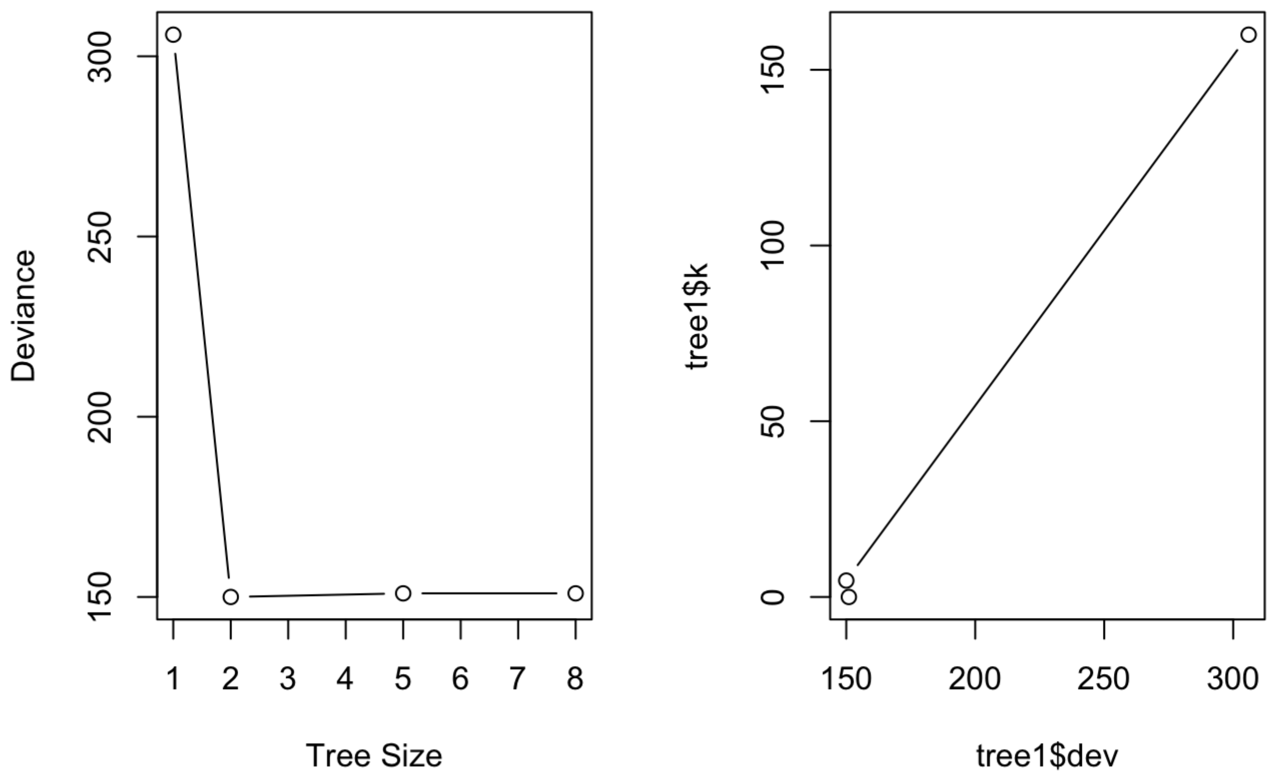
```
set.seed(2)
tree1= cv.tree(tree.OJ, FUN = prune.misclass)
names(tree1)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```
tree1
```

```
## $size
## [1] 8 5 2 1
##
## $dev
## [1] 151 151 150 306
##
## $k
## [1]       -Inf   0.000000   4.666667 160.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
par(mfrow=c(1,2))
plot(tree1$size, tree1$dev, type = "b", xlab = "Tree Size", ylab = "Deviance")
plot(tree1$dev, tree1$k, type = "b")
```



####

Tre size with 2 nodes corresponding to the lowest cross-validated classification error rate

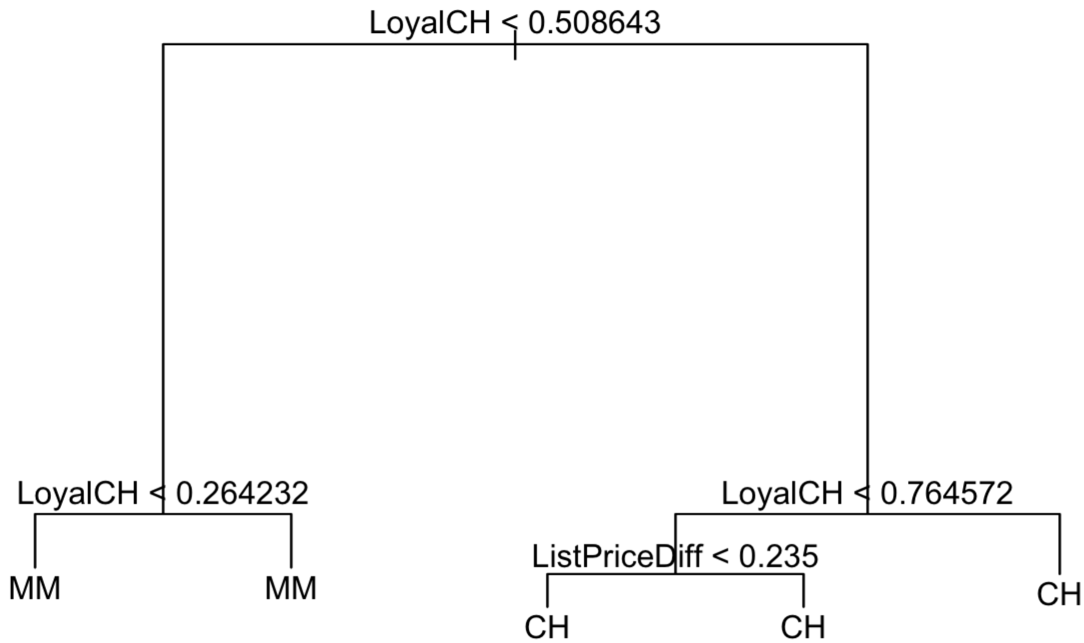
C

Plot a pruned tree with five terminal nodes. What is the test error rate?

```
pruned1= prune.tree(tree.OJ, best = 5)
summary(pruned1)
```

```
##
## Classification tree:
## snip.tree(tree = tree.OJ, nodes = 4:5)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "ListPriceDiff"
## Number of terminal nodes:  5
## Residual mean deviance:  0.7829 = 622.4 / 795
## Misclassification error rate: 0.1825 = 146 / 800
```

```
plot(pruned1)
text(pruned1)
```




```
pred2= predict(pruned1, OJ.test, type = "class")
table(OJ.test$Purchase, pred2)
```

```
##     pred2
##       CH  MM
##   CH 119  40
##   MM  30  81
```

```
mean(OJ.test$Purchase != pred2)
```

```
## [1] 0.2592593
```

Test error rate is 25.92593%

d) (10 pts.) Which predictors are the most important? What is the test error rate? When fitting a boosted tree the number of trees, depth of trees, shrinkage, should be carefully selected. If a tuning grid is defined, the train function can be used to tune these parameters (see p217, handout).

```
train2=randomForest(Purchase~.,data=OJ,subset=train,mtry=17,importance = T)
summary(train2)
```

```
##                 Length Class  Mode
## call               6   -none- call
## type               1   -none- character
## predicted        800   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes           1600   matrix numeric
## oob.times        800   -none- numeric
## classes            2   -none- character
## importance        68   -none- numeric
## importanceSD      51   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                800   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
pred3= predict(train2, OJ.test,type = "class")
table(OJ.test$Purchase, pred3)
```

```
##     pred3
##       CH  MM
##   CH 132  27
##   MM  29  82
```

```
mean(OJ.test$Purchase != pred3)
```

```
## [1] 0.2074074
```

```
importance(train2)
```

```
##                         CH         MM MeanDecreaseAccuracy MeanDecreaseGini
## WeekofPurchase 13.5795438   4.783217            14.412380        34.224017
## StoreID         8.0761610  13.166715            15.948238        10.793909
## PriceCH         7.7672071   3.189580             8.843823         4.529538
## PriceMM         3.2181449   5.801087             6.652419         3.928933
## DiscCH         -1.2936648   6.297739             4.761091         2.094399
## DiscMM          7.2990487   2.036354             7.795884         2.840041
## SpecialCH       6.1797561   2.362523             6.413429         5.830216
## SpecialMM      -0.6595525  -3.236131            -3.034125         2.100163
## LoyalCH        78.6907996 110.331214           120.096993       233.811843
## SalePriceMM     3.4815395   7.505012             8.435593         8.753717
## SalePriceCH     8.5697938   2.381339             8.517221         5.741133
## PriceDiff      13.6688973  14.401945            21.685324        21.911899
## Store7          0.7390921   4.155137             3.973069         1.522719
## PctDiscMM       7.1333996   2.550171             7.903262         3.064694
## PctDiscCH       0.2702835   6.305326             5.391850         2.457964
## ListPriceDiff  17.1769251   3.236658            16.899851        17.608657
## STORE           6.7506902  16.343647            17.864219         9.497779
```

LoyalCH is the most important predictor, test error rate is 20.74074%.

e) (10 pts.) Fit a boosted tree selecting the best parameter values. What is the test error rate?

```
gbmGrid = expand.grid(.n.minobsinnode = 10, .interaction.depth = seq(1,7,by = 2), .n.tre
es = seq(100,1000,by = 50),.shrinkage = c(0.01,0.1))
set.seed(100)
ytrain  = OJ.train$Purchase
gbmTune = train(OJ.train[,-1],ytrain,method = 'gbm',tuneGrid = gbmGrid,verbose = FALSE)
```

```
gbmTune
```

```
## Stochastic Gradient Boosting
##
## 800 samples
##  17 predictor
##   2 classes: 'CH', 'MM'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 800, 800, 800, 800, 800, 800, ...
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  Accuracy   Kappa
##   0.01       1                   100     0.8126448  0.6023160
##   0.01       1                   150     0.8152637  0.6095701
##   0.01       1                   200     0.8167365  0.6127686
##   0.01       1                   250     0.8193767  0.6182519
##   0.01       1                   300     0.8203167  0.6198000
##   0.01       1                   350     0.8217974  0.6225704
##   0.01       1                   400     0.8223281  0.6236185
##   0.01       1                   450     0.8231352  0.6249320
##   0.01       1                   500     0.8238163  0.6262516
##   0.01       1                   550     0.8237836  0.6257345
##   0.01       1                   600     0.8229956  0.6237809
##   0.01       1                   650     0.8225685  0.6230099
##   0.01       1                   700     0.8224265  0.6225570
##   0.01       1                   750     0.8224336  0.6225724
##   0.01       1                   800     0.8224355  0.6226222
##   0.01       1                   850     0.8224490  0.6225704
##   0.01       1                   900     0.8223038  0.6222319
##   0.01       1                   950     0.8223075  0.6223010
##   0.01       1                  1000     0.8215190  0.6208595
##   0.01       3                   100     0.8201629  0.6159922
##   0.01       3                   150     0.8208651  0.6189627
##   0.01       3                   200     0.8216441  0.6211115
##   0.01       3                   250     0.8226003  0.6234273
##   0.01       3                   300     0.8237094  0.6258893
##   0.01       3                   350     0.8236863  0.6258643
##   0.01       3                   400     0.8251697  0.6293162
##   0.01       3                   450     0.8247903  0.6283911
##   0.01       3                   500     0.8234409  0.6255892
##   0.01       3                   550     0.8233144  0.6252757
##   0.01       3                   600     0.8210015  0.6205556
##   0.01       3                   650     0.8209976  0.6208720
##   0.01       3                   700     0.8208282  0.6205073
##   0.01       3                   750     0.8194877  0.6178076
##   0.01       3                   800     0.8184110  0.6155573
##   0.01       3                   850     0.8177266  0.6141966
##   0.01       3                   900     0.8176231  0.6139424
##   0.01       3                   950     0.8174567  0.6135368
##   0.01       3                  1000     0.8166941  0.6122986
##   0.01       5                   100     0.8197373  0.6138201
##   0.01       5                   150     0.8208315  0.6183149
##   0.01       5                   200     0.8209848  0.6196734
```

```
##    0.01        5                        250      0.8204666  0.6190242
##    0.01        5                        300      0.8196539  0.6172704
##    0.01        5                        350      0.8199135  0.6179549
##    0.01        5                        400      0.8200224  0.6182694
##    0.01        5                        450      0.8196449  0.6177171
##    0.01        5                        500      0.8177505  0.6137580
##    0.01        5                        550      0.8181572  0.6148982
##    0.01        5                        600      0.8178796  0.6144737
##    0.01        5                        650      0.8174796  0.6134921
##    0.01        5                        700      0.8165297  0.6113271
##    0.01        5                        750      0.8153202  0.6088914
##    0.01        5                        800      0.8152297  0.6089080
##    0.01        5                        850      0.8140269  0.6064283
##    0.01        5                        900      0.8137724  0.6058923
##    0.01        5                        950      0.8130690  0.6044259
##    0.01        5                       1000      0.8122638  0.6026553
##    0.01        7                        100      0.8150343  0.6031882
##    0.01        7                        150      0.8161335  0.6081195
##    0.01        7                        200      0.8173236  0.6118925
##    0.01        7                        250      0.8154248  0.6084090
##    0.01        7                        300      0.8158179  0.6096941
##    0.01        7                        350      0.8148888  0.6078400
##    0.01        7                        400      0.8135864  0.6051312
##    0.01        7                        450      0.8136979  0.6058094
##    0.01        7                        500      0.8133259  0.6050940
##    0.01        7                        550      0.8130395  0.6046176
##    0.01        7                        600      0.8125303  0.6037894
##    0.01        7                        650      0.8122349  0.6034510
##    0.01        7                        700      0.8112841  0.6014884
##    0.01        7                        750      0.8099307  0.5984575
##    0.01        7                        800      0.8082035  0.5948043
##    0.01        7                        850      0.8088841  0.5961994
##    0.01        7                        900      0.8090183  0.5962857
##    0.01        7                        950      0.8095526  0.5975911
##    0.01        7                       1000      0.8075258  0.5933824
##    0.10        1                        100      0.8209658  0.6195814
##    0.10        1                        150      0.8240673  0.6267431
##    0.10        1                        200      0.8195954  0.6176686
##    0.10        1                        250      0.8178754  0.6138461
##    0.10        1                        300      0.8174594  0.6133033
##    0.10        1                        350      0.8156283  0.6096862
##    0.10        1                        400      0.8177487  0.6141740
##    0.10        1                        450      0.8151349  0.6084526
##    0.10        1                        500      0.8147162  0.6081483
##    0.10        1                        550      0.8131441  0.6046098
##    0.10        1                        600      0.8134169  0.6047599
##    0.10        1                        650      0.8122277  0.6025311
##    0.10        1                        700      0.8112592  0.5998015
##    0.10        1                        750      0.8113880  0.6003380
##    0.10        1                        800      0.8114174  0.6003453
##    0.10        1                        850      0.8087160  0.5943290
##    0.10        1                        900      0.8071255  0.5913198
##    0.10        1                        950      0.8070055  0.5907607
##    0.10        1                       1000      0.8098409  0.5964215
```

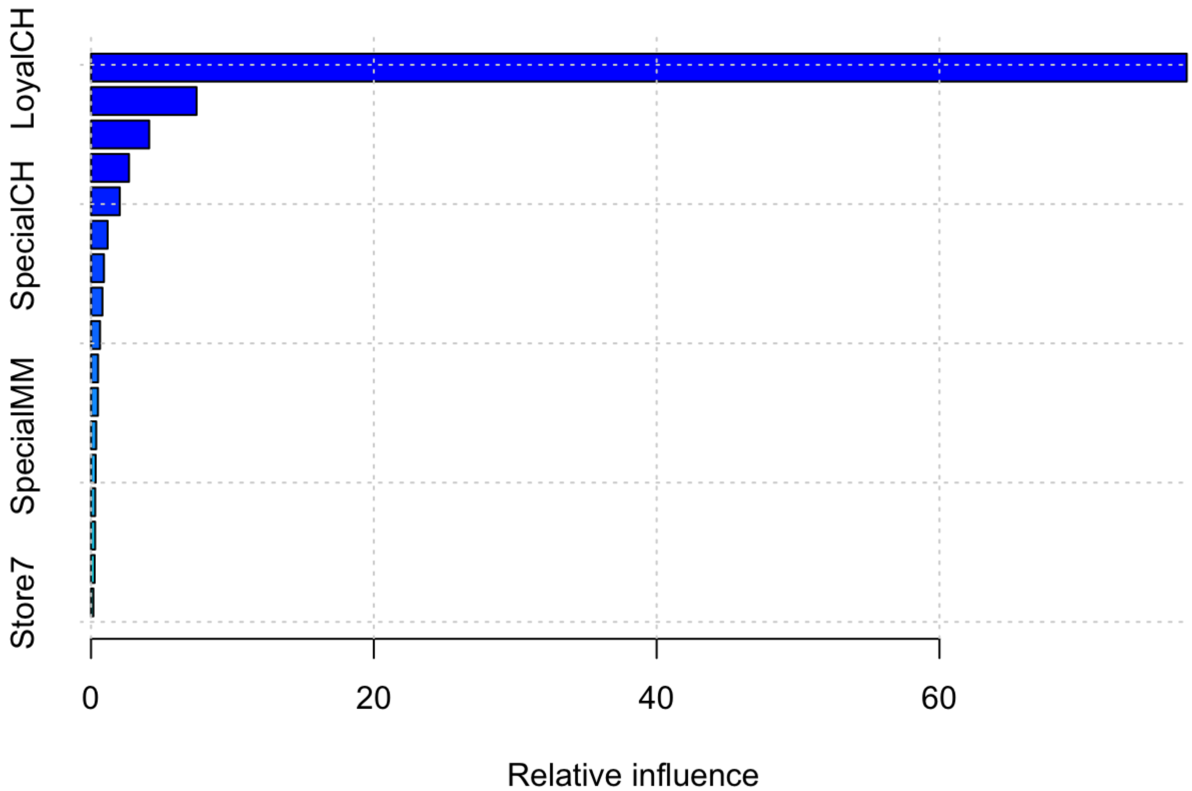
```
##    0.10        3                     100      0.8191453  0.6171008
##    0.10        3                     150      0.8136199  0.6054119
##    0.10        3                     200      0.8098641  0.5972038
##    0.10        3                     250      0.8073145  0.5916498
##    0.10        3                     300      0.8047638  0.5861040
##    0.10        3                     350      0.8041805  0.5844813
##    0.10        3                     400      0.8001504  0.5762865
##    0.10        3                     450      0.8008055  0.5773875
##    0.10        3                     500      0.8012323  0.5786874
##    0.10        3                     550      0.7999920  0.5761827
##    0.10        3                     600      0.8000481  0.5763654
##    0.10        3                     650      0.7980568  0.5721807
##    0.10        3                     700      0.7959195  0.5675869
##    0.10        3                     750      0.7972588  0.5704880
##    0.10        3                     800      0.7949569  0.5659828
##    0.10        3                     850      0.7950661  0.5659688
##    0.10        3                     900      0.7969301  0.5702408
##    0.10        3                     950      0.7957497  0.5676135
##    0.10        3                    1000      0.7954923  0.5671848
##    0.10        5                     100      0.8122057  0.6028173
##    0.10        5                     150      0.8083589  0.5946402
##    0.10        5                     200      0.8042848  0.5859704
##    0.10        5                     250      0.8044721  0.5862229
##    0.10        5                     300      0.8013718  0.5797095
##    0.10        5                     350      0.8009573  0.5789649
##    0.10        5                     400      0.7986622  0.5738113
##    0.10        5                     450      0.7983941  0.5729467
##    0.10        5                     500      0.7956921  0.5671547
##    0.10        5                     550      0.7955383  0.5670518
##    0.10        5                     600      0.7942250  0.5643898
##    0.10        5                     650      0.7951858  0.5663763
##    0.10        5                     700      0.7935666  0.5631283
##    0.10        5                     750      0.7922183  0.5602626
##    0.10        5                     800      0.7924911  0.5614454
##    0.10        5                     850      0.7901876  0.5559559
##    0.10        5                     900      0.7911353  0.5580953
##    0.10        5                     950      0.7922358  0.5603577
##    0.10        5                    1000      0.7883590  0.5524622
##    0.10        7                     100      0.8056272  0.5882130
##    0.10        7                     150      0.8038788  0.5843220
##    0.10        7                     200      0.8008446  0.5778369
##    0.10        7                     250      0.7990711  0.5744364
##    0.10        7                     300      0.7973362  0.5706409
##    0.10        7                     350      0.7961146  0.5678503
##    0.10        7                     400      0.7970469  0.5699733
##    0.10        7                     450      0.7967633  0.5692359
##    0.10        7                     500      0.7956256  0.5668483
##    0.10        7                     550      0.7953777  0.5662241
##    0.10        7                     600      0.7943265  0.5642705
##    0.10        7                     650      0.7952270  0.5661568
##    0.10        7                     700      0.7952089  0.5664765
##    0.10        7                     750      0.7949540  0.5660101
##    0.10        7                     800      0.7941144  0.5639014
##    0.10        7                     850      0.7942835  0.5644148
```

```
##   0.10       7                   900     0.7940017  0.5639658
##   0.10       7                   950     0.7926924  0.5608502
##   0.10       7                  1000     0.7921381  0.5597924
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 400,
##  interaction.depth = 3, shrinkage = 0.01 and n.minobsinnode = 10.
```

```
boost1 = gbm(Purchase~.,data = OJ.train,distribution = "gaussian", n.trees = 400, intera
ction.depth = 3, shrinkage = 0.01, n.minobsinnode = 10)
summary(boost1)
```

```
##                            var    rel.inf
## LoyalCH                LoyalCH 77.4896890
## PriceDiff            PriceDiff  7.4636900
## ListPriceDiff    ListPriceDiff  4.1137062
## StoreID                StoreID  2.6859843
## WeekofPurchase WeekofPurchase  2.0340355
## SpecialCH            SpecialCH  1.1741960
## SalePriceMM        SalePriceMM  0.9147030
## STORE                    STORE  0.8066566
## DiscCH                  DiscCH  0.6369599
## PriceMM                PriceMM  0.4977882
## SalePriceCH        SalePriceCH  0.4887161
## SpecialMM            SpecialMM  0.3653863
## PctDiscMM            PctDiscMM  0.3279045
## PriceCH                PriceCH  0.2934163
## DiscMM                  DiscMM  0.2879382
## PctDiscCH            PctDiscCH  0.2543679
## Store7                  Store7  0.1648620
```

```
grid()
```




```
pred5=predict(boost1,newdata=OJ[-train,],n.trees=400)
pred5test = ifelse(pred5>1.5,2,1)
table(OJ.test$Purchase,pred5test)
```

```
##     pred5test
##        1   2
##   CH 139  20
##   MM  25  86
```

```
result = table(OJ.test$Purchase,pred5test)
```

```
result5 =(result["MM",1]+result["CH",2])/(sum(result))
result5
```

```
## [1] 0.1666667
```

The final values used for the model were n.trees = 400, interaction.depth = 3, shrinkage = 0.01 and n.minobsinnode = 10.

test error rate is 16.66667%.

##2. In segmenting the market, a breakfast cereal manufacturer uses health and diet consciousness as the segmentation variable. Four segments are developed: ### 1 = Concerned about eating healthy foods ### 2 = Concerned primarily about weight ### 3 = Concerned about health because of illness ### 4 = Unconcerned ###

To distinguish between groups, a survey is conducted (see cereal.csv). In the survey, people are categorized as belonging to one of these groups. The most recent census reveals that 234,564,000 Americans are 18 and older.

a) (20 pts.) Use the prop.test function to find a 95% Confidence interval for the true proportion of American adults who are concerned about eating healthy foods. Then use it to estimate how many American adults belong to group 1.

```
data2 <- read.csv("cereal.csv")
head(data2)
```

```
##   Group Spend Breakfast
## 1     1 14.77         4
## 2     2  8.15         4
## 3     2  8.00         2
## 4     4  9.31         1
## 5     2 12.09         4
## 6     2  7.13         4
```

```
table(data2$Group)
```

```
##
##   1   2   3   4
## 269 484 241 256
```

```
dim(data2)
```

```
## [1] 1250    3
```

```
prop.test(269,1250)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  269 out of 1250, null probability 0.5
## X-squared = 404.42, df = 1, p-value < 2.2e-16
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.1929252 0.2392505
## sample estimates:
##      p
## 0.2152
```

```
lower = 234564000 * 0.1929252
upper = 234564000 * 0.2392505
lower
```

```
## [1] 45253307
```

```
upper
```

```
## [1] 56119554
```

The 95% confidence interval of American adults in Group 1 is: [45253307,56119554]

b) (20 pts.) Each respondent was also asked the amount spent on breakfast cereal in an average month. The company would like to know whether on average the market segment concerned about eating healthy foods outspends the other market segments.

```
d2 = data2
d2[d2==3]=2
d2[d2==4]=2
```

```
aux2 = tapply(d2$Spend, d2$Group, mean)
aux2
```

```
##        1        2
## 11.57926 10.59541
```

```
abs_diff = aux2[1]-aux2[2]
abs_diff
```

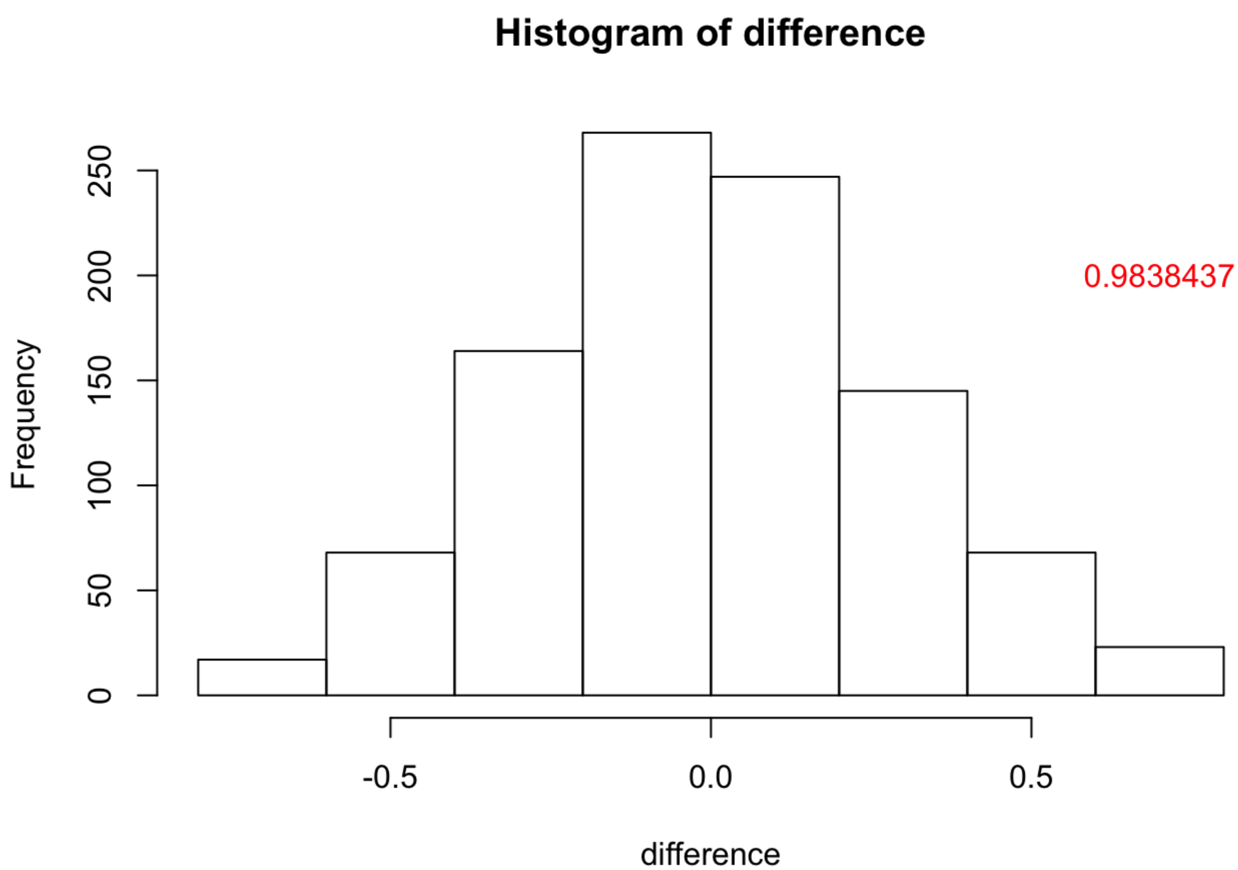
```
##         1
## 0.9838437
```

```
function1 <- function(x, n1, n2)
{
  n <- n1 + n2
  idx_b <- sample(1:n, n1)
  idx_a <- setdiff(1:n, idx_b)
  mean_diff <- mean(x[idx_b]) - mean(x[idx_a])
  return(mean_diff)
}
```

```
x = d2$Spend
difference = rep(0,1000)
for(i in 1:1000) difference[i] = function1(x,269,981)
hist(difference)
abline(v=abs_diff,col = 'red')
abs_diff
```

```
##         1
## 0.9838437
```

```
text('0.9838437',x = 0.7,y = 200, col="red")
```




```
mean(difference > abs_diff)
```

```
## [1] 0
```

```
t.test(Spend~Group, data=d2, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  Spend by Group
## t = 4.146, df = 643.95, p-value = 1.918e-05
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.5929601       Inf
## sample estimates:
## mean in group 1 mean in group 2
##        11.57926        10.59541
```

p-values from random sampling and test on proportions agree

p-value is smaller than alpha(0.675)

Reject H0;

Conclude: Group 1 is outspend all other group.

3

USC ID: 8831737894

Row 8

CHits = 42 < 450

AtBat = 185 >147

CRBI = 9 < 114.5

Result: 141.8