Assignment 2

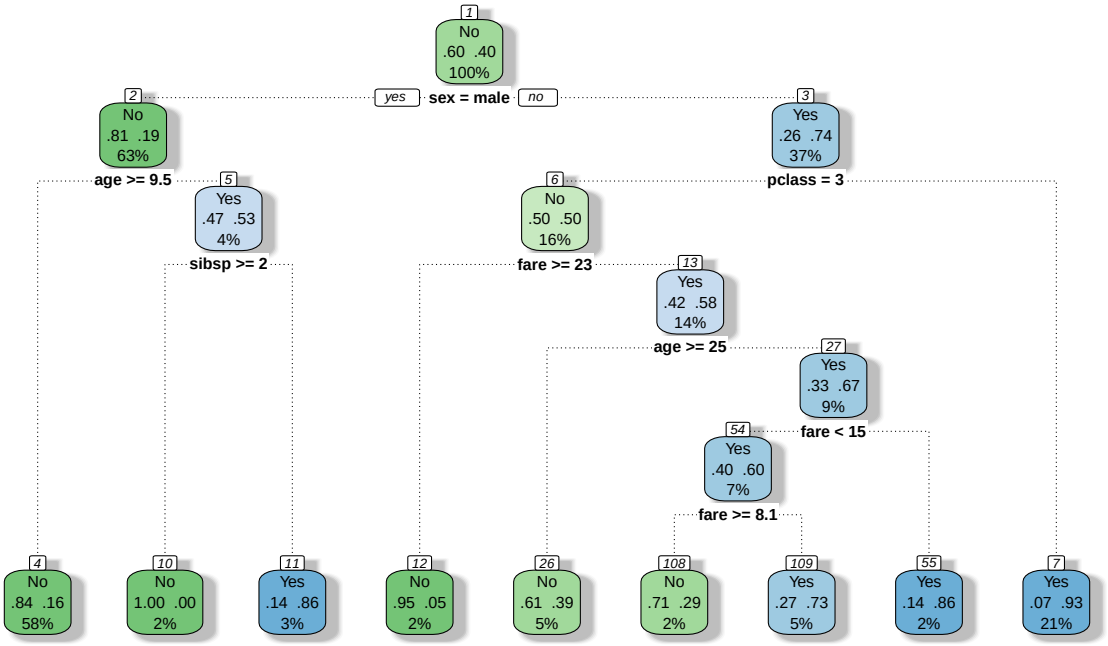
Exercise 1

Varian (2014): Replicate classification and regression trees and pruning in connection to Titanic example. (Notice that in 1. and 2. exercises, seemingly the obtained results are not necessarily exactly the same as reported in Varian (2014). Report and interpret your findings.)

```
# Fitting a classification tree with all variables that could have something
# to do with survival

library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)

# Preparing the data

sum(is.na(titanic))
```

```
[1] 1452
```

```
titanic$survived = factor(titanic$survived, levels = c(0, 1), labels = c("No",
    "Yes"))
factors = c("pclass", "sex", "parch", "embarked")
titanic[factors] = lapply(titanic[factors], as.factor)

# Splitting the data into training and test set

set.seed(123)
train.titanic = sample(1:nrow(titanic), size = (floor((2 * nrow(titanic))/3)),
    replace = F)

# Training the tree and plotting it

model.titanic = rpart(survived ~ pclass + age + sex + sibsp + parch + fare +
    embarked, data = titanic, subset = train.titanic, method = "class")

fancyRpartPlot(model.titanic, caption = "")
```



It is good to notice that the data has a lot of missing values and the results may vary a bit with different seeds.

In general the word "Yes" means survival and "No" means passing away. The percentage on the bottom means how many observations belongs to this group definied by the earlier node. The decimal number on the left hand side below the word "Yes/No" is the fraction for observations in this node to belong to "No" and the right hand side is the fraction for "Yes". If the answer for the node's criterion is no, you move to the right child node and if the answer yes, you move to left child node. The correct absolute percentage when using all the data is shown in brackets (if calculated) after the result of the classification tree.

If variables *pclass, age, sex, sibsp, parch, fare* and *embarked* are used and a classification tree is fitted using training data (2/3 of the data), 40 % of passengers survive (38 %) and 60 % don't. If a passenger is a woman, her probability of surviving is 0.74 (0.68) and for a man passenger it's 0.19. This is pretty evident due to traditional attitude "*women and children first*" in the case of emergency.

If a woman belongs to passenger class 1 or 2 (upper and middle, respectively), she survives with a probability of 0.93 (0.93), note that only 21 % of the training set's passengers belong to this group. If a woman belongs to lower class (3), she has a 50-50 chance to survive (0.49). If a woman who belongs to passenger class 3 has paid equal or over 23 for her trip, her probability of survival is 0.05 and if she has paid less, it's 0.58. Women who have paid less 23 and are under 25 years old, have 67 % chance to survive. Again, depending on how much a woman has paid, her chance to survive changes.

If the passenger is a man and 9.5 years old or older, his chance to survive is 16 %. If he's younger, his probability of survival is 0.53. This is again pretty evident due to fact that 9.5-year-old-passenger is obviously a child. If a boy has only one sibling, his probability to survive is 0.86.

```r
# Making predictions with model.titanic

titanic.test = titanic[-train.titanic, ]

predict.titanic = predict(model.titanic, newdata = titanic.test, type = "class")

# Classification rates

misclass.titanic = table(predict.titanic, titanic.test$survived)
misclass.titanic
```

```
predict.titanic  No Yes
            No  264  70
            Yes  18  85
```

```r
# Misclassification rate

misrate = (misclass.titanic[1, 2] + misclass.titanic[2, 1])/nrow(titanic[-train.titanic,
    ])
misrate
```

```
[1] 0.201373
```

```r
# Correct classification rate

1 - misrate
```

```
[1] 0.798627
```

It seems that this model predicts correctly approximately in 80 % of the cases but one of five predictions is wrong which is pretty high but quite common phenomenon for classification trees.

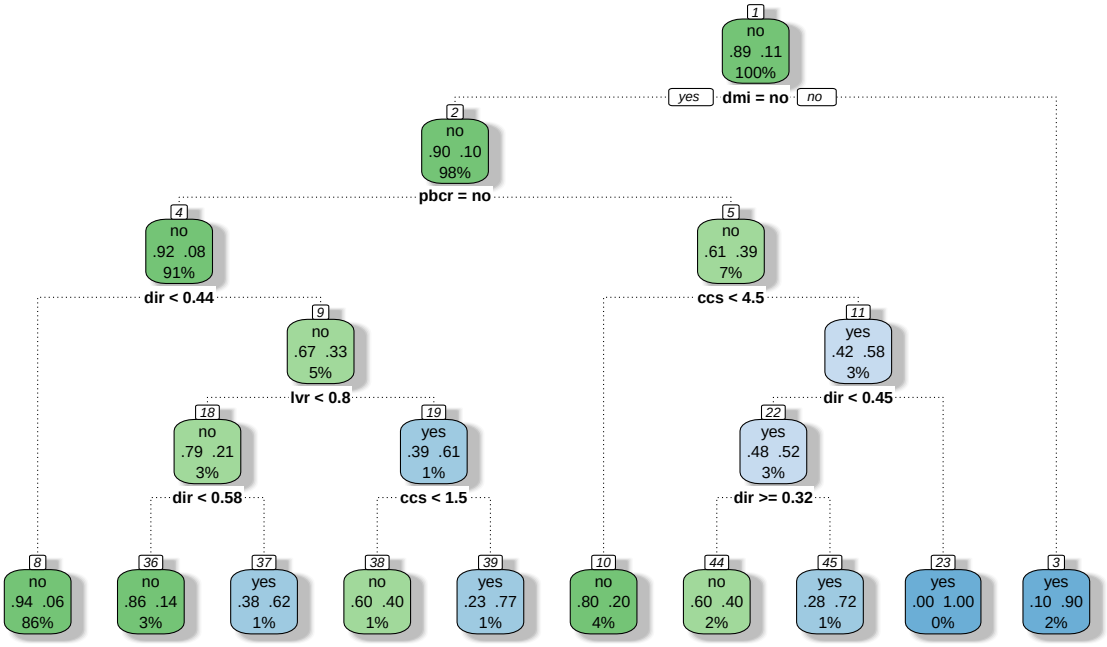
Exercise 2

Varian (2014): Replicate the example of Home Mortgage Disclosure Act (HMDA) data. In particular, see and reproduce Figure 5 (interpret your findings).

```
library(Ecdat)
library(party)

HMDA = Hmda

# Checking for missing values

sum(is.na(HMDA))
```

```
[1] 2
```

```
HMDA = na.omit(HMDA)

# Splitting the data

set.seed(123)
train.HMDA = sample(1:nrow(HMDA), size = floor((2 * nrow(HMDA)/3)), replace = F)

# Training the tree and plotting it

model.HMDA = rpart(deny ~ ., data = HMDA, subset = train.HMDA, method = "class")

fancyRpartPlot(model.HMDA, caption = "")
```



When training set is used to fit the tree, the overall denial of mortages is 11 %. If a customer has *a denial on mortage incurance* (`dmi`), his mortage application is denied in 90 % of the cases, but it is good to notice

that only 2 % of the observations belong to this node. This is the same amount as in Varian's figure 5. If a customer has the mortage incurance, his chance to get the mortage is 90 %.

The tree differs from Varian's tree after the first node because the second important criteria is *public bad credit record* (`pbcr`) not *consumer credit score* (`ccs`) as in Varian's figure 5. If a customer who has the mortage incurance has a bad public credit record, his probability to get the mortage is 0.61 (approximately the same as in figure 5) and if he has no mark on pbcr, the probability is 0.92.

If the customer has bad public credit record and his consumer credit score is less than 4.5, he is granted the mortage in 80 % of the cases otherwise his chace to get the mortage is 0.42. If css is equal or higher than 4.5, the chance to get the mortage depends on *debt payments to total income ratio* (`dir`).

If the customer has no mark on public bad credit record and his dir is less than 0.44, he is granted the mortage in 94 % of the cases. It is good to notice that the first criterion value for dir in in both branches are approximately same and it is also almost the same as in Varian's classification tree (0.431) even though the trees differ from each other. If the dir is equal or higher than 0.44, the mortage is granted in 67 % of the cases. After which it depends on *ratio of size of loan to assessed value of property* (`lvr`), and after that on css and dir depending on the value of lvr.

The figure above is the result of one spesific split into training and test set. If different splits are used, the results vary a bit, but in most of the classification trees the variable css is not the second node as in Varian's tree and with 100 simulations none of them had the variable `black` as a criterion. So the trees differ to some extent from Varian's classification tree (figure 5).

```
# Making predictions with model.HMDA

HMDA.test = HMDA[-train.HMDA, ]

predict.HMDA = predict(model.HMDA, newdata = HMDA.test, type = "class")

# Classification rates

misclass.HMDA = table(predict.HMDA, HMDA.test$deny)
misclass.HMDA
```

```
predict.HMDA  no yes
         no  681  71
         yes  10  32
```

```
# Misclassification rate

misrate2 = (misclass.HMDA[1, 2] + misclass.HMDA[2, 1])/nrow(HMDA.test)
misrate2
```

```
[1] 0.1020151
```

```
# Correct classification rate

1 - misrate2
```

```
[1] 0.8979849
```

The misclassification rate is 0.10 and the model predicts correctly in 90 % of cases.

Exercise 3

In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

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

```
library(ISLR)

# Checking for missing values

sum(is.na(Carseats))
```

```
[1] 0
```

```
CAR = Carseats

# Splitting the data (2/3 into training set)

set.seed(123)
train.CAR = sample(1:nrow(Carseats), size = floor(((2 * nrow(Carseats))/3)),
    replace = F)
```

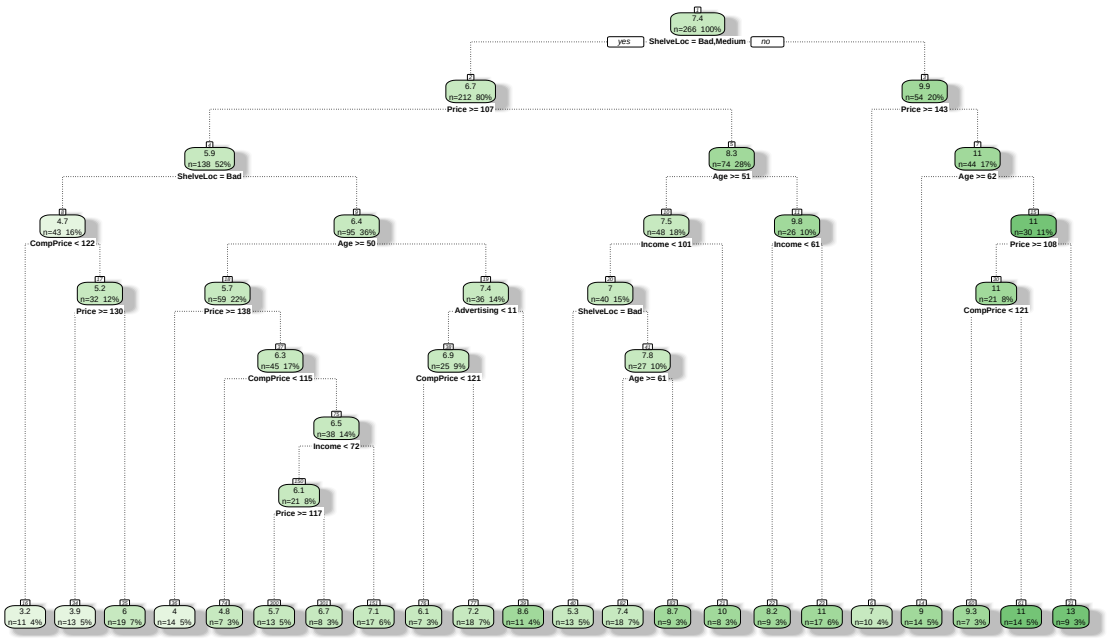
b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

```
# Fitting the tree usinf rpart() without complexity parameter and
# cross-validation

no.control = rpart.control(cp = 0, xval = 0)
model.CAR = rpart(Sales ~ ., data = CAR, subset = train.CAR, control = no.control)

fancyRpartPlot(model.CAR, caption = "")

# Variable importance

model.CAR$variable.importance
```

```
  ShelveLoc       Price         Age   CompPrice      Income Advertising
  576.50017   543.62073   254.50043   217.96412   161.25586    62.90509
 Population   Education          US
   39.70750    22.93007    17.05137
```

```
# Predictions

CAR.test = CAR[-train.CAR, ]

predict.CAR = predict(model.CAR, newdata = CAR.test, type = "vector")

# Test MSE

mean((CAR.test$Sales - predict.CAR)^2)
```

```
[1] 4.071909
```



The size of the regression tree is 22. The first dividing node is *the quality of shelving location for the car seats* (`ShelveLoc`). The second dividing node is *the price for car seats* (`Price`). After that the dividing variables vary. The variable importance is printed above.

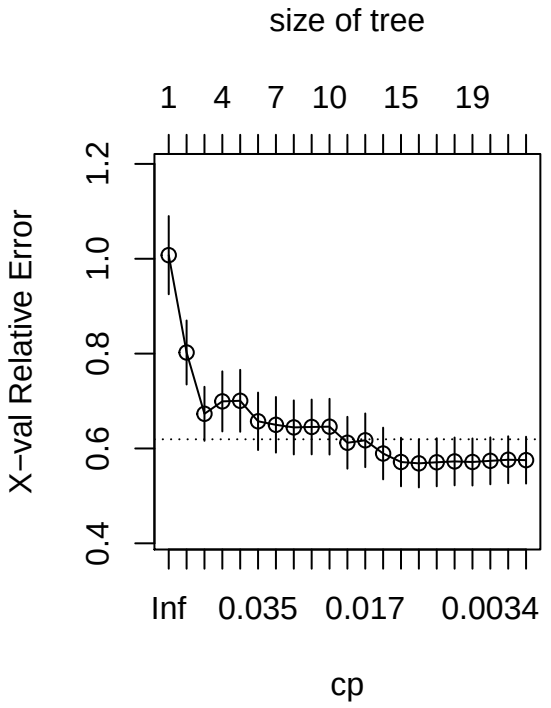
Test set error is 4.07.

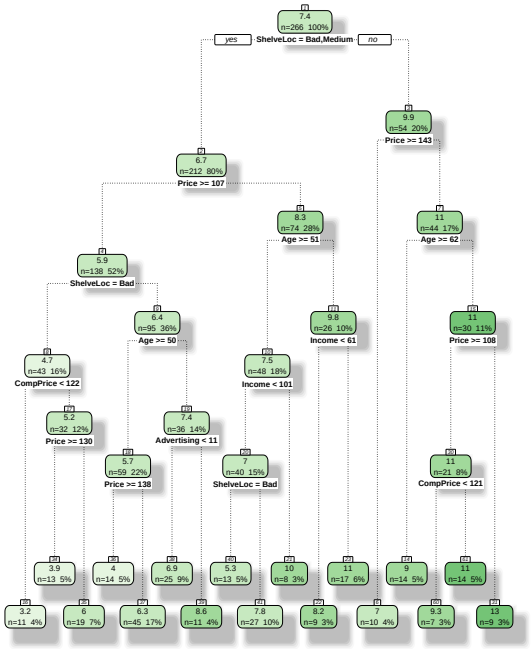
c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```
# Fitting the tree using rpart (cv with 10-folds is applied)

set.seed(123)

cv.control = rpart.control(cp = 0, xval = 10)
model.cv.CAR = rpart(Sales ~ ., data = CAR, subset = train.CAR, control = cv.control)

par(mfrow = c(1, 2))

plotcp(model.cv.CAR)

# Pruning the tree

prune.CAR = prune(model.cv.CAR, cp = 0.010092356)

fancyRpartPlot(prune.CAR, caption = "")

# Predictions
```

```r
predict.cv.CAR = predict(prune.CAR, newdata = CAR.test, type = "vector")

# Test MSE

mean((CAR.test$Sales - predict.cv.CAR)^2)

par(mfrow = c(1, 1))
```

```
[1] 4.262554
```





The size of the regression tree decreased into 17 (complexity parameter $\alpha = 0.01$) and the test MSE increased to 4.26. Note that due to randomness of splitting the date and cross-validation, results may vary. It is also good to keep in mind, that the method `tree()` gives a little different solution.

Exercise 4

This problem involves the OJ data set which is part of the ISLR package

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

```
# Checking for missing values

sum(is.na(OJ))
```

```
[1] 0
```

```
# Splitting the data into training and test sets

train.OJ = sample(1:nrow(OJ), size = 800, replace = F)

OJ.test = OJ[-train.OJ, ]
```

b) Fit a tree to the training data, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

```
# Using rpart() instead of tree() which does the cross-validation
# automatically based on CP

model.OJ = rpart(Purchase ~ ., data = OJ, subset = train.OJ, method = "class",
    control = no.control)

# Summary for method rpart() which is close to summary of tree()

printcp(model.OJ)
```

```
Classification tree:
rpart(formula = Purchase ~ ., data = OJ, subset = train.OJ, method = "class",
    control = no.control)

Variables actually used in tree construction:
 [1] DiscCH         DiscMM         ListPriceDiff  LoyalCH
 [5] PriceDiff      PriceMM        SalePriceCH    SpecialMM
 [9] StoreID        WeekofPurchase

Root node error: 313/800 = 0.39125

n= 800

         CP nsplit rel error
1 0.5207668      0   1.00000
2 0.0223642      1   0.47923
3 0.0175719      4   0.41214
4 0.0127796      6   0.37700
```

```
5 0.0095847      7   0.36422
6 0.0019169      8   0.35463
7 0.0015974     13   0.34505
8 0.0000000     18   0.33546
```

```
# Training error rate (root node error * minimum rel error)

0.39125 * 0.33546
```

```
[1] 0.1312487
```

```
# The importance of the variables

model.OJ$variable.importance
```

```
      LoyalCH        StoreID      PriceDiff    SalePriceMM WeekofPurchase
   178.136032      34.827941      31.617168      24.909689      23.291265
     PriceMM         DiscMM      PctDiscMM        PriceCH  ListPriceDiff
    20.466042      20.054776      19.255712      16.421933      11.181027
    SpecialMM    SalePriceCH          STORE         DiscCH      PctDiscCH
    10.523512       9.891476       4.141219       2.467228       2.279323
    SpecialCH
     0.087500
```

Note that method **rpart** was used instead of **tree** which gives a bit different result due to fact that it uses complexity parameter and it has been set to 0, which creates the most complex tree possible. Not all variables are used in the construction of the tree, training error is 0.13 and the three has 19 terminal nodes (splits + 1).

c) 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.

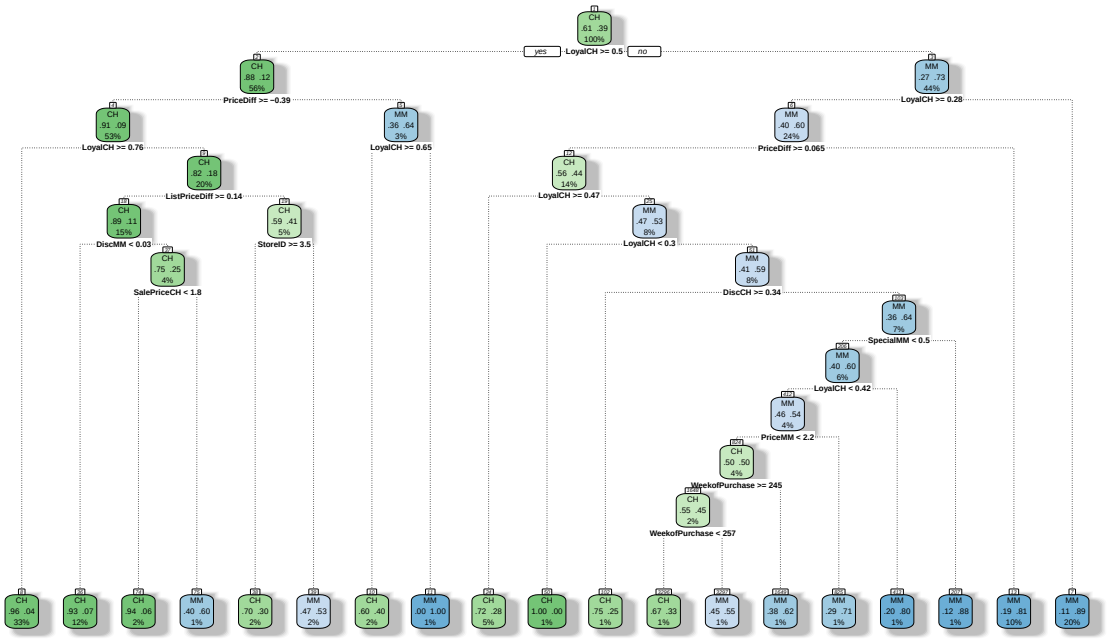
```
n= 800

node), split, n, loss, yval, (yprob)
      * denotes terminal node

   1) root 800 313 CH (0.60875000 0.39125000)
     2) LoyalCH>=0.5036 447  55 CH (0.87695749 0.12304251)
       4) PriceDiff>=-0.39 422  39 CH (0.90758294 0.09241706)
         8) LoyalCH>=0.7645725 263  10 CH (0.96197719 0.03802281) *
         9) LoyalCH< 0.7645725 159  29 CH (0.81761006 0.18238994)
          18) ListPriceDiff>=0.135 122  14 CH (0.88524590 0.11475410)
            36) DiscMM< 0.03 94   7 CH (0.92553191 0.07446809) *
            37) DiscMM>=0.03 28   7 CH (0.75000000 0.25000000)
              74) SalePriceCH< 1.755 18   1 CH (0.94444444 0.05555556) *
              75) SalePriceCH>=1.755 10   4 MM (0.40000000 0.60000000) *
          19) ListPriceDiff< 0.135 37  15 CH (0.59459459 0.40540541)
            38) StoreID>=3.5 20   6 CH (0.70000000 0.30000000) *
            39) StoreID< 3.5 17   8 MM (0.47058824 0.52941176) *
       5) PriceDiff< -0.39 25   9 MM (0.36000000 0.64000000)
        10) LoyalCH>=0.648441 15   6 CH (0.60000000 0.40000000) *
        11) LoyalCH< 0.648441 10   0 MM (0.00000000 1.00000000) *
     3) LoyalCH< 0.5036 353  95 MM (0.26912181 0.73087819)
       6) LoyalCH>=0.2753535 191  77 MM (0.40314136 0.59685864)
        12) PriceDiff>=0.065 108  47 CH (0.56481481 0.43518519)
```

```
   24) LoyalCH>=0.469289 40  11 CH (0.72500000 0.27500000) *
   25) LoyalCH< 0.469289 68  32 MM (0.47058824 0.52941176)
     50) LoyalCH< 0.303104 7   0 CH (1.00000000 0.00000000) *
     51) LoyalCH>=0.303104 61  25 MM (0.40983607 0.59016393)
      102) DiscCH>=0.335 8   2 CH (0.75000000 0.25000000) *
      103) DiscCH< 0.335 53  19 MM (0.35849057 0.64150943)
        206) SpecialMM< 0.5 45  18 MM (0.40000000 0.60000000)
          412) LoyalCH< 0.418969 35  16 MM (0.45714286 0.54285714)
            824) PriceMM< 2.205 28  14 CH (0.50000000 0.50000000)
             1648) WeekofPurchase>=245 20   9 CH (0.55000000 0.45000000)
               3296) WeekofPurchase< 257 9   3 CH (0.66666667 0.33333333) *
               3297) WeekofPurchase>=257 11   5 MM (0.45454545 0.54545455) *
             1649) WeekofPurchase< 245 8   3 MM (0.37500000 0.62500000) *
            825) PriceMM>=2.205 7   2 MM (0.28571429 0.71428571) *
          413) LoyalCH>=0.418969 10   2 MM (0.20000000 0.80000000) *
        207) SpecialMM>=0.5 8   1 MM (0.12500000 0.87500000) *
  13) PriceDiff< 0.065 83  16 MM (0.19277108 0.80722892) *
 7) LoyalCH< 0.2753535 162  18 MM (0.11111111 0.88888889) *
```

I picked the node 8 for which the split prior is $LoyalCH \geq 0.76$ and the node has 263 observations. In this node the predicted class is Citrus Hill with the probability of 0.96. This is pretty intuitive: if a customer is loyal to CH it is quite likely that he'll buy Citrus Hill.

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

```
fancyRpartPlot(model.OJ, caption = "")
```



The size of the classification tree is 19 and the first split is done by the criterion if the loyality to Citrus Hill is at least 0.5. After the first split the rest of the splits are done mainly based on loyalty to CH and price

differences. The importance of the variables is shown earlier in subsection b.

e) 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?

```
# Creating the confusion matrix

conf.matrix = table(OJ.test$Purchase, predict(model.OJ, newdata = OJ.test, type = "class"))
rownames(conf.matrix) <- paste("Actual", rownames(conf.matrix), sep = ":")
colnames(conf.matrix) <- paste("Pred", colnames(conf.matrix), sep = ":")

conf.matrix
```

```
            Pred:CH Pred:MM
  Actual:CH     130      36
  Actual:MM      22      82
```

```
# Test error rate

(conf.matrix[1, 2] + conf.matrix[2, 1])/nrow(OJ.test)
```

```
[1] 0.2148148
```

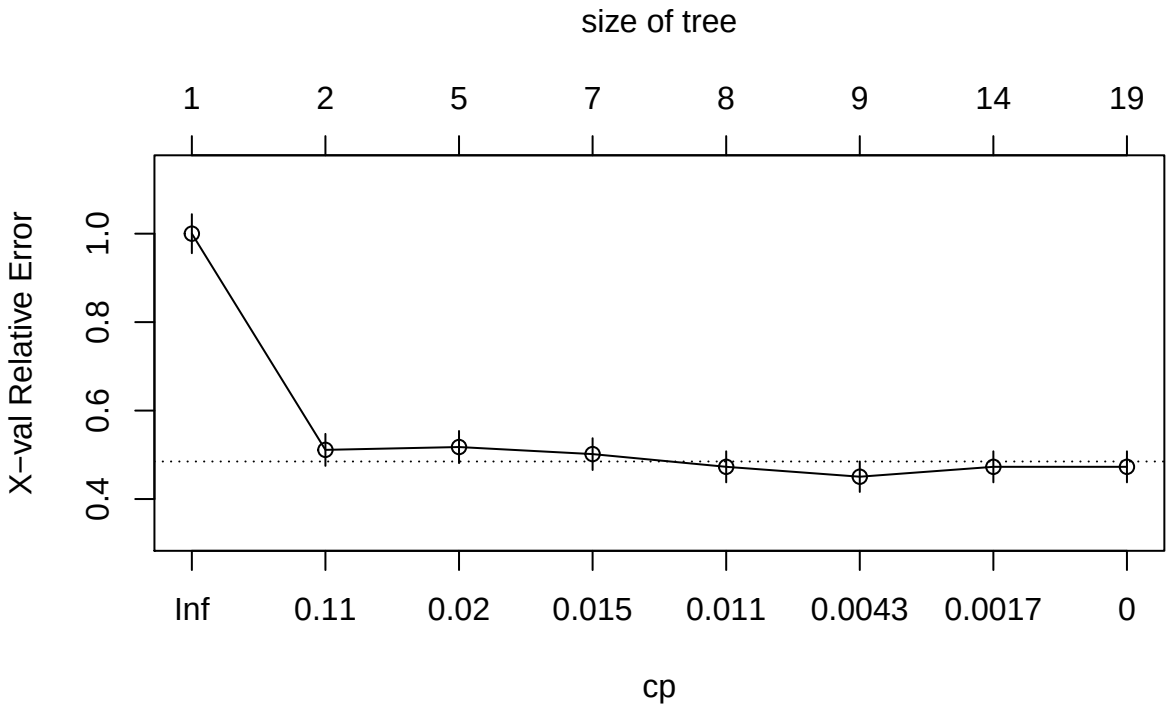
f) Apply the cv.tree() function to the training set in order to determine the optimal tree size.

```
set.seed(123)
model.cv.OJ = rpart(Purchase ~ ., data = OJ, subset = train.OJ, method = "class",
    control = cv.control)
```

Using rpart-method instead which does cross-validation as well.

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

```
plotcp(model.cv.OJ)
```

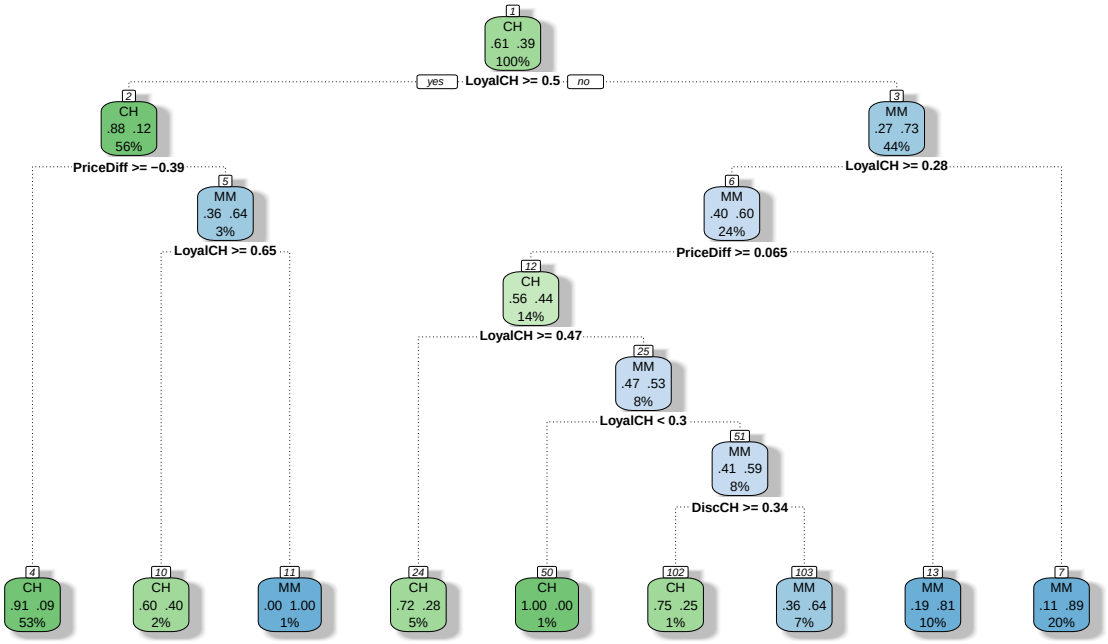


h) Which tree size corresponds to the lowest cross-validated classification error rate?

Tree size 9 minimizes the error.

Exercise 5

i) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.



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

```
Classification tree:
rpart(formula = Purchase ~ ., data = OJ, subset = train.OJ, method = "class",
    control = cv.control)

Variables actually used in tree construction:
[1] DiscCH    LoyalCH   PriceDiff

Root node error: 313/800 = 0.39125

n= 800

         CP nsplit rel error  xerror     xstd
1 0.5207668      0   1.00000 1.00000 0.044101
2 0.0223642      1   0.47923 0.51118 0.036146
3 0.0175719      4   0.41214 0.51757 0.036314
4 0.0127796      6   0.37700 0.50160 0.035889
5 0.0095847      7   0.36422 0.47284 0.035089
6 0.0043000      8   0.35463 0.45048 0.034432
```

```
[1] 0.138749
```

The training error for pruned tree is a bit higher (difference 0.008).

k) Compare the test error rates between the pruned and unpruned trees. Which is higher?

```
            CP nsplit rel error
1  0.209479048      0 1.0000000
2  0.137678067      1 0.7905210
3  0.050103719      2 0.6528429
4  0.042360282      3 0.6027392
5  0.041050532      4 0.5603789
6  0.030370283      5 0.5193284
7  0.029111169      6 0.4889581
8  0.026128710      7 0.4598469
9  0.025455691      8 0.4337182
10 0.024515185      9 0.4082625
11 0.017404358     10 0.3837473
12 0.015933824     11 0.3663430
13 0.012540785     13 0.3344753
14 0.010335742     14 0.3219345
15 0.010092356     15 0.3115988
16 0.008235451     16 0.3015064
17 0.004803927     17 0.2932710
18 0.004212668     18 0.2884670
19 0.002740852     19 0.2842544
20 0.002340189     20 0.2815135
21 0.000000000     21 0.2791733
```

```
[1] 0.2222222
```

The test error for the pruned tree is a bit higher (difference 0.007).