HW4

Sijia Yue

11/13/2019

Q1

Write two function to generate the results:

```
x_ised_boot = c(0.1,0.15,0.2,0.2,0.55,0.6,0.6,0.65,0.7, 0.75)

majority_clf = function(votes){
  pro_votes = sum(x_ised_boot>0.5)
  majority_is_pro = (length(x_ised_boot)/2) < pro_votes
  return(majority_is_pro)
}
avg_clf = function(votes){
  avg = mean(votes)
  return(avg>0.5)
}

majority_clf(x_ised_boot)
```

```
## [1] TRUE
```

```
avg_clf(x_ised_boot)
```

```
## [1] FALSE
```

When implementing majority vote, the final classification is red.

When calculating the average probability, the final classification is green.

Q2

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

```
set.seed(200)
data(OJ)
random_code = sample(1:nrow(OJ), 800, replace = F)
df_train = OJ[random_code,]
df_test = OJ[-random_code,]
head(df_train)
```

```
##     Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM
## 166       CH            274       4    1.99    2.09    0.0    0.4
## 232       CH            228       7    1.69    1.69    0.0    0.0
## 727       MM            276       3    2.09    2.09    0.2    0.4
## 181       CH            249       4    1.99    2.23    0.0    0.0
## 539       CH            259       7    1.86    2.18    0.0    0.0
## 280       MM            242       3    1.99    2.23    0.0    0.0
##     SpecialCH SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff Store7
```

```
## 166         0         0 0.990993        1.69        1.99     -0.30     No
## 232         0         0 0.535142        1.69        1.69      0.00    Yes
## 727         0         0 0.000011        1.69        1.89     -0.20     No
## 181         0         0 0.995388        2.23        1.99      0.24     No
## 539         0         0 0.320000        2.18        1.86      0.32    Yes
## 280         0         0 0.007206        2.23        1.99      0.24     No
##     PctDiscMM PctDiscCH ListPriceDiff STORE
## 166  0.191388  0.000000          0.10     4
## 232  0.000000  0.000000          0.00     0
## 727  0.191388  0.095694          0.00     3
## 181  0.000000  0.000000          0.24     4
## 539  0.000000  0.000000          0.32     0
## 280  0.000000  0.000000          0.24     3
```

(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?

```
fit_tree = tree(Purchase ~., df_train)
summary(fit_tree)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = df_train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "ListPriceDiff" "PctDiscMM"
## Number of terminal nodes:  6
## Residual mean deviance:  0.7964 = 632.4 / 794
## Misclassification error rate: 0.1713 = 137 / 800
```

6 parameters are used in the tree construction. There are 10 terminal nodes and the training error rate is 0.1425.

(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.

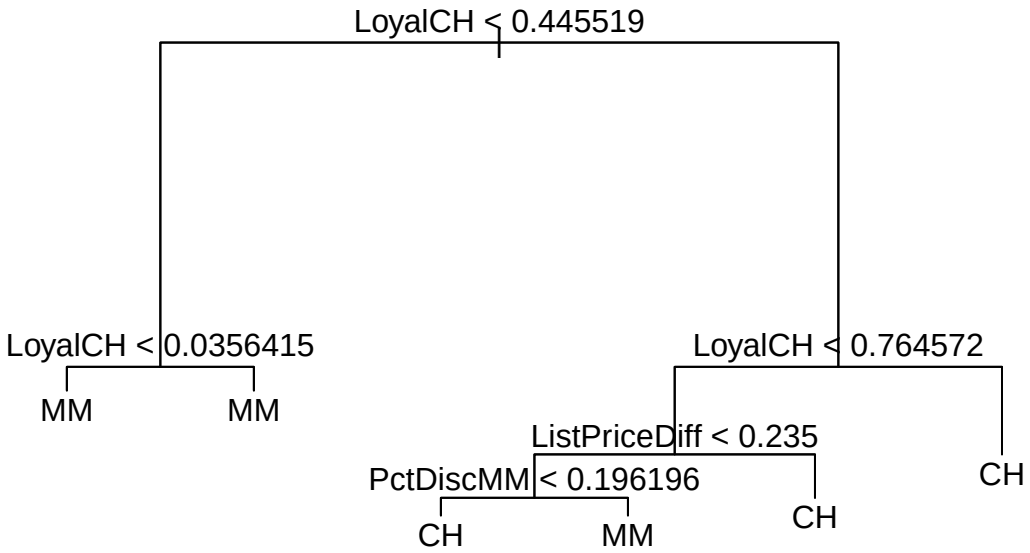
```
fit_tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1075.000 CH ( 0.60250 0.39750 )
##    2) LoyalCH < 0.445519 279  285.200 MM ( 0.20789 0.79211 )
##      4) LoyalCH < 0.0356415 55    9.996 MM ( 0.01818 0.98182 ) *
##      5) LoyalCH > 0.0356415 224  254.100 MM ( 0.25446 0.74554 ) *
##    3) LoyalCH > 0.445519 521  500.800 CH ( 0.81382 0.18618 )
##      6) LoyalCH < 0.764572 269  338.600 CH ( 0.67658 0.32342 )
##       12) ListPriceDiff < 0.235 110  151.900 MM ( 0.46364 0.53636 )
##         24) PctDiscMM < 0.196196 86  118.100 CH ( 0.55814 0.44186 ) *
##         25) PctDiscMM > 0.196196 24   18.080 MM ( 0.12500 0.87500 ) *
##       13) ListPriceDiff > 0.235 159  148.000 CH ( 0.82390 0.17610 ) *
##      7) LoyalCH > 0.764572 252   84.130 CH ( 0.96032 0.03968 ) *
```

I pick the 9th node. The node is separated according to `LoyalCH` > 0.0356415. There are 121 subjects in the class that 0.0356415 < `LoyalCH` < 0.276142 and the deviance is 117.700. The overall prediction for this group is MM and the proportion of data points in this gorup having class MM is 0.80992. This node is a significant node.

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

```
plot(fit_tree)
text(fit_tree)
```



The tree only use `LoyalCH`, `SalePriceMM`, `SpecialCH`, `PriceDiff`, `ListPriceDiff` and `STORE` for splitting. The root is splitted base on `LoyalCH` < 0.48285. Four out of five leaves on the right branch give the prediction `CH`, while two out of five on the left brance give the prediction `MM`.

(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?

```
pred_tree = predict(fit_tree, df_test, type = 'class')
confusionMatrix(pred_tree,df_test$Purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 147  24
##         MM  24  75
##
##                Accuracy : 0.8222
##                  95% CI : (0.7713, 0.8659)
##     No Information Rate : 0.6333
##     P-Value [Acc > NIR] : 8.433e-12
##
##                   Kappa : 0.6172
##
```

```
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8596
##             Specificity : 0.7576
##          Pos Pred Value : 0.8596
##          Neg Pred Value : 0.7576
##              Prevalence : 0.6333
##          Detection Rate : 0.5444
##    Detection Prevalence : 0.6333
##       Balanced Accuracy : 0.8086
##
##        'Positive' Class : CH
##
```

```
1 - sum(pred_tree == df_test$Purchase)/nrow(df_test)
```

```
## [1] 0.1777778
```

The test error rate is 0.178.

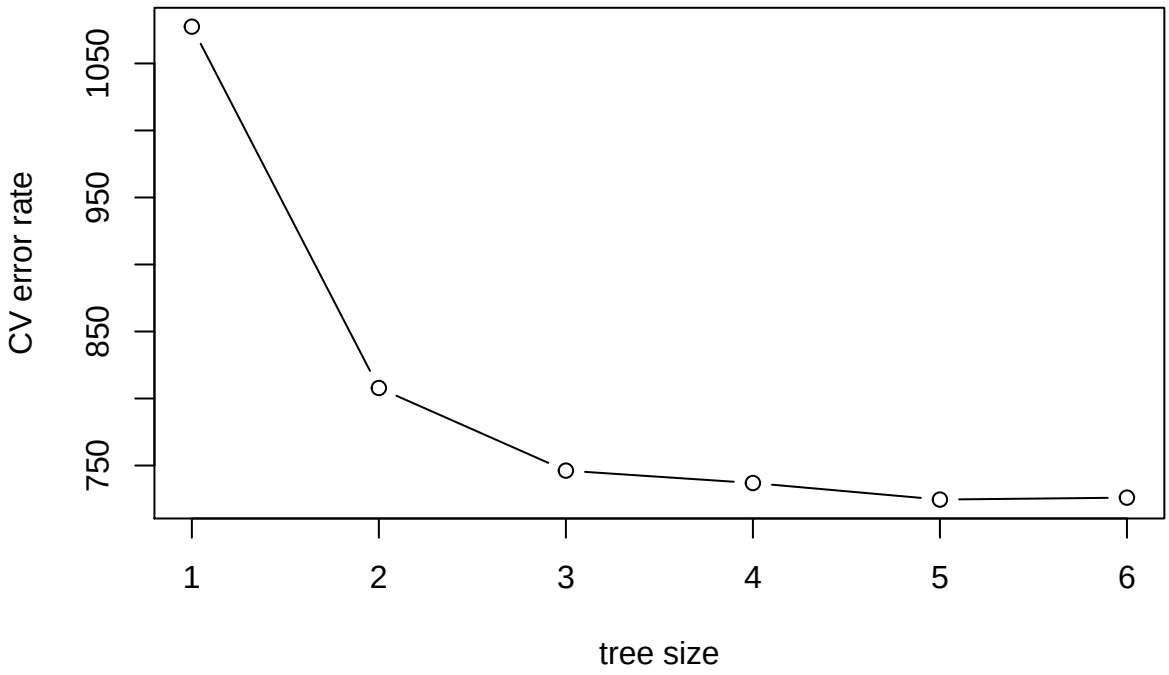
(f) Apply the cv.tree() function to the training set in order to determine the optimal tree size.

```
set.seed(200)
cv_tree = cv.tree(fit_tree)
cv_tree
```

```
## $size
## [1] 6 5 4 3 2 1
##
## $dev
## [1]  726.0312  724.5895  736.9482  746.1978  807.8629 1077.4384
##
## $k
## [1]      -Inf  15.76919  21.12084  38.71184  78.06067 289.13544
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

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

```
plot(x = cv_tree$size, y =cv_tree$dev, xlab = "tree size", ylab = "CV error rate", type = "b")
```



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

```
cv_tree$size[which(cv_tree$dev== min(cv_tree$dev))]
```

```
## [1] 5
```

Tree size equals to 6 and 7 correspond to the lowest cross-validation error rate.

(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.

```
set.seed(200)
prune_tree = prune.misclass(fit_tree)
prune_tree$size[which(prune_tree$dev == min(prune_tree$dev))]
```

```
## [1] 6 5
```

```
prune_tree = prune.misclass(fit_tree, best = 5)
```

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

```
pred_prune_tree = predict(prune_tree, df_test, type ='class')
1 - sum(pred_prune_tree == df_test$Purchase)/nrow(df_test)
```

```
## [1] 0.1777778
```

They are the same.

Q3

(a) Read data

```
gene = read.csv("Ch10Ex11.csv", header = F)
```

(b) Apply hierarchical clustering to the samples using correlation based distance, and plot the dendrogram.