Homework 3

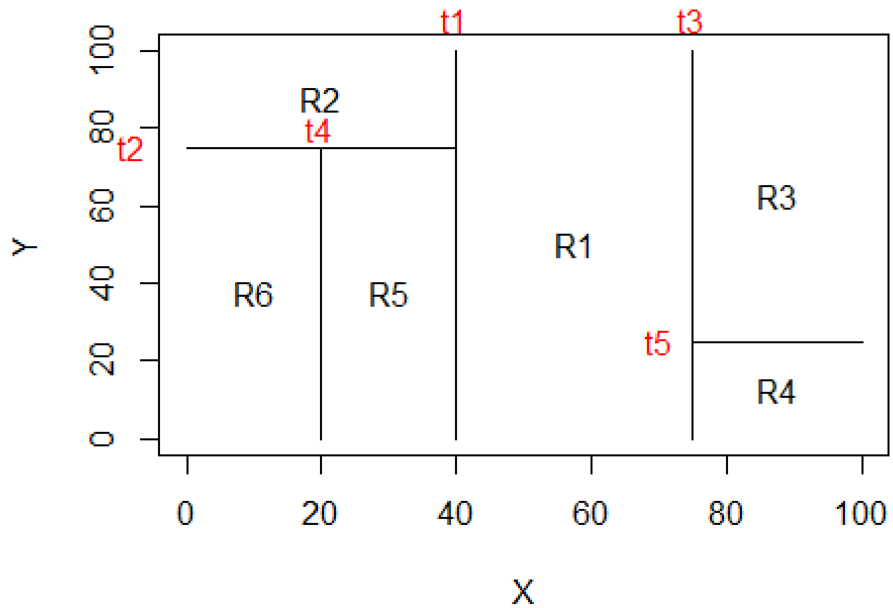
Renuka Ramachandran

9 November 2017

Question 1

Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least six regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions R1, R2, ..., the cut-points t1, t2, ..., and so forth.

```r
par(xpd = NA)
plot(NA, NA, type = "n", xlim = c(0,100), ylim = c(0,100), xlab = "X", ylab =
"Y")
# t1: x = 40; (40, 0) (40, 100)
lines(x = c(40,40), y = c(0,100))
text(x = 40, y = 108, labels = c("t1"), col = "red")
# t2: y = 75; (0, 75) (40, 75)
lines(x = c(0,40), y = c(75,75))
text(x = -8, y = 75, labels = c("t2"), col = "red")
# t3: x = 75; (75,0) (75, 100)
lines(x = c(75,75), y = c(0,100))
text(x = 75, y = 108, labels = c("t3"), col = "red")
# t4: x = 20; (20,0) (20, 75)
lines(x = c(20,20), y = c(0,75))
text(x = 20, y = 80, labels = c("t4"), col = "red")
# t5: y=25; (75,25) (100,25)
lines(x = c(75,100), y = c(25,25))
text(x = 70, y = 25, labels = c("t5"), col = "red")

text(x = (40+75)/2, y = 50, labels = c("R1"))
text(x = 20, y = (100+75)/2, labels = c("R2"))
text(x = (75+100)/2, y = (100+25)/2, labels = c("R3"))
text(x = (75+100)/2, y = 25/2, labels = c("R4"))
text(x = 30, y = 75/2, labels = c("R5"))
text(x = 10, y = 75/2, labels = c("R6"))
```

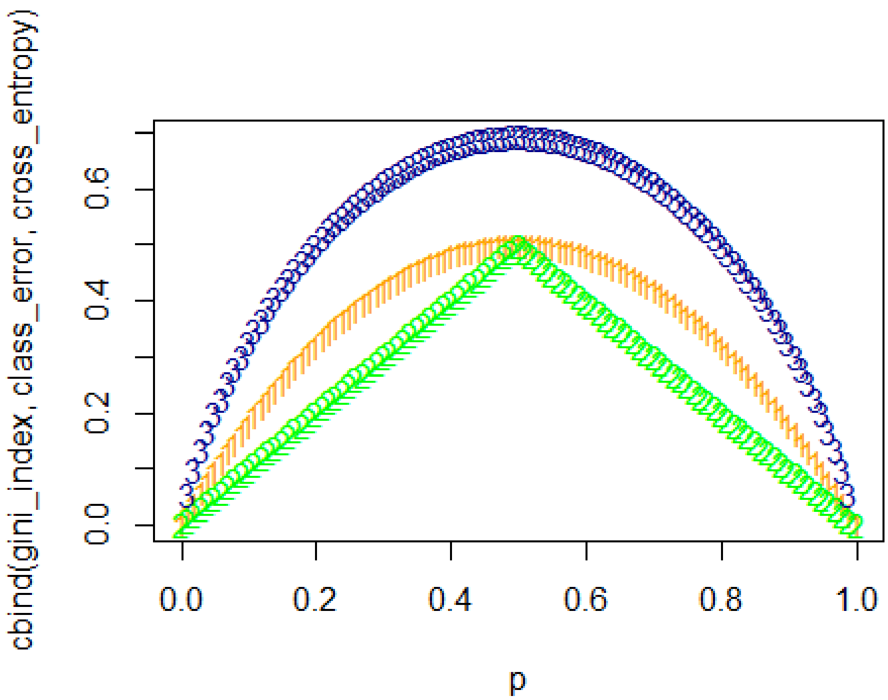


Question 2

Consider the Gini index, classification error, and entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of ^pm1. The x-axis should display ^pm1, ranging from 0 to 1, and the y-axis should display the value of the Gini index, classification error, and entropy.

Hint: In a setting with two classes, p^m1 = 1 ??? p^m2. You could make this plot by hand, but it will be much easier to make in R.

```
p <- seq(0, 1, 0.01)
gini_index <- 2 * p * (1 - p)
class_error <- 1 - pmax(p, 1 - p)
cross_entropy <- - (p * log(p) + (1 - p) * log(1 - p))
matplot(p, cbind(gini_index, class_error, cross_entropy), col = c("orange",
"green", "dark blue"))
```



Question 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.

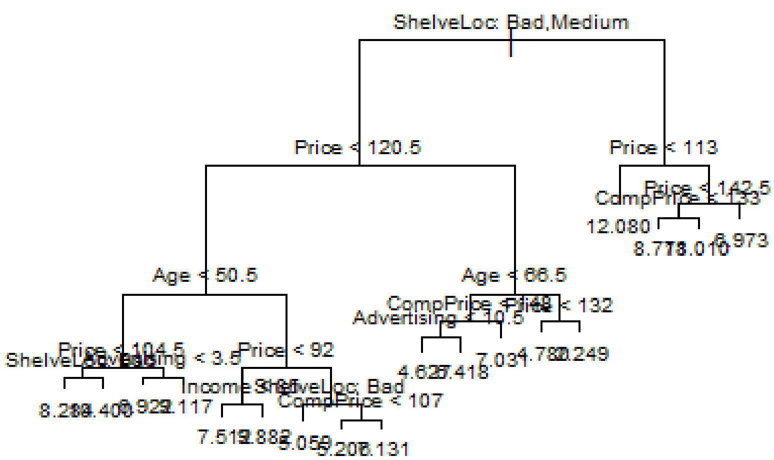
```
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.3.3

set.seed(1)
train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)
train_data <- Carseats[train, ]
test_data <- Carseats[-train, ]
```

There are 400 observations in total. Data is split equally, train data consists of 200 observations and test data also contains 200 observations

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

```
library(tree)

## Warning: package 'tree' was built under R version 3.3.3
```

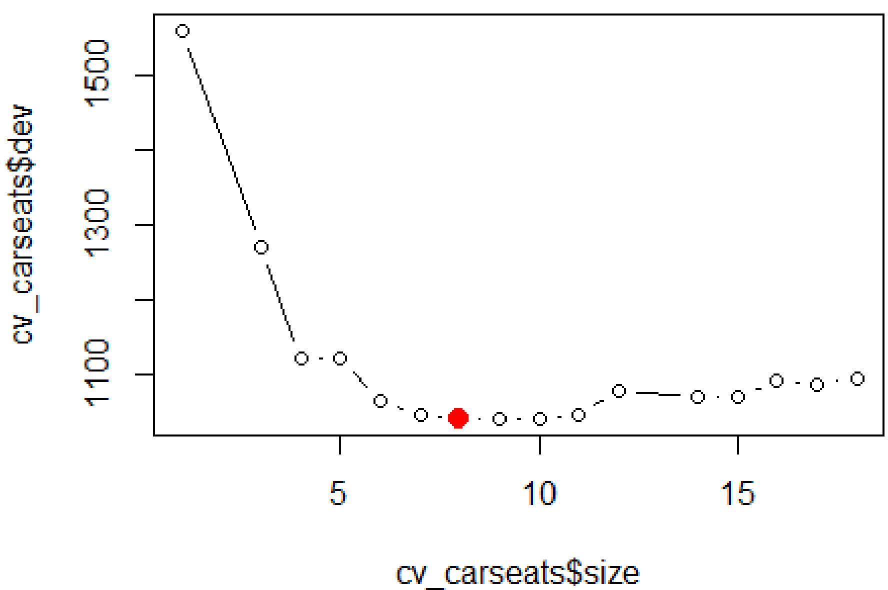
```r
tree_carseats <- tree(Sales ~ ., data = train_data)
summary(tree_carseats)

##
## Regression tree:
## tree(formula = Sales ~ ., data = train_data)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "Income"
## [6] "CompPrice"
## Number of terminal nodes:  18
## Residual mean deviance:  2.36 = 429.5 / 182
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.2570 -1.0360  0.1024  0.0000  0.9301  3.9130

plot(tree_carseats)
text(tree_carseats, pretty = 0, cex = 0.6)
```




```r
pred <- predict(tree_carseats, newdata = test_data)
mean((pred - test_data$Sales)^2)

## [1] 4.148897
```

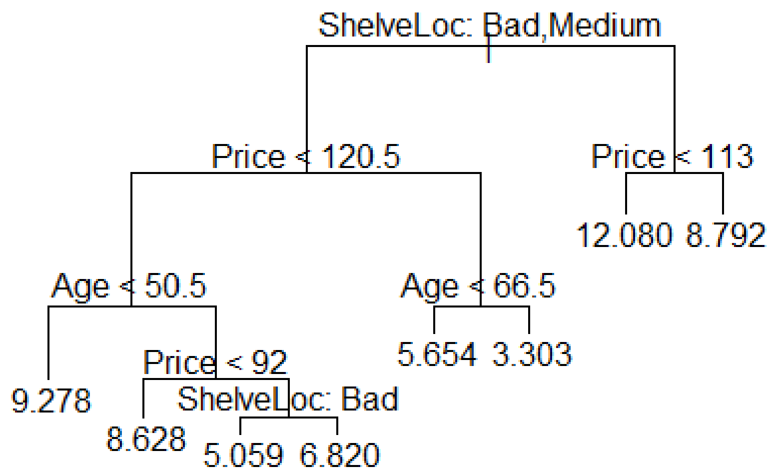
(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```
cv_carseats <- cv.tree(tree_carseats)
plot(cv_carseats$size, cv_carseats$dev, type = "b")
mintree <- which.min(cv_carseats$dev)
points(mintree, cv_carseats$dev[mintree], col = "red", cex = 2, pch = 20)
```



From the graph, tree of **size 8** is chosen as the optimal level of tree complexity by cross-validation.
Pruning the tree to obtain the 8-node tree.

```
prune_carseats <- prune.tree(tree_carseats, best = 8)
plot(prune_carseats)
text(prune_carseats, pretty = 0)
```




```r
pred <- predict(prune_carseats, newdata = test_data)
mean((pred - test_data$Sales)^2)

## [1] 5.09085
```

Inference: Pruning the tree has increased the Test MSE from 4.14 to 5.09

(d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

```r
library("randomForest")

## Warning: package 'randomForest' was built under R version 3.3.3

## randomForest 4.6-12

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

bag_carseats <- randomForest(Sales ~ ., data = train_data, mtry = 10, ntree =
500, importance = TRUE)
pred_bag <- predict(bag_carseats, newdata = test_data)
mean((pred_bag - test_data$Sales)^2)

## [1] 2.604369
```

Observation: Bagging **decreased** the Test MSE to 2.6.

```r
importance(bag_carseats)

##                %IncMSE IncNodePurity
## CompPrice   14.4124562    133.731797
## Income       6.5147532     74.346961
## Advertising 15.7607104    117.822651
## Population   0.6031237     60.227867
## Price       57.8206926    514.802084
## ShelveLoc   43.0486065    319.117972
## Age         19.8789659    192.880596
## Education    2.9319161     39.490093
## Urban       -3.1300102      8.695529
## US           7.6298722     15.723975
```

Inference: We can conclude that **Price** and **ShelveLoc** are the two most important variables

(e) Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```r
rf_carseats <- randomForest(Sales ~ ., data = train_data, mtry = 3, ntree =
500, importance = TRUE)
pred.rf <- predict(rf_carseats, newdata = test_data)
mean((pred.rf - test_data$Sales)^2)

## [1] 3.296078
```

Observation: When m = ???p, we have a **Test MSE of 3.3**

```r
importance(rf_carseats)

##                %IncMSE IncNodePurity
## CompPrice    7.5233429     127.36625
## Income       4.3612064     119.19152
## Advertising 12.5799388     138.13567
## Population  -0.2974474     100.28836
## Price       37.1612032     383.12126
## ShelveLoc   30.3751253     246.19930
## Age         16.0261885     197.44865
## Education    1.7855151      63.87939
## Urban       -1.3928225      16.01173
## US           5.6393475      32.85850
```

Inference: In this case also, **Price** and **ShelveLoc** are the two most important variables

Question 4

We now use boosting to predict Salary in the Hitters data set.

(a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

```
Hitters <- na.omit(Hitters)
Hitters$Salary <- log(Hitters$Salary)
```

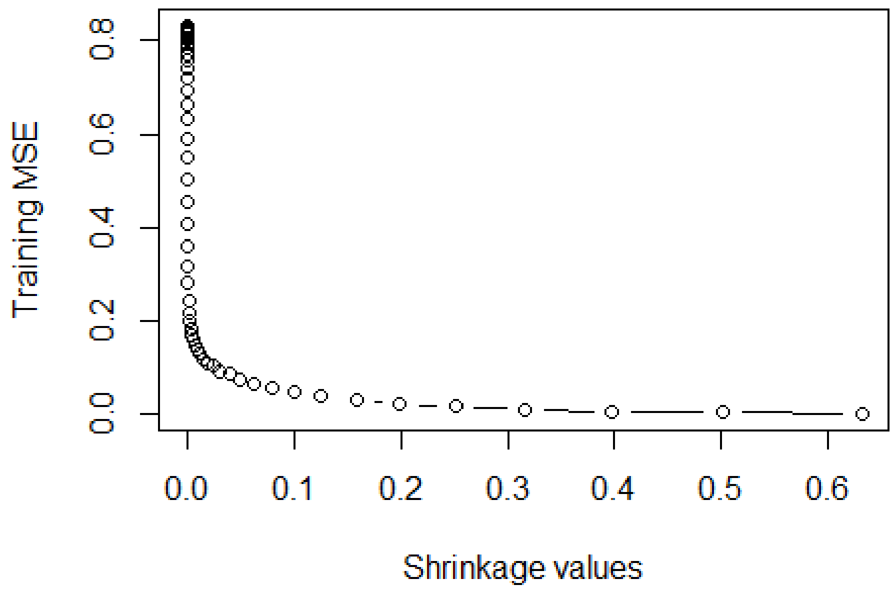
(b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```
train <- 1:200
Hitters_train <- Hitters[train, ]
Hitters_test <- Hitters[-train, ]
```

(c) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter ??. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

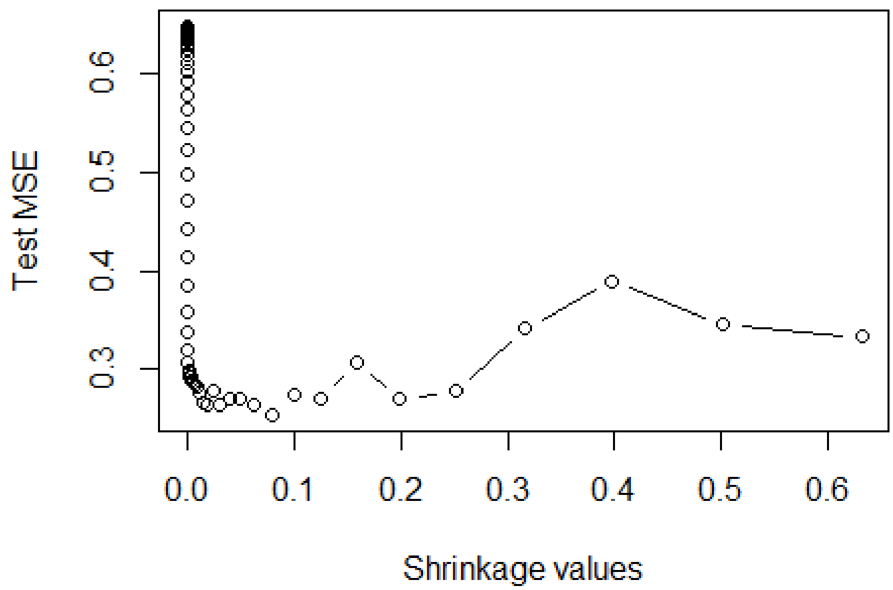
```
library(gbm)

## Warning: package 'gbm' was built under R version 3.3.3

## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3

set.seed(1)
pows <- seq(-10, -0.2, by = 0.1)
lambdas <- 10^pows
trainerr <- rep(NA, length(lambdas))
for (i in 1:length(lambdas)) {
    boost_hitters <- gbm(Salary ~ ., data = Hitters_train, distribution =
"gaussian", n.trees = 1000, shrinkage = lambdas[i])
    pred_train <- predict(boost_hitters, Hitters_train, n.trees = 1000)
    trainerr[i] <- mean((pred_train - Hitters_train$Salary)^2)
}
```

```r
plot(lambdas, trainerr, type = "b", xlab = "Shrinkage values", ylab =
"Training MSE")
```



(d) Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

```r
set.seed(1)
testerr <- rep(NA, length(lambdas))
for (i in 1:length(lambdas)) {
    boost_hitters <- gbm(Salary ~ ., data = Hitters_train, distribution =
"gaussian", n.trees = 1000, shrinkage = lambdas[i])
    pred_test <- predict(boost_hitters, Hitters_test, n.trees = 1000)
    testerr[i] <- mean((pred_test - Hitters_test$Salary)^2)
}
plot(lambdas, testerr, type = "b", xlab = "Shrinkage values", ylab = "Test
MSE")
```




```
min(testerr)

## [1] 0.2540265

lambdas[which.min(testerr)]

## [1] 0.07943282
```

Observation: The minimum test MSE is **0.25**, and is obtained for **??=0.079**

(e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

Implementing linear regression and ridge regression to compare MSE against boosting method

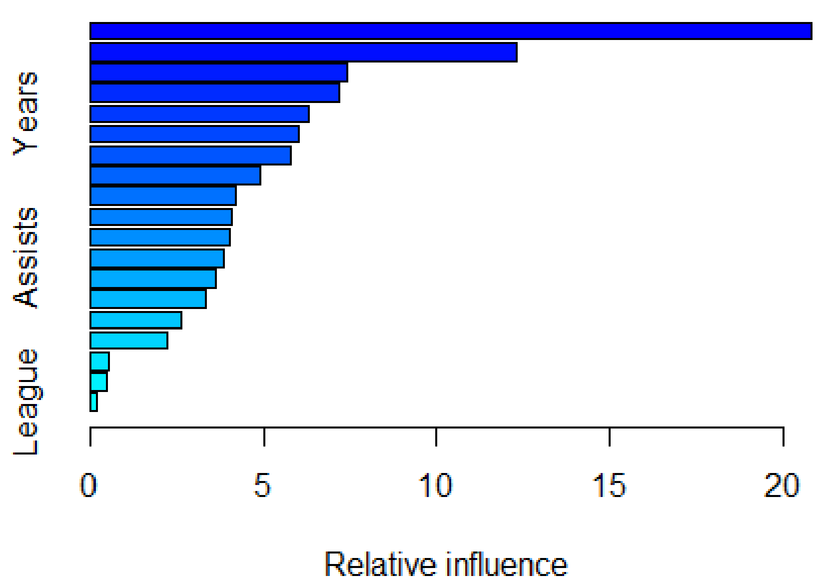
```
library(glmnet)

## Warning: package 'glmnet' was built under R version 3.3.3

## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.3.3

## Loaded glmnet 2.0-13
```

```r
fit1 <- lm(Salary ~ ., data = Hitters_train)
pred1 <- predict(fit1, Hitters_test)
mean((pred1 - Hitters_test$Salary)^2)

## [1] 0.4917959

x <- model.matrix(Salary ~ ., data = Hitters_train)
x_test <- model.matrix(Salary ~ ., data = Hitters_test)
y <- Hitters_train$Salary
fit2 <- glmnet(x, y, alpha = 0)
pred2 <- predict(fit2, s = 0.01, newx = x_test)
mean((pred2 - Hitters_test$Salary)^2)

## [1] 0.4570283
```

Observation: The test MSE for boosting (0.25) is lower than for linear regression(0.49) and ridge regression(0.45)

(f) Which variables appear to be the most important predictors in the boosted model?

```r
boost_hitters <- gbm(Salary ~ ., data = Hitters_train, distribution =
"gaussian", n.trees = 1000, shrinkage = lambdas[which.min(testerr)])
summary(boost_hitters)
```



```
##                 var    rel.inf
## CAtBat       CAtBat 20.8404970
## CRBI           CRBI 12.3158959
## Walks         Walks  7.4186037
## PutOuts     PutOuts  7.1958539
## Years         Years  6.3104535
## CWalks       CWalks  6.0221656
## CHmRun       CHmRun  5.7759763
## CHits         CHits  4.8914360
## AtBat         AtBat  4.2187460
## RBI             RBI  4.0812410
## Hits           Hits  4.0117255
## Assists     Assists  3.8786634
## HmRun         HmRun  3.6386178
## CRuns         CRuns  3.3230296
## Errors       Errors  2.6369128
## Runs           Runs  2.2048386
## Division   Division  0.5347342
## NewLeague NewLeague  0.4943540
## League       League  0.2062551
```

Observation: CAtBat is the most important variable.

(g) Now apply bagging to the training set. What is the test set MSE for this approach?

```
set.seed(1)
bag_hitters <- randomForest(Salary ~ ., data = Hitters_train, mtry = 19,
ntree = 500)
pred.bag <- predict(bag_hitters, newdata = Hitters_test)
mean((pred.bag - Hitters_test$Salary)^2)

## [1] 0.2313593
```

Observation: The test MSE for bagging is 0.23, which is slightly lower than the test MSE for boosting.