Assignment 8 – bagging & random forsests

Math 154, Computational Statistics
Fall 2015, Maria Martinez

Due: Tuesday, November 10, 2015, noon

NOT RELATED TO HW SCORE:

Total hours spent on assignment: 6

Number of different "sittings" to finish assignment: 4

Summary

This assignment extends ideas about classification and regression trees. First bagging is used to improve the variance of trees. Random Forests are an extension of bagging using a prediction which is an average over many trees which have been built on a subset of predictor variables.

Requisites

Read relevant sections of *An Introduction to Statistical Learning*, `http://www-bcf.usc.edu/~gareth/ISL/`. bagging and random forests (section 8.2) [no boosting].

Note that the lab in section 8.3 walks through much of the needed bagging and random forest code.

Assignment

1. In class we have discussed two reasons for using cross validation. 1. For model building. 2. For model assessment.

   (a) How is cross validation used for model building: in particular, for selecting a parameter (e.g., k in knn, α in the cost complexity when pruning trees in CART, m = number of variables to choose at each split in random forest)?

      i. Divide the set (1,..., n) into k subsets (i.e., folds) of roughly equal size.
      ii. For $k = 1, ...K$: Consider training on (x_i, y_i), and validating on (x_i, y_i), where i is in the training. For each value of the tuning parameter θ in $(\theta_1, ..., \theta_m)$, record the total error on the validation set:Using mean square error.

iii. For each tuning parameter value θ, compute the average error over all fold and chose the tuning parameter that minimizes the error.

(b) How is cross validation used for model assessment?
So wed like to know how well a model classifies observations, but if we test on the samples at hand, the error rate will be much lower than the models inherent accuracy rate. So instead, wed like to predict new observations that were not used to create the model. Here we use CV. One way of doing it is leave one out cross validation (LOOCV)

i. remove one observation
ii. build the model using the remaining n-1 points
iii. predict class membership for the observation which was removed
iv. repeat by removing each observation one at a time

(c) Write out the steps for the entire tree algorithm indicating how you could use cross validation TWICE within the same tree problem: once for modeling building and once for model assessment.

i. Partition the data in K_1 groups.
ii. Remove the first group, and train the data on the remaining $K_1 - 1$ groups.
iii. Use K_2-fold cross-validation (on the $K_1 - 1$ groups) to choose α. That is, divide the training observations into K_2 folds and find α that minimizes the error.
iv. Using the subtree that corresponds to the chosen value of α, predict the first of the K_1 hold out samples.
v. Repeat steps 2-4 using the remaining $K_1 - 1$ groups

2. Problem 5 in section 8.4 of *An Introduction to Statistical Learning.*

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of P(Class is Red$|X$):

$$0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75$$

There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in this chapter. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?

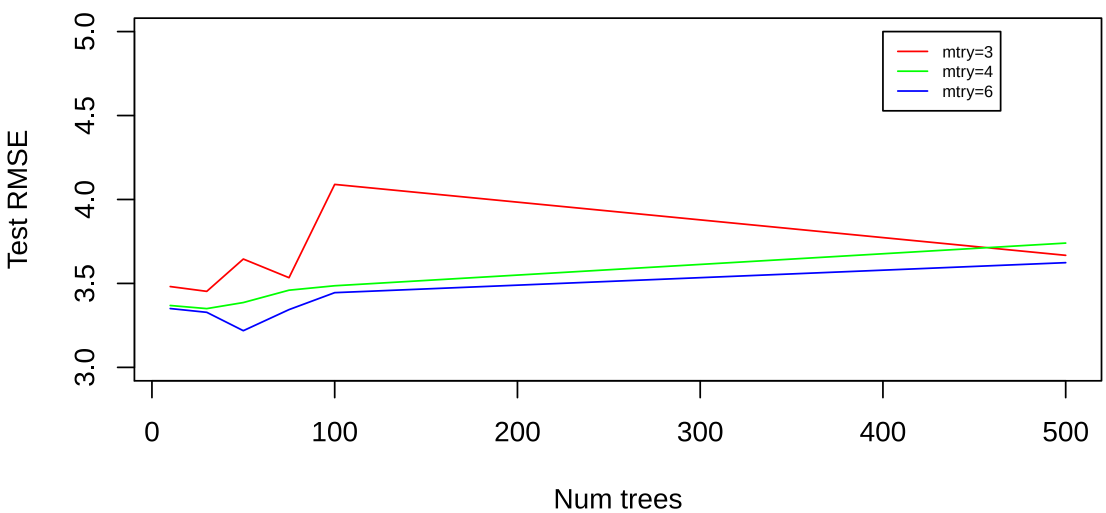
```
X = c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75);

# Majority vote approach
mean(X > 0.5) > 0.5

## [1] TRUE

# Classify based on the average probability
mean(X) > 0.5

## [1] FALSE
```

3. Problem 7 in section 8.4 of *An Introduction to Statistical Learning*.

   In the lab, we applied random forests to the Boston data using `mtry= 6` and using `ntree=25` and `ntree=500`. Create a plot displaying the test error resulting from random forests on this data set for a more comprehensive range of values for mtry and ntree. You can model your plot after Figure 8.10. Describe the results obtained.

```
library(ISLR)
library(MASS)
library(randomForest)
library(tree)
set.seed(10)

mtry  = c(3,4,6)
ntree = c(10,30,50,75,100,500)

train = sample(1:nrow(Boston), nrow(Boston)/2)
boston.test = Boston[-train,'medv']

getForrestError = function(mtryNum, numTree){

    rf.boston = randomForest(medv~.,data = Boston,subset = train,
                             mtry = mtryNum, ntree = numTree,
                             importance=TRUE)
    yhat.rf = predict(rf.boston, newdata = Boston[-train,])
    return(sqrt(mean((yhat.rf-boston.test)^2)))
}

# mtry = 3
mtry3 = c()
mtry3 = c(getForrestError(3,10)  , mtry3)
mtry3 = c(getForrestError(3,30)  , mtry3)
mtry3 = c(getForrestError(3,50)  , mtry3)
mtry3 = c(getForrestError(3,75)  , mtry3)
mtry3 = c(getForrestError(3,100) , mtry3)
mtry3 = c(getForrestError(3,500) , mtry3)

# mtry = 4
mtry4 = c()
mtry4 = c(getForrestError(4,10)  , mtry4)
mtry4 = c(getForrestError(4,30)  , mtry4)
mtry4 = c(getForrestError(4,50)  , mtry4)
mtry4 = c(getForrestError(4,75)  , mtry4)
mtry4 = c(getForrestError(4,100) , mtry4)
mtry4 = c(getForrestError(4,500) , mtry4)
```

```
# mtry = 5
mtry6 = c()
mtry6 = c(getForrestError(6,10)  , mtry6)
mtry6 = c(getForrestError(6,30)  , mtry6)
mtry6 = c(getForrestError(6,50)  , mtry6)
mtry6 = c(getForrestError(6,75)  , mtry6)
mtry6 = c(getForrestError(6,100) , mtry6)
mtry6 = c(getForrestError(6,500) , mtry6)

plot(ntree,mtry3,xlab="Num trees",
     ylim=c(3,5),ylab="Test RMSE",col = "red",type='l')
lines(ntree,mtry4 ,col = "green")
lines(ntree,mtry6 ,col = "blue")
legend(400, 5, cex = .59,sprintf("mtry=%g",mtry),
       lty = 1,col= c("red","green","blue"))
```



4. Problem 8.4.8 of *An Introduction to Statistical Learning.*

   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.

```
train = sample(1:nrow(Carseats),nrow(Carseats)/2)
Carseats.train = Carseats[train,]
Carseats.test  = Carseats[-train,]
```

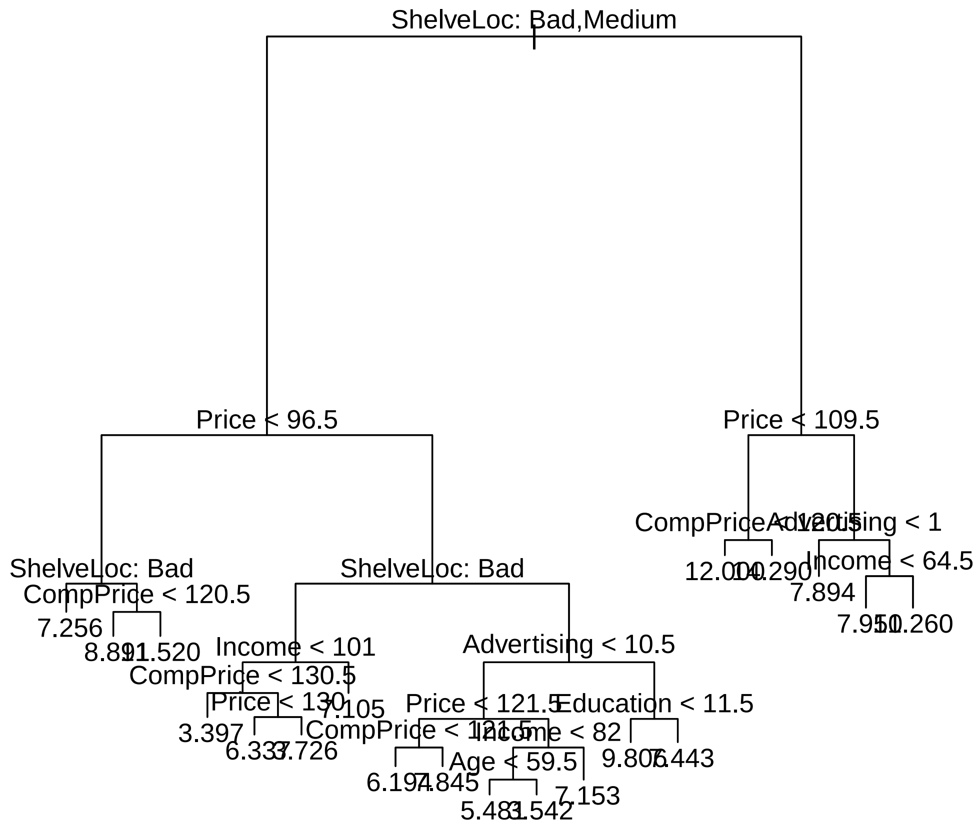
(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

Hint: try using `?Carseat` to find out more information about the data set.

```
tree.carseats = tree(Sales~.,Carseats.train)
summary(tree.carseats)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "CompPrice"   "Income"      "Advertising"
## [6] "Age"         "Education"
## Number of terminal nodes:  19
## Residual mean deviance:  2.148 = 388.8 / 181
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.73100 -0.88650 -0.02111  0.00000  0.96640  3.59900
```

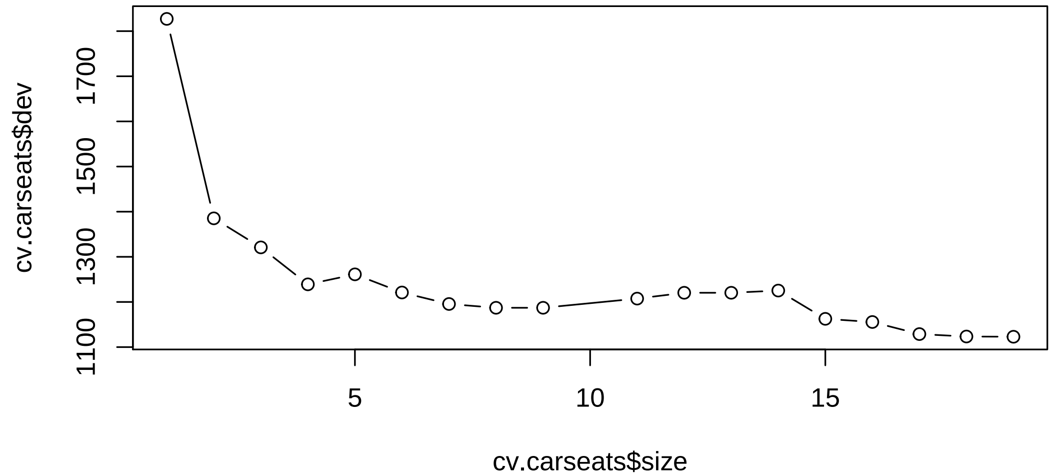
```
plot(tree.carseats);text(tree.carseats,pretty=0)
```




```r
pred = predict(tree.carseats,Carseats.test)
mean((Carseats.test$Sales - pred)^2)

## [1] 4.817759
```

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

```r
cv.carseats = cv.tree(tree.carseats, FUN = prune.tree)
plot(cv.carseats$size, cv.carseats$dev, type="b")
```




```
  minimum = which.min(cv.carseats$dev)

#5 is min
  prune.carseats = prune.tree(tree.carseats, best = minimum)
  plot(prune.carseats); text(prune.carseats, pretty = 0)

## Error in xy.coords(x, y, xlabel, ylabel, log):  'x' is a list, but does
not have components 'x' and 'y'
## Error in xy.coords(x, y, recycle = TRUE): 'x' is a list, but does not have
components 'x' and 'y'

  pred = predict(prune.carseats,Carseats.test)

## Error in UseMethod("predict"):  no applicable method for 'predict' applied
to an object of class "singlenode"

  mean((Carseats.test$Sales - pred)^2)

## [1] 4.817759
```

(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.

```
  bag.carseats = randomForest(Sales~.,data = Carseats,subset = train,
  mtry = ncol(Carseats)-1,importance =TRUE)
  importance(bag.carseats)

##                %IncMSE IncNodePurity
## CompPrice   28.876524    185.887175
## Income       9.047483     93.212845
```

```
## Advertising 14.793516    126.744353
## Population  -0.198644     59.432814
## Price       54.514744    502.152495
## ShelveLoc   65.544358    572.869151
## Age         13.134072    116.652972
## Education    2.941045     42.758318
## Urban        1.295811      6.527136
## US           5.171915     10.152769
```

```
#varImpPlot(bag.carseats)
pred = predict(bag.carseats,Carseats.test)
mean((Carseats.test$Sales - pred)^2)
```

```
## [1] 2.371646
```

Bagging improves the test MSE. The 3 most importatnt predictors are ShelveLoc, Price and CompPrice.

(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.

```
error = rep(0,6)

getErr = function(num=1){
  rf.carseats = randomForest(Sales~.,data=Carseats,
                             subset = train,mtry = num,importance=T)
  pred = predict(rf.carseats, Carseats.test)
  error[num] = mean((Carseats.test$Sales - pred)^2)
}

rf.carseats = randomForest(Sales~.,data=Carseats,
                           subset = train,importance=T)
importance(rf.carseats)
```

```
##               %IncMSE IncNodePurity
## CompPrice   15.286635     163.96981
## Income       6.706880     133.67453
## Advertising 12.848714     165.40835
## Population  -1.665737     101.67389
## Price       37.517740     412.82921
## ShelveLoc   39.924855     402.20094
## Age         12.096821     172.51782
## Education    1.606208      86.90153
## Urban       -2.078645      16.38904
## US           3.445732      15.51935
```

```
err = sapply(1:6, getErr); err

## [1] 4.204995 3.050337 2.674860 2.535897 2.394203 2.386104

mean(err)

## [1] 2.874399
```

Compared to otheres this has a higher MSE, but as m increaseses, the MSE goes down. Nevertheless, ShelveLoc, Price and CompPrice are still the most important variables.