Homework3

Xingyuan Chen

3/25/2019

1. Using basic statistical properties of the variance, as well as single- variable calculus, derive (5.6). In other words, prove that α given by (5.6) does indeed minimize $Var(\alpha X + (1-\alpha)Y)$.

Answer:
(5.6) is

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

Since

$$Var(\alpha X + (1-\alpha)Y) =$$
$$\alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)Cov(X,Y) =$$
$$\alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2\alpha(1-\alpha)\sigma_{XY}$$

To minimize $Var(\alpha X + (1-\alpha)Y)$,

$$\frac{dVar(\alpha X + (1-\alpha)Y)}{d\alpha} = 0$$
$$2\alpha\sigma_X^2 - 2(1-\alpha)\sigma_Y^2 - (4\alpha - 2)\sigma_{XY} = 0$$
$$\alpha\sigma_X^2 - (1-\alpha)\sigma_Y^2 - (2\alpha - 1)\sigma_{XY} = 0$$
$$\alpha(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}) = \sigma_Y^2 - \sigma_{XY}$$

so

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

3. We now review k-fold cross-validation.

(a) Explain how k-fold cross-validation is implemented.
Answer:
K-fold cross-validation involves randomly dividing the set of observations into k folds of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k-1 folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, $MSE_1, MSE_2, ..., MSE_k$. The k-fold CV estimate is computed by averaging these values.

(b) What are the advantages and disadvantages of k-fold cross-validation relative to:

i. The validation set approach?
K-fold's advantage: reduce the uncertainty of test error caused by different partition method of validation set, better estimate the actual test error of the model.
K-fold's disadvantage: it's a little more complex than validation set approach and need a little bit more computing power.

ii. LOOCV?
K-fold's advantage: take less computing power and less complex, with lower variance.
K-fold's disadvantage: has higher bias comparing to LOOCV.

5. In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. Split the sample set into a training set and a validation set.

```r
library(ISLR)
set.seed(1)
train=sample(10000,5000)
```

ii. Fit a multiple logistic regression model using only the training observations.

```r
glm.fit = glm(default ~ income + balance, data = Default, family = binomial,
              subset = train)
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.

```r
prob <- predict(glm.fit, Default[-train,], type="response")
pred <- ifelse(prob > 0.5, "Yes", "No")
table(pred, Default[-train,]$default)
```

```
##
## pred    No  Yes
##   No  4805  115
##   Yes   28   52
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```r
mean(pred != Default[-train, ]$default)
```

```
## [1] 0.0286
```

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```r
train=sample(10000,5000)
glm.fit = glm(default ~ income + balance, data = Default, family = binomial,
              subset = train)
prob <- predict(glm.fit, Default[-train,], type="response")
pred <- ifelse(prob > 0.5, "Yes", "No")
mean(pred != Default[-train, ]$default)
```

```
## [1] 0.0236
```

```r
train=sample(10000,5000)
glm.fit = glm(default ~ income + balance, data = Default, family = binomial,
              subset = train)
prob <- predict(glm.fit, Default[-train,], type="response")
pred <- ifelse(prob > 0.5, "Yes", "No")
mean(pred != Default[-train, ]$default)
```

```
## [1] 0.028
```

The test error seems around 2.6%.

5. 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, and 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?

Answer:
Majority vote: 4 are less than 0.5, 6 are greater than 0.5, so the majority is greater than 0.5, so the final classification would be red.
Average probability: The average probability is $0.45 < 0.5$, so the final classification would be green.

8. 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.

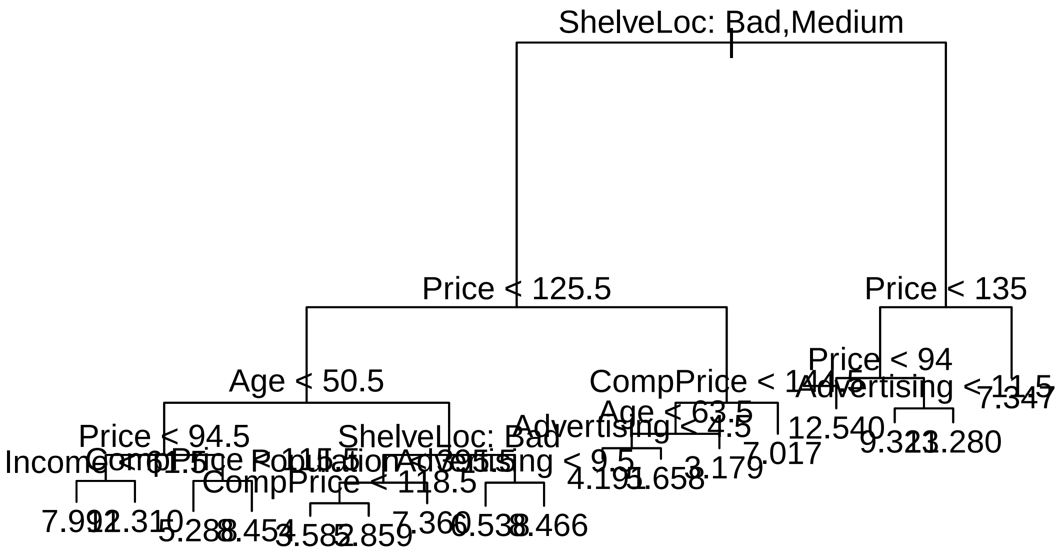
```
set.seed(2)
train = sample(400,200)
training = Carseats[train, ]
test = Carseats[-train, ]
```

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

```
library(tree)
tree.model = tree(Sales ~ ., data=training)
summary(tree.model)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = training)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"
## [6] "Population"  "Advertising"
## Number of terminal nodes:  17
## Residual mean deviance:  2.341 = 428.4 / 183
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.76700 -1.00900 -0.01558  0.00000  0.94900  3.58600
```

```
plot(tree.model)
text(tree.model, pretty = 0)
```



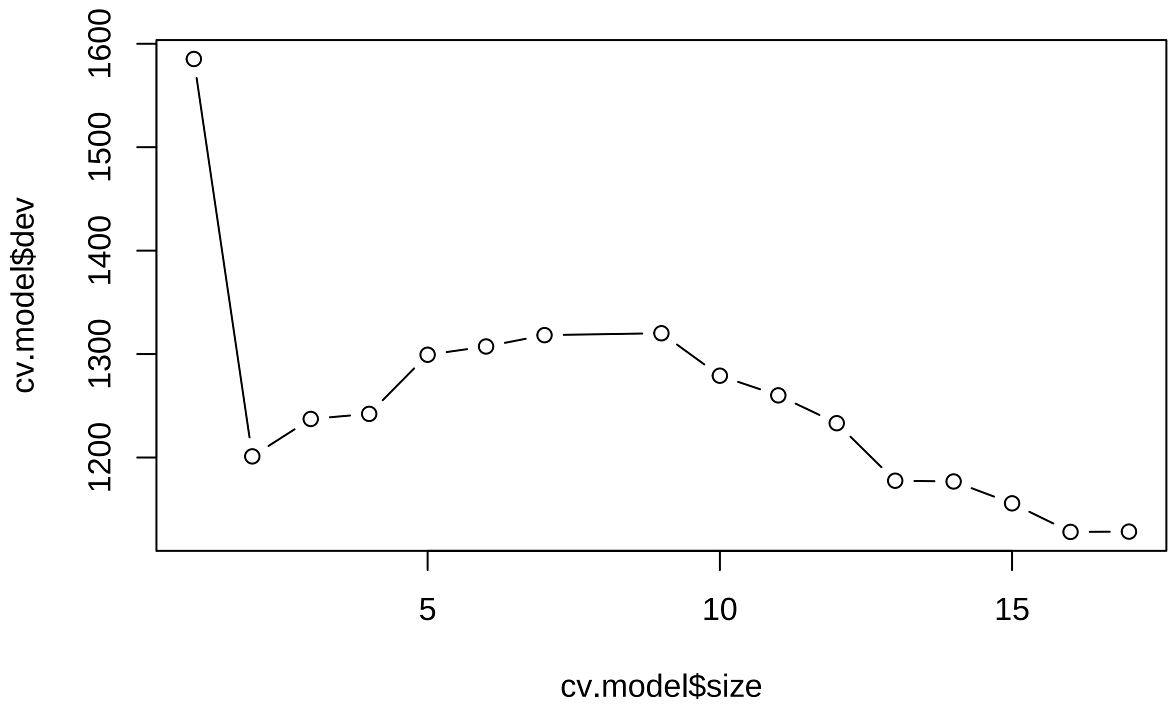
```r
pred = predict(tree.model, test)
mean((test$Sales - pred)^2)
```

```
## [1] 4.844991
```

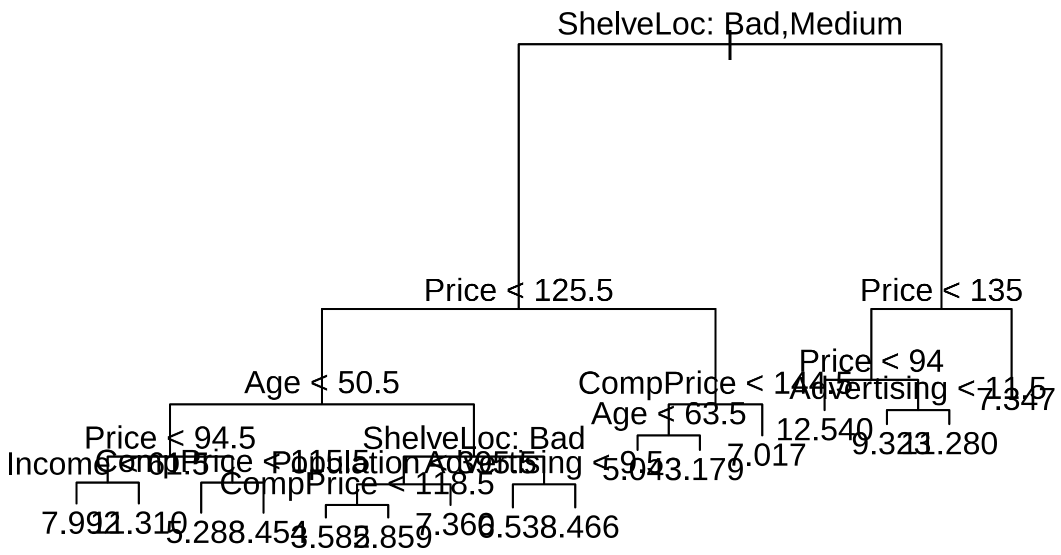
The test MSE is 4.85.

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

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



```r
pruned.model = prune.tree(tree.model, best = 16)
plot(pruned.model)
text(pruned.model, pretty = 0)
```



```r
pred = predict(pruned.model, test)
mean((test$Sales - pred)^2)
```

```
## [1] 4.893985
```

Pruning the tree didn't improve the test MSE.

(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)
```

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

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

```r
bag.model = randomForest(Sales~., data=training, mtry=10, importance=TRUE)
pred = predict(bag.model, test)
mean((test$Sales - pred)^2)
```

```
## [1] 2.369187
```

Bagging improved the MSE to 2.37.

```r
importance(bag.model)
```

```
##                %IncMSE IncNodePurity
## CompPrice   26.8209582    166.979714
## Income       2.5178689     70.424671
## Advertising 12.7943382     95.674806
## Population   1.5809962     66.767407
## Price       57.3318051    477.292357
## ShelveLoc   50.8691964    475.187526
## Age         12.9786136    126.420511
## Education   -1.8091675     37.001724
## Urban       -3.5410771      5.936702
## US          -0.8447167      6.800383
```

(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.

```
rf.model = randomForest(Sales~., data=training, mtry=3, importance=TRUE)
pred = predict(rf.model, test)
mean((test$Sales - pred)^2)
```

```
## [1] 2.961581
```

Random forests improved the MSE to 2.96.

```
importance(rf.model)
```

```
##                %IncMSE IncNodePurity
## CompPrice   13.9082873     143.70588
## Income      -1.1881180      95.40114
## Advertising  8.1765225     116.01780
## Population  -1.4440692     112.32290
## Price       37.5120708     386.90793
## ShelveLoc   39.9627269     350.46676
## Age         11.8981617     158.21652
## Education   -0.5830359      67.83125
## Urban       -0.7235525      14.13967
## US           0.5890730      16.34130
```

Here in random forest model, since it's a regression problem, we use m = p/3 as default.