Homework 03

Mingxue (Jacqueline) Li

2019-03

Exercises from Section 5.4 of ISLR

Exercise 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 $Var(\alpha X + (1-\alpha)Y)$.

Answers:

First of all, let's decompose $Var(\alpha X + (1-\alpha)Y)$:

$$Var(\alpha X + (1-\alpha)Y) = \alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)Cov(X,Y)$$

$$= \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2\alpha(1-\alpha)\sigma_{XY}$$

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

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

$$2\alpha\sigma_X^2 - 2(1-\alpha)\sigma_Y^2 - (4\alpha - 2)\sigma_{XY} = 0$$

$$\alpha\sigma_X^2 - (1-\alpha)\sigma_Y^2 - (2\alpha - 1)\sigma_{XY} = 0$$

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

Therefore:

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

Exercise 3

We now review k-fold cross-validation.

Answers:

(a) Explain how k-fold cross-validation is implemented.

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 trained on the remaining k-1 folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. Repeate k times for in each time, a different fold 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 overall performance estimate is computed by averaging these $MSEs$.

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

i. The validation set approach?

ii. LOOCV?

Compare to the validation set approach, k-fold CV is more complex and requires more computing power and time to execute. But on the other hand, it helps reduce the variance of test error due to the partition methods for the validation set and increases the estimate performance of the test error.

Compare to LOOCV, k-fold CV has higher bias but k-fold CV requires less computing power and time to execute. k-fold CV also has lower variance.

Exercise 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.

Answers:

(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.

```
library(ISLR)
nrow(Default)
```

```
## [1] 10000
```

```
set.seed(1)
train = sample(1:10000, 8000)
train.set = Default[train,]
validation.set = Default[-train,]
```

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

```
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.

```
prob = predict(glm.fit, validation.set, type = "response")
pred = ifelse(prob > 0.5, "Yes", "No")
table(pred, validation.set$default)
```

```
##
## pred    No  Yes
##   No  1929   45
##   Yes    7   19
```

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

```
mean(pred != validation.set$default)
```

```
## [1] 0.026
```

(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.

```
set.seed(1)
train = sample(10000, 5000)
train.set = Default[train,]
```

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

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

```
set.seed(1)
train = sample(10000, 6000)
train.set = Default[train,]
validation.set = Default[-train,]
glm.fit = glm(default ~ income + balance, data = Default, family = binomial, subset = train)
prob = predict(glm.fit, validation.set, type = "response")
pred = ifelse(prob > 0.5, "Yes", "No")
mean(pred != validation.set$default)
```

```
## [1] 0.02775
```

```
set.seed(1)
train = sample(10000, 7000)
train.set = Default[train,]
validation.set = Default[-train,]
glm.fit = glm(default ~ income + balance, data = Default, family = binomial, subset = train)
prob = predict(glm.fit, validation.set, type = "response")
pred = ifelse(prob > 0.5, "Yes", "No")
mean(pred != validation.set$default)
```

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

Conclusions: Validation error seems to be around 2.6%-2.8%. The more samples are in the training set, the better the validation error seems to be.

Exercises from Section 8.4 of ISLR

Exercise 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?

Answers:

Majority vote approach: There are 4 estimates that are less than 0.5 and 6 estimates that are greater than 0.5. Therefore, according to majority vote, the final classification would be red.

Average probability approach: The average probability is $\frac{0.1+0.15+0.2+0.2+0.55+0.6+0.6+0.65+0.7+0.75}{10} = 0.45$ which is smaller than 0.5. So the final classification would be green.

Exercise 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.

Answers:

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

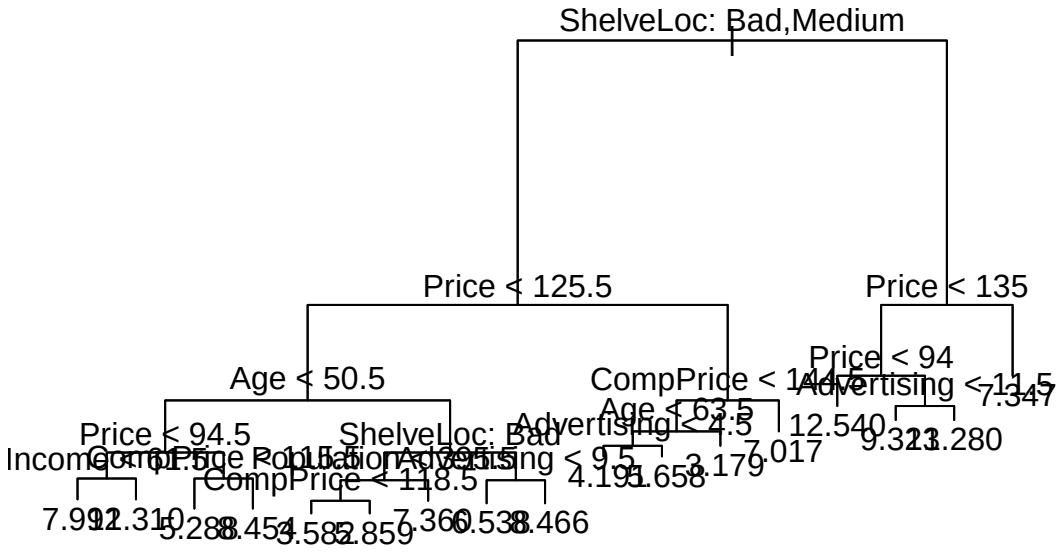
```
set.seed(2)
train2 = sample(1:nrow(Carseats),200)
training = Carseats[train2, ]
test = Carseats[-train2, ]
```

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

```
library(tree)
model = tree(Sales ~ ., data=training)
summary(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(model)
text(model, pretty=0)
```



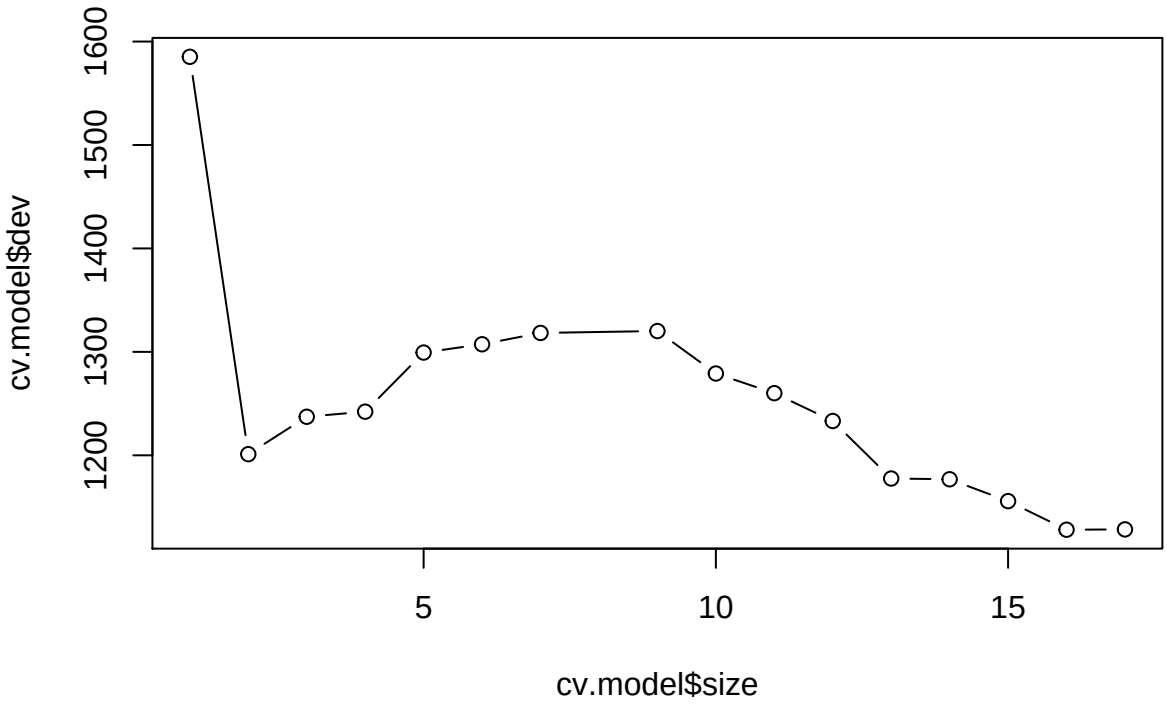

```
pred = predict(model, test)
mean((test$Sales - pred)^2)
```

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

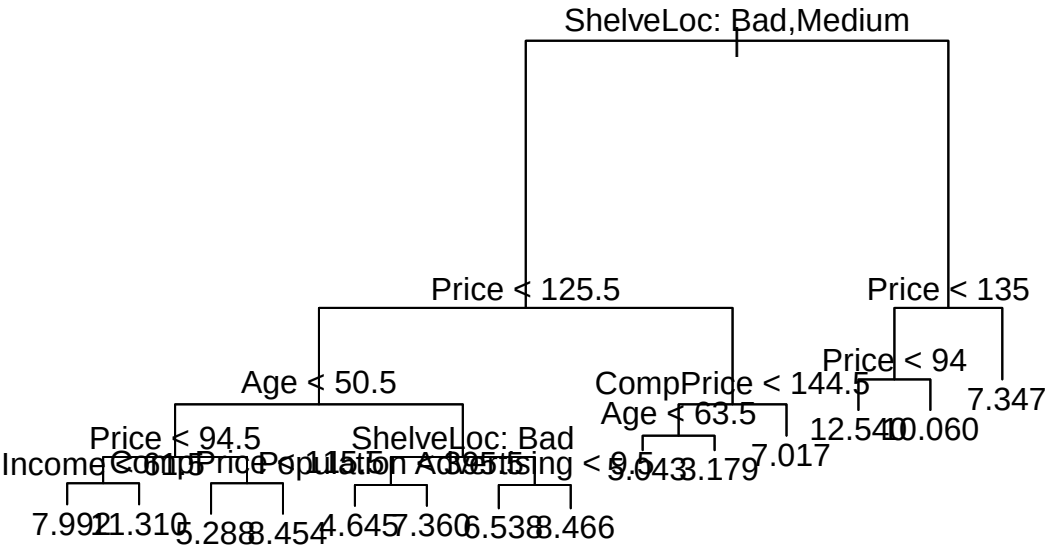
Conclusions: The test MSE is 4.845.

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

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




```
new.model = prune.tree(model, best = 14)
plot(new.model)
text(new.model, pretty = 0)
```




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

```
## [1] 4.844325
```

Conclusions: Pruning the tree did not 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.

```
library(randomForest)
```

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

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

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

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

```
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
```

Conclusions: The test MSE is about 2.4 and most important variables are `Price`, `ShelveLoc` 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.

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

```
## [1] 2.401699
```

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

```
##                %IncMSE IncNodePurity
## CompPrice   26.9820468    170.935188
## Income       2.5080170     72.277923
## Advertising 12.6045493     94.291541
## Population   2.1808267     66.579843
## Price       54.1613165    486.190554
## ShelveLoc   52.5721311    454.927145
## Age         13.1081099    126.858189
## Education   -3.3223932     38.831450
## Urban       -1.9420852      5.382564
## US          -0.4036435      5.991532
```

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

```
## [1] 2.950447
```

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

```
##                %IncMSE IncNodePurity
## CompPrice   14.0911906     145.40781
## Income      -1.1102414      98.25475
## Advertising 10.4214728     109.15008
## Population   1.9755846     108.60396
## Price       36.9976267     391.78465
## ShelveLoc   38.8903246     359.98055
## Age         11.0389535     163.00516
## Education   -3.6236749      64.74610
## Urban       -0.3546517      14.13217
## US           2.4737739      17.70364
```

Conclusions: The test MSE is around 2.4 and 2.9 while m represents the number of variables randomly sampled as candidates at each split. The greater the m is, the better performance the model would have. Despite of the low MSE we will get from greater m, we can also suffer from overfitting.