1 Chapter 6, Exercise 3 (p. 260)

a

iv. Steadily decrease.

As s increases, the model becomes more flexible and fits the traning data better, so the training RSS always decreases.

b

ii. Decrease initially, and then eventually start increasing in a U shape.

The test RSS decreases first, and then increases. Increases of s generates a closer fit before overfitting.

c

iii. Steadily increase.

The variance increases as s increases because the model relies on the input data more.

d

iv. Steadily decrease.

The bias decreases as s increases because the model has less constraints and becomes more flexible.

e

v. Remain constant.

The irreducible error depends on the distribution of ϵ, regardless of the model.

2 Chapter 6, Exercise 4 (p. 260)

a

iii. Steadily increase.

As we increase λ, the input data plays a smaller role in the model and β_1, β_2, $\cdots$ β_j decrease to 0. Therefore, the training RSS will increase.

b

ii. Decrease initially, and then eventually start increasing in a U shape.

The test RSS decreases first as increasing λ make some β_j close to zero, because overfitting is reduced. As λ continues to increase, the model becomes too weak and test RSS increases.

c

iv. Steadily decrease.

As we increase λ, the model relies on the input data less. So the variance is reduced.

d

iii. Steadily increase.

As we increase λ, the model relies on the input data less. More and more β_j becomes zero and the model becomes simpler. Therefore, the bias increases.

e

v. Remain constant.

The irreducible error depends on the distribution of ϵ, regardless of the model.

3 Chapter 6, Exercise 9 (p. 263)

Don't do parts (e), (f), and (g).

```
library(ISLR)
data(College)
attach(College)
```

```
Warning message:
"package 'ISLR' was built under R version 3.4.2"
```

a

```
set.seed(2333)
n = nrow(College)
tid = sample(1:n, n/2)
train = College[tid,]
test = College[-tid,]
# summary(train)
# summary(test)
```

b

```
lm.fit = lm(Apps~., data=train)
# summary(lm.fit)
lm.pred = predict(lm.fit, test)
mean((test[, "Apps"] - lm.pred)^2)
```

1566742.06897896

c

```
library(glmnet)
```

```
Warning message:
“package ‘glmnet’ was built under R version 3.4.4”Loading required package:
Matrix
Loading required package: foreach
Warning message:
“package ‘foreach’ was built under R version 3.4.3”Loaded glmnet 2.0-16
```

```
train_mat = model.matrix(Apps~., data=train)
test_mat = model.matrix(Apps~., data=test)
ridge.fit = cv.glmnet(train_mat, train$Apps, alpha=0)
ridge.lambda = ridge.fit$lambda.min
ridge.lambda
ridge.pred = predict(ridge.fit, s=ridge.lambda, newx=test_mat)
mean((test[, "Apps"] - ridge.pred)^2)
```

370.478646853595

2455351.35744938

d

```
lasso.fit = cv.glmnet(train_mat, train$Apps, alpha=1)
lasso.lambda = lasso.fit$lambda.min
lasso.lambda
lasso.pred = predict(lasso.fit, s=lasso.lambda, newx=test_mat)
mean((test[, "Apps"] - lasso.pred)^2)
lasso.coef = predict(lasso.fit, type="coefficients", s=lasso.lambda)
[1:ncol(College),]
# lasso.coef
length(lasso.coef[lasso.coef != 0])
```

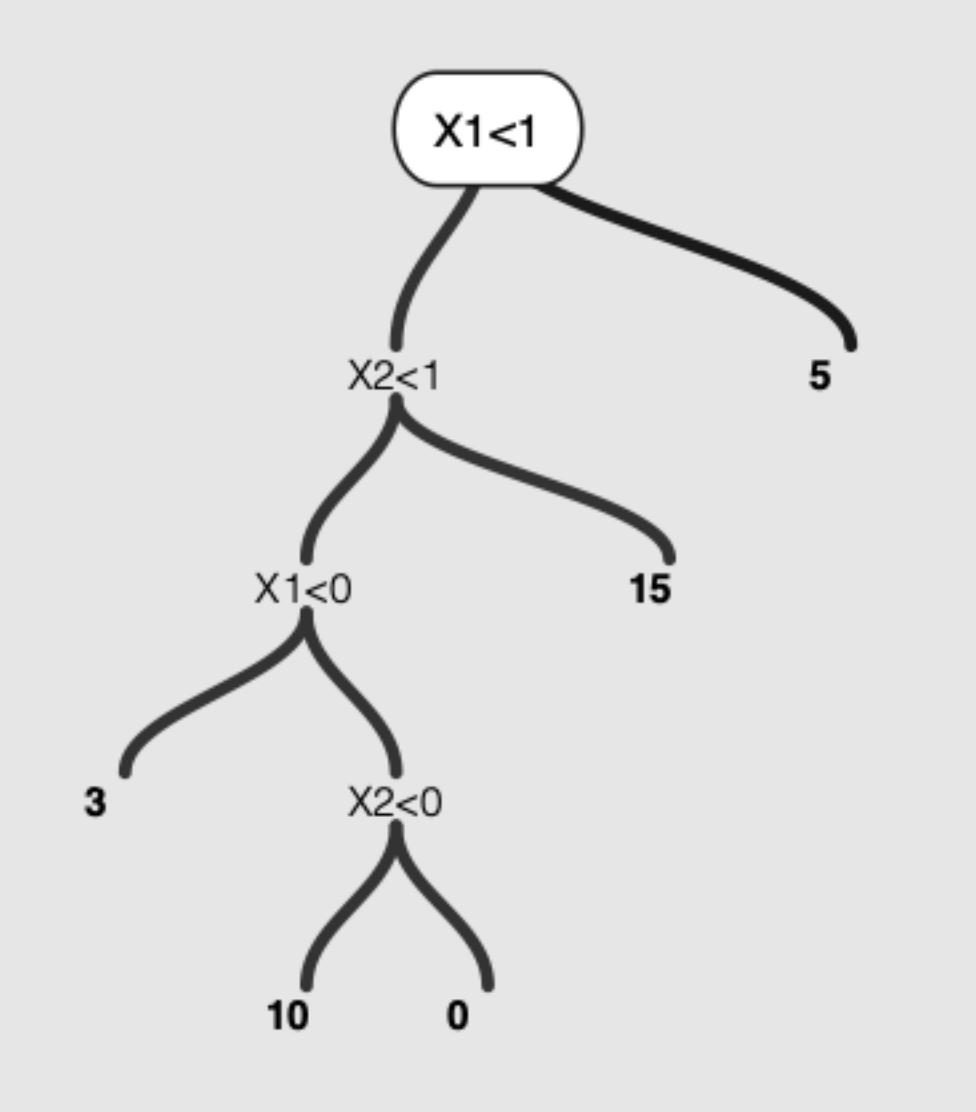
12.7091561508229

1663737.36939639

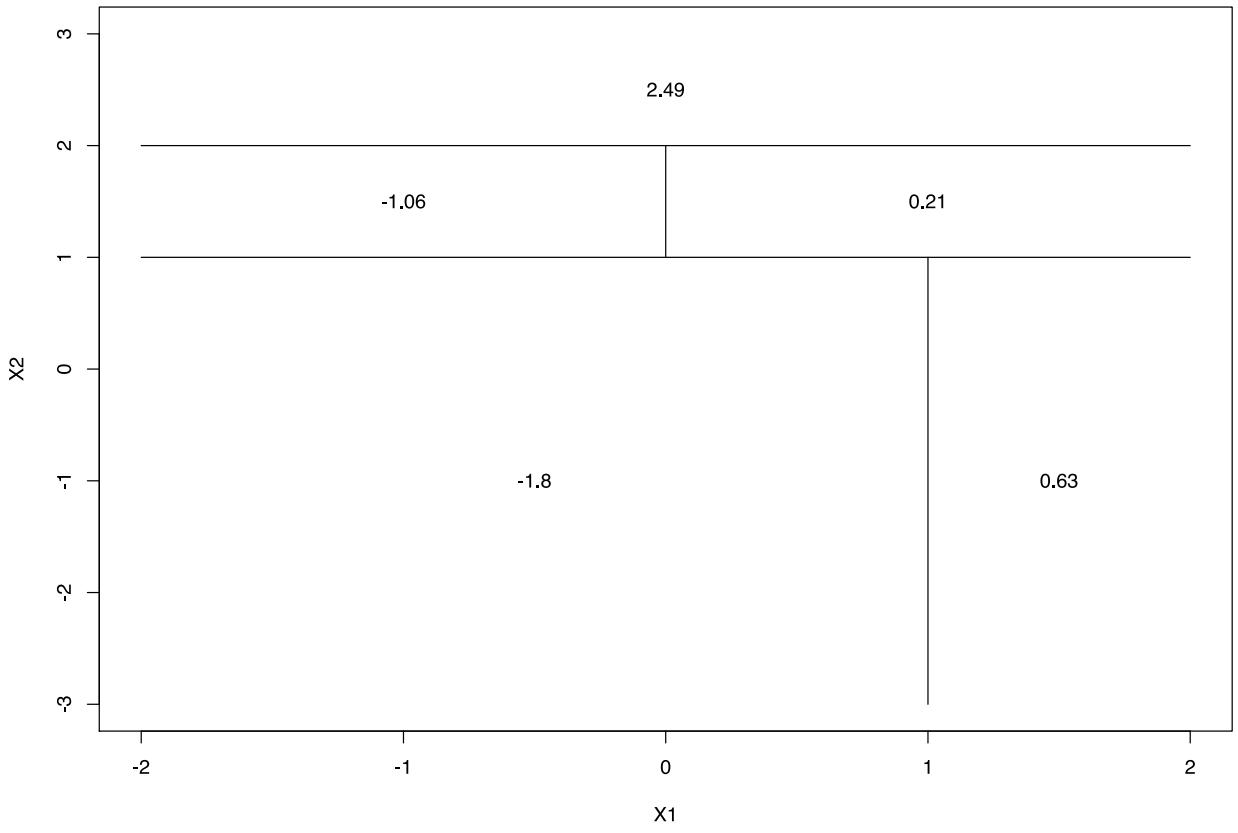
15

4 Chapter 8, Exercise 4 (p. 332)

a



b



5 Chapter 8, Exercise 8 (p. 333)

```
library(ISLR)
data(Carseats)
attach(Carseats)
```

a

```
set.seed(1)
n = nrow(Carseats)
tid = sample(1:n, n/2)
train = Carseats[tid,]
test = Carseats[-tid,]
# summary(train)
# summary(test)
```

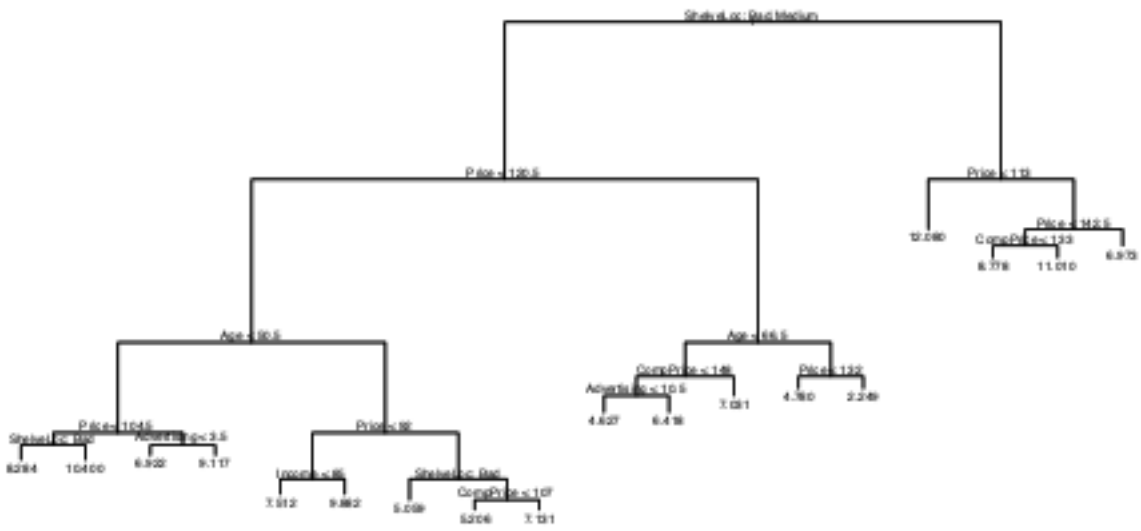
b

```
library(tree)
tree.fit = tree(Sales ~ ., data = train)
summary(tree.fit)
plot(tree.fit)
text(tree.fit, pretty = 0)
tree.pred = predict(tree.fit, test)
mean((test$Sales - tree.pred)^2)
```

```
Warning message:
“package ‘tree’ was built under R version 3.4.4”
```

```
Regression tree:
tree(formula = Sales ~ ., data = train)
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
```

4.14889745049246

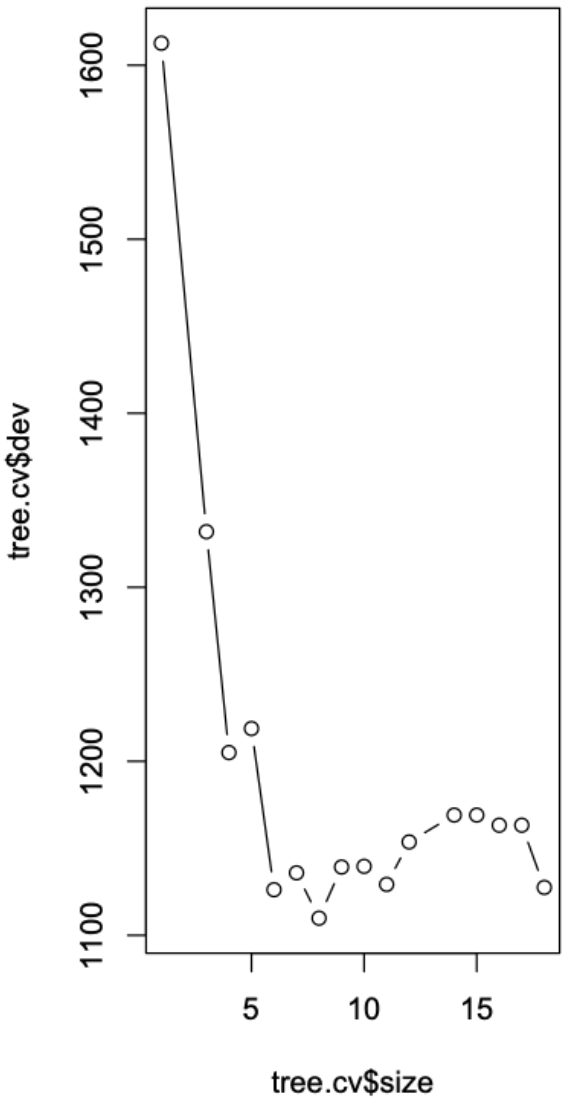


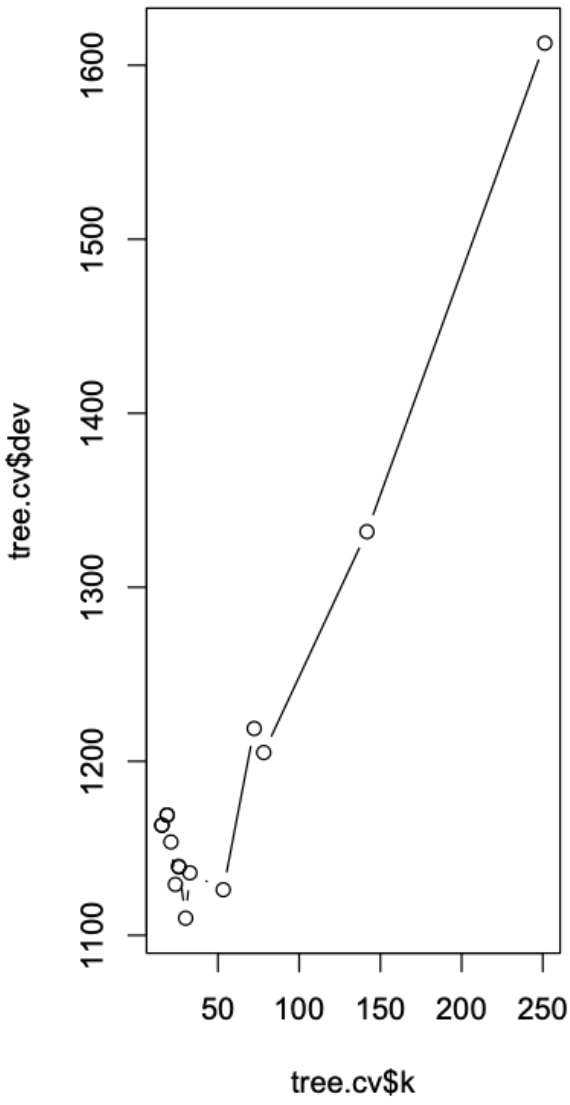
Brief Interpretation:

- For ShelveLoc, whether it's bad/medium is important.
- The price and age are predictors of great importance.
- For good ShelveLoc, less factors are needed to predict.
- CompPrice and Advertising are also useful predictors.
- ...

c

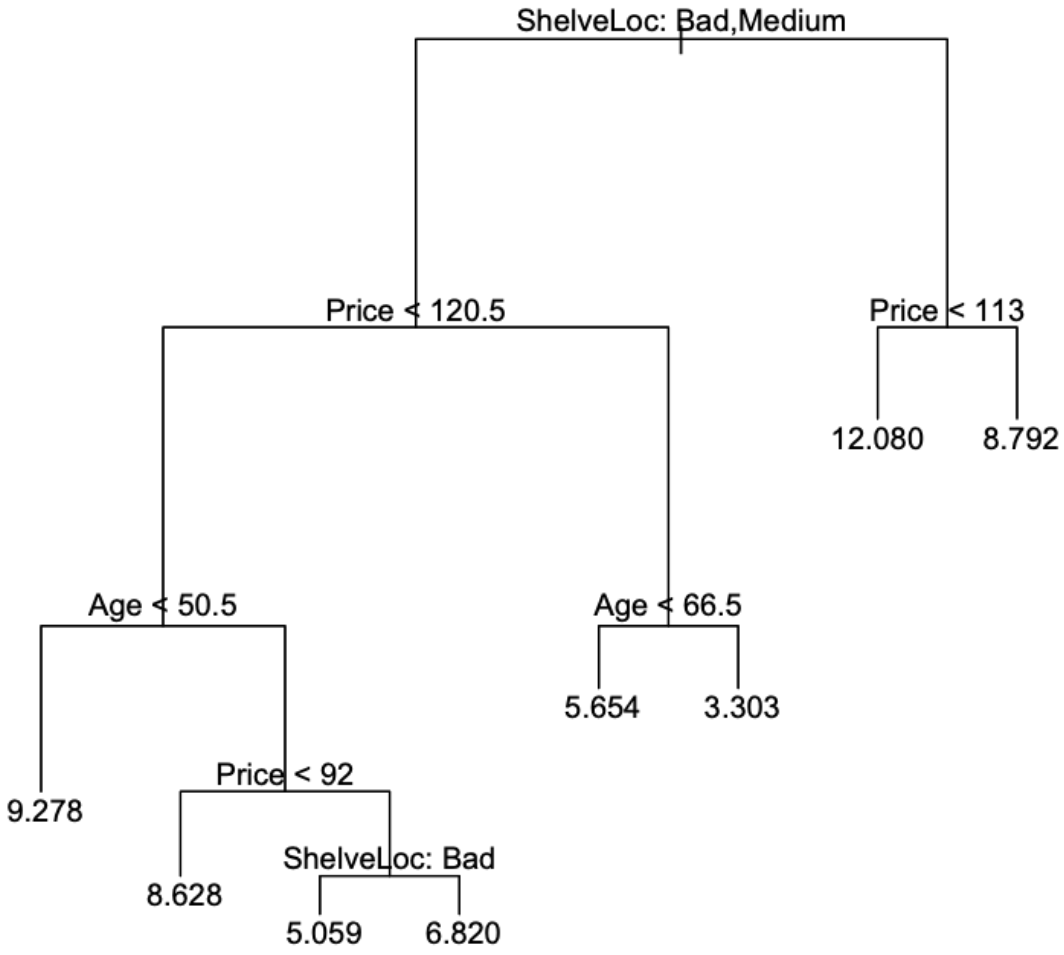
```
set.seed(1)
tree.cv = cv.tree(tree.fit)
par(mfrow = c(1, 2))
plot(tree.cv$size, tree.cv$dev, type = "b")
plot(tree.cv$k, tree.cv$dev, type = "b")
# tree.cv
```





Best size for the tree is 8.

```
tree.pruned = prune.tree(tree.fit, best = 8)
plot(tree.pruned)
text(tree.pruned, pretty = 0)
```




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

5.09085018184689

Pruning the tree increases the test MSE.

d

```
library(randomForest)
```

```
Warning message:
“package ‘randomForest’ was built under R version 3.4.4”randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
bag.fit = randomForest(Sales ~ ., data = train, mtry = 10, ntree = 300,
    importance = T)
bag.pred = predict(bag.fit, test)
mean((test$Sales - bag.pred)^2)
```

2.58062582085615

```
importance(bag.fit)
```

	%IncMSE	IncNodePurity
CompPrice	13.503782	128.303295
Income	3.391201	80.808139
Advertising	13.570680	120.163245
Population	1.043478	63.157960
Price	41.621536	504.890215
ShelveLoc	33.631952	319.316806
Age	15.802791	187.304102
Education	2.381255	39.993564
Urban	-1.616887	8.876977
US	5.399367	17.180459

Use bagging method, we have a better MSE 2.58. The most important variables are `Price` and `ShelveLoc`, followed by `Age`, `CompPrice` and `Advertising`.

e

```
set.seed(1)
rf.fit = randomForest(Sales ~ ., data = train, mtry = 5, ntree = 300,
    importance = T)
rf.pred = predict(rf.fit, test)
mean((test$Sales - rf.pred)^2)
importance(rf.fit)
```

2.90022769413291

	%IncMSE	IncNodePurity
CompPrice	8.1099119	129.65065
Income	3.7060291	108.72545
Advertising	9.8747924	132.37284
Population	0.2013809	83.94597
Price	35.7637894	459.96381
ShelveLoc	30.0537396	277.72879
Age	12.0634040	198.20101
Education	0.8831465	52.49984
Urban	0.3487895	10.64666
US	3.1519968	23.83161

```
set.seed(1)
rf.fit = randomForest(Sales ~ ., data = train, mtry = 6, ntree = 300,
    importance = T)
rf.pred = predict(rf.fit, test)
mean((test$Sales - rf.pred)^2)
```

2.71823568196124

```
set.seed(1)
rf.fit = randomForest(Sales ~ ., data = train, mtry = 7, ntree = 300,
    importance = T)
rf.pred = predict(rf.fit, test)
mean((test$Sales - rf.pred)^2)
```

2.65826742049745

```
set.seed(1)
rf.fit = randomForest(Sales ~ ., data = train, mtry = 8, ntree = 300,
    importance = T)
rf.pred = predict(rf.fit, test)
mean((test$Sales - rf.pred)^2)
```

2.65822227373281

```
set.seed(1)
rf.fit = randomForest(Sales ~ ., data = train, mtry = 9, ntree = 300,
    importance = T)
rf.pred = predict(rf.fit, test)
mean((test$Sales - rf.pred)^2)
```

2.61719983416528

```
set.seed(1)
rf.fit = randomForest(Sales ~ ., data = train, mtry = 10, ntree = 300,
    importance = T)
rf.pred = predict(rf.fit, test)
mean((test$Sales - rf.pred)^2)
```

2.58062582085615

We have `Price` and `ShelveLoc` are the most important variables, followed by `Age`, `CompPrice`, `Advertising` and `income`.

As m increases from 5 to 10, the MSE decreases.