HW3 Tianchun Jiang

SUID ending in 0710

tcjiang108@gmail.com

Problem 1

Chapter 6, Exercise 3 (p. 260).

a)

Steadily decrease. At s=0, all estimators are equal to zero and we have a null model. As s increases, the estimators approach their least square estimate values and the RSS decreases to what ordinary least squares would yield.

b)

Decrease initially, then eventually start increasing. At s=0 and null model, test RSS will be high as the model has very high bias. As s increases there will be a point where the model fits the data well and test RSS decreases, before test RSS increases again due to overfitting and high variance

c)

Steadily increase. At s=0, the model will output a constant value and there is very low variance. As s increases the model becomes more flexible and variance increases.

d)

Steadily decrease. As s increases the model becomes more flexible and squared bias will fall.

e)

Remain constant. Irreducible error is independent of model coefficients and shrinknage methods.

Problem 2

Chapter 6, Exercise 4 (p. 260).

a)

Steadily increase: at lambda=0, the Beta estimates are at their ordinary least squares value. As lamdba increases the estimators approach 0 e.g. the null model thus training RSS steadily increases with lambda

b)

Decreases initially, then eventually increasing in a U shape. At lambda = 0, all Beta estimates are uneffected from their ordinary least squared model values. If there is overfitting, as lambda increases the beta values move towards zero and the overfitting is reduced, thus there will be a point where the model is optimal and test RSS is at its low point. However, as lambda increases further the model becomes underfit and test RSS will increase.

c)

Steadily decreases; As lambda increases from 0 the model becomes simpler, less flexible and variance will fall

d)

Steadily increases; As lambda increases from 0 the model goes from having estimator values similar to an ordinary least squared model all the way to a null model. Along the way the model becomes underfit and bias increases.

e)

Remain constant. Irreducible error is independent of model coefficients and shrinknage methods.

Problem 3

Chapter 6, Exercise 9 (p. 263). Don't do parts (e), (f), and (g).

a)

```r
set.seed(100)
train <- sample(1:nrow(College), nrow(College)/2)
test <- (-train)
train_college  <- College[train, ]
test_college <- College[test, ]

dim(train_college)
```

```
## [1] 388  18
```

```r
dim(test_college)
```

```
## [1] 389  18
```

```r
dim(College)
```

```
## [1] 777  18
```

b)

```r
linear_model <- lm(Apps~., data=train_college)
pred <- predict(linear_model, test_college)
sprintf('Test RSS of linear model: %0.2f',mean((test_college[,'Apps'] - pred)^2))
```

```
## [1] "Test RSS of linear model: 1355556.91"
```

c)

```r
train_mat <- model.matrix(Apps~., data=train_college)
test_mat <- model.matrix(Apps~., data=test_college)
grid <- 10^seq(10, -2, length=100)
ridge_mod <- cv.glmnet(train_mat, train_college[,'Apps'], alpha=0, lambda=grid, thresh =1e-12)
optimal_lambda <- ridge_mod$lambda.min
pred <- predict(ridge_mod, newx=test_mat, s=optimal_lambda)
sprintf('Test RSS of ridge regression model: %0.2f', mean((test_college[,'Apps'] - pred)^2))
```

```
## [1] "Test RSS of ridge regression model: 1405132.49"
```

d)

```r
lasso_mod <- cv.glmnet(train_mat, train_college[,'Apps'], alpha=1, lambda=grid, thresh =1e-12)
optimal_lambda_lasso <- lasso_mod$lambda.min
pred <- predict(lasso_mod, newx=test_mat, s=optimal_lambda_lasso)
sprintf('Test RSS of LASSO model: %0.2f', mean((test_college[,'Apps'] - pred)^2))
```

```
## [1] "Test RSS of LASSO model: 1421633.82"
```

Problem 4

Chapter 8, Exercise 4 (p. 332).

a)

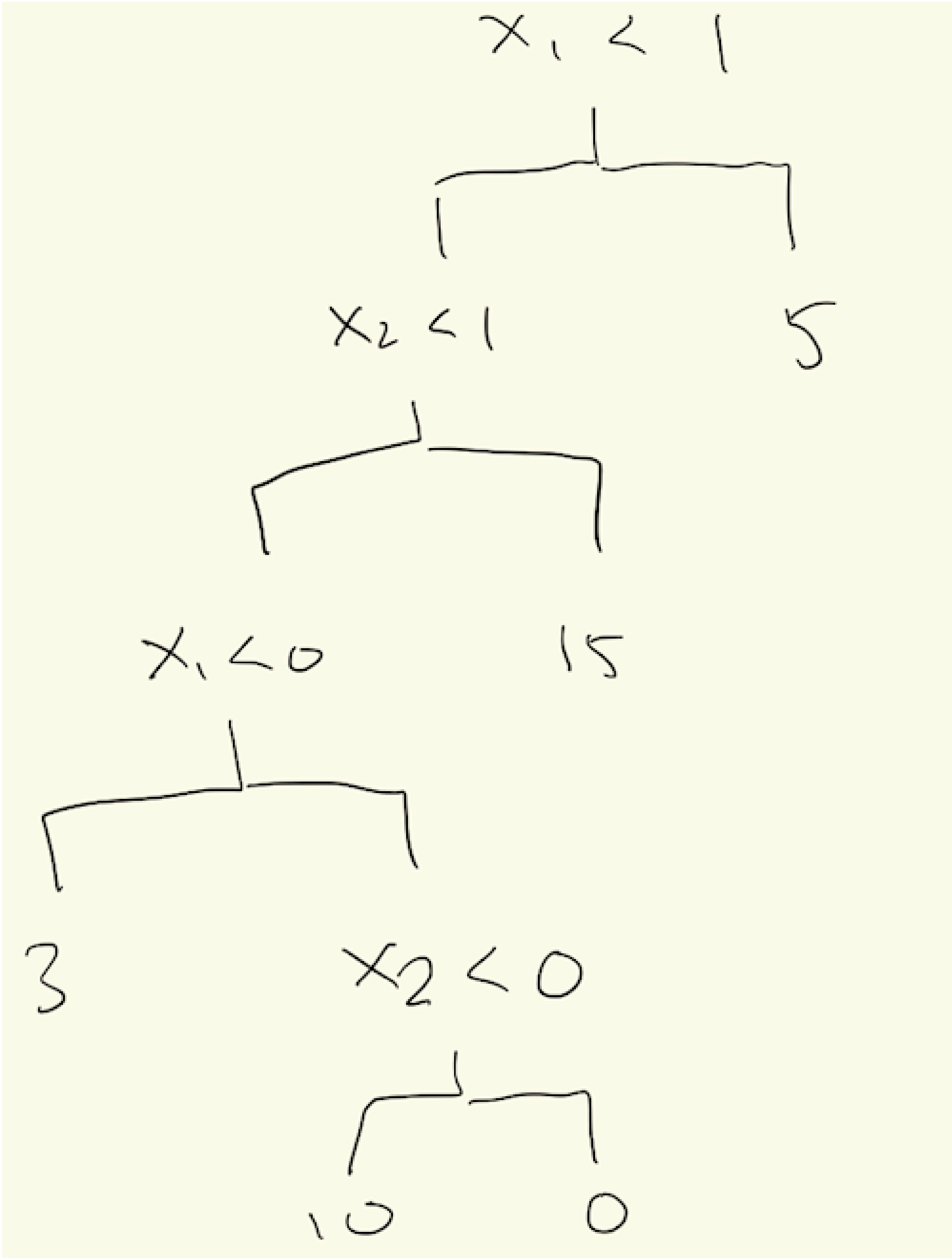


Figure 1: Tree for problem 4a

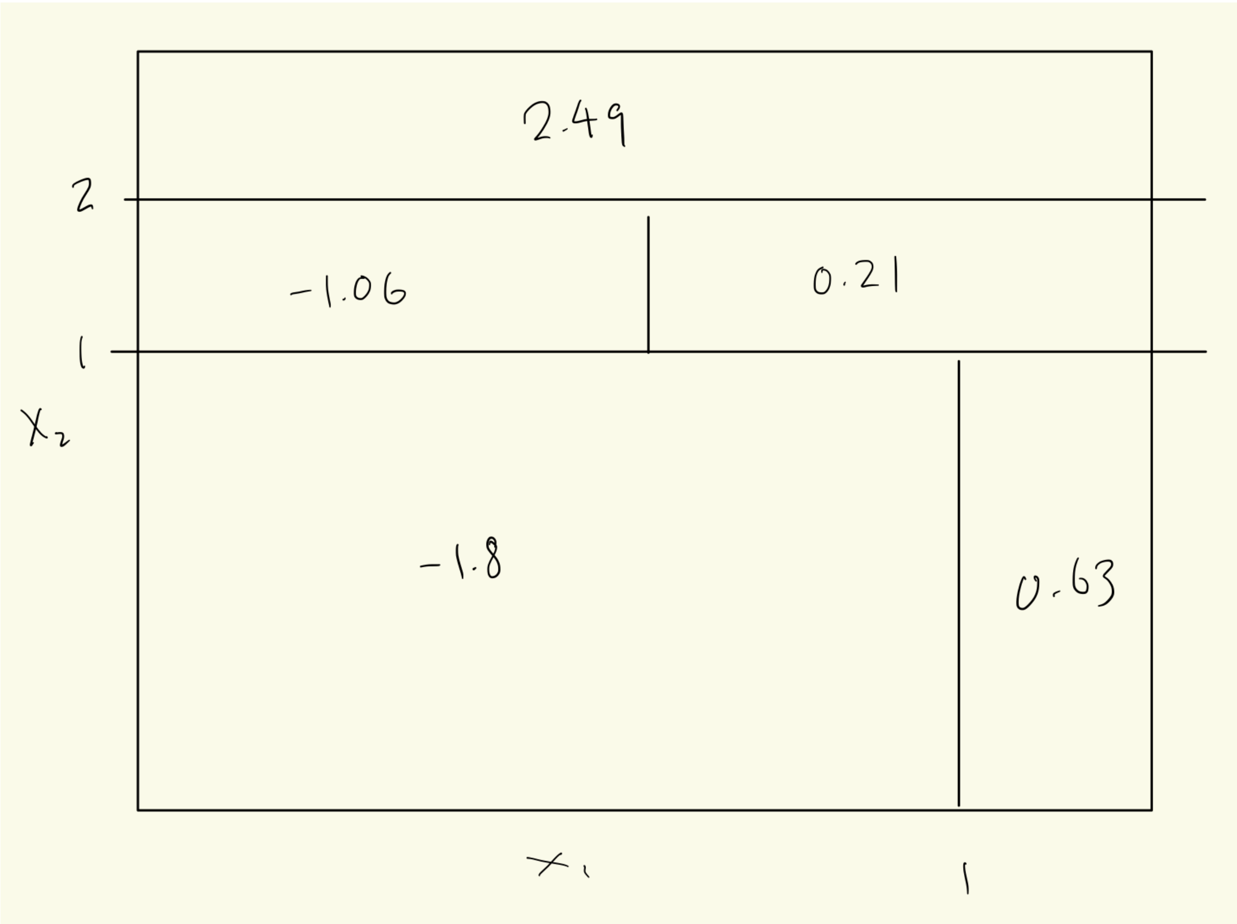


Figure 2: Tree for problem 4b

b)

Problem 5

Chapter 8, Exercise 8 (p. 333).

a)

```
set.seed(100)
attach(Carseats)
train <- sample(1:nrow(Carseats), nrow(Carseats)/2)
test <- (-train)
train_carseats  <- Carseats[train, ]
test_carseats <- Carseats[test, ]

dim(train_carseats)
```

```
## [1] 200  11
```

```
dim(test_carseats)
```

```
## [1] 200  11
```

```
dim(Carseats)
```

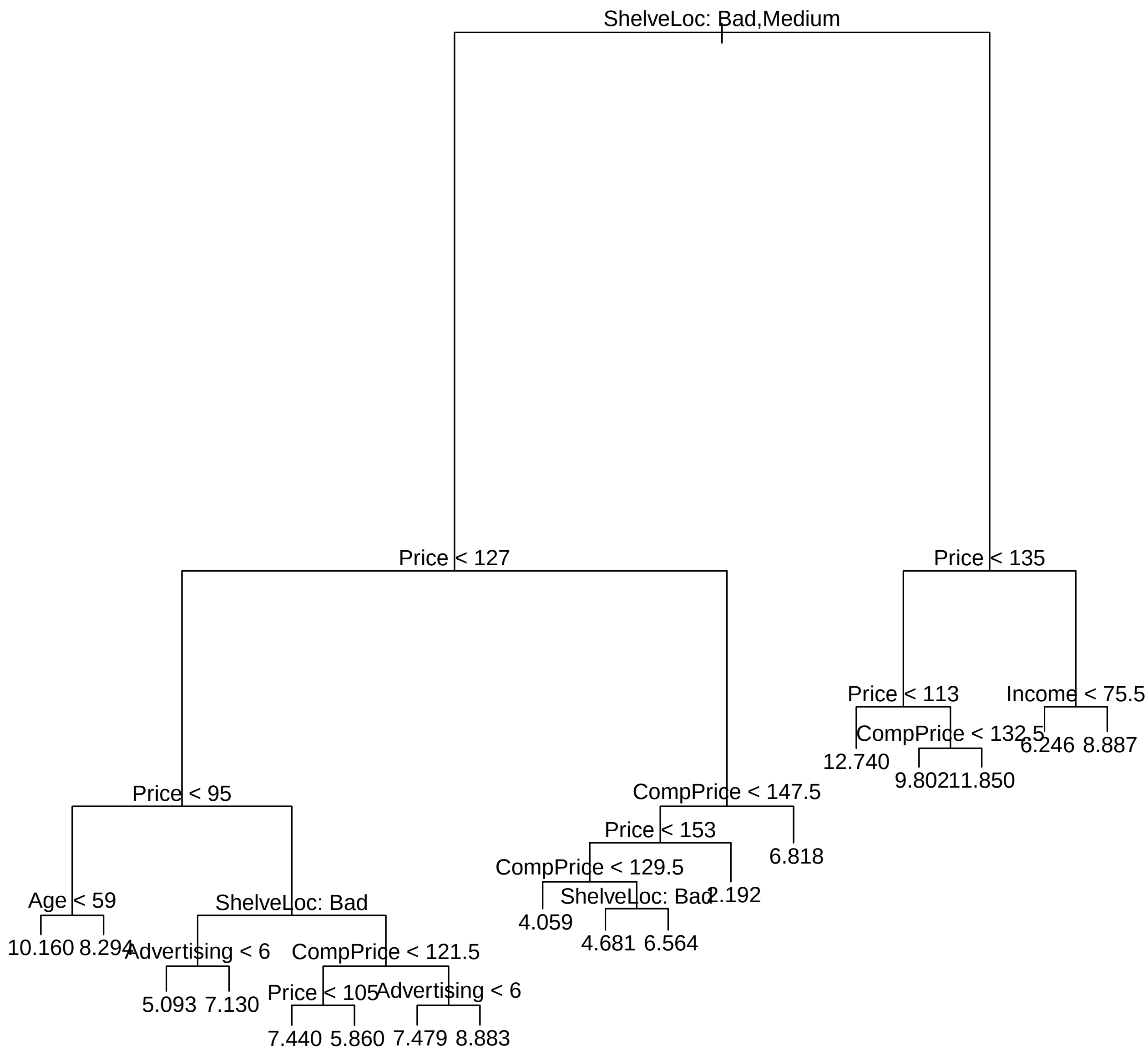
```
## [1] 400  11
```

b)

```
tree_carseats <- tree(Sales~., data=train_carseats)
# summary(Carseats)
summary(tree_carseats)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = train_carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "CompPrice"
## [6] "Income"
## Number of terminal nodes:  18
## Residual mean deviance:  1.924 = 350.2 / 182
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.27800 -0.96670  0.04631  0.00000  0.77910  3.52600
```

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



Carseats is a dataset of child carseat sales at 400 stores. Only six of the variables in Carseats have been used in constructing the tree. ShelveLoc is a factor indicating the quality of shelving locations for carseats at each store. The tree indicates that bad and medium shelving location qualities corresponds to lower carseat sales. The tree predicts highest sales of 12.74k carseats at stores with good shelving locations and a price of less than $113.

```
pred <- predict(tree_carseats, test_carseats)
sprintf('the test MSE obtained is %0.2f', mean((test_carseats$Sales - pred)^2))
```

```
## [1] "the test MSE obtained is 4.94"
```

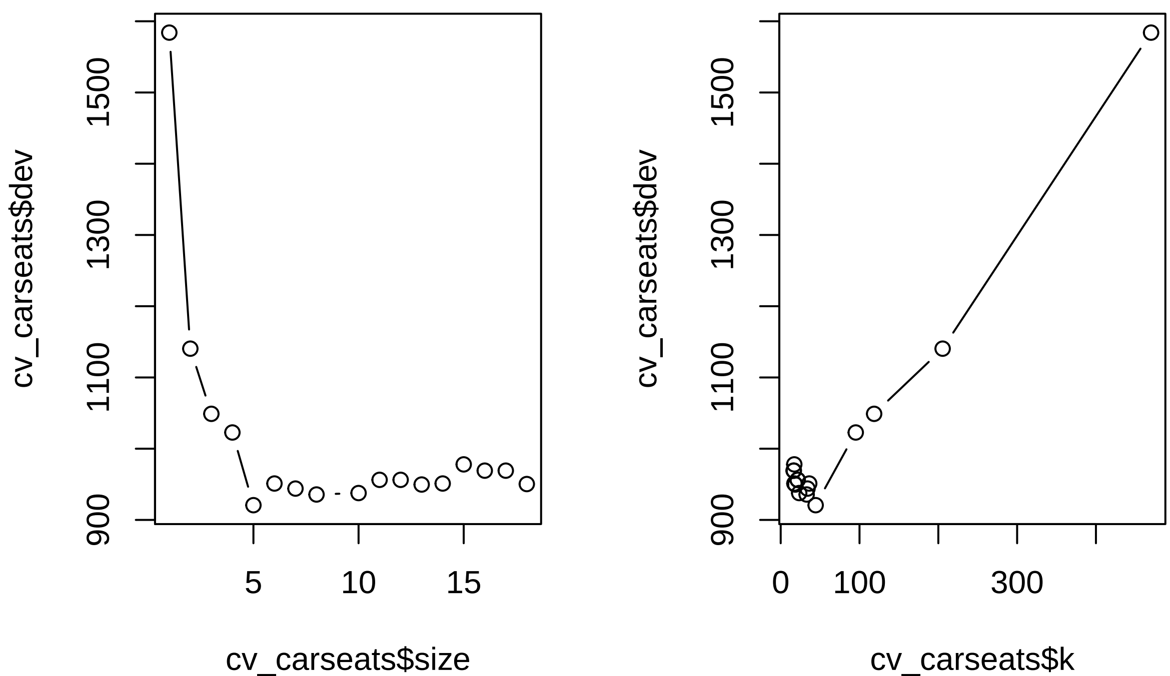
c)

```
set.seed(3)
cv_carseats <- cv.tree(tree_carseats, FUN=prune.tree)
cv_carseats
```

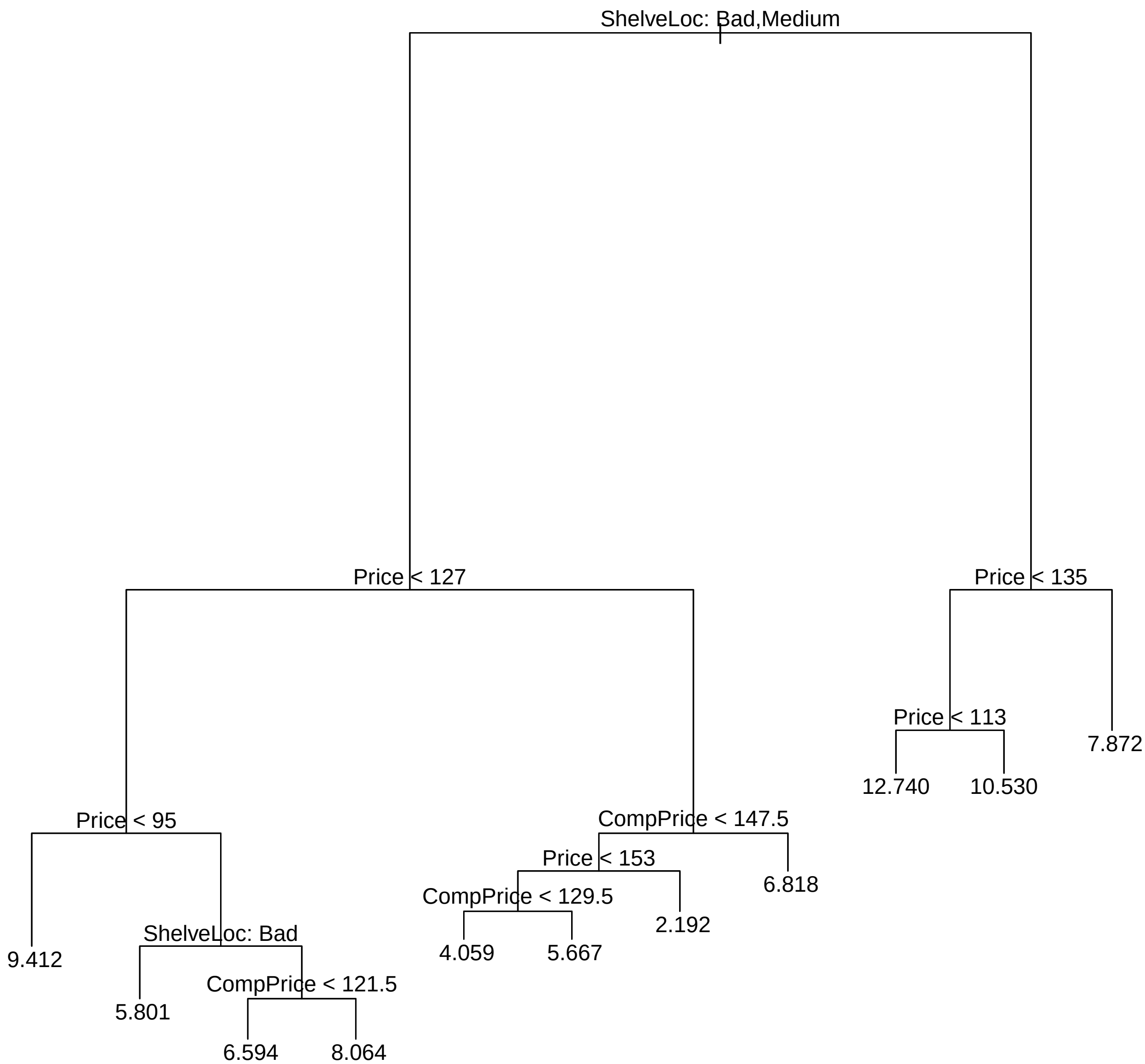
```
## $size
##  [1] 18 17 16 15 14 13 12 11 10  8  7  6  5  4  3  2  1
##
## $dev
##  [1]  950.3688  969.1779  969.1779  977.9726  951.1119  949.6944  956.2823
##  [8]  956.2823  937.7159  935.6981  944.0214  951.0920  920.7122 1022.7277
## [15] 1048.9273 1140.4169 1584.0404
##
## $k
##  [1]      -Inf  16.36627  16.68802  17.23878  17.38564  18.56548  21.46930
##  [8]  21.65597  23.47385  32.93709  34.04888  36.22009  44.36641  95.23265
## [15] 118.57789 205.48993 469.97839
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

dev corresponds to the cross-validation error rate. The tree with 11 terminal nodes results in the lowest cross-validation error rate.

```r
par(mfrow=c(1,2))
plot(cv_carseats$size, cv_carseats$dev, type='b')
plot(cv_carseats$k, cv_carseats$dev, type='b')
```




```r
pruned <- prune.tree(tree_carseats, best=11)
plot(pruned)
text(pruned, pretty=0)
```




```
pred <- predict(pruned, test_carseats)
sprintf('In this problem pruning the tree did not improve test MSE; test MSE increased to %0.2f',mean((t
```

```
## [1] "In this problem pruning the tree did not improve test MSE; test MSE increased to 5.12"
```

d)

```
set.seed(1)
# dim(Carseats)
bag <- randomForest(Sales~., data=train_carseats, mtry=10, ntree=25,
                    importance=TRUE)
pred <- predict(bag, test_carseats)
sprintf('using bagging, test MSE improved to %0.2f',
        mean((test_carseats$Sales-pred)^2))
```

```
## [1] "using bagging, test MSE improved to 3.34"
```

```r
importance(bag)
```

```
##              %IncMSE IncNodePurity
## CompPrice    4.5358041    118.623002
## Income       1.4086749     70.807952
## Advertising  2.6733470     75.108011
## Population   0.5905624     50.794658
## Price       16.4304202    547.940912
## ShelveLoc   17.8131872    535.528798
## Age          3.2996441     88.223396
## Education    2.5365484     30.351911
## Urban       -0.6291531      4.049871
## US           1.7180036     15.249450
```

ShelveLoc and price are the most important predictors of Sales

e)

```r
for (m in 1:10) {
  set.seed(1)
  rf_carseats <- randomForest(Sales~., data=train_carseats, mtry=m,
                              importance=TRUE)
  pred <- predict(rf_carseats, test_carseats)
  x <- sprintf('using random forests with m of %d, obtained test MSE of %0.2f',
         m, mean((test_carseats$Sales-pred)^2))
  print(x)
}
```

```
## [1] "using random forests with m of 1, obtained test MSE of 4.85"
## [1] "using random forests with m of 2, obtained test MSE of 3.69"
## [1] "using random forests with m of 3, obtained test MSE of 3.39"
## [1] "using random forests with m of 4, obtained test MSE of 3.24"
## [1] "using random forests with m of 5, obtained test MSE of 3.20"
## [1] "using random forests with m of 6, obtained test MSE of 3.23"
## [1] "using random forests with m of 7, obtained test MSE of 3.23"
## [1] "using random forests with m of 8, obtained test MSE of 3.24"
## [1] "using random forests with m of 9, obtained test MSE of 3.31"
## [1] "using random forests with m of 10, obtained test MSE of 3.33"
```

Changing the m value yields test MSE range of 3.2 to 4.85, with 5 predictors being considered at each split giving the optimal result.

```r
importance(rf_carseats)
```

```
##              %IncMSE IncNodePurity
## CompPrice   20.9151550    122.633506
## Income       4.5399731     63.370708
## Advertising 15.0311965     82.858698
## Population  -1.8046933     48.041107
## Price       62.9175958    544.475470
## ShelveLoc   72.6488829    528.320204
## Age         13.0282805     89.956262
## Education    4.3581110     36.608530
## Urban        0.8178007      4.283056
## US           6.8333772      9.637609
```

Again, Price and ShelveLoc are the two most important predictors of Sale.