Chapter 8 Problem 8

Andira Putri

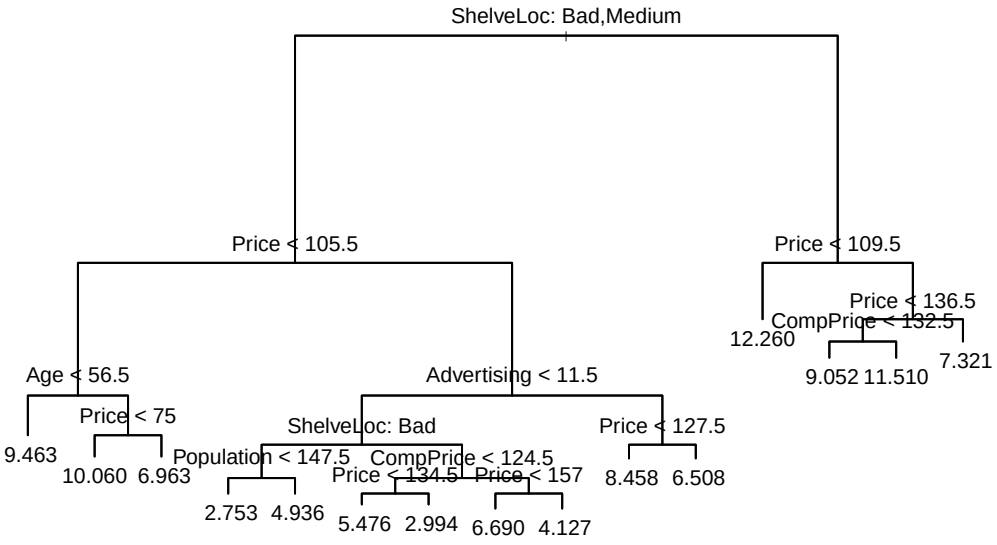
We seek to use the `Carseats` data set to predict `Sales` using regression trees and related approaches, treating the response as a qualitative variable.

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

```
library(ISLR)
data(Carseats) #contains 400 obsv.
set.seed(1)
train=sample(400,300) #300 obsv. for training
train.set=Carseats[train,]
test.set=Carseats[-train,] #100 obsv. for testing
```

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

```
library(tree)
car.tree=tree(Sales~.,train.set) #syntax close to lm()
plot(car.tree,cex=0.3)
text(car.tree,pretty=0,cex=0.65)
```



Based on branching patterns, it seems like the most important variable is ShelveLoc, or the quality of the shelving location. Then, the next important variable is carseat price.

```
#Computing MSE
predict=predict(car.tree,newdata=test.set) #yhat
#pretty much using just the MSE formula
MSE=mean((predict-test.set$Sales)^2)
MSE
```

```
## [1] 5.491716
```

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

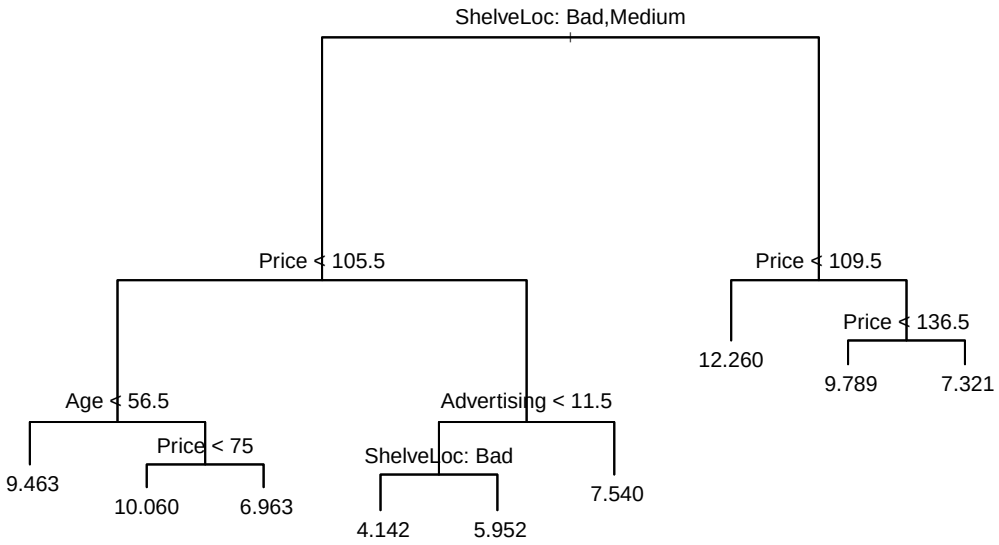
```
cv=cv.tree(car.tree)
names(cv)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```
cv #dev = CV error rate
```

```
## $size
##  [1] 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 1455.010 1430.865 1449.409 1467.981 1473.416 1473.416 1465.608
##  [8] 1448.548 1421.460 1502.799 1576.644 1657.045 1655.728 1877.846
## [15] 2416.253
##
## $k
##  [1]      -Inf  25.67629  32.19794  35.26337  36.20242  36.37037  38.01692
##  [8]  50.04224  52.22993  78.35626  93.12446 115.29783 132.30364 311.30392
## [15] 533.05880
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
#the lowest error rate results from 9 nodes
prune=prune.tree(car.tree,best=9)
plot(prune,cex=0.3)
text(prune,pretty=0,cex=0.65)
```




```
#Computing MSE
predict=predict(prune,newdata=test.set) #yhat
MSE=mean((predict-test.set$Sales)^2)
MSE
```

```
## [1] 5.33714
```

9 nodes yields the best MSE. By pruning the tree, the MSE is reduced, but only by a little bit. This suggests that pruning the tree doesn't provide a significant impact when using the model on test data.

d.) Use the bagging approach in order to analyze the data. What test MSE do you obtain? Use the `importance()` function to determine which variables are most important.

As a personal note, decision trees suffer from high variance–averaging a set of observations is a way to reduce that variance. Bagging works by bootstrapping the training set B times. Then, it trains the method on the b-th bootstrapped training set in order to get a value: $\hat{f}^b(x)$. Finally, it averages all predictions to get:

$\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$

Using a large value for B is not dangerous for overfitting...just make sure it is sufficiently large :)

```
set.seed(1)
library(randomForest)
```

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

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

```
bag.car=randomForest(Sales~.,data=train.set,mtry=10,importance=TRUE)
#mtry=11 b/c there are 10 predictors to consider
```

```
bag.car
```

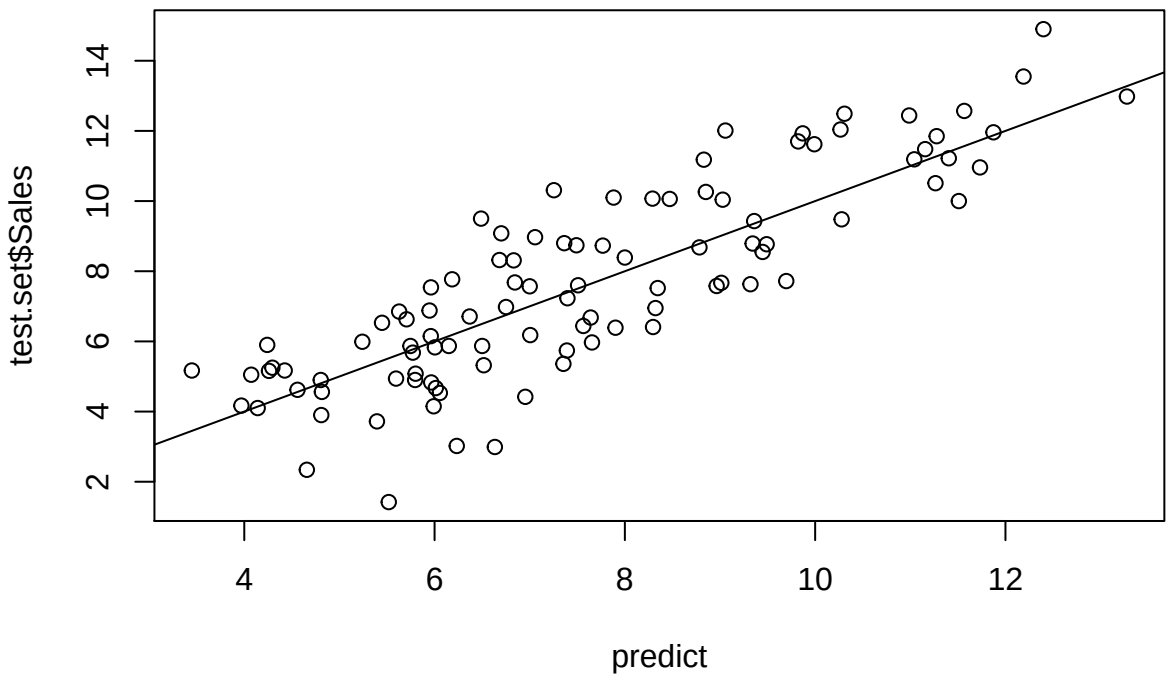
```
##
## Call:
##  randomForest(formula = Sales ~ ., data = train.set, mtry = 10,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 2.539553
##                     % Var explained: 68.01
```

```
#Computing MSE
predict=predict(bag.car,newdata=test.set) #yhat
MSE=mean((predict-test.set$Sales)^2)
MSE
```

```
## [1] 2.209486
```

Our test MSE is 2.209486, which is greatly reduced from our previous two trees!

```
plot(predict,test.set$Sales)
abline(0,1)
```




```
importance(bag.car)
```

```
##               %IncMSE IncNodePurity
## CompPrice   30.260612     228.92041
## Income       8.102690     116.70390
```

```
## Advertising 21.884122     190.20770
## Population   3.621385      95.10519
## Price       72.265798     731.37342
## ShelveLoc   76.197166     655.36199
## Age         22.506044     218.59352
## Education    2.551942      63.46486
## Urban       -3.905544      10.01294
## US           4.222018      13.39688
```

The `importance()` function suggests that our most important variables are ShelveLoc and Price. This is because %IncMSE, aka mean decrease in accuracy, values are the highest for these particular variables. When ShelveLoc and Price are randomly permuted, they change the prediction much more than other predictors in the model.

e.) Use random forests to analyze the 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.

Random forests decorrelate the trees. Each time a split is considered, `m` predictors are randomly chosen as split candidates from a full set of `p` predictors. In general, m=$\sqrt{p}$. If m=p, this is just bagging.

```
set.seed(1)
rf.car=randomForest(Sales~.,data=train.set,mtry=5,importance=TRUE)
rf.car
```

```
##
## Call:
##  randomForest(formula = Sales ~ ., data = train.set, mtry = 5,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 2.598248
##                     % Var explained: 67.27
```

```
#Computing MSE
predict=predict(rf.car,newdata=test.set) #yhat
MSE=mean((predict-test.set$Sales)^2)
MSE
```

```
## [1] 2.346116
```

```
importance(rf.car)
```

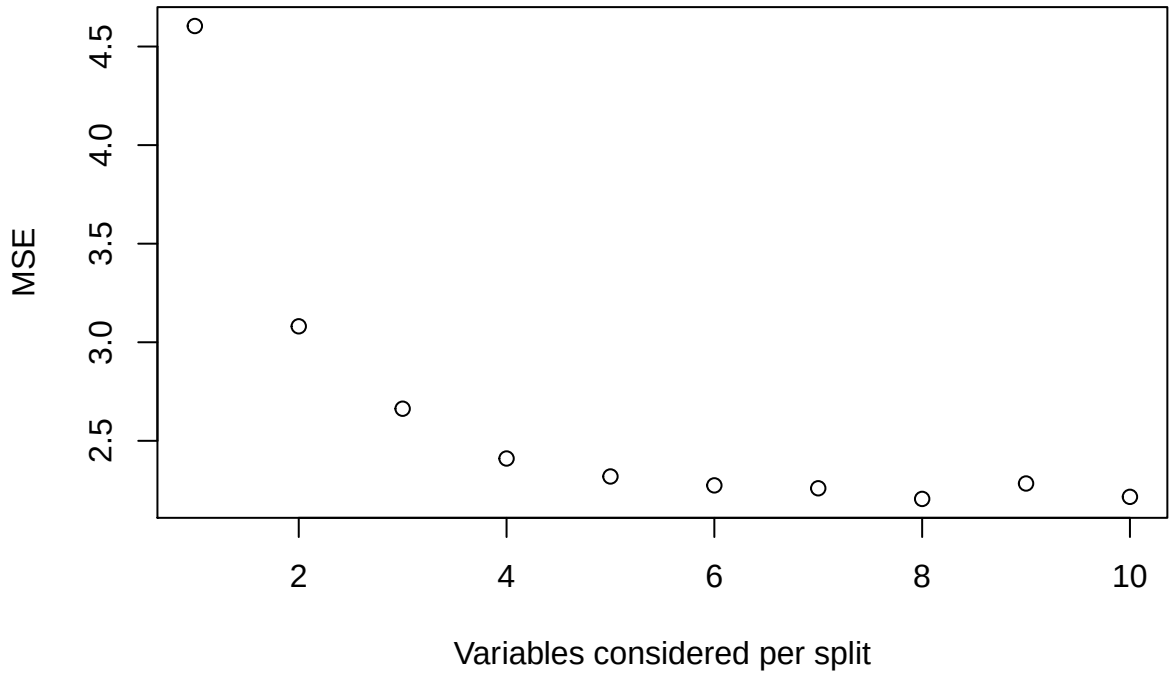
```
##               %IncMSE IncNodePurity
## CompPrice   21.589957     211.02988
## Income       6.466369     144.13991
## Advertising 19.920040     211.86513
## Population   1.890401     124.95181
## Price       58.121555     639.66910
## ShelveLoc   60.294851     604.18384
## Age         20.690142     227.53935
## Education    3.107312      81.61082
## Urban       -3.075403      14.14043
## US           1.534287      25.94580
```

The test MSE is 2.346116, which is higher than the bagging approach but only slightly. ShelveLoc and Price are still the most important predictors.

How does test MSE change as m changes?

```
for (i in 1:10){
  rf.car.loop=randomForest(Sales~.,data=train.set,mtry=i,importance=TRUE)
  predict=predict(rf.car.loop,newdata=test.set)
  MSE[i]=mean((predict-test.set$Sales)^2)
}

plot(1:10,MSE,xlab="Variables considered per split")
```



From the plot, MSE greatly reduces as m increases from 1, starts leveling off at 6, and reaches a minimum at 8. We could make do with any `mtry` value 6 or greater.