Homework 7

Anthony Anderson

May 8, 2019

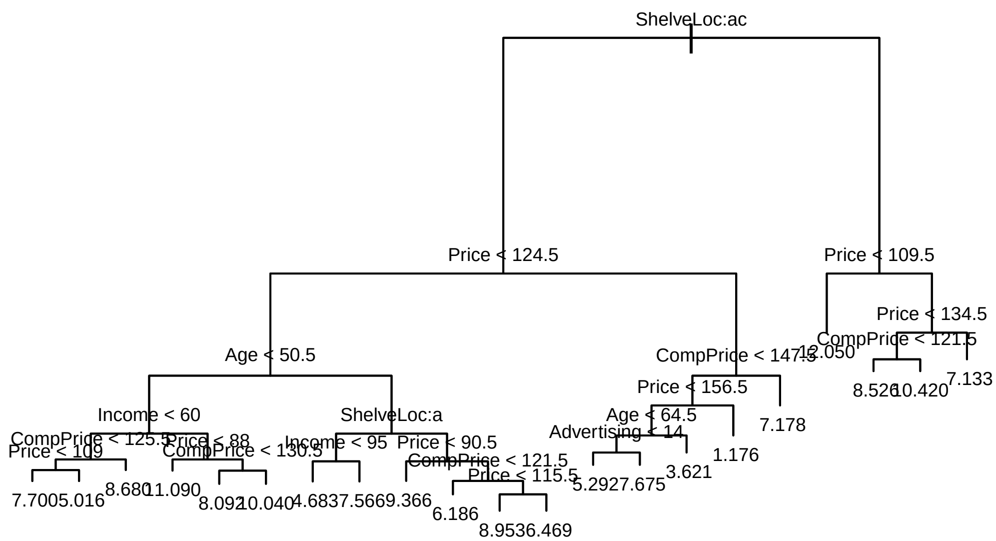
Chapter 8, exercise 8:

```r
#A)
library(ISLR)
carseats<-Carseats
set.seed(42)
train=sample(nrow(carseats),.8*nrow(carseats))
train.data <- carseats[train,]
test.data <- carseats[-train,]

#B)
library(tree)
tree.fit<- tree(Sales~.,data=carseats,subset=train)
summary(tree.fit)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = carseats, subset = train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"
## [6] "Advertising"
## Number of terminal nodes:  21
## Residual mean deviance:  2.173 = 649.8 / 299
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.77300 -1.00200  0.04967  0.00000  1.03600  4.22200
```

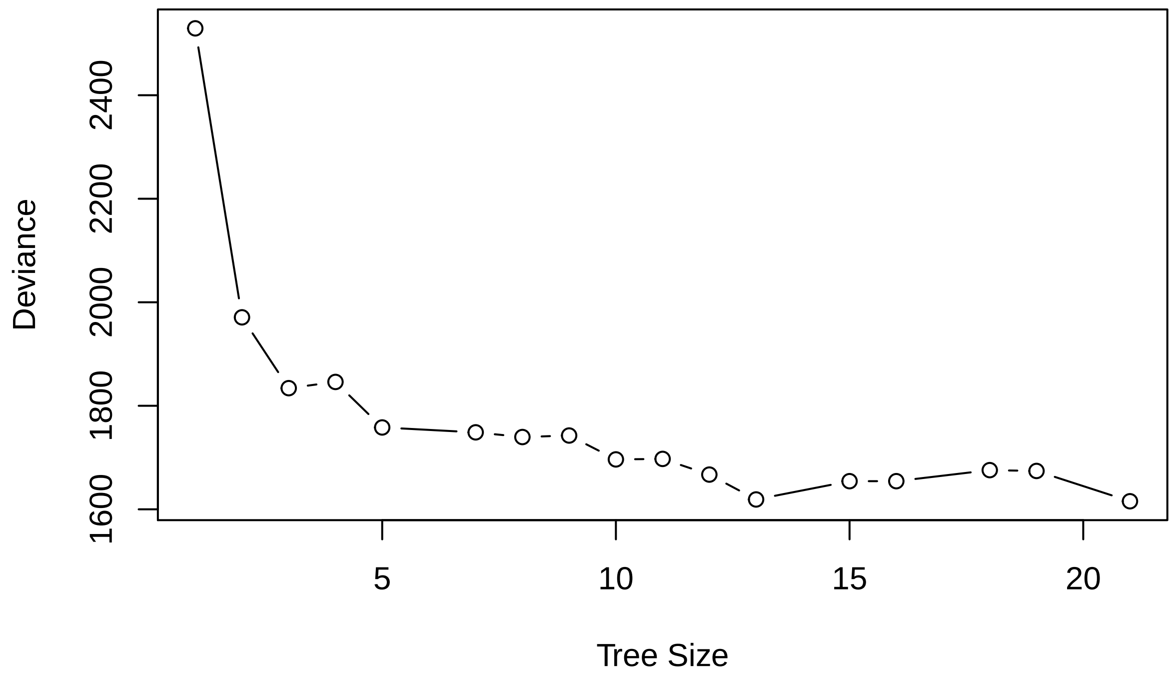
```r
plot(tree.fit)
text(tree.fit, cex=.6)
```




```
tree.fit.pred<-predict(tree.fit,newdata=test.data)
mean((tree.fit.pred-test.data$Sales)^2) #Test MSE is about 3.9
```

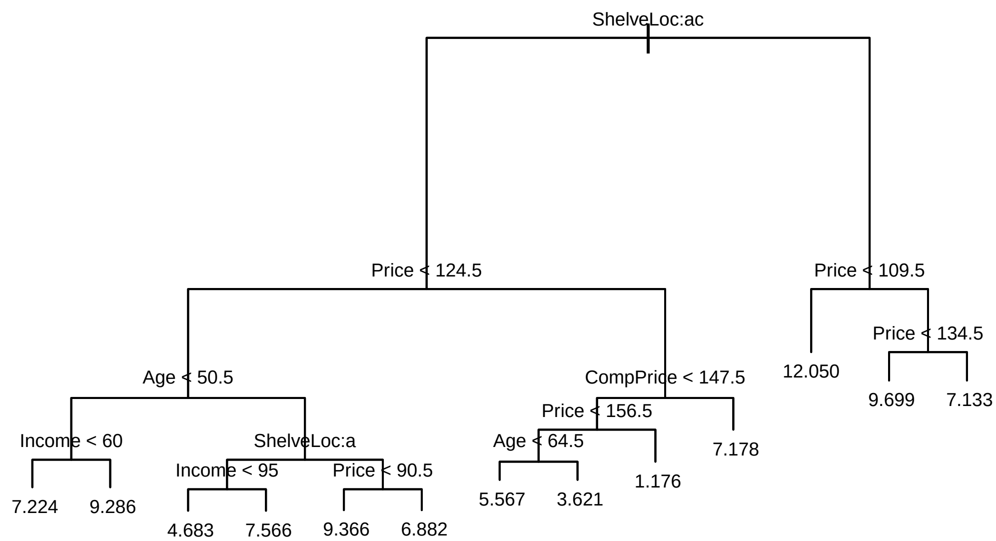
```
## [1] 3.906465
```

```
#C)
set.seed(42)
cv.fit<-cv.tree(tree.fit)
plot(cv.fit$size, cv.fit$dev, type="b", xlab="Tree Size", ylab="Deviance")
```




```
#Size 13 has lowest deviance

prune.fit<-prune.tree(tree.fit, best=13)
plot(prune.fit)
text(prune.fit, cex=.6)
```




```
prune.fit.pred<-predict(prune.fit,newdata=test.data)
mean((prune.fit.pred-test.data$Sales)^2) #Test MSE is 5.27, which is higher so pruning didn't help
```

```
## [1] 4.355675
```

```
#D)
library(randomForest)
```

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

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

```
set.seed(42)
bag.fit<- randomForest(Sales~.,data=train.data,mtry=10,importance=T)
#mtry= number of predictors for bagging
bag.fit.pred<-predict(bag.fit,newdata=test.data)
mean((bag.fit.pred-test.data$Sales)^2) #Test MSE here is only 2.35
```

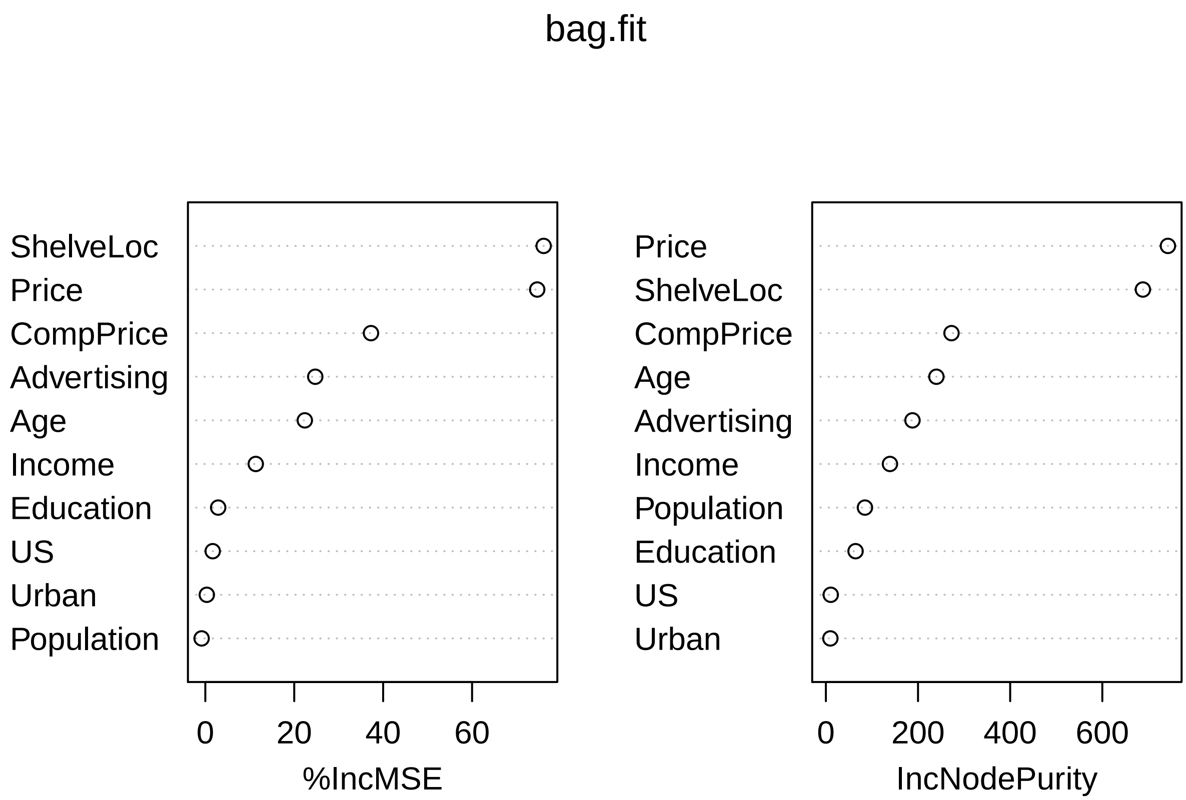
```
## [1] 2.247194
```

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

```
##                %IncMSE IncNodePurity
## CompPrice   37.2655918    272.684989
## Income      11.3481020    139.099550
## Advertising 24.7312015    187.974015
## Population  -0.8309134     84.799869
## Price       74.6565917    742.426277
## ShelveLoc   76.1027278    688.088937
```

```
## Age          22.3794426    239.876755
## Education     2.8933345     64.433523
## Urban         0.3376574      9.737118
## US            1.6649153     10.587431
```

```r
varImpPlot(bag.fit) #The most importance variables are ShelveLoc and Price.
```




```r
#Highest Inc. in node purity and %Inc in MSE from importance output, and plotted with varImpPlot()

#E)
forest.fit<-randomForest(Sales~.,data=train.data,mtry=3,importance=T)
forest.fit.pred<-predict(forest.fit,newdata=test.data)
mean((forest.fit.pred-test.data$Sales)^2) #The test MSE for r.f. is 2.93, higher than bagging
```

```
## [1] 2.835609
```

```r
importance(forest.fit) #Most important variables are the same as above, ShelveLoc and Price.
```

```
##                %IncMSE IncNodePurity
## CompPrice   16.5827316     225.67910
## Income       6.3862954     193.16698
## Advertising 17.5873572     218.29447
## Population   1.4117218     155.91267
## Price       44.9860497     581.88181
## ShelveLoc   49.9640240     538.27366
## Age         15.2400202     271.29274
## Education    0.8508437     111.00186
## Urban       -2.1433634      22.38776
```

```
## US           3.8134550      34.28175
```

```
#By using a smaller mtry, we avoid collinearity (decorrelate) in the trees, a problem in bagging where
#mtry is equal to the number of all predictors, so they will all be used
```

Chapter 9, exercise 8:

```
#A)
library(ISLR)
OJdata<- OJ
set.seed(42)
train1<-sample(nrow(OJdata), 800)
OJ.train<-OJdata[train1,]
OJ.test<-OJdata[-train1,]

#B)
library(e1071)
svm.fit<- svm(Purchase~.,data=OJ.train, kernel="linear",cost=.01)
summary(svm.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  432
##
##  ( 215 217 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
#There are 432/800 data points considered support vectors for predicting the 2 classes

#C)
svm.predict<- predict(svm.fit,newdata = OJ.test)
table(predict=svm.predict, truth=OJ.test$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 142  25
##      MM  19  84
```

```
#This model misclassifies a total of 44/270 observations in test set
svm.predict.train<- predict(svm.fit,newdata = OJ.train)
table(predict=svm.predict.train, truth=OJ.train$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 432  77
##      MM  60 231
```

```r
#This model misclassified 137/800 in training set

#D)
set.seed(42)
svm.tuned<-tune(svm,Purchase~.,data=OJ.train,kernel="linear", ranges=list(cost=c(.01,.05,.1,.5,1,5,10)))
summary(svm.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.175
##
## - Detailed performance results:
##    cost   error dispersion
## 1  0.01 0.17750 0.02415229
## 2  0.05 0.17750 0.03322900
## 3  0.10 0.17625 0.03356689
## 4  0.50 0.17750 0.02751262
## 5  1.00 0.17500 0.02886751
## 6  5.00 0.18375 0.02703521
## 7 10.00 0.18625 0.02729087
```

```r
#Optimal cost value is 1 since it has lowest cv.error rate

#E)
best.svm<-svm.tuned$best.model
best.svm.pred<- predict(best.svm,newdata = OJ.test)
table(predict=best.svm.pred, truth=OJ.test$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 140  23
##      MM  21  86
```

```r
#Best model misclassified 44/270 observations in test set
best.svm.pred.train<- predict(best.svm,newdata = OJ.train)
table(predict=best.svm.pred.train, truth=OJ.train$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 434  76
##      MM  58 232
```

```r
#Best model misclassified 134/800 observations in training set.

#F)
```

```r
svm.rad.fit<- svm(Purchase~.,data=OJ.train, kernel="radial",cost=.01)
summary(svm.rad.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  621
##
##  ( 308 313 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
svm.rad.predict<- predict(svm.rad.fit,newdata = OJ.test)
table(predict=svm.rad.predict, truth=OJ.test$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 161 109
##      MM   0   0
```

```r
#This model misclassifies a total of 109/270 observations in test set,
#all of which are truly MM but predicted as CH
svm.rad.predict.train<- predict(svm.rad.fit,newdata = OJ.train)
table(predict=svm.rad.predict.train, truth=OJ.train$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 492 308
##      MM   0   0
```

```r
#This model misclassified 308/800 in training set, all of which are truly MM but predicted as CH

set.seed(42)
svm.rad.tuned<-tune(svm,Purchase~.,data=OJ.train,kernel="radial",
                    ranges=list(cost=c(.01,.05,.1,.5,1,5,10)))
summary(svm.rad.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
```

```
##  cost
##     1
##
## - best performance: 0.18
##
## - Detailed performance results:
##     cost   error dispersion
## 1  0.01 0.38500 0.04199868
## 2  0.05 0.20500 0.05407043
## 3  0.10 0.18125 0.03784563
## 4  0.50 0.18375 0.02503470
## 5  1.00 0.18000 0.03343734
## 6  5.00 0.18625 0.03701070
## 7 10.00 0.19375 0.03738408
```

```r
#Optimal cost value is 1 since it has lowest cv.error rate

best.rad.svm<-svm.rad.tuned$best.model
best.rad.svm.pred<- predict(best.rad.svm,newdata = OJ.test)
table(predict=best.rad.svm.pred, truth=OJ.test$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 146  28
##      MM  15  81
```

```r
#Best model misclassified 43/270 observations in test set
best.rad.svm.pred.train<- predict(best.rad.svm,newdata = OJ.train)
table(predict=best.rad.svm.pred.train, truth=OJ.train$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 453  81
##      MM  39 227
```

```r
#Best model misclassified 120/800 observations in training set.

#G)
svm.poly.fit<- svm(Purchase~.,data=OJ.train, kernel="polynomial",degree=2,cost=.01)
summary(svm.poly.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "polynomial",
##     degree = 2, cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##       gamma:  0.05555556
##      coef.0:  0
##
## Number of Support Vectors:  621
```

```
##
##  ( 308 313 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
svm.poly.predict<- predict(svm.poly.fit,newdata = OJ.test)
table(predict=svm.poly.predict, truth=OJ.test$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 161 109
##      MM   0   0
```

```r
#This model misclassifies a total of 109/270 observations in test set,
#all of which are truly MM but predicted as CH, same as radial
svm.poly.predict.train<- predict(svm.poly.fit,newdata = OJ.train)
table(predict=svm.poly.predict.train, truth=OJ.train$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 492 308
##      MM   0   0
```

```r
#This model misclassified 308/800 in training set, all of which are truly MM but predicted as CH

set.seed(42)
svm.poly.tuned<-tune(svm,Purchase~.,data=OJ.train,kernel="polynomial", degree=2,
                    ranges=list(cost=c(.01,.05,.1,.5,1,5,10)))
summary(svm.poly.tuned) #Optimal cost value is 5 since it has lowest cv.error rate
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.18375
##
## - Detailed performance results:
##    cost   error dispersion
## 1  0.01 0.38625 0.04308019
## 2  0.05 0.32750 0.04594683
## 3  0.10 0.31625 0.05529278
## 4  0.50 0.20125 0.03884174
## 5  1.00 0.19250 0.04216370
## 6  5.00 0.18375 0.04041881
## 7 10.00 0.19000 0.03425801
```

```r
best.poly.svm<-svm.poly.tuned$best.model
best.poly.svm.pred<- predict(best.poly.svm,newdata = OJ.test)
```

```r
table(predict=best.poly.svm.pred, truth=OJ.test$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 146  30
##      MM  15  79
```

```r
#Best model misclassified 45/270 observations in test set
best.poly.svm.pred.train<- predict(best.poly.svm,newdata = OJ.train)
table(predict=best.poly.svm.pred.train, truth=OJ.train$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 459  85
##      MM  33 223
```

```r
#Best model misclassified 118/800 observations in training set.
```

H) Overall, the lowest misclassification rates come from the best radial model, with cost .5 Some error rates between models are very similar, but in some cases there is a total misclassification of a single group.