JieTang_Lab7.R

tjj

2020-08-18

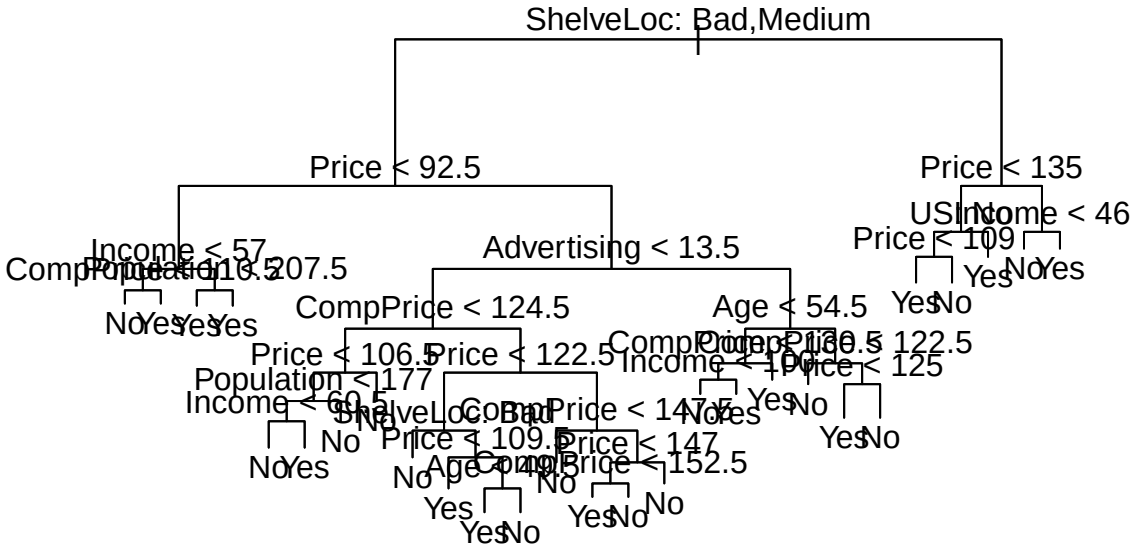
```
# Chapter 8 Lab: Decision Trees

#Name:Jie Tang
#Course:Machine learning using R 374815
#Quarter:Summer
#Instructor name : Michael Chang

#Quiz part
#Basic setting
library(tree)
library(ISLR)
attach(Carseats)
High=factor(ifelse(Sales<=8,"No","Yes"))
Carseats=data.frame(Carseats,High)
tree.carseats=tree(High~.-Sales,Carseats)
summary(tree.carseats)
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

```
plot(tree.carseats)
text(tree.carseats,pretty=0)
```




```
tree.carseats
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 400 541.500 No ( 0.59000 0.41000 )
##     2) ShelveLoc: Bad,Medium 315 390.600 No ( 0.68889 0.31111 )
##       4) Price < 92.5 46  56.530 Yes ( 0.30435 0.69565 )
##         8) Income < 57 10  12.220 No ( 0.70000 0.30000 )
##          16) CompPrice < 110.5 5   0.000 No ( 1.00000 0.00000 ) *
##          17) CompPrice > 110.5 5   6.730 Yes ( 0.40000 0.60000 ) *
##         9) Income > 57 36  35.470 Yes ( 0.19444 0.80556 )
##          18) Population < 207.5 16  21.170 Yes ( 0.37500 0.62500 ) *
##          19) Population > 207.5 20   7.941 Yes ( 0.05000 0.95000 ) *
##       5) Price > 92.5 269 299.800 No ( 0.75465 0.24535 )
##        10) Advertising < 13.5 224 213.200 No ( 0.81696 0.18304 )
##          20) CompPrice < 124.5 96  44.890 No ( 0.93750 0.06250 )
##            40) Price < 106.5 38  33.150 No ( 0.84211 0.15789 )
##              80) Population < 177 12  16.300 No ( 0.58333 0.41667 )
##               160) Income < 60.5 6   0.000 No ( 1.00000 0.00000 ) *
##               161) Income > 60.5 6   5.407 Yes ( 0.16667 0.83333 ) *
##              81) Population > 177 26   8.477 No ( 0.96154 0.03846 ) *
##            41) Price > 106.5 58   0.000 No ( 1.00000 0.00000 ) *
##          21) CompPrice > 124.5 128 150.200 No ( 0.72656 0.27344 )
##            42) Price < 122.5 51  70.680 Yes ( 0.49020 0.50980 )
##              84) ShelveLoc: Bad 11   6.702 No ( 0.90909 0.09091 ) *
```

```
##               85) ShelveLoc: Medium 40  52.930 Yes ( 0.37500 0.62500 )
##                170) Price < 109.5 16   7.481 Yes ( 0.06250 0.93750 ) *
##                171) Price > 109.5 24  32.600 No ( 0.58333 0.41667 )
##                  342) Age < 49.5 13  16.050 Yes ( 0.30769 0.69231 ) *
##                  343) Age > 49.5 11   6.702 No ( 0.90909 0.09091 ) *
##             43) Price > 122.5 77  55.540 No ( 0.88312 0.11688 )
##               86) CompPrice < 147.5 58  17.400 No ( 0.96552 0.03448 ) *
##               87) CompPrice > 147.5 19  25.010 No ( 0.63158 0.36842 )
##                174) Price < 147 12  16.300 Yes ( 0.41667 0.58333 )
##                  348) CompPrice < 152.5 7   5.742 Yes ( 0.14286 0.85714 ) *
##                  349) CompPrice > 152.5 5   5.004 No ( 0.80000 0.20000 ) *
##                175) Price > 147 7   0.000 No ( 1.00000 0.00000 ) *
##         11) Advertising > 13.5 45  61.830 Yes ( 0.44444 0.55556 )
##           22) Age < 54.5 25  25.020 Yes ( 0.20000 0.80000 )
##             44) CompPrice < 130.5 14  18.250 Yes ( 0.35714 0.64286 )
##               88) Income < 100 9  12.370 No ( 0.55556 0.44444 ) *
##               89) Income > 100 5   0.000 Yes ( 0.00000 1.00000 ) *
##             45) CompPrice > 130.5 11   0.000 Yes ( 0.00000 1.00000 ) *
##           23) Age > 54.5 20  22.490 No ( 0.75000 0.25000 )
##             46) CompPrice < 122.5 10   0.000 No ( 1.00000 0.00000 ) *
##             47) CompPrice > 122.5 10  13.860 No ( 0.50000 0.50000 )
##               94) Price < 125 5   0.000 Yes ( 0.00000 1.00000 ) *
##               95) Price > 125 5   0.000 No ( 1.00000 0.00000 ) *
##      3) ShelveLoc: Good 85  90.330 Yes ( 0.22353 0.77647 )
##        6) Price < 135 68  49.260 Yes ( 0.11765 0.88235 )
##         12) US: No 17  22.070 Yes ( 0.35294 0.64706 )
##           24) Price < 109 8   0.000 Yes ( 0.00000 1.00000 ) *
##           25) Price > 109 9  11.460 No ( 0.66667 0.33333 ) *
##         13) US: Yes 51  16.880 Yes ( 0.03922 0.96078 ) *
##        7) Price > 135 17  22.070 No ( 0.64706 0.35294 )
##         14) Income < 46 6   0.000 No ( 1.00000 0.00000 ) *
##         15) Income > 46 11  15.160 Yes ( 0.45455 0.54545 ) *
```

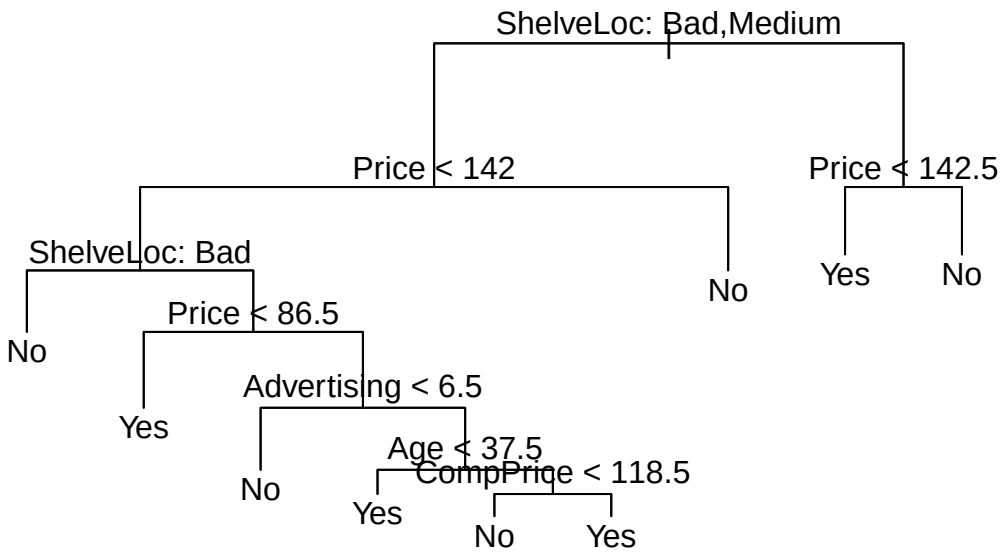
```
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(2)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
train=sample(1:nrow(Carseats), 200)
Carseats.test=Carseats[-train,]
High.test=High[-train]
tree.carseats=tree(High~.-Sales,Carseats,subset=train)
#Q1
#Take tree.carseats from the lab and run prune.misclass while setting best to 9.
prune.carseats=prune.misclass(tree.carseats,best=9)
plot(prune.carseats)
text(prune.carseats,pretty=0)
```



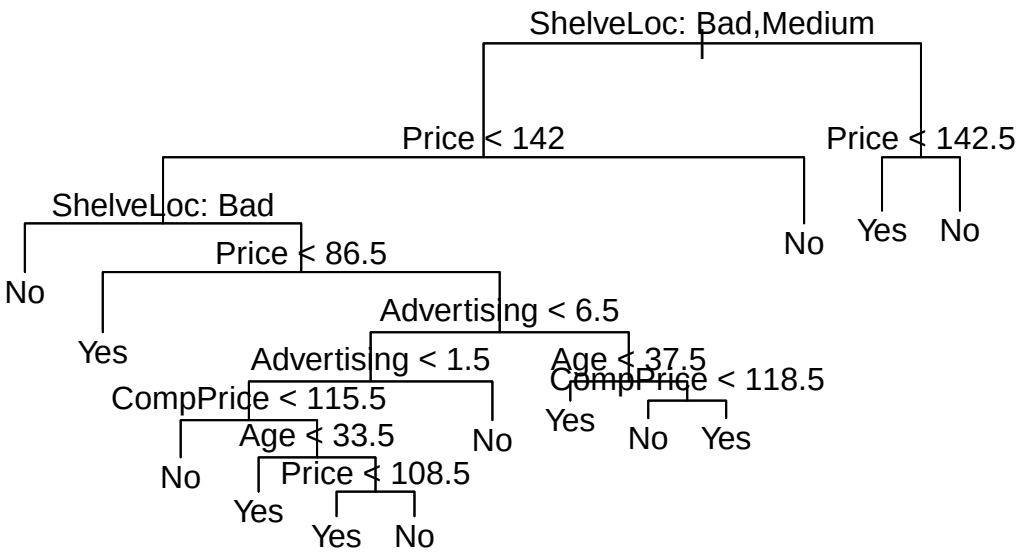

```
#Q2
#produce predictions and calculate accuracy
tree.pred=predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred No Yes
##       No  94  24
##       Yes 22  60
```

```
(94+60)/200
```

```
## [1] 0.77
```

```
#Q3
#set best to 13 and do the comparsion
prune.carseats=prune.misclass(tree.carseats,best=13)
plot(prune.carseats)
text(prune.carseats,pretty=0)
```




```
tree.pred=predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred No Yes
##       No  91  21
##       Yes 25  63
```

```
(91+63)/200
```

```
## [1] 0.77
```

```
#Q4
#check the plot
library(MASS)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

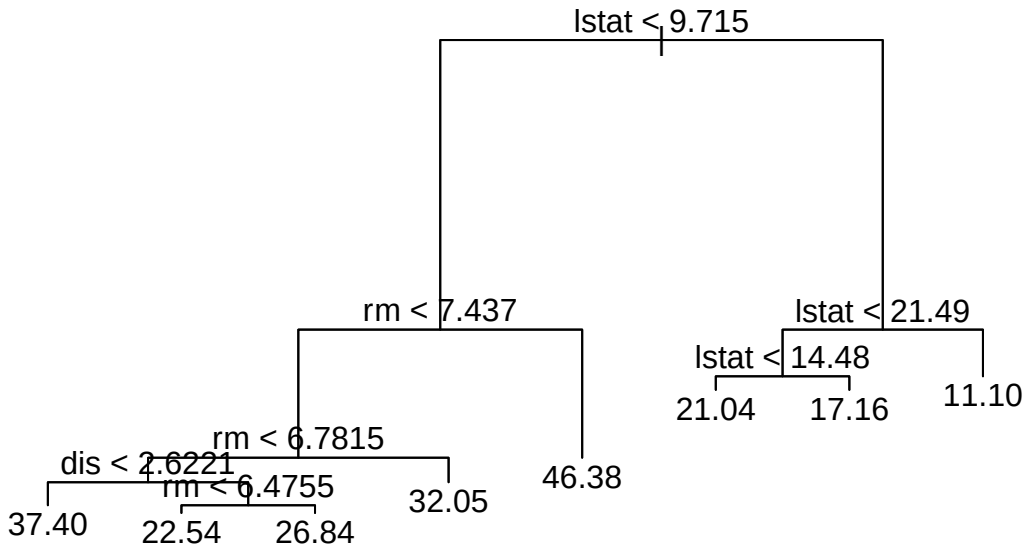
```
set.seed(1)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
train = sample(1:nrow(Boston), nrow(Boston)/2)
tree.boston=tree(medv~.,Boston,subset=train)
summary(tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "lstat" "rm"    "dis"
## Number of terminal nodes:  8
## Residual mean deviance:  12.65 = 3099 / 245
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -14.10000  -2.04200  -0.05357   0.00000   1.96000  12.60000
```

```
plot(tree.boston)
text(tree.boston,pretty=0)
```




```
#Q5
#used above trainning data, use random forest to do the predictions
library(randomForest)
```

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

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

```
Boston[-train,"medv"]
```

```
##   [1] 24.0 21.6 34.7 33.4 36.2 27.1 16.5 18.9 18.9 18.2 19.9 23.1 17.5 20.2 18.2
##  [16] 13.6 15.2 14.5 13.9 14.8 13.2 13.1 18.9 30.8 26.6 25.3 24.7 19.3 20.0 14.4
##  [31] 19.4 25.0 23.4 24.7 31.6 16.0 33.0 23.5 22.0 17.4 20.9 24.2 21.7 24.1 20.8
##  [46] 20.3 28.0 24.8 22.9 23.9 22.6 22.9 20.6 27.5 19.3 20.1 19.5 20.4 18.8 18.5
##  [61] 19.2 20.3 17.3 18.8 21.4 16.2 14.3 19.6 23.0 18.4 18.1 17.4 13.3 17.8 14.4
##  [76] 13.8 15.6 17.8 19.6 15.3 17.0 15.6 24.3 27.0 50.0 22.7 25.0 50.0 17.4 19.1
##  [91] 22.6 29.4 24.6 29.9 37.2 39.8 37.9 26.4 30.5 31.1 33.3 34.9 32.9 24.1 42.3
## [106] 50.0 24.4 24.4 20.0 23.7 25.0 26.7 27.5 30.1 50.0 37.6 31.6 46.7 24.3 29.0
## [121] 24.0 23.7 20.1 18.5 24.5 24.4 24.8 29.6 42.8 20.9 50.0 36.0 43.1 48.8 31.0
## [136] 30.7 20.7 25.2 24.4 35.2 33.2 29.1 45.4 50.0 20.1 37.3 27.9 23.9 21.7 20.3
## [151] 29.0 33.1 36.1 16.1 22.1 19.4 19.8 23.1 23.1 20.4 18.5 25.0 23.0 19.3 17.1
## [166] 22.2 20.7 18.5 18.7 32.7 16.5 31.2 17.2 23.1 24.5 26.6 22.9 18.6 18.2 20.6
## [181] 22.7 25.0 21.9 50.0 50.0 50.0 13.8 15.0 10.9 10.2 11.5  9.7 12.7 13.1  8.5
## [196]  6.3  7.2 12.1  8.3  5.0 11.9 17.2 17.2 17.9  7.0  7.5  8.8  8.4 20.8 10.2
## [211] 10.9  9.5 14.5 14.1 14.3 11.7 13.4  8.4 12.8 15.4 11.8 14.9 14.1 13.0 16.1
## [226] 14.1 13.5 20.0 16.4 19.5 19.0 20.1 19.9 19.6 29.8 13.3 12.0 21.4 23.7 19.1
## [241] 15.2  7.0 20.1 21.8 24.5 19.7 18.3 21.2 16.8 22.4 20.6 22.0 11.9
```

```
boston.test=Boston[-train,"medv"]
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(1)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
bag.boston=randomForest(medv~.,data=Boston,subset=train,importance=TRUE)
bag.boston
```

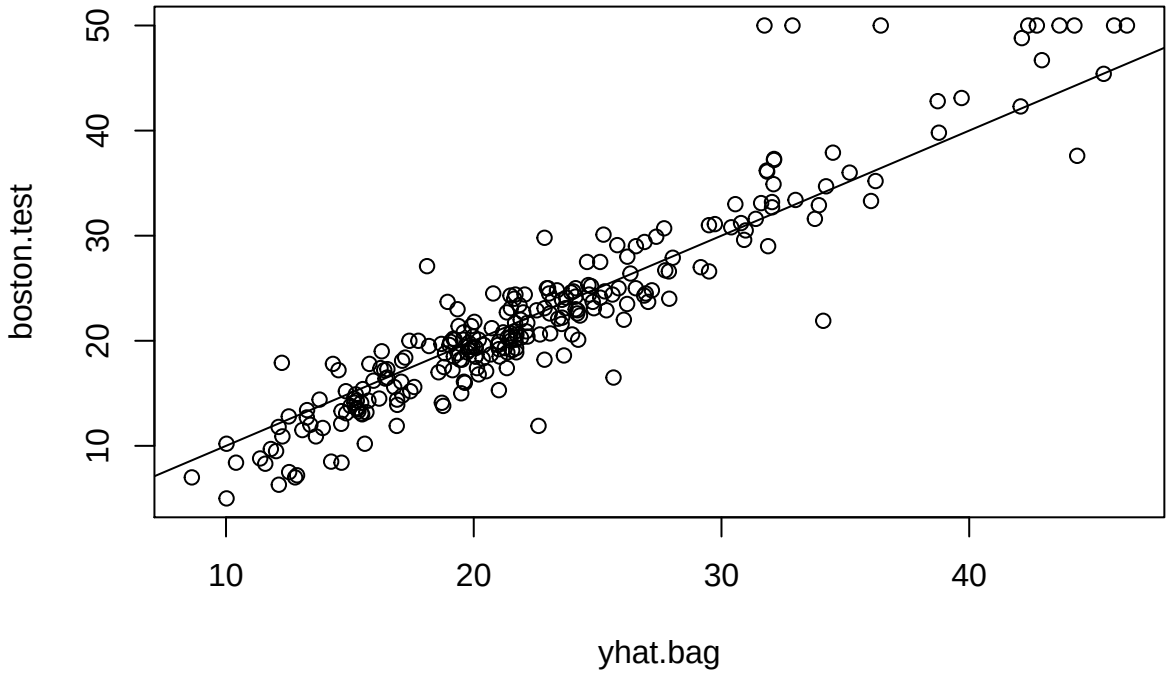
```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 12.48435
##                     % Var explained: 84.88
```

```
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
plot(yhat.bag, boston.test)
abline(0,1)
mean((yhat.bag-boston.test)^2)
```

```
## [1] 11.6076
```

```
#Q6
#set the given parameters and calculate MSE
library(gbm)
```

```
## Loaded gbm 2.1.8
```




```
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(1)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=5,sh
yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
```

```
## [1] 10.36727
```

```
#Q7
#Shrinkage is learning rate of model, and it make the MSE lower
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=5,sh
yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
```

```
## [1] 9.891304
```

```
#
# # Fitting Classification Trees
# library(tree)
# library(ISLR)
# attach(Carseats)
# High=factor(ifelse(Sales<=8,"No","Yes"))
# Carseats=data.frame(Carseats,High)
# tree.carseats=tree(High~.-Sales,Carseats)
# summary(tree.carseats)
# plot(tree.carseats)
# text(tree.carseats,pretty=0)
# tree.carseats
# set.seed(2)
# train=sample(1:nrow(Carseats), 200)
# Carseats.test=Carseats[-train,]
# High.test=High[-train]
# tree.carseats=tree(High~.-Sales,Carseats,subset=train)
# tree.pred=predict(tree.carseats,Carseats.test,type="class")
# table(tree.pred,High.test)
# (86+57)/200
# set.seed(3)
# cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)
# names(cv.carseats)
# cv.carseats
# par(mfrow=c(1,2))
# plot(cv.carseats$size,cv.carseats$dev,type="b")
# plot(cv.carseats$k,cv.carseats$dev,type="b")
# prune.carseats=prune.misclass(tree.carseats,best=9)
# plot(prune.carseats)
# text(prune.carseats,pretty=0)
# tree.pred=predict(prune.carseats,Carseats.test,type="class")
# table(tree.pred,High.test)
# (94+60)/200
# prune.carseats=prune.misclass(tree.carseats,best=15)
# plot(prune.carseats)
# text(prune.carseats,pretty=0)
# tree.pred=predict(prune.carseats,Carseats.test,type="class")
# table(tree.pred,High.test)
```

```
# (86+62)/200
#
# # Fitting Regression Trees
#
# library(MASS)
# set.seed(1)
# train = sample(1:nrow(Boston), nrow(Boston)/2)
# tree.boston=tree(medv~.,Boston,subset=train)
# summary(tree.boston)
# plot(tree.boston)
# text(tree.boston,pretty=0)
# cv.boston=cv.tree(tree.boston)
# plot(cv.boston$size,cv.boston$dev,type='b')
# prune.boston=prune.tree(tree.boston,best=5)
# plot(prune.boston)
# text(prune.boston,pretty=0)
# yhat=predict(tree.boston,newdata=Boston[-train,])
# boston.test=Boston[-train,"medv"]
# plot(yhat,boston.test)
# abline(0,1)
# mean((yhat-boston.test)^2)
#
# # Bagging and Random Forests
#
# library(randomForest)
# set.seed(1)
# bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,importance=TRUE)
# bag.boston
# yhat.bag = predict(bag.boston,newdata=Boston[-train,])
# plot(yhat.bag, boston.test)
# abline(0,1)
# mean((yhat.bag-boston.test)^2)
# bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,ntree=25)
# yhat.bag = predict(bag.boston,newdata=Boston[-train,])
# mean((yhat.bag-boston.test)^2)
# set.seed(1)
# rf.boston=randomForest(medv~.,data=Boston,subset=train,mtry=6,importance=TRUE)
# yhat.rf = predict(rf.boston,newdata=Boston[-train,])
# mean((yhat.rf-boston.test)^2)
# importance(rf.boston)
# varImpPlot(rf.boston)
#
# # Boosting
#
# library(gbm)
# set.seed(1)
# boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4)
# summary(boost.boston)
# par(mfrow=c(1,2))
# plot(boost.boston,i="rm")
# plot(boost.boston,i="lstat")
# yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
# mean((yhat.boost-boston.test)^2)
```

```
# boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4,
# yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
# mean((yhat.boost-boston.test)^2)
```