8.(a)

> library(MASS)

> train = sample(1:nrow(Carseats),nrow(Carseats)/2)

(b)

> tree.carseats=tree(Sales~.,Carseats,subset = train)

> summary(tree.carseats)

Regression tree:

tree(formula = Sales ~ ., data = Carseats, subset = train)

Variables actually used in tree construction:

[1] "High" "ShelveLoc" "Price" "Education" "Population"

[6] "Age"

Number of terminal nodes: 9

Residual mean deviance: 1.543 = 294.7 / 191

Distribution of residuals:

Min. 1st Qu. Median Mean 3rd Qu. Max.

-5.0400 -0.7775 0.0146 0.0000 0.9104 3.8980

(c)

> plot(tree.carseats)

> text(tree.carseats,pretty = 0)

> cv.carseats=cv.tree(tree.carseats)

> plot(cv.carseats$size,cv.carseats$dev,type="b")

> yhat=predict(tree.carseats,newdata=Carseats[-train,])

> carseats.test=Carseats[-train,"Sales"]

> plot(yhat,carseats.test)

> abline(0,1)

> mean((yhat-carseats.test)^2)

[1] 2.927332

(d)

> install.packages("randomForest")

> library(randomForest)

> bag.carseats=randomForest(Sales~.,data=Carseats,subset = train,mtry=9,importance=TRUE)

> bag.carseats

Call:

randomForest(formula = Sales ~ ., data = Carseats, mtry = 9, importance = TRUE, subset = train)

Type of random forest: regression

Number of trees: 500

No. of variables tried at each split: 9

Mean of squared residuals: 1.843248

% Var explained: 75.52

> yhat.bag=predict(bag.carseats,newdata=Carseats[-train,])

> plot(yhat.bag,carseats.test)

> abline(0,1)

> mean((yhat.bag-carseats.test)^2)

[1] 2.167285

(e)

> rf.carseats=randomForest(Sales~.,data=Carseats,subset = train, mtry=4,importance=TRUE)

> yhat.rf=predict(rf.carseats,newdata = Carseats[-train,])

> mean((yhat.rf-carseats.test)^2)

[1] 2.087228

> importance(rf.carseats)

%IncMSE IncNodePurity

CompPrice 6.7643706 66.250117

Income -1.7535113 65.887602

Advertising 4.2538782 55.624643

Population 1.4908830 54.165153

Price 15.4674196 181.428740

ShelveLoc 23.4922556 234.784408

Age 4.2732062 89.494347

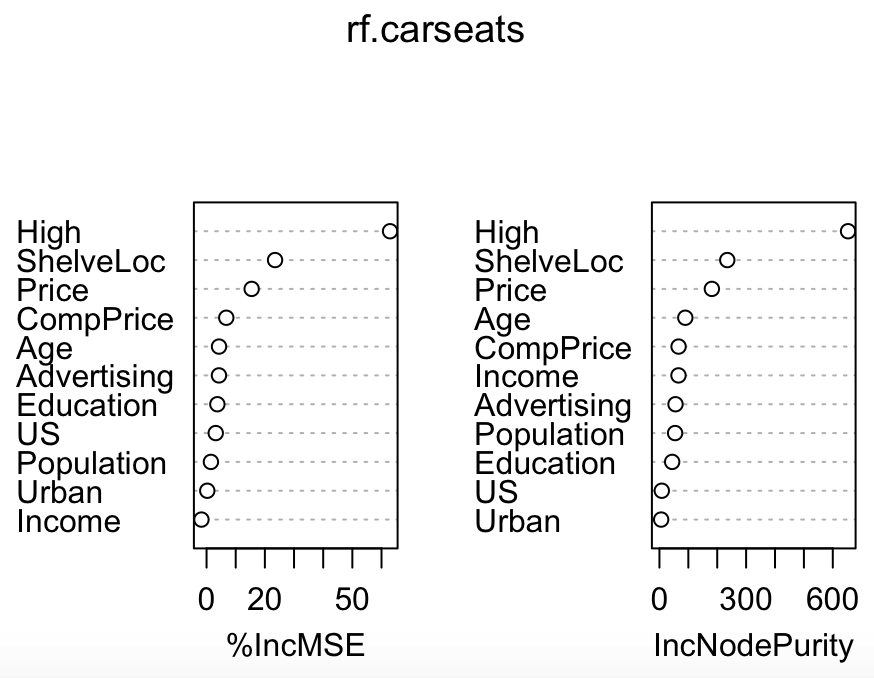
Education 3.7054991 44.037379

Urban 0.2458298 5.943982

US 3.1537085 8.135322

High 62.9306943 652.819074

> varImpPlot(rf.carseats)



9.(a)

> library("ISLR", lib.loc="~/R/win-library/3.4")

> library("tree", lib.loc="~/R/win-library/3.4")

> names(OJ)

> dim(OJ)

> summary(OJ)

> train = sample(1:nrow(OJ),800)

> test = OJ[-train,]

(b)

> tree.oj=tree(Purchase~.,data=OJ,subset = train)

> summary(tree.oj)

Classification tree:

tree(formula = Purchase ~ ., data = OJ, subset = train)

Variables actually used in tree construction:

[1] "LoyalCH" "PriceDiff"

Number of terminal nodes: 7

Residual mean deviance: 0.7785 = 617.4 / 793

Misclassification error rate: 0.1662 = 133 / 800

> tree.oj

node), split, n, deviance, yval, (yprob)

\* denotes terminal node

1) root 800 1063.000 CH ( 0.62000 0.38000 )

2) LoyalCH < 0.5036 355 434.600 MM ( 0.30141 0.69859 )

4) LoyalCH < 0.276142 169 145.100 MM ( 0.15385 0.84615 )

8) LoyalCH < 0.0356415 54 9.959 MM ( 0.01852 0.98148 ) \*

9) LoyalCH > 0.0356415 115 120.400 MM ( 0.21739 0.78261 ) \*

5) LoyalCH > 0.276142 186 254.700 MM ( 0.43548 0.56452 )

10) PriceDiff < 0.05 74 77.270 MM ( 0.21622 0.78378 ) \*

11) PriceDiff > 0.05 112 152.400 CH ( 0.58036 0.41964 ) \*

3) LoyalCH > 0.5036 445 336.800 CH ( 0.87416 0.12584 )

6) LoyalCH < 0.764572 190 212.600 CH ( 0.75263 0.24737 )

12) PriceDiff < -0.165 28 33.500 MM ( 0.28571 0.71429 ) \*

13) PriceDiff > -0.165 162 146.000 CH ( 0.83333 0.16667 ) \*

7) LoyalCH > 0.764572 255 77.870 CH ( 0.96471 0.03529 ) \*

> tree.pred=predict(tree.oj,OJ[-train,],type = "class")

> table(tree.pred,OJ[-train,]$Purchase)

tree.pred CH MM

CH 146 31

MM 11 82

> (146+82)/200

[1] 1.14

> (146+82)/270

[1] 0.8444444

(c,d)

> plot(tree.oj)

> text(tree.oj,pretty = 0)

(e)

(f)

> cv.oj=cv.tree(tree.oj,FUN = prune.misclass)

> names(cv.oj)

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

> cv.oj

$size

[1] 7 6 4 2 1

$dev

[1] 161 160 176 179 305

$k

[1] -Inf 0 6 9 141

$method

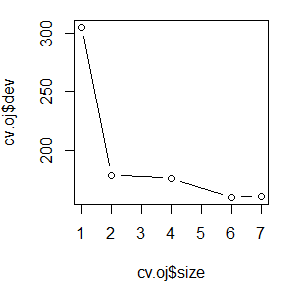
[1] "misclass"

attr(,"class")

[1] "prune" "tree.sequence"

> plot(cv.oj$size,cv.oj$dev,type = 'b')

> plot(cv.oj$k,cv.oj$dev,type = 'b')

(g) 

(h)6

(i)

> plot(cv.oj$size,cv.oj$dev,type = 'b')

> plot(cv.oj$k,cv.oj$dev,type = 'b')

> prune.oj=prune.misclass(tree.oj,best = 6)

> tree.pred=predict(prune.oj,OJ[-train,],type = "class")

> table(tree.pred,Purchase.test)

Purchase.test

tree.pred CH MM

CH 146 31

MM 11 82

> (146+82)/270

[1] 0.8444444

(j)

> tree.pred=predict(tree.oj,OJ[train,],type = "class")

> table(tree.pred,OJ[train,]$Purchase)

tree.pred CH MM

CH 446 83

MM 50 221

> (446+221)/800

[1] 0.83375

> tree.pred=predict(prune.oj,OJ[train,],type = "class")

> table(tree.pred,OJ[train,]$Purchase)

tree.pred CH MM

CH 446 83

MM 50 221

(k)the test error rates pruned trees shows is the same

10.(a)

> install.packages("gbm")

> fix(Hitters)

> names(Hitters)

[1] "AtBat" "Hits" "HmRun" "Runs"

[5] "RBI" "Walks" "Years" "CAtBat"

[9] "CHits" "CHmRun" "CRuns" "CRBI"

[13] "CWalks" "League" "Division" "PutOuts"

[17] "Assists" "Errors" "Salary" "NewLeague"

> dim(Hitters)

[1] 322 20

> Hitters=na.omit(Hitters)

> dim(Hitters)

[1] 263 20

> Hitters$Salary=log(Hitters$Salary)

(b)

> train=sample(1:nrow(Hitters),200)

> Hitters.test=Hitters[-train,]

(c)

Depth:2 shrinkage:0.001

>boost.hitters=gbm(Salary~.,data=Hitters[train,],distribution="gaussian",n.trees=1000,interaction.depth=2)

> summary(boost.hitters)

var rel.inf

CHits CHits 22.608197970

CRuns CRuns 22.328924101

CAtBat CAtBat 21.019532536

CRBI CRBI 15.252520690

CWalks CWalks 5.094487151

AtBat AtBat 3.085750950

Walks Walks 3.019251552

RBI RBI 1.883650638

CHmRun CHmRun 1.381220043

Hits Hits 1.321297372

Years Years 1.093532336

PutOuts PutOuts 0.673957336

Runs Runs 0.634362337

HmRun HmRun 0.504401041

Division Division 0.089322213

Errors Errors 0.009591734

League League 0.000000000

Assists Assists 0.000000000

NewLeague NewLeague 0.000000000

> yhat.boost=predict(boost.hitters,newdata = Hitters.test,n.trees = 1000)

> mean((yhat.boost-Hitters.test$Salary)^2)

[1] 0.3214179

Depth:1 shrinkage:0.001

>boost.hitters2=gbm(Hitters$Salary[train]~.,data=Hitters[train,],distribution="gaussian",n.trees=1000,interaction.depth=1)

> summary(boost.hitters2)

var rel.inf

CHits CHits 26.29352269

CAtBat CAtBat 24.17814131

CRuns CRuns 23.64226660

CRBI CRBI 17.00350368

CWalks CWalks 4.68233228

CHmRun CHmRun 1.18903191

Years Years 1.03846286

RBI RBI 0.82973410

Hits Hits 0.79537035

Walks Walks 0.30209679

HmRun HmRun 0.04553743

AtBat AtBat 0.00000000

Runs Runs 0.00000000

League League 0.00000000

Division Division 0.00000000

PutOuts PutOuts 0.00000000

Assists Assists 0.00000000

Errors Errors 0.00000000

NewLeague NewLeague 0.00000000

> yhat.boost2=predict(boost.hitters,newdata = Hitters.test,n.trees = 1000)

> mean((yhat.boost2-Hitters.test$Salary)^2)

[1] 0.3509733

Depth:1, shrinkage:0.2

>boost.hitters3=gbm(Hitters$Salary[train]~.,data=Hitters[train,],distribution="gaussian",n.trees=1000,interaction.depth=1,shrinkage = 0.2,verbose = F)

> yhat.boost3=predict(boost.hitters3,newdata = Hitters.test,n.trees = 1000)

> mean((yhat.boost3-Hitters.test$Salary)^2)

[1] 0.2375388

Depth:2, shrinkage:0.2

>boost.hitters4=gbm(Hitters$Salary[train]~.,data=Hitters[train,],distribution="gaussian",n.trees=1000,interaction.depth=2,shrinkage = 0.2,verbose = F)

> yhat.boost4=predict(boost.hitters4,newdata = Hitters.test,n.trees = 1000)

> mean((yhat.boost4-Hitters.test$Salary)^2)

[1] 0.2937826

(f)

Chits/CRuns/CAtBat

(g)

> bag.hitters=randomForest(Salary~.,data=Hitters,subset=train,mtry=19,importance=TRUE)

> bag.hitters

Call:

randomForest(formula = Salary ~ ., data = Hitters, mtry = 19, importance = TRUE, subset = train)

Type of random forest: regression

Number of trees: 500

No. of variables tried at each split: 19

Mean of squared residuals: 0.2160861

% Var explained: 73.49

> yhat.bag=predict(bag.hitters,newdata = Hitters.test)

> mean((yhat.bag-Hitters.test$Salary)^2)

[1] 0.2028371