HW5-1\_吳明軒

RE6124019\_吳明軒

2023-12-04

## 6.5 Lab: Linear Models and Regularization Methods

library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.3.2

View(Hitters)  
names(Hitters)

## [1] "AtBat" "Hits" "HmRun" "Runs" "RBI" "Walks"   
## [7] "Years" "CAtBat" "CHits" "CHmRun" "CRuns" "CRBI"   
## [13] "CWalks" "League" "Division" "PutOuts" "Assists" "Errors"   
## [19] "Salary" "NewLeague"

dim(Hitters)

## [1] 322 20

sum(is.na(Hitters$Salary))

## [1] 59

Hitters<-na.omit(Hitters)  
dim(Hitters)

## [1] 263 20

sum(is.na(Hitters))

## [1] 0

library(leaps)

## Warning: package 'leaps' was built under R version 4.3.2

regfit.full<-regsubsets(Salary~.,Hitters)  
summary(regfit.full)

## Subset selection object  
## Call: regsubsets.formula(Salary ~ ., Hitters)  
## 19 Variables (and intercept)  
## Forced in Forced out  
## AtBat FALSE FALSE  
## Hits FALSE FALSE  
## HmRun FALSE FALSE  
## Runs FALSE FALSE  
## RBI FALSE FALSE  
## Walks FALSE FALSE  
## Years FALSE FALSE  
## CAtBat FALSE FALSE  
## CHits FALSE FALSE  
## CHmRun FALSE FALSE  
## CRuns FALSE FALSE  
## CRBI FALSE FALSE  
## CWalks FALSE FALSE  
## LeagueN FALSE FALSE  
## DivisionW FALSE FALSE  
## PutOuts FALSE FALSE  
## Assists FALSE FALSE  
## Errors FALSE FALSE  
## NewLeagueN FALSE FALSE  
## 1 subsets of each size up to 8  
## Selection Algorithm: exhaustive  
## AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI  
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " "\*"   
## 2 ( 1 ) " " "\*" " " " " " " " " " " " " " " " " " " "\*"   
## 3 ( 1 ) " " "\*" " " " " " " " " " " " " " " " " " " "\*"   
## 4 ( 1 ) " " "\*" " " " " " " " " " " " " " " " " " " "\*"   
## 5 ( 1 ) "\*" "\*" " " " " " " " " " " " " " " " " " " "\*"   
## 6 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " " " " " "\*"   
## 7 ( 1 ) " " "\*" " " " " " " "\*" " " "\*" "\*" "\*" " " " "   
## 8 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " "\*" "\*" " "   
## CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN  
## 1 ( 1 ) " " " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " "   
## 3 ( 1 ) " " " " " " "\*" " " " " " "   
## 4 ( 1 ) " " " " "\*" "\*" " " " " " "   
## 5 ( 1 ) " " " " "\*" "\*" " " " " " "   
## 6 ( 1 ) " " " " "\*" "\*" " " " " " "   
## 7 ( 1 ) " " " " "\*" "\*" " " " " " "   
## 8 ( 1 ) "\*" " " "\*" "\*" " " " " " "

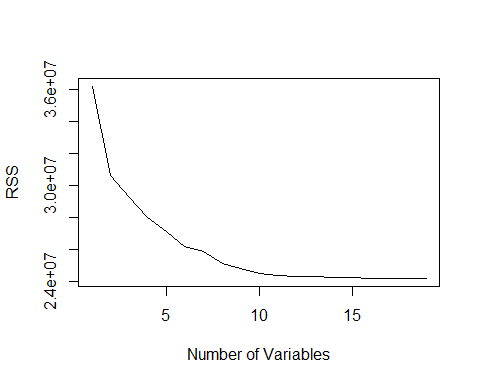
regfit.full<-regsubsets(Salary~.,data=Hitters,nvmax=19)  
reg.summary<-summary(regfit.full)  
  
names(reg.summary)

## [1] "which" "rsq" "rss" "adjr2" "cp" "bic" "outmat" "obj"

reg.summary$rsq

## [1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227  
## [8] 0.5285569 0.5346124 0.5404950 0.5426153 0.5436302 0.5444570 0.5452164  
## [15] 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159

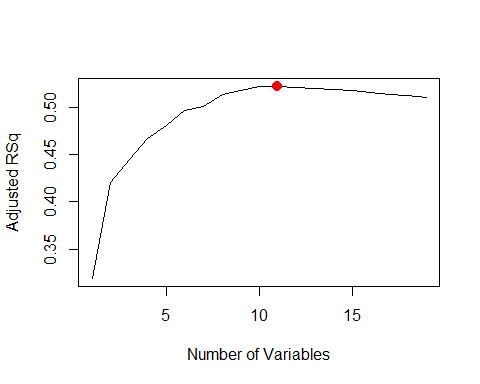
par(mfrow=c(1,1))  
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")



plot (reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")  
  
which.max(reg.summary$adjr2)

## [1] 11

points(11,reg.summary$adjr2[11],col="red",cex=2,pch=20)



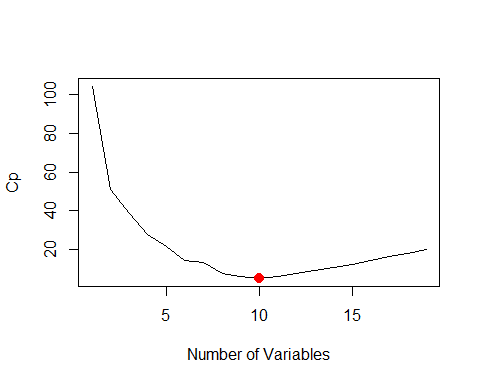
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type="l")  
which.min(reg.summary$cp)

## [1] 10

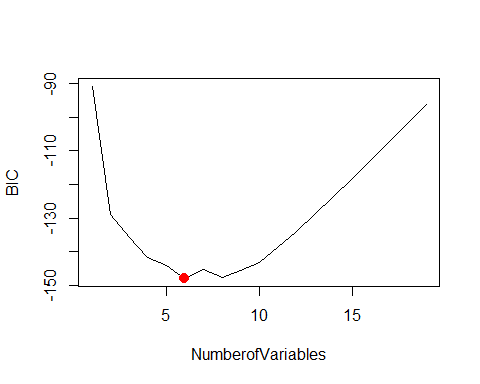
points(10,reg.summary$cp[10],col="red",cex=2,pch=20)  
which.min(reg.summary$bic)

## [1] 6

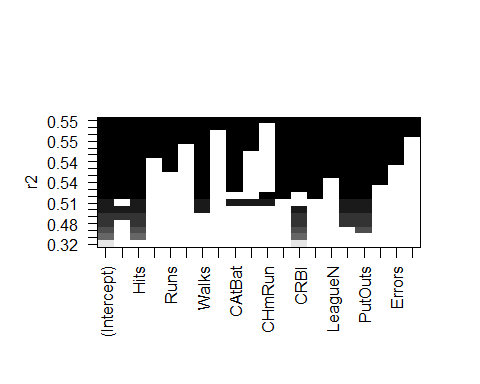
points(6,reg.summary$bic[6],col="red",cex=2,pch=20)



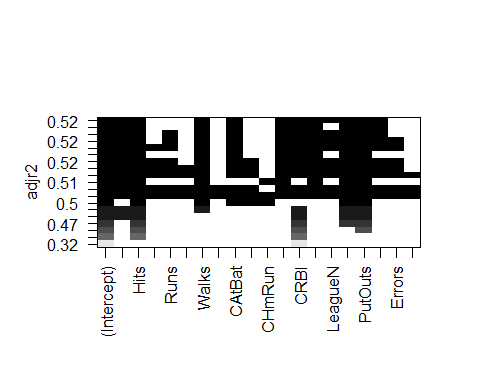
plot(reg.summary$bic,xlab="NumberofVariables",ylab="BIC",type="l")  
points (6, reg.summary$bic[6], col = "red", cex = 2, pch = 20)



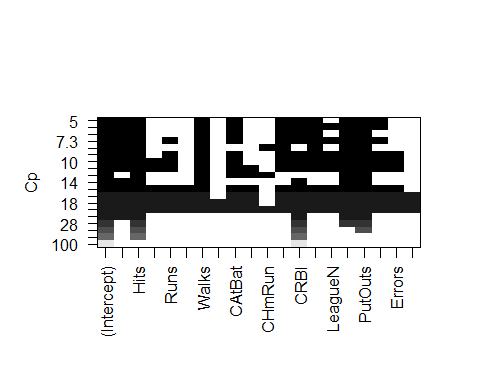
plot(regfit.full,scale="r2")



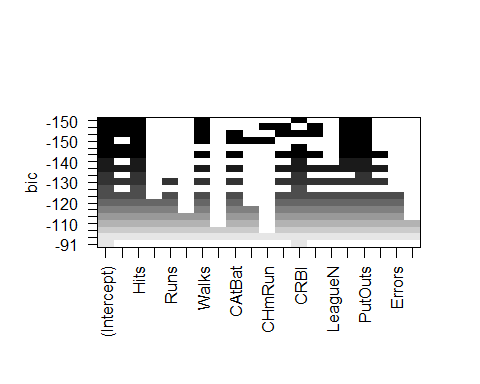
plot(regfit.full,scale="adjr2")



plot(regfit.full,scale="Cp")



plot(regfit.full,scale="bic")



coef(regfit.full,6)

## (Intercept) AtBat Hits Walks CRBI DivisionW   
## 91.5117981 -1.8685892 7.6043976 3.6976468 0.6430169 -122.9515338   
## PutOuts   
## 0.2643076

regfit.fwd<-regsubsets(Salary~.,data=Hitters,nvmax=19, method = "forward")  
summary (regfit.fwd)

## Subset selection object  
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "forward")  
## 19 Variables (and intercept)  
## Forced in Forced out  
## AtBat FALSE FALSE  
## Hits FALSE FALSE  
## HmRun FALSE FALSE  
## Runs FALSE FALSE  
## RBI FALSE FALSE  
## Walks FALSE FALSE  
## Years FALSE FALSE  
## CAtBat FALSE FALSE  
## CHits FALSE FALSE  
## CHmRun FALSE FALSE  
## CRuns FALSE FALSE  
## CRBI FALSE FALSE  
## CWalks FALSE FALSE  
## LeagueN FALSE FALSE  
## DivisionW FALSE FALSE  
## PutOuts FALSE FALSE  
## Assists FALSE FALSE  
## Errors FALSE FALSE  
## NewLeagueN FALSE FALSE  
## 1 subsets of each size up to 19  
## Selection Algorithm: forward  
## AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI  
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " "\*"   
## 2 ( 1 ) " " "\*" " " " " " " " " " " " " " " " " " " "\*"   
## 3 ( 1 ) " " "\*" " " " " " " " " " " " " " " " " " " "\*"   
## 4 ( 1 ) " " "\*" " " " " " " " " " " " " " " " " " " "\*"   
## 5 ( 1 ) "\*" "\*" " " " " " " " " " " " " " " " " " " "\*"   
## 6 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " " " " " "\*"   
## 7 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " " " " " "\*"   
## 8 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " " " "\*" "\*"   
## 9 ( 1 ) "\*" "\*" " " " " " " "\*" " " "\*" " " " " "\*" "\*"   
## 10 ( 1 ) "\*" "\*" " " " " " " "\*" " " "\*" " " " " "\*" "\*"   
## 11 ( 1 ) "\*" "\*" " " " " " " "\*" " " "\*" " " " " "\*" "\*"   
## 12 ( 1 ) "\*" "\*" " " "\*" " " "\*" " " "\*" " " " " "\*" "\*"   
## 13 ( 1 ) "\*" "\*" " " "\*" " " "\*" " " "\*" " " " " "\*" "\*"   
## 14 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" " " "\*" " " " " "\*" "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*" " " "\*" "\*"   
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" " " "\*" "\*"   
## 17 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" " " "\*" "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN  
## 1 ( 1 ) " " " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " "   
## 3 ( 1 ) " " " " " " "\*" " " " " " "   
## 4 ( 1 ) " " " " "\*" "\*" " " " " " "   
## 5 ( 1 ) " " " " "\*" "\*" " " " " " "   
## 6 ( 1 ) " " " " "\*" "\*" " " " " " "   
## 7 ( 1 ) "\*" " " "\*" "\*" " " " " " "   
## 8 ( 1 ) "\*" " " "\*" "\*" " " " " " "   
## 9 ( 1 ) "\*" " " "\*" "\*" " " " " " "   
## 10 ( 1 ) "\*" " " "\*" "\*" "\*" " " " "   
## 11 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " " "   
## 12 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " " "   
## 13 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 14 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 15 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 17 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"

regfit.bwd<-regsubsets(Salary~.,data=Hitters,nvmax=19, method = "backward")  
summary (regfit.bwd)

## Subset selection object  
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "backward")  
## 19 Variables (and intercept)  
## Forced in Forced out  
## AtBat FALSE FALSE  
## Hits FALSE FALSE  
## HmRun FALSE FALSE  
## Runs FALSE FALSE  
## RBI FALSE FALSE  
## Walks FALSE FALSE  
## Years FALSE FALSE  
## CAtBat FALSE FALSE  
## CHits FALSE FALSE  
## CHmRun FALSE FALSE  
## CRuns FALSE FALSE  
## CRBI FALSE FALSE  
## CWalks FALSE FALSE  
## LeagueN FALSE FALSE  
## DivisionW FALSE FALSE  
## PutOuts FALSE FALSE  
## Assists FALSE FALSE  
## Errors FALSE FALSE  
## NewLeagueN FALSE FALSE  
## 1 subsets of each size up to 19  
## Selection Algorithm: backward  
## AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI  
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " "\*" " "   
## 2 ( 1 ) " " "\*" " " " " " " " " " " " " " " " " "\*" " "   
## 3 ( 1 ) " " "\*" " " " " " " " " " " " " " " " " "\*" " "   
## 4 ( 1 ) "\*" "\*" " " " " " " " " " " " " " " " " "\*" " "   
## 5 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " " " "\*" " "   
## 6 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " " " "\*" " "   
## 7 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " " " "\*" " "   
## 8 ( 1 ) "\*" "\*" " " " " " " "\*" " " " " " " " " "\*" "\*"   
## 9 ( 1 ) "\*" "\*" " " " " " " "\*" " " "\*" " " " " "\*" "\*"   
## 10 ( 1 ) "\*" "\*" " " " " " " "\*" " " "\*" " " " " "\*" "\*"   
## 11 ( 1 ) "\*" "\*" " " " " " " "\*" " " "\*" " " " " "\*" "\*"   
## 12 ( 1 ) "\*" "\*" " " "\*" " " "\*" " " "\*" " " " " "\*" "\*"   
## 13 ( 1 ) "\*" "\*" " " "\*" " " "\*" " " "\*" " " " " "\*" "\*"   
## 14 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" " " "\*" " " " " "\*" "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*" " " "\*" "\*"   
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" " " "\*" "\*"   
## 17 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" " " "\*" "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN  
## 1 ( 1 ) " " " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " "   
## 3 ( 1 ) " " " " " " "\*" " " " " " "   
## 4 ( 1 ) " " " " " " "\*" " " " " " "   
## 5 ( 1 ) " " " " " " "\*" " " " " " "   
## 6 ( 1 ) " " " " "\*" "\*" " " " " " "   
## 7 ( 1 ) "\*" " " "\*" "\*" " " " " " "   
## 8 ( 1 ) "\*" " " "\*" "\*" " " " " " "   
## 9 ( 1 ) "\*" " " "\*" "\*" " " " " " "   
## 10 ( 1 ) "\*" " " "\*" "\*" "\*" " " " "   
## 11 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " " "   
## 12 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " " "   
## 13 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 14 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 15 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " "   
## 17 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"

coef(regfit.full,7)

## (Intercept) Hits Walks CAtBat CHits CHmRun   
## 79.4509472 1.2833513 3.2274264 -0.3752350 1.4957073 1.4420538   
## DivisionW PutOuts   
## -129.9866432 0.2366813

coef(regfit.fwd,7)

## (Intercept) AtBat Hits Walks CRBI CWalks   
## 109.7873062 -1.9588851 7.4498772 4.9131401 0.8537622 -0.3053070   
## DivisionW PutOuts   
## -127.1223928 0.2533404

coef(regfit.bwd,7)

## (Intercept) AtBat Hits Walks CRuns CWalks   
## 105.6487488 -1.9762838 6.7574914 6.0558691 1.1293095 -0.7163346   
## DivisionW PutOuts   
## -116.1692169 0.3028847

set.seed(1)  
train<-sample(c(TRUE,FALSE),nrow(Hitters),replace=TRUE)  
test<-(!train)  
  
regfit.best<-regsubsets(Salary~.,data=Hitters[train,],nvmax=19)  
  
test.mat<-model.matrix(Salary~.,data=Hitters[test,])  
  
val.errors<-rep(NA,19)  
for(i in 1:19){  
coefi<-coef(regfit.best,id=i)  
pred<-test.mat[,names(coefi)]%\*%coefi  
val.errors[i]<-mean((Hitters$Salary[test]-pred)^2)  
}  
  
val.errors

## [1] 164377.3 144405.5 152175.7 145198.4 137902.1 139175.7 126849.0 136191.4  
## [9] 132889.6 135434.9 136963.3 140694.9 140690.9 141951.2 141508.2 142164.4  
## [17] 141767.4 142339.6 142238.2

which.min(val.errors)

## [1] 7

coef(regfit.best,7)

## (Intercept) AtBat Hits Walks CRuns CWalks   
## 67.1085369 -2.1462987 7.0149547 8.0716640 1.2425113 -0.8337844   
## DivisionW PutOuts   
## -118.4364998 0.2526925

predict.regsubsets<-function(object,newdata,id,...){  
form<-as.formula(object$call[[2]])  
mat<-model.matrix(form,newdata)  
coefi<-coef(object,id=id)  
xvars<-names(coefi)  
mat[,xvars]%\*%coefi  
}  
  
regfit.best<-regsubsets(Salary~.,data=Hitters,nvmax=19)  
coef(regfit.best,7)

## (Intercept) Hits Walks CAtBat CHits CHmRun   
## 79.4509472 1.2833513 3.2274264 -0.3752350 1.4957073 1.4420538   
## DivisionW PutOuts   
## -129.9866432 0.2366813

k<-10  
n<-nrow(Hitters)  
set.seed(1)  
folds<-sample(rep(1:k,length=n))  
cv.errors<-matrix(NA,k,19,dimnames=list(NULL,paste(1:19)))  
  
for(j in 1:k){  
best.fit<-regsubsets(Salary~.,  
 data=Hitters[folds!=j,],  
 nvmax=19)  
for(i in 1:19){  
pred<-predict(best.fit,Hitters[folds==j,],id=i)  
cv.errors[j,i]<-mean((Hitters$Salary[folds==j]-pred)^2)  
}  
}  
  
mean.cv.errors<-apply(cv.errors,2,mean)  
mean.cv.errors

## 1 2 3 4 5 6 7 8   
## 143439.8 126817.0 134214.2 131782.9 130765.6 120382.9 121443.1 114363.7   
## 9 10 11 12 13 14 15 16   
## 115163.1 109366.0 112738.5 113616.5 115557.6 115853.3 115630.6 116050.0   
## 17 18 19   
## 116117.0 116419.3 116299.1

par(mfrow=c(1,1))  
plot(mean.cv.errors,type="b")  
  
reg.best<-regsubsets(Salary~.,data=Hitters,nvmax=19)  
coef(reg.best,10)

## (Intercept) AtBat Hits Walks CAtBat CRuns   
## 162.5354420 -2.1686501 6.9180175 5.7732246 -0.1300798 1.4082490   
## CRBI CWalks DivisionW PutOuts Assists   
## 0.7743122 -0.8308264 -112.3800575 0.2973726 0.2831680

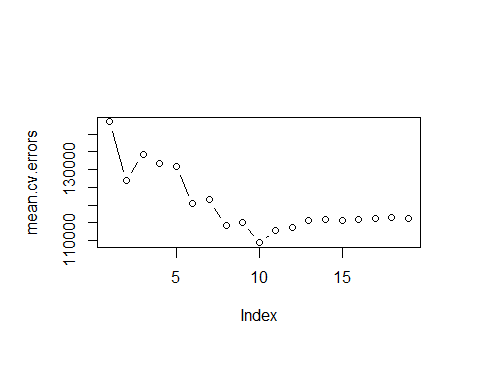
x <- model.matrix ( Salary~., Hitters )[, -1]  
y <- Hitters$Salary  
  
library(glmnet)

## Warning: package 'glmnet' was built under R version 4.3.2

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.3.2

## Loaded glmnet 4.1-8



grid <- 10^seq(10,-2,length=100)  
ridge.mod <- glmnet(x,y,alpha=0,lambda=grid)  
  
dim(coef(ridge.mod))

## [1] 20 100

ridge.mod$lambda[50]

## [1] 11497.57

coef(ridge.mod)[,50]

## (Intercept) AtBat Hits HmRun Runs   
## 407.356050200 0.036957182 0.138180344 0.524629976 0.230701523   
## RBI Walks Years CAtBat CHits   
## 0.239841459 0.289618741 1.107702929 0.003131815 0.011653637   
## CHmRun CRuns CRBI CWalks LeagueN   
## 0.087545670 0.023379882 0.024138320 0.025015421 0.085028114   
## DivisionW PutOuts Assists Errors NewLeagueN   
## -6.215440973 0.016482577 0.002612988 -0.020502690 0.301433531

sqrt(sum(coef(ridge.mod)[-1,50]^2))

## [1] 6.360612

ridge.mod$lambda[60]

## [1] 705.4802

coef(ridge.mod)[,60]

## (Intercept) AtBat Hits HmRun Runs RBI   
## 54.32519950 0.11211115 0.65622409 1.17980910 0.93769713 0.84718546   
## Walks Years CAtBat CHits CHmRun CRuns   
## 1.31987948 2.59640425 0.01083413 0.04674557 0.33777318 0.09355528   
## CRBI CWalks LeagueN DivisionW PutOuts Assists   
## 0.09780402 0.07189612 13.68370191 -54.65877750 0.11852289 0.01606037   
## Errors NewLeagueN   
## -0.70358655 8.61181213

sqrt(sum(coef(ridge.mod)[-1,60]^2))

## [1] 57.11001

predict(ridge.mod,s=50,type="coefficients")[1:20,]

## (Intercept) AtBat Hits HmRun Runs   
## 4.876610e+01 -3.580999e-01 1.969359e+00 -1.278248e+00 1.145892e+00   
## RBI Walks Years CAtBat CHits   
## 8.038292e-01 2.716186e+00 -6.218319e+00 5.447837e-03 1.064895e-01   
## CHmRun CRuns CRBI CWalks LeagueN   
## 6.244860e-01 2.214985e-01 2.186914e-01 -1.500245e-01 4.592589e+01   
## DivisionW PutOuts Assists Errors NewLeagueN   
## -1.182011e+02 2.502322e-01 1.215665e-01 -3.278600e+00 -9.496680e+00

set.seed(1)  
train <- sample(1:nrow(x),nrow(x)/2)   
test <- (-train)  
y.test <- y[test]  
  
ridge.mod <- glmnet(x[train,],y[train],alpha=0,lambda=grid,thresh=1e-12)  
ridge.pred <- predict(ridge.mod,s=4,newx=x[test,])  
mean((ridge.pred-y.test)^2)

## [1] 142199.2

mean((mean(y[train])-y.test)^2)

## [1] 224669.9

ridge.pred <- predict(ridge.mod,s=0,newx=x[test,],exact=T,x=x[train,],y=y[train])  
mean((ridge.pred-y.test)^2)

## [1] 168588.6

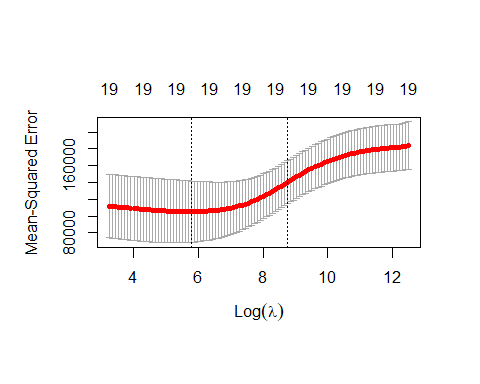
lm(y ~ x, subset = train)

##   
## Call:  
## lm(formula = y ~ x, subset = train)  
##   
## Coefficients:  
## (Intercept) xAtBat xHits xHmRun xRuns xRBI   
## 274.0145 -0.3521 -1.6377 5.8145 1.5424 1.1243   
## xWalks xYears xCAtBat xCHits xCHmRun xCRuns   
## 3.7287 -16.3773 -0.6412 3.1632 3.4008 -0.9739   
## xCRBI xCWalks xLeagueN xDivisionW xPutOuts xAssists   
## -0.6005 0.3379 119.1486 -144.0831 0.1976 0.6804   
## xErrors xNewLeagueN   
## -4.7128 -71.0951

predict ( ridge.mod , s = 0, exact = T , type = "coefficients",  
 x = x [ train , ], y = y[ train ]) [1:20 , ]

## (Intercept) AtBat Hits HmRun Runs RBI   
## 274.0200994 -0.3521900 -1.6371383 5.8146692 1.5423361 1.1241837   
## Walks Years CAtBat CHits CHmRun CRuns   
## 3.7288406 -16.3795195 -0.6411235 3.1629444 3.4005281 -0.9739405   
## CRBI CWalks LeagueN DivisionW PutOuts Assists   
## -0.6003976 0.3378422 119.1434637 -144.0853061 0.1976300 0.6804200   
## Errors NewLeagueN   
## -4.7127879 -71.0898914

set.seed(1)  
cv.out <- cv.glmnet(x[train,],y[train],alpha=0)   
plot(cv.out)



bestlam <- cv.out$lambda.min  
bestlam

## [1] 326.0828

ridge.pred <- predict(ridge.mod,s=bestlam,newx=x[test,])  
  
mean((ridge.pred-y.test)^2)

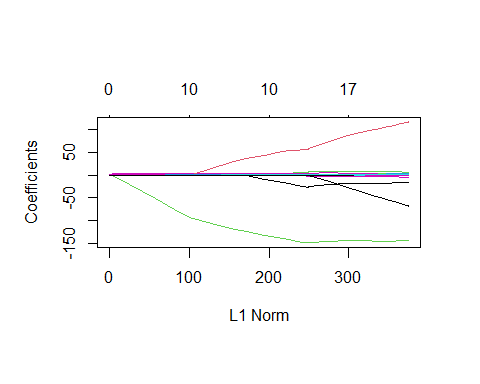
## [1] 139856.6

out <- glmnet(x,y,alpha=0)  
predict(out,type="coefficients",s=bestlam)[1:20,]

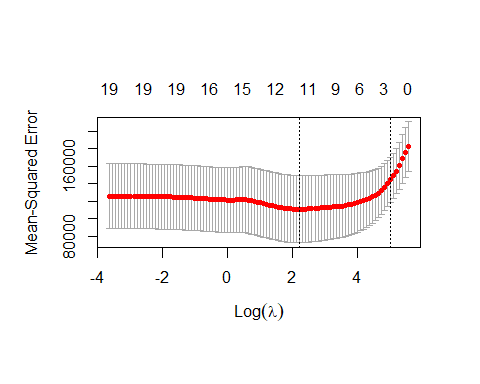
## (Intercept) AtBat Hits HmRun Runs RBI   
## 15.44383120 0.07715547 0.85911582 0.60103106 1.06369007 0.87936105   
## Walks Years CAtBat CHits CHmRun CRuns   
## 1.62444617 1.35254778 0.01134999 0.05746654 0.40680157 0.11456224   
## CRBI CWalks LeagueN DivisionW PutOuts Assists   
## 0.12116504 0.05299202 22.09143197 -79.04032656 0.16619903 0.02941950   
## Errors NewLeagueN   
## -1.36092945 9.12487765

lasso.mod <- glmnet(x[train, ], y[train], alpha = 1, lambda = grid)  
plot(lasso.mod)

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values



set.seed(1)  
cv.out <- cv.glmnet(x[train, ],y[train],alpha=1)   
plot(cv.out)



bestlam <- cv.out$lambda.min  
lasso.pred <- predict (lasso.mod , s = bestlam ,  
 newx = x[ test , ])  
mean((lasso.pred - y.test)^2)

## [1] 143673.6

out <- glmnet(x,y, alpha = 1, lambda = grid)  
lasso.coef <- predict (out , type = "coefficients",s = bestlam )[1:20 , ]  
lasso.coef

## (Intercept) AtBat Hits HmRun Runs   
## 1.27479059 -0.05497143 2.18034583 0.00000000 0.00000000   
## RBI Walks Years CAtBat CHits   
## 0.00000000 2.29192406 -0.33806109 0.00000000 0.00000000   
## CHmRun CRuns CRBI CWalks LeagueN   
## 0.02825013 0.21628385 0.41712537 0.00000000 20.28615023   
## DivisionW PutOuts Assists Errors NewLeagueN   
## -116.16755870 0.23752385 0.00000000 -0.85629148 0.00000000

library(pls)

## Warning: package 'pls' was built under R version 4.3.2

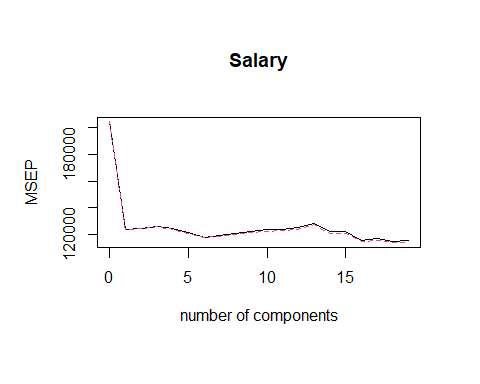
##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

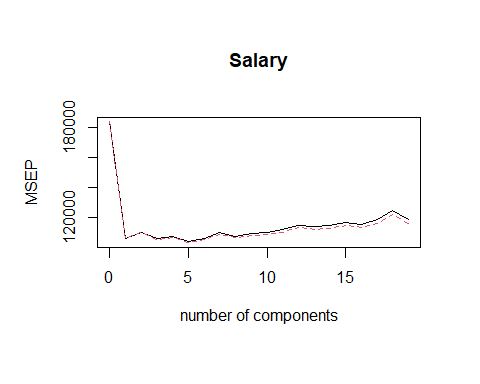
set.seed(2)  
pcr.fit <- pcr ( Salary ~ ., data = Hitters , scale = TRUE , validation = "CV")  
  
summary(pcr.fit)

## Data: X dimension: 263 19   
## Y dimension: 263 1  
## Fit method: svdpc  
## Number of components considered: 19  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 452 351.9 353.2 355.0 352.8 348.4 343.6  
## adjCV 452 351.6 352.7 354.4 352.1 347.6 342.7  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 345.5 347.7 349.6 351.4 352.1 353.5 358.2  
## adjCV 344.7 346.7 348.5 350.1 350.7 352.0 356.5  
## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
## CV 349.7 349.4 339.9 341.6 339.2 339.6  
## adjCV 348.0 347.7 338.2 339.7 337.2 337.6  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 38.31 60.16 70.84 79.03 84.29 88.63 92.26 94.96  
## Salary 40.63 41.58 42.17 43.22 44.90 46.48 46.69 46.75  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 96.28 97.26 97.98 98.65 99.15 99.47 99.75  
## Salary 46.86 47.76 47.82 47.85 48.10 50.40 50.55  
## 16 comps 17 comps 18 comps 19 comps  
## X 99.89 99.97 99.99 100.00  
## Salary 53.01 53.85 54.61 54.61

validationplot (pcr.fit , val.type = "MSEP")



set.seed (1)  
pcr.fit <- pcr(Salary~., data = Hitters , subset = train ,  
 scale = TRUE , validation = "CV")  
validationplot (pcr.fit , val.type = "MSEP")



pcr.pred <- predict ( pcr.fit , x[ test , ], ncomp = 5)  
mean (( pcr.pred - y.test ) ^2)

## [1] 142811.8

pcr.fit <- pcr (y ~ x , scale = TRUE , ncomp = 5)  
summary ( pcr.fit )

## Data: X dimension: 263 19   
## Y dimension: 263 1  
## Fit method: svdpc  
## Number of components considered: 5  
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps  
## X 38.31 60.16 70.84 79.03 84.29  
## y 40.63 41.58 42.17 43.22 44.90

set.seed (1)  
pls.fit <- plsr ( Salary ~ ., data = Hitters , subset = train ,  
 scale = TRUE , validation = "CV")  
summary (pls.fit)

## Data: X dimension: 131 19   
## Y dimension: 131 1  
## Fit method: kernelpls  
## Number of components considered: 19  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 428.3 325.5 329.9 328.8 339.0 338.9 340.1  
## adjCV 428.3 325.0 328.2 327.2 336.6 336.1 336.6  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 339.0 347.1 346.4 343.4 341.5 345.4 356.4  
## adjCV 336.2 343.4 342.8 340.2 338.3 341.8 351.1  
## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
## CV 348.4 349.1 350.0 344.2 344.5 345.0  
## adjCV 344.2 345.0 345.9 340.4 340.6 341.1  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 39.13 48.80 60.09 75.07 78.58 81.12 88.21 90.71  
## Salary 46.36 50.72 52.23 53.03 54.07 54.77 55.05 55.66  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 93.17 96.05 97.08 97.61 97.97 98.70 99.12  
## Salary 55.95 56.12 56.47 56.68 57.37 57.76 58.08  
## 16 comps 17 comps 18 comps 19 comps  
## X 99.61 99.70 99.95 100.00  
## Salary 58.17 58.49 58.56 58.62

pls.pred <- predict ( pls.fit , x[ test , ], ncomp = 1)  
  
mean((pls.pred - y.test) ^2)

## [1] 151995.3

pls.fit <- plsr ( Salary ~., data = Hitters , scale = TRUE ,  
 ncomp = 1)  
summary ( pls.fit )

## Data: X dimension: 263 19   
## Y dimension: 263 1  
## Fit method: kernelpls  
## Number of components considered: 1  
## TRAINING: % variance explained  
## 1 comps  
## X 38.08  
## Salary 43.05

## 7.8 Lab: Non-linear Modeling

library(ISLR2)  
attach(Wage)  
  
fit <- lm(wage ~ poly(age, 4), data = Wage)  
coef(summary(fit))

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 111.70361 0.7287409 153.283015 0.000000e+00  
## poly(age, 4)1 447.06785 39.9147851 11.200558 1.484604e-28  
## poly(age, 4)2 -478.31581 39.9147851 -11.983424 2.355831e-32  
## poly(age, 4)3 125.52169 39.9147851 3.144742 1.678622e-03  
## poly(age, 4)4 -77.91118 39.9147851 -1.951938 5.103865e-02

fit2 <- lm(wage ~ poly(age, 4, raw = T), data = Wage)  
coef(summary(fit2))

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -1.841542e+02 6.004038e+01 -3.067172 0.0021802539  
## poly(age, 4, raw = T)1 2.124552e+01 5.886748e+00 3.609042 0.0003123618  
## poly(age, 4, raw = T)2 -5.638593e-01 2.061083e-01 -2.735743 0.0062606446  
## poly(age, 4, raw = T)3 6.810688e-03 3.065931e-03 2.221409 0.0263977518  
## poly(age, 4, raw = T)4 -3.203830e-05 1.641359e-05 -1.951938 0.0510386498

fit2a <- lm(wage ~ age + I(age^2) + I(age^3) + I(age^4), data = Wage)  
coef(fit2a)

## (Intercept) age I(age^2) I(age^3) I(age^4)   
## -1.841542e+02 2.124552e+01 -5.638593e-01 6.810688e-03 -3.203830e-05

fit2b <- lm(wage ~ cbind(age, age^2, age^3, age^4), data = Wage)  
  
agelims <- range(age)  
age.grid <- seq(from = agelims[1], to = agelims[2])  
preds <- predict(fit, newdata = list(age = age.grid), se = TRUE)  
se.bands <- cbind(preds$fit+2\*preds$se.fit, preds$fit-2\*preds$se.fit)  
  
par(mfrow = c(1, 2), mar = c(4.5, 4.5, 1, 1), oma = c(0, 0, 4, 0))  
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")  
title("Degree-4 Polynomial", outer = T)  
lines(age.grid, preds$fit,lwd=2,col="blue")  
matlines(age.grid,se.bands,lwd=1,col="blue",lty=3)  
  
preds2<-predict(fit2,newdata=list(age=age.grid),se=TRUE)  
max(abs(preds$fit - preds2$fit))

## [1] 6.842527e-11

fit.1 <- lm(wage ~ age, data = Wage)  
fit.2 <- lm(wage ~ poly(age, 2), data = Wage)  
fit.3 <- lm(wage ~ poly(age, 3), data = Wage)  
fit.4 <- lm(wage ~ poly(age, 4), data = Wage)  
fit.5 <- lm(wage ~ poly(age, 5), data = Wage)  
anova (fit.1 , fit.2 , fit.3 , fit.4 , fit.5)

## Analysis of Variance Table  
##   
## Model 1: wage ~ age  
## Model 2: wage ~ poly(age, 2)  
## Model 3: wage ~ poly(age, 3)  
## Model 4: wage ~ poly(age, 4)  
## Model 5: wage ~ poly(age, 5)  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 2998 5022216   
## 2 2997 4793430 1 228786 143.5931 < 2.2e-16 \*\*\*  
## 3 2996 4777674 1 15756 9.8888 0.001679 \*\*   
## 4 2995 4771604 1 6070 3.8098 0.051046 .   
## 5 2994 4770322 1 1283 0.8050 0.369682   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

coef(summary(fit.5))

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 111.70361 0.7287647 153.2780243 0.000000e+00  
## poly(age, 5)1 447.06785 39.9160847 11.2001930 1.491111e-28  
## poly(age, 5)2 -478.31581 39.9160847 -11.9830341 2.367734e-32  
## poly(age, 5)3 125.52169 39.9160847 3.1446392 1.679213e-03  
## poly(age, 5)4 -77.91118 39.9160847 -1.9518743 5.104623e-02  
## poly(age, 5)5 -35.81289 39.9160847 -0.8972045 3.696820e-01

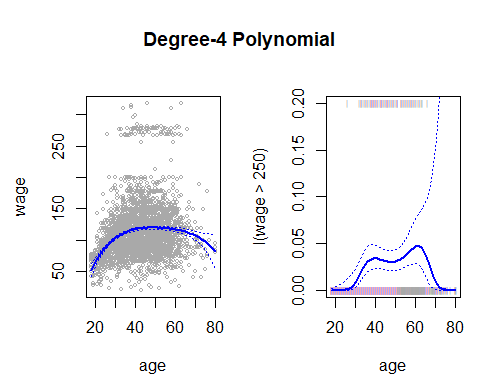
(-11.983) ^2

## [1] 143.5923

fit.1 <- lm(wage ~ education + age, data = Wage)  
fit.2 <- lm(wage ~ education + poly(age, 2), data = Wage)  
fit.3 <- lm(wage ~ education + poly(age, 3), data = Wage)  
anova (fit.1 , fit.2 , fit.3)

## Analysis of Variance Table  
##   
## Model 1: wage ~ education + age  
## Model 2: wage ~ education + poly(age, 2)  
## Model 3: wage ~ education + poly(age, 3)  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 2994 3867992   
## 2 2993 3725395 1 142597 114.6969 <2e-16 \*\*\*  
## 3 2992 3719809 1 5587 4.4936 0.0341 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

fit <- glm (I( wage > 250) ~ poly ( age , 4) , data = Wage ,family = binomial )  
  
preds <- predict ( fit , newdata = list ( age = age.grid ) , se = T)  
  
pfit <- exp(preds$fit)/(1+exp(preds$fit))  
se.bands.logit <- cbind ( preds$fit + 2 \* preds$se.fit ,  
 preds $ fit - 2 \* preds$se.fit )  
se.bands <- exp ( se.bands.logit ) / (1 + exp ( se.bands.logit ))  
  
preds <- predict ( fit , newdata = list ( age = age.grid ) ,type = "response", se = T)  
  
plot(age, I(wage > 250), xlim = agelims, type = "n", ylim = c(0, .2))  
points(jitter(age), I((wage > 250)/5), cex = .5, pch = "|", col = "darkgrey")  
lines(age.grid, pfit, lwd = 2, col = "blue")  
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)



table(cut(age, 4))

##   
## (17.9,33.5] (33.5,49] (49,64.5] (64.5,80.1]   
## 750 1399 779 72

fit <- lm(wage ~ cut(age, 4), data = Wage)  
coef(summary(fit))

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 94.158392 1.476069 63.789970 0.000000e+00  
## cut(age, 4)(33.5,49] 24.053491 1.829431 13.148074 1.982315e-38  
## cut(age, 4)(49,64.5] 23.664559 2.067958 11.443444 1.040750e-29  
## cut(age, 4)(64.5,80.1] 7.640592 4.987424 1.531972 1.256350e-01

library(splines)  
fit <- lm(wage ~ bs(age, knots = c(25, 40, 60)), data = Wage)  
pred <- predict(fit, newdata = list(age = age.grid), se = T)  
plot(age, wage, col = "gray")  
lines(age.grid, pred$fit,lwd=2)  
lines(age.grid,pred$fit + 2 \* pred$se,lty="dashed")  
lines(age.grid,pred$fit - 2 \* pred$se, lty = "dashed")  
  
dim (bs( age , knots = c(25 , 40 , 60) ))

## [1] 3000 6

dim (bs( age , df = 6) )

## [1] 3000 6

attr (bs( age , df = 6) , "knots")

## [1] 33.75 42.00 51.00

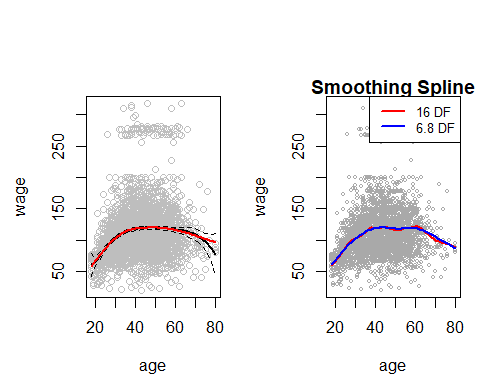
fit2<-lm(wage ~ ns(age,df=4),data=Wage)  
pred2<-predict(fit2,newdata=list(age=age.grid),se=T)  
lines(age.grid,pred2$fit, col = "red", lwd = 2)  
  
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")  
title("Smoothing Spline")  
fit <- smooth.spline(age, wage, df = 16)  
fit2 <- smooth.spline(age, wage, cv = TRUE)

## Warning in smooth.spline(age, wage, cv = TRUE): cross-validation with  
## non-unique 'x' values seems doubtful

fit2$df

## [1] 6.794596

lines(fit, col = "red", lwd = 2)  
lines(fit2, col = "blue", lwd = 2)  
legend("topright", legend = c("16 DF", "6.8 DF"), col = c("red", "blue"), lty = 1, lwd = 2, cex = .8)



plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")  
title("Local Regression")  
fit <- loess(wage ~ age, span = .2, data = Wage)  
fit2 <- loess(wage ~ age, span = .5, data = Wage)  
lines(age.grid, predict(fit, data.frame(age = age.grid)), col = "red", lwd = 2)  
lines(age.grid, predict(fit2, data.frame(age = age.grid)), col = "blue", lwd = 2)  
legend("topright", legend = c("Span = 0.2", "Span = 0.5"), col = c("red", "blue"), lty = 1, lwd = 2, cex = .8)  
  
gam1 <- lm(wage ~ ns(year, 4) + ns(age, 5) + education, data = Wage)  
  
library(gam)

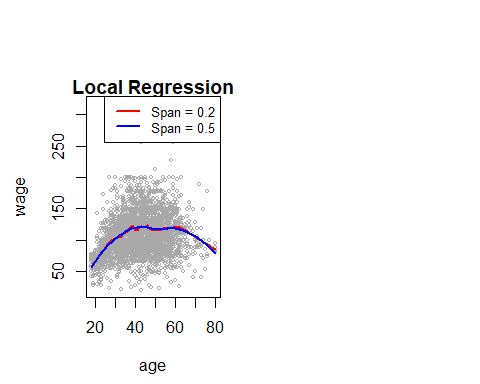
## Warning: package 'gam' was built under R version 4.3.2

## Loading required package: foreach

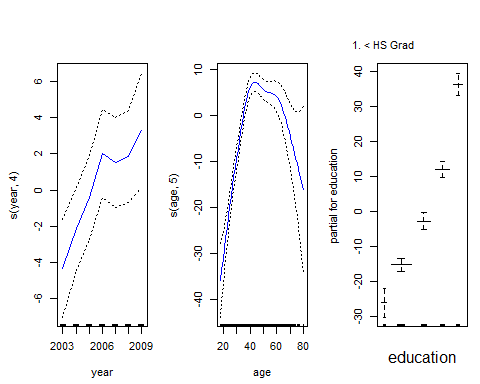
## Warning: package 'foreach' was built under R version 4.3.2

## Loaded gam 1.22-3

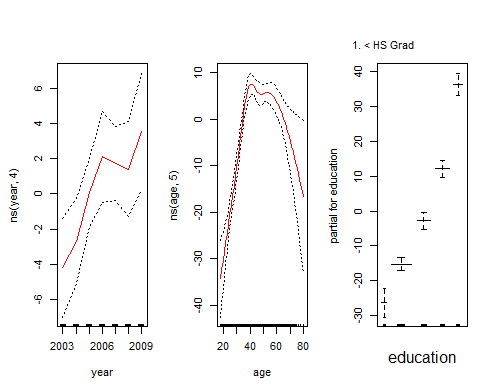
gam.m3 <- gam(wage ~ s(year, 4) + s(age, 5) + education, data = Wage)  
  
par(mfrow = c(1, 3))



plot(gam.m3, se = TRUE, col = "blue")



plot.Gam ( gam1 , se = TRUE , col = "red")



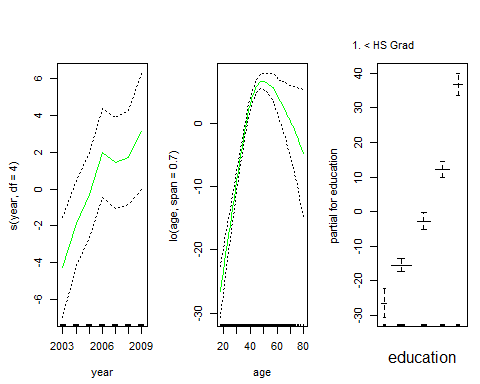
gam.m1 <- gam(wage ~ s(age, 5) + education, data = Wage)  
gam.m2 <- gam(wage ~ year + s(age, 5) + education, data = Wage)  
anova(gam.m1, gam.m2, gam.m3, test = "F")

## Analysis of Deviance Table  
##   
## Model 1: wage ~ s(age, 5) + education  
## Model 2: wage ~ year + s(age, 5) + education  
## Model 3: wage ~ s(year, 4) + s(age, 5) + education  
## Resid. Df Resid. Dev Df Deviance F Pr(>F)   
## 1 2990 3711731   
## 2 2989 3693842 1 17889.2 14.4771 0.0001447 \*\*\*  
## 3 2986 3689770 3 4071.1 1.0982 0.3485661   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

summary(gam.m3)

##   
## Call: gam(formula = wage ~ s(year, 4) + s(age, 5) + education, data = Wage)  
## Deviance Residuals:  
## Min 1Q Median 3Q Max   
## -119.43 -19.70 -3.33 14.17 213.48   
##   
## (Dispersion Parameter for gaussian family taken to be 1235.69)  
##   
## Null Deviance: 5222086 on 2999 degrees of freedom  
## Residual Deviance: 3689770 on 2986 degrees of freedom  
## AIC: 29887.75   
##   
## Number of Local Scoring Iterations: NA   
##   
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## s(year, 4) 1 27162 27162 21.981 2.877e-06 \*\*\*  
## s(age, 5) 1 195338 195338 158.081 < 2.2e-16 \*\*\*  
## education 4 1069726 267432 216.423 < 2.2e-16 \*\*\*  
## Residuals 2986 3689770 1236   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Anova for Nonparametric Effects  
## Npar Df Npar F Pr(F)   
## (Intercept)   
## s(year, 4) 3 1.086 0.3537   
## s(age, 5) 4 32.380 <2e-16 \*\*\*  
## education   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

preds <- predict(gam.m2, newdata = Wage)  
gam.lo <- gam(wage ~ s(year, df = 4) + lo(age, span = 0.7) + education, data = Wage)  
plot.Gam(gam.lo, se = TRUE, col = "green")



gam.lo.i <- gam(wage ~ lo(year, age, span = 0.5) + education, data = Wage)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, :  
## liv too small. (Discovered by lowesd)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : lv  
## too small. (Discovered by lowesd)

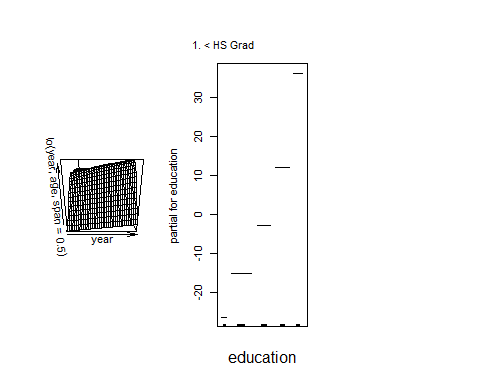
## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, :  
## liv too small. (Discovered by lowesd)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : lv  
## too small. (Discovered by lowesd)

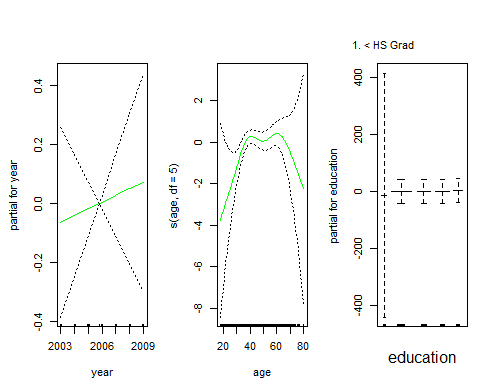
library(akima)

## Warning: package 'akima' was built under R version 4.3.2

plot(gam.lo.i)  
gam.lr <- gam(I(wage > 250) ~ year + s(age, df = 5) + education, family = binomial, data = Wage)  
par(mfrow = c(1, 3))



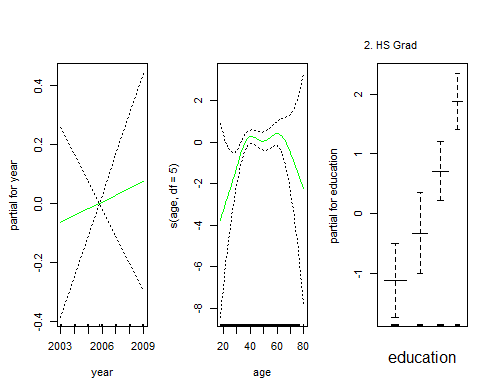
plot(gam.lr, se = T, col = "green")



table(education, I(wage > 250))

##   
## education FALSE TRUE  
## 1. < HS Grad 268 0  
## 2. HS Grad 966 5  
## 3. Some College 643 7  
## 4. College Grad 663 22  
## 5. Advanced Degree 381 45

gam.lr.s <- gam(I(wage > 250) ~ year + s(age, df = 5) + education, family = binomial, data = Wage, subset = (education != "1. < HS Grad"))  
plot(gam.lr.s, se = T, col = "green")



## 8.3 Lab: Decision Trees

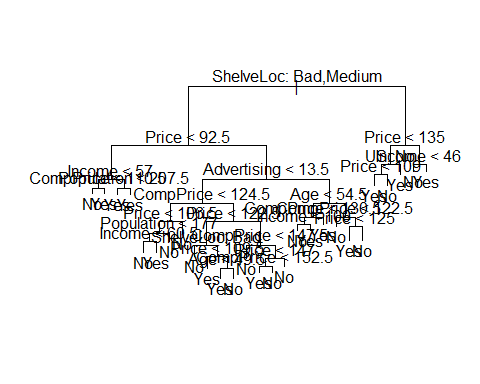
library(tree)

## Warning: package 'tree' was built under R version 4.3.2

library(ISLR2)  
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 ) \*

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)

## High.test  
## tree.pred No Yes  
## No 104 33  
## Yes 13 50

(104+50)/200

## [1] 0.77

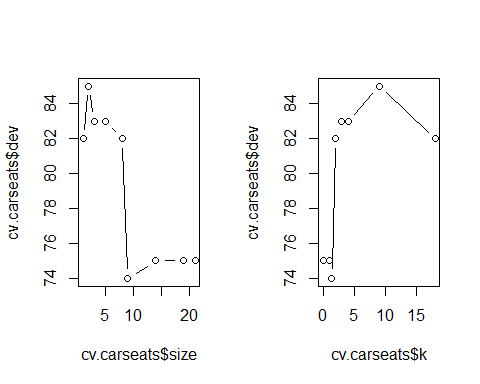
set.seed(7)  
cv.carseats <- cv.tree(tree.carseats,FUN=prune.misclass)   
names(cv.carseats)

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

cv.carseats

## $size  
## [1] 21 19 14 9 8 5 3 2 1  
##   
## $dev  
## [1] 75 75 75 74 82 83 83 85 82  
##   
## $k  
## [1] -Inf 0.0 1.0 1.4 2.0 3.0 4.0 9.0 18.0  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

par(mfrow=c(1,2))  
plot(cv.carseats$size,cv.carseats$dev,type="b")  
plot(cv.carseats$k,cv.carseats$dev,type="b")



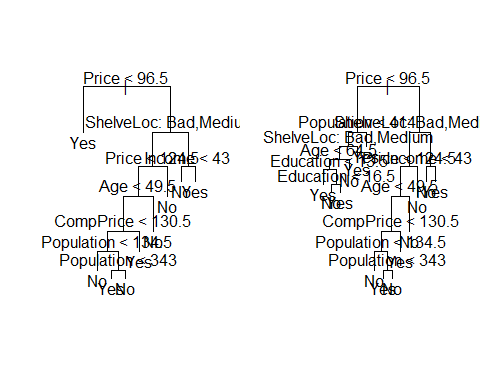
prune.carseats <- prune.misclass(tree.carseats,best=9)   
plot(prune.carseats)  
text(prune.carseats,pretty=0)   
  
tree.pred <- predict(prune.carseats,Carseats.test,type="class")  
table(tree.pred,High.test)

## High.test  
## tree.pred No Yes  
## No 97 25  
## Yes 20 58

(97 + 58) / 200

## [1] 0.775

prune.carseats <- prune.misclass(tree.carseats,best=14)   
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 102 31  
## Yes 15 52

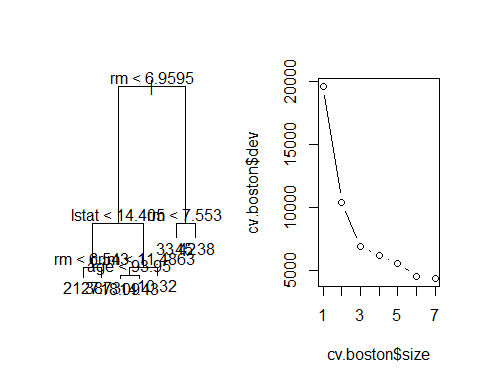
(102 + 52) / 200

## [1] 0.77

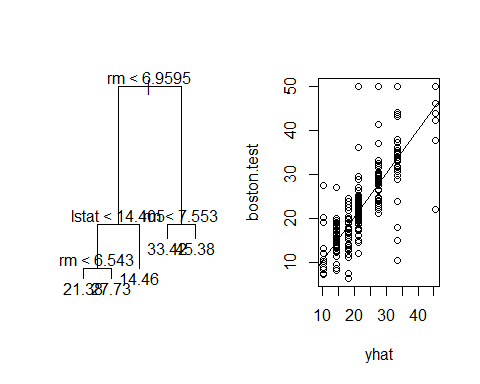
set.seed(1)  
train <- sample(1:nrow(Boston),nrow(Boston)/2)   
tree.boston <- tree(medv~.,Boston,subset=train)   
summary(tree.boston)

##   
## Regression tree:  
## tree(formula = medv ~ ., data = Boston, subset = train)  
## Variables actually used in tree construction:  
## [1] "rm" "lstat" "crim" "age"   
## Number of terminal nodes: 7   
## Residual mean deviance: 10.38 = 2555 / 246   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -10.1800 -1.7770 -0.1775 0.0000 1.9230 16.5800

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)

## [1] 35.28688

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.3.2

## randomForest 4.7-1.1

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

set.seed(1)  
bag.boston <- randomForest(medv~.,data=Boston,subset=train,mtry=12,importance=TRUE)   
bag.boston

##   
## Call:  
## randomForest(formula = medv ~ ., data = Boston, mtry = 12, importance = TRUE, subset = train)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 12  
##   
## Mean of squared residuals: 11.40162  
## % Var explained: 85.17

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

## [1] 23.41916

bag.boston <- randomForest(medv~.,data=Boston,subset=train,mtry=12,ntree=25)   
yhat.bag <- predict(bag.boston,newdata=Boston[-train,])   
mean((yhat.bag-boston.test)^2)

## [1] 25.75055

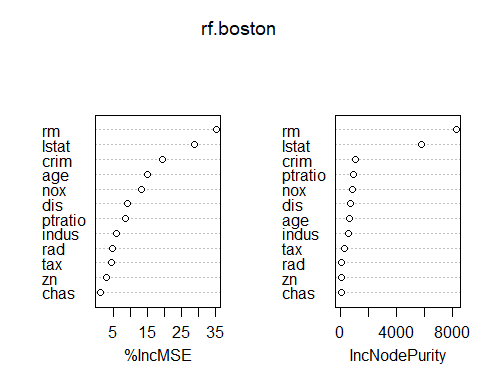
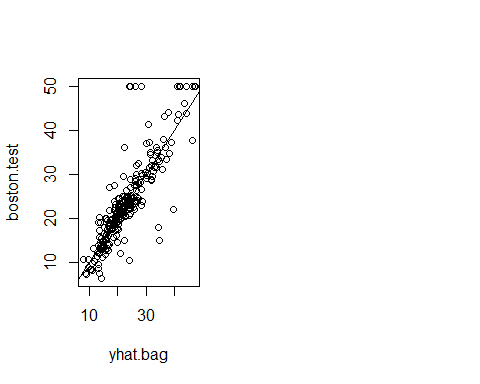
set.seed(1)  
rf.boston <- randomForest(medv~.,data=Boston,subset=train,mtry=6,importance=TRUE)   
yhat.rf <- predict(rf.boston,newdata=Boston[-train,])   
mean((yhat.rf-boston.test)^2)

## [1] 20.06644

importance(rf.boston)

## %IncMSE IncNodePurity  
## crim 19.435587 1070.42307  
## zn 3.091630 82.19257  
## indus 6.140529 590.09536  
## chas 1.370310 36.70356  
## nox 13.263466 859.97091  
## rm 35.094741 8270.33906  
## age 15.144821 634.31220  
## dis 9.163776 684.87953  
## rad 4.793720 83.18719  
## tax 4.410714 292.20949  
## ptratio 8.612780 902.20190  
## lstat 28.725343 5813.04833

varImpPlot(rf.boston)



library(gbm)

## Warning: package 'gbm' was built under R version 4.3.2

## Loaded gbm 2.1.8.1

set.seed(1)  
boost.boston <- gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4)   
  
summary(boost.boston)

## var rel.inf  
## rm rm 44.48249588  
## lstat lstat 32.70281223  
## crim crim 4.85109954  
## dis dis 4.48693083  
## nox nox 3.75222394  
## age age 3.19769210  
## ptratio ptratio 2.81354826  
## tax tax 1.54417603  
## indus indus 1.03384666  
## rad rad 0.87625748  
## zn zn 0.16220479  
## chas chas 0.09671228

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)

## [1] 18.39057

boost.boston <- gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4,shrinkage=0.2,verbose=F)   
yhat.boost <- predict(boost.boston,newdata=Boston[-train,],n.trees=5000)  
mean((yhat.boost-boston.test)^2)

## [1] 16.54778

library(BART)

## Warning: package 'BART' was built under R version 4.3.2

## Loading required package: nlme

## Loading required package: nnet

## Loading required package: survival

x <- Boston[,1:12]   
y <- Boston[, "medv"]   
xtrain <- x[train,]   
ytrain <- y[train]   
xtest <- x[-train,]   
ytest <- y[-train]   
set.seed(1)  
bartfit <- gbart(xtrain,ytrain,x.test=xtest)

## \*\*\*\*\*Calling gbart: type=1  
## \*\*\*\*\*Data:  
## data:n,p,np: 253, 12, 253  
## y1,yn: 0.213439, -5.486561  
## x1,x[n\*p]: 0.109590, 20.080000  
## xp1,xp[np\*p]: 0.027310, 7.880000  
## \*\*\*\*\*Number of Trees: 200  
## \*\*\*\*\*Number of Cut Points: 100 ... 100  
## \*\*\*\*\*burn,nd,thin: 100,1000,1  
## \*\*\*\*\*Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.795495,3,3.71636,21.7866  
## \*\*\*\*\*sigma: 4.367914  
## \*\*\*\*\*w (weights): 1.000000 ... 1.000000  
## \*\*\*\*\*Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,12,0  
## \*\*\*\*\*printevery: 100  
##   
## MCMC  
## done 0 (out of 1100)  
## done 100 (out of 1100)  
## done 200 (out of 1100)  
## done 300 (out of 1100)  
## done 400 (out of 1100)  
## done 500 (out of 1100)  
## done 600 (out of 1100)  
## done 700 (out of 1100)  
## done 800 (out of 1100)  
## done 900 (out of 1100)  
## done 1000 (out of 1100)  
## time: 2s  
## trcnt,tecnt: 1000,1000

yhat.bart <- bartfit$yhat.test.mean   
mean((ytest-yhat.bart)^2)

## [1] 15.94718

ord <- order(bartfit$varcount.mean,decreasing=T)  
bartfit$varcount.mean[ord]

## nox lstat tax rad rm indus chas ptratio age zn   
## 22.952 21.329 21.250 20.781 19.890 19.825 19.051 18.976 18.274 15.952   
## dis crim   
## 14.457 11.007

