Illustrating Regression Tree Using Boston Housing Data

Tahani Omer

April 16, 2019

## Packages required

We will Load required packages,and have a look at the “Boston” data from “MASS” packages.

library(MASS)  
library(ISLR)  
library(tree)  
library(psych)  
  
names(Boston)

## [1] "crim" "zn" "indus" "chas" "nox" "rm" "age"   
## [8] "dis" "rad" "tax" "ptratio" "black" "lstat" "medv"

str(Boston)

## 'data.frame': 506 obs. of 14 variables:  
## $ crim : num 0.00632 0.02731 0.02729 0.03237 0.06905 ...  
## $ zn : num 18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...  
## $ indus : num 2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...  
## $ chas : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ nox : num 0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...  
## $ rm : num 6.58 6.42 7.18 7 7.15 ...  
## $ age : num 65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...  
## $ dis : num 4.09 4.97 4.97 6.06 6.06 ...  
## $ rad : int 1 2 2 3 3 3 5 5 5 5 ...  
## $ tax : num 296 242 242 222 222 222 311 311 311 311 ...  
## $ ptratio: num 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...  
## $ black : num 397 397 393 395 397 ...  
## $ lstat : num 4.98 9.14 4.03 2.94 5.33 ...  
## $ medv : num 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

## Exploratory data analysis

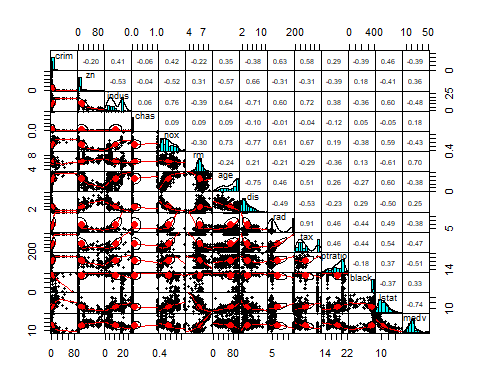
#summary statistics  
attach(Boston)  
summary(Boston)

## crim zn indus chas   
## Min. : 0.00632 Min. : 0.00 Min. : 0.46 Min. :0.00000   
## 1st Qu.: 0.08204 1st Qu.: 0.00 1st Qu.: 5.19 1st Qu.:0.00000   
## Median : 0.25651 Median : 0.00 Median : 9.69 Median :0.00000   
## Mean : 3.61352 Mean : 11.36 Mean :11.14 Mean :0.06917   
## 3rd Qu.: 3.67708 3rd Qu.: 12.50 3rd Qu.:18.10 3rd Qu.:0.00000   
## Max. :88.97620 Max. :100.00 Max. :27.74 Max. :1.00000   
## nox rm age dis   
## Min. :0.3850 Min. :3.561 Min. : 2.90 Min. : 1.130   
## 1st Qu.:0.4490 1st Qu.:5.886 1st Qu.: 45.02 1st Qu.: 2.100   
## Median :0.5380 Median :6.208 Median : 77.50 Median : 3.207   
## Mean :0.5547 Mean :6.285 Mean : 68.57 Mean : 3.795   
## 3rd Qu.:0.6240 3rd Qu.:6.623 3rd Qu.: 94.08 3rd Qu.: 5.188   
## Max. :0.8710 Max. :8.780 Max. :100.00 Max. :12.127   
## rad tax ptratio black   
## Min. : 1.000 Min. :187.0 Min. :12.60 Min. : 0.32   
## 1st Qu.: 4.000 1st Qu.:279.0 1st Qu.:17.40 1st Qu.:375.38   
## Median : 5.000 Median :330.0 Median :19.05 Median :391.44   
## Mean : 9.549 Mean :408.2 Mean :18.46 Mean :356.67   
## 3rd Qu.:24.000 3rd Qu.:666.0 3rd Qu.:20.20 3rd Qu.:396.23   
## Max. :24.000 Max. :711.0 Max. :22.00 Max. :396.90   
## lstat medv   
## Min. : 1.73 Min. : 5.00   
## 1st Qu.: 6.95 1st Qu.:17.02   
## Median :11.36 Median :21.20   
## Mean :12.65 Mean :22.53   
## 3rd Qu.:16.95 3rd Qu.:25.00   
## Max. :37.97 Max. :50.00

#Check for missing values  
sum(is.na(Boston))

## [1] 0

#checking correlation between variables  
  
pairs.panels(Boston, gap=0)



Comments: out of all the variable,

medv and lstat seem to have high negative correlation.In other words we find that as the lower status of the population percentage increase the median value of owner-occupied homes decrease.

And crim is strongly associated with variables rad and tax which implies as accessibility to radial highways increases, per capita crime rate increases.

Also, indus has strong positive correlation with nox, which supports the notion that nitrogen oxides concentration is high in industrial areas.

## Fitting Regression Trees

Now we fit a tree to these data, and summarize and plot it, and annotate with the command text.

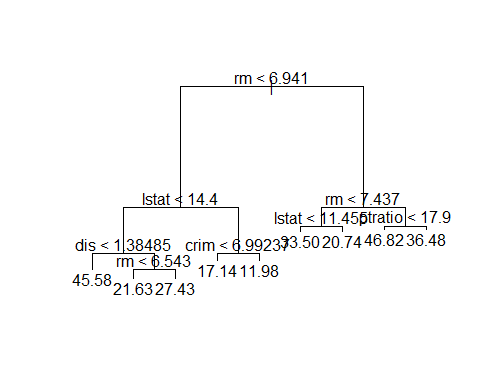
tree.boston=tree(medv~.,Boston)  
summary(tree.boston)

##   
## Regression tree:  
## tree(formula = medv ~ ., data = Boston)  
## Variables actually used in tree construction:  
## [1] "rm" "lstat" "dis" "crim" "ptratio"  
## Number of terminal nodes: 9   
## Residual mean deviance: 13.55 = 6734 / 497   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -17.68000 -2.23000 0.07026 0.00000 2.22100 16.50000

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

## Warning in text.default(xy$x[ind], xy$y[ind] + 0.5 \* charht, rows[ind], :  
## "plot.new" is not a graphical parameter

## Warning in text.default(xy$x[leaves], xy$y[leaves] - 0.5 \* charht, labels =  
## stat, : "plot.new" is not a graphical parameter



comments: the output of summary() indicates that only five of the variables have been used in constructing the tree. In the context of a regression tree, the deviance is simply the sum of squared errors for the tree.So, we got a binary big tree with all terminal nodes labeled by the splitting variables and splitting point.

Now we print the tree for a detalled summary

tree.boston

## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 506 42720.0 22.53   
## 2) rm < 6.941 430 17320.0 19.93   
## 4) lstat < 14.4 255 6632.0 23.35   
## 8) dis < 1.38485 5 390.7 45.58 \*  
## 9) dis > 1.38485 250 3721.0 22.91   
## 18) rm < 6.543 195 1636.0 21.63 \*  
## 19) rm > 6.543 55 643.2 27.43 \*  
## 5) lstat > 14.4 175 3373.0 14.96   
## 10) crim < 6.99237 101 1151.0 17.14 \*  
## 11) crim > 6.99237 74 1086.0 11.98 \*  
## 3) rm > 6.941 76 6059.0 37.24   
## 6) rm < 7.437 46 1900.0 32.11   
## 12) lstat < 11.455 41 844.2 33.50 \*  
## 13) lstat > 11.455 5 329.8 20.74 \*  
## 7) rm > 7.437 30 1099.0 45.10   
## 14) ptratio < 17.9 25 340.7 46.82 \*  
## 15) ptratio > 17.9 5 312.7 36.48 \*

Comments: we got a print out with details of every single terminal node.Such as how many observation, The mean deviance and Overall perdiction value

## Split data set into 50:50 train and test data

Now we creat a training and test set (253,253) split of 506 observations, grow the tree on the training srt , and evaluate its perfomance on the test set.

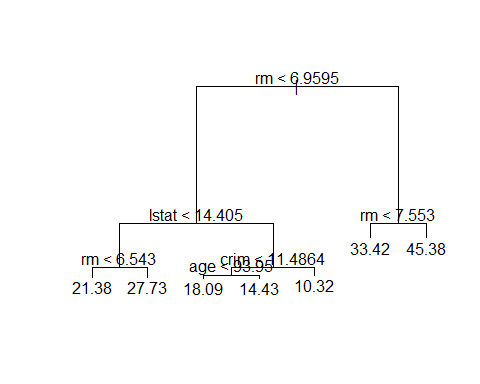
set.seed(1)  
train = sample(1:nrow(Boston), nrow(Boston)/2)  
tree.boston1=tree(medv~.,Boston,subset=train)  
summary(tree.boston1)

##   
## 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.boston1)  
text(tree.boston1,plot.new = plot(tree.boston1), pretty=0)

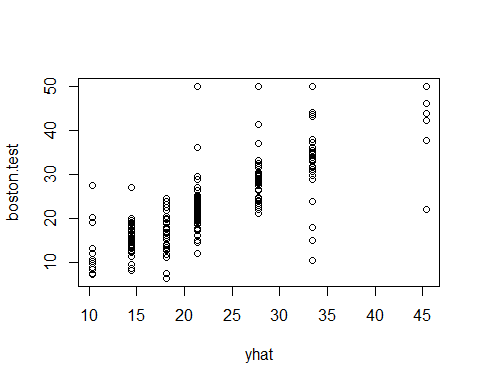
## Warning in text.default(xy$x[ind], xy$y[ind] + 0.5 \* charht, rows[ind], :  
## "plot.new" is not a graphical parameter

## Warning in text.default(xy$x[leaves], xy$y[leaves] - 0.5 \* charht, labels =  
## stat, : "plot.new" is not a graphical parameter



We will see how it performs in the test dataset.

yhat=predict(tree.boston1,newdata=Boston[-train,])  
boston.test=Boston[-train,"medv"]  
plot(yhat,boston.test)



mean((yhat-boston.test)^2)

## [1] 35.28688

Comments: the plot seem to look a little different, but the complexity of the tree looks roughly the same. And we get the mean square error of 25.05 indicating that this model leads to test predictions that are within around $5.005 of true median home value for the suburb

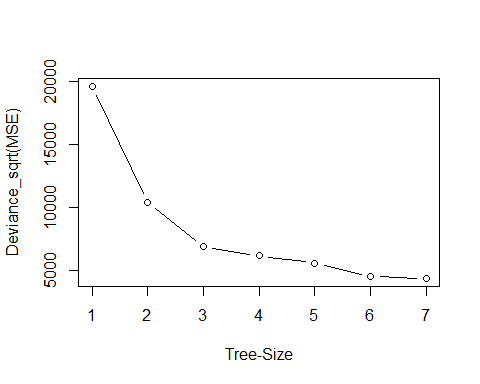
## Pruning

Now we use CV to prune the big tree, which might have too many variables, and print out the result.

cv.boston=cv.tree(tree.boston1)  
cv.boston

## $size  
## [1] 7 6 5 4 3 2 1  
##   
## $dev  
## [1] 4380.849 4544.815 5601.055 6171.917 6919.608 10419.472 19630.870  
##   
## $k  
## [1] -Inf 203.9641 637.2707 796.1207 1106.4931 3424.7810  
## [7] 10724.5951  
##   
## $method  
## [1] "deviance"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

plot(cv.boston$size,cv.boston$dev,type='b',xlab = "Tree-Size",ylab = "Deviance\_sqrt(MSE)")



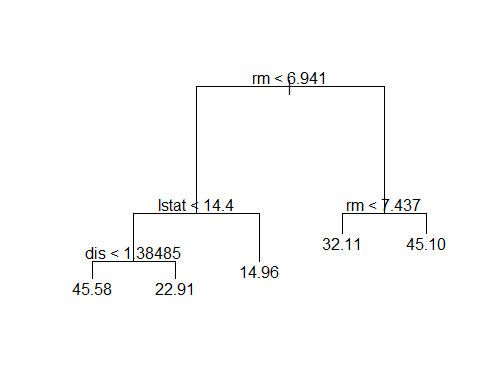
Comments: we got the size of the trees, and the deviance as the pruning proceeded. alse the cost complexity parameter from the plot we can pick the minimum value down near the minimum

Now we repeat the commands we had before

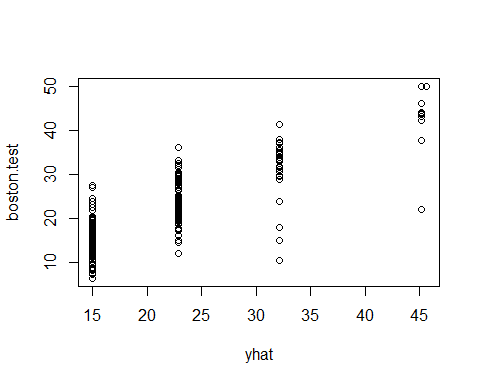
prune.boston=prune.tree(tree.boston ,best=5)  
plot(prune.boston)  
text(prune.boston,plot.new=plot(prune.boston), pretty =0)

## Warning in text.default(xy$x[ind], xy$y[ind] + 0.5 \* charht, rows[ind], :  
## "plot.new" is not a graphical parameter

## Warning in text.default(xy$x[leaves], xy$y[leaves] - 0.5 \* charht, labels =  
## stat, : "plot.new" is not a graphical parameter



yhat=predict(prune.boston,newdata=Boston[-train,])  
boston.test=Boston[-train,"medv"]  
plot(yhat,boston.test)



mean((boston.test-yhat)^2)

## [1] 23.38954

Comments: We did not get too much from pruning, except we get a small tree which is easier to interpret

## Fitting Linear-Regression

Following linear model were fitted to the training data. \* Using lm function resulted in model with 11 variables are significant and 2 variables are insignificant based on p-value. Remove the insignificant variable, based on the best subset regression and Cross-validation, resulted in a model with adjusted R-squared 0.7097 and test prediction error of 15.7751.

Now we will Split data set into 80:20 train and test data And fit the linear regression model

set.seed(2)  
index <- sample(nrow(Boston), nrow(Boston) \* 0.80)  
Boston.train <- Boston[index, ]  
Boston.test <- Boston[-index, ]  
model1 = lm(medv ~ ., data = Boston.train)  
model1.sum = summary(model1)  
model1.sum

##   
## Call:  
## lm(formula = medv ~ ., data = Boston.train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -14.2584 -2.8095 -0.5954 2.0201 25.6482   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 40.727321 5.794321 7.029 9.36e-12 \*\*\*  
## crim -0.105196 0.036574 -2.876 0.00425 \*\*   
## zn 0.049606 0.015061 3.294 0.00108 \*\*   
## indus -0.006587 0.069267 -0.095 0.92429   
## chas 2.287160 0.973989 2.348 0.01936 \*   
## nox -18.211779 4.311364 -4.224 2.99e-05 \*\*\*  
## rm 3.457179 0.468521 7.379 9.70e-13 \*\*\*  
## age 0.001418 0.014637 0.097 0.92290   
## dis -1.612346 0.226054 -7.133 4.82e-12 \*\*\*  
## rad 0.311951 0.073742 4.230 2.91e-05 \*\*\*  
## tax -0.012519 0.004115 -3.042 0.00251 \*\*   
## ptratio -1.015388 0.150133 -6.763 4.95e-11 \*\*\*  
## black 0.009722 0.003054 3.183 0.00157 \*\*   
## lstat -0.525672 0.058725 -8.951 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.785 on 390 degrees of freedom  
## Multiple R-squared: 0.7377, Adjusted R-squared: 0.7289   
## F-statistic: 84.37 on 13 and 390 DF, p-value: < 2.2e-16

Comments: variables indus and age are insignificant (based on p-value)

Now we will build a model without variables indus and age

model2 = lm(medv ~ . -indus -age, data = Boston.train)  
model2.sum = summary(model2)  
model2.sum

##   
## Call:  
## lm(formula = medv ~ . - indus - age, data = Boston.train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -14.2834 -2.7949 -0.5826 2.0226 25.6966   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 40.715622 5.738308 7.095 6.08e-12 \*\*\*  
## crim -0.105070 0.036431 -2.884 0.004143 \*\*   
## zn 0.049556 0.014915 3.323 0.000976 \*\*\*  
## chas 2.278615 0.963903 2.364 0.018568 \*   
## nox -18.211751 3.984044 -4.571 6.51e-06 \*\*\*  
## rm 3.472055 0.453655 7.654 1.54e-13 \*\*\*  
## dis -1.614129 0.209878 -7.691 1.19e-13 \*\*\*  
## rad 0.313444 0.070866 4.423 1.26e-05 \*\*\*  
## tax -0.012678 0.003709 -3.418 0.000696 \*\*\*  
## ptratio -1.016953 0.146691 -6.933 1.71e-11 \*\*\*  
## black 0.009750 0.003037 3.211 0.001433 \*\*   
## lstat -0.524260 0.054731 -9.579 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.773 on 392 degrees of freedom  
## Multiple R-squared: 0.7377, Adjusted R-squared: 0.7303   
## F-statistic: 100.2 on 11 and 392 DF, p-value: < 2.2e-16

Comments: All variables are significant (based on p-value).

## Variable Selection

Now we will do the Best subset and cross-validation techniques, to come up with the best linear regression model for the dependent variable medv.

## Best subset selection

library(leaps)  
  
model.subset <- regsubsets(medv ~ ., data = Boston.train, nvmax = 13)  
(reg.sum =summary(model.subset))

## Subset selection object  
## Call: regsubsets.formula(medv ~ ., data = Boston.train, nvmax = 13)  
## 13 Variables (and intercept)  
## Forced in Forced out  
## crim FALSE FALSE  
## zn FALSE FALSE  
## indus FALSE FALSE  
## chas FALSE FALSE  
## nox FALSE FALSE  
## rm FALSE FALSE  
## age FALSE FALSE  
## dis FALSE FALSE  
## rad FALSE FALSE  
## tax FALSE FALSE  
## ptratio FALSE FALSE  
## black FALSE FALSE  
## lstat FALSE FALSE  
## 1 subsets of each size up to 13  
## Selection Algorithm: exhaustive  
## crim zn indus chas nox rm age dis rad tax ptratio black lstat  
## 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 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

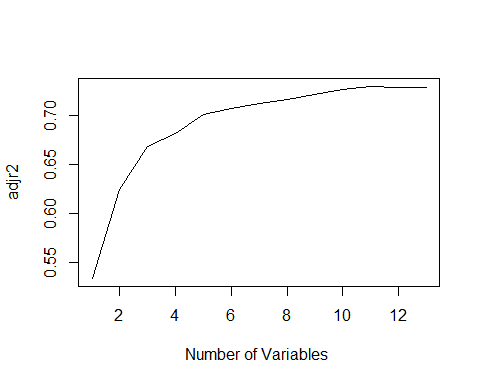
names(reg.sum)

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

reg.sum$rsq

## [1] 0.5344553 0.6268839 0.6713073 0.6850002 0.7055037 0.7121755 0.7176239  
## [8] 0.7223758 0.7280063 0.7339366 0.7376763 0.7376829 0.7376890

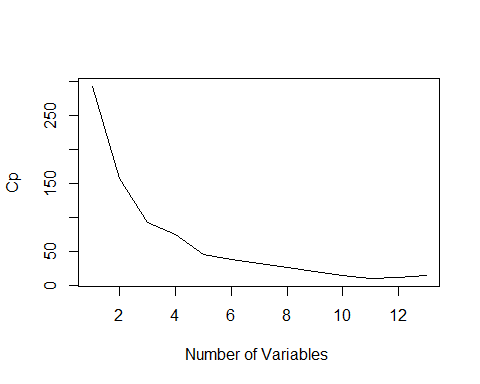
plot(reg.sum$adjr2,xlab="Number of Variables ",ylab="adjr2",  
 type="l")



which.max(reg.sum$adjr2)

## [1] 11

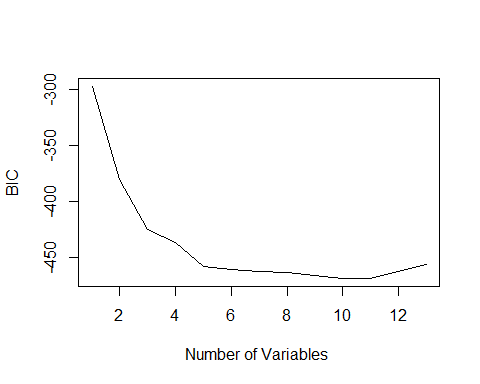
plot(reg.sum$cp ,xlab="Number of Variables ",ylab="Cp",  
 type="l")



which.min(reg.sum$cp )

## [1] 11

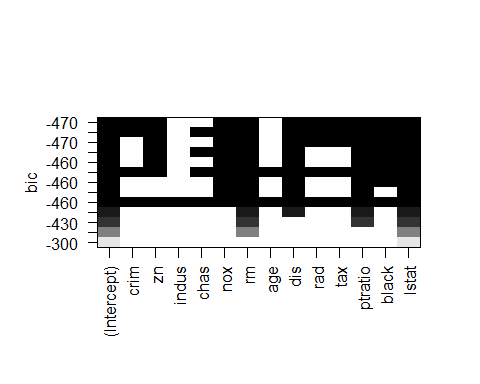
plot(reg.sum$bic ,xlab="Number of Variables ",ylab="BIC",  
 type="l")



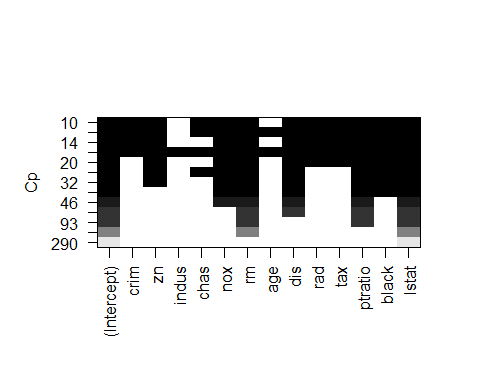
which.min(reg.sum$bic )

## [1] 10

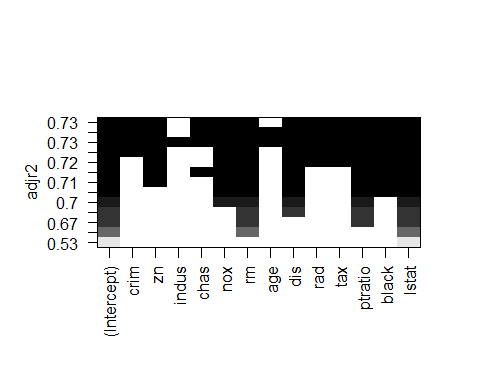
plot(model.subset, scale = "bic")



plot(model.subset, scale = "Cp")



plot(model.subset, scale = "adjr2")

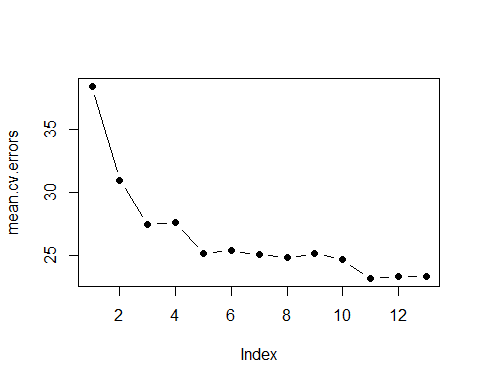


## Cross-validation

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  
}  
  
k=10  
set.seed(1)  
folds=sample (1:k,nrow(Boston),replace=TRUE)  
cv.errors =matrix (NA,k,13, dimnames =list(NULL , paste (1:13) ))  
for(j in 1:k){  
 model.subset=regsubsets (medv~.,data=Boston [folds!=j,],  
 nvmax=13)  
 for(i in 1:13){  
 pred=predict (model.subset ,Boston [folds ==j,],id=i)  
 cv.errors[j,i]= mean( ( Boston$medv[ folds==j]-pred)^2)  
 }  
}  
  
mean.cv.errors=apply(cv.errors ,2, mean)  
mean.cv.errors

## 1 2 3 4 5 6 7 8   
## 38.39005 30.93288 27.48079 27.60953 25.18985 25.40145 25.09708 24.83743   
## 9 10 11 12 13   
## 25.19292 24.68941 23.22625 23.35491 23.39588

plot(mean.cv.errors, pch=19 ,type="b")



which.min(mean.cv.errors)

## 11   
## 11

reg.best=regsubsets (medv~.,data= Boston , nvmax=13)  
coef(reg.best,11)

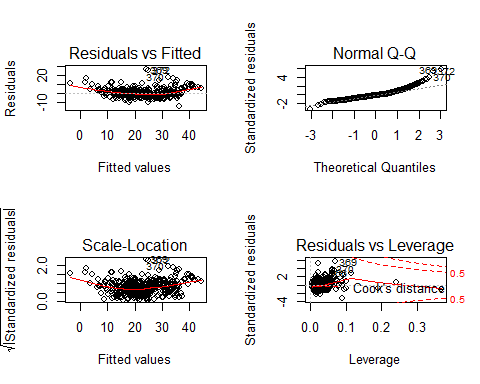
## (Intercept) crim zn chas nox   
## 36.341145004 -0.108413345 0.045844929 2.718716303 -17.376023429   
## rm dis rad tax ptratio   
## 3.801578840 -1.492711460 0.299608454 -0.011777973 -0.946524570   
## black lstat   
## 0.009290845 -0.522553457

Commonet: From Best subset regression and Cross-validation , we see that all variables except indus and age are significant

## Model Assessment

Model Diagnostics for model 2

par(mfrow = c(2,2))  
plot(model2)



par(mfrow = c(1,1))

Commonets: Residuals vs Fitted plot shows that the relationship between medv and predictors is not completely linear. Also, normal qq plot is skewed implying that residuals are not normally distributed. A different functional form may be required.

## Model selection

#R-squared  
model1.sum$r.squared

## [1] 0.737689

model2.sum$r.squared

## [1] 0.7376763

#AIC  
AIC(model1)

## [1] 2427.186

AIC(model2)

## [1] 2423.206

#BIC  
BIC(model1)

## [1] 2487.207

BIC(model2)

## [1] 2475.224

#Test error (MSSE)  
model1.pred.test <- predict(model1, newdata = Boston.test)  
model1.msse = mean((model1.pred.test - Boston.test$medv) ^ 2)  
model1.msse

## [1] 21.48905

model2.pred.test <- predict(model2, newdata = Boston.test)  
model2.mspe <- mean((model2.pred.test - Boston.test$medv) ^ 2)  
model2.mspe

## [1] 21.4667

Comments: Based on AIC,Bic criteria and R square values, model 2 is slightly better than model 1, also based on prediction error, model 2 is slightly better than model 1. MSSE of model 1 is 15.80331 while that of model 2 is 15.7751 we choose model2 for the best model for linear-regression

## K-Nearest Neighbors Model

Now As a comparison we fit a K-NN to Boston data, using Library Class with K = 1, 2, and 10

library(class)  
medv01 = rep(0, length(medv))  
medv01[medv > median(medv)] = 1  
Boston = data.frame(Boston, medv01)  
train = 1:(dim(Boston)[1]/2)  
test = (dim(Boston)[1]/2 + 1):dim(Boston)[1]  
Boston.train = Boston[train, ]  
Boston.test = Boston[test, ]  
medv01.test = medv01[test]  
train.X = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black,   
 lstat, crim)[train, ]  
test.X = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black,   
 lstat, crim)[test, ]  
train.medv01 = medv01[train]  
set.seed(1)  
knn.pred1 = knn(train.X, test.X, train.medv01, k = 1)  
mean(knn.pred1 != medv01.test)

## [1] 0.2687747

knn.pred2 = knn(train.X, test.X, train.medv01, k = 2)  
mean(knn.pred2 != medv01.test)

## [1] 0.2648221

knn.pred3 = knn(train.X, test.X, train.medv01, k = 10)  
mean(knn.pred3 != medv01.test)

## [1] 0.2055336

Comments: we got (1) k=1, %26.9 test error rate (2) k=2, %26.5 test error rate (3) k=10,% 20,6 test error rate

Now we fit K-NN when k=10 with subset of variables

train.X = cbind(zn, chas, nox, rm, dis, rad, tax, ptratio, black,   
 lstat, crim)[train, ]  
test.X = cbind(zn, chas, nox, rm, dis, rad, tax, ptratio, black,   
 lstat, crim)[test, ]  
knn.pred4 = knn(train.X, test.X, train.medv01, k = 10)  
mean(knn.pred4 != medv01.test)

## [1] 0.1936759

Comments: we got %19.4 test error rate, we choose when K =10 with subset of variables And still we do not get too much inprovement compare to tree model