DA7

2025-06-26

8.3.1 Fitting Classification Trees

```
library(tree)
library(ISLR)
rm(list=ls())

attach(Carseats)
summary(Carseats$Sales)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   5.390   7.490   7.496   9.320  16.270
```

the man and median is 7.49 and 7.496, so in order to make categorical value from continuous, I will check whether it is higher or lower than 8

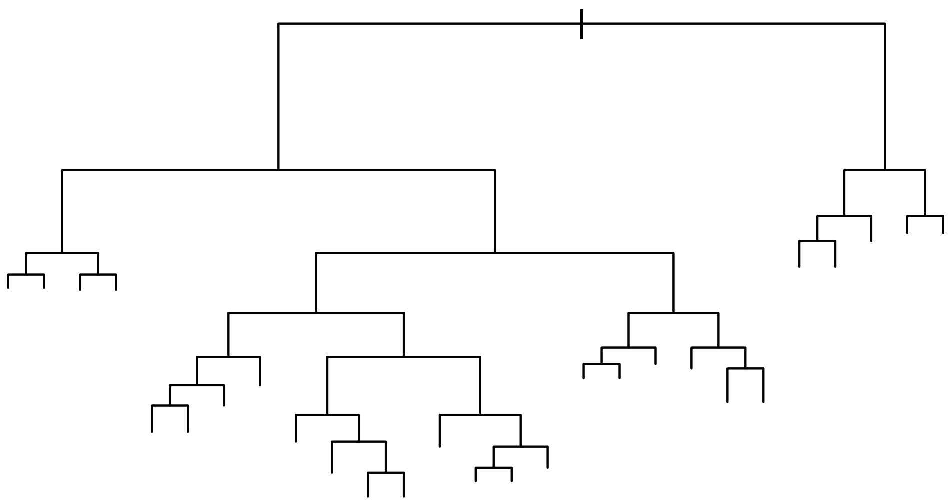
```
high = ifelse(Sales >= 8, "Yes", "No")
high = factor(high)
Carseats = data.frame(Carseats, high)
```

then I made high a factor variable and merged the carseats data set

```
carseats.tree <- tree(
  formula = high ~ . - Sales,
  data    = Carseats
)
summary(carseats.tree)
```

```
##
## 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(carseats.tree)
```



```r
set.seed(2)
lines=sample(1: nrow(Carseats), 200)
train = Carseats[lines, ]
test = Carseats [-lines, ]
high.test = high[-lines]
tree.carseats =tree(high ~ .-Sales, train)
tree.pred = predict(tree.carseats, test, type="class")
table(tree.pred, high.test)
```

```
##          high.test
## tree.pred  No Yes
##       No  104  33
##       Yes  13  50
```

```r
mean(tree.pred == high.test)
```

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

77% of accuracy

```r
set.seed(3)
cv.carseats =cv.tree(tree.carseats, FUN=prune.misclass)
names(cv.carseats)
```

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

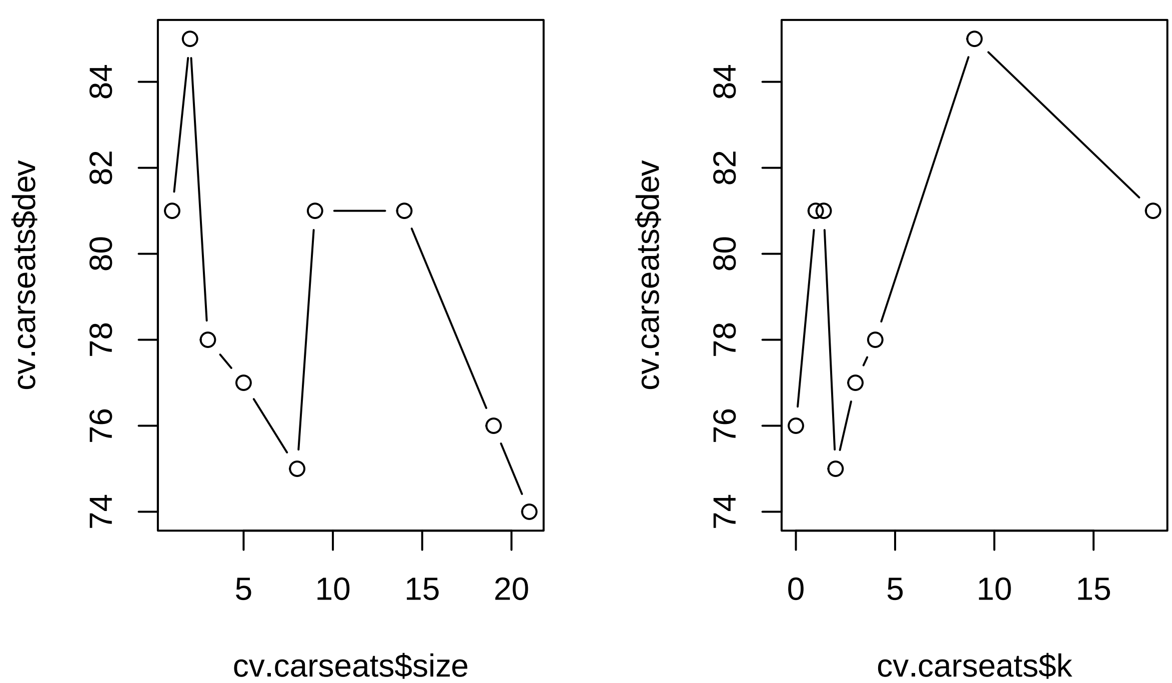
```
cv.carseats
```

```
## $size
## [1] 21 19 14  9  8  5  3  2  1
##
## $dev
## [1] 74 76 81 81 75 77 78 85 81
##
## $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"
```

using this method we can understand which complexity is the best to use. Here we can see that the minimum dev is 74 when the size is 21.

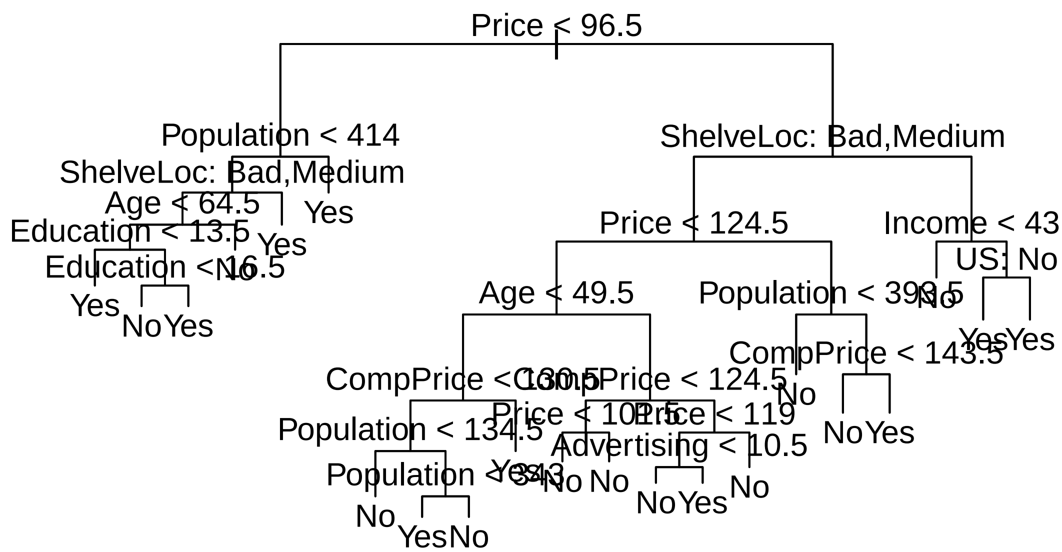
```
par(mfrow=c(1,2))

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



on these graphs we can see that 21 results the minimum dev;

```r
prune.carseats = prune.misclass (tree.carseats, best=21)
plot(prune.carseats)
text(prune.carseats, pretty =0)
```




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

```
##          high.test
## tree.pred  No Yes
##       No  104  32
##       Yes  13  51
```

```r
mean(tree.pred == high.test)
```

```
## [1] 0.775
```

77.5% accuracy

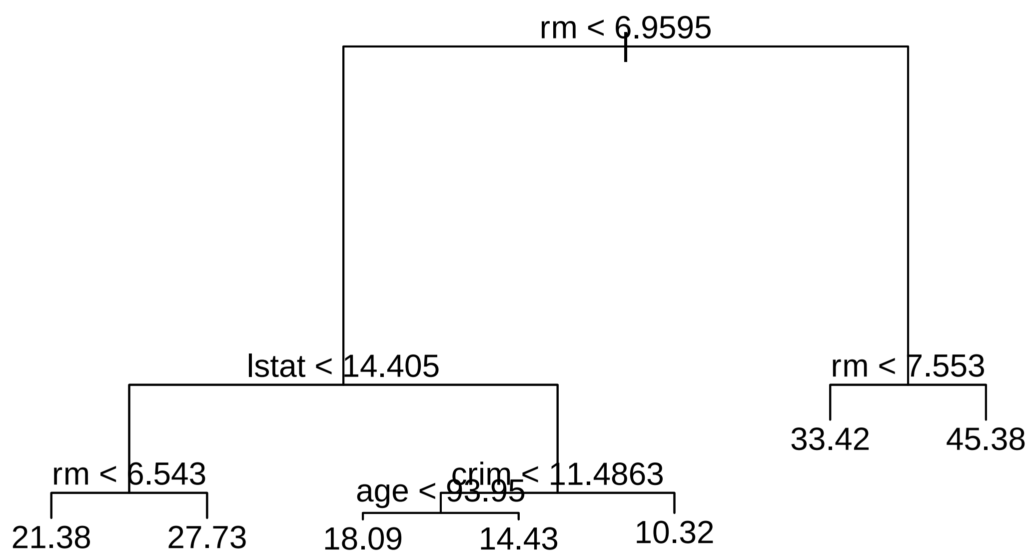
if i try increasing or decreasing it, it will be lower. 77.5% is the best accuracy I get.

8.3.2 Fitting Regression Trees

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

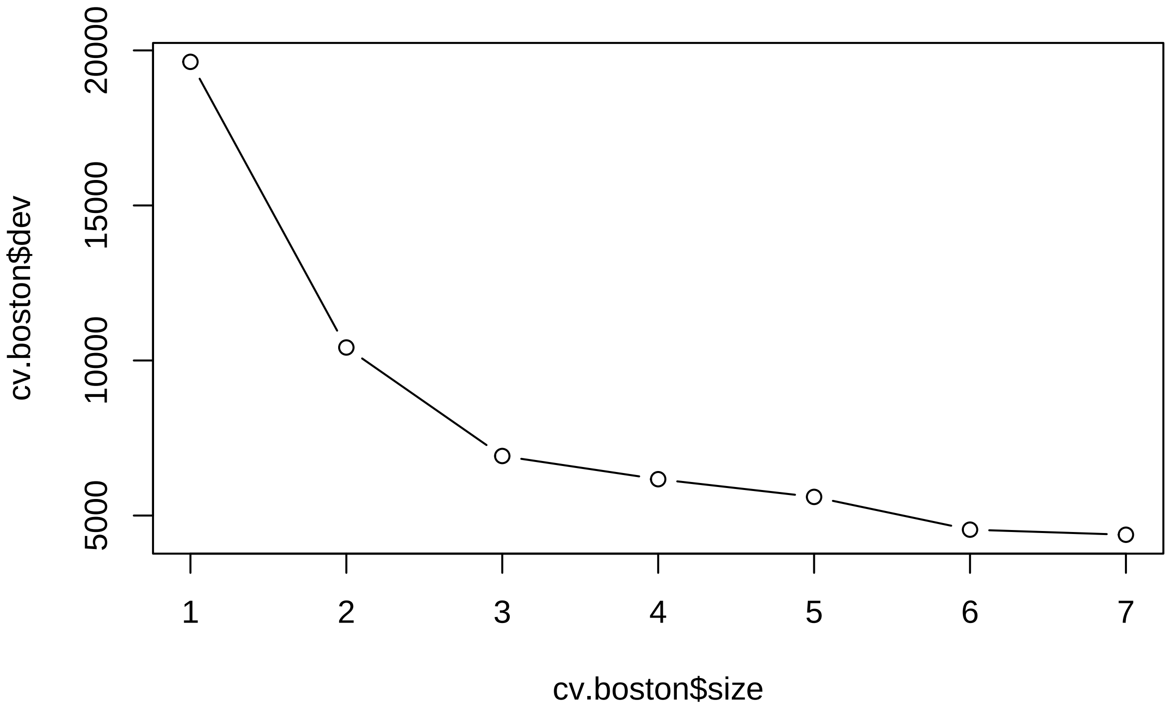
```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.38 = 2555 / 246
## Distribution of residuals:
##     Min.  1st Qu.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

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



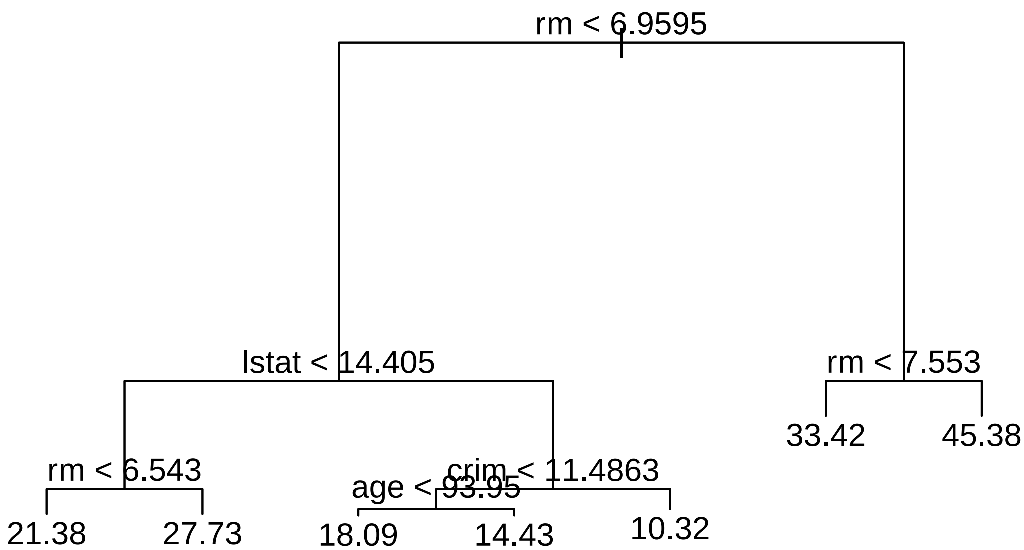
lower values of lstat correspond to more expensive houses.

```r
cv.boston=cv.tree(tree.boston)
plot(cv.boston$size, cv.boston$dev, type='b')
```

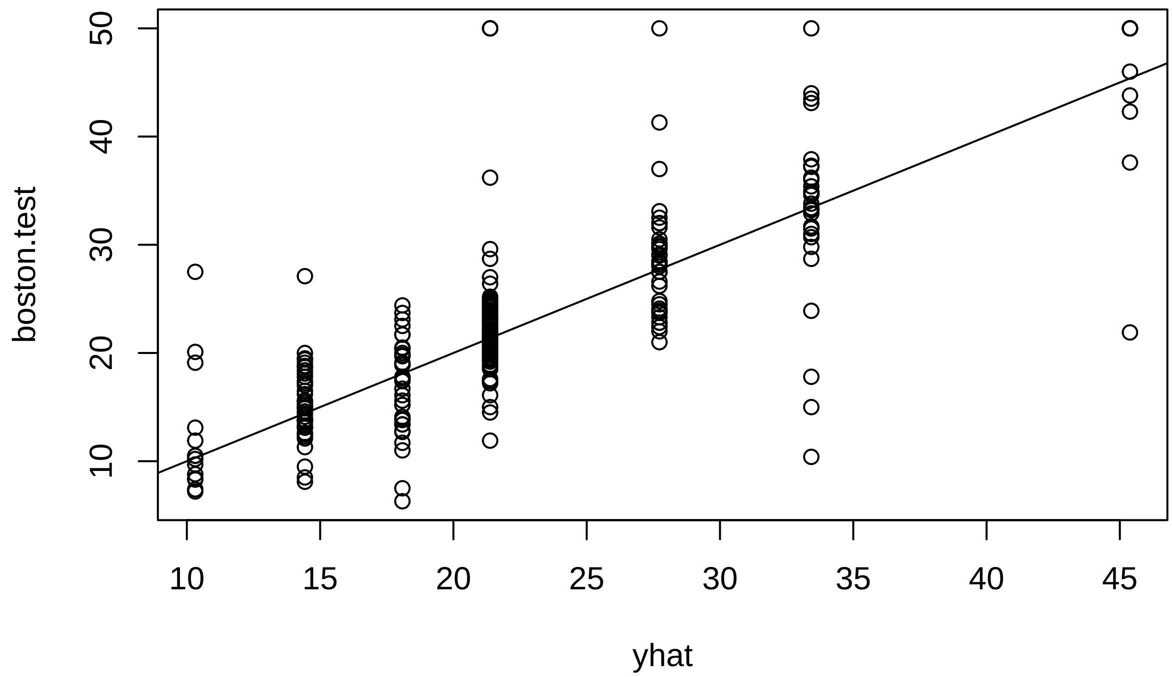


lower dev is with size of 7

```
prune.boston=prune.tree(tree.boston ,best=7)
plot(prune.boston)
text(prune.boston , pretty = 0)
```




```r
yhat=predict(tree.boston, newdata=Boston[- train, ])
boston.test = Boston[-train ,"medv"]
plot(yhat, boston.test)
abline(0, 1)
```




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

```
## [1] 35.28688
```

MSE associated with the regression tree is 35.29. Tree explains about 59 % of the variance in sale price.

8.3.3 Bagging and Random Forests

```r
library( randomForest)
```

```
## randomForest 4.7-1.2
```

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

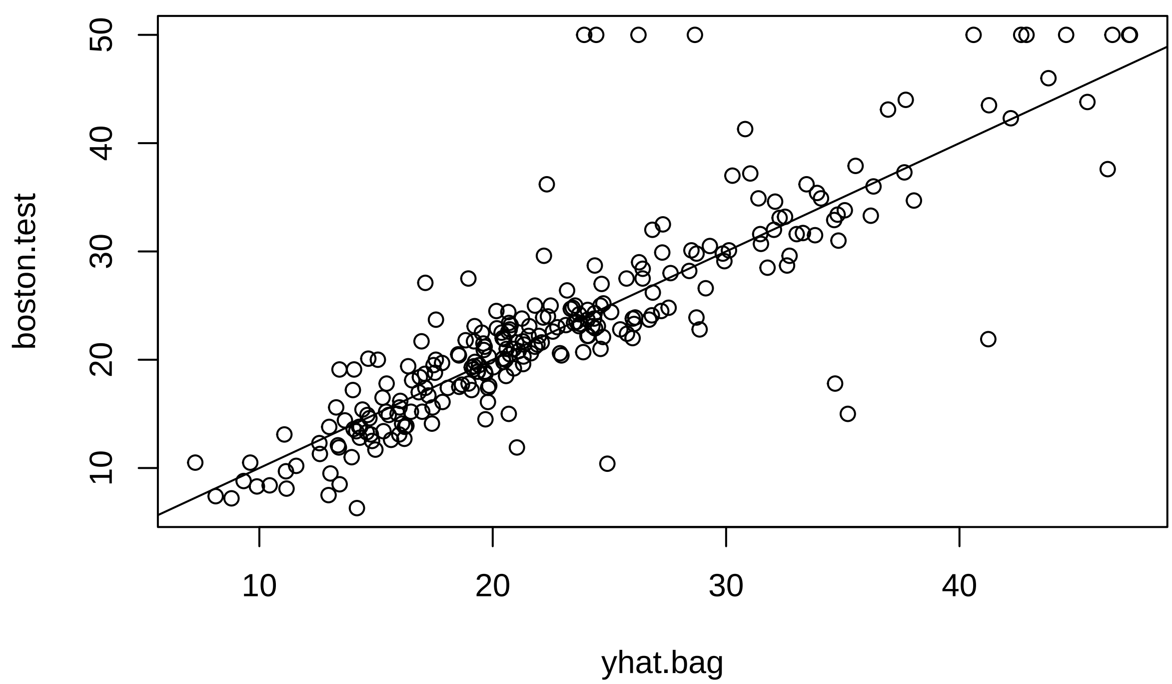
```r
set.seed(1)
bag.boston= randomForest( medv ~ ., data=Boston, subset=train,
  mtry = 13, importance =TRUE)
bag.boston
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, mtry = 13, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 11.39601
##                     % Var explained: 85.17
```

The argument mtry=13 indicates that all 13 predictors should be considered for each split of the tree

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



Most points lie reasonably close to the identity line, which tells that bagged model is doing a pretty good job predicting house prices

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

```
## [1] 23.59273
```

mse of 23.59 is a pretty good figure

```
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)
```

```
## [1] 23.66716
```

higher ntree figure makes the mse 23.45

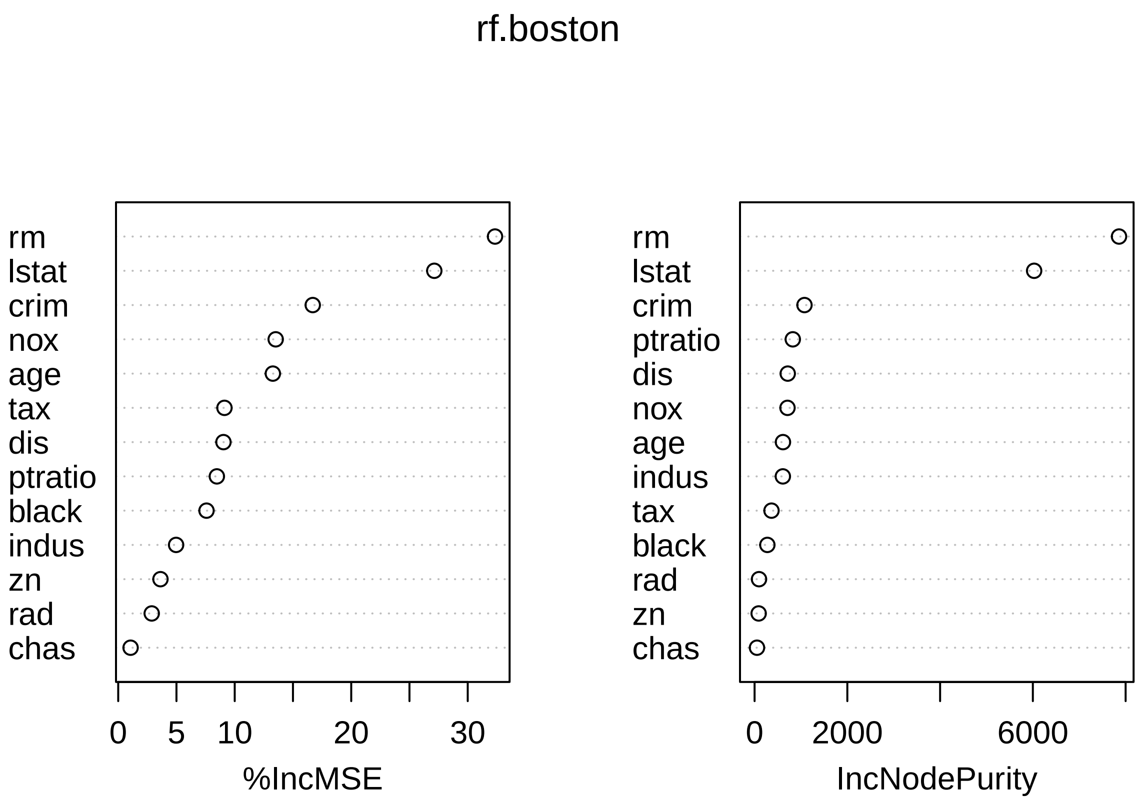
```r
set.seed(1)
rf.boston= randomForest(medv ~ ., data=Boston, subset=train,
  mtry=6, importance =TRUE)
yhat.rf = predict(rf.boston, newdata=Boston[- train ,])
mean((yhat.rf-boston.test)^2)
```

```
## [1] 19.62021
```

```r
importance (rf.boston)
```

```
##           %IncMSE IncNodePurity
## crim    16.697017    1076.08786
## zn       3.625784      88.35342
## indus    4.968621     609.53356
## chas     1.061432      52.21793
## nox     13.518179     709.87339
## rm      32.343305    7857.65451
## age     13.272498     612.21424
## dis      9.032477     714.94674
## rad      2.878434      95.80598
## tax      9.118801     364.92479
## ptratio  8.467062     823.93341
## black    7.579482     275.62272
## lstat   27.129817    6027.63740
```

```r
varImpPlot(rf.boston)
```



rm and lstat are the best predictors for tree