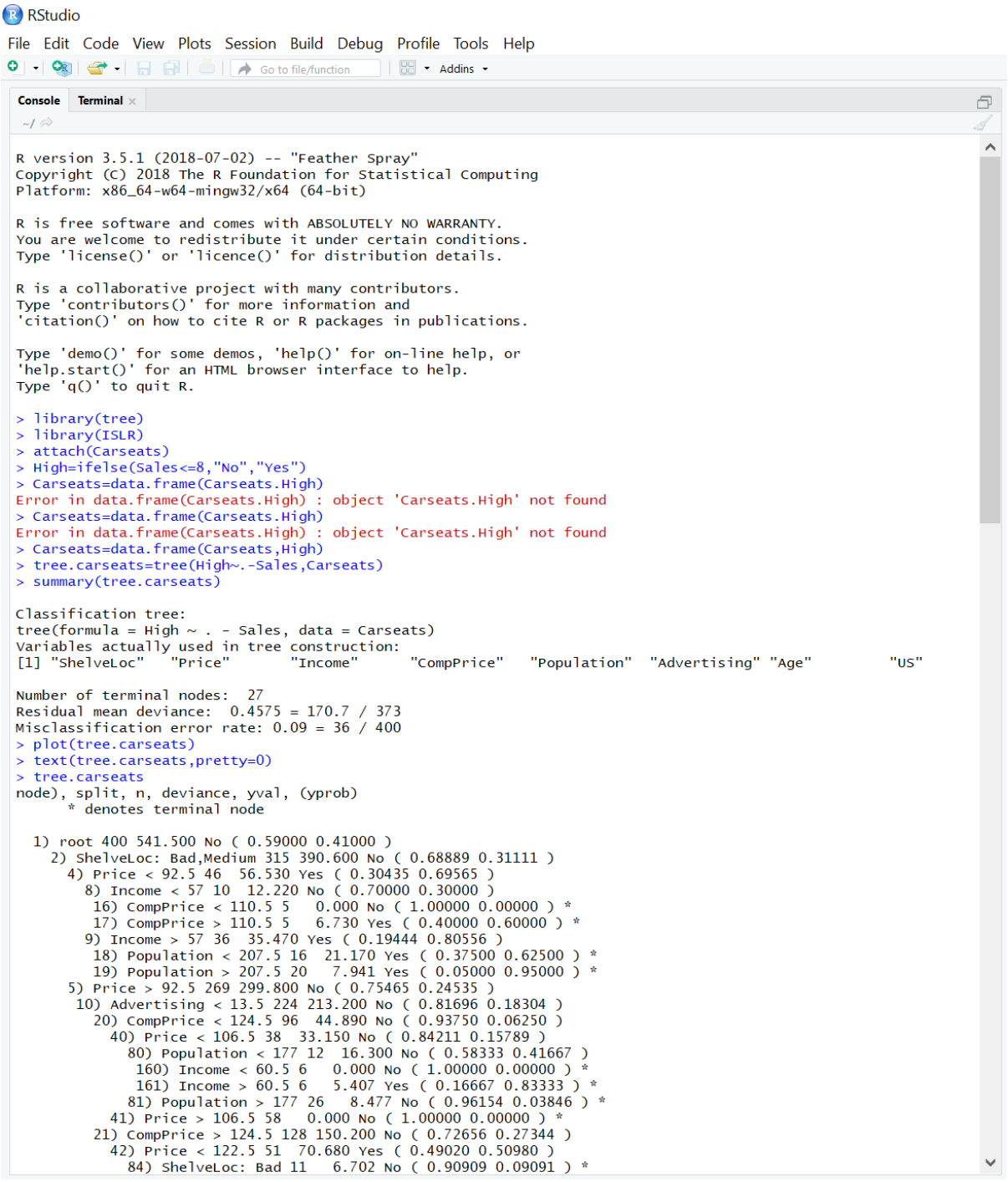
1) Fitting Classification Trees

R code:



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Console Terminal

```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(tree)
> library(ISLR)
> attach(Carseats)
> High=ifelse(Sales<=8,"No","Yes")
> Carseats=data.frame(Carseats.High)
Error in data.frame(Carseats.High) : object 'Carseats.High' not found
> Carseats=data.frame(Carseats.High)
Error in data.frame(Carseats.High) : object 'Carseats.High' not found
> 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"  "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 ) *
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

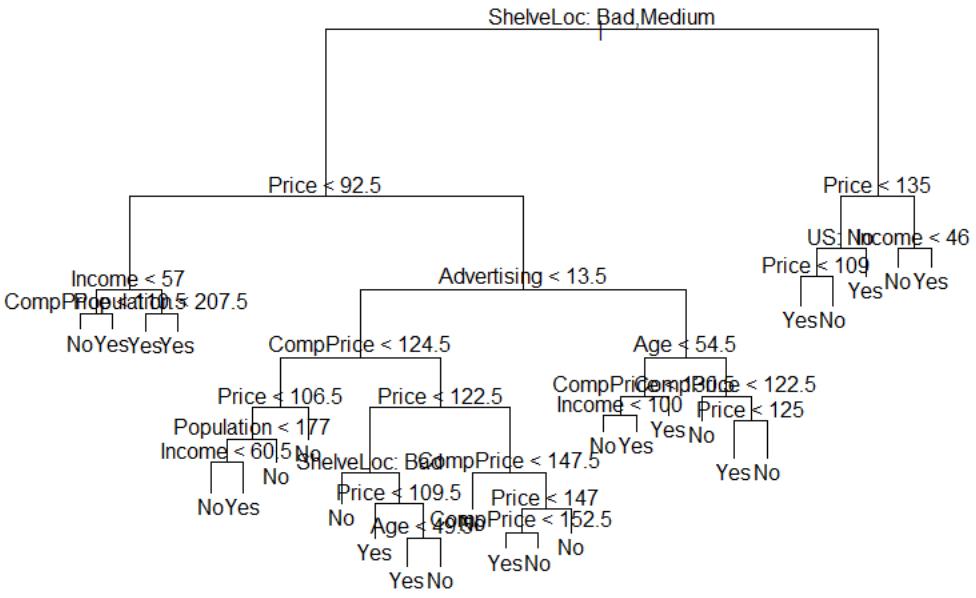
Go to file/function Addins

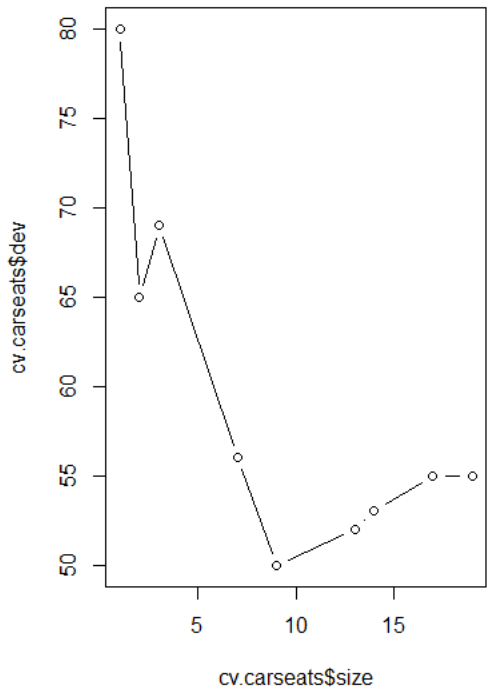
Console Terminal

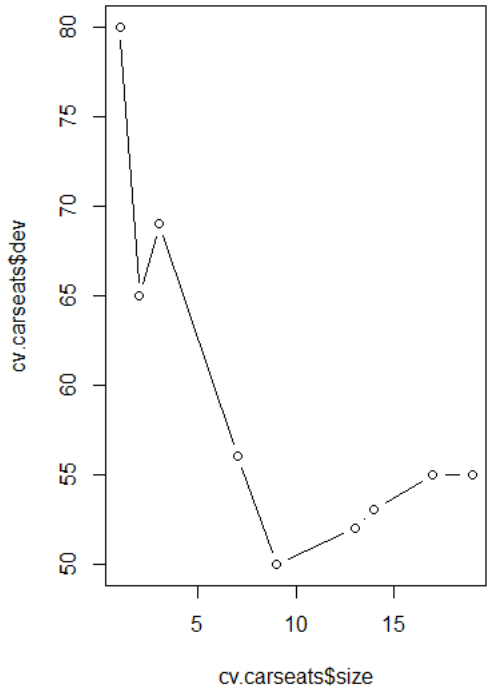
```
            84) ShelveLoc: Bad 11   6.702 No ( 0.90909 0.09091 ) *
            85) ShelveLoc: Medium 40  52.930 Yes ( 0.37500 0.62500 )
             170) Price < 109.5 16   7.481 Yes ( 0.06250 0.93750 ) *
             171) Price > 109.5 24  32.600 No ( 0.58333 0.41667 )
               342) Age < 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")
Error in predict.tree(tree.carseats, carseats.test, type = "class") :
  object 'carseats.test' not found
> tree.pred=predict(tree.carseats,Carseats.test,type="class")
> table(tree.pred,High.test)
         High.test
tree.pred No Yes
      No  86  27
      Yes 30  57
> set.seed(3)
> cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)
> names(cv.carseats)
[1] "size"   "dev"    "k"      "method"
> cv.carseats
$`size`
[1] 19 17 14 13  9  7  3  2  1

$dev
[1] 55 55 53 52 50 56 69 65 80

$k
[1]       -Inf  0.0000000  0.6666667  1.0000000  1.7500000  2.0000000  4.2500000  5.0000000 23.0000000

$method
[1] "misclass"
```

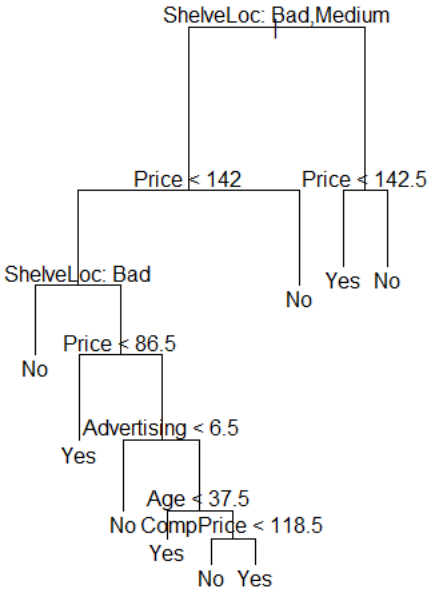
```
attr(,"class")
[1] "prune"         "tree.sequence"
> par(mfrow=c(1,2))
> plot(cv.carseats$size,cv.carseats$dev,type="b")
> prune.carseats=prune.misclass(tree.carseats,best=9)
> plot(prune.carseats)
> text(prune.carseats,pretty=0)
> tree.pred=predict(prune.carseats,Carseats.test,type="class")
> table(tree.pred,High.test)
         High.test
tree.pred No Yes
      No  94  24
      Yes 22  60
> 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)
         High.test
tree.pred No Yes
      No  86  22
      Yes 30  62
> prune.carseats=prune.misclass(tree.carseats,best=15)
```

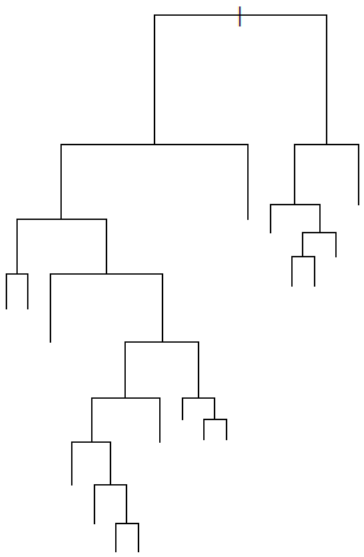
Plotted tree:

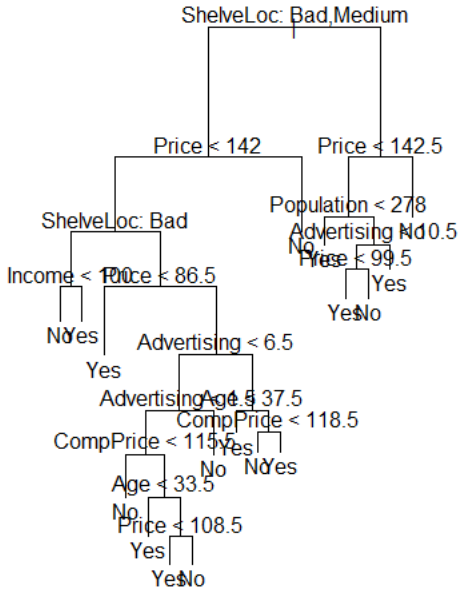



cv.carseats$dev
cv.carseats$size
50
55
60
65
70
75
80
5
10
15


cv.carseats$dev
cv.carseats$size
50
55
60
65
70
75
80
5
10
15


ShelveLoc: Bad,Medium
Price < 142
Price < 142.5
ShelveLoc: Bad
No
Yes
No
No
Price < 86.5
Advertising < 6.5
Yes
Age < 37.5
No
CompPride < 118.5
Yes
No
Yes



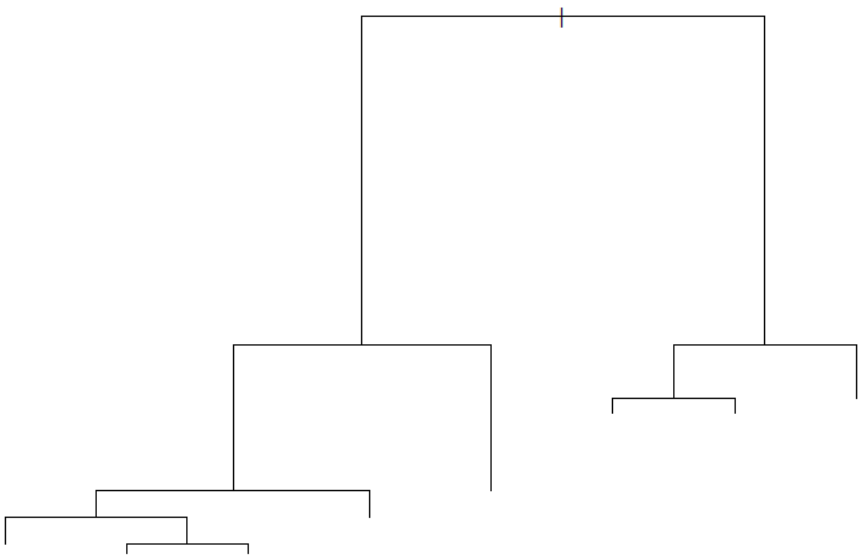

ShelveLoc: Bad,Medium
Price < 142
Price < 142.5
Population < 278
ShelveLoc: Bad
Price < 99.5
Yes
No
Yes
Price < 86.5
Advertising < 6.5
Yes
CompPrice < 118.5
CompPrice < 115.5
Yes
No
Yes
Age < 33.5
No
Price < 108.5
Yes
Yes
No

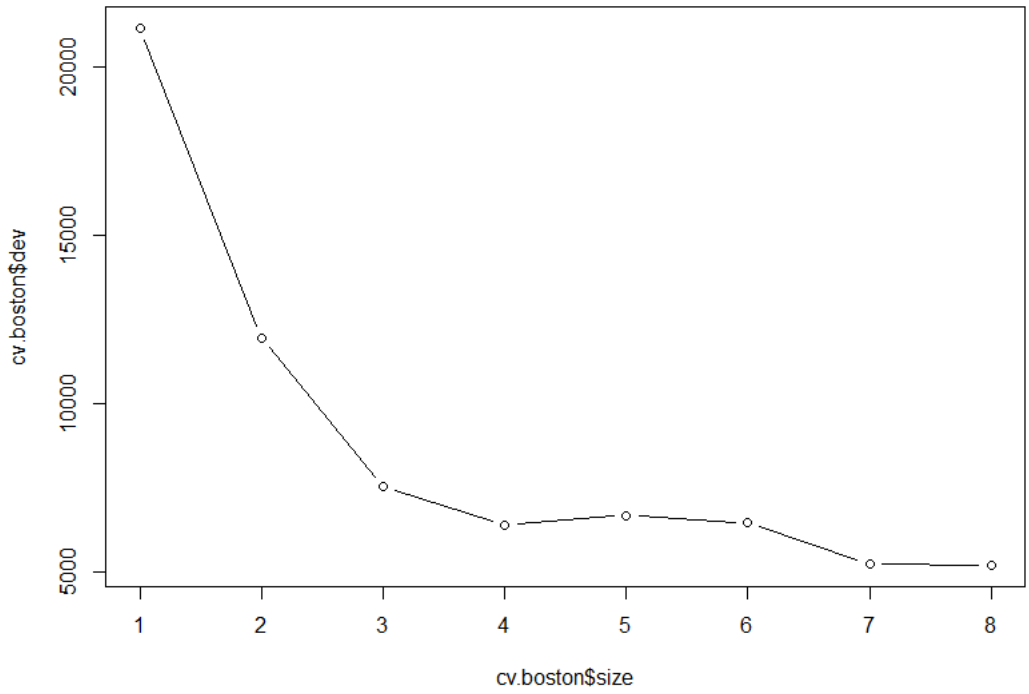
2) Fitting Regression Trees

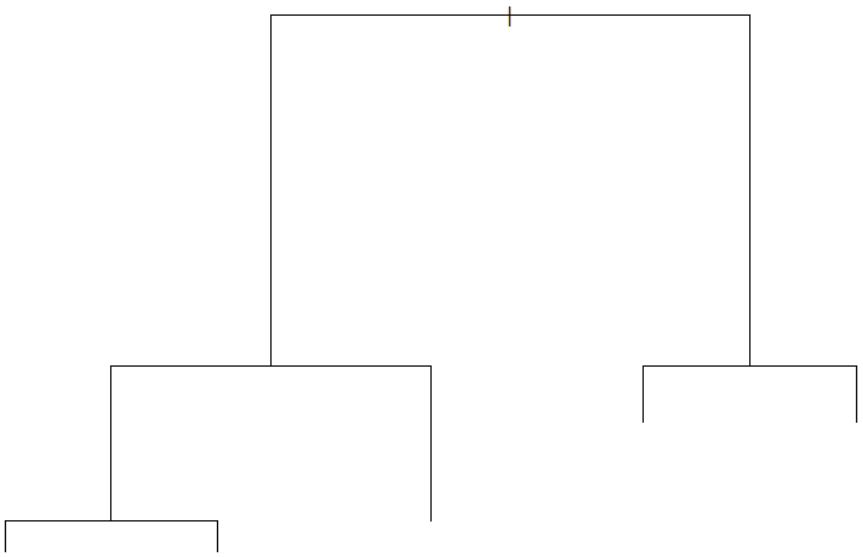
R Code:

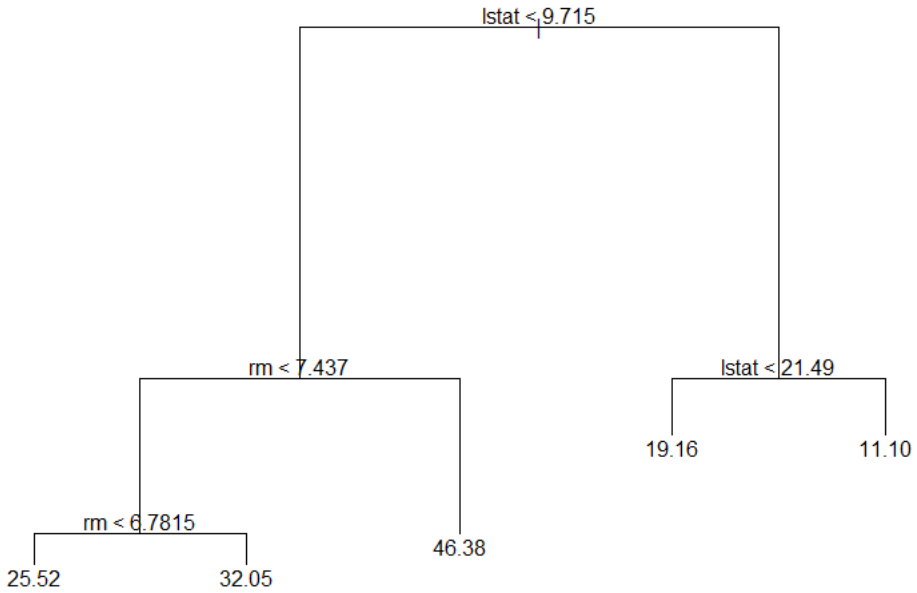
```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> library(tree)
> 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] "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)
> 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] 25.04559
>
```

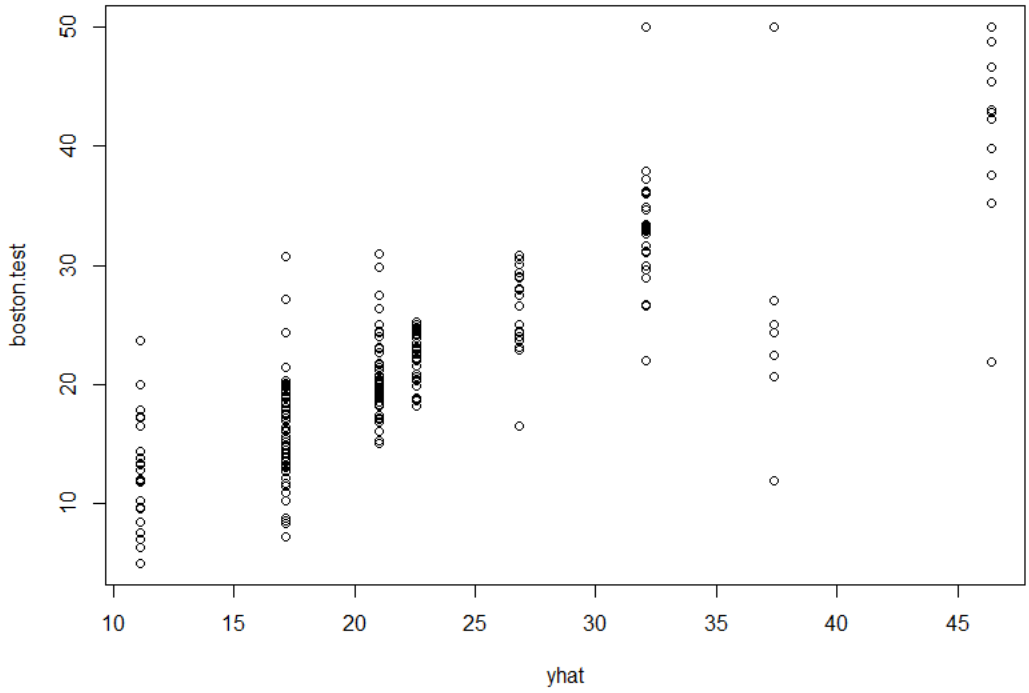
Plotted graphs:

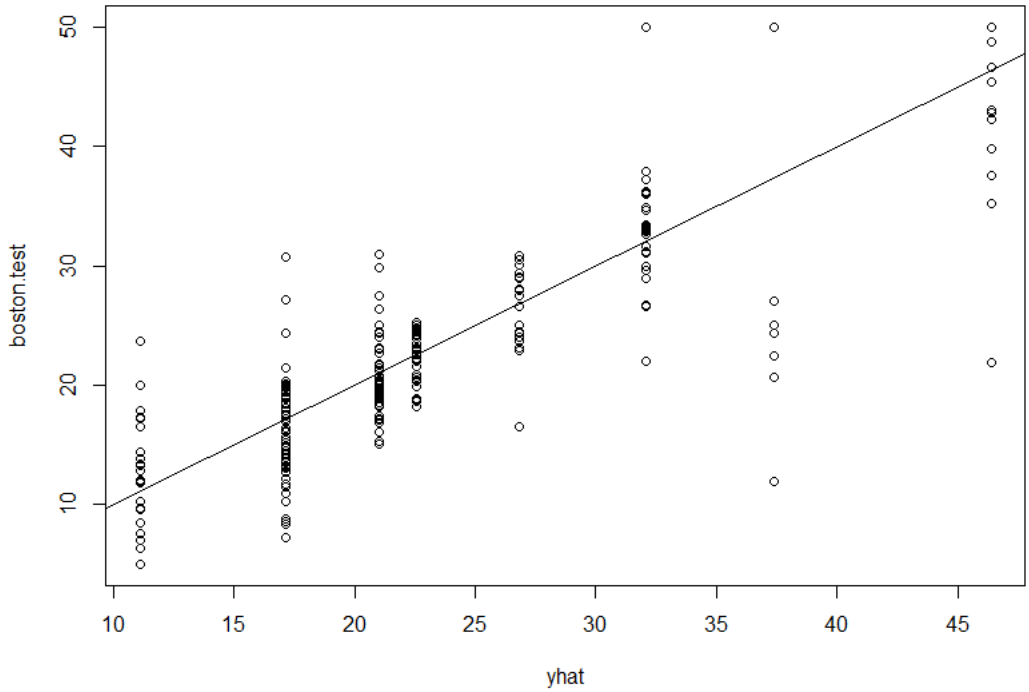







lstat < 9.715
rm < 7.437
lstat < 21.49
19.16
11.10
rm < 6.7815
46.38
25.52
32.05


boston.test
yhat
10
15
20
25
30
35
40
45
50


boston.test
yhat
10
15
20
25
30
35
40
45
50

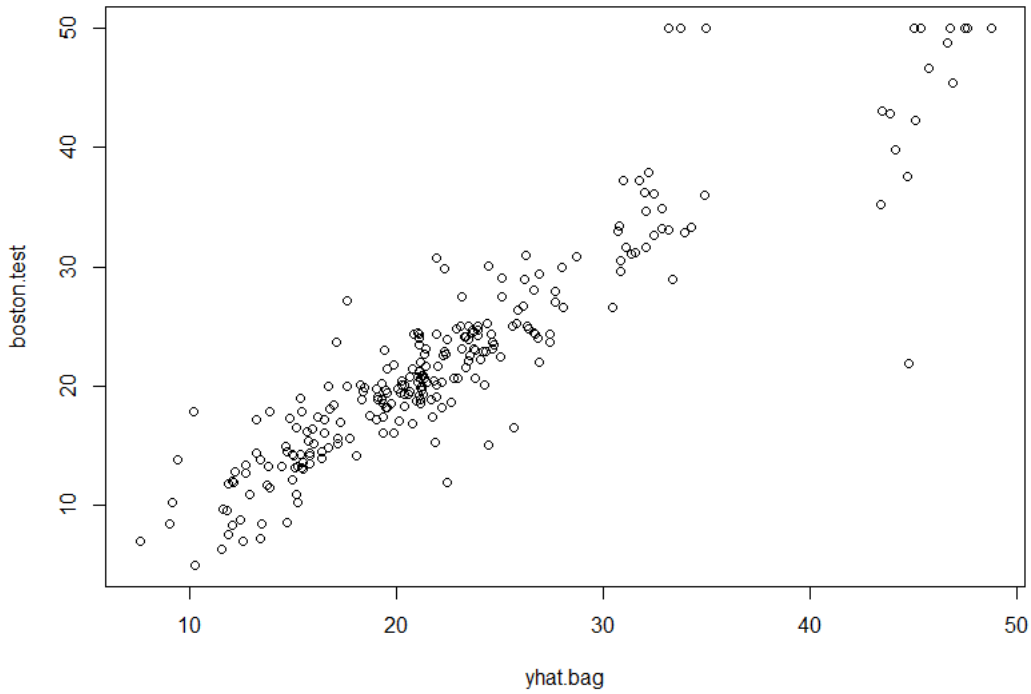
3) Bagging and Random Forest

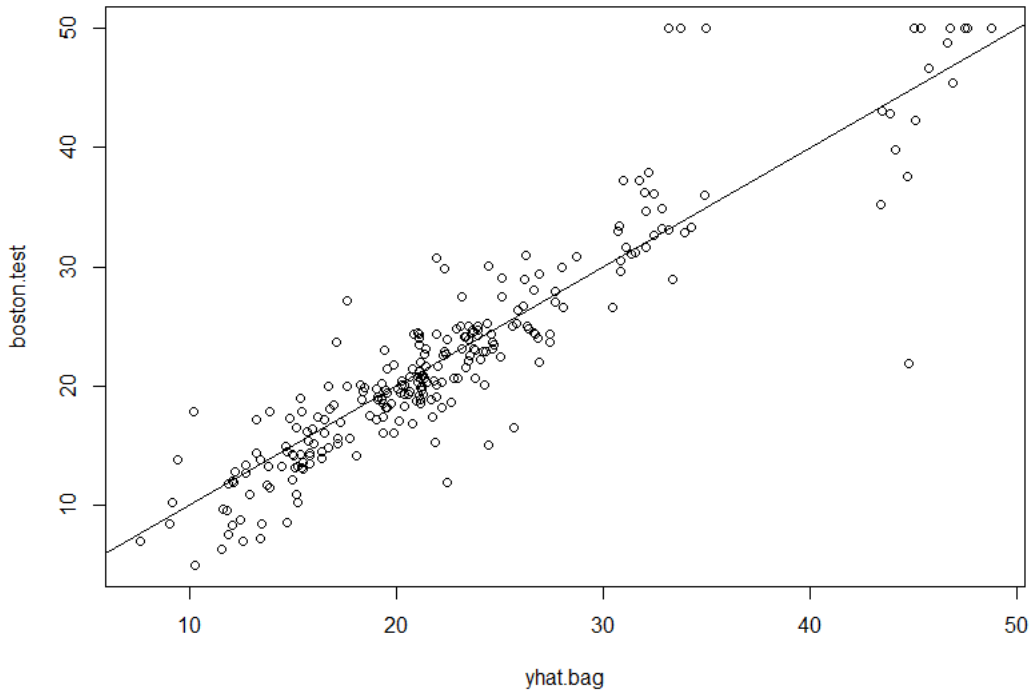
R Code:

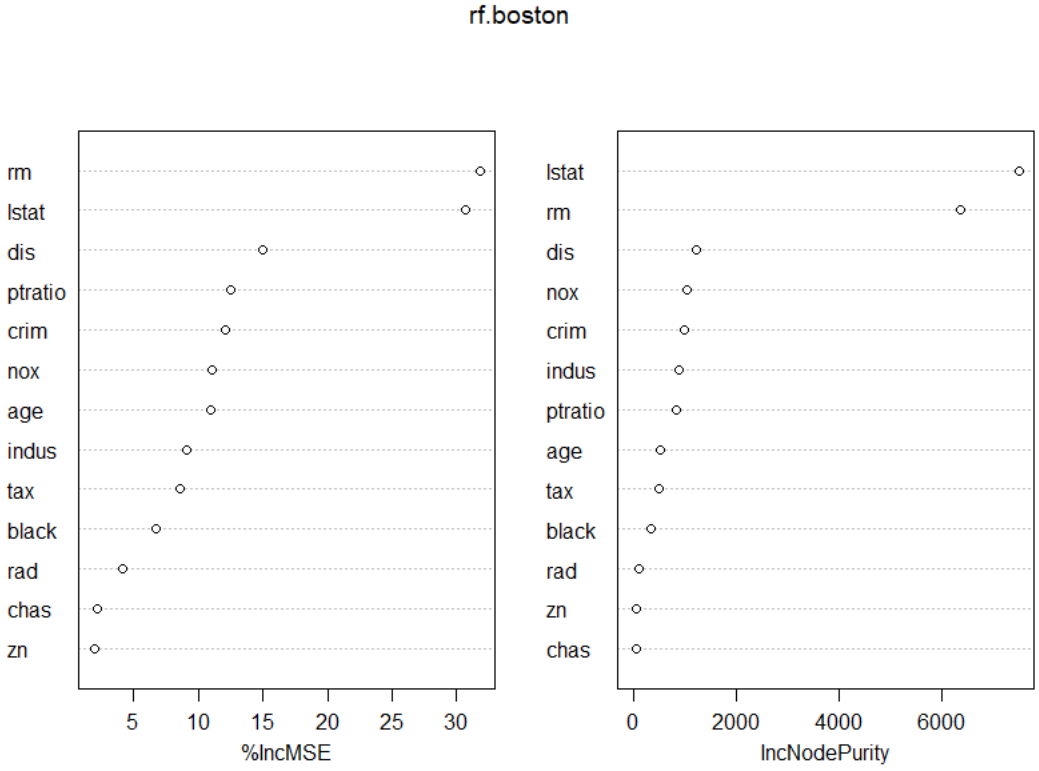
```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> library(MASS)
> library(randomForest)
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.
> 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: 11.15723
                    % Var explained: 86.49
> yhat.bag = predict(bag.boston,newdata=Boston[-train,])
> plot(yhat.bag, boston.test)
> abline(0,1)
> mean((yhat.bag-boston.test)^2)
[1] 13.50808
> 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] 13.94835
> 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] 11.66454
```

```
[1] 11.66454
> importance(rf.boston)
         %IncMSE IncNodePurity
crim    12.132320     986.50338
zn       1.955579      57.96945
indus    9.069302     882.78261
chas     2.210835      45.22941
nox     11.104823    1044.33776
rm      31.784033    6359.31971
age     10.962684     516.82969
dis     15.015236    1224.11605
rad      4.118011      95.94586
tax      8.587932     502.96719
ptratio 12.503896     830.77523
black    6.702609     341.30361
lstat   30.695224    7505.73936
> varImpPlot(rf.boston)
>
```

Plotted Graph:





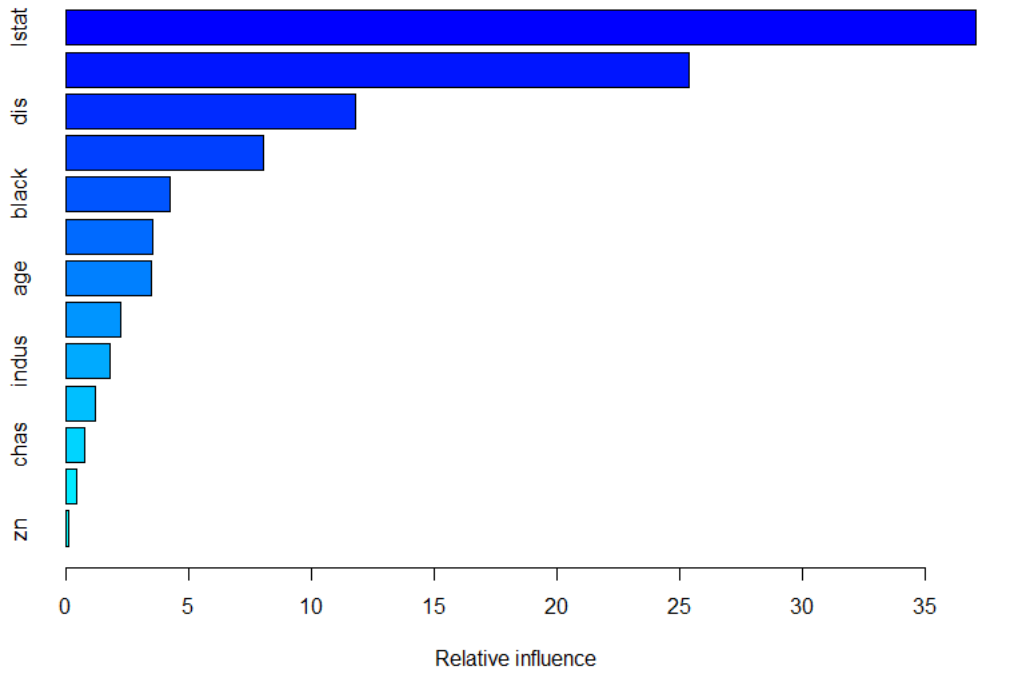

rf.boston
rm
lstat
dis
ptratio
crim
nox
age
indus
tax
black
rad
chas
zn
5
10
15
20
25
30
%IncMSE
lstat
rm
dis
nox
crim
indus
ptratio
age
tax
black
rad
zn
chas
0
2000
4000
6000
IncNodePurity

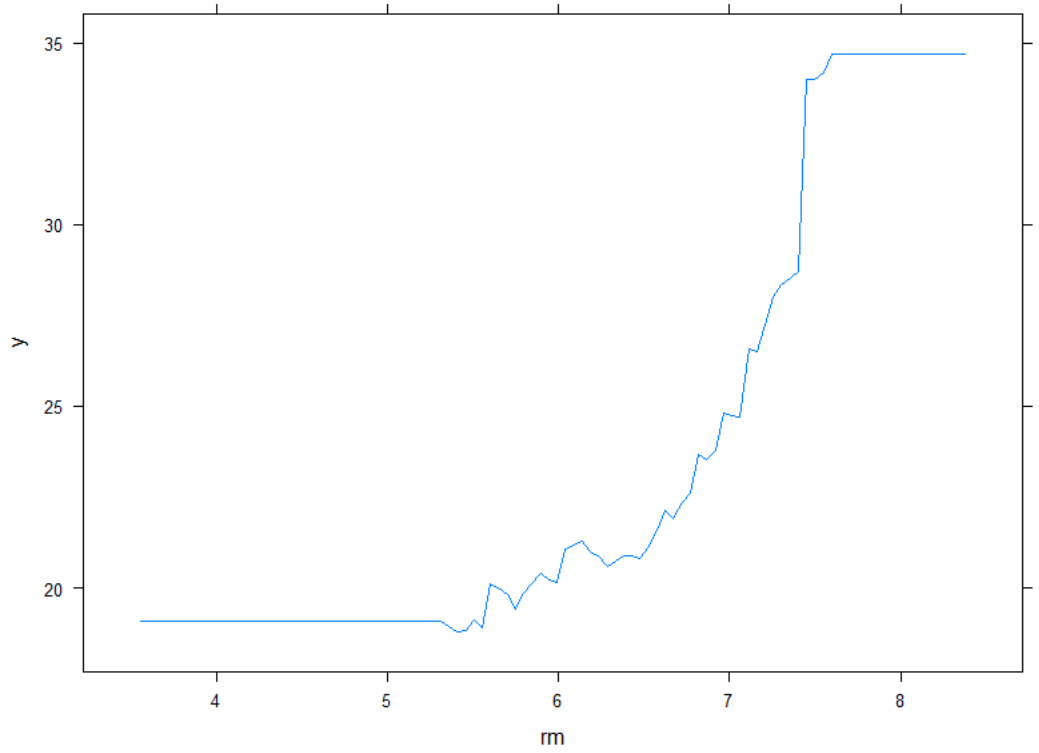
4) Boosting

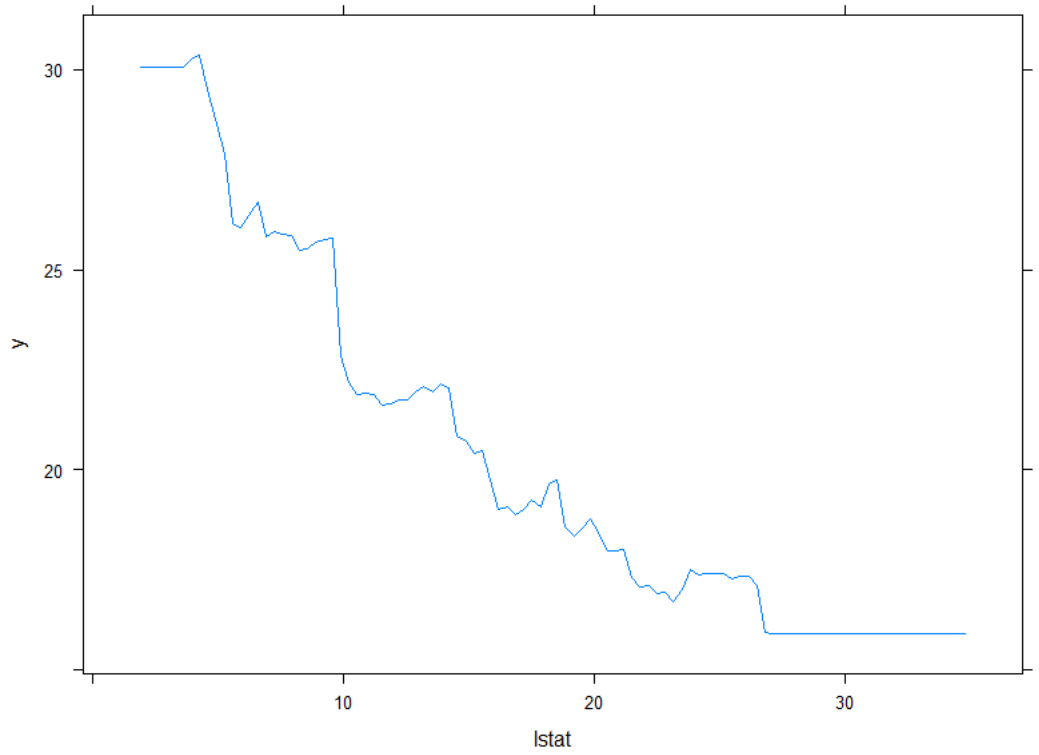
R Code:

```
> library(gbm)
Loaded gbm 2.1.4
> library(MASS)
> set.seed(1)
> boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4)
> summary(boost.boston)
             var    rel.inf
lstat     lstat 37.0661275
rm           rm 25.3533123
dis         dis 11.7903016
crim       crim  8.0388750
black     black  4.2531659
nox         nox  3.5058570
age         age  3.4868724
ptratio ptratio  2.2500385
indus     indus  1.7725070
tax         tax  1.1836592
chas       chas  0.7441319
rad         rad  0.4274311
zn           zn  0.1277206
> 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)
[1] 10.81479
> 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] 11.51109
>
```

Plotted Graphs:




35
30
25
20
y
4
5
6
7
8
rm


30
25
20
y
10
20
30
lstat