Introduction To Machine Learning: Project 2

SARIMZIA | 50245868

17 April 2018

Task 0

Preparations

Environment used: Rstudio Version 1.1.442 on Windows 10

TASK 1

8.3.1 Fitting classification Trees

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.4.4
```

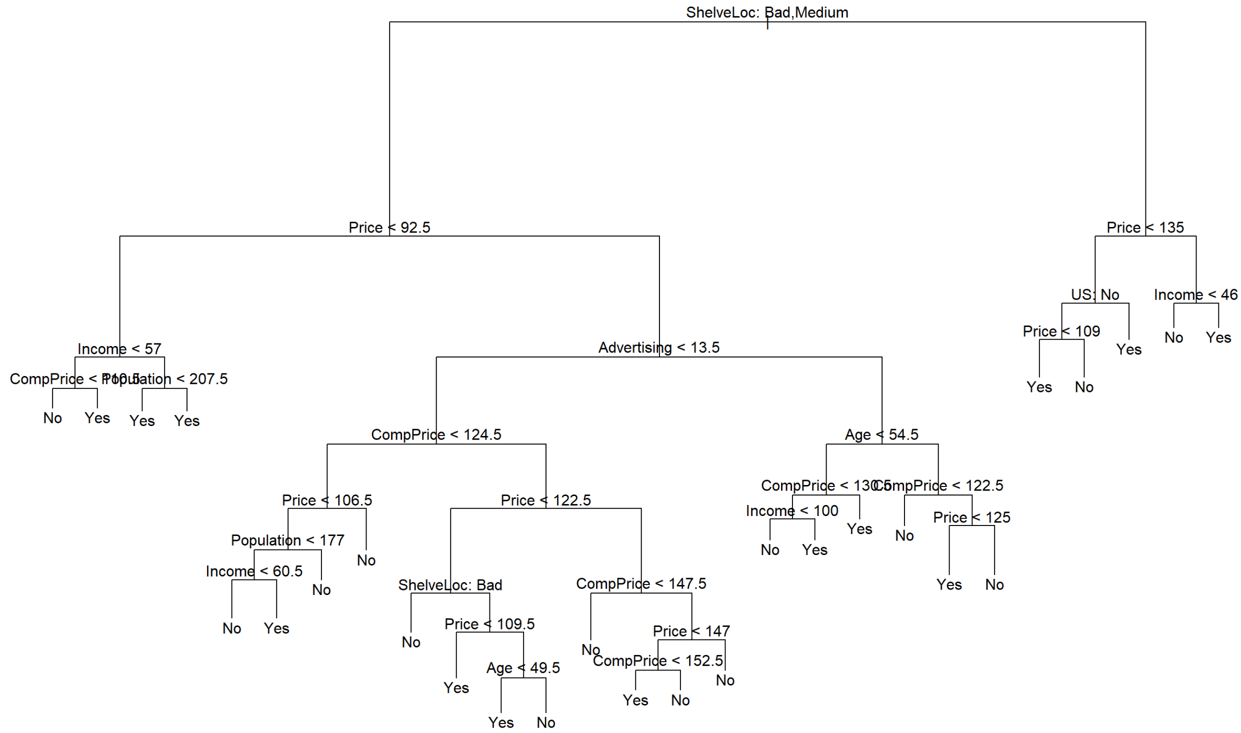
```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.4.4
```

```
attach(Carseats)
High <- with(Carseats, 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  86  27
##       Yes 30  57
```

```
(86+57)/200
```

```
## [1] 0.715
```

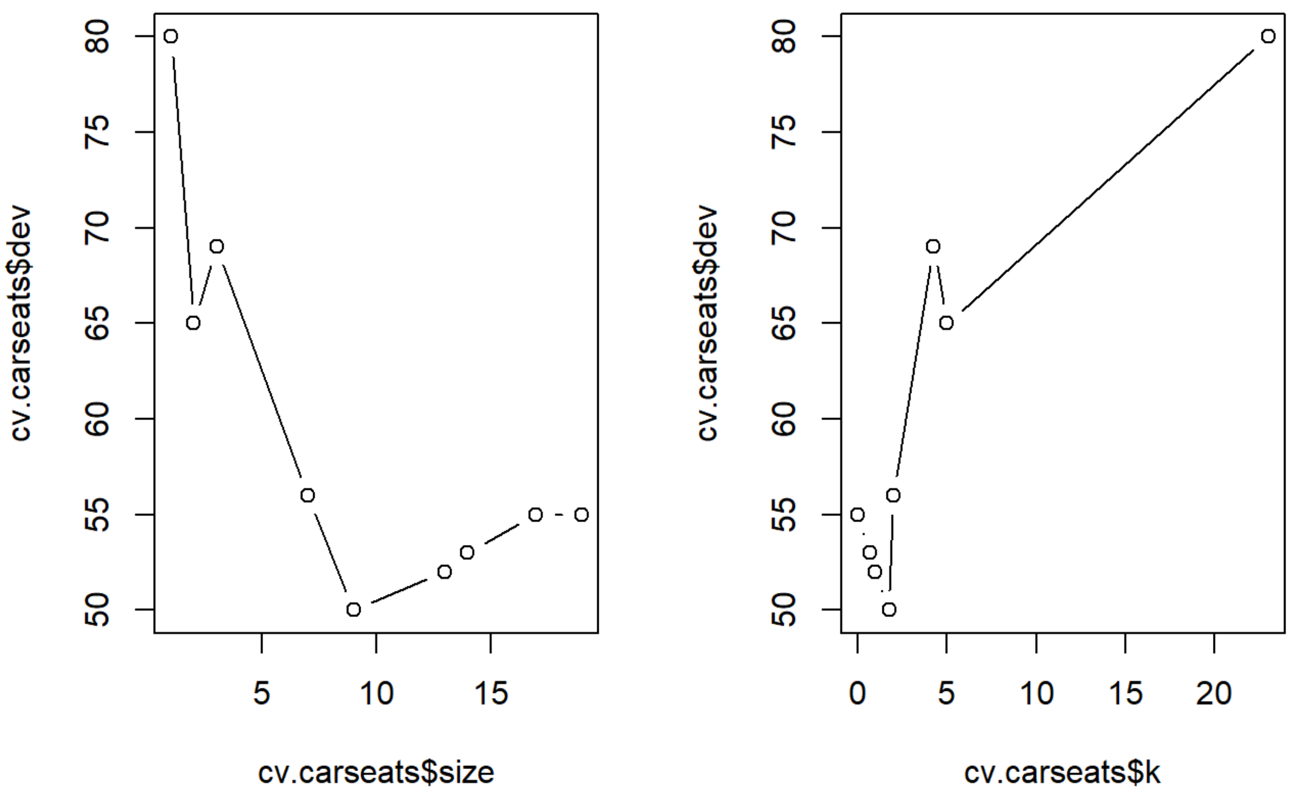
```
set.seed(3)
cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)
names(cv.carseats)
```

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

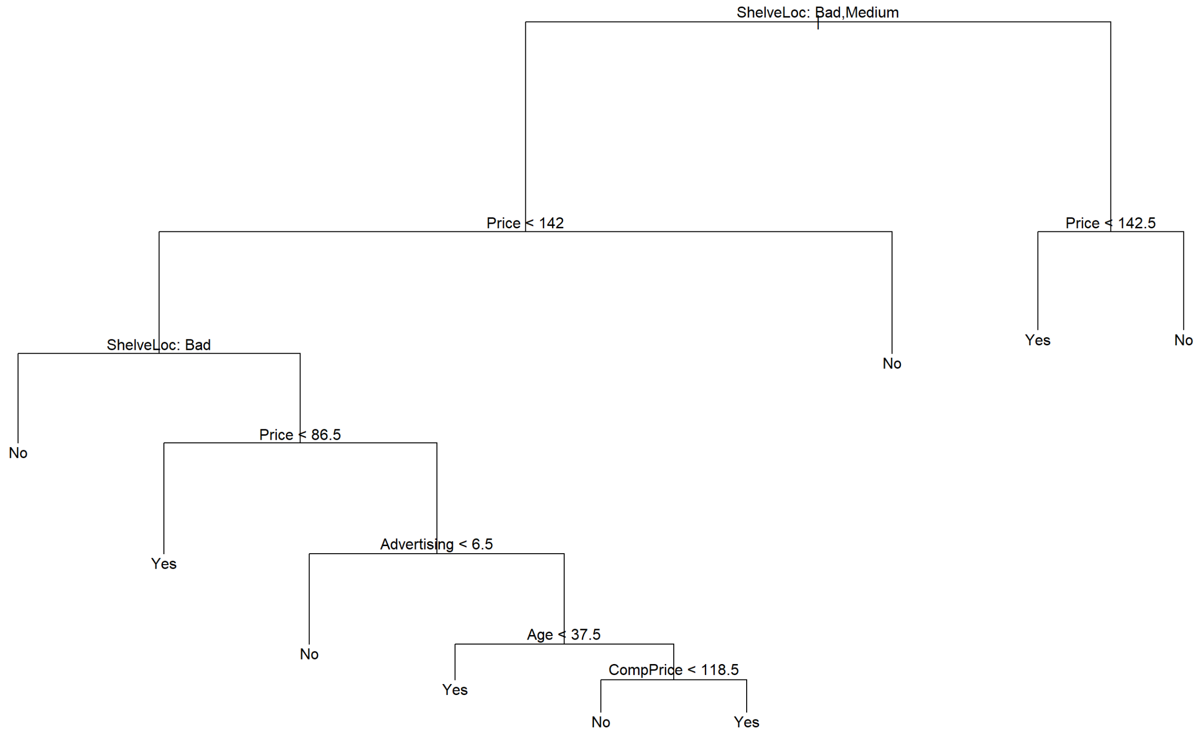
```
cv.carseats
```

```
## $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
## [7]  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")
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)
```



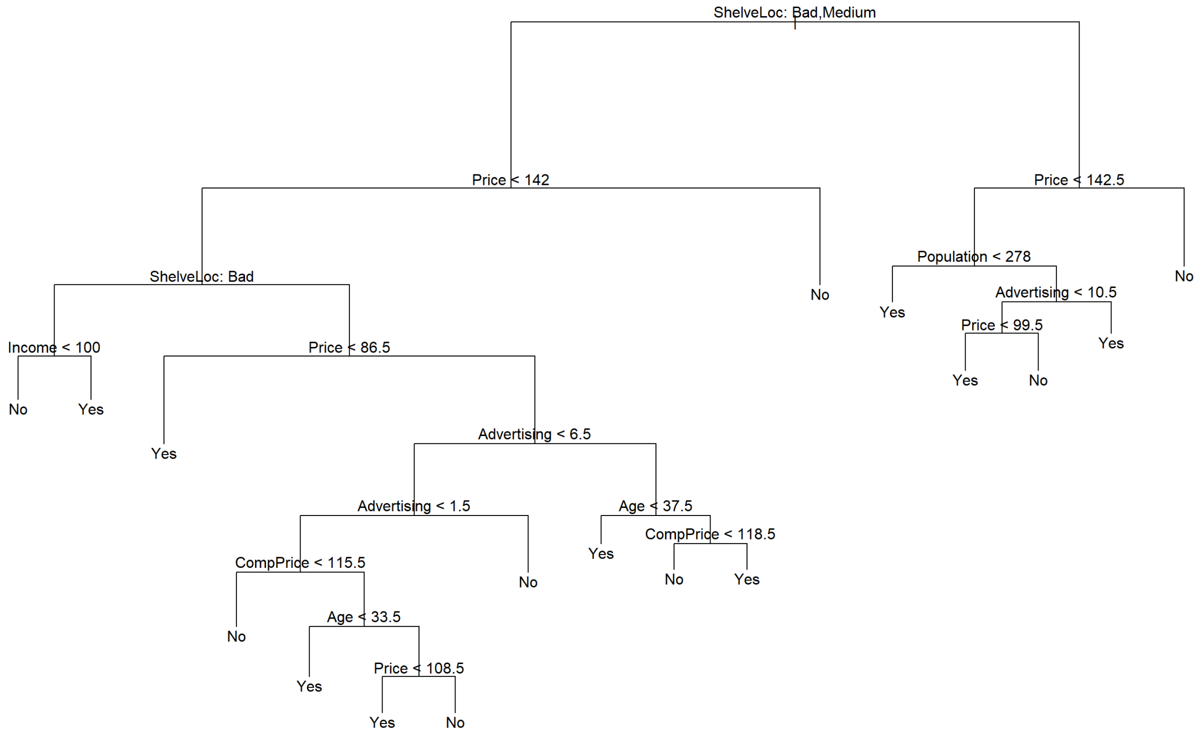

```
tree.pred=predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
```

```
##           High.test
## tree.pred No Yes
##       No  94  24
##       Yes 22  60
```

```
(94+60)/200
```

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

```
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
```

```
(86+62)/200
```

```
## [1] 0.74
```

TASK 2

8.3.2 Fitting Regression Trees

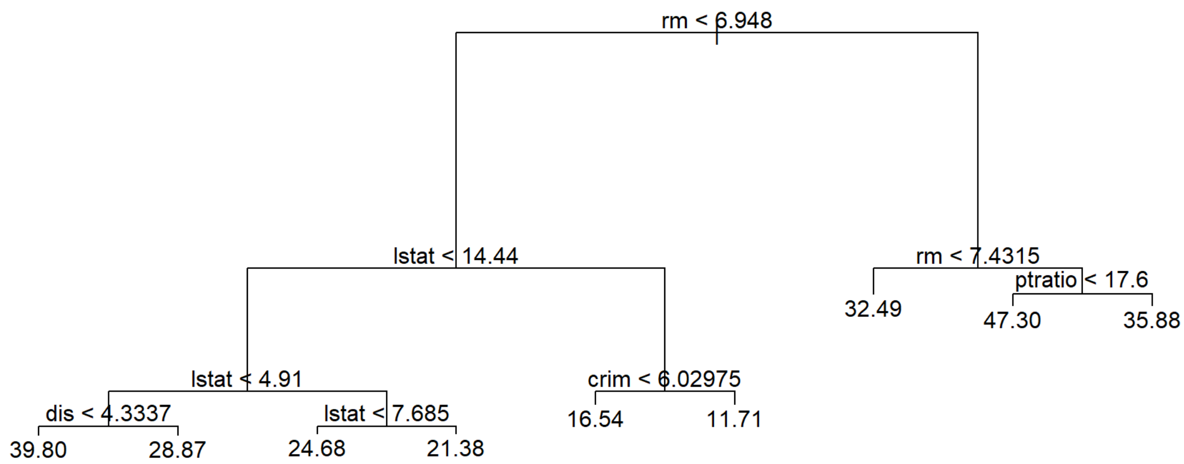
```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.4.4
```

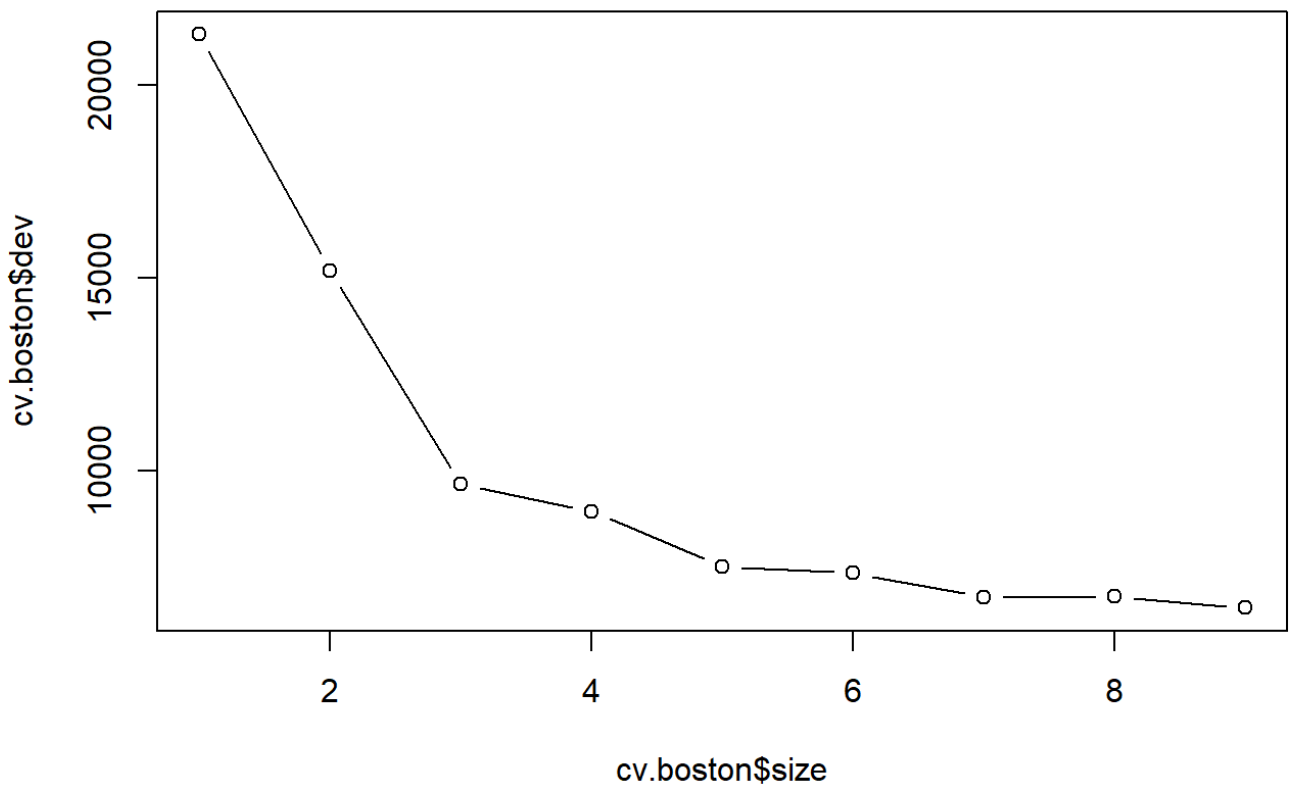
```
set.seed=1
train=sample(1:nrow(Boston),nrow(Boston)/2)
tree.boston=tree(medv~.,Boston,subset=train)
summary(tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "rm"      "lstat"   "dis"     "crim"    "ptratio"
## Number of terminal nodes:  9
## Residual mean deviance:  13.51 = 3298 / 244
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -14.6900  -1.7790  -0.1793   0.0000   2.0210  17.5100
```

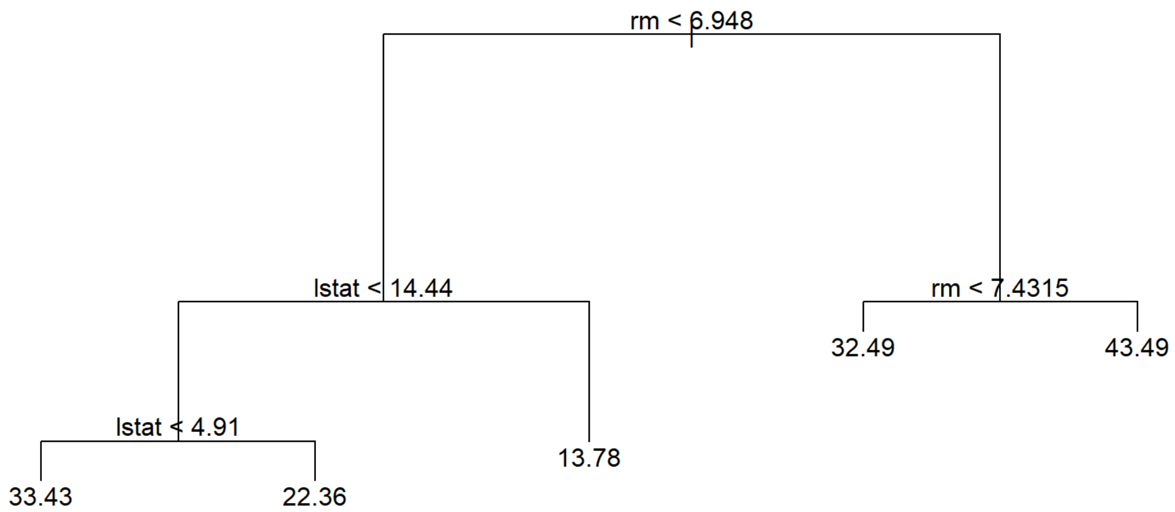
```
plot(tree.boston)
text(tree.boston,pretty=0)
```



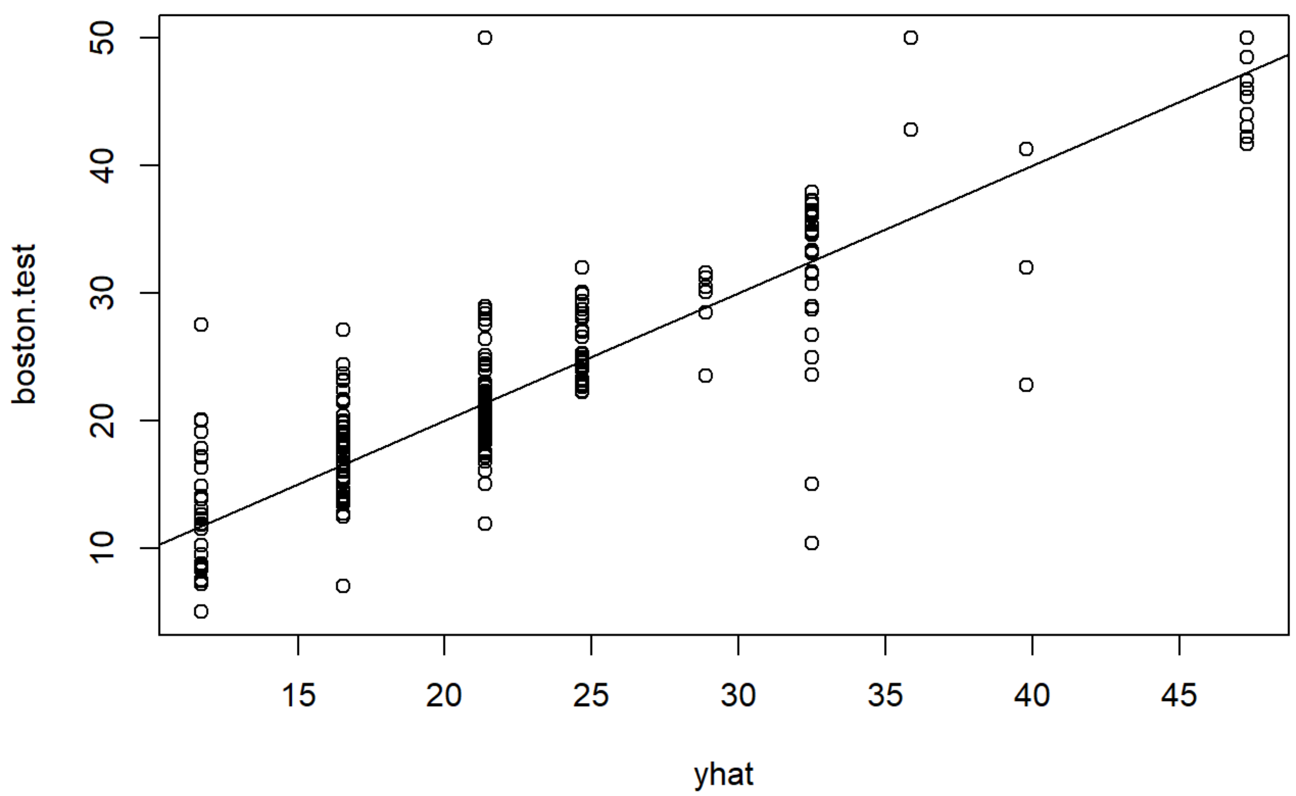

```
cv.boston=cv.tree(tree.boston)
plot(cv.boston$size,cv.boston$dev,type='b')
```




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



```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] 24.48727
```

TASK 3

8.3.3 Bagging and Random Forests

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

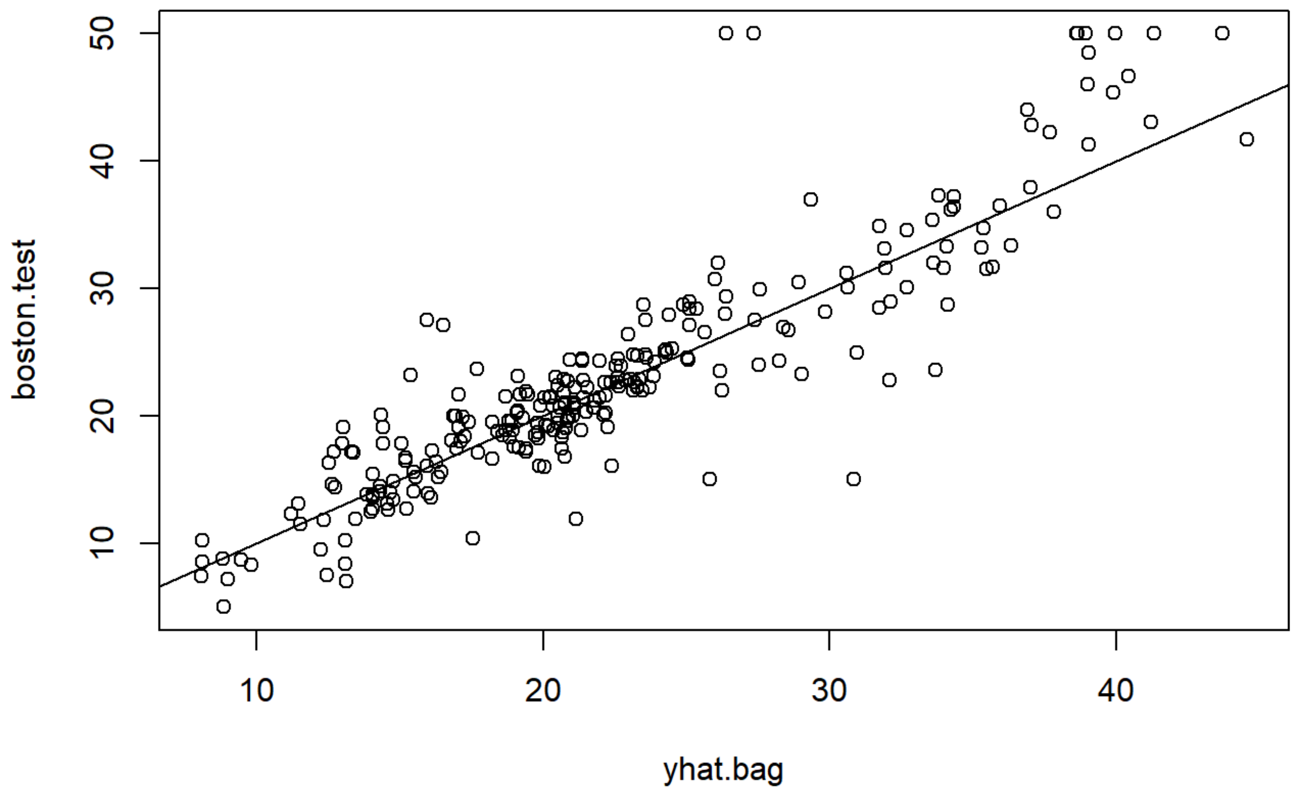
```
## randomForest 4.6-14
```

```
## 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: 11.78114
##                     % Var explained: 85.82
```

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




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

```
## [1] 17.11319
```

```
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] 17.19022
```

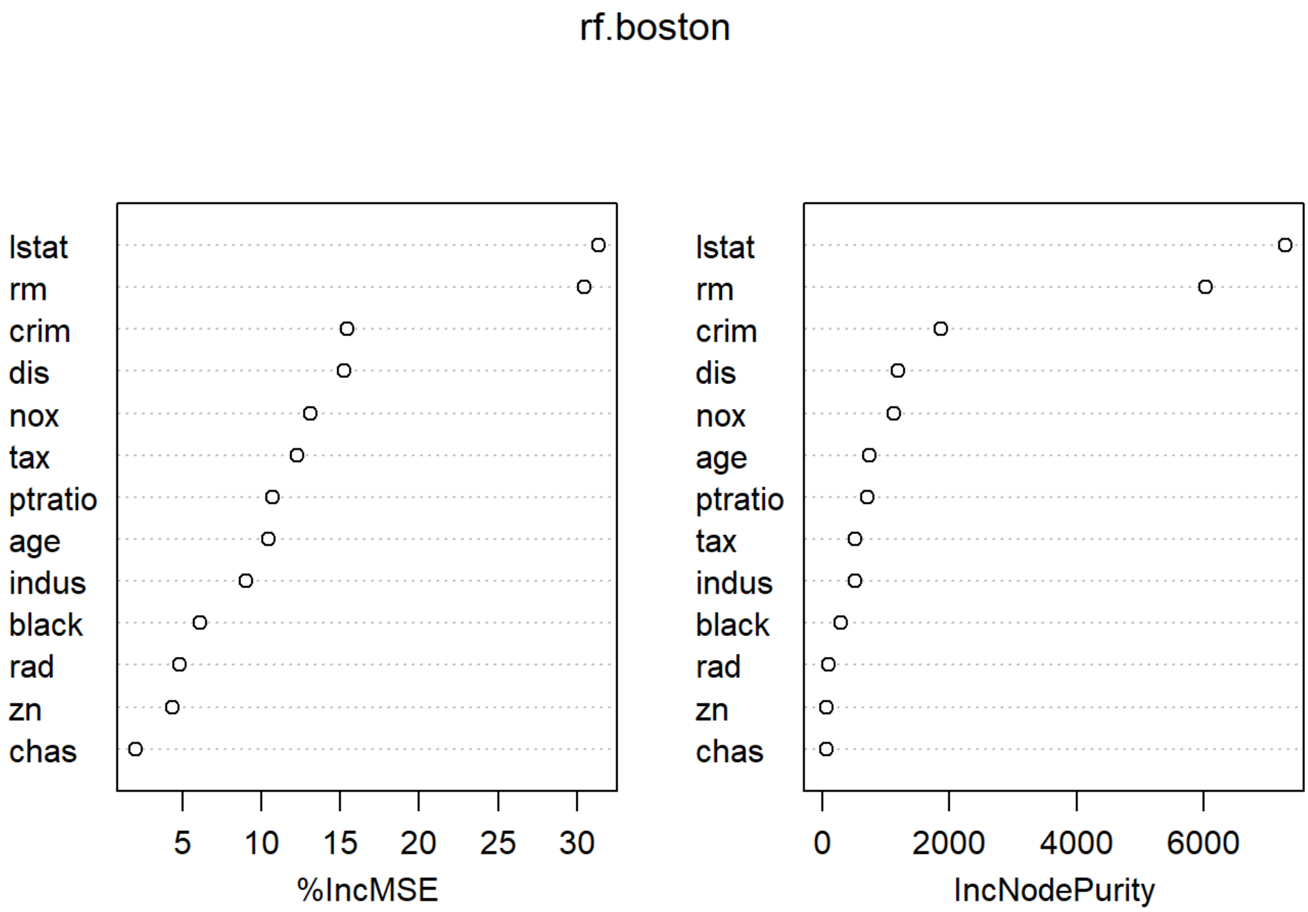
```
set.seed(1)
rf.boston=randomForest(medv~.,data=Boston,subset=train,mtry=6,importance=TRUE)
yhat.rf=predict(rf.boston,newdata=Boston[-train,])
mean((yhat.rf-boston.test)^2)
```

```
## [1] 17.50849
```

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

```
##           %IncMSE IncNodePurity
## crim    15.458985    1868.86996
## zn       4.381942      68.33245
## indus    9.014757     505.06808
## chas     2.047497      53.08569
## nox     13.085996    1130.86998
## rm      30.469371    6034.00836
## age     10.434793     733.86943
## dis     15.270284    1180.24510
## rad      4.830909      95.98200
## tax     12.299406     512.53066
## ptratio 10.686928     700.82904
## black    6.087007     285.82423
## lstat   31.377983    7286.11343
```

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



TASK 4

8.3.4 Boosting

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.4.4
```

```
## Loading required package: survival
```

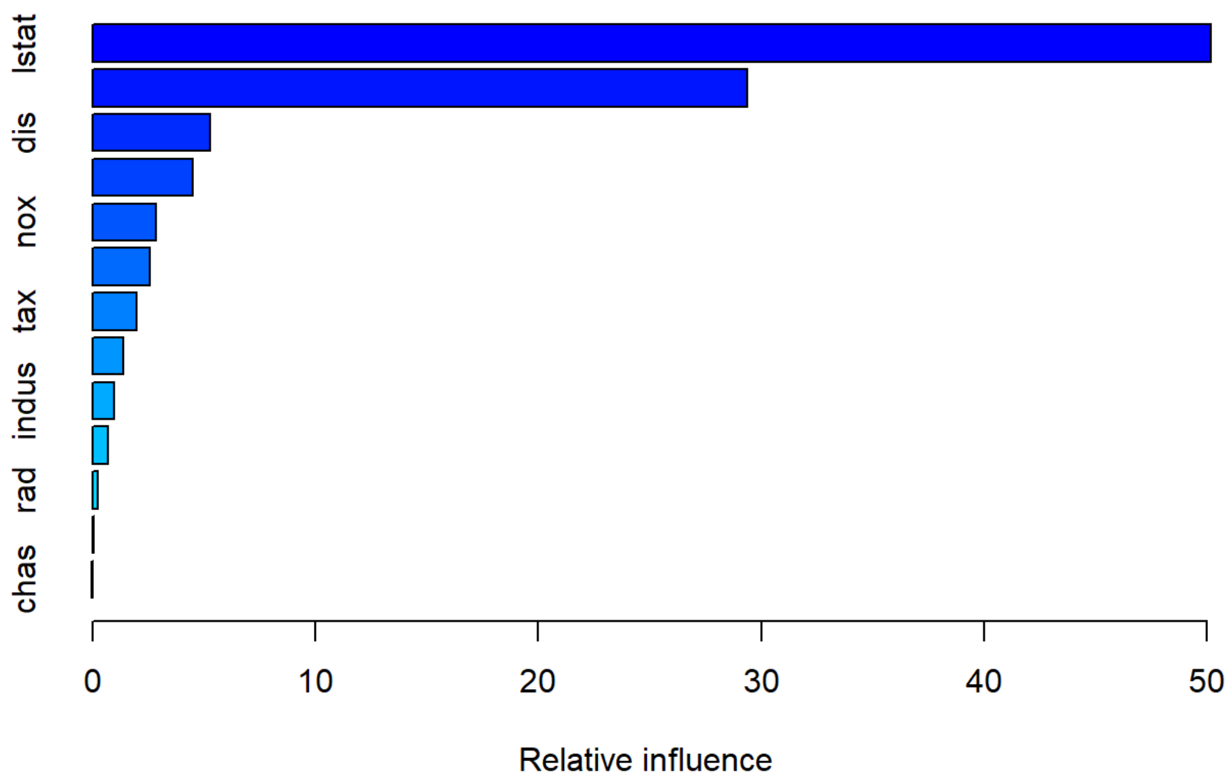
```
## Loading required package: lattice
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

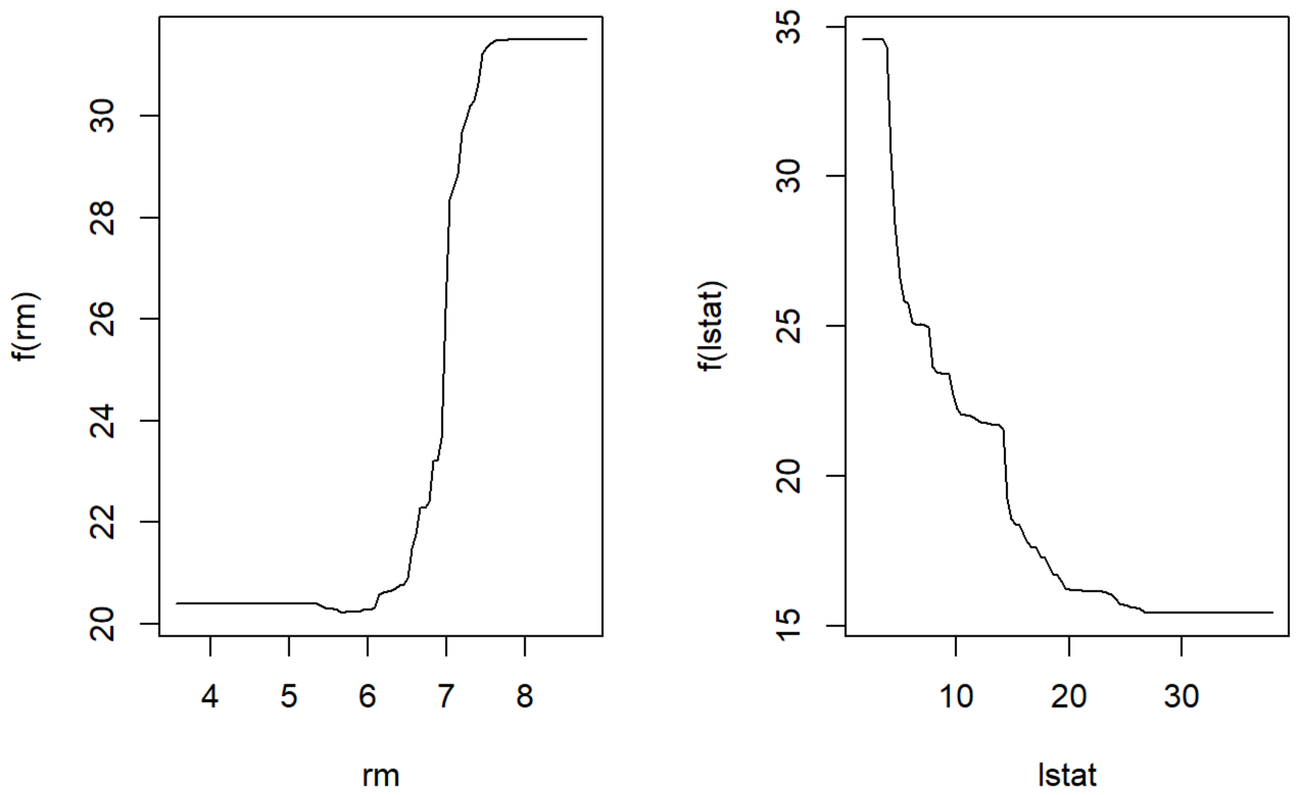
```
## Loaded gbm 2.1.3
```

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




```
##             var     rel.inf
## lstat     lstat 50.16712978
## rm           rm 29.36863544
## dis         dis  5.27479136
## crim       crim  4.47470271
## nox         nox  2.84121915
## ptratio ptratio  2.56803972
## tax         tax  1.98645724
## age         age  1.36264494
## indus     indus  0.97373021
## black     black  0.68537903
## rad         rad  0.22960165
## zn           zn  0.04429074
## chas       chas  0.02337802
```

```
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] 17.47222
```

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

```
## [1] 17.47222
```

TASK 5

Summary

It took me 2 days to finish this assignment. This time I did not encounter any difficulties as the tasks were pretty simple and the instructions were concise. I collaborated with Someshwar Rao for this assignment, only for the initial part, where I had to set up the environment and start with Rmarkdown.