**CSE 574**

**Assign 2**

**Name: Rajiv Ranjan**

**Ub id: 50249099**

**Ub it:rajivran**

**TASK 0 Preparation 10 points (state in your report the platform you use)**

**Soln:**

Platform used:

R studio: version 1.1.419

Rmarkdown (was used for producing your report).

Library used:

library(randomForest)

library(MASS)

require(tree)

require(devtools)

require(ISLR)

require(MASS)

require(randomForest)

require(gbm)

library(caret)

attach ( Carseats )

**TASK 1 Fitting Classification Trees**

**Soln:**

The solution for this is in filename : assign2\_part1.Rmd

require(tree)

require(devtools)

require(ISLR)

require(MASS)

require(randomForest)

library(gbm)

o/p:: Using classification trees to analyze the Carseats data set.Sales is a continuous variable, and so we begin by recoding it as a binary variable.

library (ISLR )

attach ( Carseats )

High= ifelse (Sales <=8 ," No"," Yes ")

Carseats = data.frame (Carseats , High)

tree.carseats =tree (High~.-Sales , Carseats )

summary (tree.carseats )

**o/p::**

Classification tree:

tree(formula = High ~ . - Sales, data = Carseats)

Variables actually used in tree construction:

[1] "ShelveLoc" "Price" "Income" "CompPrice" "Population" "Advertising" "Age"

[8] "US"

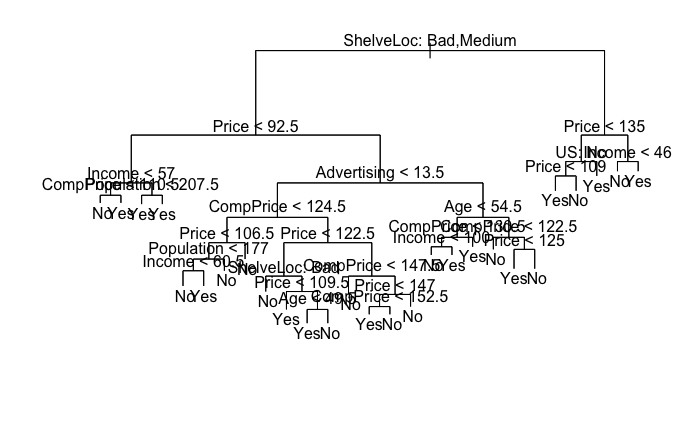
Number of terminal nodes: 27

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

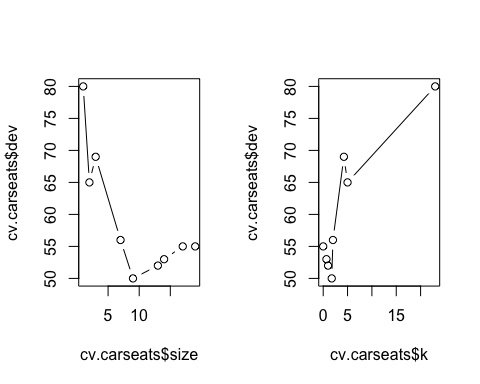
set.seed (3)  
cv.carseats =cv.tree(tree.carseats ,FUN = prune.misclass )  
names (cv.carseats )

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

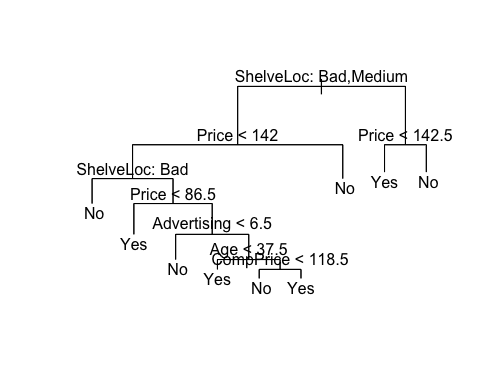
cv.carseats

## $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")



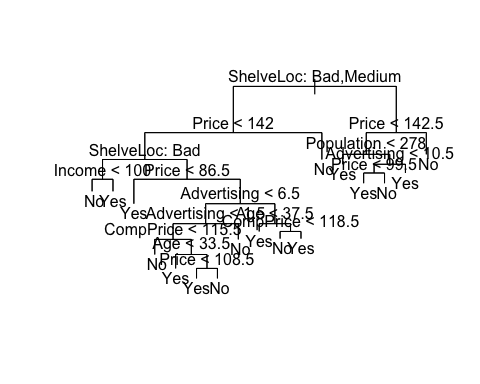
prune.carseats = prune.misclass ( tree.carseats , best =9)  
plot( prune.carseats )  
text( prune.carseats , pretty =0)



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

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

**TASK 2 Fitting Classification Trees**

**Soln:**

The solution for this is in filename : assign2\_part2.Rmd

TASK 2 : Fitting Regression Trees 20 points

library (MASS)  
require(tree)

## Loading required package: tree

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

require(devtools)

## Loading required package: devtools

require(ISLR)

## Loading required package: ISLR

require(MASS)   
require(randomForest)

## Loading required package: 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.

require(gbm)

## Loading required package: gbm

## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

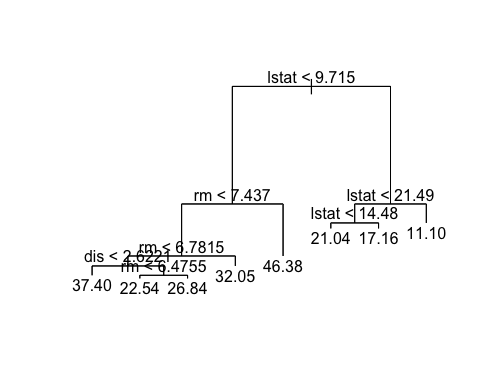
## Loading required package: parallel

## Loaded gbm 2.1.3

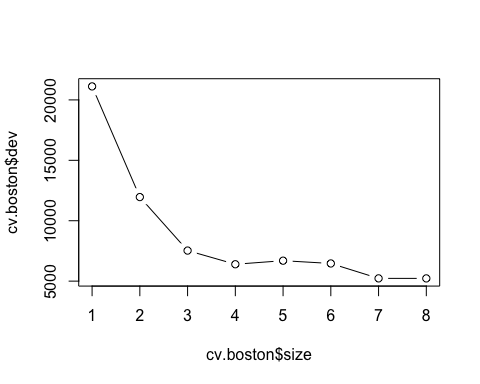
set.seed (1)  
train = sample (1: nrow ( Boston ), nrow( Boston )/2)  
tree.boston = tree(medv~.,Boston , subset = train )  
summary (tree.boston )

##   
## 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

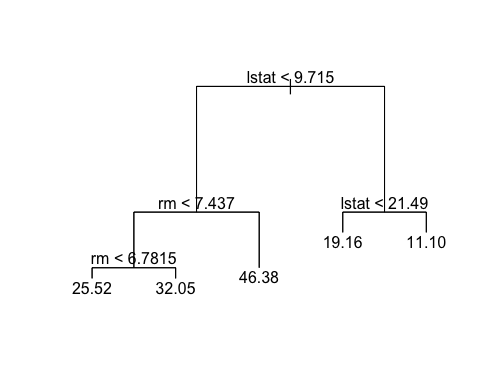
plot(tree.boston )  
text(tree.boston , pretty =0)



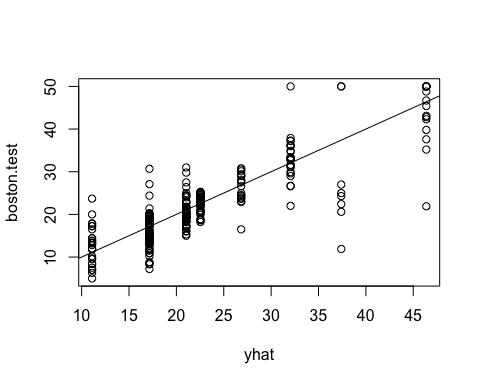
cv.boston =cv.tree( tree.boston )  
plot(cv.boston$size ,cv.boston$dev ,type="b")



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

**TASK 3 Bagging and Random Forests**

**Soln:**

The solution for this is in filename : assign2\_part3.Rmd

library(randomForest)

library(MASS)

require(tree)

require(devtools)

require(ISLR)

require(MASS)

require(randomForest)

require(gbm)

library(caret)

set.seed(1)

bag.boston = randomForest( medv~., data = Boston , subset = train, mtry =13, importance =TRUE )

bag.boston

**o/p::**

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

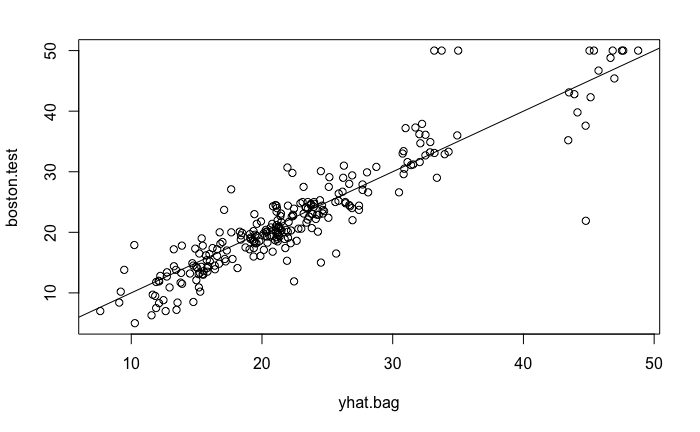
**yhat.bag = predict (bag.boston , newdata = Boston [-train ,])**

**plot(yhat.bag , boston.test)**

**abline (0 ,1)**

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

**O/p::**



```{r}

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)

```

**o/p:: [1] 13.94835**

```{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)

```

**o/p::**

**[1] 11.66454**

```{r}

importance (rf.boston )

```

**o/p**

%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

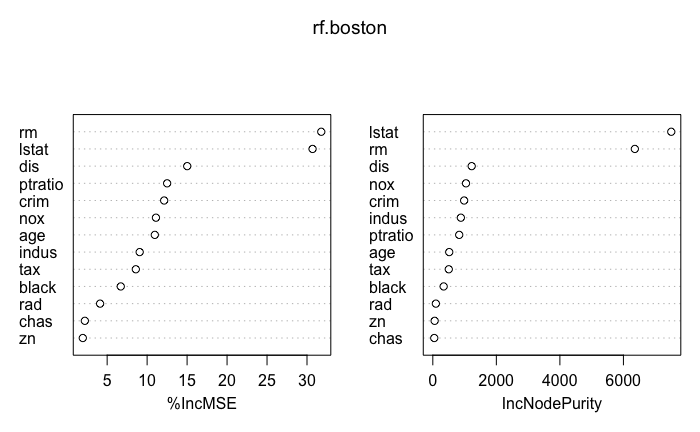
lstat 30.695224 7505.73936

```{r}

varImpPlot (rf.boston )

```

**o/p::**



**TASK 4 Boosting**

Soln:

The solution for this is in filename : assign2\_part4.Rmd

library (gbm )

library (MASS)

require(tree)

require(devtools)

require(ISLR)

require(MASS)

require(randomForest)

library(caret)

set.seed (1)

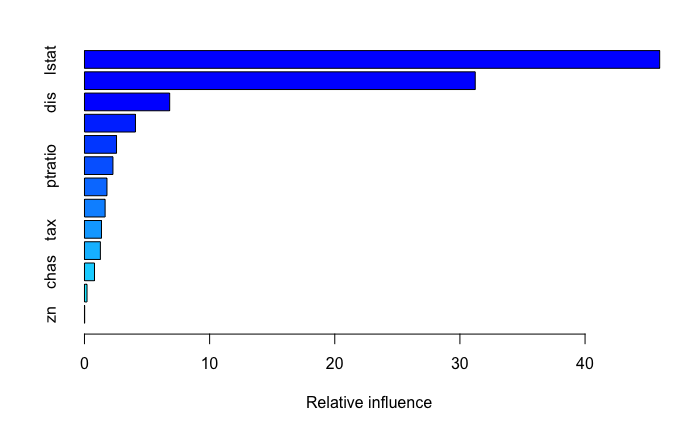
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4)

summary(boost.boston)

**o/p ::**

|  |
| --- |
|  |
|  | **var**  <fctr> | | **rel.inf**  <dbl> | |  |  | |
| lstat | lstat | | 45.9627334 | |  |  | |
| rm | rm | | 31.2238187 | |  |  | |
| dis | dis | | 6.8087398 | |  |  | |
| crim | crim | | 4.0743784 | |  |  | |
| nox | nox | | 2.5605001 | |  |  | |
| ptratio | ptratio | | 2.2748652 | |  |  | |
| black | black | | 1.7971159 | |  |  | |
| age | age | | 1.6488532 | |  |  | |
| tax | tax | | 1.3595005 | |  |  | |
| indus | indus | | 1.2705924 | |  |  | |
|  | |
| chas | | chas | | 0.8014323 | |  |  |
| rad | | rad | | 0.2026619 | |  |  |
| zn | | zn | | 0.0148083 | |  |  |

**o/p2::**



**```{r}**

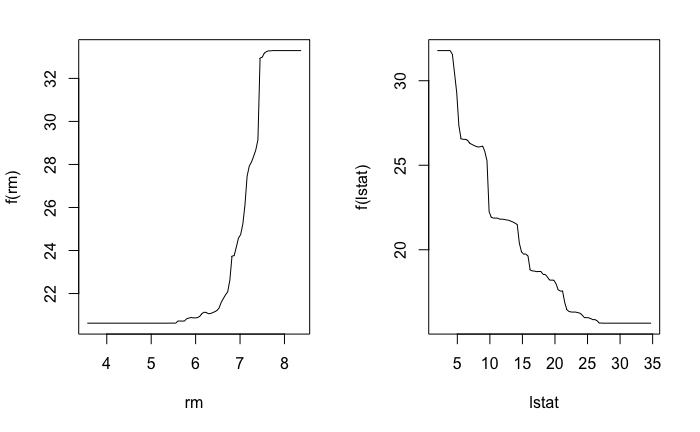
**par ( mfrow =c(1 ,2) )**

**plot( boost.boston ,i="rm")**

**plot( boost.boston ,i="lstat")**

**```**

**o/p::**



**```{r}**

**yhat.boost = predict (boost.boston , newdata = Boston [-train ,],n.trees =5000)**

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

**```**

**o/p::** [1] 11.84434

```{r}

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)

```

**o/p:: [1] 11.51109**

**TASK 5 Summary**

Report time required for homework, difficulties encountered, and collaborators

Soln:

**Report Time:** The project took 4 days to complete.

**Difficulties encountered:** Decision Trees being a new algorithm did present a few initial difficulties while understanding the concepts. However, the task became much simpler due to lucid explanation given in ISL book.

**Collaborators: None**