CSE 574 - Programming Assignment 2

Deep Narayan Mishra - 50245878

April 18, 2018

Problem Statement:

The project aims to replicate Fitting Classification Tree, Fitting Regression Tree, Bagging and Random Forests, and Boosting of dependency tree in Tree Based methods.

TASK 0 - Environment Setup:

I performed all activities on Windows 10. I started with installing R and R Studio and to complete all the tasks I installed following packages in R Studio.

- **tree** – I installed this package and used for Classification and Regression Tree. It provides functionalities such as cv.tree (for cross validation), deviance.tree (to extract deviance from tree), predict.tree (to predict from fitted tree object), prune.tree (for cost-complexity pruning of tree object) etc.
- **ISLR** – Used ISLR package for the collection of data-sets such as Carseats, which is used in the introduction to statistical learning with application in R book.
- **MASS** – Installed and used MASS package. It provides functions and datasets to support Venable and Ripley. I have used Boston dataset of MASS package to demonstrate Regression Tree.
- **randomForest** – Installed and used randomForest package. The package contains functions for Classification and Regression based on forest of trees using random inputs based on Breiman. The randomForest() function of the package can be used to perform both forests and bagging function.
- **gbm** – Installed and used gbm package for boosting. The package provides generalized boosted regression model. It has an implementation of extensions to Freund and Schapire's AdeBoost algorithm and Friedmans gradient boosting machine. I have used gbm() function of the package to fit boosted regression tree to the boston data set.
- **rmarkdown** – I installed rmarkdown package and used to create R markdown for all the given tasks. Using R mark down I was able to output the code and plots in pdf, docx and html file.
- **tinytex** – I had to install tinytex to compile R markdown document to pdf file.

TAST 1 - Fitting Classification Tree

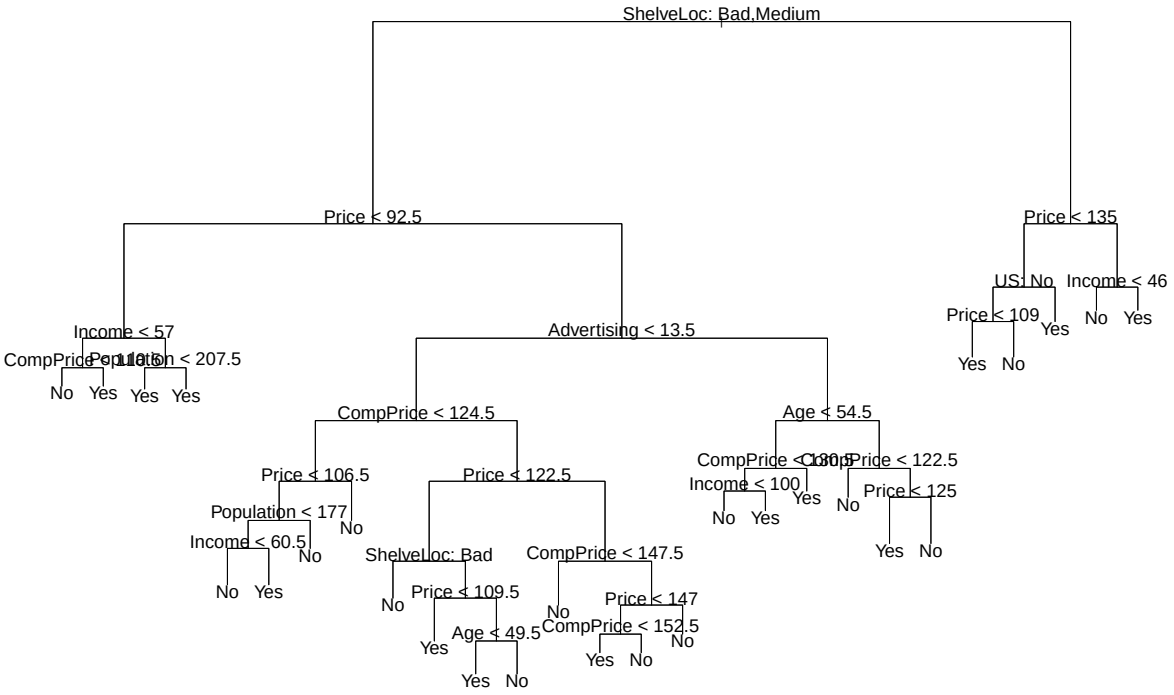
In this task we will fit classification tree. For that, we use classification trees to analyze the Carseats data set.

```
## Lists the variables used as internal nodes, terminal nodes, & (training) error rate in tree:
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats.df)
## 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 the dependency tree for carseats:
```



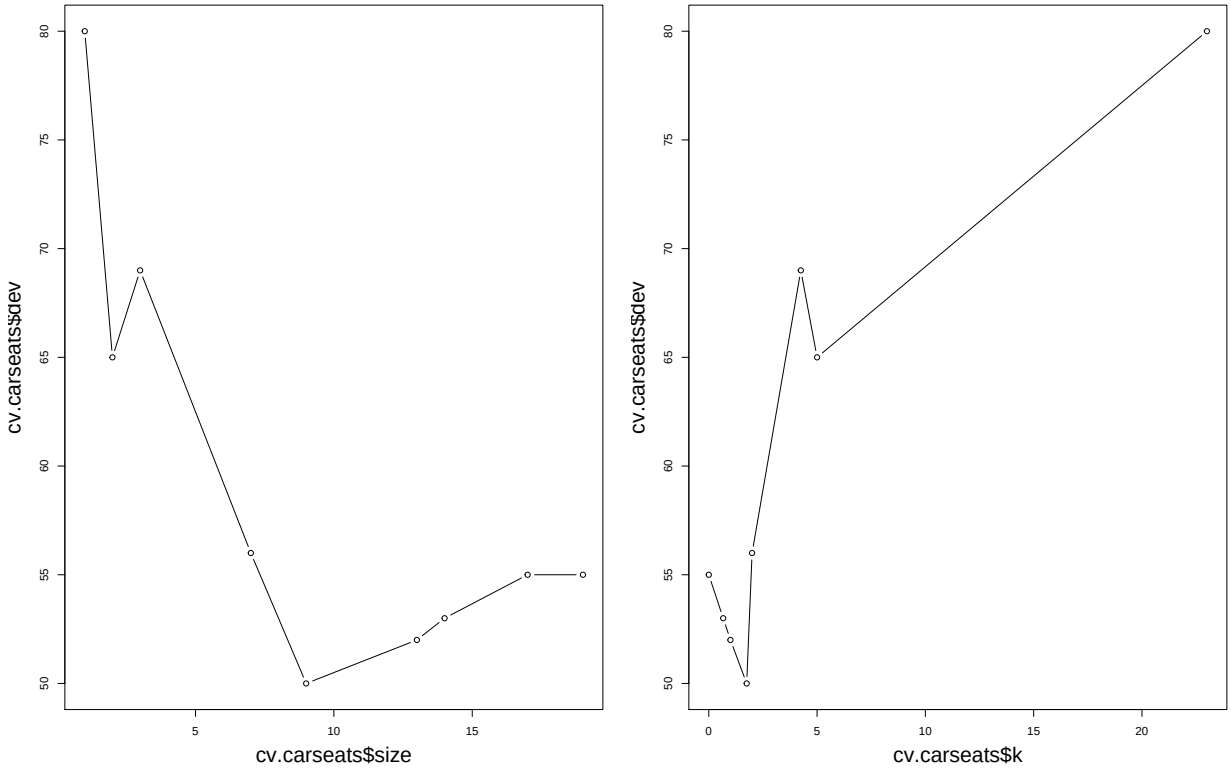

```
## The branch and split criteria:
```

```
## 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 ) *

## Evaluate tree performance on test set:

##          High.test
## tree.pred No Yes
##       No  86  27
##       Yes 30  57

## Pruning to check if it improves result accuracy

## Determine optimum level of tree complexity:

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

## $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"
```

Prune the tree based on cv result:

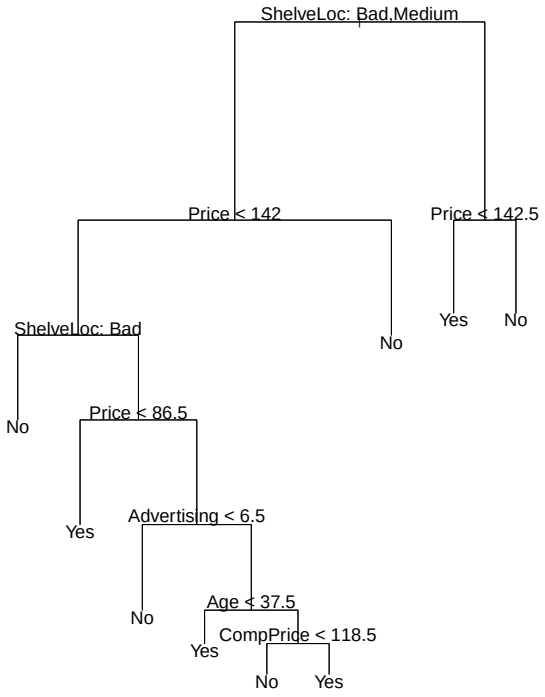


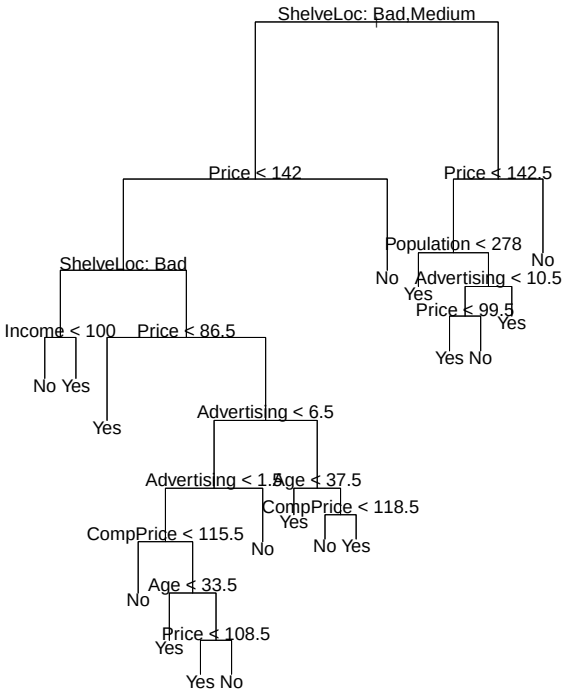
Apply prune.misclass() function to obtained 9 node tree:

Test how well pruned tree performs:

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

Obtained larger pruned by increasing the value of best with lower accuracy:






```
## Test result:
```

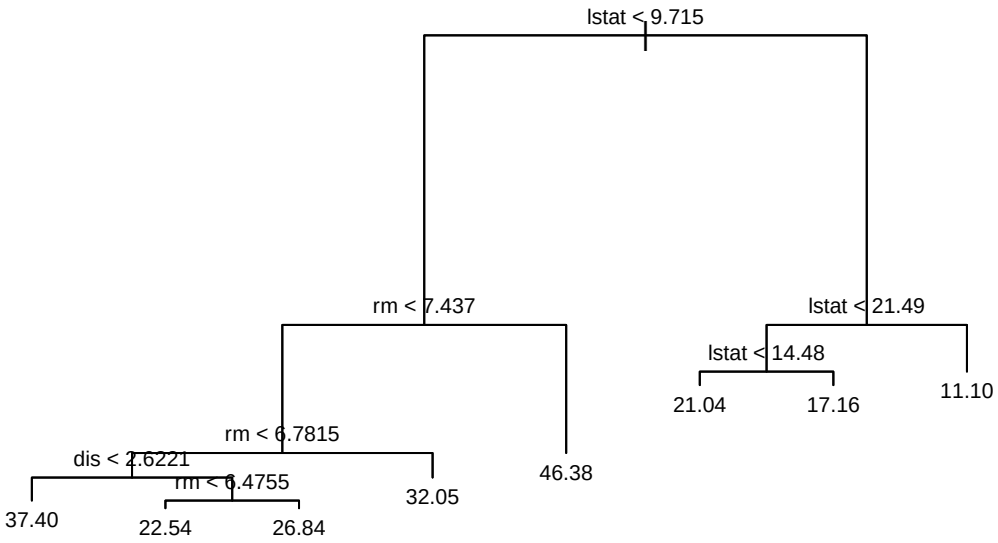
```
##          High.test
## tree.pred No Yes
##       No  86  22
##       Yes 30  62
```

TAST 2 - Fitting Regression Tree to boston dataset

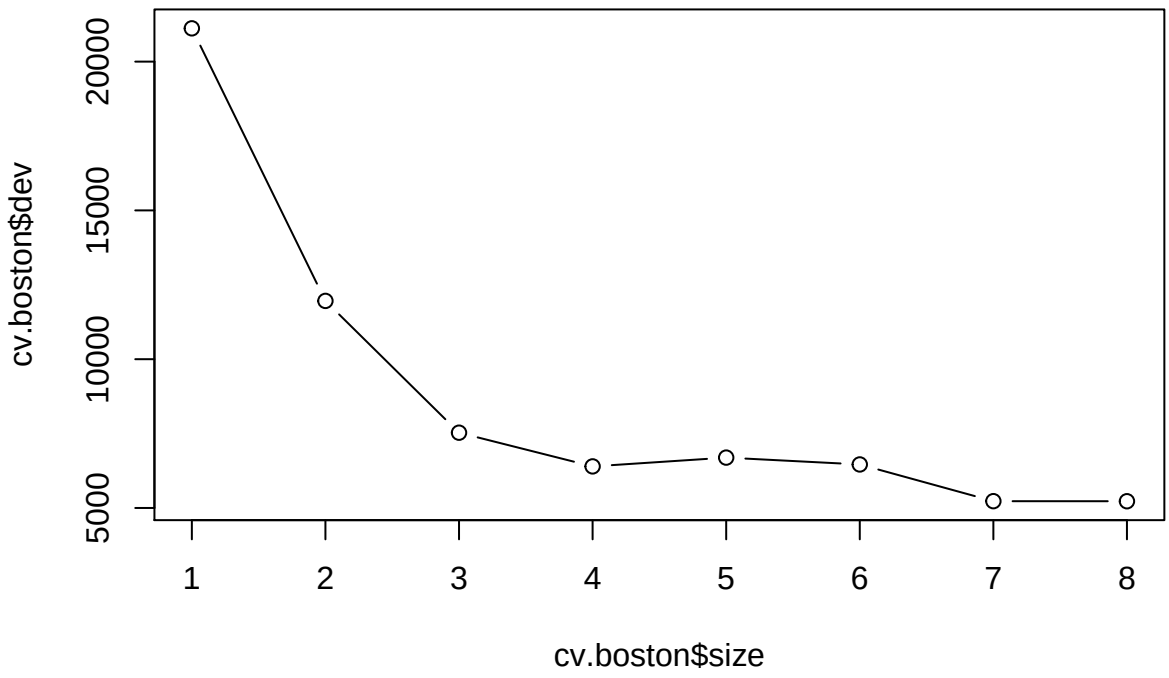
Fitting regression tree on Boston data set. First we create a training set, and fit the tree to training data.

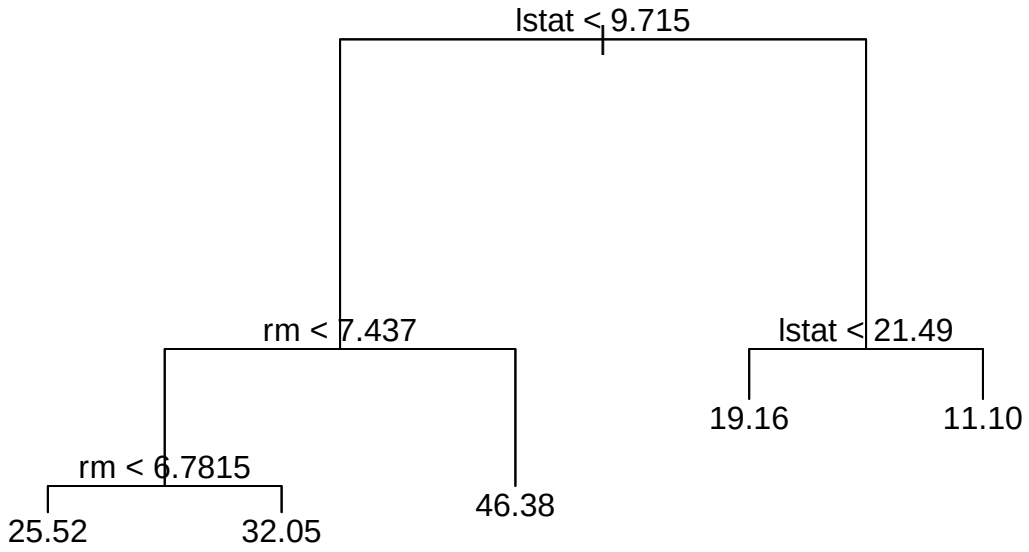
```
## Creating the training set and fitting the training data:
```

```
##
## 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
```

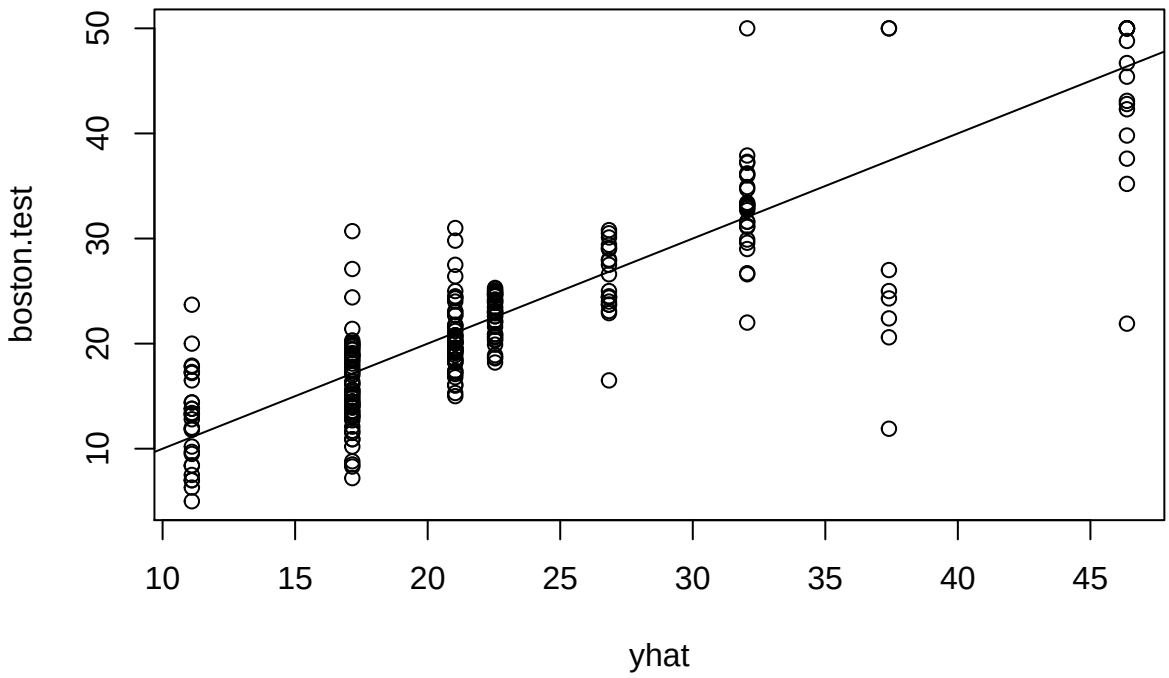


Applying pruning:


cv.boston$dev
5000
10000
15000
20000
1
2
3
4
5
6
7
8
cv.boston$size




```
## In keeping with cross validation results, we will use unpruned tree to make prediction on testset:
```

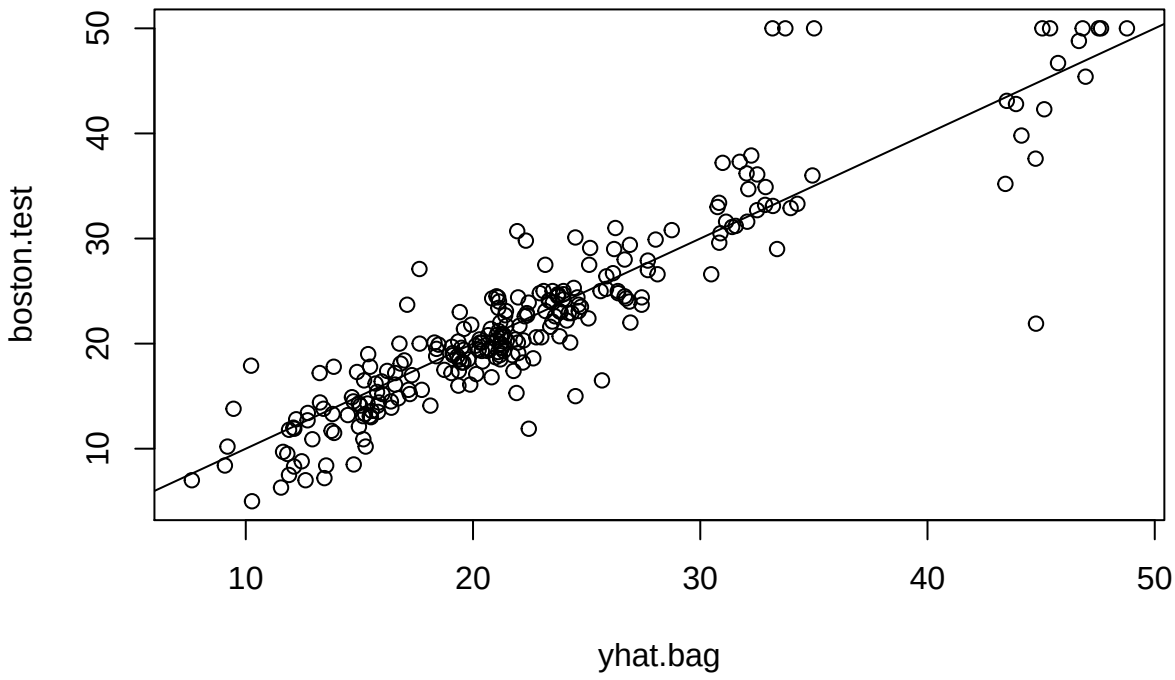



```
## Test set MSE:
## [1] 25.04559
```

TAST 3 - Bagging and Random forests

Applying Bagging and Random forests to the Boston data using the randomForest package in R.

```
##
## 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
## Test on how well does the bagged model performed on the testset:
```




```
## [1] 13.50808
```

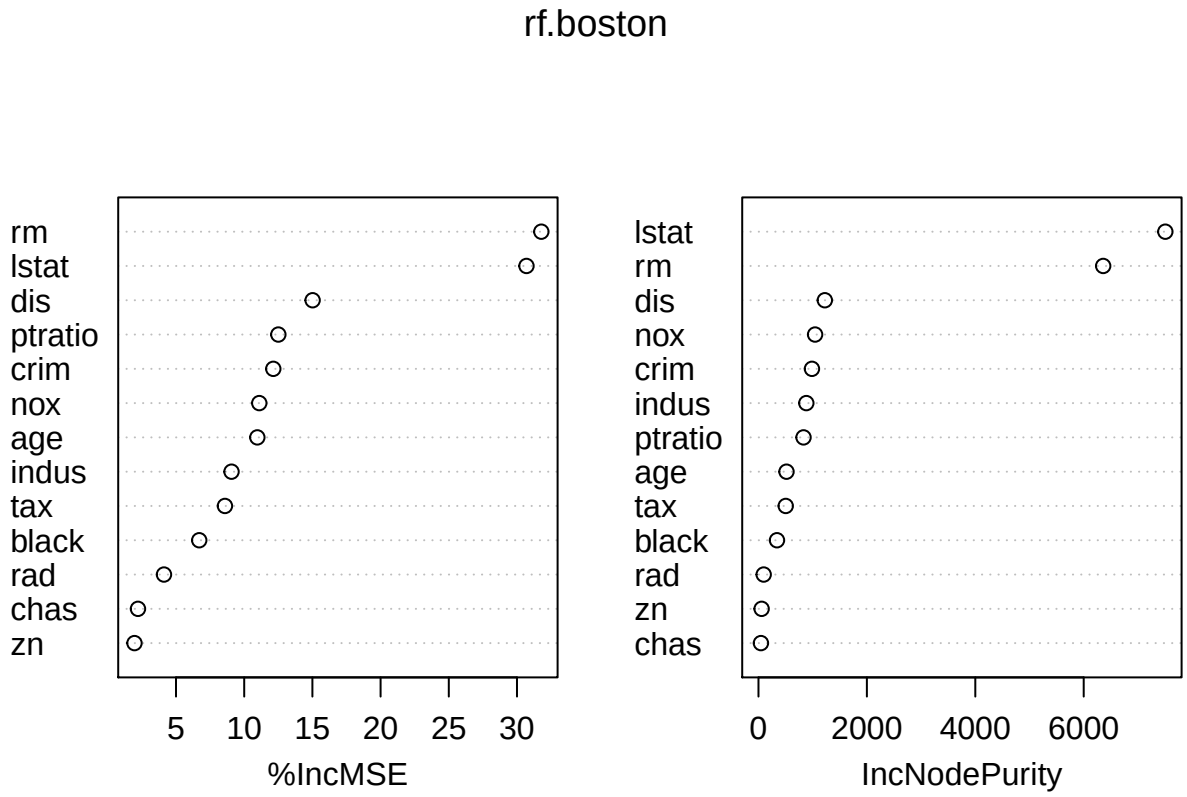
```
## [1] 13.94835
```

```
## [1] 11.66454
```

```
## Importance of each variable:
```

```
##           %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
```

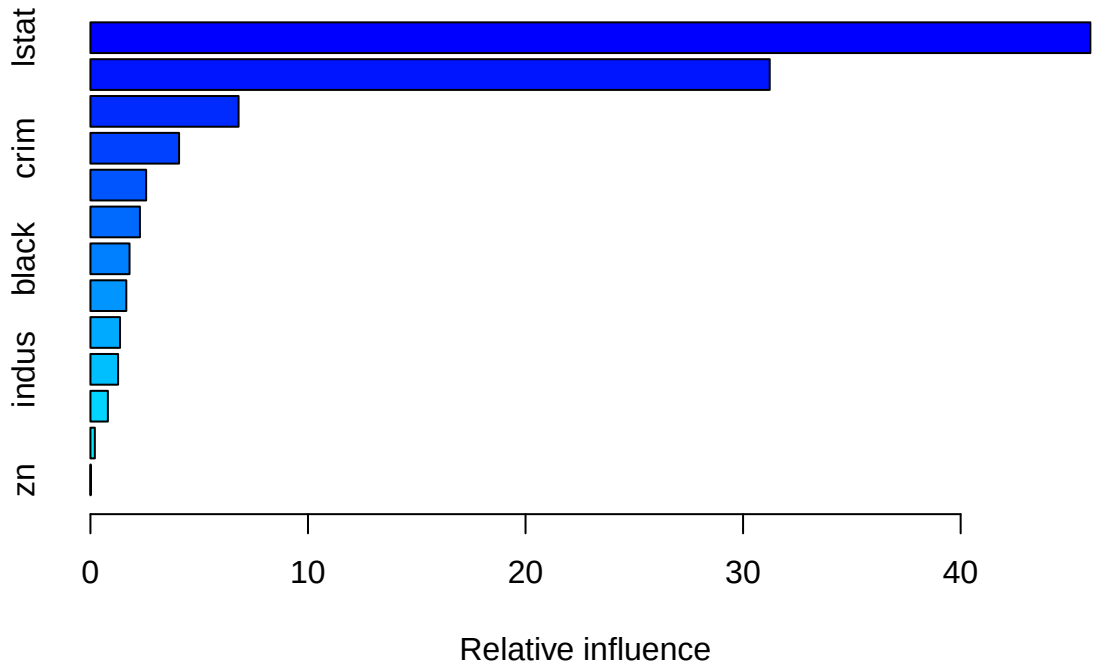
```
## Plot for each importance measures:
```



TAST 4 - Fitting Boosted Regression Tree

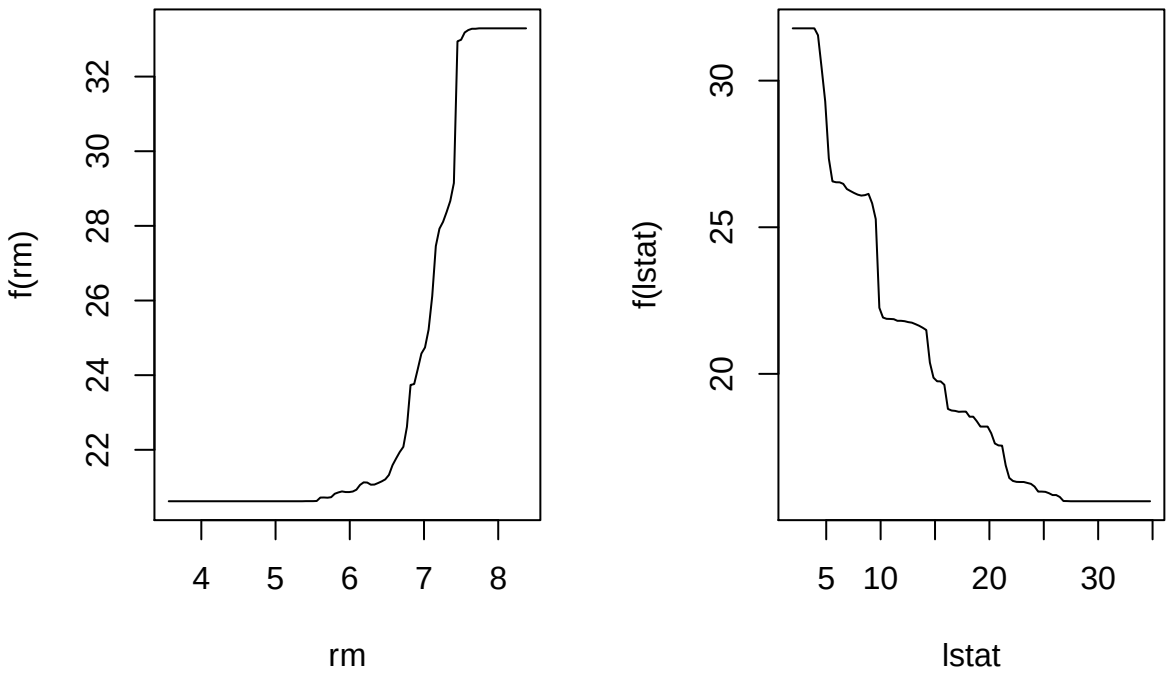
Here we will use gbm function of gbm package to fit boosted regression trees to the Boston data set. we will be using gaussian distribution to perform this task.

```
## Summary on relative influence and relative influence statistics:
```




```
##             var    rel.inf
## 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
```

```
## Produce partial dependence plots for rm and lstat variables:
```




```
## Now using the boosted model to predict medv:
## [1] 11.84434
## Boosting with lambda 0.2:
## [1] 11.51109
```

TASK 5 – Summary:

I started working on the Assignment since 18th of April. I have spent 3 to 4 hours daily for 4 days to complete the task.

Initially, I spent some time in setting up the environment (R, R Studio), then I started this assignment by reading the ISLR Chapter 8 introduction to understand the basics of Decision Tree and how the decision tree is used for Regression and Classification problem. I read and understood the first example Regression tree illustration on Hitters data. It gave me a preliminary understanding of the approach used in Decision Tree. The bagging, random forests and boosting explanations helped me understand how these concepts can be used to construct more powerful prediction models.

After having a fair idea on Tree Based Methods, I replicated Task1, Task2, Task3 and Task4. There were some small modifications I did in order to replicate this examples. Such as, in Task1 I have used Carseats.df as the name for dataframe with Carseats' and High variable so that R doesn't complain of masking variable name. Also, I have used 'cex' attribute to change the size of text for plots so that co-ordinate texts in decision tree doesn't overlap much with each other. Apart from this the tasks were straightforward and I was able to complete it without any hurdle.

To complete this task I have not collaborated with anyone. I have referred the prescribed book.

Reference: I have referred "An Introduction to Statistical Learning with Applications in R"" (ISLR) chapter 8 Tree-Based Method to replicate all the tasks of Programming assignment.
