hw4

Valeriia

27 05 2020

```r
library(MASS)
library(dplyr)
library(data.table)
library(ggplot2)
library(caret)
library(boot)
library(tree)
library(rpart)
library(randomForest)
library(gbm)
data(Boston)
```

```r
bos <- Boston
str(bos)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```r
dim(bos)
```

```
## [1] 506  14
```

```r
sum(is.na(bos))
```

```
## [1] 0
```

```r
summary(bos)
```

```
##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          black
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

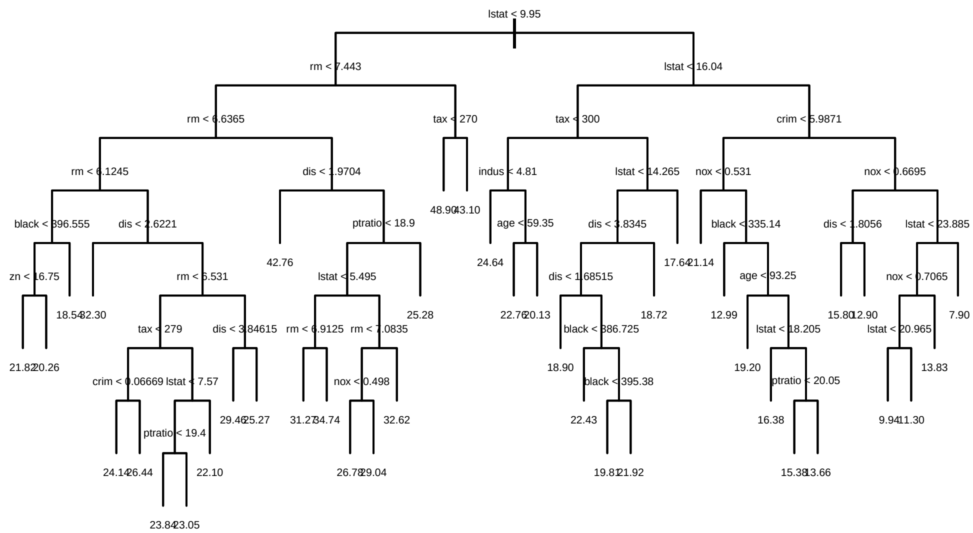
```r
sum(duplicated(bos))
```

```
## [1] 0
```

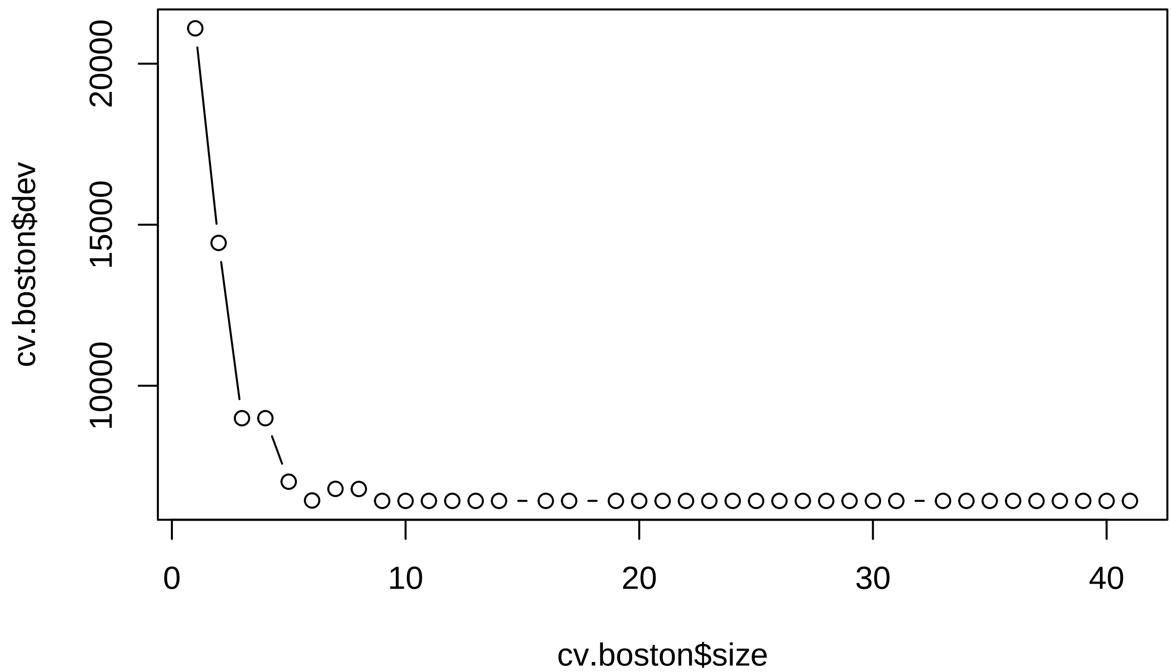
This plot shows the Error and the Number of Trees. We can easily notice that how the Error is dropping as we keep on adding more and more trees and average them.

Although the most complex tree is selected by cross-validation (the lowest error rate corresponds to the most complex tree with 8 leaves), if we wanted to prune the tree, we would do it as follows, using the prune.tree() function.

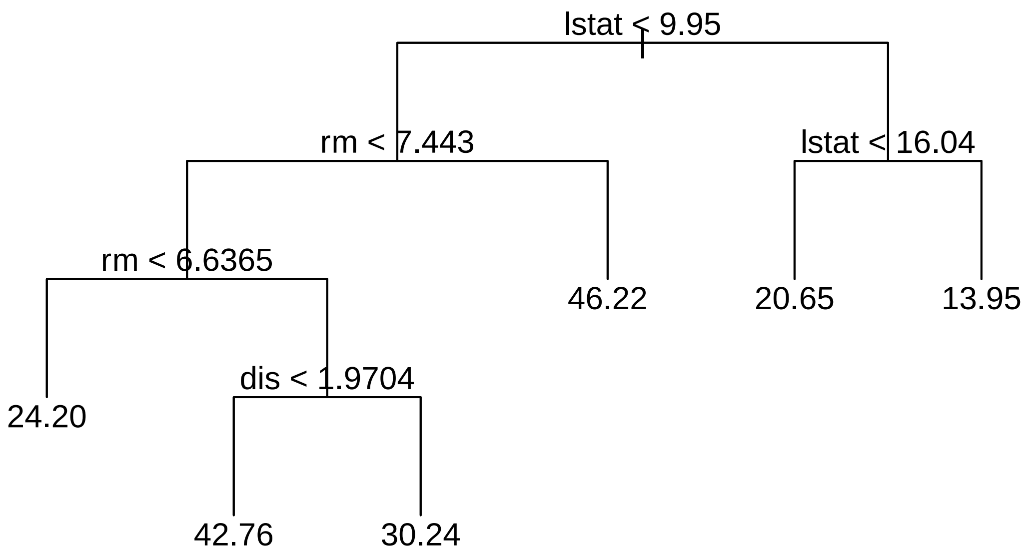
```r
train <- sample(nrow(bos), 0.5*nrow(bos), replace = FALSE)
TrainSet <- bos[train,]
ValidSet <- bos[-train,]
tree.boston = tree(medv~.,Boston ,subset =train,mindev=.0001)
plot(tree.boston,type="u")
text(tree.boston,pretty=0,cex=0.3)
```



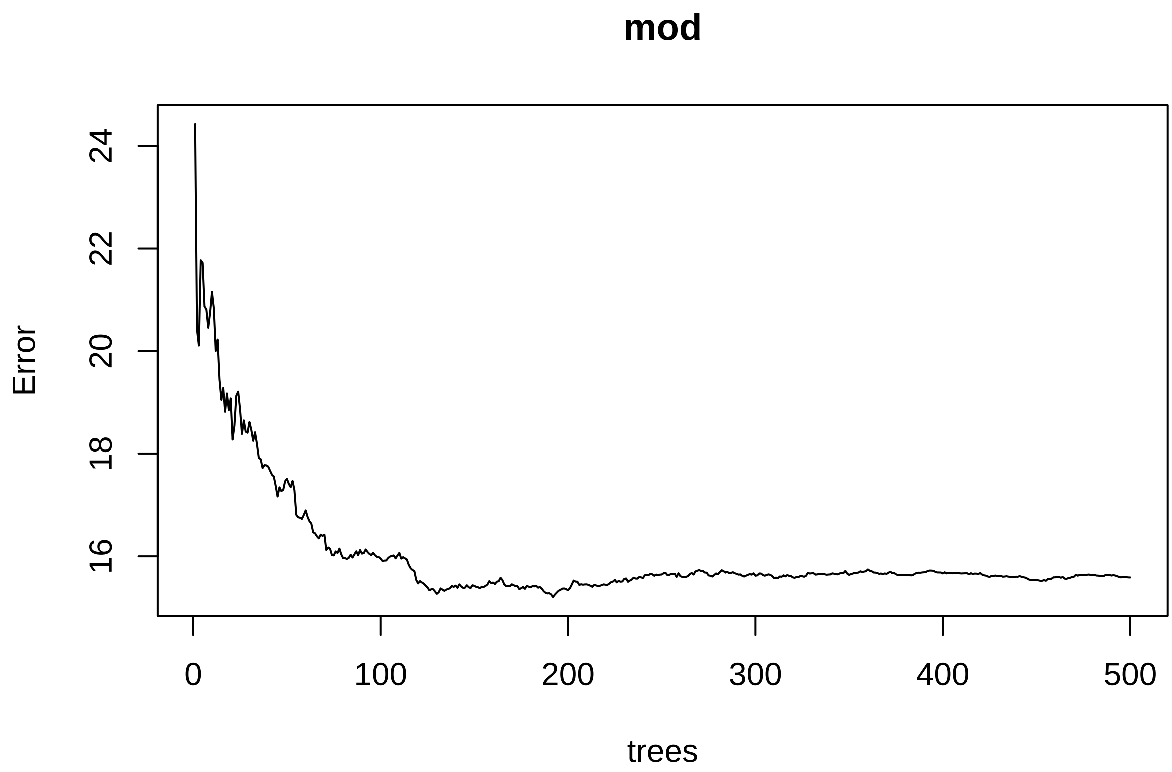

```
cv.boston = cv.tree(tree.boston)
plot(cv.boston$size,cv.boston$dev,type="b")
```




```
prune.boston = prune.tree(tree.boston,best=6)
plot(prune.boston,type="u")
text(prune.boston,pretty=0)
```



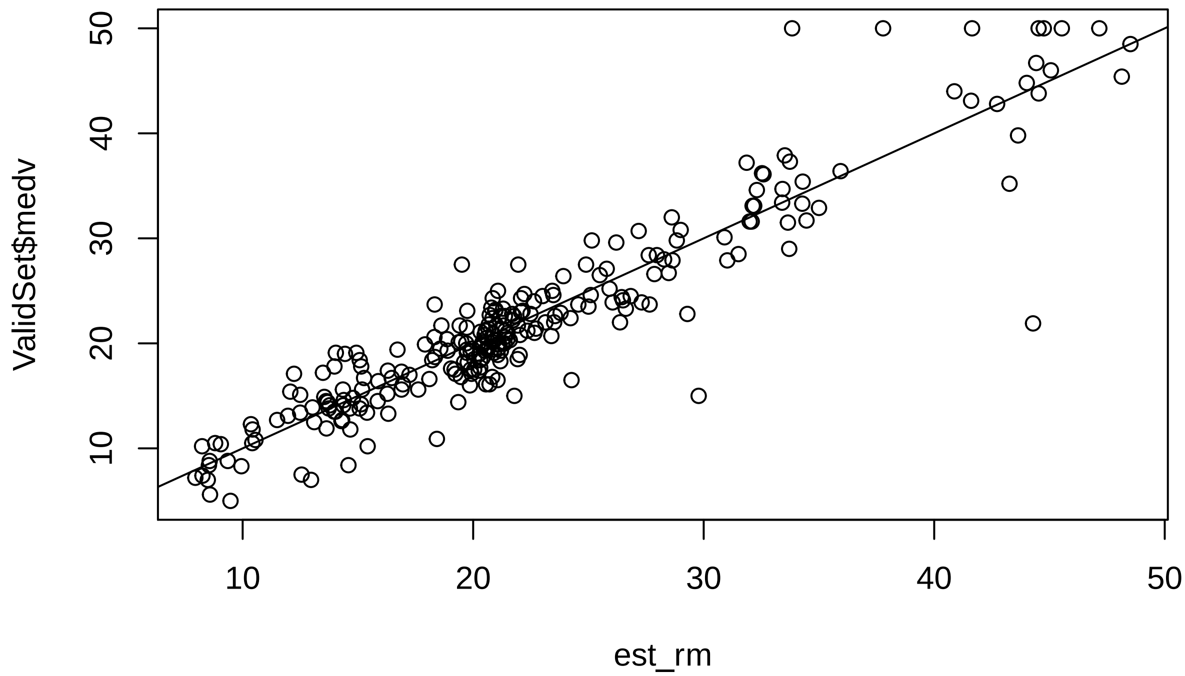

```
mod <- randomForest(medv ~ ., data = TrainSet, mtry = 13, importance = T)
plot(mod)
```




```
mod
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = TrainSet, mtry = 13,      importance = T)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 15.58778
##                     % Var explained: 81.2
```

```
est_rm <- predict(mod, newdata = ValidSet)
plot(est_rm, ValidSet$medv)
abline(0,1)
```




```
mean((est_rm - ValidSet$medv)^2)
```

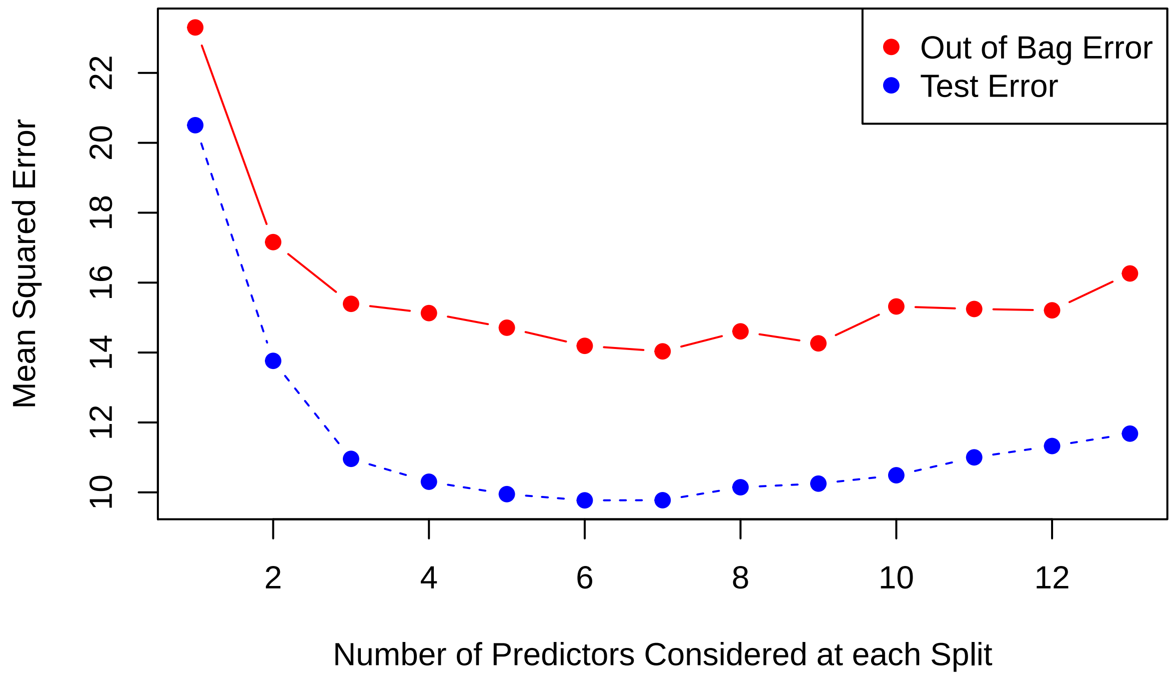
```
## [1] 11.19723
```

The above Random Forest model chose Randomly 4 variables to be considered at each split. We could now try all possible 13 predictors which can be found at each split. Now what we observe is that the Red line is the Out of Bag Error Estimates and the Blue Line is the Error calculated on Test Set. Both curves are quite smooth and the error estimates are somewhat correlated too.

```
oob.err<-double(13)
test.err<-double(13)

#mtry is no of Variables randomly chosen at each split
for(mtry in 1:13)
{
  rf=randomForest(medv ~ . , data = TrainSet,mtry=mtry,ntree=400)
  oob.err[mtry] = rf$mse[400] #Error of all Trees fitted

  pred<-predict(rf,ValidSet) #Predictions on Test Set for each Tree
  test.err[mtry]= with(ValidSet, mean( (medv - pred)^2)) #Mean Squared Test Error

  cat(mtry," ")

}
```

```
## 1   2   3   4   5   6   7   8   9   10   11   12   13
```

```
matplot(1:mtry , cbind(oob.err,test.err), pch=19 , col=c("red","blue"),type="b",ylab="Mean Squared Erro
legend("topright",legend=c("Out of Bag Error","Test Error"),pch=19, col=c("red","blue"))
```

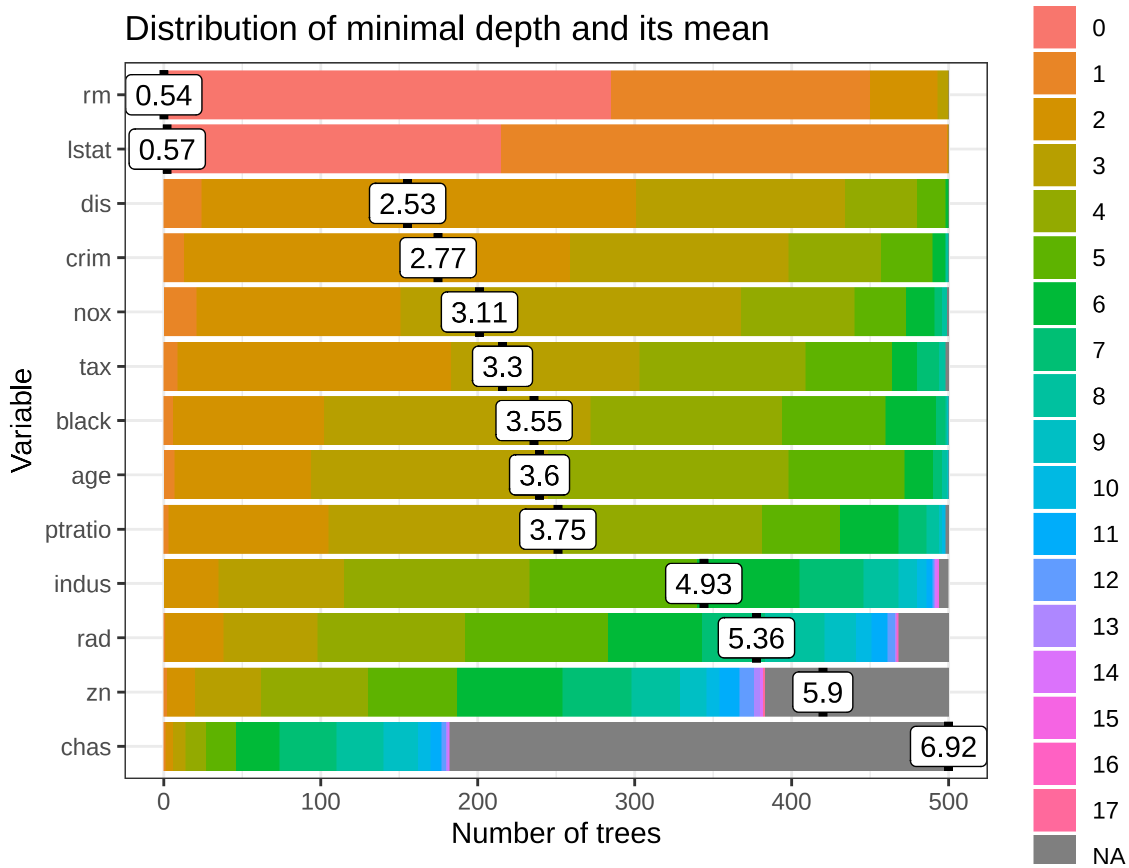


Next, we pass it to the function plot_min_depth_distribution and under default settings obtain obtain a plot of the distribution of minimal depth for top ten variables according to mean minimal depth calculated using top trees (mean_sample = "top_trees"). We could also pass our forest directly to the plotting function but if we want to make more than one plot of the minimal depth distribution is more efficient to pass the min_depth_frame to the plotting function so that it will not be calculated again for each plot (this works similarly for other plotting functions of randomForestExplainer).

```
library(randomForestExplainer)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
min_depth_frame <- min_depth_distribution(mod)
plot_min_depth_distribution(min_depth_frame, mean_sample = "relevant_trees", k = 15)
```



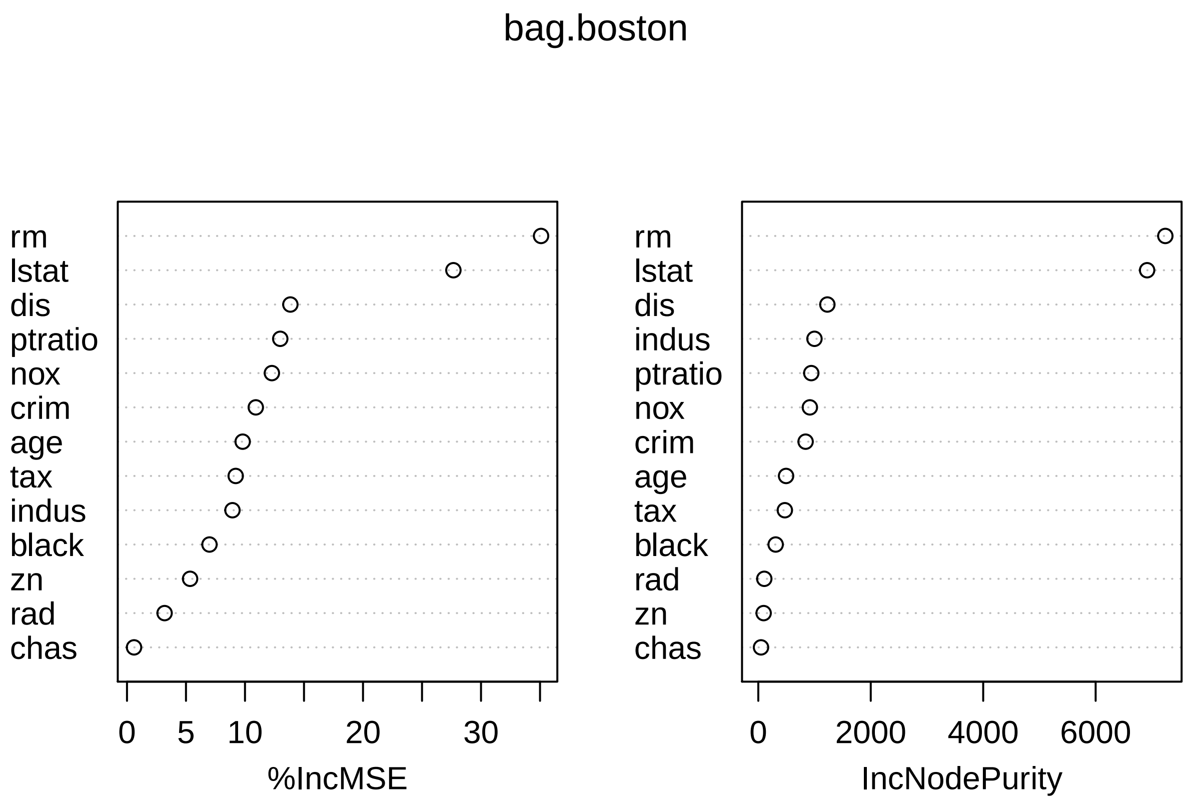

```r
set.seed(3)
bag.boston=randomForest(medv~.,data=TrainSet,mtry=7, importance=TRUE)
yhat.bag = predict(bag.boston,newdata=ValidSet)
mean((yhat.bag-ValidSet$medv)^2)
```

```
## [1] 9.893178
```

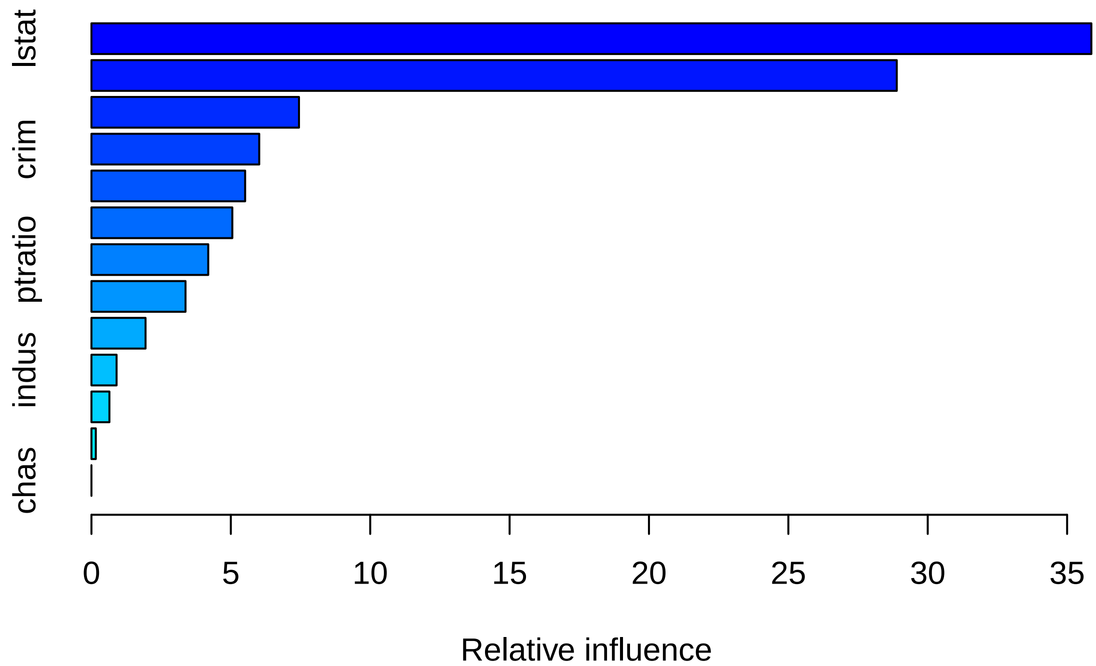
```r
importance(bag.boston)
```

```
##            %IncMSE IncNodePurity
## crim    10.9157651     841.38319
## zn       5.3476905      94.76382
## indus    8.9363977     999.01262
## chas     0.5976062      48.16985
## nox     12.2773757     917.12199
## rm      35.0874786    7239.47264
## age      9.8025949     493.86970
## dis     13.8456981    1228.71351
## rad      3.1859068     106.04755
## tax      9.2126296     471.87662
## ptratio 12.9869511     942.11408
## black    6.9908896     309.09949
## lstat   27.6587545    6915.68795
```

```r
varImpPlot(bag.boston)
```




```r
set.seed(1)
boost.boston=gbm(medv~.,data=TrainSet,distribution="gaussian",n.trees=5000,interaction.depth=4)
summary(boost.boston)
```




```
##             var      rel.inf
## lstat     lstat 3.586946e+01
## rm           rm 2.888933e+01
## dis         dis 7.445296e+00
## crim       crim 6.018850e+00
## nox         nox 5.514172e+00
## age         age 5.053677e+00
## ptratio ptratio 4.189652e+00
## black     black 3.374426e+00
## tax         tax 1.941365e+00
## indus     indus 9.016799e-01
## rad         rad 6.449322e-01
## zn           zn 1.571651e-01
## chas       chas 6.955336e-07
```

```
boost.boston=gbm(medv~.,data=TrainSet,distribution="gaussian",n.trees=5000,interaction.depth=4,shrinkage
yhat.boost=predict(boost.boston,newdata=ValidSet,n.trees=5000)
mean((yhat.boost-ValidSet$medv)^2)
```

```
## [1] 10.76961
```

```
n.trees = seq(from=100 ,to=10000, by=100) #no of trees-a vector of 100 values

#Generating a Prediction matrix for each Tree
```

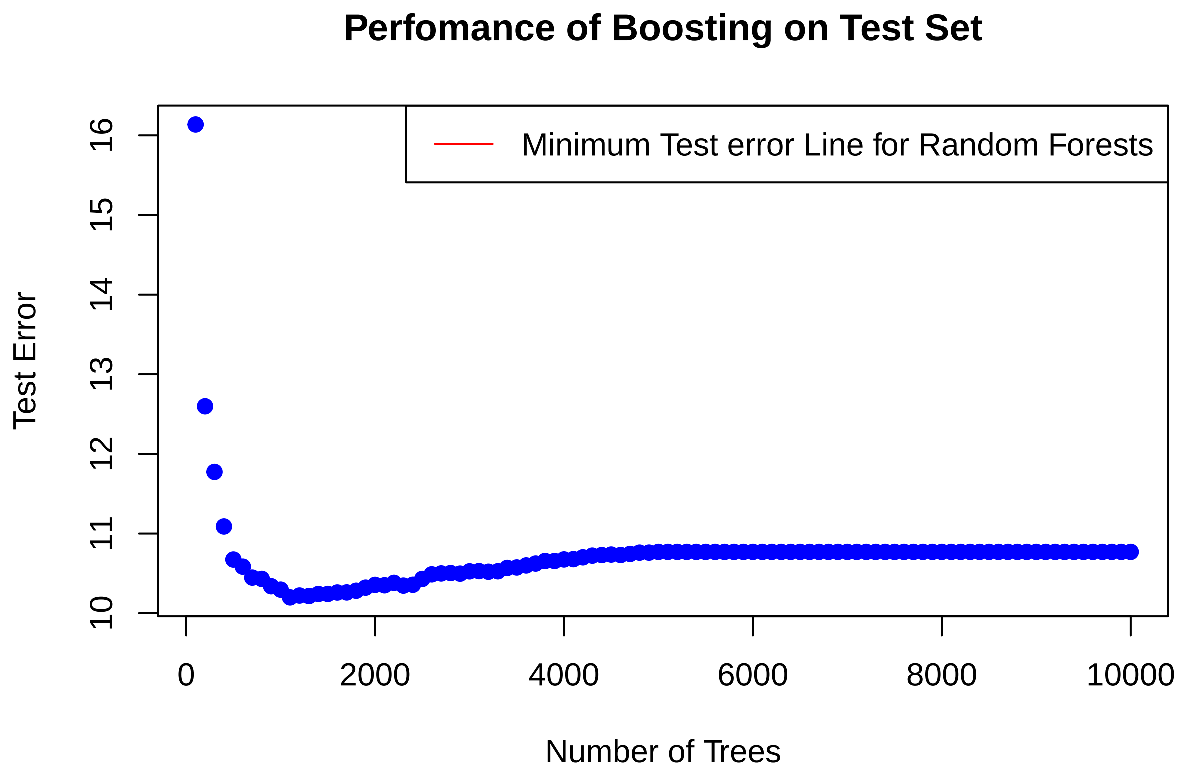
```r
predmatrix<-predict(boost.boston,Boston[-train,],n.trees = n.trees)
dim(predmatrix) #dimentions of the Prediction Matrix
```

```
## [1] 253 100
```

```r
#Calculating The Mean squared Test Error
test.error<-with(Boston[-train,],apply( (predmatrix-medv)^2,2,mean))
head(test.error) #contains the Mean squared test error for each of the 100 trees averaged
```

```
##      100      200      300      400      500      600
## 16.13655 12.59777 11.77399 11.08868 10.67346 10.58391
```

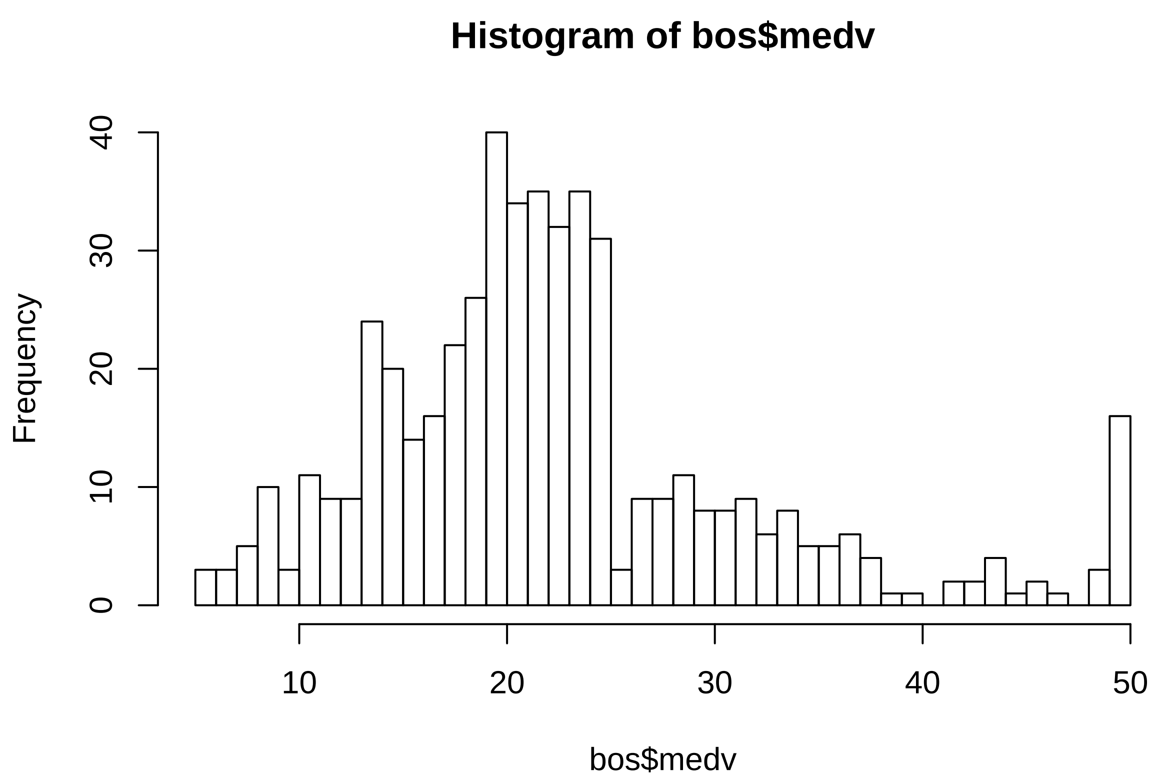
```r
#Plotting the test error vs number of trees

plot(n.trees , test.error , pch=19,col="blue",xlab="Number of Trees",ylab="Test Error", main = "Perfomar

#adding the RandomForests Minimum Error line trained on same data and similar parameters
abline(h = min(test.err),col="red") #test.err is the test error of a Random forest fitted on same data
legend("topright",c("Minimum Test error Line for Random Forests"),col="red",lty=1,lwd=1)
```




```r
boost.boston=gbm(medv~.,data=TrainSet,distribution="gaussian",n.trees=1000,interaction.depth=4,shrinkage
yhat.boost=predict(boost.boston,newdata=ValidSet,n.trees=1000)
mean((yhat.boost-ValidSet$medv)^2)
```

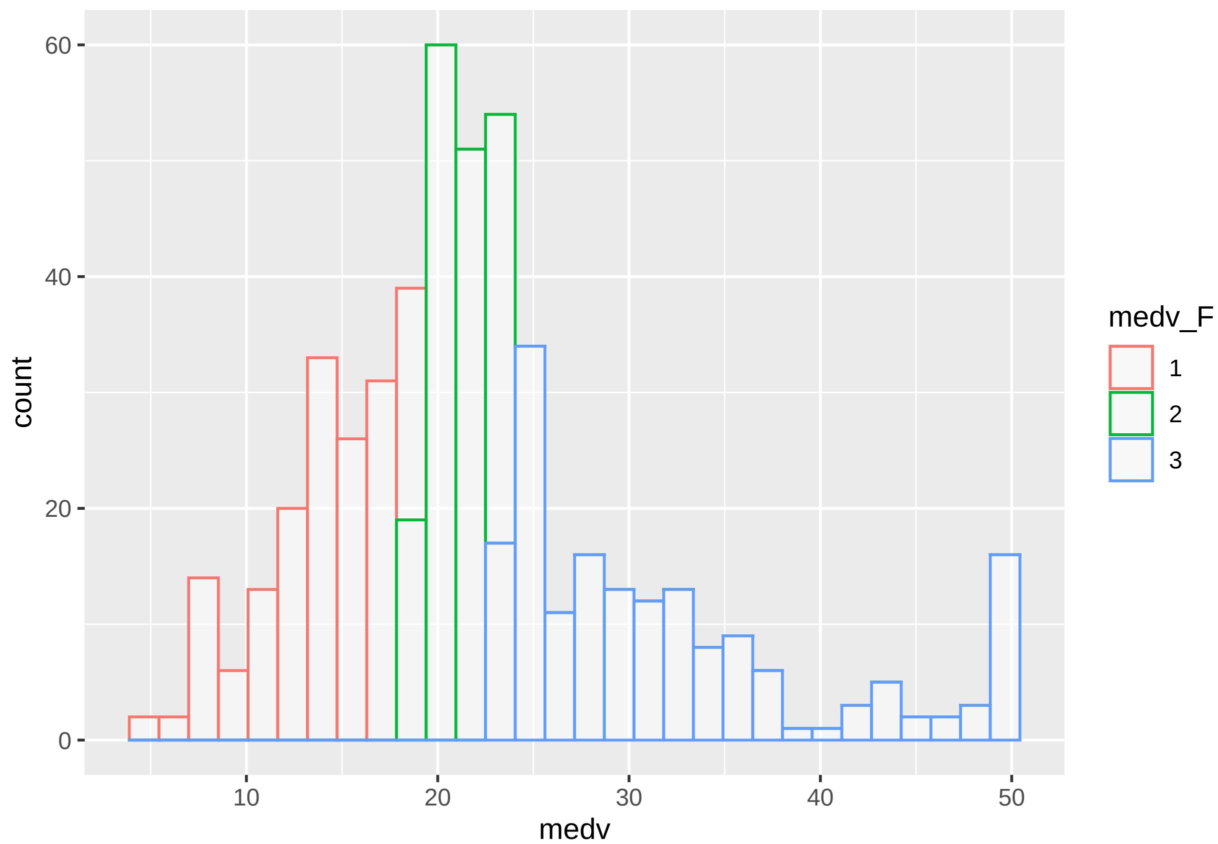
```
## [1] 9.810932
```

```r
hist(bos$medv, breaks = 50)
```




```r
bos <- data.table(bos)
bos[,medv_F:= cut(medv,c(0,quantile(bos$medv, 0.33),quantile(bos$medv, 0.66),
                         max(bos$medv)),labels=c("1","2","3"))]
bos %>%
  ggplot(aes(medv, color = medv_F))+
  geom_histogram(fill="white", alpha=0.5)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
bos <- bos[,-14]
sampsize <-  ceiling(0.6325*nrow(bos[-train]))
mtry <- floor(sqrt(ncol(bos)))
set.seed(3)
ntrees <- 50
rf <- randomForest(medv_F ~ ., data=bos,
                   subset = train,
                   ntree = ntrees,
                   mtry = mtry,
                   sampsize = sampsize,
                   importance = TRUE,
                   do.trace = ntrees/25)
```

```
## ntree      OOB      1      2      3
##     2:  37.37% 33.96% 46.03% 32.43%
##     4:  35.90% 23.88% 54.55% 28.89%
##     6:  34.29% 22.54% 49.38% 30.11%
##     8:  34.66% 21.92% 59.04% 23.16%
##    10:  33.60% 18.92% 54.22% 27.08%
##    12:  34.39% 22.97% 54.22% 26.04%
##    14:  30.83% 24.32% 44.58% 23.96%
##    16:  28.46% 17.57% 46.99% 20.83%
##    18:  27.67% 18.92% 42.17% 21.88%
##    20:  26.88% 20.27% 39.76% 20.83%
##    22:  26.88% 17.57% 40.96% 21.88%
##    24:  26.88% 18.92% 38.55% 22.92%
```

```
##    26:  27.27% 17.57% 38.55% 25.00%
##    28:  26.88% 17.57% 40.96% 21.88%
##    30:  27.27% 18.92% 40.96% 21.88%
##    32:  25.30% 16.22% 36.14% 22.92%
##    34:  26.09% 18.92% 36.14% 22.92%
##    36:  26.09% 18.92% 36.14% 22.92%
##    38:  26.48% 18.92% 37.35% 22.92%
##    40:  25.69% 17.57% 37.35% 21.88%
##    42:  26.88% 17.57% 37.35% 25.00%
##    44:  26.48% 18.92% 37.35% 22.92%
##    46:  25.30% 18.92% 33.73% 22.92%
##    48:  24.51% 17.57% 33.73% 21.88%
##    50:  25.69% 17.57% 34.94% 23.96%
```

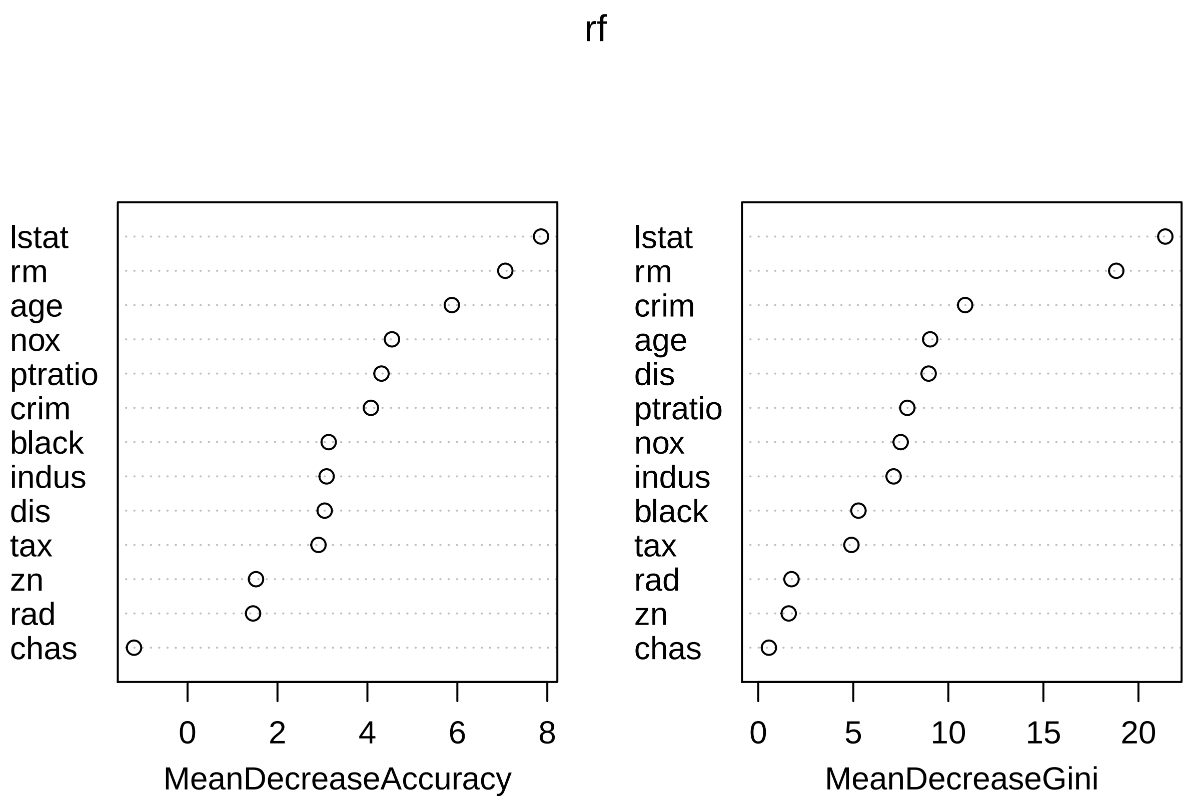
```r
set.seed(3)
rf <- randomForest(medv_F ~ ., data=bos,
                   subset = train,
                   ntree = 30,
                   mtry = mtry,
                   sampsize = sampsize,
                   importance = TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = medv_F ~ ., data = bos, ntree = 30, mtry = mtry,      sampsize = sampsize, i
##                Type of random forest: classification
##                      Number of trees: 30
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 27.27%
## Confusion matrix:
##    1  2  3 class.error
## 1 60 14  0   0.1891892
## 2 11 49 23   0.4096386
## 3  1 20 75   0.2187500
```

```r
pred <- predict(rf, newdata = bos[-train,])
mean(pred!=bos$medv_F[-train])
```

```
## [1] 0.2173913
```

```r
varImpPlot(rf)
```


rf
lstat
rm
age
nox
ptratio
crim
black
indus
dis
tax
zn
rad
chas
0
2
4
6
8
MeanDecreaseAccuracy
lstat
rm
crim
age
dis
ptratio
nox
indus
black
tax
rad
zn
chas
0
5
10
15
20
MeanDecreaseGini