# HW2.4_Mary_Futey

Mary Futey

5/8/2020

```
library(tree)
library(rpart)
library(randomForest)
```

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

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(MASS)
library(gbm)
```

```
## Loaded gbm 2.1.5
```

### Load dataset

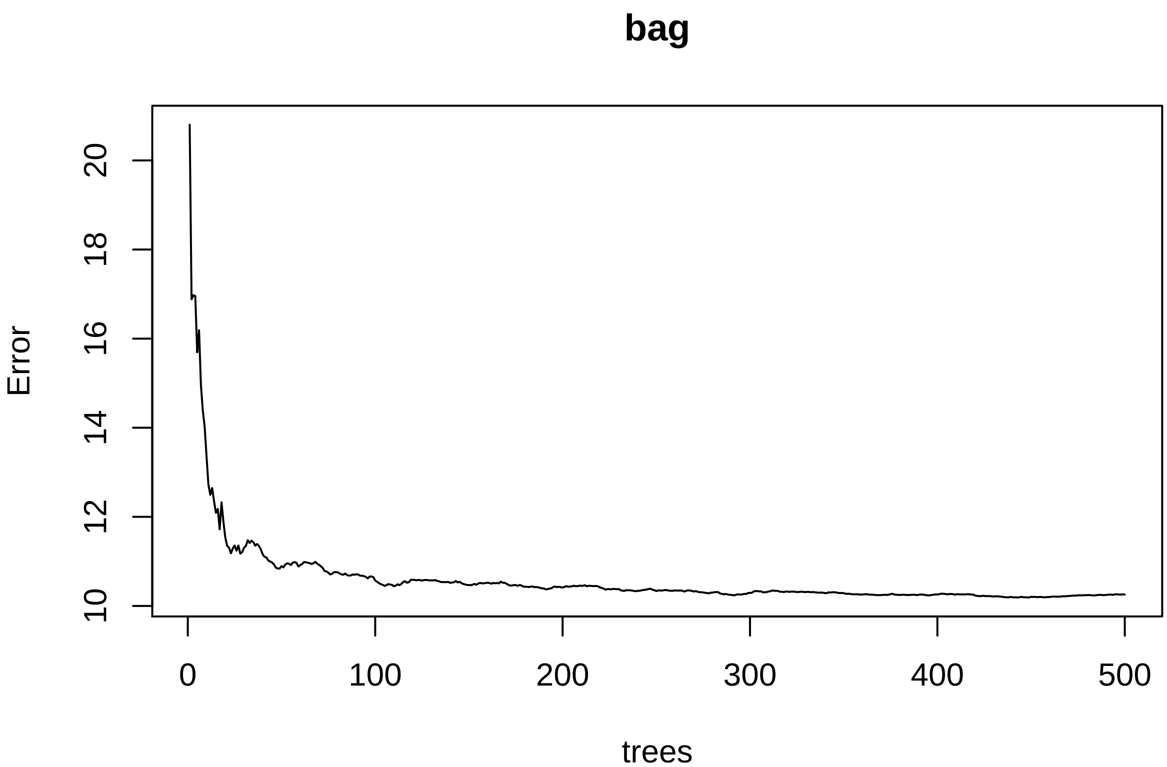
```
boston <- Boston
str(boston)
```

```
## '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 ...
```

### bagging method

```
set.seed(1)
# 50% samples are test, train
train <- sample(1:nrow(boston), nrow(boston)/2)

# 14 variables, 1 is response, others predictors
bag <- randomForest(lstat ~ ., data = boston,
                    subset = train,
```

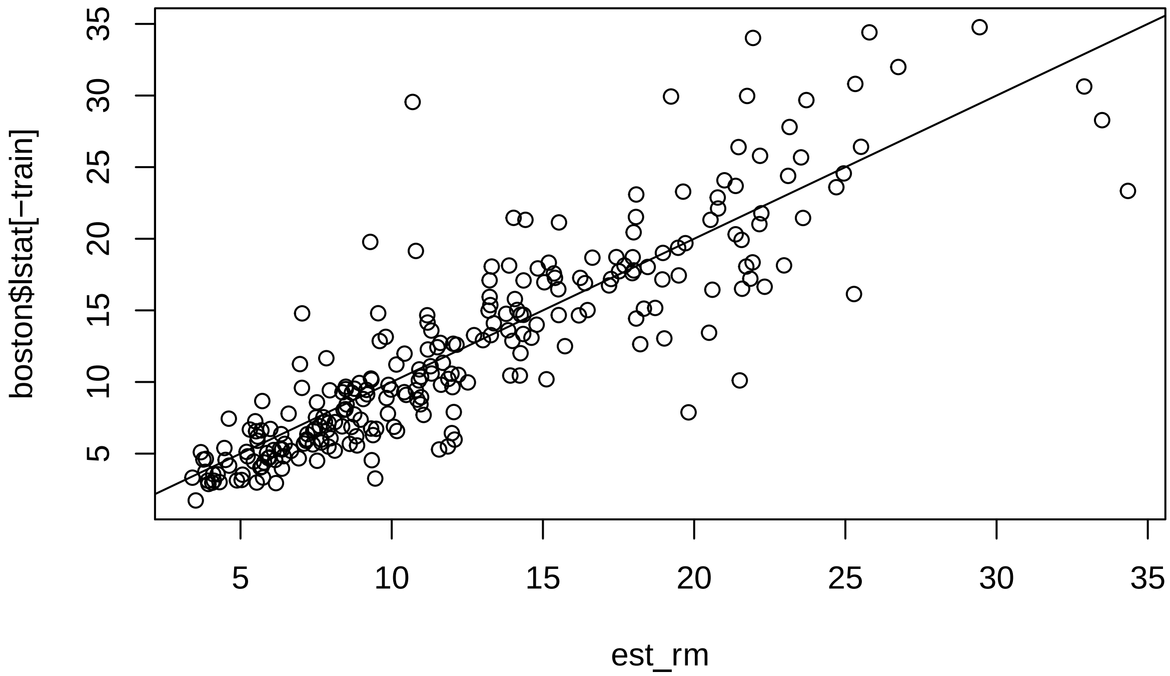
```r
                     # n - number of rows (obvs), m - number of features
                     #mtry: how many features to try (13 default =# of predictors)
                     mtry = 13,
                     #imp. of each predictor, how much it influences result and error
                     importance = T)
# plot error vs trees
plot(bag)
```




```r
#summary contains more (complicated) information
#type is regression (if response is numerical: regression, if factor: classification)
# % variance explained is similar to R^2
bag
```

```
##
## Call:
##  randomForest(formula = lstat ~ ., data = boston, mtry = 13, importance = T,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 10.25546
##                     % Var explained: 79.11
```

```r
# 1st argument is the model, 2nd test data
est_rm <- predict(bag, newdata = boston[-train,])

# x-axis: est, y-axis: real, abline is if they are equal (0: intercept, 1: slope)
plot(est_rm, boston$lstat[-train]); abline(0,1)
```




```r
#test error is higher (see next plot)
mean((est_rm -boston$lstat[-train])^2)
```

```
## [1] 12.18186
```

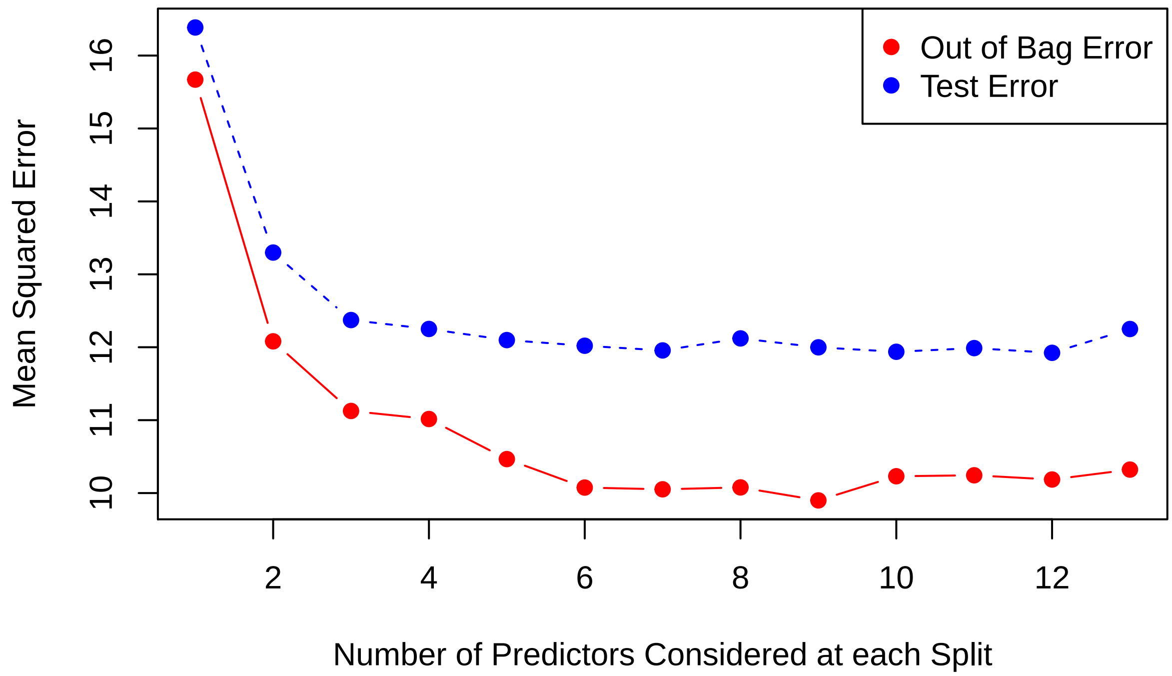
## Plot test and out of bag error

```r
# 13 times (13 predictors)
oob.err=double(13)
test.err=double(13)

#mtry is num of variables randomly chosen at each split
for(mtry in 1:13)
{
  rf=randomForest(lstat ~ . , data = boston ,
                  subset = train,
                  mtry=mtry,
                  ntree=400)
  oob.err[mtry] = rf$mse[400] #error of all trees fitted

  pred<-predict(rf,boston[-train,]) #predictions on test set for each tree
  test.err[mtry]= with(boston[-train,],
                       mean( (lstat - pred)^2)) #Mean Squared Test error

  cat(mtry," ") #printing the output to the console

}
```

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

```r
matplot(1:mtry ,
        cbind(oob.err,test.err),
```

```r
         pch=19 ,
         col=c("red","blue"),
         type="b",
         ylab="Mean Squared Error",
         xlab="Number of Predictors Considered at each Split")
legend("topright",
        legend=c("Out of Bag Error","Test Error"),
        pch=19,
        col=c("red","blue"))
```




### random forest instead of bagging

```r
set.seed(1)

#check fewer predictors
rf <- randomForest(lstat ~ ., data = boston,
                    subset = train,
                    mtry = 6,
                    ntree = 25,
                    importance = T)
est_lstat <- predict(rf, newdata = boston[-train,])
# better than for bagging
mean((est_lstat - boston$lstat[-train])^2)
```

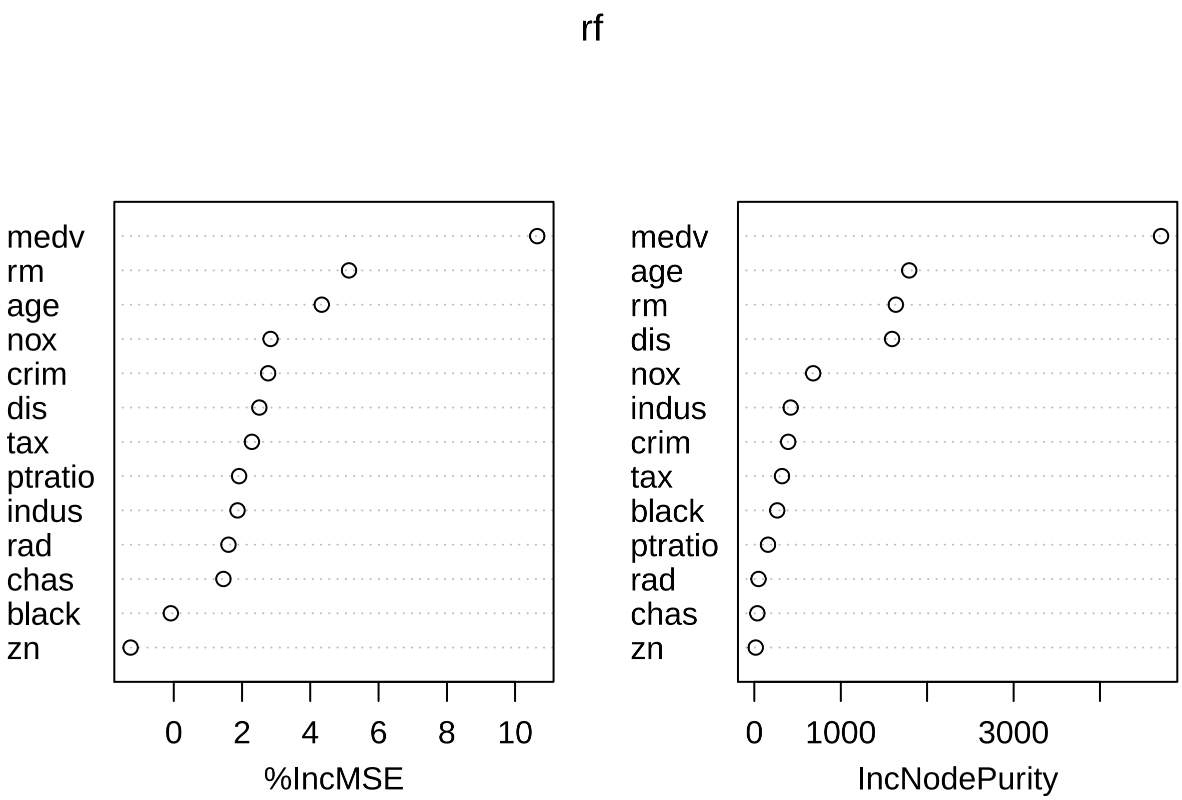
```
## [1] 11.60747
```

```r
# higher InceMSE and purity - better
importance(rf)
```

```
##              %IncMSE IncNodePurity
## crim      2.76523046     392.03112
## zn       -1.26409507      15.68211
```

```
## indus     1.86865080     420.19691
## chas      1.45593949      34.95599
## nox       2.83622316     681.01621
## rm        5.13343886    1637.37276
## age       4.33505499    1792.25523
## dis       2.50833171    1594.38678
## rad       1.60347918      48.14158
## tax       2.28907869     320.57520
## ptratio   1.91406701     158.11352
## black    -0.08608235     264.86238
## medv     10.64906524    4706.48603
```

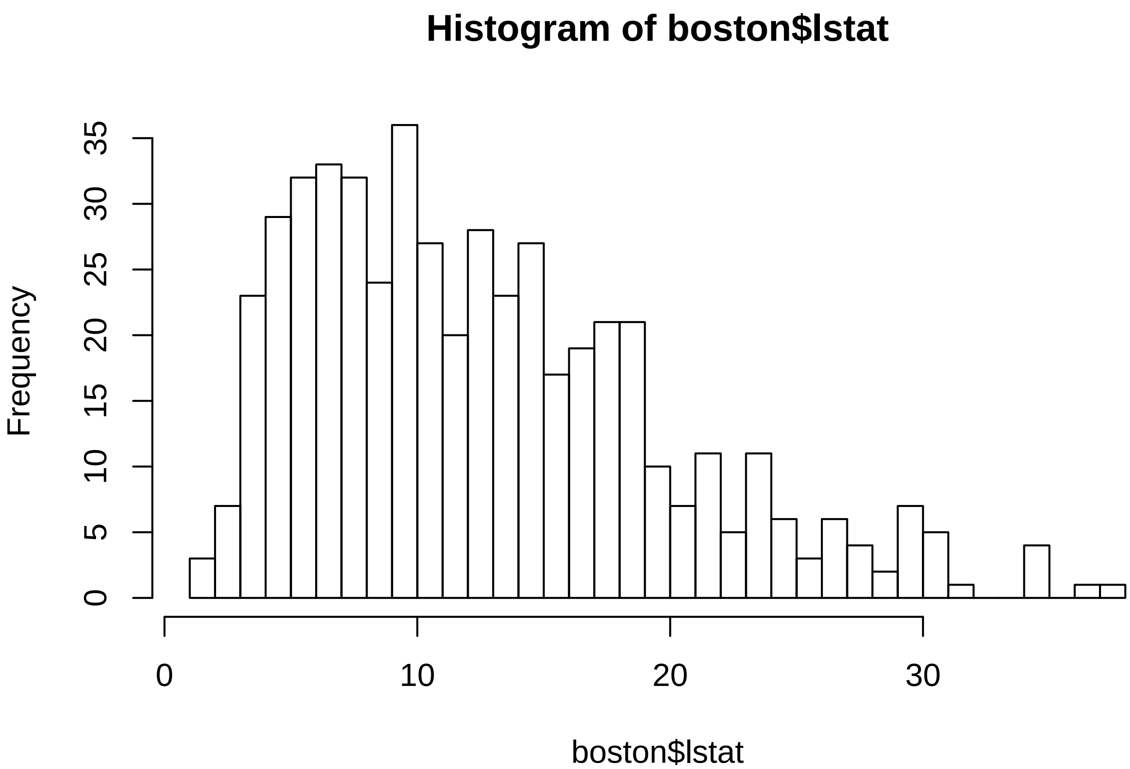
```
# plots results of importance(), which are not sorted
varImpPlot(rf)
```




```
summary(boston$lstat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.73    6.95   11.36   12.65   16.95   37.97
```

```
# response can be split into 2-4 classes
# should have approx same number of observations in each class
hist(boston$lstat, breaks = 45)
```




```
# will work with two classes: above and below 12

boston$lstat <- as.factor(ifelse(boston$lstat > 12, 1, 2))
table(boston$lstat)
```

```
##
##   1   2
## 240 266
```

```
# sample size, n of N
ceiling(0.632*nrow(boston[-train,]))
```

```
## [1] 160
```

```
# number of predictors to use (mtry), m of M
floor(sqrt(ncol(boston)))
```

```
## [1] 3
```

```
set.seed(1)
#response is a factor so rf knows to do classification
rf_class <- randomForest(lstat ~ ., data = boston,
                         subset = train,
                         mtry = 3,
                         sampsize = 160,
                         importance = T)

rf_class
```

```
##
## Call:
##  randomForest(formula = lstat ~ ., data = boston, mtry = 3, sampsize = 160,      importance = T, subs
```

```
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 15.02%
## Confusion matrix:
##     1   2 class.error
## 1 106  20   0.1587302
## 2  18 109   0.1417323
```

```
#test
est_lstat <- predict(rf_class, newdata = boston[-train,])

mean(est_lstat != boston$lstat[-train])
```

```
## [1] 0.1067194
```

```
# compare to OBB error rate, bad if test errror is larger than train
# this is <15% from training above
```

```
# trace option, tells function to caluclate result at intermediate steps
# set ntrees, use in do.trace, so gives result every 50 trees

set.seed(1)
ntrees <- 500
rf_class <- randomForest(lstat ~ ., data = boston,
                         subset = train,
                         ntree = ntrees,
                         mtry = 3,
                         sampsize = 160,
                         importance = T,
                         do.trace = ntrees/10)
```

```
## ntree      OOB      1      2
##    50:  16.21% 15.08% 17.32%
##   100:  15.42% 15.08% 15.75%
##   150:  15.02% 15.08% 14.96%
##   200:  15.42% 15.87% 14.96%
##   250:  15.02% 15.87% 14.17%
##   300:  15.02% 15.87% 14.17%
##   350:  15.02% 15.87% 14.17%
##   400:  14.62% 15.87% 13.39%
##   450:  14.62% 15.87% 13.39%
##   500:  15.02% 15.87% 14.17%
```

```
# can see from OBB, error for 1st and 2nd class for every 50 trees
# over feeding: adding trees does not improve the model

rf_class
```

```
##
## Call:
##  randomForest(formula = lstat ~ ., data = boston, ntree = ntrees,      mtry = 3, sampsize = 160, impo
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
```

```
##         OOB estimate of  error rate: 15.02%
## Confusion matrix:
##     1   2 class.error
## 1 106  20   0.1587302
## 2  18 109   0.1417323
```

```r
# use less trees in forest
set.seed(1)
ntrees <- 50
rf_class <- randomForest(lstat ~ ., data = boston,
                         subset = train,
                         ntree = ntrees,
                         mtry = 3,
                          sampsize = 160,
                         importance = T,
                         do.trace = ntrees/25)
```

```
## ntree      OOB      1      2
##     2:  23.12% 21.05% 25.00%
##     4:  18.30% 17.09% 19.49%
##     6:  18.07% 18.40% 17.74%
##     8:  16.60% 15.08% 18.11%
##    10:  15.02% 11.90% 18.11%
##    12:  16.60% 15.08% 18.11%
##    14:  16.60% 15.08% 18.11%
##    16:  16.60% 14.29% 18.90%
##    18:  15.02% 12.70% 17.32%
##    20:  16.21% 15.87% 16.54%
##    22:  16.60% 15.87% 17.32%
##    24:  15.81% 15.08% 16.54%
##    26:  16.60% 14.29% 18.90%
##    28:  15.42% 13.49% 17.32%
##    30:  13.83% 13.49% 14.17%
##    32:  15.02% 13.49% 16.54%
##    34:  14.23% 12.70% 15.75%
##    36:  16.21% 15.08% 17.32%
##    38:  15.42% 15.08% 15.75%
##    40:  16.21% 15.87% 16.54%
##    42:  15.81% 14.29% 17.32%
##    44:  15.02% 13.49% 16.54%
##    46:  15.42% 14.29% 16.54%
##    48:  16.21% 15.08% 17.32%
##    50:  16.21% 15.08% 17.32%
```

```r
rf_class
```

```
##
## Call:
##  randomForest(formula = lstat ~ ., data = boston, ntree = ntrees,      mtry = 3, sampsize = 160, impo
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 16.21%
## Confusion matrix:
##     1   2 class.error
```

```
## 1 107  19   0.1507937
## 2  22 105   0.1732283
```

```r
# can see from OBB, error for 1st and 2nd class for every 25 trees
# calculate for every 2 trees
# can see 30 trees error is lower
```

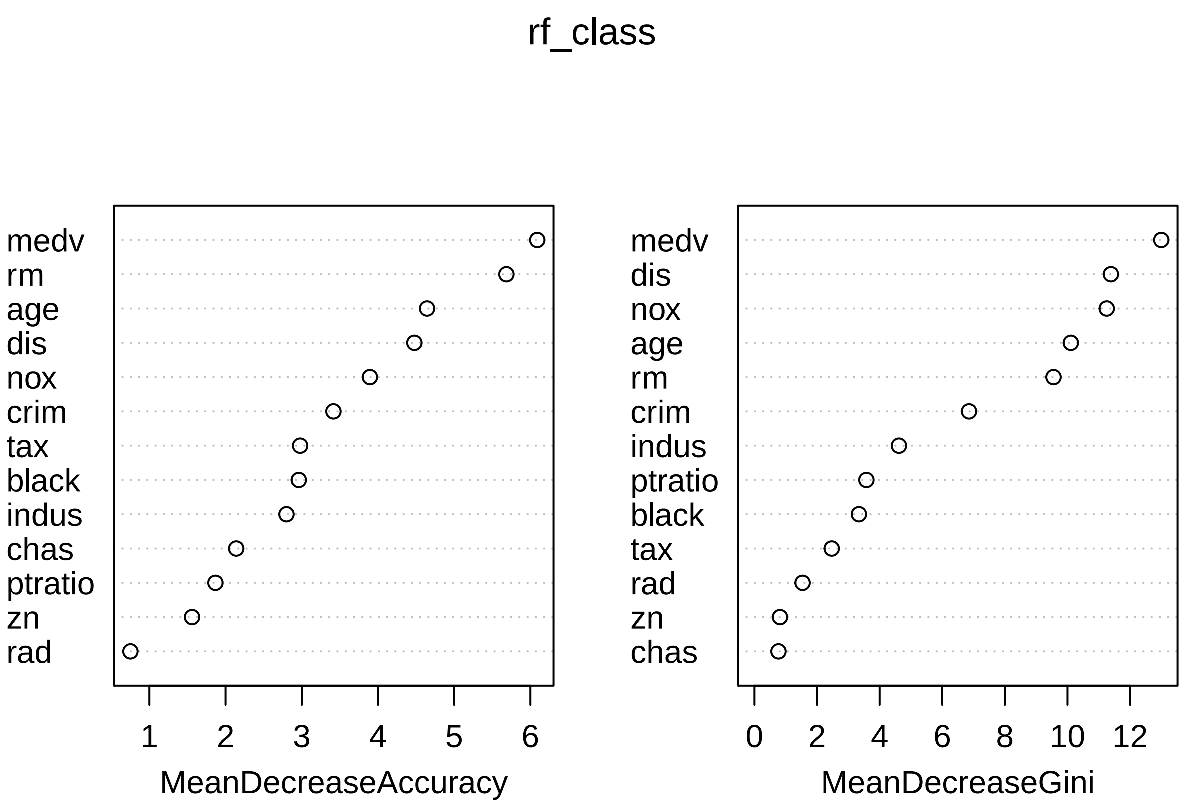
```r
# take least number of trees with best result = 30 trees

set.seed(1)
ntrees <- 30
rf_class <- randomForest(lstat ~ ., data = boston,
                         subset = train,
                         ntree = ntrees,
                         mtry = 3,
                         sampsize = 160,
                         importance = T)
rf_class
```

```
##
## Call:
##  randomForest(formula = lstat ~ ., data = boston, ntree = ntrees,      mtry = 3, sampsize = 160, impo
##                Type of random forest: classification
##                      Number of trees: 30
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.83%
## Confusion matrix:
##     1   2 class.error
## 1 109  17   0.1349206
## 2  18 109   0.1417323
```

```r
# calcluate error on test: less than original
est_lstat <- predict(rf_class, newdata = boston[-train,])
mean(est_lstat != boston$lstat[-train])
```

```
## [1] 0.1027668
```

```r
# check importance of features
# mean decrease and gini index:

varImpPlot(rf_class)
```

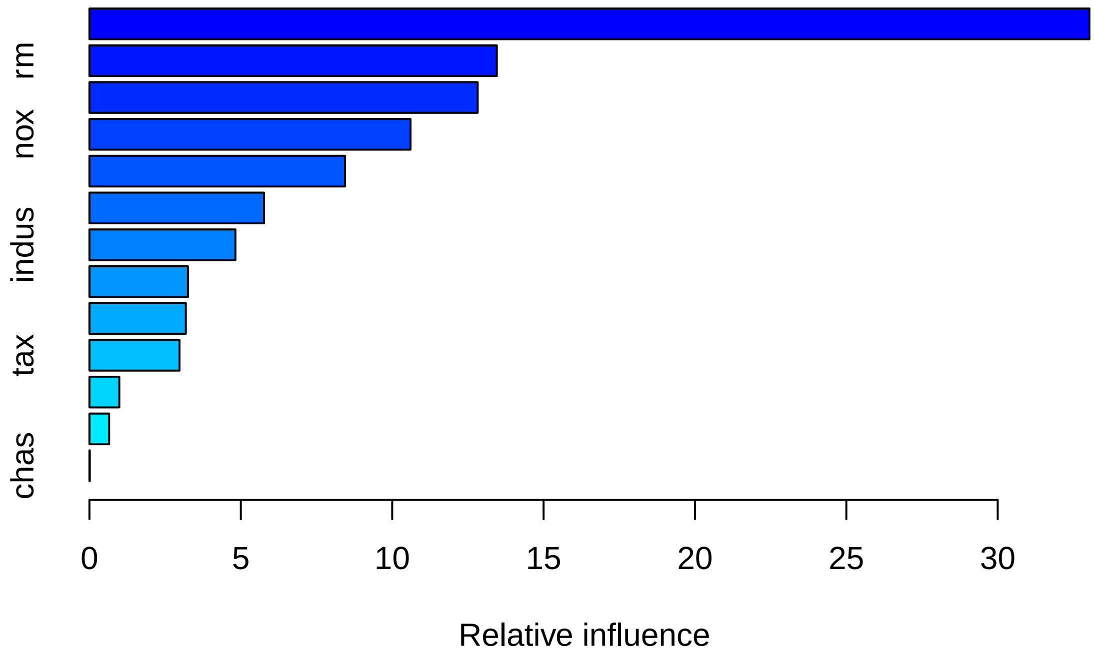



### gradient boosting machine

```r
# need to use integers(one or zero)

# working with trees one by one (not parallel)
# takes errors and adds to the next model
# so each step optimizes result based on errors


boston$lstat <- as.integer(ifelse(boston$lstat == 1, 1, 0))
table(boston$lstat)
```

```
##
##   0   1
## 266 240
```

```r
set.seed(1)
ntrees <- 5000
boost <- gbm(lstat ~ ., data = boston[train,],
             distribution = "bernoulli",
             n.trees = ntrees,
             interaction.depth = 4,
             shrinkage = 0.01)
# medv and rm highest influence - not too suprising
summary(boost)
```




```
##             var     rel.inf
## medv       medv 33.02802587
## rm           rm 13.45556943
## age         age 12.82309936
## nox         nox 10.60428750
## dis         dis  8.44191824
## crim       crim  5.76735270
## indus     indus  4.82060209
## ptratio ptratio  3.25408535
## black     black  3.18295658
## tax         tax  2.97383849
## rad         rad  0.98657335
## zn           zn  0.64974769
## chas       chas  0.01194334
```

```
# test
est_lstat <- predict(boost, newdata = boston[-train,],
                     n.trees = ntrees,
                     type = "response") > 0
mean(est_lstat != boston$lstat[-train])
```

```
## [1] 0.5494071
```