The Failures of Decision Tree

Srijit Mukherjee

28 July 2020

```
library("MASS")
```

```
## Warning: package 'MASS' was built under R version 3.6.3
```

```
library(parsnip)
```

```
## Warning: package 'parsnip' was built under R version 3.6.3
```

```
library(titanic)
```

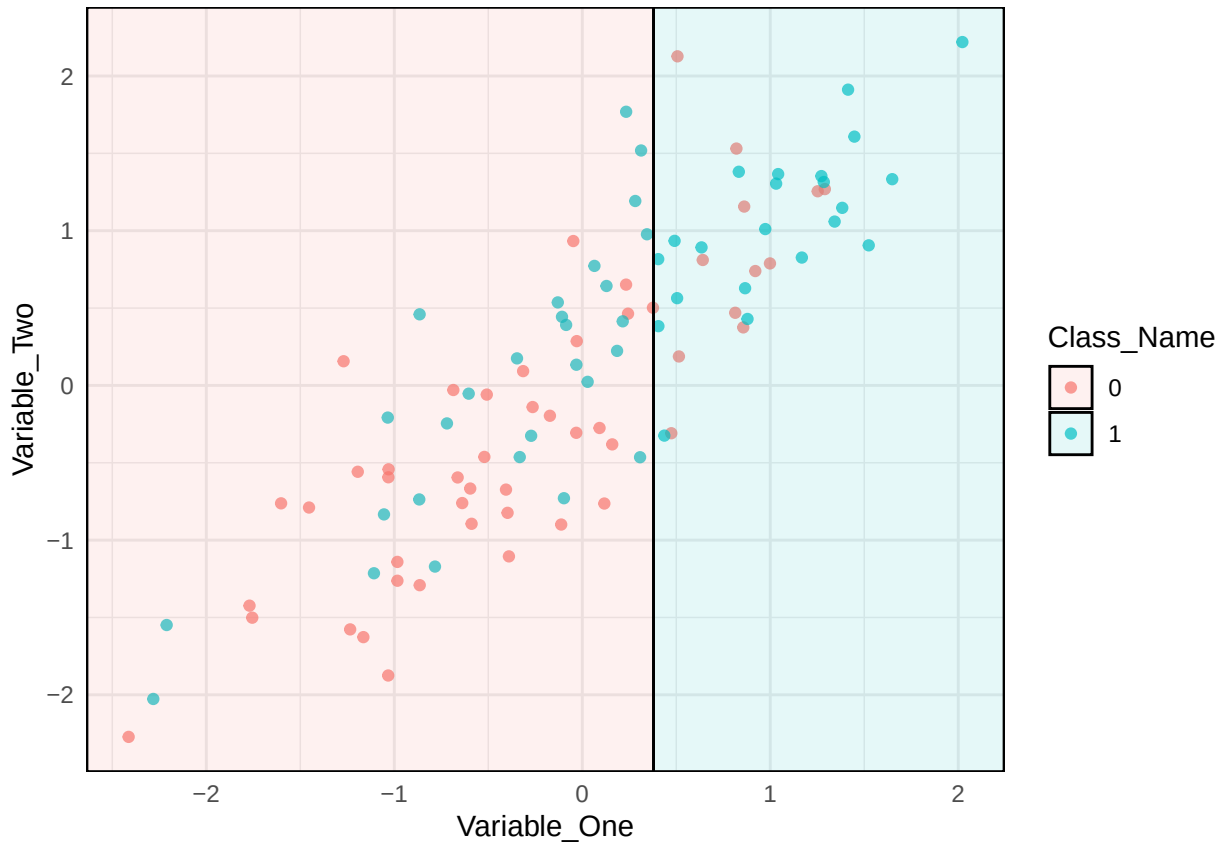
```
## Warning: package 'titanic' was built under R version 3.6.3
```

```
library(parttree)
library(rpart)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
#First Bivariate Normal
n0 <- 50
mu0 <- c(0,0)
Sigma0 <- matrix(c(1, 0.8, 0.8, 1),ncol = 2)
in_data_0 = mvrnorm(n = n0, mu = mu0, Sigma = Sigma0)
category0 = rep(0,n0)
data_0 = data.frame(in_data_0,category0)
colnames(data_0) = c("Variable_One","Variable_Two","Class_Name")
#Second Bivariate Normal
n1 <- 50
mu1 <- c(0.5, 0.7)
Sigma1 <- matrix(c(1, 0.9, 0.9, 1),ncol = 2)
in_data_1 = mvrnorm(n = n1, mu = mu1, Sigma = Sigma1)
category1 = rep(1,n1)
data_1 = data.frame(in_data_1,category1)
colnames(data_1) = c("Variable_One","Variable_Two","Class_Name")
Data = rbind(data_0,data_1)
Data$Class_Name = as.factor(Data$Class_Name)
decisiontree =
  decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("classification") %>%
  fit(Class_Name ~ Variable_One + Variable_Two, data = Data)
#decisiontree = rpart(Class_Name ~ Variable_One + Variable_Two, data = Data)
Data %>%
  ggplot(aes(x=Variable_One, y=Variable_Two)) +
  geom_jitter(aes(col=Class_Name), alpha=0.7) +
```

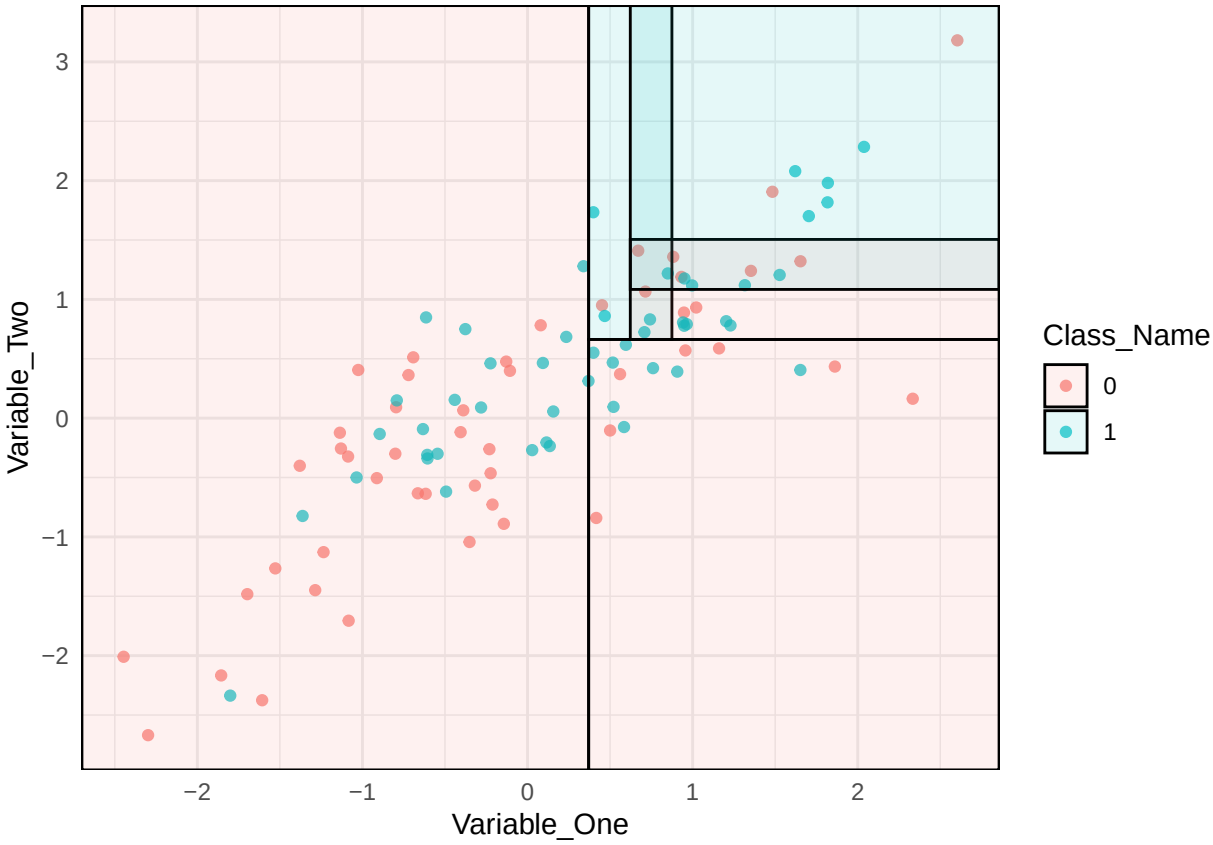
```
  geom_parttree(data = decisiontree, aes(fill=Class_Name), alpha = 0.1) +
  theme_minimal()
```




```
#Do this twice
```

```
#First Bivariate Normal
n0 <- 50
mu0 <- c(0,0)
Sigma0 <- matrix(c(1, 0.8, 0.8, 1),ncol = 2)
in_data_0 = mvrnorm(n = n0, mu = mu0, Sigma = Sigma0)
category0 = rep(0,n0)
data_0 = data.frame(in_data_0,category0)
colnames(data_0) = c("Variable_One","Variable_Two","Class_Name")
#Second Bivariate Normal
n1 <- 50
mu1 <- c(0.5, 0.7)
Sigma1 <- matrix(c(1, 0.9, 0.9, 1),ncol = 2)
in_data_1 = mvrnorm(n = n1, mu = mu1, Sigma = Sigma1)
category1 = rep(1,n1)
data_1 = data.frame(in_data_1,category1)
colnames(data_1) = c("Variable_One","Variable_Two","Class_Name")
Data = rbind(data_0,data_1)
Data$Class_Name = as.factor(Data$Class_Name)
decisiontree =
  decision_tree() %>%
  set_engine("rpart") %>%
```

```
  set_mode("classification") %>%
  fit(Class_Name ~ Variable_One + Variable_Two, data = Data)
#decisiontree = rpart(Class_Name ~ Variable_One + Variable_Two, data = Data)
Data %>%
  ggplot(aes(x=Variable_One, y=Variable_Two)) +
  geom_jitter(aes(col=Class_Name), alpha=0.7) +
  geom_parttree(data = decisiontree, aes(fill=Class_Name), alpha = 0.1) +
  theme_minimal()
```




```
#Do this twice
```

Ensemble Learning in R for Continuous Response

Srijit Mukherjee

27 July 2020

Libraries

- library(ISLR)
- library(MASS)
- library(tree)
- library(randomForest)
- library(gbm)
- library(adabag)
- library(caret)
- library(corrplot)

We will be using the data set Boston Data Set in ISLR package. Let's explore the data.

Dataset: Boston

This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass.

There are 14 attributes in each case of the dataset. They are:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

```
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
```

```
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

```
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 ...
```

Goal

Predict Price (MEDV)

Train and Test Data Set

```
train = sample (1:nrow(Boston), nrow(Boston)/2)
boston.test=Boston[-train ,"medv"]
```

We will apply the following methods on the same data set and compare them.

- Decision Tree
- Bagging
- Random Forest (Bagging + Decorrelate)
- Boosting

Decision Tree

Fit the model

```
set.seed(1)
tree.boston = tree(medv ~., Boston ,subset =train)
```

The summary of the Regression Tree

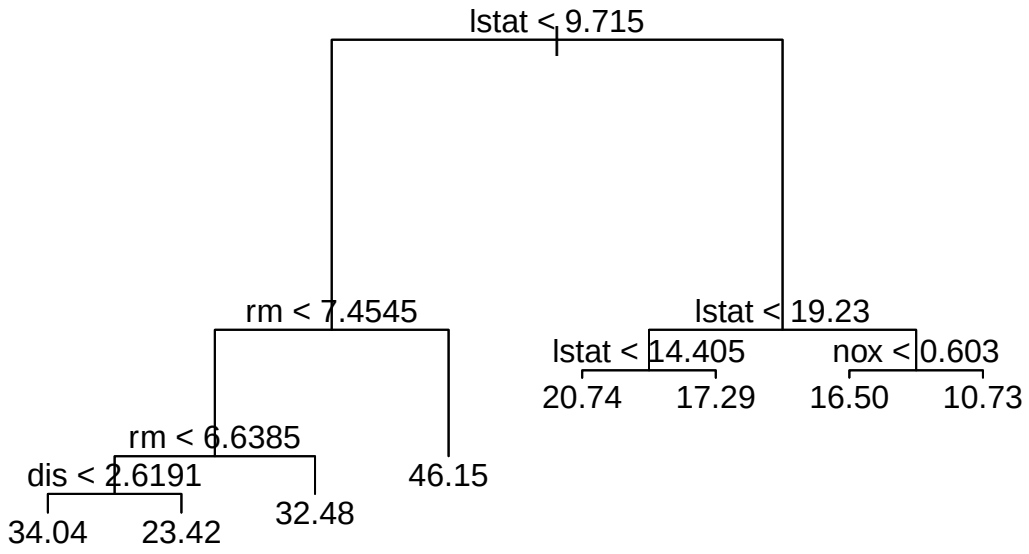
```
summary (tree.boston )
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
```

```
## Variables actually used in tree construction:
## [1] "lstat" "rm"    "dis"   "nox"
## Number of terminal nodes:  8
## Residual mean deviance:  14.71 = 3604 / 245
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.7400  -2.3840  -0.1245   0.0000   1.9630  17.5200
```

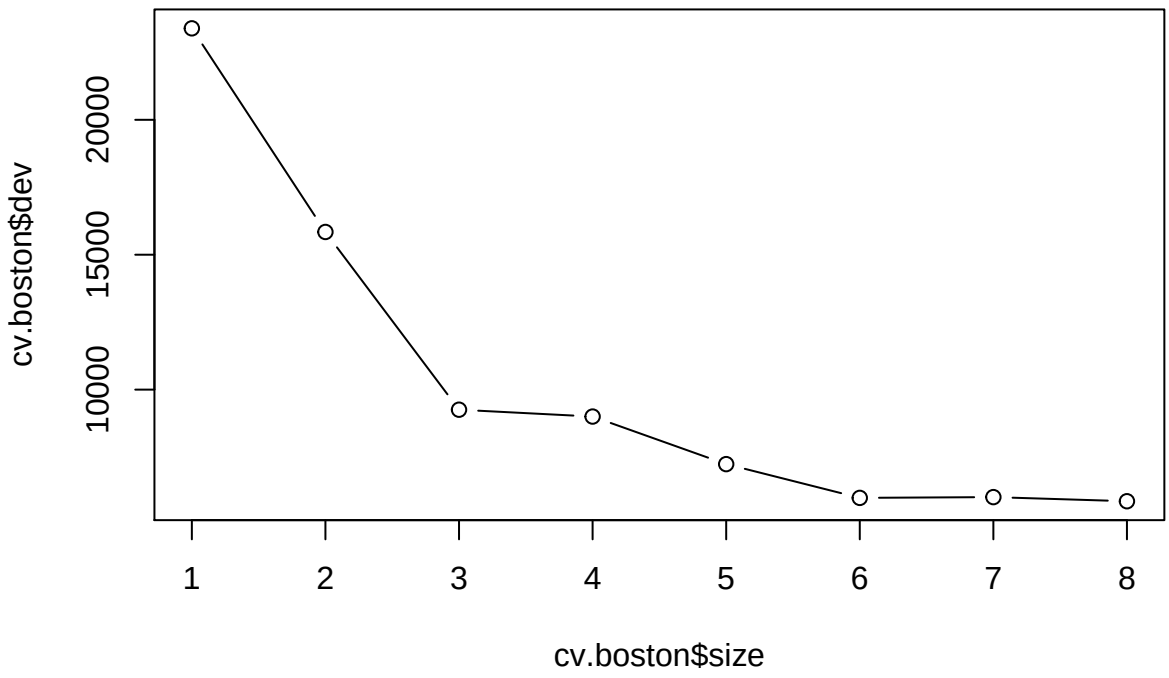
The Plot of the Decision Tree

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



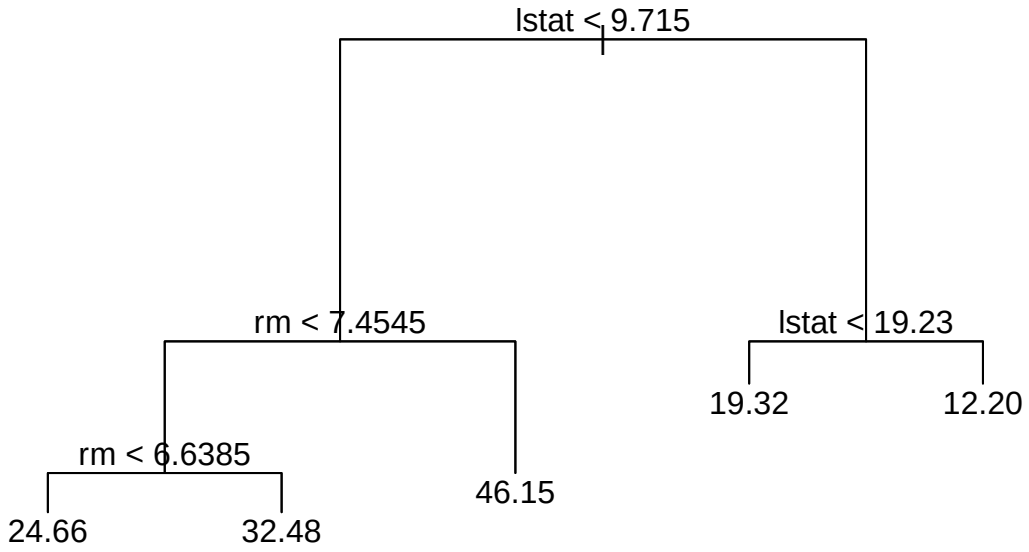
The Plot of the Cross Validation Deviance

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



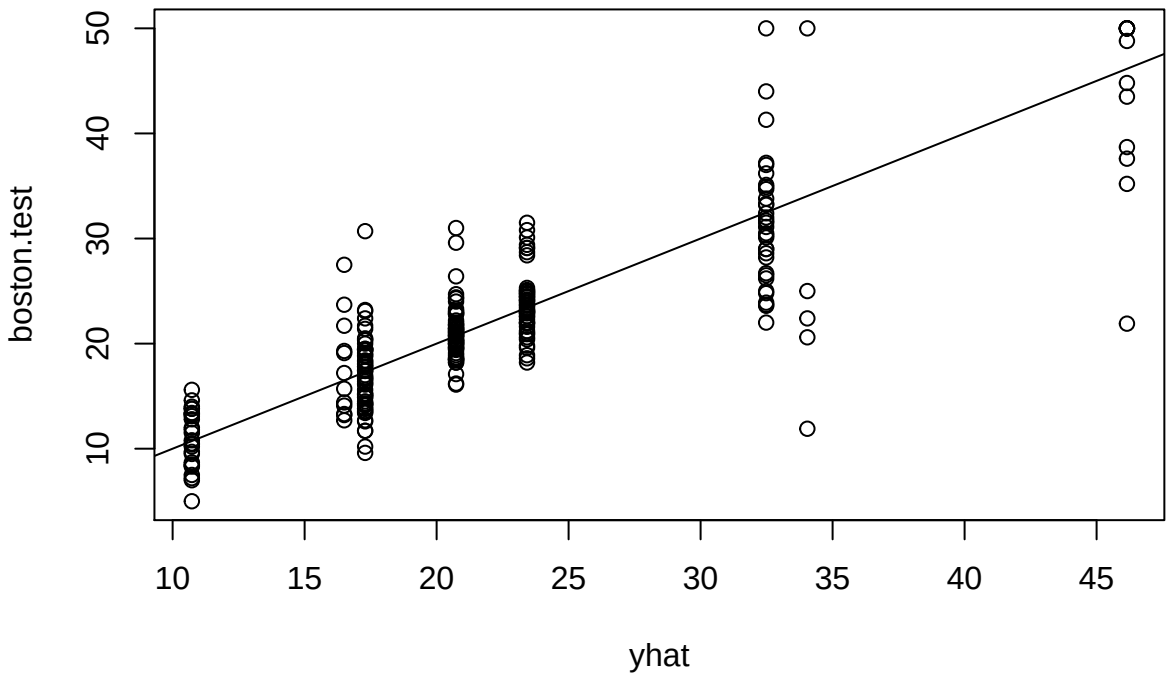
After Pruning and setting the number of leaf nodes to be 5.

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



Performance

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




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

```
## [1] 21.44007
```

Bagging

Prediction with Bagging

```
set.seed(2)
bag.boston= randomForest(medv~., data = Boston, subset=train, mtry=13,importance =TRUE)
yhat.bag = predict(bag.boston , newdata = Boston[-train ,])
mean((yhat.bag - boston.test)^2)
```

```
## [1] 11.76381
```

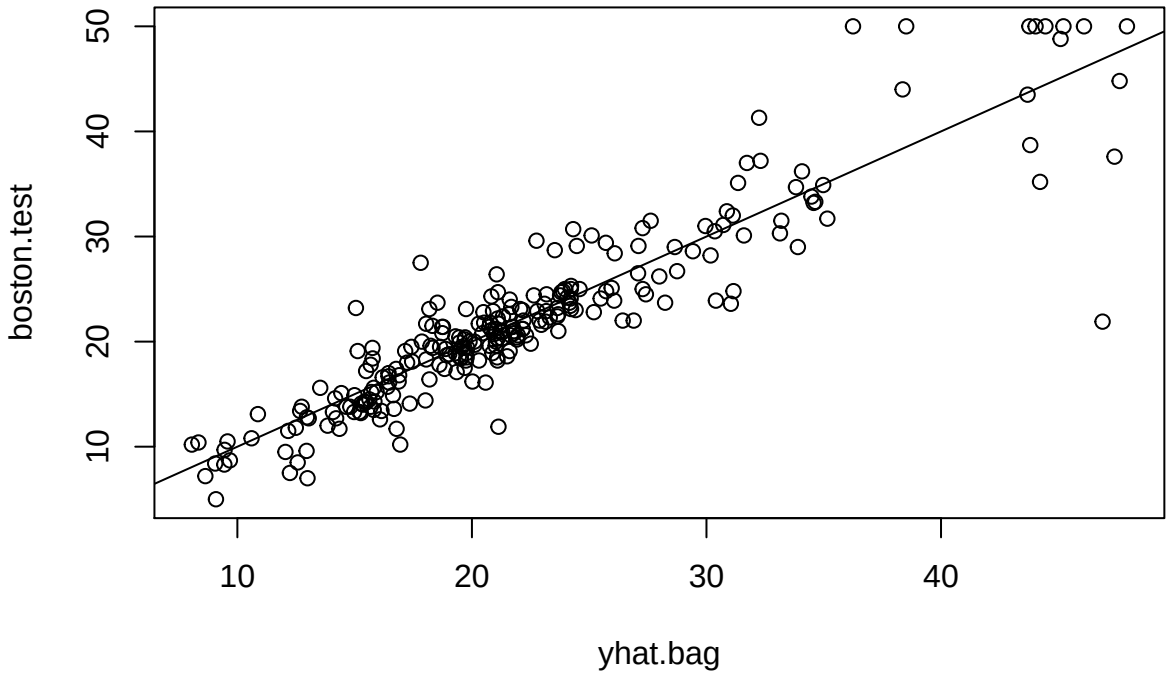
Visualize the performance of the Bagging

```
summary(bag.boston)
```

```
##                 Length Class  Mode
## call              6    -none- call
## type              1    -none- character
## predicted       253    -none- numeric
## mse             500    -none- numeric
## rsq             500    -none- numeric
## oob.times       253    -none- numeric
## importance       26    -none- numeric
## importanceSD     13    -none- numeric
```

```
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y               253    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

```
plot(yhat.bag, boston.test)
abline (0,1)
```



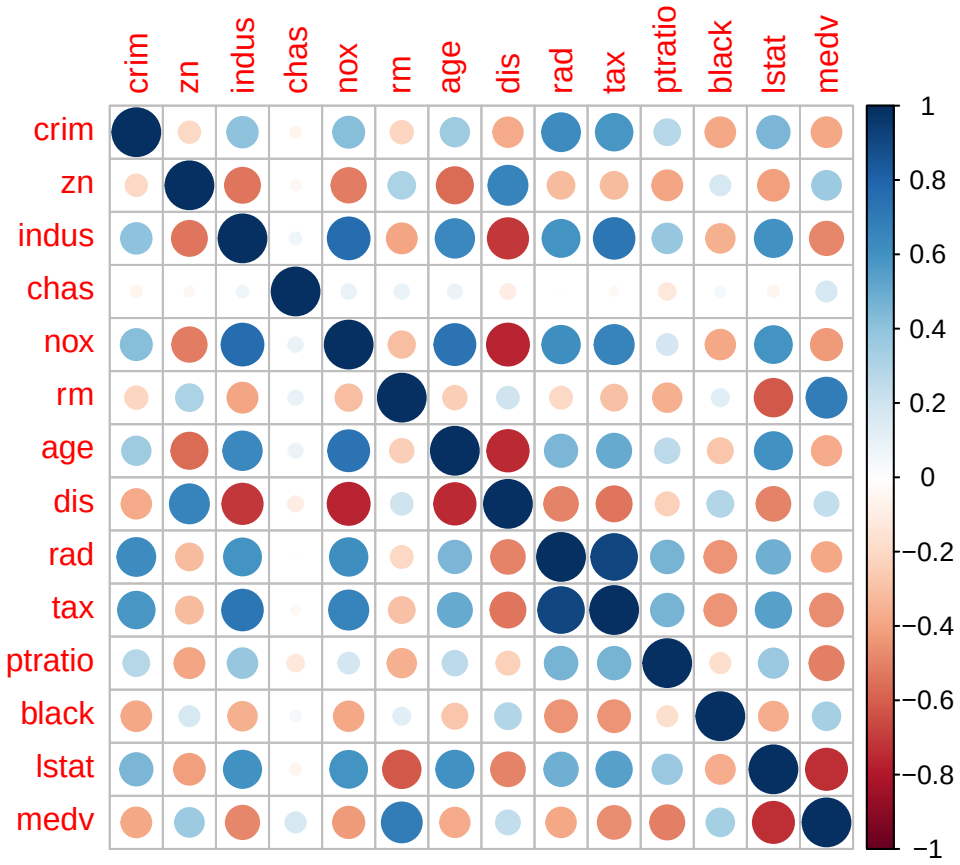
Change in the number of trees: Performance Measure

```
bag.boston= randomForest(medv~., data = Boston, subset=train, mtry=13,importance =TRUE, ntree = 25)
yhat.bag = predict (bag.boston , newdata = Boston[-train ,])
mean((yhat.bag - boston.test)^2)
```

```
## [1] 11.31016
```

Random Forest

```
corrplot(cor(Boston))
```



Change in the random feature selection

```
rf.boston= randomForest(medv~., data = Boston, subset=train, mtry=6,importance =TRUE, ntree = 25)
yhat.bag = predict (rf.boston , newdata = Boston[-train ,])
mean((yhat.bag - boston.test)^2)
```

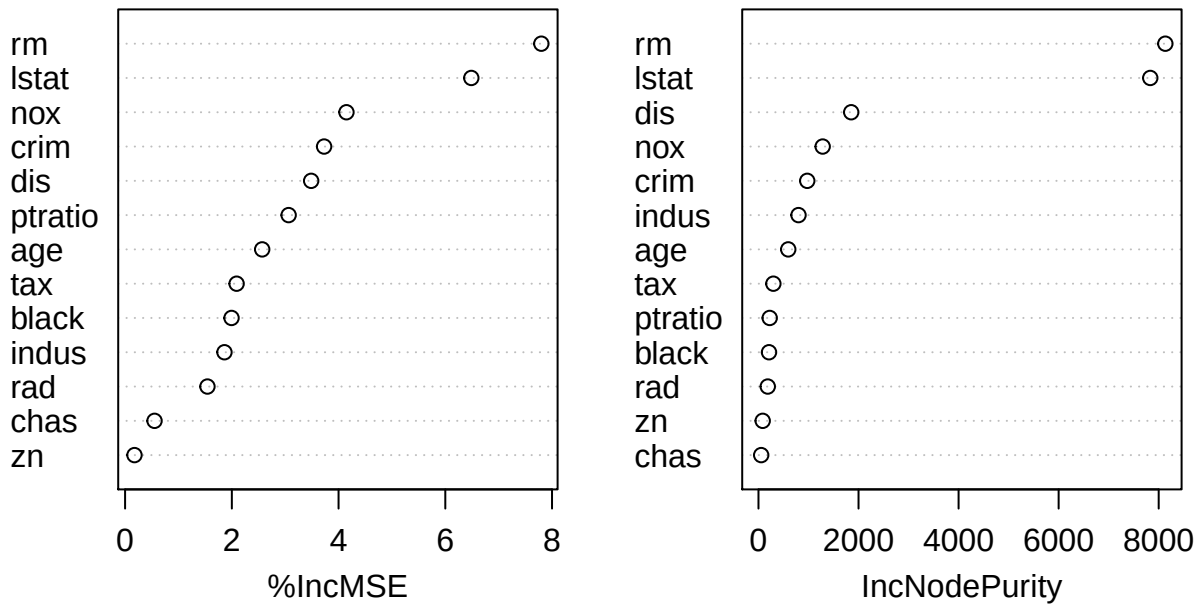
```
## [1] 11.37602
```

Variable Importance Plot

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

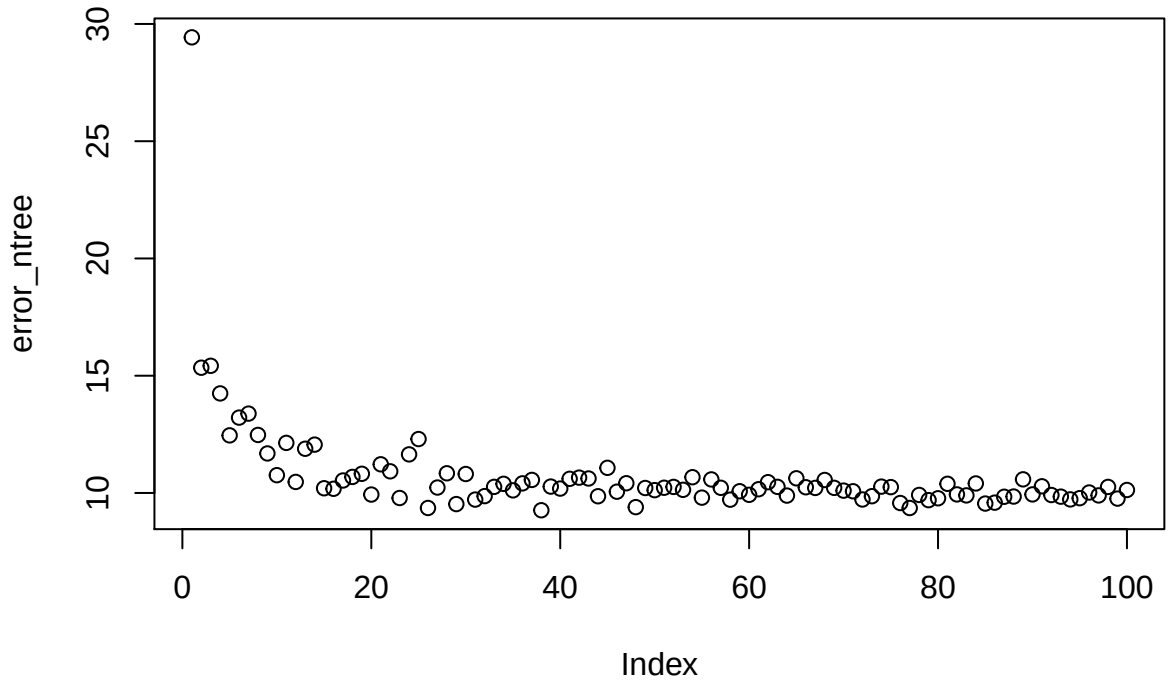
```
##           %IncMSE IncNodePurity
## crim    3.7288991     974.81169
## zn      0.1757520      84.06701
## indus   1.8616006     800.03483
## chas    0.5517209      54.50043
## nox     4.1479699    1282.75239
## rm      7.7993103    8132.42528
## age     2.5684879     595.04854
## dis     3.4868850    1853.87596
## rad     1.5429946     185.42431
## tax     2.0897403     297.41190
## ptratio 3.0643209     224.11632
## black   1.9940922     210.39630
## lstat   6.4874809    7831.26581
```

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



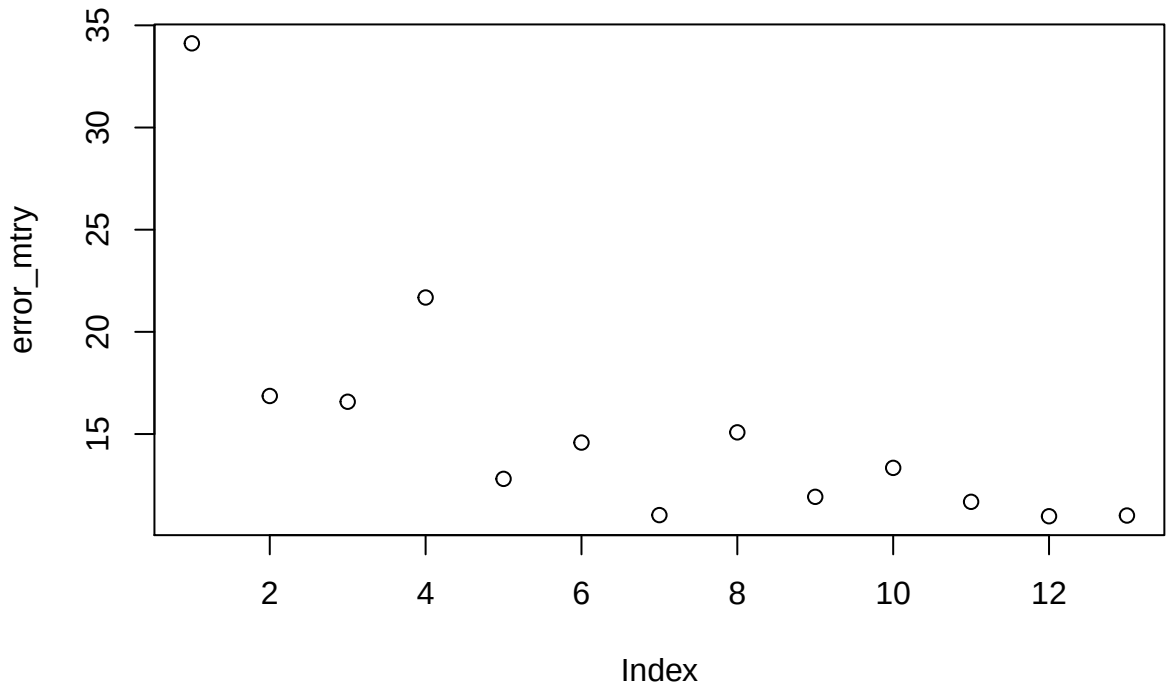
How the number of trees result in the error?

```
error_ntree = NULL

for (n in 1:100)
{
rf.boston= randomForest(medv~., data = Boston, subset=train, mtry=6, importance =TRUE, ntree = n)
yhat.bag = predict (rf.boston , newdata = Boston[-train ,])
error_ntree = c(error_ntree, mean((yhat.bag - boston.test)^2))
}
plot(error_ntree)
```



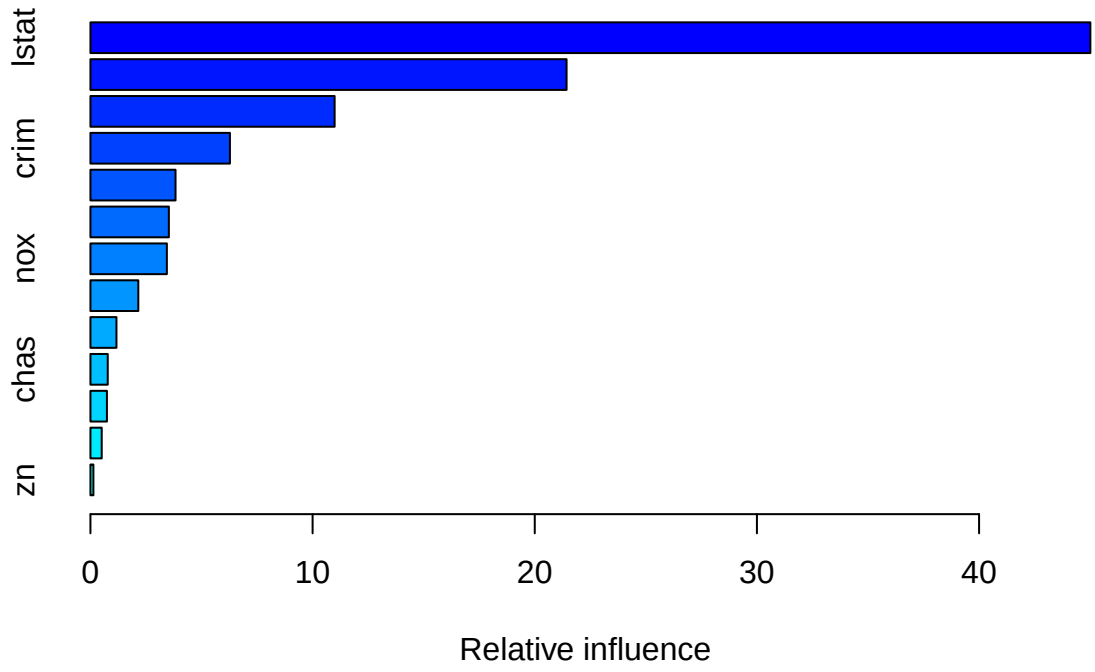
How the number of predictor selection result in the error?

```
error_mtry = NULL

for (n in 1:(ncol(Boston)-1))

{
rf.boston= randomForest(medv~., data = Boston, subset=train, mtry=6, importance =TRUE, ntree = n)
yhat.bag = predict (rf.boston , newdata = Boston[-train ,])
error_mtry = c(error_mtry, mean((yhat.bag - boston.test)^2))
}
plot(error_mtry)
```



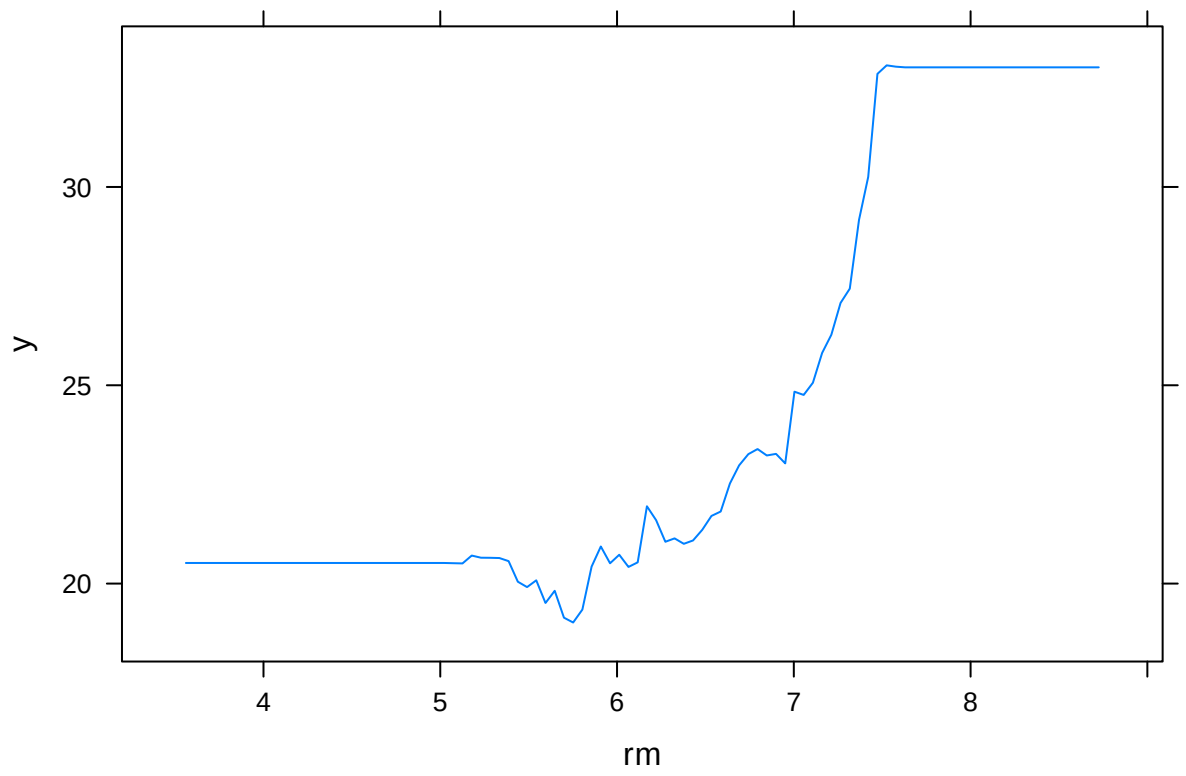
Gradient Boosting

```
set.seed (3)
boost.boston =gbm(medv ~ ., data= Boston[train ,], distribution = "gaussian", n.trees =5000 , interacti
summary (boost.boston )
```

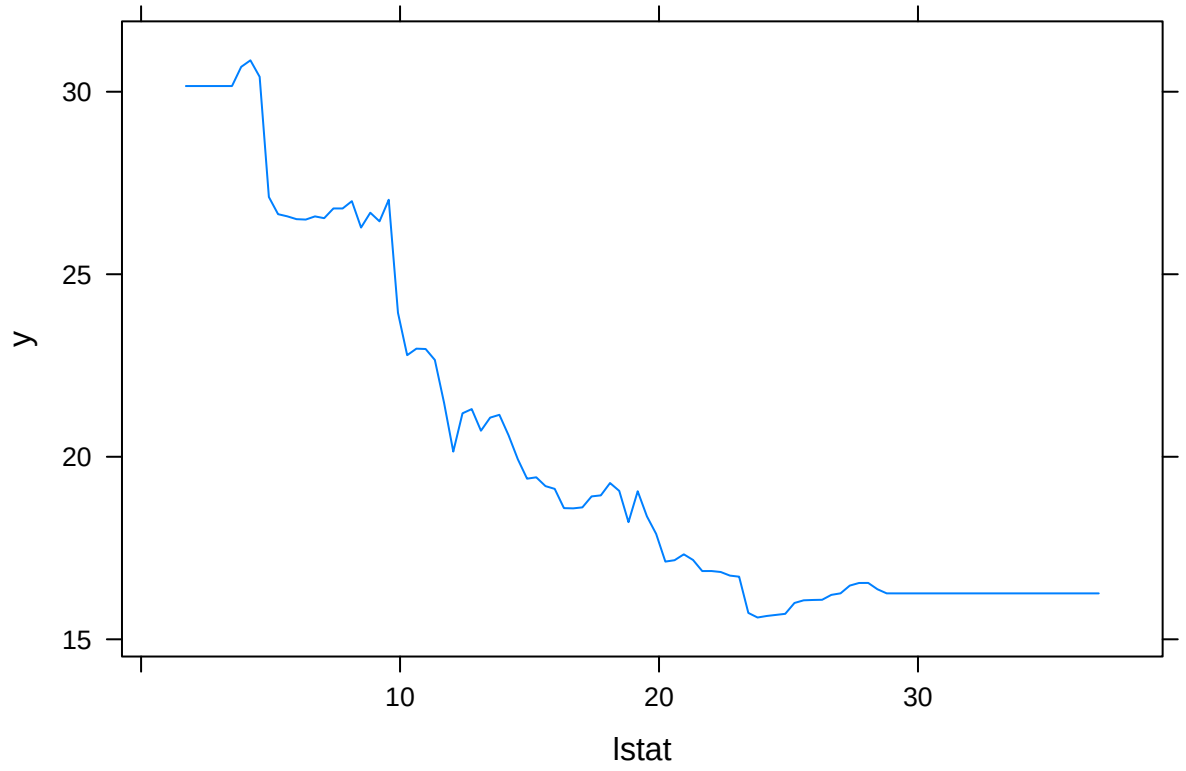



```
##             var    rel.inf
## lstat     lstat 45.0085511
## rm           rm 21.4320298
## dis         dis 10.9916191
## crim       crim  6.2791289
## black     black  3.8303979
## age         age  3.5298818
## nox         nox  3.4402379
## ptratio ptratio  2.1549458
## tax         tax  1.1719891
## chas       chas  0.7790845
## indus     indus  0.7417649
## rad         rad  0.5076981
## zn           zn  0.1326712
```

```
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] 11.00685
```

Ensemble Learning in R for Categorical Response

Srijit Mukherjee

27 July 2020

Libraries

- library(ISLR)
- library(MASS)
- library(tree)
- library(randomForest)
- library(gbm)
- library(adabag)
- library(caret)
- library(corrplot)

We will be using the data set Boston Data Set in ISLR package. Let's explore the data.

Dataset: Boston

This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass.

There are 14 attributes in each case of the dataset. They are:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

Categorical Dataset

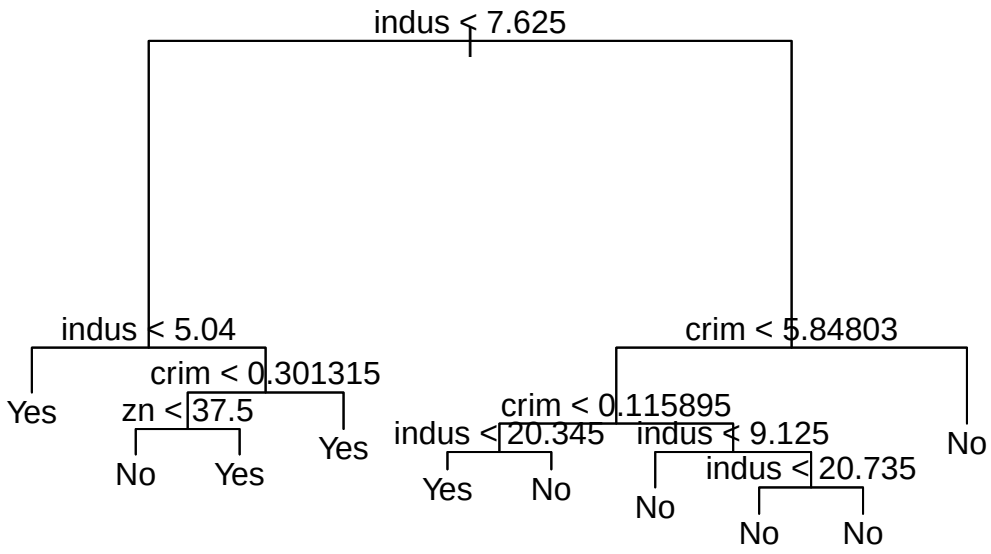
```
set.seed(1)
attach(Boston)
High = ifelse(medv <= median((Boston$medv)),"No"," Yes ")
Boston.Cat = data.frame(Boston[-Boston$medv] ,High)
```

Categorical Response

```
tree.Boston.Cat = tree(High ~. ,Boston.Cat )
summary(tree.Boston.Cat)
```

```
##
## Classification tree:
## tree(formula = High ~ ., data = Boston.Cat)
## Variables actually used in tree construction:
## [1] "indus" "crim"  "zn"
## Number of terminal nodes:  10
## Residual mean deviance:  0.911 = 451.8 / 496
## Misclassification error rate: 0.2253 = 114 / 506
```

```
plot(tree.Boston.Cat)
text(tree.Boston.Cat, pretty = 0)
```




```
set.seed (2)
train = sample(1:nrow(Boston.Cat), nrow(Boston.Cat)/2)
boston.test=Boston.Cat[-train,]
High.test=High[-train]
tree.Boston.Cat = tree(High ~ ., Boston.Cat, subset =train )
tree.pred = predict(tree.Boston.Cat, boston.test , type ="class")
table(tree.pred ,High.test)
```

```
##          High.test
## tree.pred  Yes   No
```

```
##      Yes    107  53
##     No       13  80
```

```
rf.boston= randomForest(High~., data = Boston.Cat, subset=train, mtry=13,importance =TRUE, ntree = 25)
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
pred = predict (rf.boston , newdata = Boston.Cat[-train ,], type = "class")
table(pred ,High.test)
```

```
##        High.test
## pred     Yes  No
##    Yes     88 38
##   No       32 95
```

Categorical Response

```
rf.boston= randomForest(High~., data = Boston.Cat, subset=train, mtry=6,importance =TRUE, ntree = 25)
```

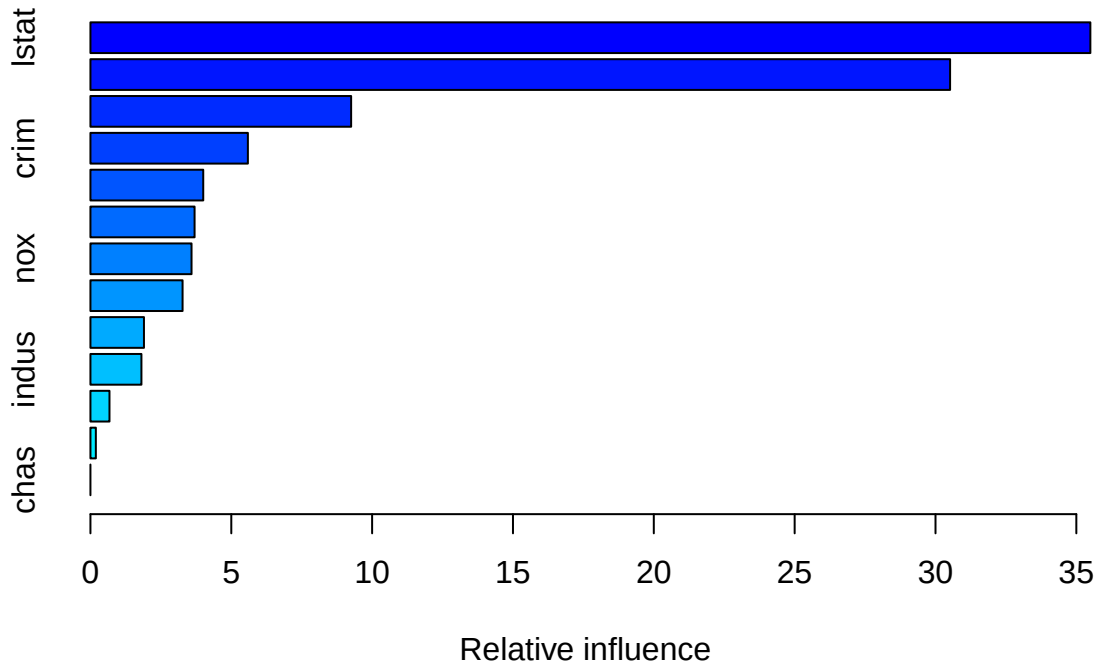
```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
pred = predict (rf.boston , newdata = Boston.Cat[-train ,], type = "class")
table(pred ,High.test)
```

```
##        High.test
## pred     Yes  No
##    Yes     90 37
##   No       30 96
```

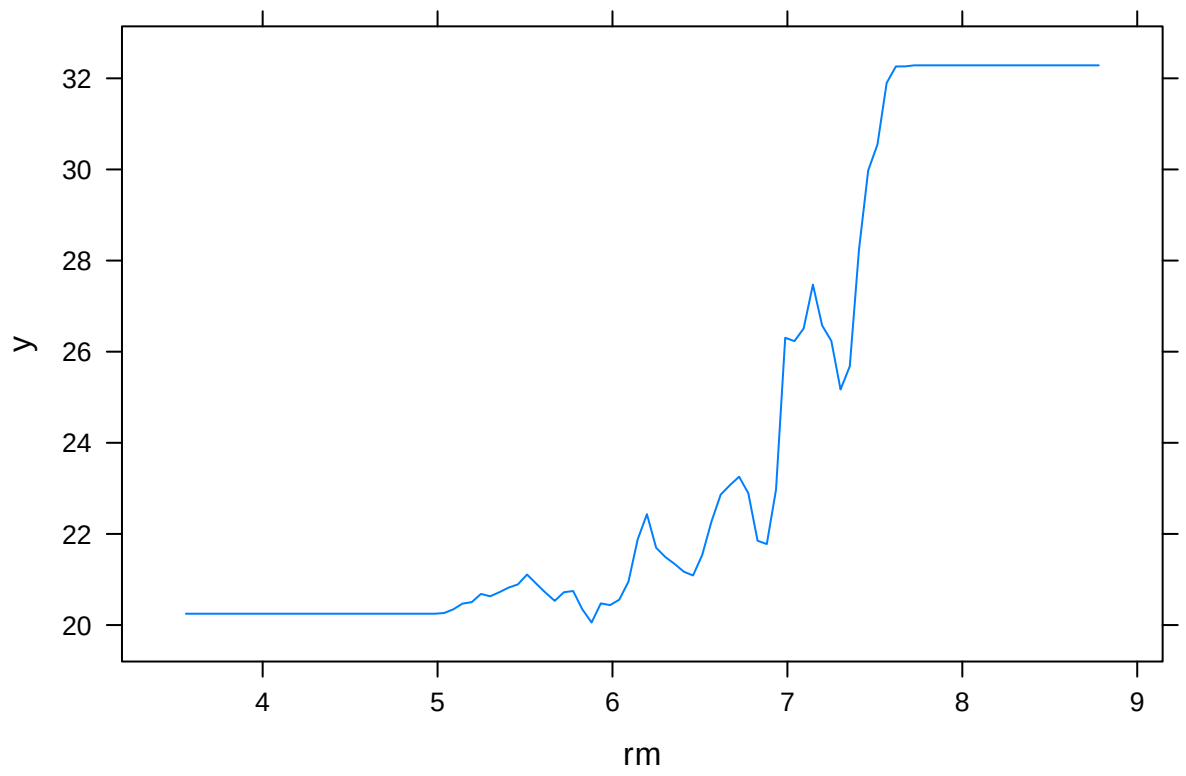
Gradient Boosting

```
set.seed (3)
boost.boston =gbm(medv ~ ., data= Boston[train ,], distribution = "gaussian", n.trees =5000 , interacti
summary (boost.boston )
```

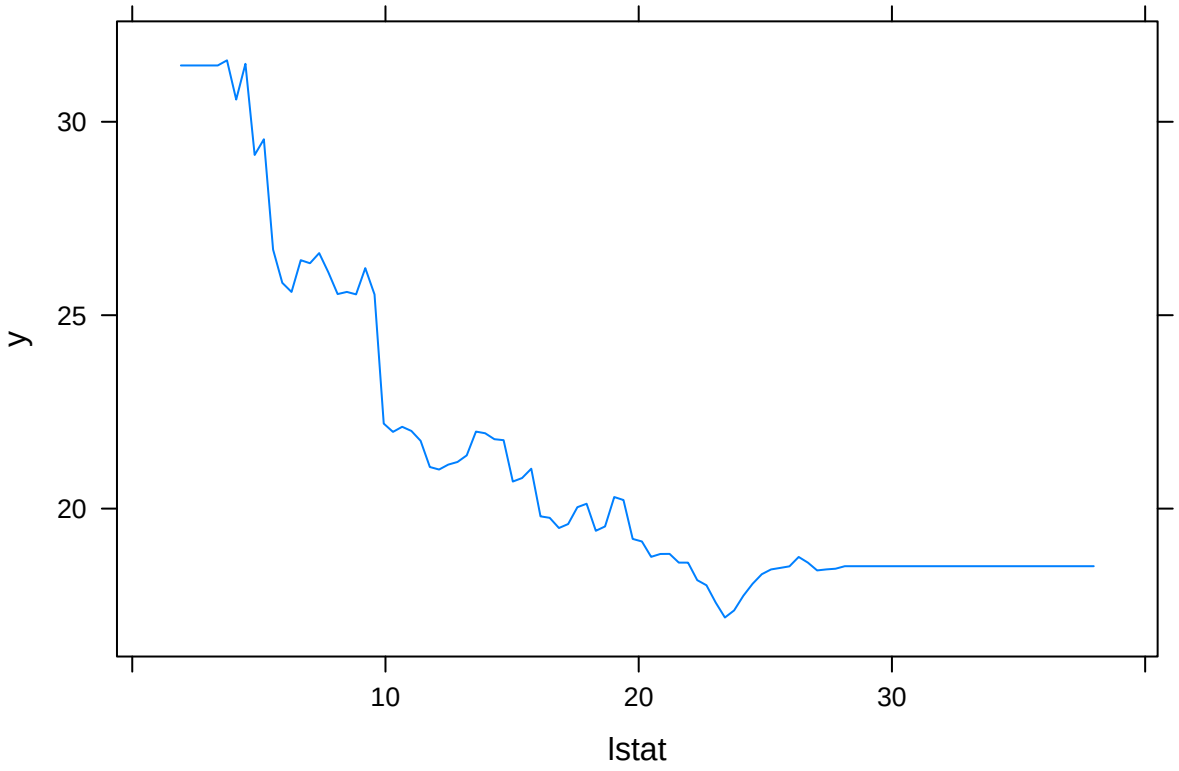



```
##             var      rel.inf
## lstat     lstat 3.549759e+01
## rm           rm 3.051791e+01
## dis         dis 9.254619e+00
## crim       crim 5.589272e+00
## black     black 4.004567e+00
## age         age 3.697216e+00
## nox         nox 3.588891e+00
## ptratio ptratio 3.270867e+00
## tax         tax 1.900395e+00
## indus     indus 1.810200e+00
## rad         rad 6.761637e-01
## zn           zn 1.919185e-01
## chas       chas 3.985729e-04
```

```
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)
```

```
## Warning in Ops.factor(left, right): '-' not meaningful for factors
```

```
## Warning in mean.default((yhat.boost - boston.test)^2): argument is not numeric
## or logical: returning NA
```

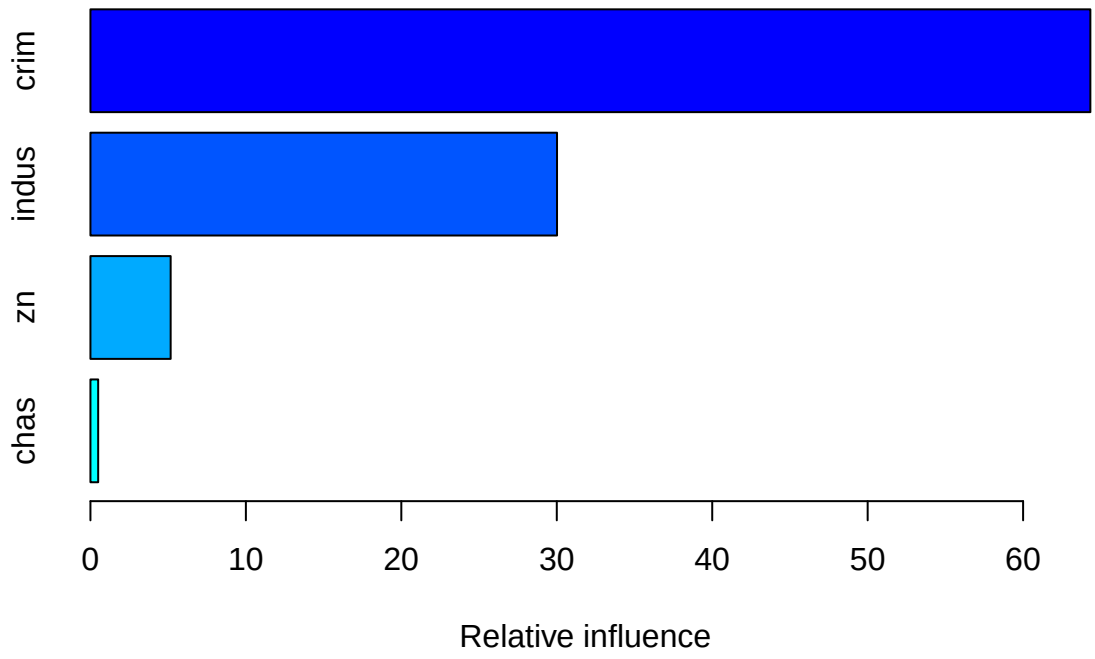
```
## [1] NA
```

Categorical Response

```
set.seed (3)
boost.boston = gbm(High ~ ., data=Boston.Cat[train,], distribution = "multinomial", n.trees =5000 , int
```

```
## Warning: Setting `distribution = "multinomial"` is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

```
summary (boost.boston )
```




```
##         var    rel.inf
## crim   crim 64.3257192
## indus indus 30.0161105
## zn       zn  5.1603906
## chas   chas  0.4977796
```

```
boost.pred = predict(boost.boston, boston.test , type ="response")
```

```
## Using 5000 trees...
```

```
labels = colnames(boost.pred)[apply(boost.pred, 1, which.max)]
result = data.frame(boston.test$High, labels)
cm = confusionMatrix(boston.test$High, as.factor(labels))
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Yes  No
##       Yes     85 35
##      No       42 91
##
##                Accuracy : 0.6957
##                  95% CI : (0.6349, 0.7517)
##     No Information Rate : 0.502
##     P-Value [Acc > NIR] : 3.235e-10
##
##                   Kappa : 0.3914
```

```
##
##  Mcnemar's Test P-Value : 0.4941
##
##             Sensitivity : 0.6693
##             Specificity : 0.7222
##          Pos Pred Value : 0.7083
##          Neg Pred Value : 0.6842
##              Prevalence : 0.5020
##          Detection Rate : 0.3360
##    Detection Prevalence : 0.4743
##       Balanced Accuracy : 0.6958
##
##        'Positive' Class :  Yes
##
```

Adaboost (for Classifier)

```
set.seed(123)
indexes=createDataPartition(iris$Species, p=.90, list = F)
train = iris[indexes, ]
test = iris[-indexes, ]
model = boosting(Species~., data=train, boos=TRUE, mfinal=50)
pred = predict(model, test)
print(pred$confusion)
```

```
##                Observed Class
## Predicted Class setosa versicolor virginica
##      setosa          5          0         0
##      versicolor      0          5         0
##      virginica       0          0         5
```

```
print(pred$error)
```

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

```
model = boosting(High~., data=Boston.Cat, boos=TRUE, mfinal=50, subset = train)
pred = predict(model, boston.test, type ="Class")
print(pred$confusion)
```

```
##                Observed Class
## Predicted Class  Yes   No
##            Yes    116   1
##           No        4 132
```

```
print(pred$error)
```

```
## [1] 0.01976285
```