Recitation 8

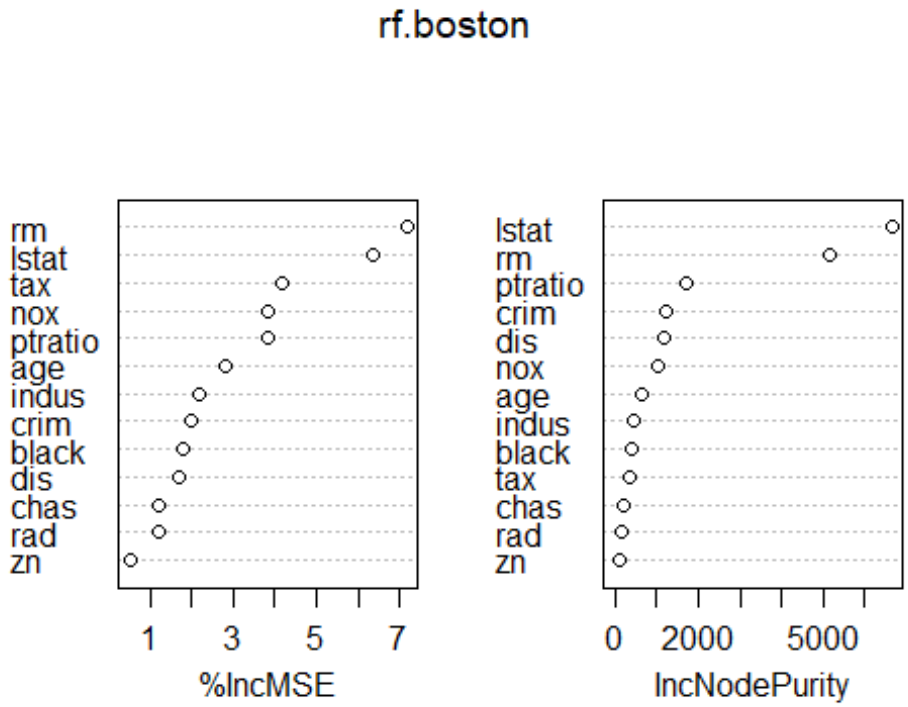
Di Zhang

April 2, 2020

##Bagging and Random Forest##

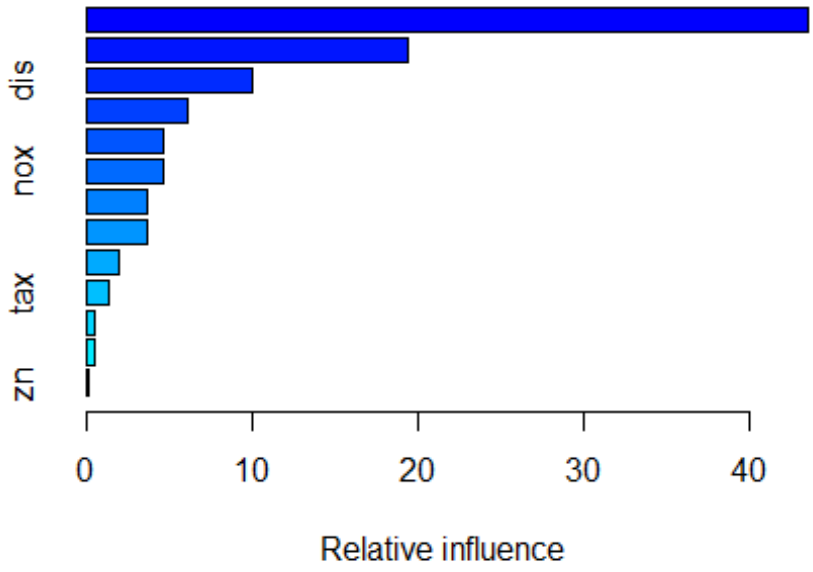
```
#install.packages("randomForest")
#install.packages("gbm")

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.6.3

## randomForest 4.6-14

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

library(gbm)

## Warning: package 'gbm' was built under R version 3.6.3

## Loaded gbm 2.1.5

library(MASS)
library(caret)

## Warning: package 'caret' was built under R version 3.6.2

## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

train=sample(1:nrow(Boston),nrow(Boston)/2) #construct training data

set.seed(1)
bag.boston=randomForest(medv~.,data=Boston,
subset=train,mtry=13,importance=TRUE) #apply bagging on decision tree
bag.boston

##
## 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: 14.19146
##                     % Var explained: 81.12

#calculate test error

yhat.bag=predict(bag.boston,newdata=Boston[-train,])
boston.test=Boston[-train,"medv"]
mean((yhat.bag-boston.test)^2)

## [1] 13.11882

bag.boston=randomForest(medv~.,data=Boston,
subset=train,mtry=13,importance=TRUE,ntree=25) #change the number of trees in
bagging
yhat.bag=predict(bag.boston,newdata=Boston[-train,])
mean((yhat.bag-boston.test)^2)

## [1] 12.75639

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

## [1] 15.58152

importance(rf.boston) #check variable selection based on random forest

##           %IncMSE IncNodePurity
## crim    1.9794793     1218.5765
## zn      0.5153608      128.5101
## indus   2.1822242      429.1297
## chas    1.2107269      216.2073
## nox     3.8445372     1010.2364
## rm      7.1776001     5135.9050
## age     2.8282951      659.8836
## dis     1.6773452     1159.0052
## rad     1.1916082      166.3568
## tax     4.1559331      332.8815
## ptratio 3.8409458     1733.3943
## black   1.7817034      382.3050
## lstat   6.3810333     6654.2797

varImpPlot(rf.boston)
```

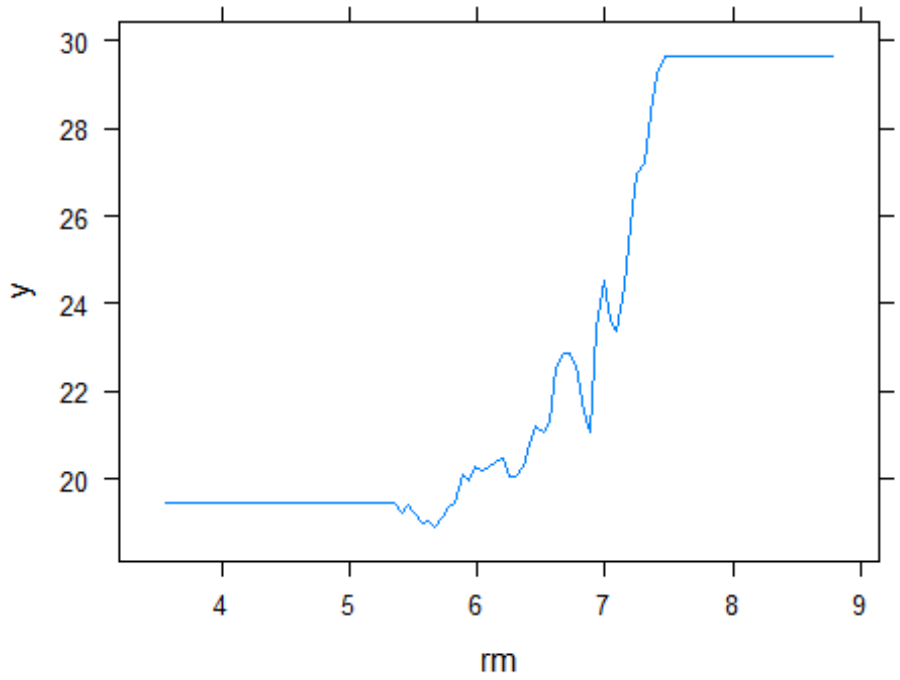


##Boosting##

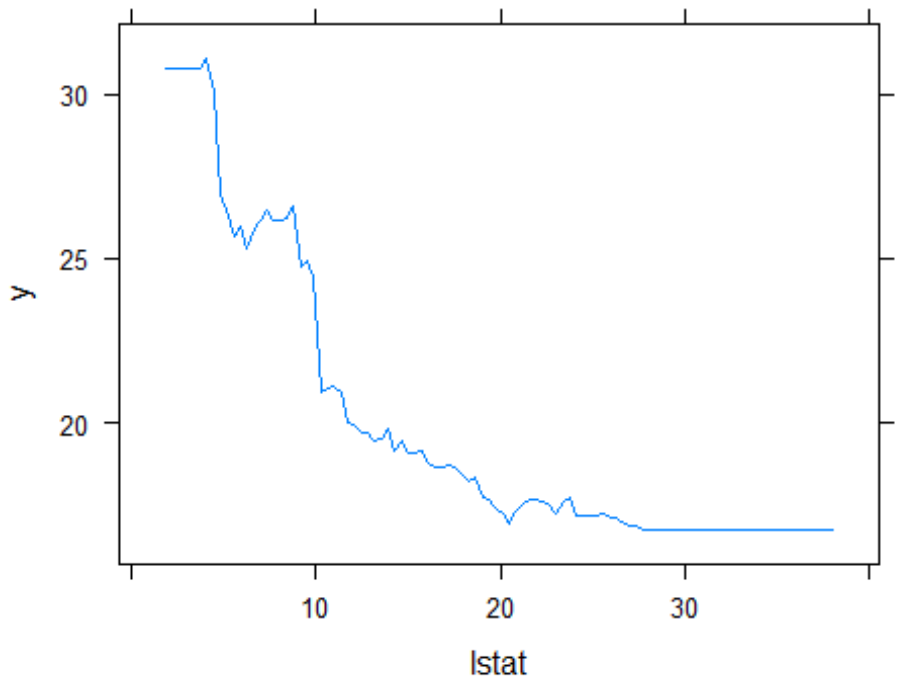
```
set.seed(1)
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.tree=50
00,interaction.depth = 4) #fit a boosted tree
summary(boost.boston)
```




```
##             var    rel.inf
## lstat     lstat 43.5117058
## rm           rm 19.3295951
## dis         dis  9.9930061
## crim       crim  6.0670505
## black     black  4.6840664
## nox         nox  4.6243641
## ptratio ptratio  3.6968979
## age         age  3.6740824
## indus     indus  1.9815526
## tax         tax  1.3173589
## rad         rad  0.5452526
## chas       chas  0.4513146
## zn           zn  0.1237530

#check marginal effect of variables rm and lstat

par(mfrow=c(1,2))
plot(boost.boston,i="rm")
```




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



```r
#calculate test error

yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
```

```
## [1] 13.53037
```

```r
#fit a boosted tree with shrinkage parameter=0.05

boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.tree=50
00,interaction.depth = 4, shrinkage=0.05)
yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
```

```
## [1] 12.66286
```

```r
#hyper-parameter tuning of gradient boosting

grid <- expand.grid(n.trees = c(1000,1500), interaction.depth=c(1:3),
shrinkage=c(0.05,0.1),n.minobsinnode=c(10))
ctrl <- trainControl(method  = "cv",number  = 5)
unwantedoutput <- capture.output(GBMModel <- train(medv~.,data =
Boston[train,],
                  method = "gbm", trControl = ctrl, tuneGrid = grid))

print(GBMModel)
```

```
## Stochastic Gradient Boosting
##
## 253 samples
##  13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 204, 202, 201, 202, 203
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  RMSE      Rsquared   MAE
##   0.05       1                  1000     3.934946  0.7983634  2.684897
##   0.05       1                  1500     3.986079  0.7942582  2.705989
##   0.05       2                  1000     3.838295  0.8097530  2.652675
##   0.05       2                  1500     3.858073  0.8095656  2.667035
##   0.05       3                  1000     3.786607  0.8167123  2.599540
##   0.05       3                  1500     3.807944  0.8155136  2.624960
##   0.10       1                  1000     4.014449  0.7916047  2.728732
##   0.10       1                  1500     4.150902  0.7799254  2.796090
##   0.10       2                  1000     4.001589  0.7980385  2.741078
##   0.10       2                  1500     4.020768  0.7976402  2.734834
##   0.10       3                  1000     3.929710  0.8055762  2.697156
##   0.10       3                  1500     3.932403  0.8056749  2.709856
##
```

```
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 1000,
##  interaction.depth = 3, shrinkage = 0.05 and n.minobsinnode = 10.
```

```
yhat.boost=predict(GBMModel,newdata=Boston[-train,],n.trees=1000)
mean((yhat.boost-boston.test)^2)
```

```
## [1] 12.59375
```