Trees and ensemble models

STAT5003

Justin Wishart

Libraries to load

```
library(tree)
```

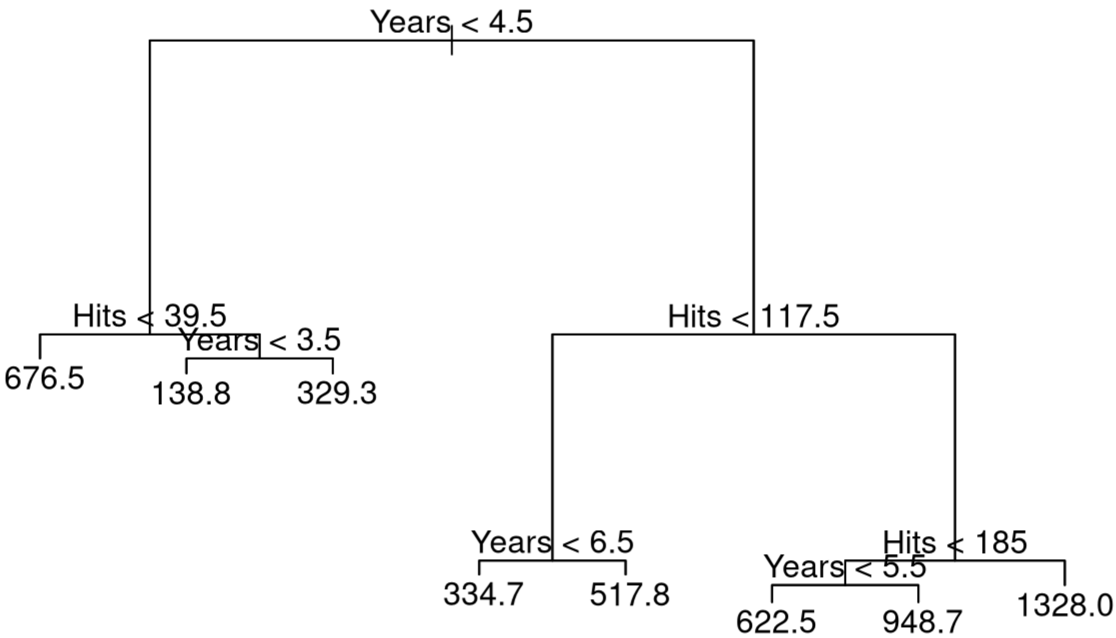
Single tree based methods

Regression tree

```
data(Hitters, package = "ISLR")
Hitters <- na.omit(Hitters)
rt <- tree(Salary ~ Hits + Years, data = Hitters)

summary(rt)
```

```
##
## Regression tree:
## tree(formula = Salary ~ Hits + Years, data = Hitters)
## Number of terminal nodes:  8
## Residual mean deviance:  101200 = 25820000 / 255
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1238.00  -157.50   -38.84     0.00    76.83  1511.00
```

```
plot(rt)
text(rt)
```



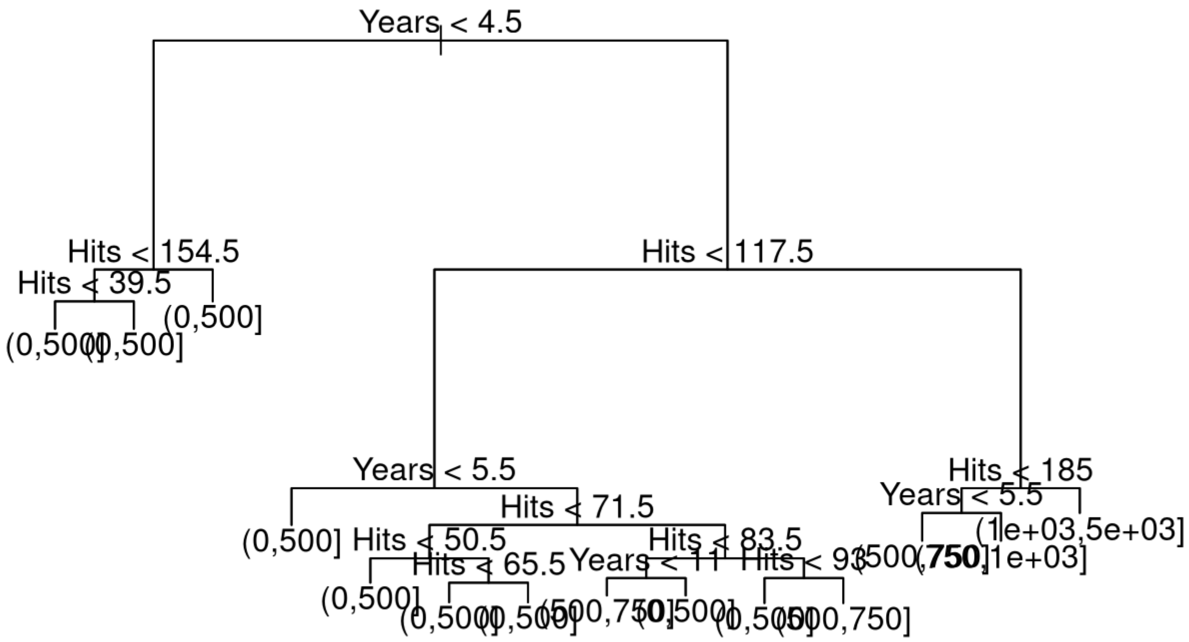

```
# Lecture tree
# rt <- tree(Salary ~ Hits + Years, data = Hitters,
#             control = tree.control(nobs = nrow(Hitters), minsize = 100)
```

Classification tree

```
salary.cat <- cut(Hitters[["Salary"]],
                  breaks = c(0, 500, 750, 1000, 5000))
ct <- tree(salary.cat ~ Hits + Years, data = Hitters)

summary(ct)
```

```
##
## Classification tree:
## tree(formula = salary.cat ~ Hits + Years, data = Hitters)
## Number of terminal nodes:  14
## Residual mean deviance:  1.39 = 346 / 249
## Misclassification error rate: 0.308 = 81 / 263
```

```
plot(ct)
text(ct)
```



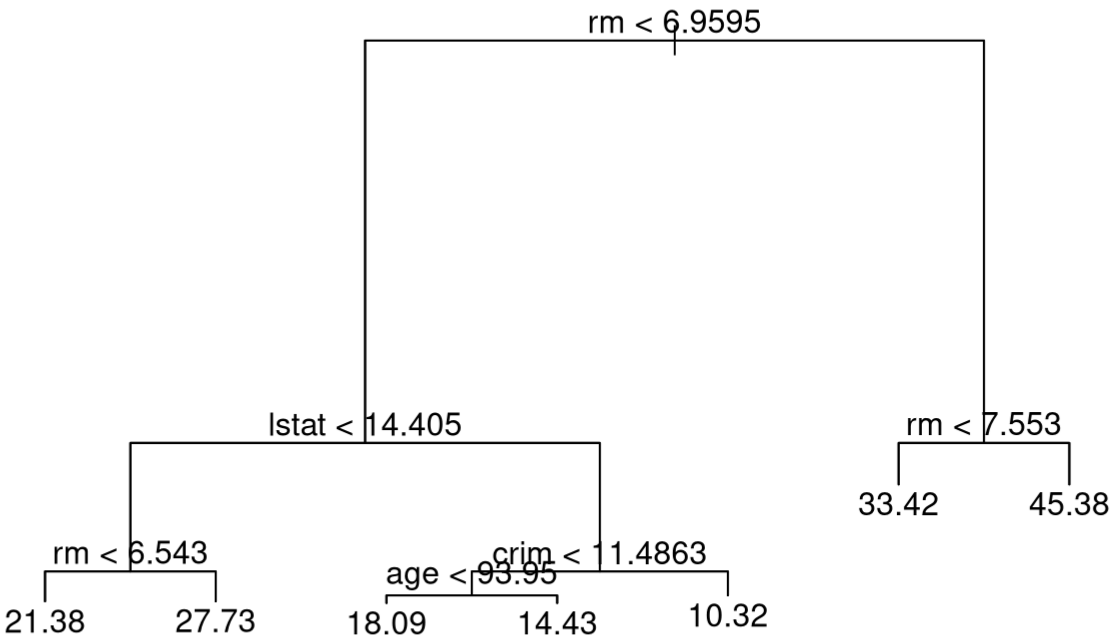
Regression tree

This section introduce regression tree using housing value dataset of Boston suburbs

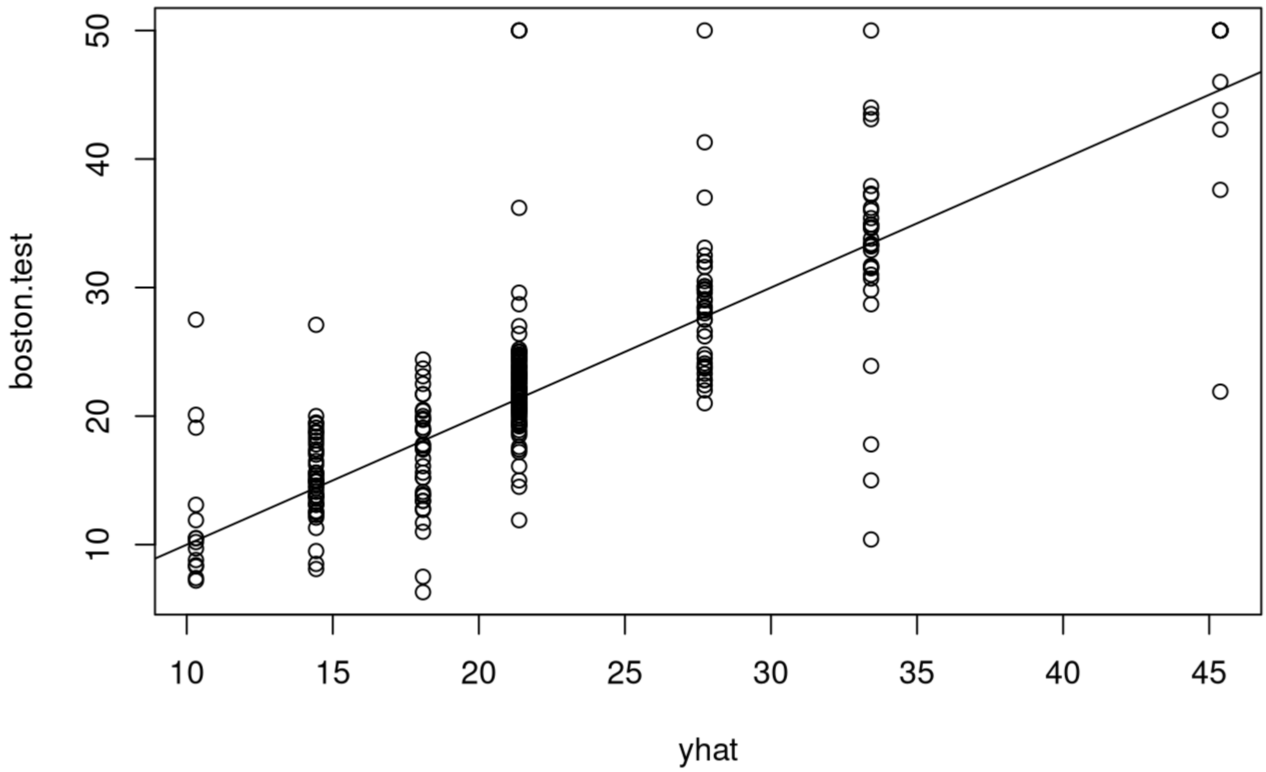
```
library(MASS)
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston)/2)

# medv: median value of owner-occupied homes in $1000s.
tree.boston <- tree(medv ~ ., Boston, subset = train)
summary (tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.38 = 2555 / 246
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

```
plot(tree.boston)
text(tree.boston)
```




```
# check the RSS of the prediction
yhat <- predict(tree.boston, newdata = Boston[-train, ])
boston.test <- Boston[-train, "medv"]
plot(yhat, boston.test)
abline(0, 1)
```




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

```
## [1] 35.28688
```

Tree ensembles

Implement bagging ourselves

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
set.seed(123)
Hitters[["salary.cat"]] <- salary.cat
Hitters[["binary.salary"]] <- factor(salary.cat == "(0,500]",
                                     labels = c(">500k", "<=500k"))
with(Hitters, table(binary.salary))
```

```
## binary.salary
##  >500k <=500k
##    112    151
```

```
with(Hitters, table(salary.cat))
```

```
## salary.cat
##       (0,500]     (500,750]   (750,1e+03] (1e+03,5e+03]
##           151            50            32            30
```

```
inTrain <- createDataPartition(Hitters[["salary.cat"]], p = 0.5)[[1]]
hit.train <- Hitters[inTrain,]
hit.test  <- Hitters[-inTrain,]

# single binary tree classification
tree.model <- tree(binary.salary ~ . - Salary - salary.cat, data = hit.train)
tree.preds <- predict(tree.model, newdata = hit.test)
tree.classified <- levels(Hitters[["binary.salary"]])[apply(tree.preds, 1, which.ma
x)]
tree.accuracy <- mean(tree.classified == hit.test[["binary.salary"]])
tree.accuracy
```

```
## [1] 0.8091603
```

```
# create bagging (a type of ensemble)
n.train <- nrow(hit.train)
bagging.predictions <- vapply(1:100, function(x) {
    idx <- sample(x = 1:n.train, size = n.train, replace = TRUE)
    tree.model <- tree(binary.salary ~ . - Salary - salary.cat,
                       data = hit.train[idx, ])
    predict(tree.model, newdata = hit.test)[,"<=500k"]
    }, numeric(nrow(hit.test)))

bagging.classified <- ifelse(rowMeans(bagging.predictions) > 0.5, "<=500k", ">500k")
bagging.accuracy <- mean(bagging.classified == hit.test[["binary.salary"]])
bagging.accuracy
```

```
## [1] 0.8549618
```

Bagging (use `ranger` package)

```
# Random forests will reduce into bagging if all features are used at every split
# Here we testing bagging by using random forest package and allowing the use of all
 features.
library(ranger)
set.seed(1)

# Bagging for classification
dim(Hitters)
```

```
## [1] 263  22
```

```
names(Hitters)
```

```
##  [1] "AtBat"          "Hits"           "HmRun"          "Runs"
##  [5] "RBI"            "Walks"          "Years"          "CAtBat"
##  [9] "CHits"          "CHmRun"         "CRuns"          "CRBI"
## [13] "CWalks"         "League"         "Division"       "PutOuts"
## [17] "Assists"        "Errors"         "Salary"         "NewLeague"
## [21] "salary.cat"     "binary.salary"
```

```
bag.hit <- ranger(binary.salary ~ . - Salary - salary.cat, data = hit.train,
                  mtry = 19)
summary(bag.hit)
```

```
##                           Length Class         Mode
## predictions               132    factor        numeric
## num.trees                   1    -none-        numeric
## num.independent.variables   1    -none-        numeric
## mtry                        1    -none-        numeric
## min.node.size               1    -none-        numeric
## prediction.error            1    -none-        numeric
## forest                      9    ranger.forest list
## confusion.matrix            4    table         numeric
## splitrule                   1    -none-        character
## treetype                    1    -none-        character
## call                        4    -none-        call
## importance.mode             1    -none-        character
## num.samples                 1    -none-        numeric
## replace                     1    -none-        logical
```

```
mean(predict(bag.hit, data = hit.test)[["predictions"]] == hit.test[["binary.salary"
]])
```

```
## [1] 0.8473282
```

```
# Bagging for regression
dim(Boston)
```

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

```
idx <- createDataPartition(Boston[["medv"]], p = 0.8)[[1L]]
boston.train <- Boston[idx, ]
boston.test <- Boston[-idx, ]

bag.boston <- ranger(medv ~ ., data = Boston,
                     mtry = 13)
summary(bag.boston)
```

```
##                           Length Class         Mode
## predictions               506    -none-        numeric
## num.trees                   1    -none-        numeric
## num.independent.variables   1    -none-        numeric
## mtry                        1    -none-        numeric
## min.node.size               1    -none-        numeric
## prediction.error            1    -none-        numeric
## forest                      7    ranger.forest list
## splitrule                   1    -none-        character
## treetype                    1    -none-        character
## r.squared                   1    -none-        numeric
## call                        4    -none-        call
## importance.mode             1    -none-        character
## num.samples                 1    -none-        numeric
## replace                     1    -none-        logical
```

```
mean((predict(bag.boston, data = boston.test)[["predictions"]] - boston.test)^2)
```

```
## Warning in mean.default((predict(bag.boston, data = boston.test)
## [["predictions"]] - : argument is not numeric or logical: returning NA
```

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

Random forest

```
set.seed(1)

# Random forest for classification
dim(iris)
```

```
## [1] 150   5
```

```
bag.iris <- ranger(Species ~ ., data = iris, mtry = 1)
print(bag.iris)
```

```
## Ranger result
##
## Call:
##  ranger(Species ~ ., data = iris, mtry = 1)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      150
## Number of independent variables:  4
## Mtry:                             1
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             4.67 %
```

```
# Random forest for regression
dim(Boston)
```

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

```
rf.boston <- ranger(medv~., data = Boston, subset = train, mtry = 6)
```

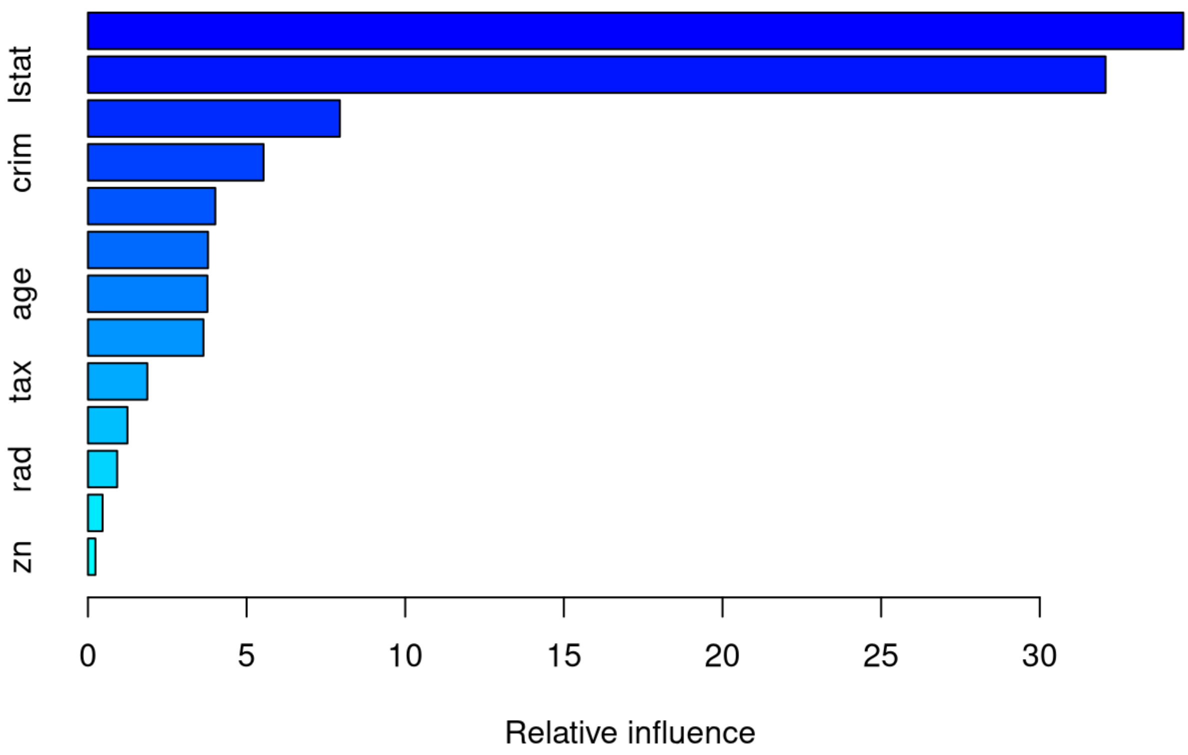
```
## Warning in ranger(medv ~ ., data = Boston, subset = train, mtry = 6): Unused
## arguments: subset
```

Boosting

```
# regression=
library(gbm)
```

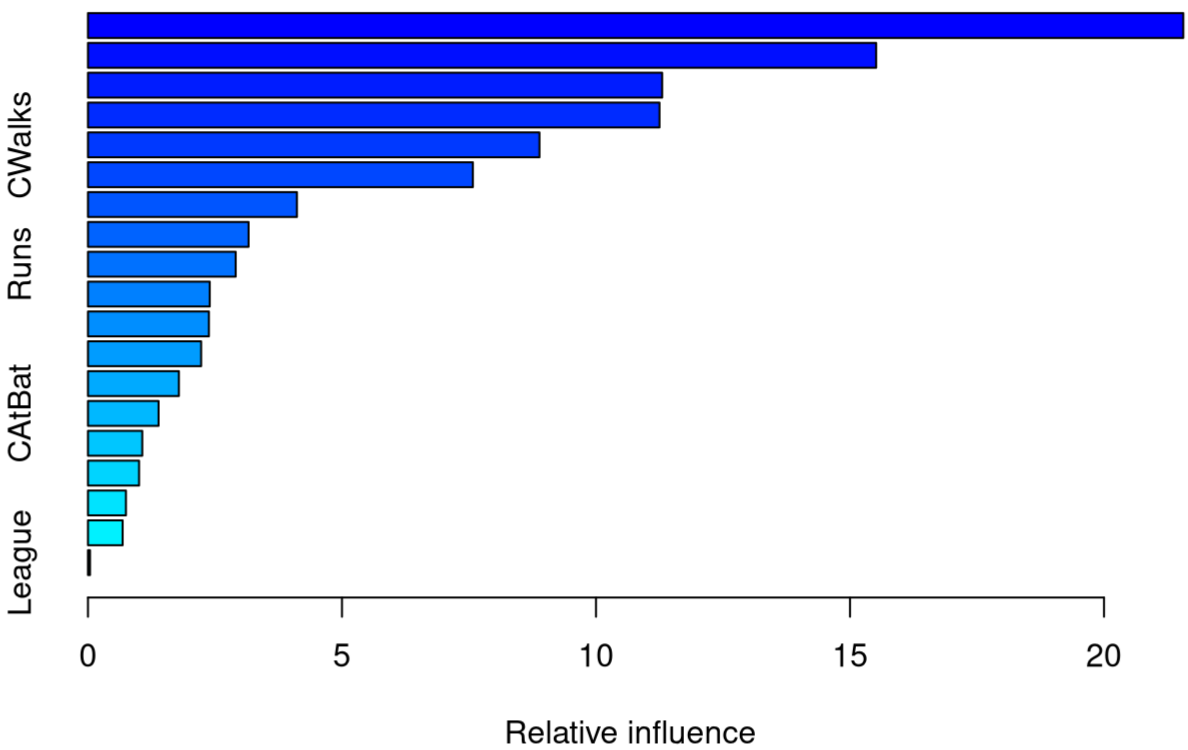
```
## Loaded gbm 2.1.8.1
```

```
set.seed(1)
boost.boston <- gbm(medv ~ ., data = Boston[train, ],
                    distribution = "gaussian", n.trees = 5000)
summary(boost.boston)
```



```
##                 var    rel.inf
## rm               rm 34.5235940
## lstat         lstat 32.0730829
## dis             dis  7.9394616
## crim           crim  5.5339488
## black         black  4.0130690
## ptratio ptratio  3.7836978
## age             age  3.7646124
## nox             nox  3.6395377
## tax             tax  1.8703031
## indus         indus  1.2439324
## rad             rad  0.9199469
## chas           chas  0.4607448
## zn               zn  0.2340686
```

```
# classification
adaBoost.model <- gbm(as.numeric(binary.salary) - 1 ~ . - Salary - salary.cat,
                      data = hit.train, distribution = "adaboost",
                      n.trees = 5000)
summary(adaBoost.model)
```



```
##                   var     rel.inf
## CHits           CHits 21.56041812
## Hits             Hits 15.51432443
## CRBI             CRBI 11.30231278
## CRuns           CRuns 11.24978881
## CWalks         CWalks  8.88705767
## PutOuts       PutOuts  7.57615788
## Years           Years  4.11295408
## AtBat           AtBat  3.16141049
## Runs             Runs  2.90773123
## Assists       Assists  2.39679154
## HmRun           HmRun  2.38027999
## NewLeague NewLeague  2.22759001
## RBI               RBI  1.78945279
## CAtBat         CAtBat  1.38958898
## Walks           Walks  1.06865094
## CHmRun         CHmRun  1.00563440
## Errors         Errors  0.74739700
## Division     Division  0.68238686
## League         League  0.04007202
```

```
preds <- predict(adaBoost.model, newdata = hit.test,
                 n.trees = 5000, type = "response")
mean(ifelse(preds > 0.5, "<=500k", ">500k") == hit.test[["binary.salary"]])
```

```
## [1] 0.8167939
```