Métodos de Agregación Homogéneos

Mathias Bourel

17/11/2018

```r
rm(list=ls())
```

1 - Paquetes que vamos/podemos usar

```r
library(rpart) #arboles
library(ipred) #bagging
library(randomForest)#radom forests
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
```

```r
#library(caret)#paquete con varias cosas de machine learning
```

2 - Regresión

```r
data=read.csv("housing_data.csv",sep=";", dec=",", header=T)
summary(data)
```

```
##       MEDV            CRIM                ZN             INDUS      
##  Min.   : 5.00   Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46  
##  1st Qu.:17.02   1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19  
##  Median :21.20   Median : 0.25651   Median :  0.00   Median : 9.69  
##  Mean   :22.53   Mean   : 3.61352   Mean   : 11.36   Mean   :11.14  
##  3rd Qu.:25.00   3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10  
##  Max.   :50.00   Max.   :88.97620   Max.   :100.00   Max.   :27.74  
##       CHAS              NOX               RM             AGE        
##  Min.   :0.00000   Min.   :0.3850   Min.   :3.561   Min.   :  2.90  
##  1st Qu.:0.00000   1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02  
##  Median :0.00000   Median :0.5380   Median :6.208   Median : 77.50  
##  Mean   :0.06917   Mean   :0.5547   Mean   :6.285   Mean   : 68.57  
##  3rd Qu.:0.00000   3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08  
##  Max.   :1.00000   Max.   :0.8710   Max.   :8.780   Max.   :100.00  
##       DIS              RAD              TAX           PTRATIO     
##  Min.   : 1.130   Min.   : 1.000   Min.   :187.0   Min.   :12.60  
##  1st Qu.: 2.100   1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40  
##  Median : 3.207   Median : 5.000   Median :330.0   Median :19.05  
##  Mean   : 3.795   Mean   : 9.549   Mean   :408.2   Mean   :18.46  
##  3rd Qu.: 5.188   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20  
##  Max.   :12.127   Max.   :24.000   Max.   :711.0   Max.   :22.00  
##        B              LSTAT      
##  Min.   :  0.32   Min.   : 1.73  
##  1st Qu.:375.38   1st Qu.: 6.95  
##  Median :391.44   Median :11.36  
```

```
##  Mean   :356.67   Mean   :12.65
##  3rd Qu.:396.23   3rd Qu.:16.95
##  Max.   :396.90   Max.   :37.97
```

```
names(data)
```

```
##  [1] "MEDV"    "CRIM"    "ZN"      "INDUS"   "CHAS"    "NOX"     "RM"
##  [8] "AGE"     "DIS"     "RAD"     "TAX"     "PTRATIO" "B"       "LSTAT"
```

```
data$CHAS=as.factor(data$CHAS)
summary(data$CHAS)
```

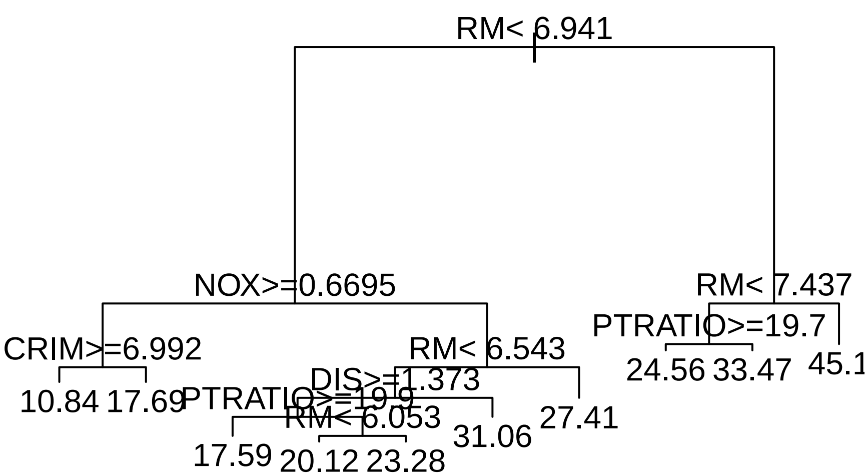
```
##   0   1
## 471  35
```

Construir un árbol de regresión considerando las variables CRIM, ZN, CHAS, NOX, RM, DIS, TAX y PTRATIO. Hallar el arbol optimo dado por el algoritmo de la poda con el minimo error de validación cruzada.

Hallar el arbol optimo dado por el algoritmo de la poda con la regla 1-SE.

Hallar el arbol optimo con paramétro de costo complejidad igual a cp=0.01 (es el que devuelve la función rpart)

```
tree=rpart(MEDV~CRIM+ZN+CHAS+NOX+RM+DIS+TAX+PTRATIO,cp=0.01,data)
cp.opt = tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
tree2=rpart(MEDV~CRIM+ZN+CHAS+NOX+RM+DIS+TAX+PTRATIO,cp=cp.opt,data)
tsse=min(tree$cptable[,"xerror"])+tree$cptable[which.min(tree$cptable[,"xerror"]),"xstd"]
cpopt1SE=tree$cptable[min(tree$cptable[tree$cptable[,"xerror"]<tsse,"nsplit"])+1,"CP"]
tree3=rpart(MEDV~CRIM+ZN+CHAS+NOX+RM+DIS+TAX+PTRATIO,cp=cpopt1SE,data)
plot(tree,margin=0.1)
text(tree)
```



Calcule el error del árbol sobre la muestra de entrenamietno

```
pred=predict(tree)
error.tree.lrn= sqrt(mean((data[,1]-pred)^2))
error.tree.lrn
```

```
## [1] 4.301118
```

```
MEDV=data$MEDV
error.tree.lrn/mean(MEDV)
```

```
## [1] 0.1908825
```

Realize el error sobre la muestra de prueba (media y SD de 20 iteraciones y partiendo la muestra en 2/3 - 1/3)

```
K=20
    error.tree=list(matrix(NA, K))
    n = nrow(data)
    for(k in 1:K)   {
        smp=sample(n,round(n/3))
        learn=data[-smp,]
        test=data[smp,]
            tree.lrn=rpart(MEDV~CRIM+ZN+CHAS+NOX+RM+DIS+TAX+PTRATIO,cp=0.01,learn)
            pred=predict(tree.lrn,test)
            error.tree[k] = sqrt(mean((test[,1]-pred)^2))

    }

mean.error.tree=mean(unlist(error.tree))
sd.error.tree=sd(unlist(error.tree))
mean.error.tree
```

```
## [1] 5.270967
```

```
sd.error.tree
```

```
## [1] 0.7265018
```

2-1 BAGGING

Construya un modelo Bagging con la función bagging del paquete ipred. Pida que el estimador se construya con 25 remuestras bootstrap y que la salida le brinde el error de las observaciones "out of bag". Indique que para la construcción de cada árbol se tome en cuenta un cp=0.01. Realice el gráfico del error cuadrático medio en función de la cantidad de arboles. Obtenga el error por muestra de prueba y su desviación estándar de la misma forma que se hizo en CART.

```
bag=ipredbagg(data[,1], data[,c(2,3,5,6,7,9,11,12)], control=rpart.control(cp = 0.01),nbagg=25, coob=F)
bag
```

```
##
## Bagging regression trees with 25 bootstrap replications
```

```
summary(bag)
```

```
##        Length Class      Mode
## y      506    -none-     numeric
## X        8    data.frame list
## mtrees  25    -none-     list
## OOB      1    -none-     logical
## comb     1    -none-     logical
```

```
print(bag)
```

```
##
## Bagging regression trees with 25 bootstrap replications
```

```
bag2=bagging(MEDV ~ CRIM + ZN + CHAS + NOX + RM + DIS  + TAX + PTRATIO,control=rpart.control(cp=0.01),nb
```

```r
bag3=ipredbagg(data[,1], data[,c(2,3,5,6,7,9,11,12)], control=rpart.control(cp = 0.01),nbagg=100, coob=
bag3
```

```
##
## Bagging regression trees with 100 bootstrap replications
## Out-of-bag estimate of root mean squared error:  4.8044
```

```r
K=20
    error.bag=list(matrix(NA, K))
    n = nrow(data)
    for(k in 1:K)   {
        smp=sample(n,round(n/3))
        learn=data[-smp,]
        test=data[smp,]
            bag.lrn=ipredbagg(learn[,1], learn[,c(2,3,5,6,7,9,11,12)], control=rpart.control(cp = 0.01)
            pred=predict(tree.lrn,test)
            error.bag[k] = sqrt(mean((test[,1]-pred)^2))

        }

mean.error.bag=mean(unlist(error.bag))
sd.error.bag=sd(unlist(error.bag))
mean.error.bag
```

```
## [1] 4.82587
```

```r
sd.error.bag
```

```
## [1] 0.4885358
```

Observaciones: 1) Se puede encontrar más información en la página: http://www.inside-r.org/packages/cran/ipred/docs/bagging

2) Realizar las mismas simulaciones con el paquete caret, el paquete randomForest, el paquete adabag (mirar los helps)

2-2 RANDOM FORESTS

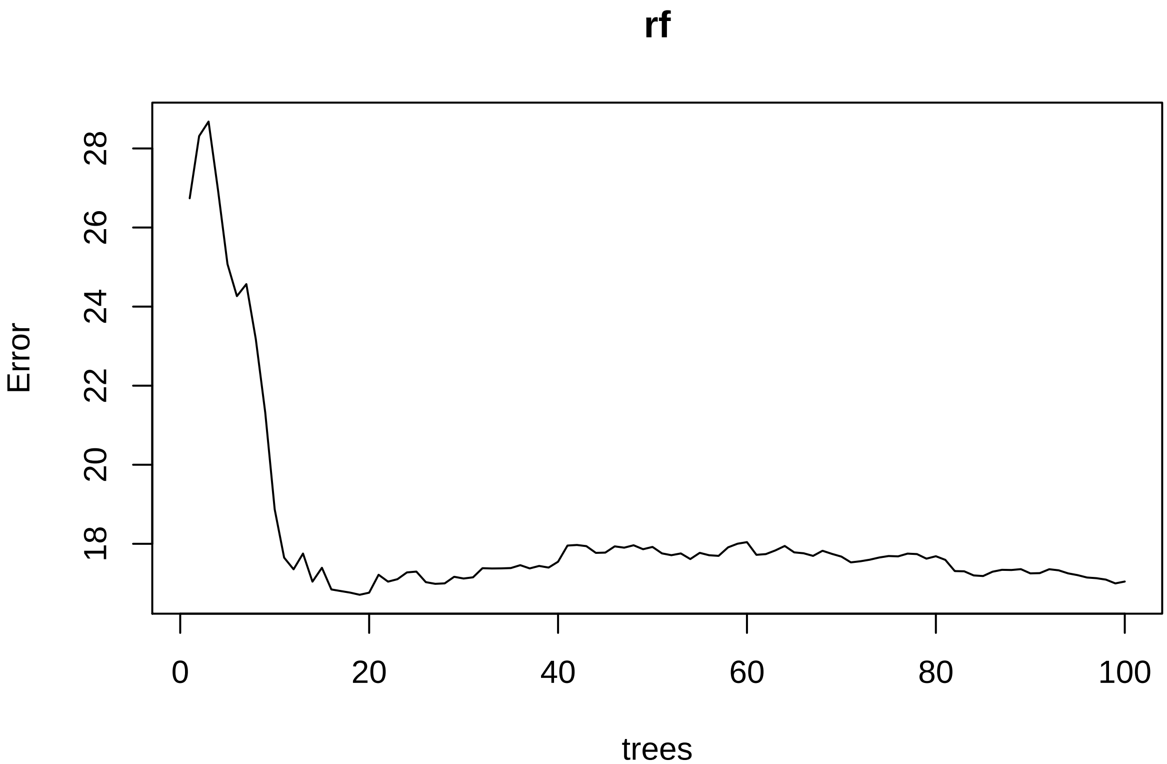
Misma consigna que en 2.1, ahora con la función randomForest. Construya un modelo Random Forests con 100 árboles A partir de la salida del modelo indique cuántas variables se utilizaron en cada partición y cuál es el porcentaje de varianza explicada por el modelo. Realice el gráfico de número de árboles vs el error cuadrático medio. Entiende que considerar 100 árboles para construir el bosque es adecuado? Obtenga el gráfico de importancia de variables. Obtenga el error por muestra de prueba de la misma forma que se hizo antes.

```r
#?randomForest
rf=randomForest(MEDV~CRIM+ZN+CHAS+NOX+RM+DIS+TAX+PTRATIO,ntree=100,na.action=na.omit,data)
rf
```

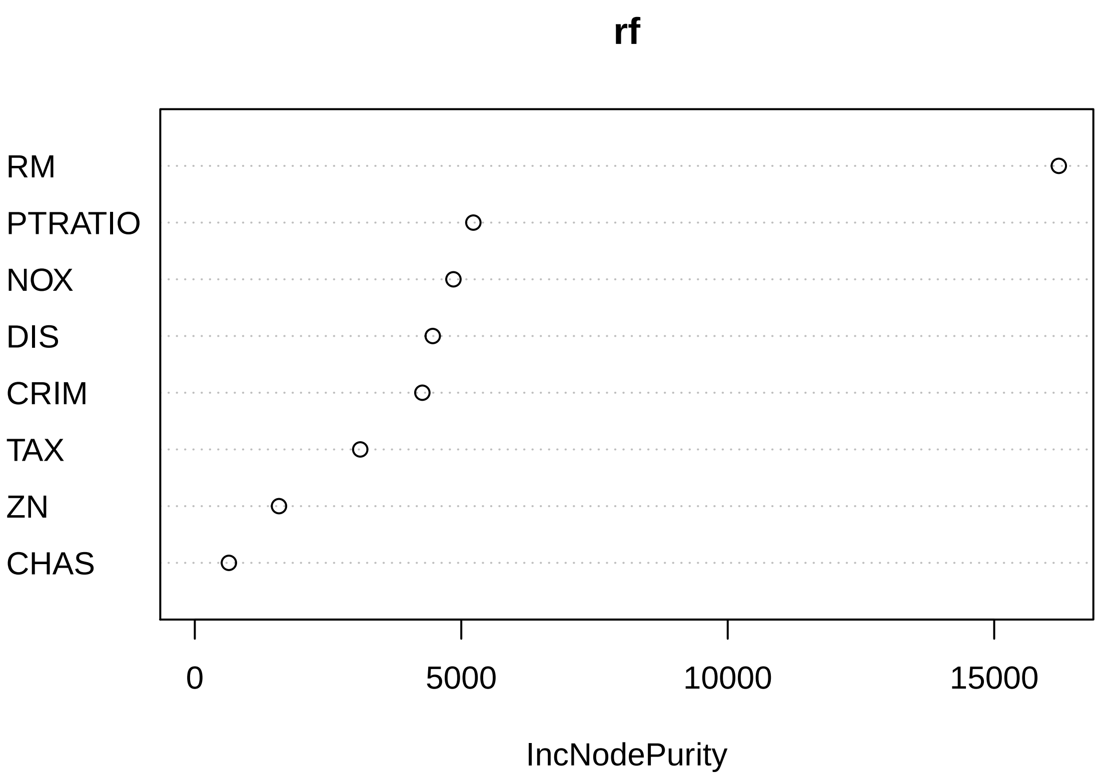
```
##
## Call:
##  randomForest(formula = MEDV ~ CRIM + ZN + CHAS + NOX + RM + DIS +      TAX + PTRATIO, data = data, 
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
```

```
##           Mean of squared residuals: 17.04621
##                     % Var explained: 79.81
```

```r
plot(rf)
```



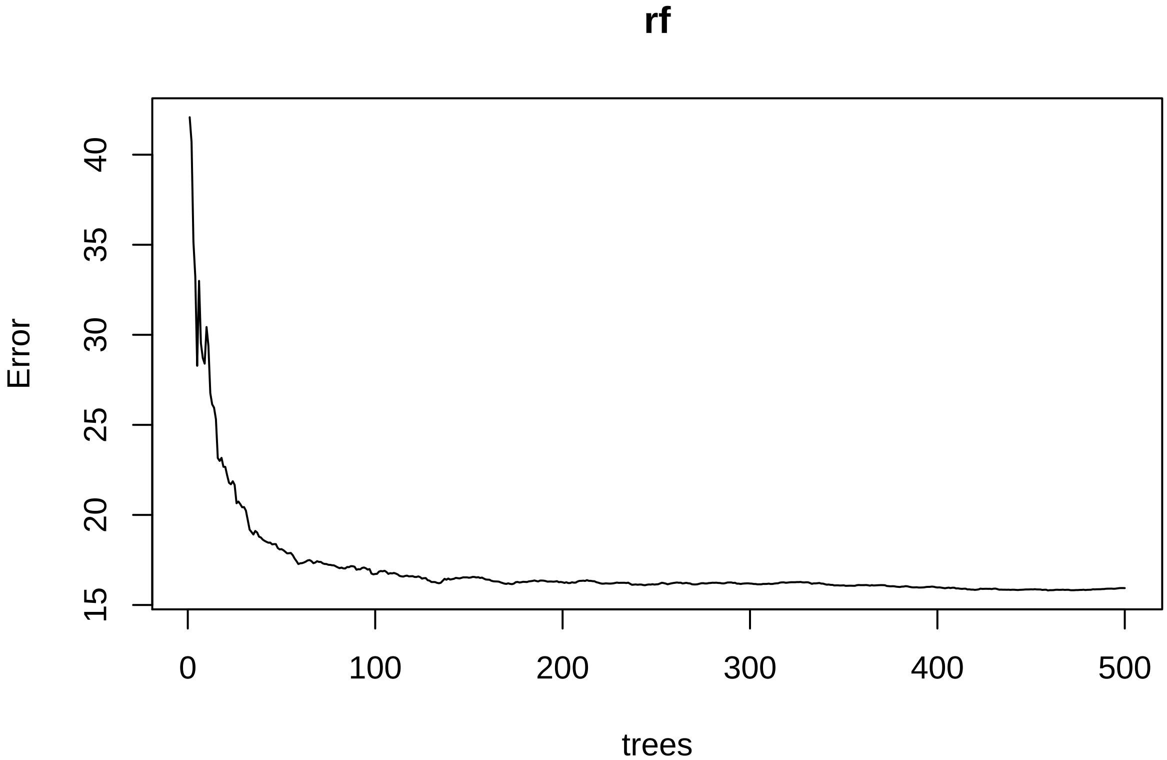

```r
varImpPlot(rf)
```



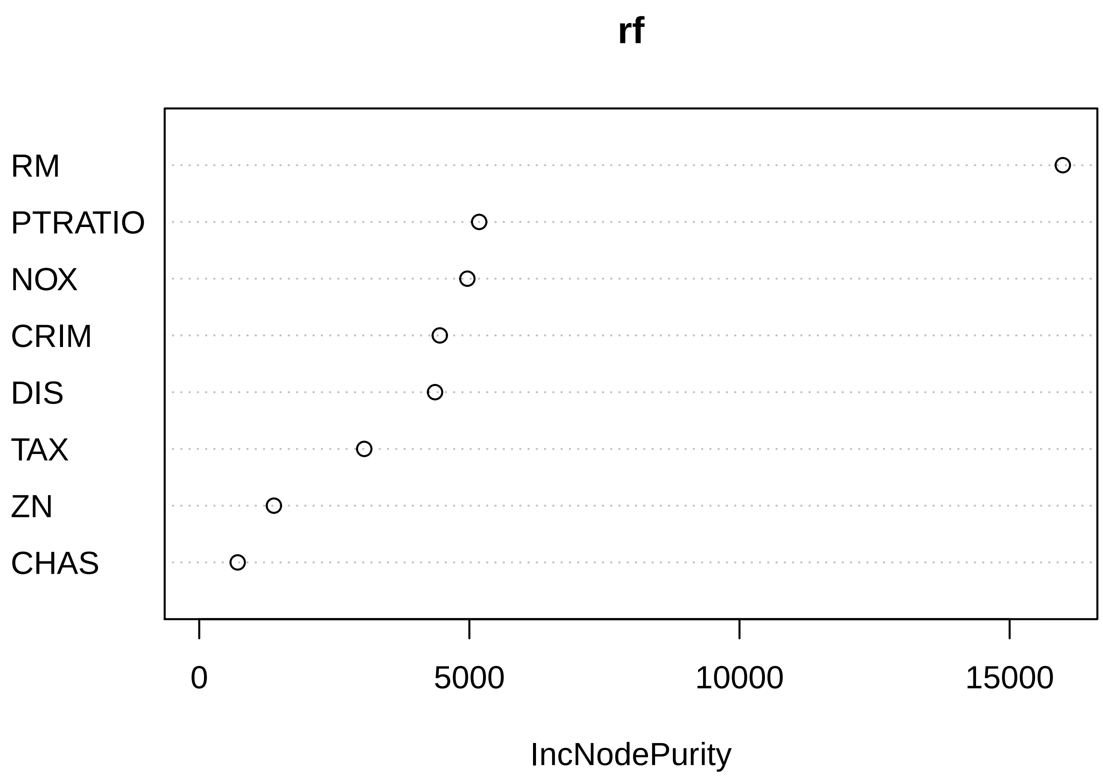
```r
#consideramos ahora 500 arboles
rf=randomForest(MEDV~CRIM+ZN+CHAS+NOX+RM+DIS+TAX+PTRATIO,ntree=500,na.action=na.omit,data)
rf
```

```
##
## Call:
##  randomForest(formula = MEDV ~ CRIM + ZN + CHAS + NOX + RM + DIS +      TAX + PTRATIO, data = data, n
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 15.93794
##                     % Var explained: 81.12
```

```r
plot(rf)
```




```r
varImpPlot(rf)
```




```
K=20
    error.rf=list(matrix(NA, K))
    n = nrow(data)
    for(k in 1:K)   {
        smp=sample(n,round(n/3))
        learn=data[-smp,]
        test=data[smp,]
            rf.lrn=randomForest(MEDV~CRIM+ZN+CHAS+NOX+RM+DIS+TAX+PTRATIO,ntree=500,na.action=na.omit,lea
            pred=predict(rf.lrn,test)
            error.rf[k] = sqrt(mean((test[,1]-pred)^2))

        }

mean.error.rf=mean(unlist(error.rf))
sd.error.rf=sd(unlist(error.rf))
mean.error.rf
```

```
## [1] 4.066656
```

```
sd.error.rf
```

```
## [1] 0.6112092
```

```
errores=c(mean.error.tree,mean.error.bag,mean.error.rf)
sd=c(sd.error.tree,sd.error.bag,sd.error.rf)
tabla=cbind(errores,sd)
rownames(tabla) = c("CART","BAGGING","RF")
tabla
```

```
##          errores        sd
## CART    5.270967 0.7265018
## BAGGING 4.825870 0.4885358
```

```
## RF       4.066656 0.6112092
```

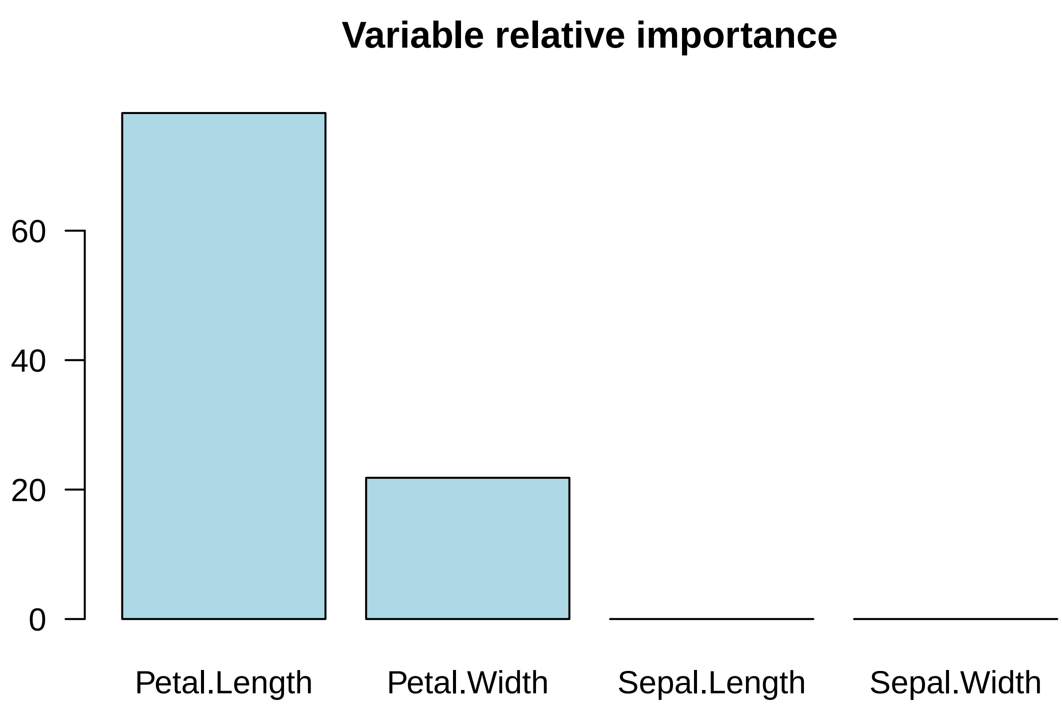
3- CLASIFICACIÓN

3-1 Boosting

```r
library(adabag)
```

```
## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

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

## Loading required package: foreach

## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel

##
## Attaching package: 'adabag'

## The following object is masked from 'package:ipred':
##
##     bagging
```

```r
data(iris)
iris.adaboost <- boosting(Species~., data=iris, boos=TRUE,
                          mfinal=3)
importanceplot(iris.adaboost)
```




```r
#Ver como se calcula esta importancia de variables.
```

3-2 Comparación.

Implementar CART, RF y BOOSTING y comparar sus errores de clasificación considerando un error por muestra de prueba con 20 iteraciones y los demás valores por defecto que usamos para regresión.

```r
#Vamos a recodificar la variable MEDV

data$MEDVcat <- ifelse(data$MEDV > 20, c("high"), c("low"))
summary(data)
```

```
##       MEDV            CRIM                ZN             INDUS
##  Min.   : 5.00   Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46
##  1st Qu.:17.02   1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19
##  Median :21.20   Median : 0.25651   Median :  0.00   Median : 9.69
##  Mean   :22.53   Mean   : 3.61352   Mean   : 11.36   Mean   :11.14
##  3rd Qu.:25.00   3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10
##  Max.   :50.00   Max.   :88.97620   Max.   :100.00   Max.   :27.74
##  CHAS        NOX               RM             AGE
##  0:471   Min.   :0.3850   Min.   :3.561   Min.   :  2.90
##  1: 35   1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02
##          Median :0.5380   Median :6.208   Median : 77.50
##          Mean   :0.5547   Mean   :6.285   Mean   : 68.57
##          3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08
##          Max.   :0.8710   Max.   :8.780   Max.   :100.00
##       DIS              RAD              TAX           PTRATIO
##  Min.   : 1.130   Min.   : 1.000   Min.   :187.0   Min.   :12.60
##  1st Qu.: 2.100   1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40
##  Median : 3.207   Median : 5.000   Median :330.0   Median :19.05
##  Mean   : 3.795   Mean   : 9.549   Mean   :408.2   Mean   :18.46
```

```
##  3rd Qu.: 5.188   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20
##  Max.   :12.127   Max.   :24.000   Max.   :711.0   Max.   :22.00
##        B              LSTAT          MEDVcat
##  Min.   :  0.32   Min.   : 1.73   Length:506
##  1st Qu.:375.38   1st Qu.: 6.95   Class :character
##  Median :391.44   Median :11.36   Mode  :character
##  Mean   :356.67   Mean   :12.65
##  3rd Qu.:396.23   3rd Qu.:16.95
##  Max.   :396.90   Max.   :37.97
```

```
MEDVcat=as.factor(data$MEDVcat)
data=data[,-1]
data=cbind(data,MEDVcat)
summary(data)
```

```
##       CRIM                ZN             INDUS       CHAS
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1: 35
##  Median : 0.25651   Median :  0.00   Median : 9.69
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74
##       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           B
##  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          MEDVcat            MEDVcat
##  Min.   : 1.73   Length:506         high:291
##  1st Qu.: 6.95   Class :character   low :215
##  Median :11.36   Mode  :character
##  Mean   :12.65
##  3rd Qu.:16.95
##  Max.   :37.97
```

3-3 Curvas ROC

```
library(rpart)
library(rattle) # para data set
```

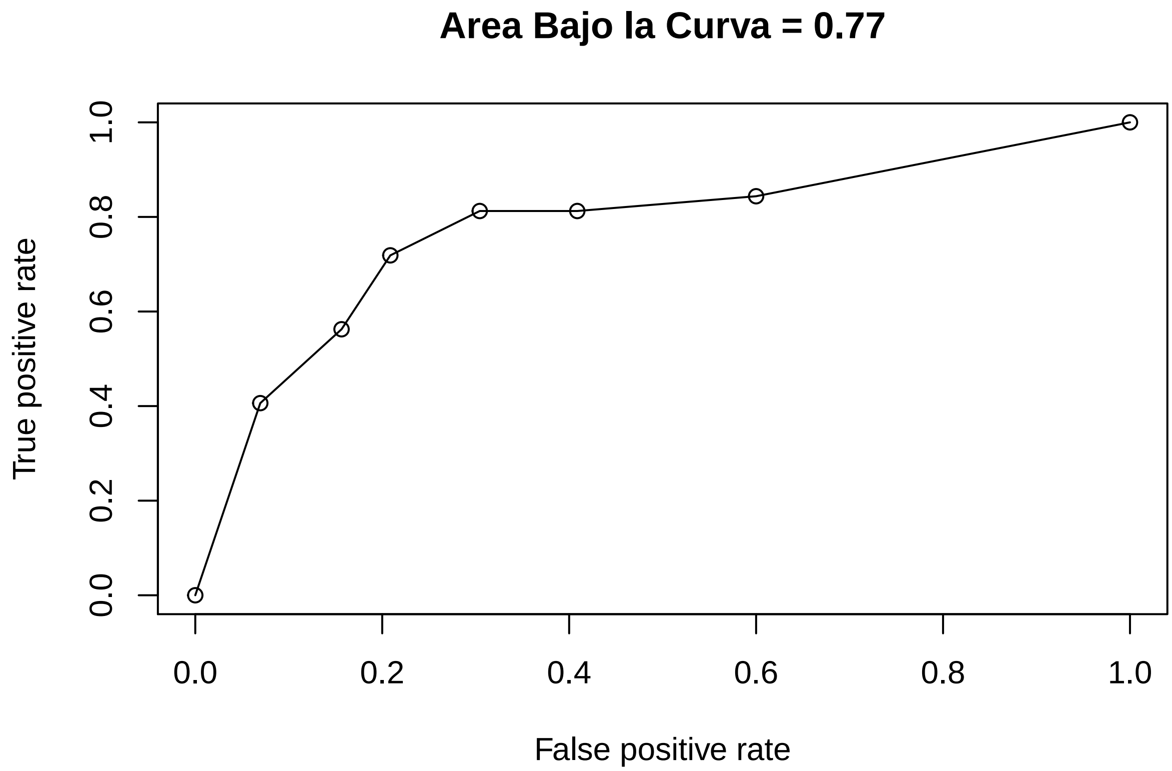
```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
## The following object is masked from 'package:randomForest':
##
##     importance
```

```r
library(ROCR)
```

```
## Loading required package: gplots
##
## Attaching package: 'gplots'
## The following object is masked from 'package:stats':
##
##     lowess
```

```r
datos          <- weather
datos          <- within(datos, rm("Date","Location","RISK_MM")) #borra columnas dummy
set.seed(42)  #  fija la secuencia de numeros aleatorios
sampleTrain    <- sample(nrow(datos),(nrow(datos)*.6))
Train          <- datos[sampleTrain,]
Test           <- datos[-sampleTrain,]

modelo.rpart  <- rpart(RainTomorrow ~ .,Train, method="class")


# PREDICCION
#-----------------------------------------------------------------------------
predict.rpart <- predict(modelo.rpart,Test,type = "prob")[,2] #prob. clase=yes
predict.rocr  <- prediction (predict.rpart,Test$RainTomorrow)
perf.rocr     <- performance(predict.rocr,"tpr","fpr") #True y False postivie.rate


# GRAFICO CURVA ROC
#-----------------------------------------------------------------------------
auc <- as.numeric(performance(predict.rocr ,"auc")@y.values)
plot(perf.rocr,type='o', main = paste('Area Bajo la Curva =',round(auc,2)))
```


Area Bajo la Curva = 0.77
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate

```
#abline(a=0, b= 1)
```