**Listado de librerías en R para hacer Machine Learning:**

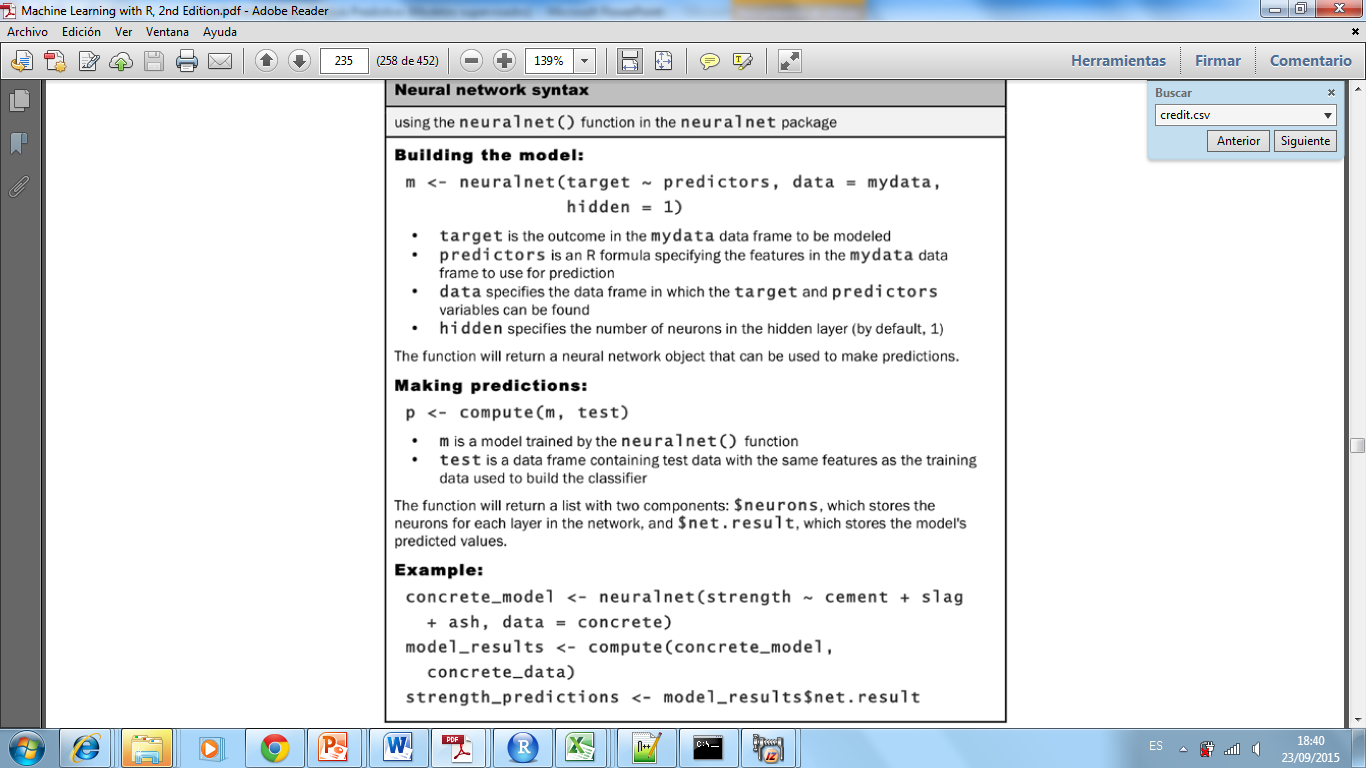
1. **REDES NEURONALES**

Library(nnet): feed-forward neural networks with a single hidden layer

Multiple hidden layers:

[RSNNS](http://cran.r-project.org/web/packages/RSNNS/RSNNS.pdf)

[neuralnet](http://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf)



<https://beckmw.wordpress.com/2013/03/04/visualizing-neural-networks-from-the-nnet-package/>

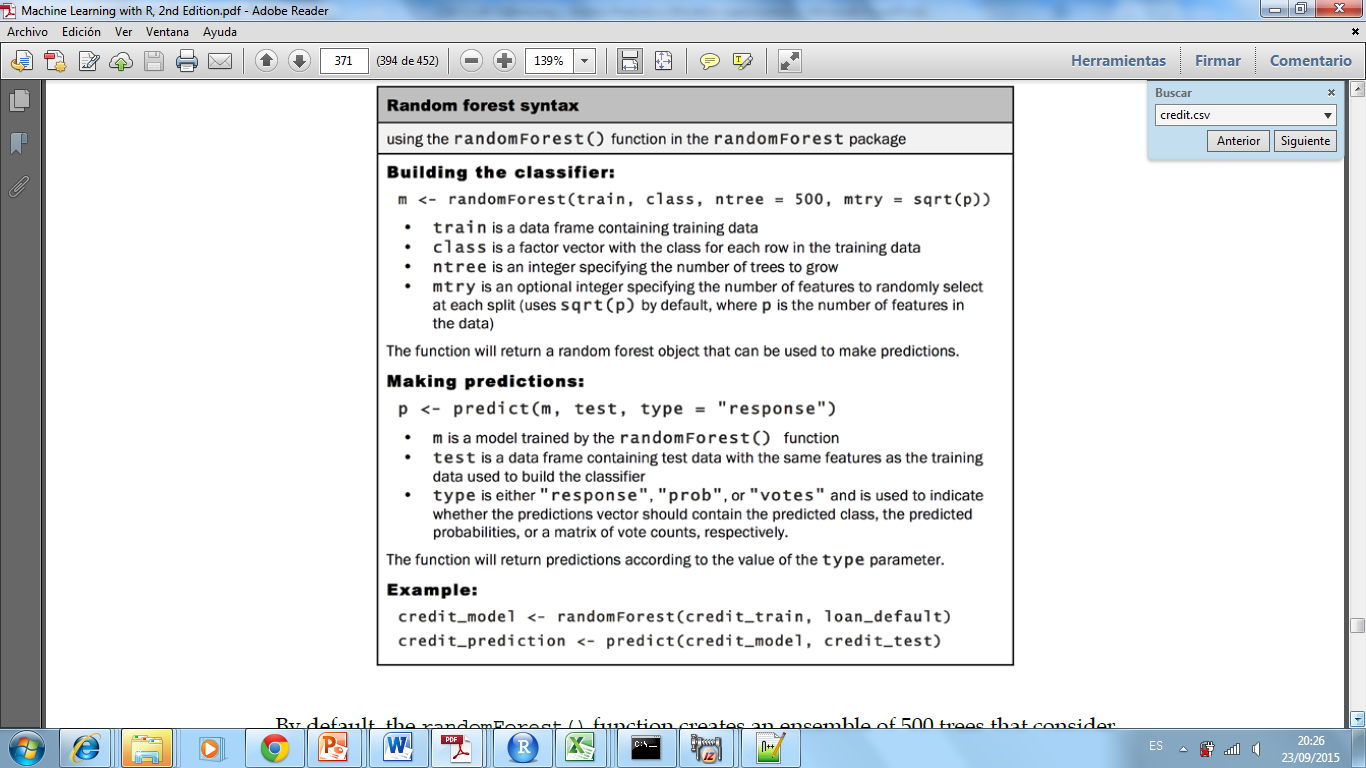
http://journal.r-project.org/archive/2010-1/RJournal\_2010-1\_Guenther+Fritsch.pdf

<http://www.louisaslett.com/Courses/Data_Mining/ST4003-Lab5-Introduction_to_Neural_Networks.pdf>

1. **RANDOM FOREST**

library(randomForest)

Función randomForest

****

Paralelización del cálculo:

RF es una colección de árboles independientes entrenados sobre muestras de observaciones y variables aleatorias

Que cada core de nuestro ordenador entrene unos árboles diferentes

La función combine del paquete randomForest los puede juntar

Ejemplo “de juguete” (hay que utilizar matrices en vez de fórmula)

library(doMC)

library(randomForest)

registerDoMC(2) #número de nucleos en la máquina

runs <- 10

set.seed(123)

a <- plyr::llply(seq\_len(runs),

function(x){

rf <- randomForest(Species ~ ., iris, ntree=50, norm.votes=FALSE)

rf

}, .parallel= TRUE)

result <- do.call("combine",a)

Result

http://stats.stackexchange.com/questions/37370/random-forest-computing-time-in-r

Otras opciones de paralelización en R: foreach

<https://cran.r-project.org/web/packages/foreach/vignettes/foreach.pdf>

#Paralelización

Y<-iris[,5]

X<-iris[,-5]

library(randomForest)

library(foreach)

#Secuencial

rf <- foreach(ntree=rep(50, 10), .combine=combine) %do% randomForest(X, Y, ntree=ntree)

#Paralelización

rf <- foreach(ntree=rep(50, 10), .combine=combine, .packages='randomForest') %dopar% randomForest(X, Y, ntree=ntree)

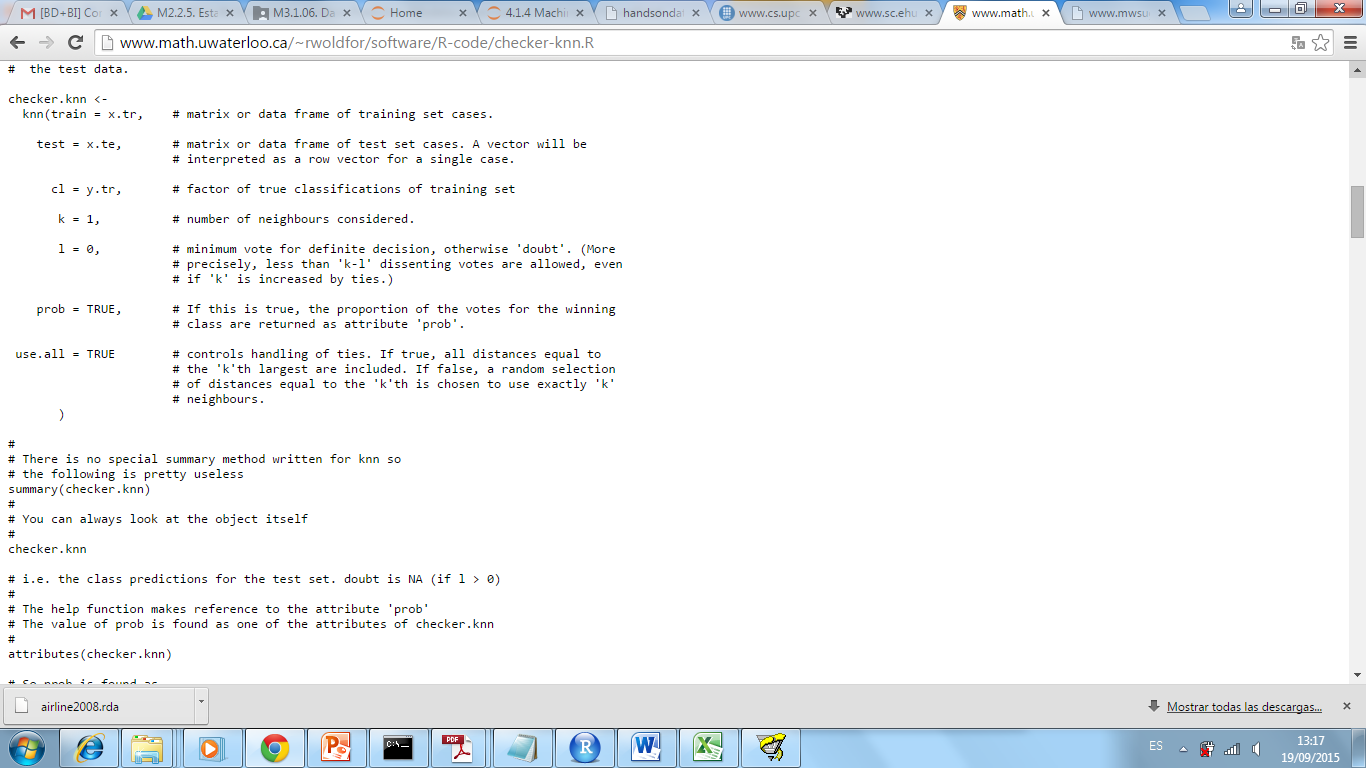
Big Data: Una implementación Rhadoop

https://github.com/cloudera/poisson\_sampling/blob/master/src/fitRandomForest.R

1. **K VECINOS**

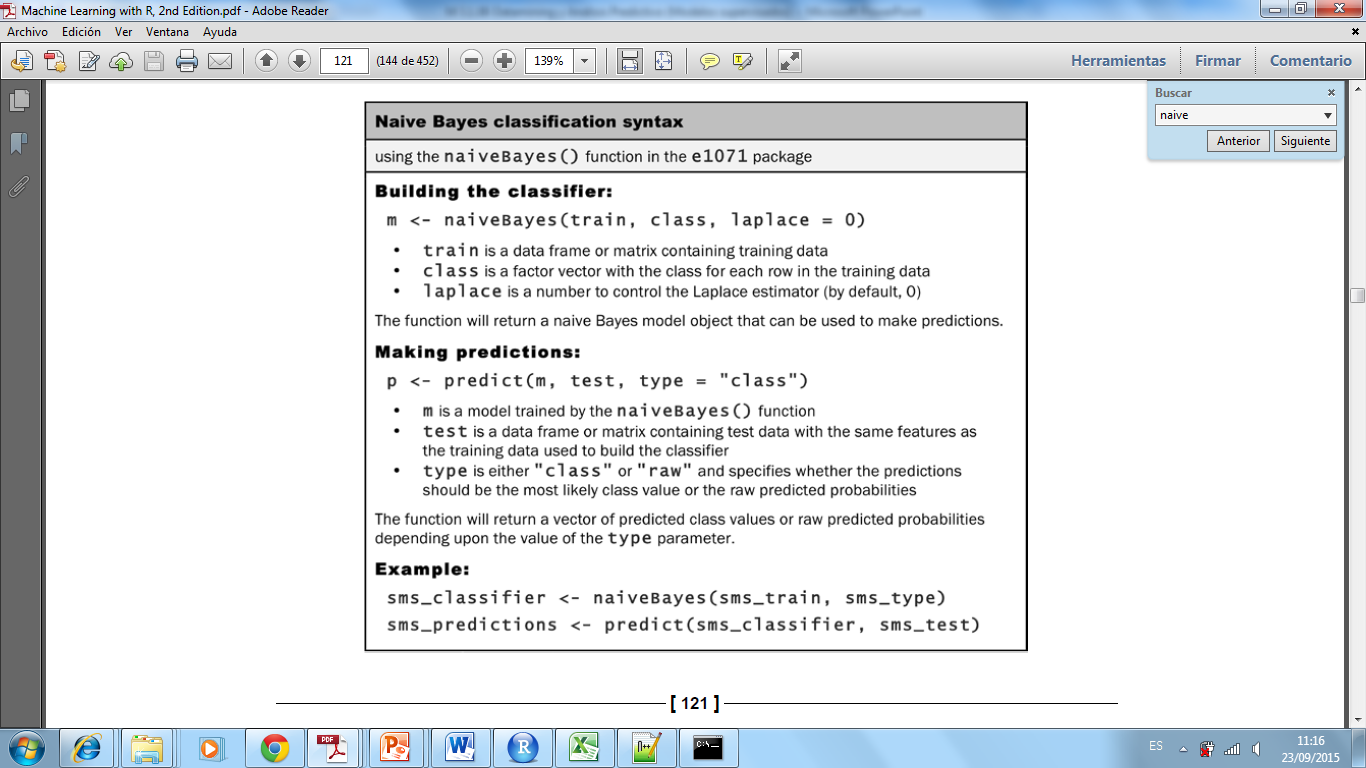
library(class)

Función knn y knn.cv



1. **NAIVE BAYES**

naiveBayes((formula, data, laplace = 0, ..., subset, na.action = na.pass))



Librería klaR

Función NaiveBayes

1. **SVM**

library(e1071)

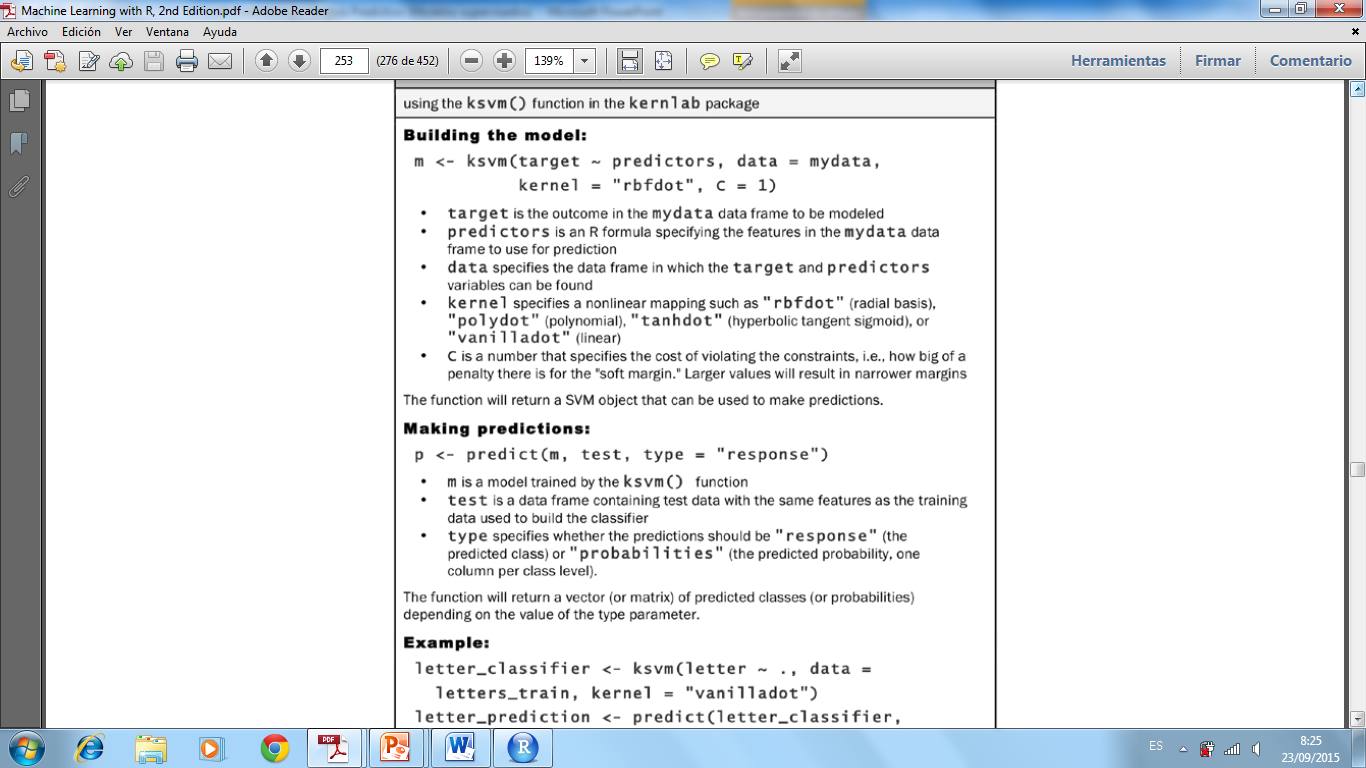
Función svm

Library(kernlab)

Función ksvm

Library(LIBSVM)

Library(klaR)



1. **ÁRBOLES DE DECISIÓN y CLASIFICACIÓN**

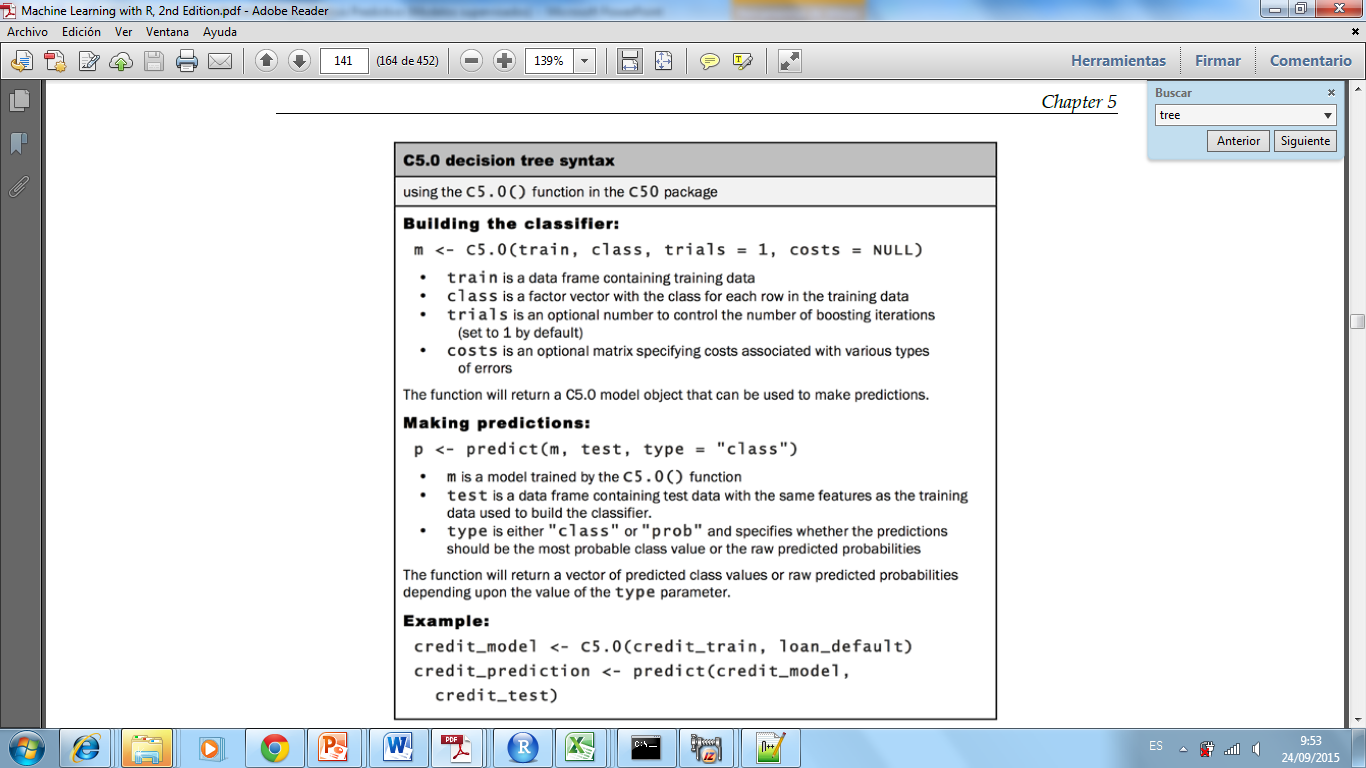
**library(rpart) CART**

****

**Cortar un árbol: prune**

**library(ggdendro) para dibujar árboles**

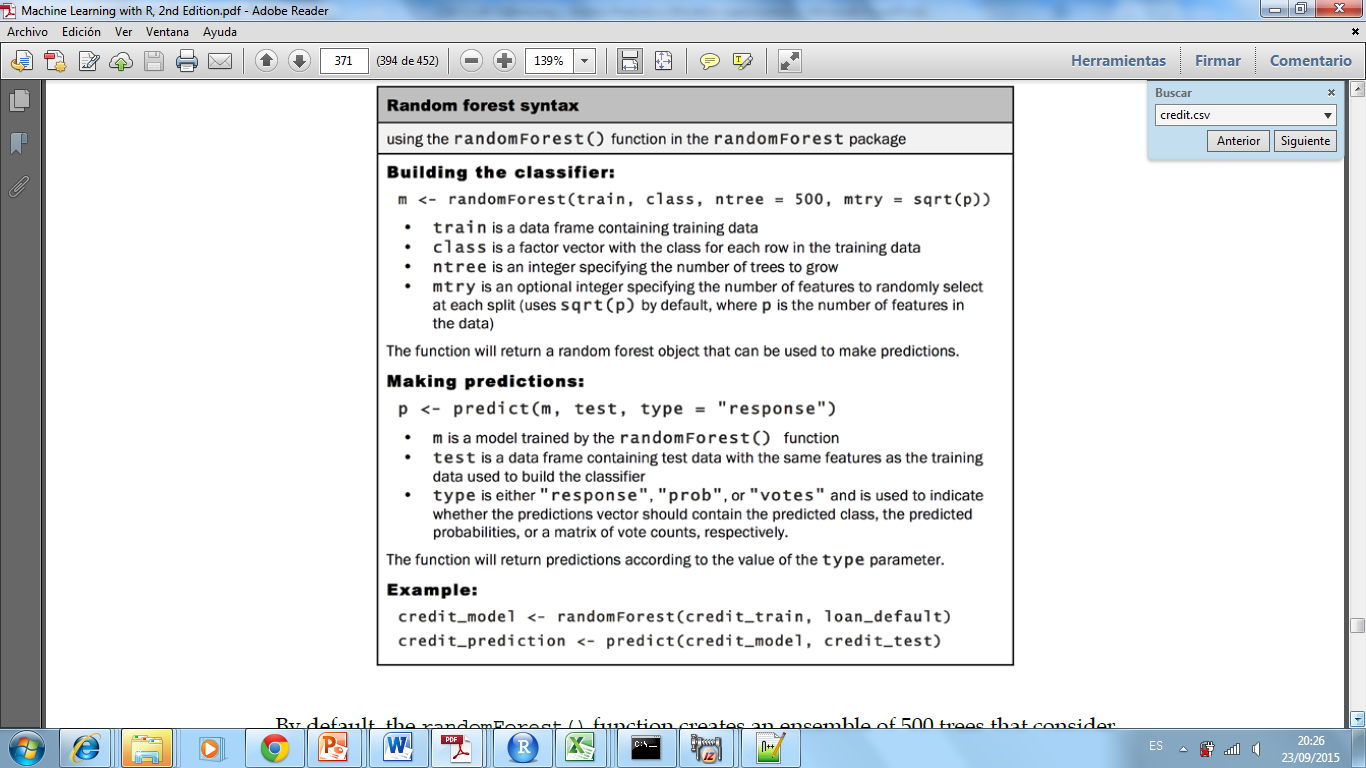
**library(C50) C5 Permite introducir costes**

****

1. **RANDOM FOREST**

**library(randomForest)**

**Función randomForest**

****

Otras opciones de paralelización en R: foreach

<https://cran.r-project.org/web/packages/foreach/vignettes/foreach.pdf>

#Paralelización

Y<-iris[,5]

X<-iris[,-5]

library(randomForest)

library(foreach)

#Secuencial

rf <- foreach(ntree=rep(50, 10), .combine=combine) %do% randomForest(X, Y, ntree=ntree)

#Paralelización

rf <- foreach(ntree=rep(50, 10), .combine=combine, .packages='randomForest') %dopar% randomForest(X, Y, ntree=ntree)

Ejemplo “de juguete” (hay que utilizar matrices en vez de fórmula)

library(doMC)

library(randomForest)

registerDoMC(2) #número de nucleos en la máquina

runs <- 10

set.seed(123)

a <- plyr::llply(seq\_len(runs),

function(x){

rf <- randomForest(Species ~ ., iris, ntree=50, norm.votes=FALSE)

rf

}, .parallel= TRUE)

result <- do.call("combine",a)

Result

http://stats.stackexchange.com/questions/37370/random-forest-computing-time-in-r

Big Data: Una implementación Rhadoop

https://github.com/cloudera/poisson\_sampling/blob/master/src/fitRandomForest.R