Evaluación MLII: Ejercicio 1

Boosting, Aprendizaje Supervisado Secuencial, Selección de Atributos

Inmaculada Perea Fernández

junio 2017

Carga de librerías

```
if (!require('kernlab')) install.packages('kernlab'); library('kernlab')
if (!require('adabag')) install.packages('adabag'); library('adabag')
if (!require('caret')) install.packages('caret'); library('caret')
```

Establecimiento de la semilla

```
set.seed(123456789)
```

Carga, inspección y preparación de los datos

El conjunto de datos *spam* consta de 4601 observaciones y 58 variables:

```
data(spam)
dim(spam)
```

```
## [1] 4601   58
```

```
summary(spam)
```

```
##       make           address            all             num3d        
##  Min.   :0.0000   Min.   : 0.000   Min.   :0.0000   Min.   : 0.00000  
##  1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 0.00000  
##  Median :0.0000   Median : 0.000   Median :0.0000   Median : 0.00000  
##  Mean   :0.1046   Mean   : 0.213   Mean   :0.2807   Mean   : 0.06542  
##  3rd Qu.:0.0000   3rd Qu.: 0.000   3rd Qu.:0.4200   3rd Qu.: 0.00000  
##  Max.   :4.5400   Max.   :14.280   Max.   :5.1000   Max.   :42.81000  
##       our              over             remove          internet      
##  Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.0000  
##  1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0000  
##  Median : 0.0000   Median :0.0000   Median :0.0000   Median : 0.0000  
##  Mean   : 0.3122   Mean   :0.0959   Mean   :0.1142   Mean   : 0.1053  
##  3rd Qu.: 0.3800   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.: 0.0000  
##  Max.   :10.0000   Max.   :5.8800   Max.   :7.2700   Max.   :11.1100  
##      order             mail            receive            will        
##  Min.   :0.00000   Min.   : 0.0000   Min.   :0.00000   Min.   :0.0000  
##  1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.:0.00000   1st Qu.:0.0000  
##  Median :0.00000   Median : 0.0000   Median :0.00000   Median :0.1000  
##  Mean   :0.09007   Mean   : 0.2394   Mean   :0.05982   Mean   :0.5417  
##  3rd Qu.:0.00000   3rd Qu.: 0.1600   3rd Qu.:0.00000   3rd Qu.:0.8000  
##  Max.   :5.26000   Max.   :18.1800   Max.   :2.61000   Max.   :9.6700  
##      people            report          addresses           free        
##  Min.   :0.00000   Min.   : 0.00000   Min.   :0.0000   Min.   : 0.0000  
##  1st Qu.:0.00000   1st Qu.: 0.00000   1st Qu.:0.0000   1st Qu.: 0.0000  
##  Median :0.00000   Median : 0.00000   Median :0.0000   Median : 0.0000  
##  Mean   :0.09393   Mean   : 0.05863   Mean   :0.0492   Mean   : 0.2488  
##  3rd Qu.:0.00000   3rd Qu.: 0.00000   3rd Qu.:0.0000   3rd Qu.: 0.1000  
##  Max.   :5.55000   Max.   :10.00000   Max.   :4.4100   Max.   :20.0000  
```

```
##     business          email             you             credit
##  Min.   :0.0000   Min.   :0.0000   Min.   : 0.000   Min.   : 0.00000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.: 0.00000
##  Median :0.0000   Median :0.0000   Median : 1.310   Median : 0.00000
##  Mean   :0.1426   Mean   :0.1847   Mean   : 1.662   Mean   : 0.08558
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.: 2.640   3rd Qu.: 0.00000
##  Max.   :7.1400   Max.   :9.0900   Max.   :18.750   Max.   :18.18000
##       your              font             num000           money
##  Min.   : 0.0000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.00000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.00000
##  Median : 0.2200   Median : 0.0000   Median :0.0000   Median : 0.00000
##  Mean   : 0.8098   Mean   : 0.1212   Mean   :0.1016   Mean   : 0.09427
##  3rd Qu.: 1.2700   3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.: 0.00000
##  Max.   :11.1100   Max.   :17.1000   Max.   :5.4500   Max.   :12.50000
##        hp               hpl              george           num650
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   :0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median :0.0000
##  Mean   : 0.5495   Mean   : 0.2654   Mean   : 0.7673   Mean   :0.1248
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.:0.0000
##  Max.   :20.8300   Max.   :16.6600   Max.   :33.3300   Max.   :9.0900
##       lab               labs             telnet            num857
##  Min.   : 0.00000   Min.   :0.0000   Min.   : 0.00000   Min.   :0.00000
##  1st Qu.: 0.00000   1st Qu.:0.0000   1st Qu.: 0.00000   1st Qu.:0.00000
##  Median : 0.00000   Median :0.0000   Median : 0.00000   Median :0.00000
##  Mean   : 0.09892   Mean   :0.1029   Mean   : 0.06475   Mean   :0.04705
##  3rd Qu.: 0.00000   3rd Qu.:0.0000   3rd Qu.: 0.00000   3rd Qu.:0.00000
##  Max.   :14.28000   Max.   :5.8800   Max.   :12.50000   Max.   :4.76000
##       data              num415            num85          technology
##  Min.   : 0.00000   Min.   :0.00000   Min.   : 0.0000   Min.   :0.00000
##  1st Qu.: 0.00000   1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.:0.00000
##  Median : 0.00000   Median :0.00000   Median : 0.0000   Median :0.00000
##  Mean   : 0.09723   Mean   :0.04784   Mean   : 0.1054   Mean   :0.09748
##  3rd Qu.: 0.00000   3rd Qu.:0.00000   3rd Qu.: 0.0000   3rd Qu.:0.00000
##  Max.   :18.18000   Max.   :4.76000   Max.   :20.0000   Max.   :7.69000
##     num1999          parts              pm              direct
##  Min.   :0.000   Min.   :0.0000   Min.   : 0.00000   Min.   :0.00000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.: 0.00000   1st Qu.:0.00000
##  Median :0.000   Median :0.0000   Median : 0.00000   Median :0.00000
##  Mean   :0.137   Mean   :0.0132   Mean   : 0.07863   Mean   :0.06483
##  3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.: 0.00000   3rd Qu.:0.00000
##  Max.   :6.890   Max.   :8.3300   Max.   :11.11000   Max.   :4.76000
##        cs             meeting          original         project
##  Min.   :0.00000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.0000
##  1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Median :0.00000   Median : 0.0000   Median :0.0000   Median : 0.0000
##  Mean   :0.04367   Mean   : 0.1323   Mean   :0.0461   Mean   : 0.0792
##  3rd Qu.:0.00000   3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.: 0.0000
##  Max.   :7.14000   Max.   :14.2800   Max.   :3.5700   Max.   :20.0000
##        re               edu              table           conference
##  Min.   : 0.0000   Min.   : 0.0000   Min.   :0.000000   Min.   : 0.00000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:0.000000   1st Qu.: 0.00000
##  Median : 0.0000   Median : 0.0000   Median :0.000000   Median : 0.00000
##  Mean   : 0.3012   Mean   : 0.1798   Mean   :0.005444   Mean   : 0.03187
```

```
##  3rd Qu.: 0.1100   3rd Qu.: 0.0000   3rd Qu.:0.000000   3rd Qu.: 0.00000
##  Max.   :21.4200   Max.   :22.0500   Max.   :2.170000   Max.   :10.00000
##  charSemicolon     charRoundbracket charSquarebracket charExclamation
##  Min.   :0.00000   Min.   :0.000    Min.   :0.00000   Min.   : 0.0000
##  1st Qu.:0.00000   1st Qu.:0.000    1st Qu.:0.00000   1st Qu.: 0.0000
##  Median :0.00000   Median :0.065    Median :0.00000   Median : 0.0000
##  Mean   :0.03857   Mean   :0.139    Mean   :0.01698   Mean   : 0.2691
##  3rd Qu.:0.00000   3rd Qu.:0.188    3rd Qu.:0.00000   3rd Qu.: 0.3150
##  Max.   :4.38500   Max.   :9.752    Max.   :4.08100   Max.   :32.4780
##    charDollar         charHash          capitalAve        capitalLong
##  Min.   :0.00000   Min.   : 0.00000   Min.   :   1.000   Min.   :   1.00
##  1st Qu.:0.00000   1st Qu.: 0.00000   1st Qu.:   1.588   1st Qu.:   6.00
##  Median :0.00000   Median : 0.00000   Median :   2.276   Median :  15.00
##  Mean   :0.07581   Mean   : 0.04424   Mean   :   5.191   Mean   :  52.17
##  3rd Qu.:0.05200   3rd Qu.: 0.00000   3rd Qu.:   3.706   3rd Qu.:  43.00
##  Max.   :6.00300   Max.   :19.82900   Max.   :1102.500   Max.   :9989.00
##   capitalTotal          type
##  Min.   :    1.0   nonspam:2788
##  1st Qu.:   35.0   spam   :1813
##  Median :   95.0
##  Mean   :  283.3
##  3rd Qu.:  266.0
##  Max.   :15841.0
```

```
str(spam)
```

```
## 'data.frame':    4601 obs. of  58 variables:
##  $ make            : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ address         : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ all             : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ num3d           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our             : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
##  $ over            : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ remove          : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
##  $ internet        : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...
##  $ order           : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06 ...
##  $ mail            : num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.76 0 ...
##  $ receive         : num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...
##  $ will            : num  0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...
##  $ people          : num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.25 ...
##  $ report          : num  0 0.21 0 0 0 0 0 0 0 0 ...
##  $ addresses       : num  0 0.14 1.75 0 0 0 0 0 0 0.12 ...
##  $ free            : num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ business        : num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ email           : num  1.29 0.28 1.03 0 0 0 0.32 0 0.15 0.12 ...
##  $ you             : num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ credit          : num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ your            : num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ font            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num000          : num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ money           : num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ hp              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ hpl             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ george          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num650          : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ lab              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ labs             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ telnet           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num857           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ data             : num  0 0 0 0 0 0 0 0 0.15 0 ...
##  $ num415           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num85            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ technology       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num1999          : num  0 0.07 0 0 0 0 0 0 0 0 ...
##  $ parts            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pm               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ direct           : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ cs               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ meeting          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ original         : num  0 0 0.12 0 0 0 0 0 0.3 0 ...
##  $ project          : num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ re               : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ edu              : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ table            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ conference       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ charSemicolon    : num  0 0 0.01 0 0 0 0 0 0 0.04 ...
##  $ charRoundbracket : num  0 0.132 0.143 0.137 0.135 0.223 0.054 0.206 0.271 0.03 ...
##  $ charSquarebracket: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ charExclamation  : num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
##  $ charDollar       : num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
##  $ charHash         : num  0 0.048 0.01 0 0 0 0 0 0.022 0 ...
##  $ capitalAve       : num  3.76 5.11 9.82 3.54 3.54 ...
##  $ capitalLong      : num  61 101 485 40 40 15 4 11 445 43 ...
##  $ capitalTotal     : num  278 1028 2259 191 191 ...
##  $ type             : Factor w/ 2 levels "nonspam","spam": 2 2 2 2 2 2 2 2 2 2 ...
```

```
head(spam)
```

```
##   make address  all num3d  our over remove internet order mail receive
## 1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00  0.00 0.00    0.00
## 2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07  0.00 0.94    0.21
## 3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38
## 4 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31
## 5 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31
## 6 0.00    0.00 0.00     0 1.85 0.00   0.00     1.85  0.00 0.00    0.00
##   will people report addresses free business email  you credit your font
## 1 0.64   0.00   0.00      0.00 0.32     0.00  1.29 1.93   0.00 0.96    0
## 2 0.79   0.65   0.21      0.14 0.14     0.07  0.28 3.47   0.00 1.59    0
## 3 0.45   0.12   0.00      1.75 0.06     0.06  1.03 1.36   0.32 0.51    0
## 4 0.31   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0
## 5 0.31   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0
## 6 0.00   0.00   0.00      0.00 0.00     0.00  0.00 0.00   0.00 0.00    0
##   num000 money hp hpl george num650 lab labs telnet num857 data num415
## 1   0.00  0.00  0   0      0      0   0    0      0      0    0      0
## 2   0.43  0.43  0   0      0      0   0    0      0      0    0      0
## 3   1.16  0.06  0   0      0      0   0    0      0      0    0      0
## 4   0.00  0.00  0   0      0      0   0    0      0      0    0      0
## 5   0.00  0.00  0   0      0      0   0    0      0      0    0      0
## 6   0.00  0.00  0   0      0      0   0    0      0      0    0      0
##   num85 technology num1999 parts pm direct cs meeting original project
```

```
## 1     0          0    0.00     0  0   0.00  0       0     0.00       0
## 2     0          0    0.07     0  0   0.00  0       0     0.00       0
## 3     0          0    0.00     0  0   0.06  0       0     0.12       0
## 4     0          0    0.00     0  0   0.00  0       0     0.00       0
## 5     0          0    0.00     0  0   0.00  0       0     0.00       0
## 6     0          0    0.00     0  0   0.00  0       0     0.00       0
##     re  edu table conference charSemicolon charRoundbracket
## 1 0.00 0.00     0          0          0.00            0.000
## 2 0.00 0.00     0          0          0.00            0.132
## 3 0.06 0.06     0          0          0.01            0.143
## 4 0.00 0.00     0          0          0.00            0.137
## 5 0.00 0.00     0          0          0.00            0.135
## 6 0.00 0.00     0          0          0.00            0.223
##   charSquarebracket charExclamation charDollar charHash capitalAve
## 1                 0           0.778      0.000    0.000      3.756
## 2                 0           0.372      0.180    0.048      5.114
## 3                 0           0.276      0.184    0.010      9.821
## 4                 0           0.137      0.000    0.000      3.537
## 5                 0           0.135      0.000    0.000      3.537
## 6                 0           0.000      0.000    0.000      3.000
##   capitalLong capitalTotal type
## 1          61          278 spam
## 2         101         1028 spam
## 3         485         2259 spam
## 4          40          191 spam
## 5          40          191 spam
## 6          15           54 spam
```

```
table(is.na(spam))
```

```
##
##  FALSE
## 266858
```

```
table(spam$type)
```

```
##
## nonspam    spam
##    2788    1813
```

Tras la observación de los datos podemos concluir lo siguiente:

- La variable objetivo es *type* y consta de 2 clases (*nonspam* y *spam*).
- Todas las variables regresoras son numéricas.
- Todos los datos están completos, no hay valores perdidos.
- Los datos están ligeramente desbalanceados, porque existen más observaciones de la clase *nonspam*

1 División en conjunto entrenamiento y validación

A partir de la base de datos spam de la librería *kernlab*, construya una muestra de aprendizaje aleatoria formado por el 70% de las instancias, y una muestra de validación formada por el 30% restante.

```
n=nrow(spam)
train.index=sort(sample(1:n, ceiling(0.7*n)))
train=spam[train.index,]
```

```
test=spam[-train.index,]
```

1.1 Conjunto de entrenamiento

```
dim(train)
```

```
## [1] 3221   58
```

```
table(train$type)
```

```
##
## nonspam    spam
##    1971    1250
```

1.2 Conjunto de test

```
dim(test)
```

```
## [1] 1380   58
```

```
table(test$type)
```

```
##
## nonspam    spam
##     817     563
```

2 Construcción del modelo

Construya un modelo boosting a partir de la muestra de aprendizaje generada para pronosticar la variable *type* a partir de las restantes variables (utilice la librería *adabag*)

```
modelo = boosting(type~.,
                  data=train,
                  mfinal=200,
                  control = rpart.control(maxdepth=1))
```

3 Predicciones sobre conjunto test

Realice predicciones para la muestra de validación y obtenga la matriz de confusión y el porcentaje de observaciones mal clasificadas. Obtenga el margen de las observaciones de la muestra de validación y determine los índices correspondientes a las que han sido mal clasificadas

3.1 Predicciones en el conjunto test

```
pred = predict.boosting(modelo, newdata=test)
```

3.2 Matriz de confusión

```
pred$confusion
```

```
##                Observed Class
## Predicted Class nonspam spam
##         nonspam     783   54
##         spam         34  509
```

3.3 Porcentaje de clasificación incorrecta

```
model.misclass=round(100*pred$error, 3)
cat("Tasa de clasificación incorrecta:", model.misclass, "%")
```

```
## Tasa de clasificación incorrecta: 6.377 %
```

3.4 Margen de las observaciones test

```
margin.test=margins(pred, test)
```

3.5 Indices de observaciones mal clasificadas

Las observaciones mal clasificadas son aquellas que presentan un margen negativo.

```
(misclassified.index=which(margin.test$margins < 0))
```

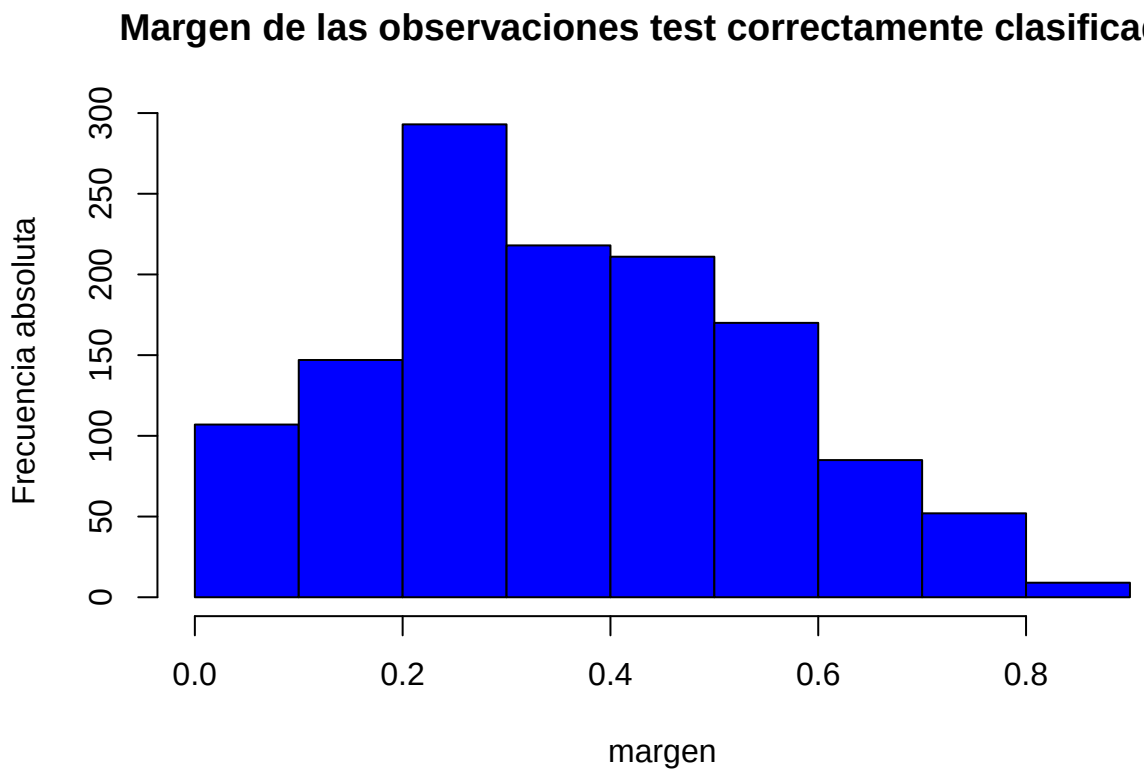
```
##  [1]    1    2   10   12   25   77  109  114  116  117  132  152  155  156
## [15]  177  178  191  217  237  253  301  303  323  342  349  356  371  376
## [29]  380  382  383  395  436  437  470  481  488  497  501  503  504  510
## [43]  516  517  519  522  527  529  532  534  549  551  552  556  623  658
## [57]  678  704  720  721  734  749  803  814  874  898  911  920 1007 1072
## [71] 1083 1085 1232 1245 1257 1268 1269 1271 1296 1301 1303 1305 1320 1322
## [85] 1330 1331 1332 1337
```

```
table(pred$class[misclassified.index])
```

```
##
## nonspam    spam
##      54      34
```

Las muestras correctamente clasificadas son las que presentan un margen positivo, considerándose mejor clasificadas aquellas cuyo margen está próximo a 1. Veamos a continuación la distribución de los valores de margen para las observaciones correctamente clasificadas:

```
hist(margin.test$margins[-misclassified.index],
     col="blue", freq=T,
     main="Margen de las observaciones test correctamente clasificadas",
     xlab="margen", ylab="Frecuencia absoluta")
```



El valor del margen para muestras del conjunto test correctamente clasificadas se encuentra en el intervalo:

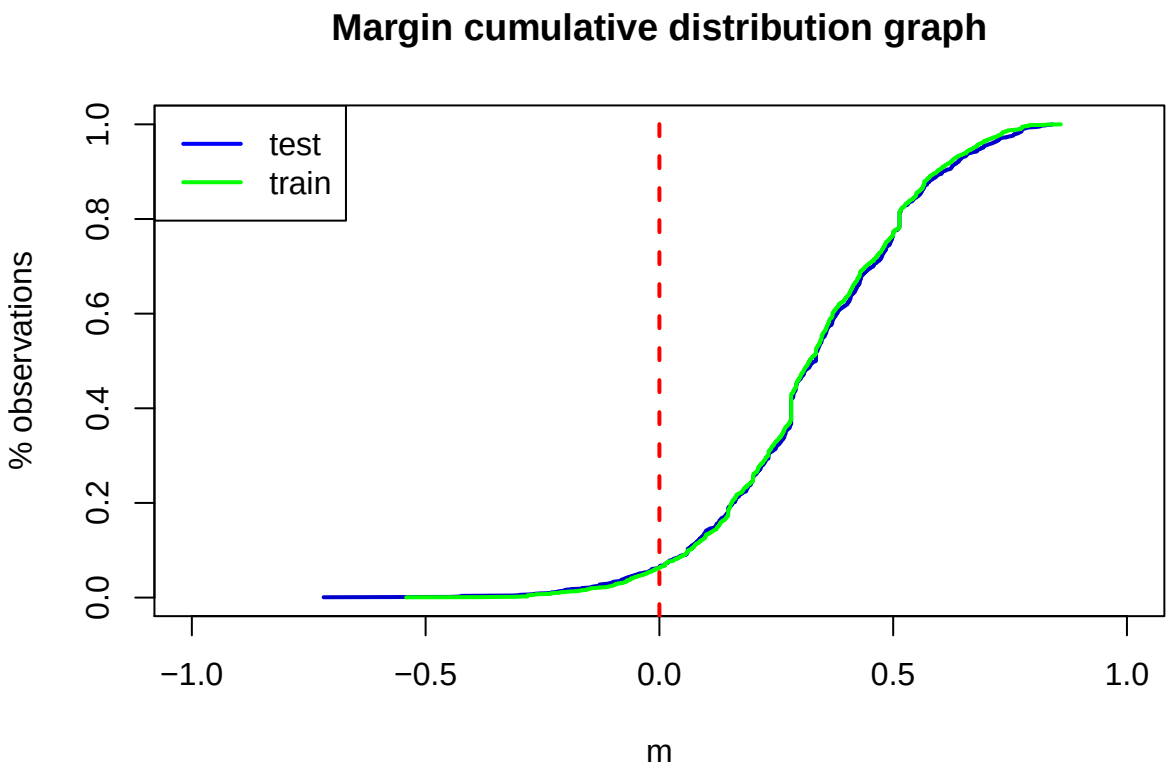
```
cat("[",  round(min(margin.test$margins[-misclassified.index]), 3), "-",
    round(max(margin.test$margins[-misclassified.index]), 3), "]" )
```

```
## [ 0 - 0.84 ]
```

Observamos que gran parte de los valores de márgenes para las observaciones correctamente clasificadas del conjunto test están bastante alejados de 1 .

3.6 Curva acumulativa del margen

```
margin.train = margins(modelo, train)
plot.margins(margin.test, margin.train, main="Curva acumulativa del margen")
```

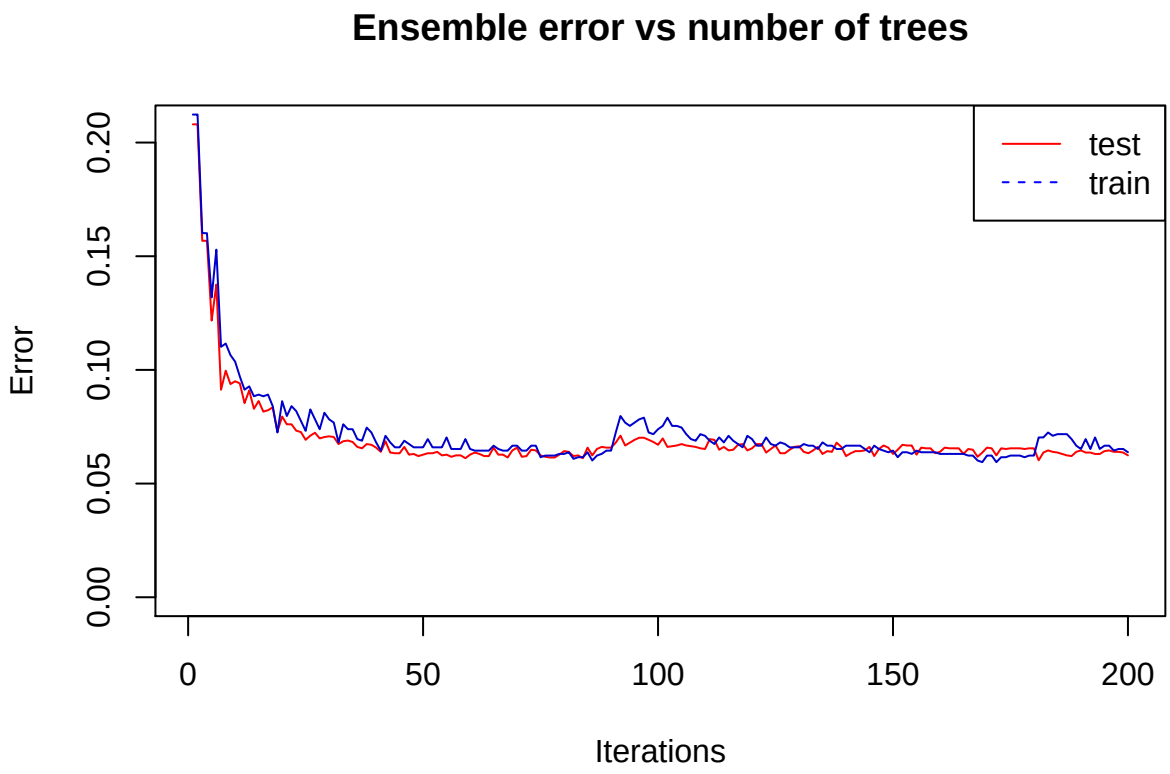


El clasificador necesita un porcentaje alto de observaciones para obtener un margen próximo a 1. Observamos que el comportamiento del conjunto test y entrenamiento son muy similares, ambas curvas están muy próximas, ligeramente por encima la del conjunto test como cabría esperar.

3.7 Representación gráfica de la evolución del error

Vamos a representar la evolución del error para comprobar si existe sobreajuste

```
error.train = errorevol(modelo,train)
error.test = errorevol(modelo,test)
plot.errorevol(error.train,error.test)
```



Observamos que en torno a 100 iteraciones el error fluctua y sube ligeramente para luego bajar. Habría que comparar si el modelo con 100 iteraciones presenta mejor o igual rendimiento en el conjunto test, porque a la vista de la gráfica se observa ligero sobreajuste al usar 200 iteraciones.

4 Validación cruzada

Utilizando validación cruzada con 10 pliegues, obtenga la matriz de confusión y el porcentaje de observaciones mal clasificadas.

4.1 Modelo entrenado con validación cruzada

```
modelo.cv = boosting.cv(type~.,
                        data=spam,    # uso todo el conjunto de datos
                        v=10,         # 10 pliegues
                        mfinal=20,    # 20 iteraciones
                        control = rpart.control(maxdepth=1))
```

```
## i:  1 Sat Jun 17 21:41:39 2017
## i:  2 Sat Jun 17 21:41:44 2017
## i:  3 Sat Jun 17 21:41:48 2017
## i:  4 Sat Jun 17 21:41:53 2017
## i:  5 Sat Jun 17 21:41:57 2017
## i:  6 Sat Jun 17 21:42:02 2017
```

```
## i:  7 Sat Jun 17 21:42:07 2017
## i:  8 Sat Jun 17 21:42:12 2017
## i:  9 Sat Jun 17 21:42:18 2017
## i:  10 Sat Jun 17 21:42:23 2017
```

4.2 Tabla de confusión

```
modelo.cv$confusion
```

```
##                 Observed Class
## Predicted Class nonspam spam
##         nonspam    2657  247
##         spam        131 1566
```

4.3 Porcentaje de observaciones mal clasificadas

```
modelo.cv.misclass= round(100*modelo.cv$error, 3)
cat("Tasa observaciones mal clasificadas:", modelo.cv.misclass, "%")
```

```
## Tasa observaciones mal clasificadas: 8.216 %
```

5 Cálculo de parámetros óptimos

Utilizando la función *train* de la librería *caret*, determine los parámetros óptimos dentro del siguiente conjunto:

- mfinal: {5, 6, 7, 8, 9, 10}
- maxdepth: {1, 2}
- coeflearn: {Breiman, Zhu}

Como técnica de validación, utilizar validación cruzada con 3 pliegues

5.1 Definición del método de validación

```
boost_valid = trainControl(method='cv',  # validación cruzada
                           number=3,     # número de pliegues
                           repeats=1)    # repeticiones del proceso validación
```

5.2 Rejilla para ajuste de parámetros

```
(boost_grid = expand.grid(mfinal=c(5, 6, 7, 8, 9, 10),
                          maxdepth=c(1, 24),
                          coeflearn=c("Breiman","Zhu")))
```

```
##    mfinal maxdepth coeflearn
## 1       5        1   Breiman
## 2       6        1   Breiman
## 3       7        1   Breiman
## 4       8        1   Breiman
## 5       9        1   Breiman
```

```
## 6      10        1   Breiman
## 7       5       24   Breiman
## 8       6       24   Breiman
## 9       7       24   Breiman
## 10      8       24   Breiman
## 11      9       24   Breiman
## 12     10       24   Breiman
## 13      5        1       Zhu
## 14      6        1       Zhu
## 15      7        1       Zhu
## 16      8        1       Zhu
## 17      9        1       Zhu
## 18     10        1       Zhu
## 19      5       24       Zhu
## 20      6       24       Zhu
## 21      7       24       Zhu
## 22      8       24       Zhu
## 23      9       24       Zhu
## 24     10       24       Zhu
```

5.3 Entrenamiento y validación del modelo

A partir de la muestra de entrenamiento obtengo los mejores párametros y el modelo construido con estos

```
modelo.params = train(type ~ .,
                      data=train,
                      method='AdaBoost.M1',
                      trControl=boost_valid,
                      tuneGrid=boost_grid)
```

5.4 Medidas de rendimiento para los distintos parámetros

```
modelo.params$results
```

```
##    coeflearn maxdepth mfinal  Accuracy     Kappa   AccuracySD     KappaSD
## 1    Breiman        1      5 0.8767473 0.7312882 0.0053436796 0.010022649
## 13       Zhu        1      5 0.8882360 0.7603719 0.0088367365 0.018937023
## 7    Breiman       24      5 0.9270408 0.8462815 0.0019662389 0.003620269
## 19       Zhu       24      5 0.9301462 0.8527225 0.0070254849 0.014910819
## 2    Breiman        1      6 0.8686746 0.7112285 0.0042541327 0.007585970
## 14       Zhu        1      6 0.8873073 0.7636486 0.0167151072 0.032781336
## 8    Breiman       24      6 0.9220744 0.8360003 0.0052841727 0.011139677
## 20       Zhu       24      6 0.9301459 0.8530843 0.0065177992 0.013511503
## 3    Breiman        1      7 0.8994156 0.7842846 0.0152832509 0.034376041
## 15       Zhu        1      7 0.8913397 0.7691777 0.0075499815 0.014550941
## 9    Breiman       24      7 0.9301465 0.8525936 0.0024396697 0.005434572
## 21       Zhu       24      7 0.9344916 0.8618909 0.0023768196 0.004986501
## 4    Breiman        1      8 0.8994101 0.7843380 0.0037247878 0.003923944
## 16       Zhu        1      8 0.9000323 0.7879913 0.0054228356 0.010560801
## 10   Breiman       24      8 0.9320098 0.8566874 0.0042407591 0.009247369
## 22       Zhu       24      8 0.9360446 0.8651834 0.0005556615 0.001244578
## 5    Breiman        1      9 0.8987891 0.7847828 0.0097818836 0.022355177
```

```
## 17       Zhu         1      9 0.8885380 0.7664532 0.0186914905 0.034292485
## 11   Breiman        24      9 0.9320104 0.8565090 0.0048939533 0.010812399
## 23       Zhu        24      9 0.9388370 0.8710101 0.0051640156 0.010957267
## 6    Breiman         1     10 0.9053102 0.7983236 0.0117744117 0.026261504
## 18       Zhu         1     10 0.9037570 0.7964036 0.0065976472 0.012471092
## 12   Breiman        24     10 0.9332522 0.8590572 0.0045601065 0.009920940
## 24       Zhu        24     10 0.9385278 0.8705307 0.0032425886 0.006880330
```

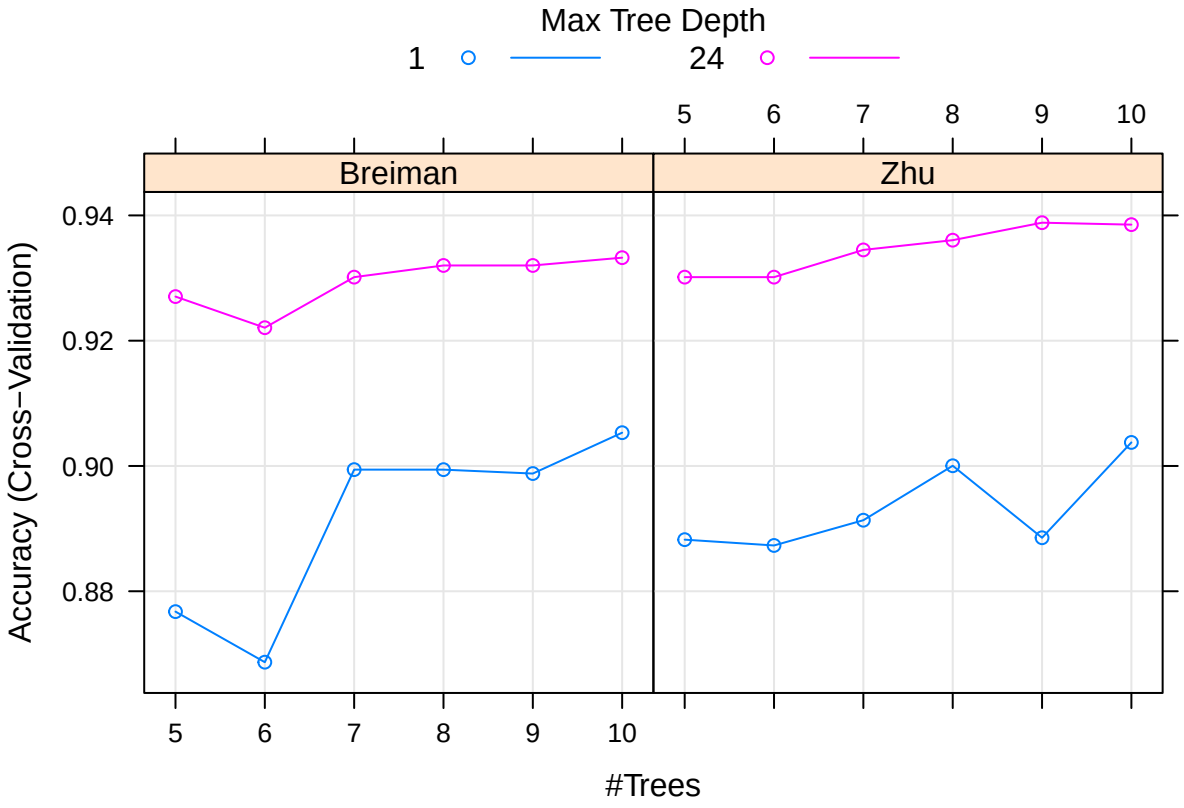
5.5 Parámetros seleccionados

```
(best.params=modelo.params$bestTune)
```

```
##    mfinal maxdepth coeflearn
## 23      9       24       Zhu
```

5.6 Representación gráfica del ajuste de parámetros

```
plot(modelo.params)
```



5.7 Obtención del mejor modelo (con los parámetros óptimos)

Obtengo el modelo con los parámetros óptimos usando la función *boosting* porque si uso *modelo.params$finalModel* la función *predict.boosting* no funciona correctamente, aunque el objeto sea de la clase

boosting

```
modelo.bestParams = boosting(type~.,
                             data=train,
                             mfinal=best.params$mfinal,
                             control=rpart.control(maxdepth=best.params$maxdepth),
                             coeflearn=best.params$coeflearn)
```

5.8 Cálculo de las predicciones sobre el conjunto test

```
pred.bestParams = predict.boosting(modelo.bestParams, newdata=test)
```

5.9 Matriz de confusión

```
pred.bestParams$confusion
```

```
##                Observed Class
## Predicted Class nonspam spam
##         nonspam     778   38
##         spam         39  525
```

5.10 Porcentaje de clasificación incorrecta

```
model.bestParams.misclass=round(100*pred.bestParams$error, 3)
cat("Tasa de clasificación incorrecta:", model.bestParams.misclass, "%")
```

```
## Tasa de clasificación incorrecta: 5.58 %
```

6 Comparativa

A continuación compararemos los resultados obtenidos con cada uno de los modelos construidos.

```
table_model1=c(model.misclass)
table_model2=c(modelo.cv.misclass)
table_model3=c(model.bestParams.misclass)


tabla_resumen = data.frame (rbind(table_model1, table_model2, table_model3),
                            row.names=c("modelo 1 (Breiman, mfinal=200)",
                                        "modelo validación cruzada (Breiman, mfinal=20, 10 pliegues)",
                                        "modelo parámetros óptimos"))

print(knitr::kable(round(tabla_resumen, 3), format = "pandoc",
                   col.names = c("Tasa clasificación incorrecta"),
                   align='c'))
```

	Tasa clasificación incorrecta
modelo 1 (Breiman, mfinal=200)	6.377
modelo validación cruzada (Breiman, mfinal=20, 10 pliegues)	8.216

	Tasa clasificación incorrecta
modelo parámetros óptimos	5.580

El mejor modelo es el modelo con parámetros ajustados, el construido en el apartado 5, ya que presenta la tasa de clasificación incorrecta más baja.