Practice on ROC and AUC

1. Usa el script {spam.R} para leer los datos de la SPAM e-mail database.

```
# SPAM E-mail Database
# downloaded from
# http://web.stanford.edu/~hastie/ElemStatLearn/datasets/spam.info.txt
# http://web.stanford.edu/~hastie/ElemStatLearn/datasets/spam.data
# http://web.stanford.edu/~hastie/ElemStatLearn/datasets/spam.traintest
# 03-05-2016
#
#
#

library(caret)
library(glmnet)
library(nnet)
library(class)
library(pROC)
library(ROC632)
#setwd("C:/Users/DanEscario/Desktop/MESIO/Statistical learning/Pr?ctiques/Corbes ROC/DadesSpam")
spam <- read.table("spambase.data.txt",sep=",")

spam.names <- c(read.table("spambase.names.txt",sep=":",skip=33,nrows=53,as.is=TRUE)[,1],
                "char_freq_#",
                read.table("spambase.names.txt",sep=":",skip=87,nrows=3,as.is=TRUE)[,1],
                "spam.01")

names(spam) <- spam.names

n<-dim(spam)[1]
p<-dim(spam)[2]-1

spam.01 <- spam[,p+1]
spam.vars <- as.matrix(spam[,1:p])

cat(paste("n = ",n,', p = ',p,sep=""))
```

```
## n = 4601, p = 57
```

```
cat(paste("Proportion of spam e-mails =",round(mean(spam.01),2),sep=""))
```

```
## Proportion of spam e-mails =0.39
```

```
glm.spam <- glm(spam.01 ~ spam.vars,family=binomial)
summary(glm.spam)
```

```
##
## Call:
## glm(formula = spam.01 ~ spam.vars, family = binomial)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -4.127  -0.203   0.000   0.114   5.364
##
```

```
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -1.569e+00  1.420e-01 -11.044  < 2e-16
## spam.varsword_freq_make        -3.895e-01  2.315e-01  -1.683 0.092388
## spam.varsword_freq_address     -1.458e-01  6.928e-02  -2.104 0.035362
## spam.varsword_freq_all          1.141e-01  1.103e-01   1.035 0.300759
## spam.varsword_freq_3d           2.252e+00  1.507e+00   1.494 0.135168
## spam.varsword_freq_our          5.624e-01  1.018e-01   5.524 3.31e-08
## spam.varsword_freq_over         8.830e-01  2.498e-01   3.534 0.000409
## spam.varsword_freq_remove       2.279e+00  3.328e-01   6.846 7.57e-12
## spam.varsword_freq_internet     5.696e-01  1.682e-01   3.387 0.000707
## spam.varsword_freq_order        7.343e-01  2.849e-01   2.577 0.009958
## spam.varsword_freq_mail         1.275e-01  7.262e-02   1.755 0.079230
## spam.varsword_freq_receive     -2.557e-01  2.979e-01  -0.858 0.390655
## spam.varsword_freq_will        -1.383e-01  7.405e-02  -1.868 0.061773
## spam.varsword_freq_people      -7.961e-02  2.303e-01  -0.346 0.729557
## spam.varsword_freq_report       1.447e-01  1.364e-01   1.061 0.288855
## spam.varsword_freq_addresses    1.236e+00  7.254e-01   1.704 0.088370
## spam.varsword_freq_free         1.039e+00  1.457e-01   7.128 1.01e-12
## spam.varsword_freq_business     9.599e-01  2.251e-01   4.264 2.01e-05
## spam.varsword_freq_email        1.203e-01  1.172e-01   1.027 0.304533
## spam.varsword_freq_you          8.131e-02  3.505e-02   2.320 0.020334
## spam.varsword_freq_credit       1.047e+00  5.383e-01   1.946 0.051675
## spam.varsword_freq_your         2.419e-01  5.243e-02   4.615 3.94e-06
## spam.varsword_freq_font         2.013e-01  1.627e-01   1.238 0.215838
## spam.varsword_freq_000          2.245e+00  4.714e-01   4.762 1.91e-06
## spam.varsword_freq_money        4.264e-01  1.621e-01   2.630 0.008535
## spam.varsword_freq_hp          -1.920e+00  3.128e-01  -6.139 8.31e-10
## spam.varsword_freq_hpl         -1.040e+00  4.396e-01  -2.366 0.017966
## spam.varsword_freq_george      -1.177e+01  2.113e+00  -5.569 2.57e-08
## spam.varsword_freq_650          4.454e-01  1.991e-01   2.237 0.025255
## spam.varsword_freq_lab         -2.486e+00  1.502e+00  -1.656 0.097744
## spam.varsword_freq_labs        -3.299e-01  3.137e-01  -1.052 0.292972
## spam.varsword_freq_telnet      -1.702e-01  4.815e-01  -0.353 0.723742
## spam.varsword_freq_857          2.549e+00  3.283e+00   0.776 0.437566
## spam.varsword_freq_data        -7.383e-01  3.117e-01  -2.369 0.017842
## spam.varsword_freq_415          6.679e-01  1.601e+00   0.417 0.676490
## spam.varsword_freq_85          -2.055e+00  7.883e-01  -2.607 0.009124
## spam.varsword_freq_technology   9.237e-01  3.091e-01   2.989 0.002803
## spam.varsword_freq_1999         4.651e-02  1.754e-01   0.265 0.790819
## spam.varsword_freq_parts       -5.968e-01  4.232e-01  -1.410 0.158473
## spam.varsword_freq_pm          -8.650e-01  3.828e-01  -2.260 0.023844
## spam.varsword_freq_direct      -3.046e-01  3.636e-01  -0.838 0.402215
## spam.varsword_freq_cs          -4.505e+01  2.660e+01  -1.694 0.090333
## spam.varsword_freq_meeting     -2.689e+00  8.384e-01  -3.207 0.001342
## spam.varsword_freq_original    -1.247e+00  8.064e-01  -1.547 0.121978
## spam.varsword_freq_project     -1.573e+00  5.292e-01  -2.973 0.002953
## spam.varsword_freq_re          -7.923e-01  1.556e-01  -5.091 3.56e-07
## spam.varsword_freq_edu         -1.459e+00  2.686e-01  -5.434 5.52e-08
## spam.varsword_freq_table       -2.326e+00  1.659e+00  -1.402 0.160958
## spam.varsword_freq_conference  -4.016e+00  1.611e+00  -2.493 0.012672
## spam.varschar_freq_;           -1.291e+00  4.422e-01  -2.920 0.003503
## spam.varschar_freq_(           -1.881e-01  2.494e-01  -0.754 0.450663
## spam.varschar_freq_[           -6.574e-01  8.383e-01  -0.784 0.432914
```

```
## spam.varschar_freq_!                 3.472e-01  8.926e-02   3.890 0.000100
## spam.varschar_freq_$                 5.336e+00  7.064e-01   7.553 4.24e-14
## spam.varschar_freq_#                 2.403e+00  1.113e+00   2.159 0.030883
## spam.varscapital_run_length_average  1.199e-02  1.884e-02   0.636 0.524509
## spam.varscapital_run_length_longest  9.118e-03  2.521e-03   3.618 0.000297
## spam.varscapital_run_length_total    8.437e-04  2.251e-04   3.747 0.000179
##
## (Intercept)                         ***
## spam.varsword_freq_make             .
## spam.varsword_freq_address          *
## spam.varsword_freq_all
## spam.varsword_freq_3d
## spam.varsword_freq_our              ***
## spam.varsword_freq_over             ***
## spam.varsword_freq_remove           ***
## spam.varsword_freq_internet         ***
## spam.varsword_freq_order            **
## spam.varsword_freq_mail             .
## spam.varsword_freq_receive
## spam.varsword_freq_will             .
## spam.varsword_freq_people
## spam.varsword_freq_report
## spam.varsword_freq_addresses        .
## spam.varsword_freq_free             ***
## spam.varsword_freq_business         ***
## spam.varsword_freq_email
## spam.varsword_freq_you              *
## spam.varsword_freq_credit           .
## spam.varsword_freq_your             ***
## spam.varsword_freq_font
## spam.varsword_freq_000              ***
## spam.varsword_freq_money            **
## spam.varsword_freq_hp               ***
## spam.varsword_freq_hpl              *
## spam.varsword_freq_george           ***
## spam.varsword_freq_650              *
## spam.varsword_freq_lab              .
## spam.varsword_freq_labs
## spam.varsword_freq_telnet
## spam.varsword_freq_857
## spam.varsword_freq_data             *
## spam.varsword_freq_415
## spam.varsword_freq_85               **
## spam.varsword_freq_technology       **
## spam.varsword_freq_1999
## spam.varsword_freq_parts
## spam.varsword_freq_pm               *
## spam.varsword_freq_direct
## spam.varsword_freq_cs               .
## spam.varsword_freq_meeting          **
## spam.varsword_freq_original
## spam.varsword_freq_project          **
## spam.varsword_freq_re               ***
## spam.varsword_freq_edu              ***
```

```
## spam.varsword_freq_table
## spam.varsword_freq_conference       *
## spam.varschar_freq_;                **
## spam.varschar_freq_(
## spam.varschar_freq_[
## spam.varschar_freq_!                ***
## spam.varschar_freq_$                ***
## spam.varschar_freq_#                *
## spam.varscapital_run_length_average
## spam.varscapital_run_length_longest ***
## spam.varscapital_run_length_total   ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6170.2  on 4600  degrees of freedom
## Residual deviance: 1815.8  on 4543  degrees of freedom
## AIC: 1931.8
##
## Number of Fisher Scoring iterations: 13
```

2. Separa un tercio de los datos para construir una muestra test. Hazlo de forma que la formen un tercio de los e-mails marcados como SPAM, y un tercio de los marcadados como NO SPAM. El resto de los datos formarán la muestra de entrenamiento.

3. Compararemos el comportamiento de 3 reglas discriminantes:

a. Regresión logística estimada por máxima verosimilitud (IRWLS, `{glm}`).

b. `Regresión logística estimada mediante Lasso ({glment}).`

c. `Red neuronal ({nnet})`

d. `k-nn ({knn} and {knn.cv} from package {}) Usa la muestra de entrenamiento para fijar los {`*`tunning parameters} y para estimar los parámetros de los diferentes métodos.`*

e. *`Regresión logística estimada por máxima verosimilitud (IRWLS, {`*`glm`*`}).`*

```
glm.spam.tr <- glm(spam.01 ~ . , data=spam, subset=spam.tr, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm.spam.tr)
```

```
##
## Call:
## glm(formula = spam.01 ~ ., family = binomial, data = spam, subset = spam.tr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.8474  -0.2303   0.0000   0.1145   4.8138
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -1.713e+00  1.748e-01  -9.800  < 2e-16 ***
## word_freq_make            -5.272e-01  3.145e-01  -1.676 0.093720 .
## word_freq_address         -1.370e-01  8.500e-02  -1.612 0.106892
## word_freq_all              9.214e-02  1.395e-01   0.661 0.508883
```

```
## word_freq_3d                3.378e+00  2.054e+00   1.645 0.100058
## word_freq_our               5.812e-01  1.176e-01   4.943 7.70e-07 ***
## word_freq_over              7.364e-01  2.772e-01   2.656 0.007904 **
## word_freq_remove            1.921e+00  3.601e-01   5.334 9.61e-08 ***
## word_freq_internet          3.855e-01  1.768e-01   2.180 0.029221 *
## word_freq_order             1.158e+00  3.816e-01   3.033 0.002418 **
## word_freq_mail              1.611e-01  9.107e-02   1.769 0.076844 .
## word_freq_receive          -4.341e-01  3.622e-01  -1.199 0.230627
## word_freq_will             -1.334e-01  9.238e-02  -1.444 0.148702
## word_freq_people           -9.193e-02  2.823e-01  -0.326 0.744657
## word_freq_report            1.531e-01  1.442e-01   1.062 0.288272
## word_freq_addresses         1.818e+00  1.162e+00   1.564 0.117784
## word_freq_free              9.314e-01  1.719e-01   5.420 5.98e-08 ***
## word_freq_business          1.136e+00  3.050e-01   3.725 0.000195 ***
## word_freq_email             2.144e-01  1.678e-01   1.277 0.201480
## word_freq_you               5.756e-02  4.233e-02   1.360 0.173915
## word_freq_credit            8.246e-01  5.413e-01   1.523 0.127658
## word_freq_your              3.381e-01  6.497e-02   5.204 1.95e-07 ***
## word_freq_font              1.505e-01  1.606e-01   0.937 0.348904
## word_freq_000               2.372e+00  6.032e-01   3.932 8.41e-05 ***
## word_freq_money             2.447e-01  1.374e-01   1.781 0.074969 .
## word_freq_hp               -1.653e+00  3.398e-01  -4.866 1.14e-06 ***
## word_freq_hpl              -1.171e+00  5.183e-01  -2.260 0.023848 *
## word_freq_george           -1.021e+01  2.873e+00  -3.552 0.000382 ***
## word_freq_650               6.209e-01  3.248e-01   1.912 0.055898 .
## word_freq_lab              -2.072e+00  1.633e+00  -1.269 0.204402
## word_freq_labs             -2.511e-01  4.548e-01  -0.552 0.580925
## word_freq_telnet           -4.119e+00  2.747e+00  -1.500 0.133688
## word_freq_857               1.295e+00  3.833e+00   0.338 0.735411
## word_freq_data             -5.409e-01  3.271e-01  -1.654 0.098212 .
## word_freq_415               3.307e-01  1.753e+00   0.189 0.850314
## word_freq_85               -2.440e+00  8.998e-01  -2.712 0.006695 **
## word_freq_technology        8.492e-01  3.582e-01   2.370 0.017765 *
## word_freq_1999             -7.133e-02  2.269e-01  -0.314 0.753204
## word_freq_parts            -5.902e-01  5.200e-01  -1.135 0.256367
## word_freq_pm               -8.993e-01  5.587e-01  -1.610 0.107460
## word_freq_direct           -4.316e-01  4.012e-01  -1.076 0.282082
## word_freq_cs               -4.924e+01  3.091e+01  -1.593 0.111178
## word_freq_meeting          -2.793e+00  1.123e+00  -2.488 0.012849 *
## word_freq_original         -7.142e-01  6.973e-01  -1.024 0.305762
## word_freq_project          -1.498e+00  5.804e-01  -2.581 0.009855 **
## word_freq_re               -6.159e-01  1.579e-01  -3.901 9.59e-05 ***
## word_freq_edu              -1.361e+00  3.019e-01  -4.508 6.54e-06 ***
## word_freq_table            -4.196e+00  2.875e+00  -1.459 0.144521
## word_freq_conference       -4.349e+00  2.070e+00  -2.101 0.035675 *
## `char_freq_;`              -1.141e+00  4.570e-01  -2.497 0.012512 *
## `char_freq_(`              -2.277e-01  2.944e-01  -0.774 0.439223
## `char_freq_[`               1.308e-01  1.333e+00   0.098 0.921840
## `char_freq_!`               7.086e-01  1.564e-01   4.532 5.86e-06 ***
## `char_freq_$`               5.165e+00  8.644e-01   5.975 2.30e-09 ***
## `char_freq_#`               2.603e+00  8.993e-01   2.895 0.003795 **
## capital_run_length_average  6.082e-03  2.411e-02   0.252 0.800822
## capital_run_length_longest  7.923e-03  3.327e-03   2.382 0.017239 *
## capital_run_length_total    1.150e-03  2.779e-04   4.137 3.51e-05 ***
```

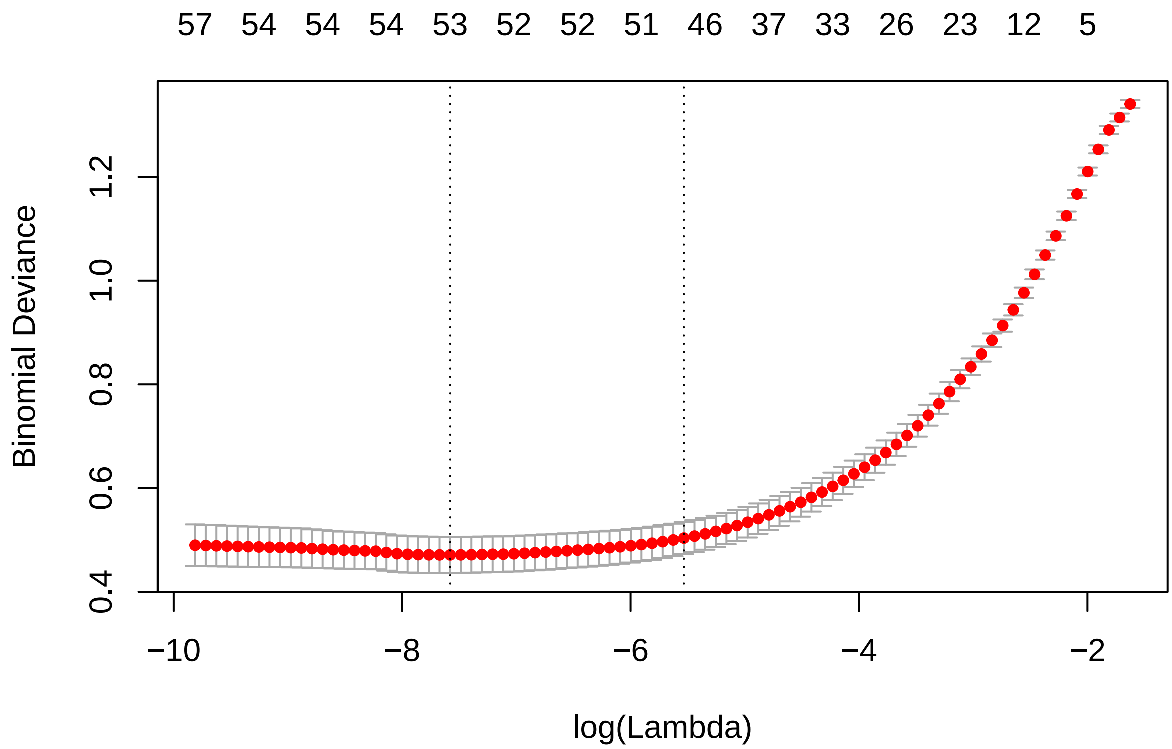
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4114.4  on 3067  degrees of freedom
## Residual deviance: 1220.0  on 3010  degrees of freedom
## AIC: 1336
##
## Number of Fisher Scoring iterations: 13
```

b. Regresión logística estimada mediante Lasso ({glmnet}).

```r
spam.tr.df<-spam[spam.tr,];
spam.tr.df.x<-spam.tr.df[,-58]
# glmnet.spam.tr<- glmnet(x=as.matrix(spam.tr.df.x) , y=as.matrix(spam.tr.df$spam.01), family='binomial
glmnet.spam.tr<-glmnet::cv.glmnet(x=as.matrix(spam.tr.df.x) , y=as.matrix(spam.tr.df$spam.01), family='
summary(glmnet.spam.tr)
```

```
##            Length Class  Mode
## lambda     89     -none- numeric
## cvm        89     -none- numeric
## cvsd       89     -none- numeric
## cvup       89     -none- numeric
## cvlo       89     -none- numeric
## nzero      89     -none- numeric
## name        1     -none- character
## glmnet.fit 13     lognet list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
```

```r
plot(glmnet.spam.tr)
```




```
#coef(glmnet.spam.tr, s = "lambda.min")
```

c. k-nn ({knn} and {knn.cv} from package {})

```
nnet.spam.tr<-nnet(x=as.matrix(spam.tr.df.x) , y=as.matrix(spam.tr.df$spam.01),size=7)
```

```
## # weights:  414
## initial  value 1197.911390
## iter  10 value 699.713043
## iter  20 value 566.195772
## iter  30 value 223.734399
## iter  40 value 143.142946
## iter  50 value 115.853176
## iter  60 value 99.945086
## iter  70 value 92.078316
## iter  80 value 86.062916
## iter  90 value 82.991931
## iter 100 value 80.764736
## final  value 80.764736
## stopped after 100 iterations
```

```
summary(nnet.spam.tr)
```

```
## a 57-7-1 network with 414 weights
## options were -
##   b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1
##   14.21   -2.10    0.98   -7.53   -0.22   -8.08   -2.60   -7.93   -6.05
##  i9->h1 i10->h1 i11->h1 i12->h1 i13->h1 i14->h1 i15->h1 i16->h1 i17->h1
```

```
##   -1.76   -5.16   -1.46  -11.97   -0.28    1.22   -0.87   -5.51   -3.47
## i18->h1 i19->h1 i20->h1 i21->h1 i22->h1 i23->h1 i24->h1 i25->h1 i26->h1
##  -14.80    3.81   -1.00   -3.13   -2.06   -1.10    0.49    0.11    0.68
## i27->h1 i28->h1 i29->h1 i30->h1 i31->h1 i32->h1 i33->h1 i34->h1 i35->h1
##    0.16    0.91   -0.35    0.45    0.52    0.36   -1.86   -0.40    0.20
## i36->h1 i37->h1 i38->h1 i39->h1 i40->h1 i41->h1 i42->h1 i43->h1 i44->h1
##   -6.09   -2.12    4.63    0.60    0.52   -0.36    0.43    0.03   -0.14
## i45->h1 i46->h1 i47->h1 i48->h1 i49->h1 i50->h1 i51->h1 i52->h1 i53->h1
##  -11.37    1.87    0.36   -0.23   -1.36    3.79    0.19  -14.18   -0.36
## i54->h1 i55->h1 i56->h1 i57->h1
##    1.92    4.20   -2.34    0.00
##   b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2
##   -0.31    0.57   -0.31    0.42   -1.50   -1.08   -1.72   -7.98   -2.71
##  i9->h2 i10->h2 i11->h2 i12->h2 i13->h2 i14->h2 i15->h2 i16->h2 i17->h2
##   -0.86   -0.02    3.30    0.10    2.09   -0.26   -1.18   -0.42   -4.08
## i18->h2 i19->h2 i20->h2 i21->h2 i22->h2 i23->h2 i24->h2 i25->h2 i26->h2
##    0.46   -0.05   -6.93    0.04    0.20   -3.18   -4.37   11.20    3.95
## i27->h2 i28->h2 i29->h2 i30->h2 i31->h2 i32->h2 i33->h2 i34->h2 i35->h2
##    8.28   -5.01    5.45   -2.32   11.42   13.14    0.25   -1.57    5.91
## i36->h2 i37->h2 i38->h2 i39->h2 i40->h2 i41->h2 i42->h2 i43->h2 i44->h2
##   -4.48    1.77   -4.90    2.76    0.28   13.79    1.57   -0.04    3.49
## i45->h2 i46->h2 i47->h2 i48->h2 i49->h2 i50->h2 i51->h2 i52->h2 i53->h2
##    2.31    6.30    6.82    5.93    1.88    0.63    2.91   -3.58   -8.14
## i54->h2 i55->h2 i56->h2 i57->h2
##    0.84   -0.46    0.01    0.00
##   b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3
##  -27.18    0.28   -2.85   -2.41    0.44   -5.41   -2.25   13.28    1.47
##  i9->h3 i10->h3 i11->h3 i12->h3 i13->h3 i14->h3 i15->h3 i16->h3 i17->h3
##   -4.36   -1.47    3.38   -7.01    0.99   -7.73   -8.89    6.19   -2.37
## i18->h3 i19->h3 i20->h3 i21->h3 i22->h3 i23->h3 i24->h3 i25->h3 i26->h3
##    8.61    7.26    2.46    3.86   -0.63    6.62    3.89  -41.88  -17.19
## i27->h3 i28->h3 i29->h3 i30->h3 i31->h3 i32->h3 i33->h3 i34->h3 i35->h3
##  -11.39   -0.49   -3.59   -4.16   -4.04   -3.08   -2.71   -3.75  -10.08
## i36->h3 i37->h3 i38->h3 i39->h3 i40->h3 i41->h3 i42->h3 i43->h3 i44->h3
##   -5.83    5.08   -9.11   -4.58   -4.71   -0.57   -1.34   -1.95   -5.87
## i45->h3 i46->h3 i47->h3 i48->h3 i49->h3 i50->h3 i51->h3 i52->h3 i53->h3
##    2.87   -5.36   -0.33   -1.93   16.63   -3.16    1.56    3.15    1.03
## i54->h3 i55->h3 i56->h3 i57->h3
##    7.06   -3.64    3.03    0.07
##   b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4  i7->h4  i8->h4
##   -0.07    0.53    0.61    0.62    0.31   -0.72   -0.58   -0.18   -0.54
##  i9->h4 i10->h4 i11->h4 i12->h4 i13->h4 i14->h4 i15->h4 i16->h4 i17->h4
##   -0.48   -0.62    0.08    0.67   -0.67   -0.59   -0.41   -0.29   -0.14
## i18->h4 i19->h4 i20->h4 i21->h4 i22->h4 i23->h4 i24->h4 i25->h4 i26->h4
##   -0.54    0.40    0.54   -0.41    0.61   -0.42    1.00   -0.51   -0.47
## i27->h4 i28->h4 i29->h4 i30->h4 i31->h4 i32->h4 i33->h4 i34->h4 i35->h4
##   -0.32   -0.17    0.46    0.13   -0.26    0.11   -0.42   -0.53   -0.20
## i36->h4 i37->h4 i38->h4 i39->h4 i40->h4 i41->h4 i42->h4 i43->h4 i44->h4
##   -0.13    0.11    0.18   -0.10   -0.50   -0.06    0.00    0.41   -0.21
## i45->h4 i46->h4 i47->h4 i48->h4 i49->h4 i50->h4 i51->h4 i52->h4 i53->h4
##   -0.35    0.10    0.28    0.31    0.39   -0.12    0.59    0.42    0.58
## i54->h4 i55->h4 i56->h4 i57->h4
##   -0.60   -0.76   -0.44   -2.25
##   b->h5  i1->h5  i2->h5  i3->h5  i4->h5  i5->h5  i6->h5  i7->h5  i8->h5
```

```
##    0.55    0.63    0.02    0.65   -0.35    0.81    0.66   -0.38    0.17
##  i9->h5 i10->h5 i11->h5 i12->h5 i13->h5 i14->h5 i15->h5 i16->h5 i17->h5
##    0.07    0.50   -0.65   -1.09   -0.26   -0.25    0.38   -0.40    0.21
## i18->h5 i19->h5 i20->h5 i21->h5 i22->h5 i23->h5 i24->h5 i25->h5 i26->h5
##    0.19   -0.06   -0.64   -1.33   -0.66    0.41    0.28   -0.32   -0.55
## i27->h5 i28->h5 i29->h5 i30->h5 i31->h5 i32->h5 i33->h5 i34->h5 i35->h5
##    0.49   -0.19   -0.45    0.25   -0.52   -0.01   -0.60    0.42    0.47
## i36->h5 i37->h5 i38->h5 i39->h5 i40->h5 i41->h5 i42->h5 i43->h5 i44->h5
##   -0.37    0.16   -0.05    0.18   -0.52    0.68   -0.40   -0.25    0.02
## i45->h5 i46->h5 i47->h5 i48->h5 i49->h5 i50->h5 i51->h5 i52->h5 i53->h5
##    0.33   -0.17   -0.50    0.32    0.13   -0.45   -0.66   -0.26   -0.46
## i54->h5 i55->h5 i56->h5 i57->h5
##    0.68   -0.12    0.63    1.54
##   b->h6  i1->h6  i2->h6  i3->h6  i4->h6  i5->h6  i6->h6  i7->h6  i8->h6
##    0.64   -0.31   -0.48   -0.06    0.67    0.99    0.65    0.20   -0.07
##  i9->h6 i10->h6 i11->h6 i12->h6 i13->h6 i14->h6 i15->h6 i16->h6 i17->h6
##   -0.22    0.42   -0.55    0.02   -0.16   -0.31    0.66   -0.53    0.65
## i18->h6 i19->h6 i20->h6 i21->h6 i22->h6 i23->h6 i24->h6 i25->h6 i26->h6
##    0.21   -0.27   -0.33   -0.69    0.11    0.12   -0.94   -0.27    0.20
## i27->h6 i28->h6 i29->h6 i30->h6 i31->h6 i32->h6 i33->h6 i34->h6 i35->h6
##    1.94   -0.13    0.04    0.11   -0.39   -0.41   -0.42   -0.45   -0.48
## i36->h6 i37->h6 i38->h6 i39->h6 i40->h6 i41->h6 i42->h6 i43->h6 i44->h6
##   -0.18    0.33   -0.23    0.05    0.12   -0.17    0.75    0.53   -0.13
## i45->h6 i46->h6 i47->h6 i48->h6 i49->h6 i50->h6 i51->h6 i52->h6 i53->h6
##   -0.41    0.13    0.43   -0.41   -0.61   -0.09    0.23    0.15   -0.15
## i54->h6 i55->h6 i56->h6 i57->h6
##   -0.69    0.96    1.73   -3.65
##   b->h7  i1->h7  i2->h7  i3->h7  i4->h7  i5->h7  i6->h7  i7->h7  i8->h7
##    0.51    0.13    0.27   -0.16   -0.01    0.40    0.30    0.04   -0.05
##  i9->h7 i10->h7 i11->h7 i12->h7 i13->h7 i14->h7 i15->h7 i16->h7 i17->h7
##    0.50    0.47   -0.70   -0.50   -0.26    0.38   -0.52    0.25   -0.51
## i18->h7 i19->h7 i20->h7 i21->h7 i22->h7 i23->h7 i24->h7 i25->h7 i26->h7
##    0.37    0.61   -0.13    0.36    0.19    0.38    0.31   -0.13    0.07
## i27->h7 i28->h7 i29->h7 i30->h7 i31->h7 i32->h7 i33->h7 i34->h7 i35->h7
##    0.35   -0.57   -0.11    0.28   -0.40   -0.30   -0.38   -0.56   -0.03
## i36->h7 i37->h7 i38->h7 i39->h7 i40->h7 i41->h7 i42->h7 i43->h7 i44->h7
##    0.35    0.19    0.60   -0.47    0.06   -0.45    0.40   -0.15   -0.18
## i45->h7 i46->h7 i47->h7 i48->h7 i49->h7 i50->h7 i51->h7 i52->h7 i53->h7
##    0.48   -0.06    0.63    0.52    0.14   -0.16    0.67    0.37   -0.25
## i54->h7 i55->h7 i56->h7 i57->h7
##   -0.31    0.69    0.39    0.85
##   b->o  h1->o  h2->o  h3->o  h4->o  h5->o  h6->o  h7->o
##   2.38 -16.08 -26.10   8.98  13.57  -6.45  -4.13   2.13
```

d. k-nn ({knn} and {knn.cv} from package {class})

```r
#we use caret package to estimate the optimal K for the knn;
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3333)
knn_fit <- train(as.factor(spam.01) ~., data = spam.tr.df, method = "knn",
 trControl=trctrl,
 #preProcess = c("center", "scale"),
 tuneLength = 10)

#selecting k=5
```

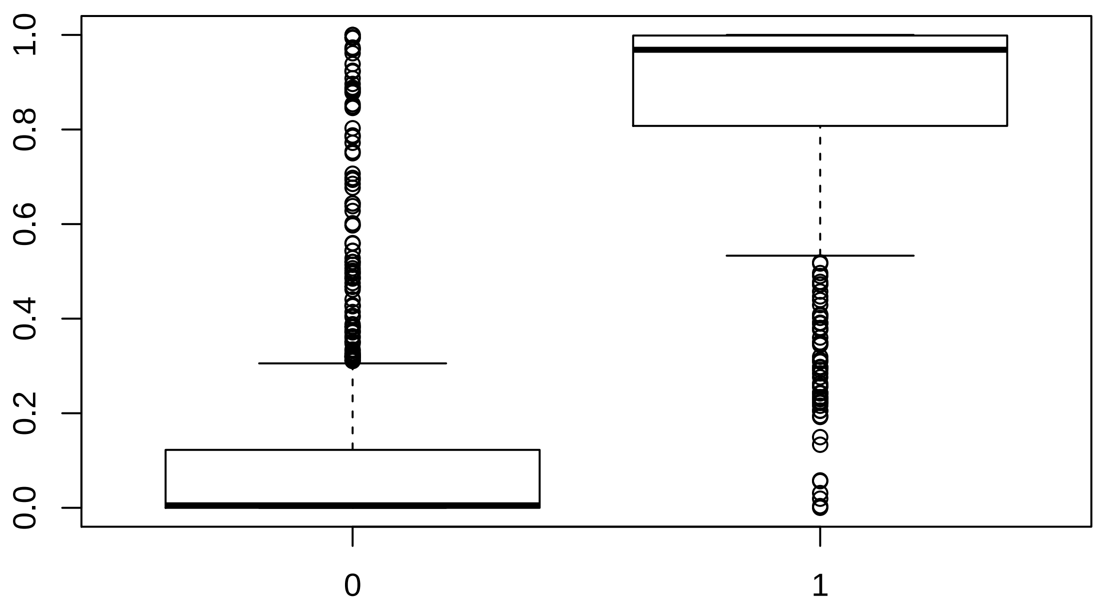
```r
spam.test.df<-spam[spam.test,]
spam.test.df.x<-spam.test.df[,-58]
cl=spam.tr.df[,58]
knn.spam.tr<-class::knn(train=as.data.frame(spam.tr.df),
                        test=as.data.frame(spam.test.df),
                        cl,
                        k=5,
                        prob=TRUE)
```

4. Usa la muestra test para construir (y dibujar) la curva ROC y calcular la AUC para cada una de estas reglas.

a. Regresión logística estimada por máxima verosimilitud (IRWLS, {glm}).

```r
names(spam.test.df) <- spam.names
pred.glm.spam.test <- predict(glm.spam.tr, newdata = spam.test.df, type="response")

boxplot(pred.glm.spam.test ~ spam.test.df$spam.01)
```




```r
table1<-table( spam.test.df$spam.01, pred.glm.spam.test>.5 )
table1p<-prop.table(table1)
table1t<-table1p[1,1]+table1p[2,2];table1t
```

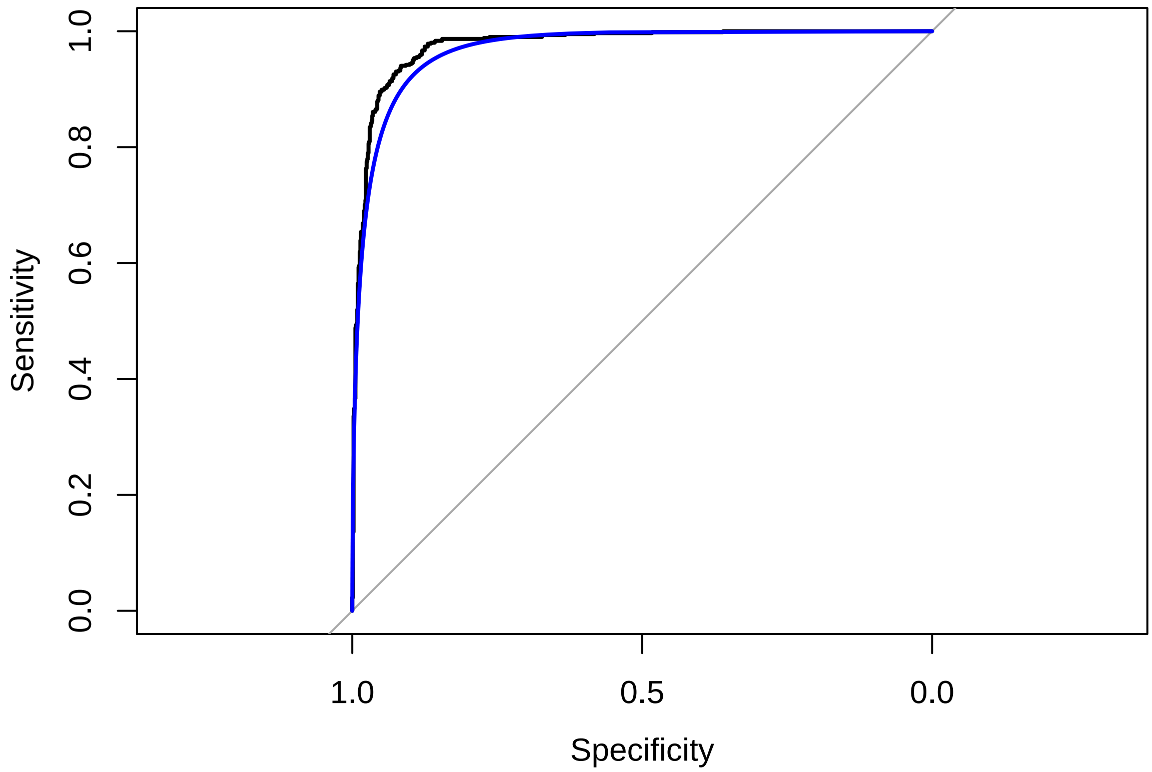
```
## [1] 0.927593
```

```r
roc(spam.test.df$spam.01 ~ pred.glm.spam.test, plot=TRUE)
```

```
##
```

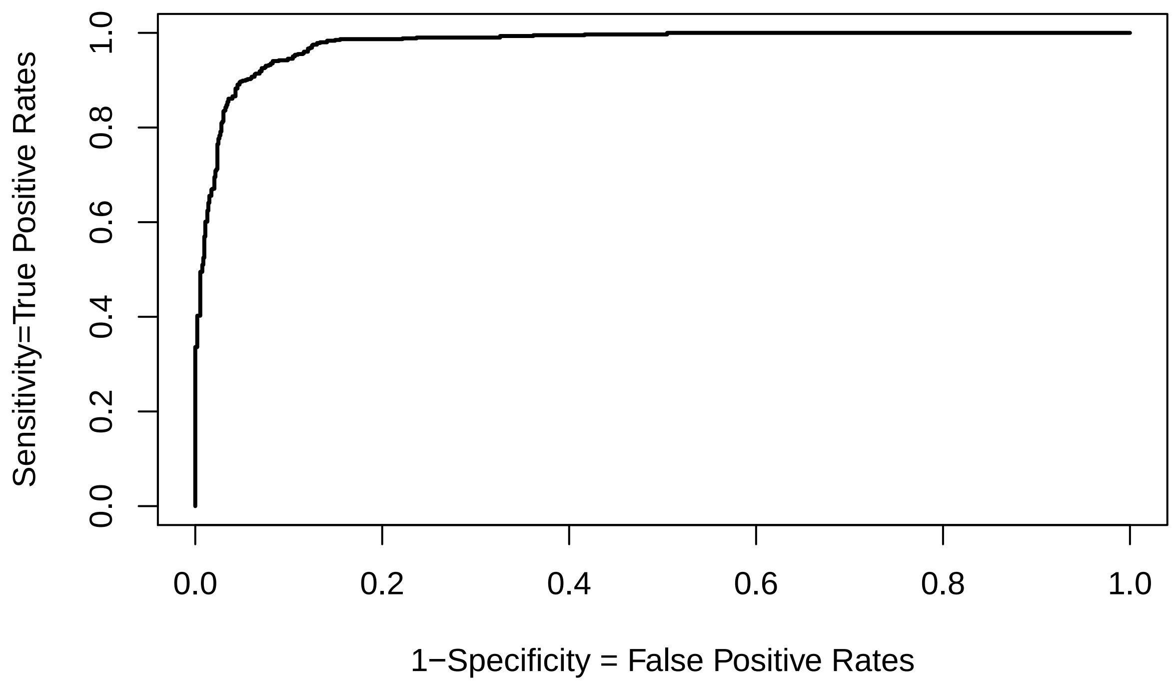
```
## Call:
## roc.formula(formula = spam.test.df$spam.01 ~ pred.glm.spam.test,     plot = TRUE)
##
## Data: pred.glm.spam.test in 929 controls (spam.test.df$spam.01 0) < 604 cases (spam.test.df$spam.01 
## Area under the curve: 0.9763
```

```
roc(spam.test.df$spam.01 ~ pred.glm.spam.test, smooth=TRUE, plot=TRUE, add=TRUE, col=4)
```




```
##
## Call:
## roc.formula(formula = spam.test.df$spam.01 ~ pred.glm.spam.test,     smooth = TRUE, plot = TRUE, add
##
## Data: pred.glm.spam.test in 929 controls (spam.test.df$spam.01 0) < 604 cases (spam.test.df$spam.01 
## Smoothing: binormal
## Area under the curve: 0.9691
```

```
J <- 201
cut.points <- (0:J)/J
ROC.obj <- ROC(status=spam.test.df$spam.01, marker=pred.glm.spam.test, cut.values=cut.points)
plot(ROC.obj$FP, ROC.obj$TP, ylab="Sensitivity=True Positive Rates",
     xlab="1-Specificity = False Positive Rates", type="s", lwd=2)
```




```
ROC.obj$AUC
```

```
## [1] 0.9759132
```

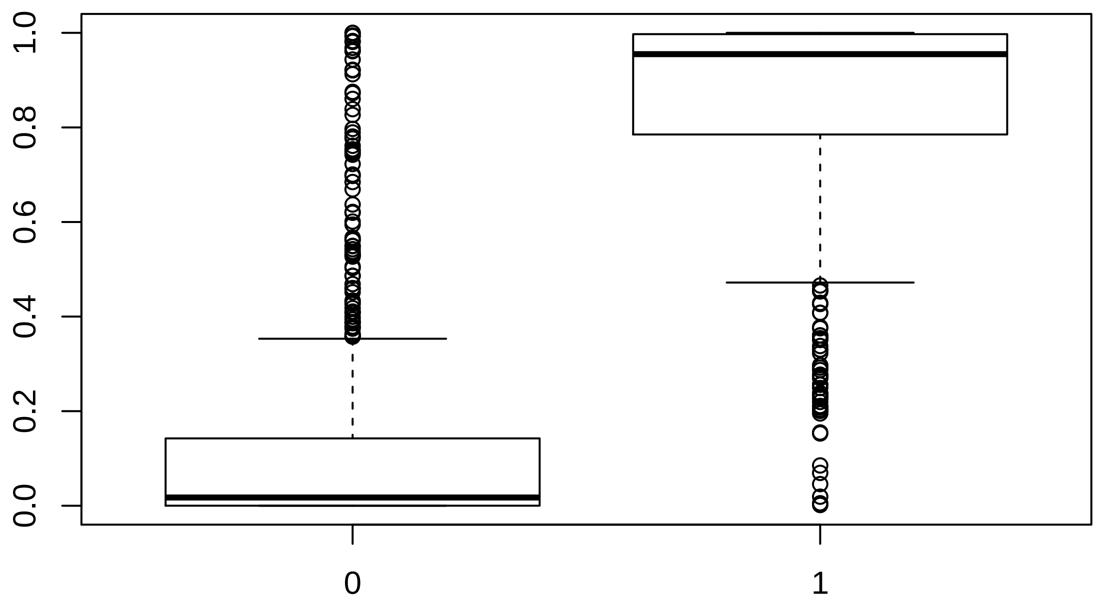
```
AUC(sens=ROC.obj$TP, spec=1-ROC.obj$FP)
```

```
## [1] 0.9774414
```

b. Regresión logística estimada mediante Lasso (`{glment}`).

```
pred.glmnet.spam.test <- predict(glmnet.spam.tr,
                                 newx= as.matrix(spam.test.df.x),
                                 type="response",
                                 s = "lambda.min")

boxplot(pred.glmnet.spam.test ~ spam.test.df$spam.01)
```




```
table2<-table( spam.test.df$spam.01, pred.glmnet.spam.test>.5 )
table2p<-prop.table(table2)
table2t<-table2p[1,1]+table2p[2,2];table2t
```

```
## [1] 0.927593
```

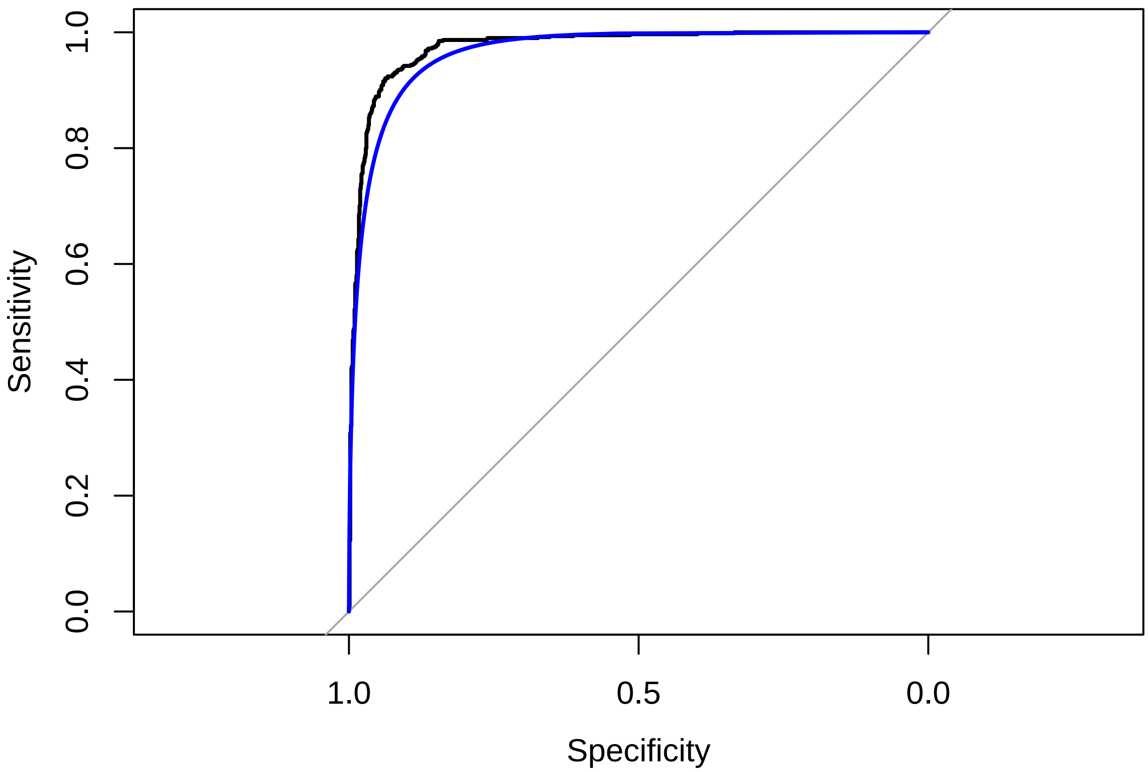
```
roc(spam.test.df$spam.01 ~ pred.glmnet.spam.test, plot=TRUE)
```

```
## Warning in roc.default(response, m[[predictors]], ...): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```

```
##
## Call:
## roc.formula(formula = spam.test.df$spam.01 ~ pred.glmnet.spam.test,     plot = TRUE)
##
## Data: pred.glmnet.spam.test in 929 controls (spam.test.df$spam.01 0) < 604 cases (spam.test.df$spam.0
## Area under the curve: 0.9752
```

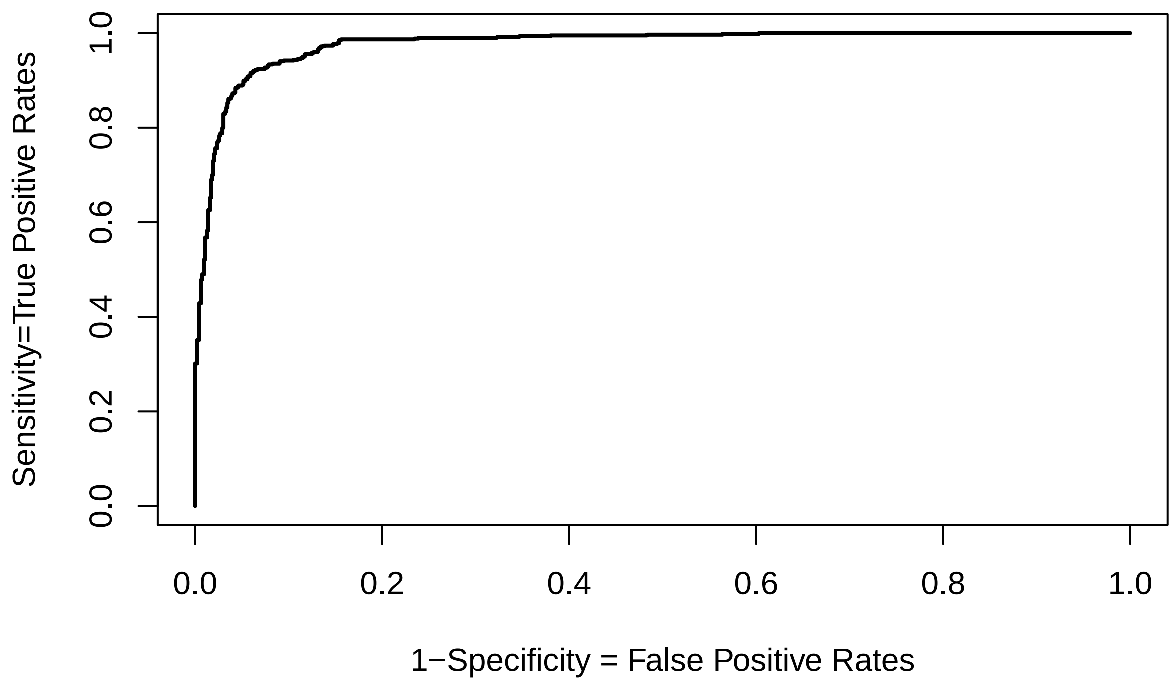
```
roc(spam.test.df$spam.01 ~ pred.glmnet.spam.test, smooth=TRUE, plot=TRUE, add=TRUE, col=4)
```

```
## Warning in roc.default(response, m[[predictors]], ...): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```




```
##
## Call:
## roc.formula(formula = spam.test.df$spam.01 ~ pred.glmnet.spam.test,     smooth = TRUE, plot = TRUE, a
##
## Data: pred.glmnet.spam.test in 929 controls (spam.test.df$spam.01 0) < 604 cases (spam.test.df$spam.0
## Smoothing: binormal
## Area under the curve: 0.9662
```

```
J <- 201
cut.points <- (0:J)/J
ROC.obj <- ROC(status=spam.test.df$spam.01, marker=pred.glmnet.spam.test, cut.values=cut.points)
plot(ROC.obj$FP, ROC.obj$TP, ylab="Sensitivity=True Positive Rates",
     xlab="1-Specificity = False Positive Rates", type="s", lwd=2)
```




```
ROC.obj$AUC
```

```
## [1] 0.9752101
```

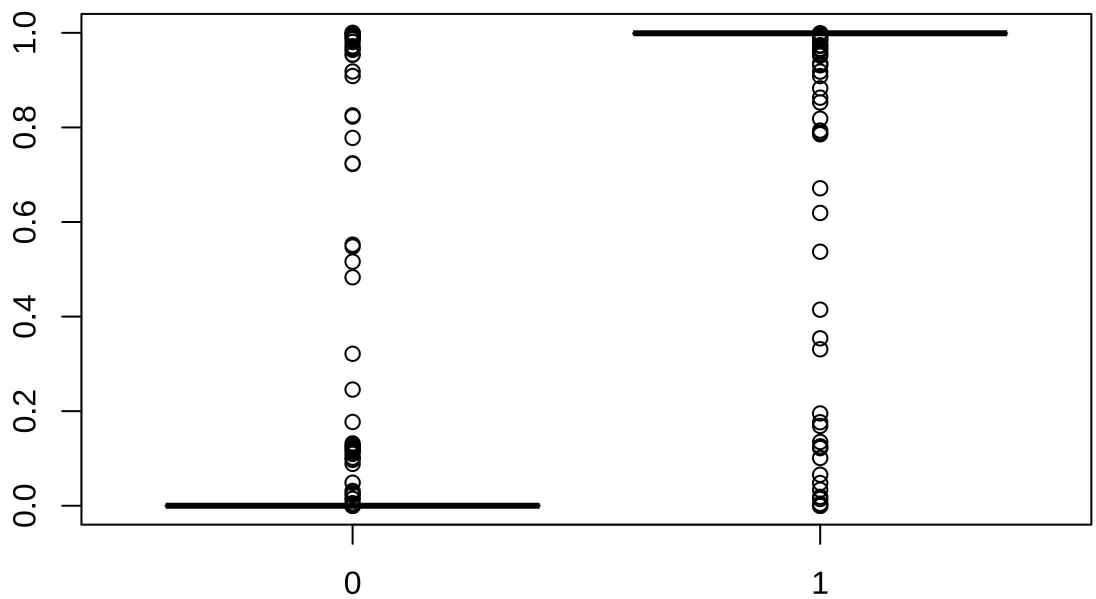
```
AUC(sens=ROC.obj$TP, spec=1-ROC.obj$FP)
```

```
## [1] 0.9762295
```

c. Red neuronal ({nnet})

```
pred.nnet.spam.test <- predict(object=nnet.spam.tr, newdata=spam.test.df.x, type=c('raw','class') )

boxplot(pred.nnet.spam.test ~ spam.test.df$spam.01)
```




```
table2<-table( spam.test.df$spam.01, pred.nnet.spam.test>.5 )
table2p<-prop.table(table2)
table2t<-table2p[1,1]+table2p[2,2];table2t
```

```
## [1] 0.9347684
```

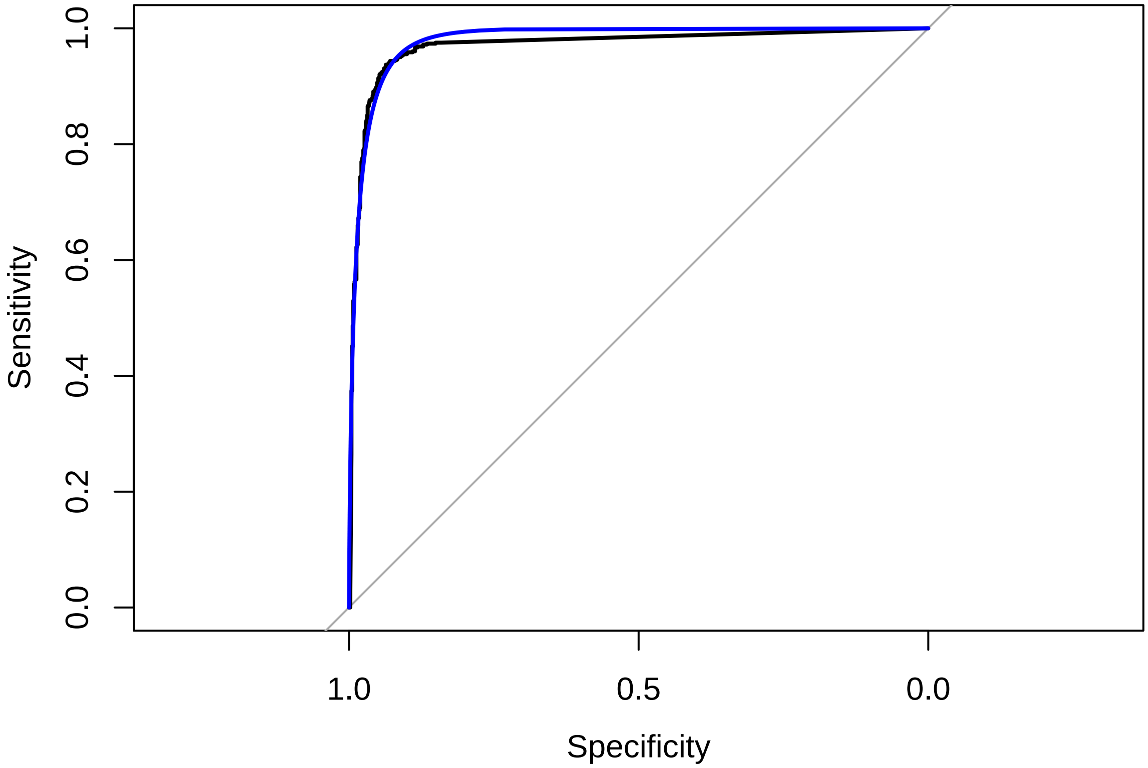
```
roc(spam.test.df$spam.01 ~ pred.nnet.spam.test, plot=TRUE)
```

```
## Warning in roc.default(response, m[[predictors]], ...): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```

```
##
## Call:
## roc.formula(formula = spam.test.df$spam.01 ~ pred.nnet.spam.test,     plot = TRUE)
##
## Data: pred.nnet.spam.test in 929 controls (spam.test.df$spam.01 0) < 604 cases (spam.test.df$spam.01
## Area under the curve: 0.9696
```

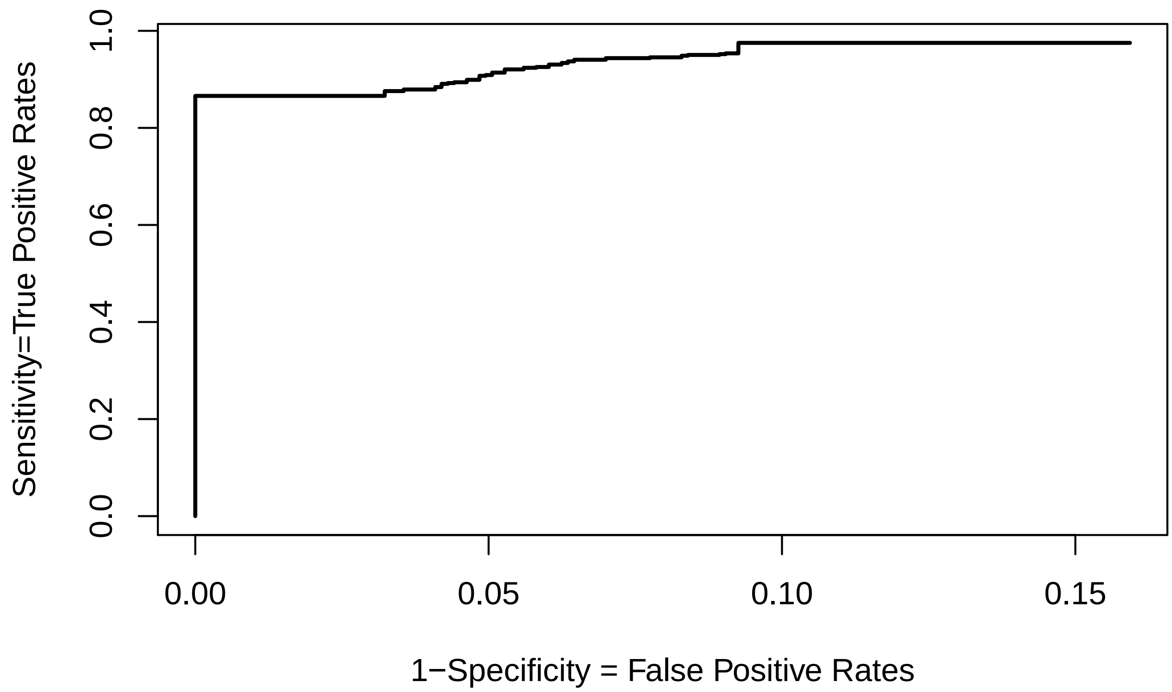
```
roc(spam.test.df$spam.01 ~ pred.nnet.spam.test, smooth=TRUE, plot=TRUE, add=TRUE, col=4)
```

```
## Warning in roc.default(response, m[[predictors]], ...): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```




```
##
## Call:
## roc.formula(formula = spam.test.df$spam.01 ~ pred.nnet.spam.test,     smooth = TRUE, plot = TRUE, ad
##
## Data: pred.nnet.spam.test in 929 controls (spam.test.df$spam.01 0) < 604 cases (spam.test.df$spam.01
## Smoothing: binormal
## Area under the curve: 0.9796
```

```
J <- 201
cut.points <- (0:J)/J
ROC.obj <- ROC(status=spam.test.df$spam.01, marker=pred.nnet.spam.test, cut.values=cut.points)
plot(ROC.obj$FP, ROC.obj$TP, ylab="Sensitivity=True Positive Rates",
     xlab="1-Specificity = False Positive Rates", type="s", lwd=2)
```




```
ROC.obj$AUC
```

```
## [1] 0.9641928
```

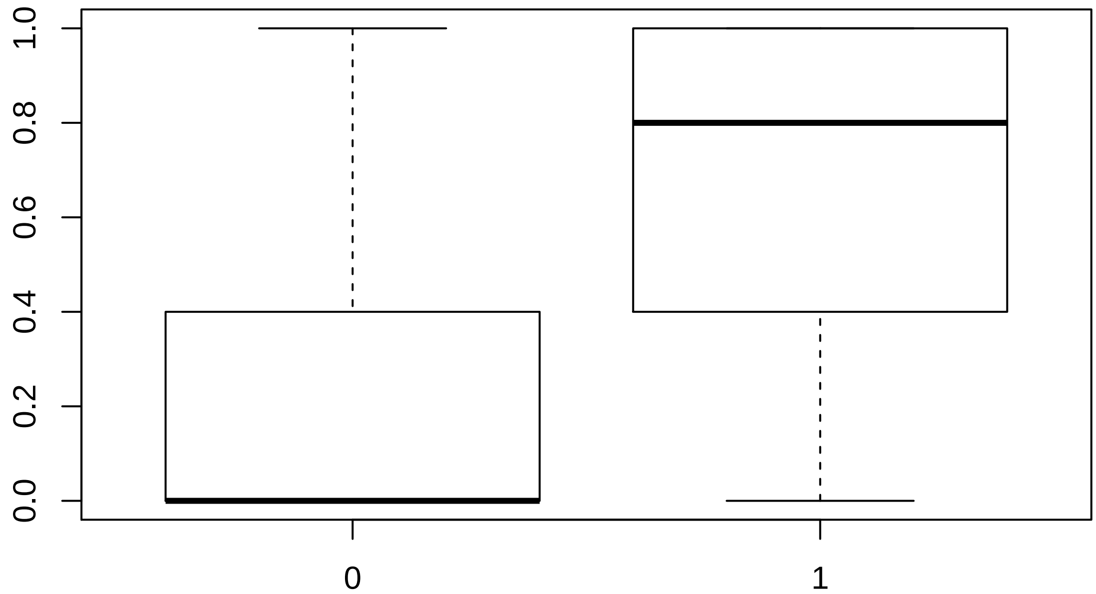
```
AUC(sens=ROC.obj$TP, spec=1-ROC.obj$FP)
```

```
## [1] 0.1487019
```

d. k-nn ({knn} and {knn.cv} from package {class})

```
pred.knn.spam.test<-ifelse(unlist(knn.spam.tr)==1,
       attributes(knn.spam.tr)$prob,
       1-attributes(knn.spam.tr)$prob)

boxplot(pred.knn.spam.test ~ spam.test.df$spam.01)
```



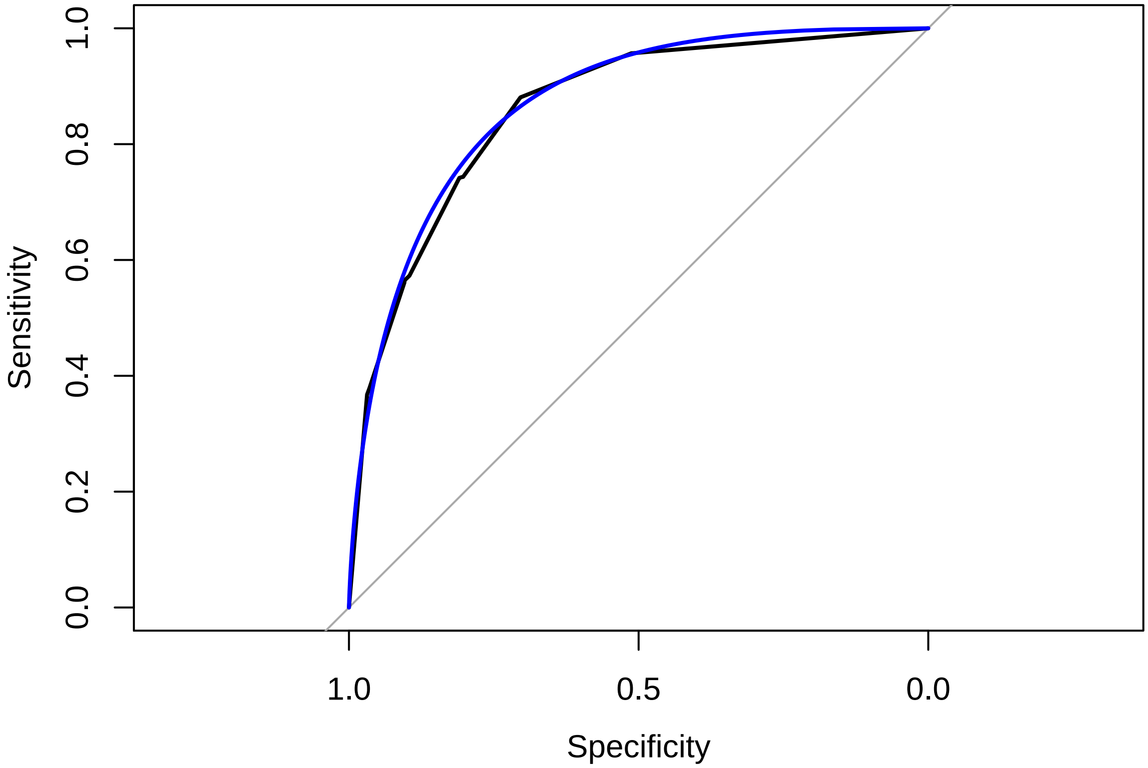

```
table2<-table( spam.test.df$spam.01, pred.knn.spam.test )
table2p<-prop.table(table2)
table2t<-table2p[1,1]+table2p[2,2];table2t
```

```
## [1] 0.3091977
```

```
roc(spam.test.df$spam.01 ~ pred.knn.spam.test, plot=TRUE)
```

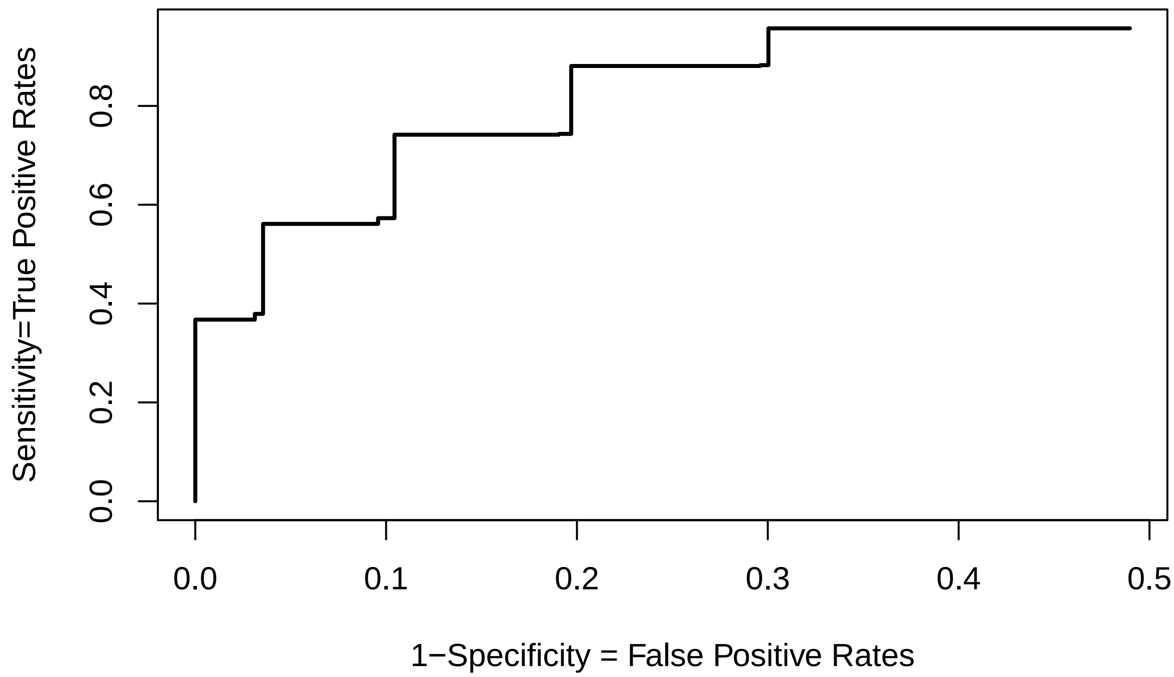
```
##
## Call:
## roc.formula(formula = spam.test.df$spam.01 ~ pred.knn.spam.test,     plot = TRUE)
##
## Data: pred.knn.spam.test in 929 controls (spam.test.df$spam.01 0) < 604 cases (spam.test.df$spam.01 
## Area under the curve: 0.8598
```

```
roc(spam.test.df$spam.01 ~ pred.knn.spam.test, smooth=TRUE, plot=TRUE, add=TRUE, col=4)
```




```
##
## Call:
## roc.formula(formula = spam.test.df$spam.01 ~ pred.knn.spam.test,     smooth = TRUE, plot = TRUE, add
##
## Data: pred.knn.spam.test in 929 controls (spam.test.df$spam.01 0) < 604 cases (spam.test.df$spam.01 1
## Smoothing: binormal
## Area under the curve: 0.8695
```

```
J <- 201
cut.points <- (0:J)/J
ROC.obj <- ROC(status=spam.test.df$spam.01, marker=pred.knn.spam.test, cut.values=cut.points)
plot(ROC.obj$FP, ROC.obj$TP, ylab="Sensitivity=True Positive Rates",
     xlab="1-Specificity = False Positive Rates", type="s", lwd=2)
```




```
ROC.obj$AUC
```

```
## [1] 0.8597643
```

```
AUC(sens=ROC.obj$TP, spec=1-ROC.obj$FP)
```

```
## [1] 0.3928671
```

Els valors de l'AUC(area sota la corba) s'empren en contexts de Machine learning com a estadistic per comparar models. Si l'AUC pren valor 1 el model es perfecte. Si l'AUC pren valor 0.5 el model es igual de be que una classificacio atzarosa. Si el model pren valor 0, es un model inutil ja que no classifica res be.

A traves de les corbes ROC es poden fer altres tipus de mesures per con?ixer la qualitat dels models de machine learning. Un exemple es el cas de l'estadistic J de Youden. L'index de Youden està definit com:

L'index de Youden pot prendre valors entre -1 i 1. Si l'estadistic pren valor 1, el model de classificacio es perfecte. Si l'estad?stic pren valors entre -0.5 o 0.5 el classificador no t? cap utilitat, es pur soroll. Si l'estad?stic pren valor -1, el model classifica just de forma contraria a la forma correcte, aquest es un cas pervers pero util en que hauriem d'adjudicar els objectes l'etiqueta contraria de la predita, obtenint aixo una classificacio perfecta.

5. Calcula también la tasa de error de cada regla cuando se usa $c = 1/2$.

```
### a. Regresión logística estimada por máxima verosimilitud (IRWLS, {\tt glm}).
a5<-1 - sum(diag(table( spam.test.df$spam.01, pred.glm.spam.test>.5 )))/n.test
```

```
### b. Regresión logística estimada mediante Lasso ({\tt glment}).
b5<-1 - sum(diag(table( spam.test.df$spam.01, pred.glmnet.spam.test>.5 )))/n.test
### c. Red neuronal ({\tt nnet})
c5<-1 - sum(diag(table( spam.test.df$spam.01, pred.nnet.spam.test>.5 )))/n.test
### d. k-nn ({\tt knn} and {\tt knn.cv} from package {\tt class})
d5<-1 - sum(diag(table( spam.test.df$spam.01, pred.knn.spam.test>.5 )))/n.test
#1 - sum(diag(table( spam.test.df$spam.01, pred.cv.knn.spam.test>.5 )))/n.test

aux<-cbind(c("Regresión logística estimada por máxima verosimilitud (IRWLS, {\tt glm})",
  "Regresión logística estimada mediante Lasso ({\tt glment})",
  "Red neuronal ({\tt nnet})",
  "k-nn ({\tt knn} and {\tt knn.cv} from package {\tt class})"),c(a5,b5,c5,d5))

colnames(aux)<-c("Regla","tasa de error")
kable(aux)
```

Regla	tasa de error
Regresin logstica estimada por mxima verosimilitud (IRWLS, { t glm})	0.072
Regresin logstica estimada mediante Lasso ({ t glment})	0.072
Red neuronal ({ t nnet})	0.065
k-nn ({ t knn} and { t knn.cv} from package { t class})	0.217

6, Calcula ℓ_{val} para cada regla.

```
epsilon<-.Machine$double.eps

### a. Regresión logística estimada por máxima verosimilitud (IRWLS, {\tt glm}).
a6<-mean( spam.test.df$spam.01*log(pred.glm.spam.test+epsilon) +
                   (1-spam.test.df$spam.01+epsilon)*log(1-pred.glm.spam.test))

### b. Regresión logística estimada mediante Lasso ({\tt glment}).
b6<-sum(spam.test.df$spam.01*log(pred.glm.spam.test+epsilon) +
                   (1-spam.test.df$spam.01+epsilon)*log(1-pred.glmnet.spam.test+epsilon),na.rm=TRUE)/su
### c. Red neuronal ({\tt nnet})
c6<-sum( spam.test.df$spam.01*log(pred.nnet.spam.test+epsilon) +
        (1-spam.test.df$spam.01+epsilon)*log(1-pred.nnet.spam.test+epsilon))/sum(!is.na(pred.nnet.spam.
### d. k-nn ({\tt knn} and {\tt knn.cv} from package {\tt class})
d6<-sum( spam.test.df$spam.01*log(pred.glm.spam.test+epsilon) +
                   (1-spam.test.df$spam.01+epsilon)*log(1-pred.knn.spam.test+epsilon))/sum(!is.na(pred.


aux<-cbind(c("Regresión logística estimada por máxima verosimilitud (IRWLS, {\tt glm})",
  "Regresión logística estimada mediante Lasso ({\tt glment})",
  "Red neuronal ({\tt nnet})",
  "k-nn ({\tt knn} and {\tt knn.cv} from package {\tt class})"),c(a6,b6,c6,d6))
colnames(aux)<-c("Regla","\ell_{\mbox{val}}")
kable(aux)
```

Regla	ℓ_{val}
Regresin logstica estimada por mxima verosimilitud (IRWLS, { t glm})	-0.208

Regla	ℓ_{val}
Regresin logstica estimada mediante Lasso ({ t glment})	-0.211
Red neuronal ({ t nnet})	-0.649
k-nn ({ t knn} and { t knn.cv} from package { t class})	-0.952