Lending Club

Álvaro de Prada

16/11/2017

-PARTE 1: CARGA Y LIMPIEZA DE LOS DATOS-

Procedemos a cargar los datos. Los campos están separados por "," y los decimales se expresan con".".

```
 library(readr)
loanstats<-read.csv("LoanStats3a.csv", skip = 1, header=T, sep=",", dec=".", fill = T)
```

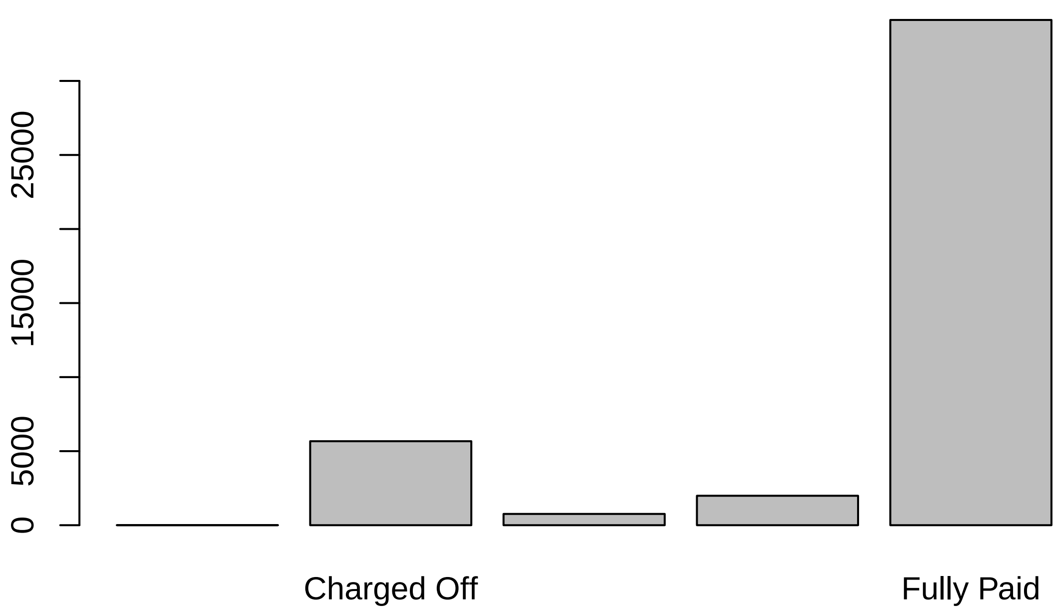
En este caso de predicción, la variable explicativa que queremos predecir es si el préstamos se va a pagar o no. Esto es expresado en el campo loan_status. Vamos a emplear "unique" para ver que tipos de resultados existen en esta variable:

```
unique(loanstats$loan_status, incomparables = FALSE, fromLast = FALSE,
        nmax = NA)
```

```
## [1] Fully Paid
## [2] Charged Off
## [3]
## [4] Does not meet the credit policy. Status:Fully Paid
## [5] Does not meet the credit policy. Status:Charged Off
## 5 Levels:  ... Fully Paid
```

Comprobamos que la variable puede tomar valores en blanco los cuales no podemos interpretar, por lo que eliinamos todas las observaciones que tomen ese valor:

```
plot(loanstats$loan_status)
```




```
table(loanstats$loan_status)
```

```
##
##
```

```
##                                                3
##                                      Charged Off
##                                             5670
## Does not meet the credit policy. Status:Charged Off
##                                              761
##  Does not meet the credit policy. Status:Fully Paid
##                                             1988
##                                       Fully Paid
##                                            34116
```

```r
loanstats<-loanstats[loanstats$loan_status!="",]
```

También vemos que el valor "Does not meet the credit policy. Status:Charged Off" es igual que "Charged Off" y que "Does not meet the credit policy. Status:Fully Paid" se corresponde con "Fully Paid", por lo que lo formateamos par hacer 2 valores únicos.

```r
table(loanstats$loanstats.loan_status)
```

```
## < table of extent 0 >
```

```r
loanstats$loan_status = gsub("Does not meet the credit policy. Status:Charged Off",
                             "Charged Off",
                             loanstats$loan_status)
loanstats$loan_status = gsub("Does not meet the credit policy. Status:Fully Paid",
                             "Fully Paid",
                             loanstats$loan_status)
```

Una vez hecho esto, la variable explicativa se convierte en una variable dicotómica de pagado/no pagado.

Comprobamos el número de préstamos que existen de cada tipo, es decir, sobre el total inicial de 42535 prestamos, 6431 no fueron pagados mientras que 36104 sí fueron pagados:

```r
EstadoPrest<-loanstats$loan_status
table(EstadoPrest)
```

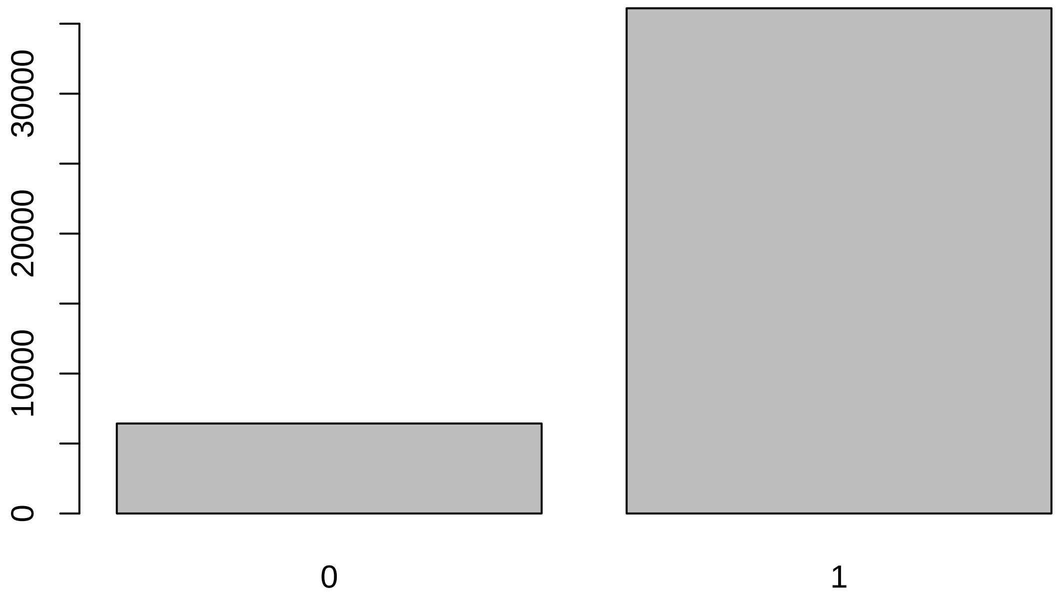
```
## EstadoPrest
## Charged Off  Fully Paid
##        6431       36104
```

Convertimos la variable en dicotómica (1/0) y la representamos gráficamente:

```r
class(EstadoPrest)
```

```
## [1] "character"
```

```r
EstadoPrest <- factor(EstadoPrest, labels=0:1)
plot( EstadoPrest)
```




```
class(EstadoPrest)
```

```
## [1] "factor"
```

Tras haber hecho una limpieza inicial de los datos, procedemos a reducir el dataset quedándonos sólo con aquellas variables que puedan ser influyentes a la hora de pagar o no un préstamo. En este caso nos quedamos con un total de 10 variables entre las que se incluyen la variable que queremos predecir (EstadoPrest):

```
cleandata<-data.frame(EstadoPrest, loanstats$dti, loanstats$grade, loanstats$int_rate,
                      loanstats$revol_util, loanstats$annual_inc, loanstats$total_acc,
                      loanstats$loan_amnt, loanstats$total_pymnt, loanstats$total_rec_int)
```

Realizamos los mismos pasos que con EstadoPrest para la variable grade, para comprobar así la limpieza de los datos de la variable.

```
table(cleandata$loanstats.grade)
```

```
##
##             A     B     C     D     E     F     G
##      0 10183 12389  8740  6016  3394  1301   512
```

```
cleandata$loanstats.grade <- factor(cleandata$loanstats.grade, labels=1:7)
table(cleandata$loanstats.grade)
```

```
##
##     1     2     3     4     5     6     7
## 10183 12389  8740  6016  3394  1301   512
```

Comprobamos el formto de los datos de cada una de las variables:

```
typeof(cleandata$EstadoPrest)
```

```
## [1] "integer"
```

```
typeof(cleandata$loanstats.dti)
```

```
## [1] "double"
```

```
typeof(cleandata$loanstats.grade)
```

```
## [1] "integer"
```

```
typeof(cleandata$loanstats.int_rate)
```

```
## [1] "integer"
```

```
typeof(cleandata$loanstats.revol_util)
```

```
## [1] "integer"
```

```
typeof(cleandata$loanstats.annual_inc)
```

```
## [1] "double"
```

```
typeof(cleandata$loanstats.total_acc)
```

```
## [1] "integer"
```

```
typeof(cleandata$loanstats.loan_amnt)
```

```
## [1] "integer"
```

```
typeof(cleandata$loanstats.total_pymnt)
```

```
## [1] "double"
```

```
typeof(cleandata$loanstats.total_rec_int)
```

```
## [1] "double"
```

Las variables int_rate y revol_util están expresadas en porcentajes, por lo que procedemos a formatearlas para que puedan ser utilizadas en nuestro modelo:

```
cleandata$loanstats.int_rate = gsub("%", "",cleandata$loanstats.int_rate)
head(cleandata$loanstats.int_rate)
```

```
## [1] " 10.65" " 15.27" " 15.96" " 13.49" " 12.69" "  7.90"
```

```
cleandata$loanstats.revol_util= gsub("%", "",cleandata$loanstats.revol_util)
head(cleandata$loanstats.revol_util)
```

```
## [1] "83.7" "9.4"  "98.5" "21"   "53.9" "28.3"
```

Y finalmente convertimos todas las variables a numeric (las que eran porcentajes las dividimos por 100 para que estén en la misma escala):

```
cleandata$loanstats.dti<-as.numeric(paste(cleandata$loanstats.dti))
cleandata$loanstats.int_rate<-as.numeric(paste(cleandata$loanstats.int_rate))/100
cleandata$loanstats.revol_util<-as.numeric(paste(cleandata$loanstats.revol_util))/100
cleandata$loanstats.annual_inc<-as.numeric(paste(cleandata$loanstats.annual_inc))
```

```
## Warning: NAs introducidos por coerción
```

```
cleandata$loanstats.total_acc<-as.numeric(paste(cleandata$loanstats.total_acc))
```

```
## Warning: NAs introducidos por coerción
```

```
cleandata$loanstats.total_pymnt<-as.numeric(paste(cleandata$loanstats.total_pymnt))
cleandata$loanstats.loan_amnt<-as.numeric(paste(cleandata$loanstats.loan_amnt))
cleandata$loanstats.total_rec_int<-as.numeric(paste(cleandata$loanstats.total_rec_int))
cleandata$EstadoPrest<-as.numeric(paste(cleandata$EstadoPrest))
head(cleandata)
```

```
##   EstadoPrest loanstats.dti loanstats.grade loanstats.int_rate
## 1           1         27.65               2             0.1065
```

```
## 2           0          1.00               3             0.1527
## 3           1          8.72               3             0.1596
## 4           1         20.00               3             0.1349
## 5           1         17.94               2             0.1269
## 6           1         11.20               1             0.0790
##   loanstats.revol_util loanstats.annual_inc loanstats.total_acc
## 1                0.837                24000                   9
## 2                0.094                30000                   4
## 3                0.985                12252                  10
## 4                0.210                49200                  37
## 5                0.539                80000                  38
## 6                0.283                36000                  12
##   loanstats.loan_amnt loanstats.total_pymnt loanstats.total_rec_int
## 1                5000              5863.155                  863.16
## 2                2500              1014.530                  435.17
## 3                2400              3005.667                  605.67
## 4               10000             12231.890                 2214.92
## 5                3000              4066.908                 1066.91
## 6                5000              5632.210                  632.21
```

```
summary(cleandata)
```

```
##   EstadoPrest     loanstats.dti   loanstats.grade loanstats.int_rate
##  Min.   :0.0000   Min.   : 0.00   1:10183         Min.   :0.0542    
##  1st Qu.:1.0000   1st Qu.: 8.20   2:12389         1st Qu.:0.0963    
##  Median :1.0000   Median :13.47   3: 8740         Median :0.1199    
##  Mean   :0.8488   Mean   :13.37   4: 6016         Mean   :0.1217    
##  3rd Qu.:1.0000   3rd Qu.:18.68   5: 3394         3rd Qu.:0.1472    
##  Max.   :1.0000   Max.   :29.99   6: 1301         Max.   :0.2459    
##                                   7:  512                           
##  loanstats.revol_util loanstats.annual_inc loanstats.total_acc
##  Min.   :0.0000       Min.   :   1896      Min.   : 1.00      
##  1st Qu.:0.2570       1st Qu.:  40000      1st Qu.:13.00      
##  Median :0.4970       Median :  59000      Median :20.00      
##  Mean   :0.4912       Mean   :  69137      Mean   :22.12      
##  3rd Qu.:0.7270       3rd Qu.:  82500      3rd Qu.:29.00      
##  Max.   :1.1900       Max.   :6000000      Max.   :90.00      
##  NA's   :90           NA's   :4            NA's   :29         
##  loanstats.loan_amnt loanstats.total_pymnt loanstats.total_rec_int
##  Min.   :  500       Min.   :    0         Min.   :    0.0        
##  1st Qu.: 5200       1st Qu.: 5464         1st Qu.:  657.1        
##  Median : 9700       Median : 9682         Median : 1339.2        
##  Mean   :11090       Mean   :12019         Mean   : 2240.0        
##  3rd Qu.:15000       3rd Qu.:16426         3rd Qu.: 2803.1        
##  Max.   :35000       Max.   :58886         Max.   :23886.5        
## 
```

Vemos que existen campos NAs pero que no son demasiado significativos en cuanto a cantidad comparado con el total, por lo que procedemos a eliminarlos de nuestro dataset:

```
cleandata = na.omit(cleandata)
summary(cleandata)
```

```
##   EstadoPrest    loanstats.dti   loanstats.grade loanstats.int_rate
##  Min.   :0.000   Min.   : 0.00   1:10171         Min.   :0.0542    
##  1st Qu.:1.000   1st Qu.: 8.21   2:12376         1st Qu.:0.0963    
```

```
##  Median :1.000   Median :13.48   3: 8719         Median :0.1199
##  Mean   :0.849   Mean   :13.38   4: 5996         Mean   :0.1216
##  3rd Qu.:1.000   3rd Qu.:18.69   5: 3380         3rd Qu.:0.1472
##  Max.   :1.000   Max.   :29.99   6: 1294         Max.   :0.2459
##                                  7:  509
##  loanstats.revol_util loanstats.annual_inc loanstats.total_acc
##  Min.   :0.0000       Min.   :   1896      Min.   : 1.00
##  1st Qu.:0.2570       1st Qu.:  40000      1st Qu.:13.00
##  Median :0.4970       Median :  59000      Median :20.00
##  Mean   :0.4912       Mean   :  69169      Mean   :22.14
##  3rd Qu.:0.7270       3rd Qu.:  82500      3rd Qu.:29.00
##  Max.   :1.1900       Max.   :6000000      Max.   :90.00
##
##  loanstats.loan_amnt loanstats.total_pymnt loanstats.total_rec_int
##  Min.   :  500       Min.   :    0         Min.   :    0.0
##  1st Qu.: 5200       1st Qu.: 5477         1st Qu.:  658.4
##  Median : 9800       Median : 9706         Median : 1342.0
##  Mean   :11103       Mean   :12035         Mean   : 2243.0
##  3rd Qu.:15000       3rd Qu.:16435         3rd Qu.: 2806.4
##  Max.   :35000       Max.   :58886         Max.   :23886.5
##
```

```r
sapply(cleandata, class)
```

```
##             EstadoPrest           loanstats.dti         loanstats.grade
##               "numeric"               "numeric"                "factor"
##      loanstats.int_rate    loanstats.revol_util    loanstats.annual_inc
##               "numeric"               "numeric"               "numeric"
##     loanstats.total_acc     loanstats.loan_amnt   loanstats.total_pymnt
##               "numeric"               "numeric"               "numeric"
## loanstats.total_rec_int
##               "numeric"
```

-PARTE 2: MODELO-

Hacemos un modelo de regresión inicial con todas las variables seleccionadas para comprobar como se comporta:

```r
modelo01=glm(EstadoPrest~loanstats.int_rate+loanstats.revol_util+loanstats.dti+
               loanstats.int_rate+loanstats.revol_util+loanstats.annual_inc+
               loanstats.total_acc+loanstats.total_pymnt+loanstats.loan_amnt+
            loanstats.total_rec_int+loanstats.grade,family="binomial",data=cleandata)
summary(modelo01)
```

```
##
## Call:
## glm(formula = EstadoPrest ~ loanstats.int_rate + loanstats.revol_util +
##     loanstats.dti + loanstats.int_rate + loanstats.revol_util +
##     loanstats.annual_inc + loanstats.total_acc + loanstats.total_pymnt +
##     loanstats.loan_amnt + loanstats.total_rec_int + loanstats.grade,
##     family = "binomial", data = cleandata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.3208   0.0160   0.1032   0.2779   5.3003
```

```
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             9.580e-01  1.897e-01   5.051 4.40e-07 ***
## loanstats.int_rate      1.352e+00  2.244e+00   0.602 0.546856
## loanstats.revol_util   -2.652e-01  9.274e-02  -2.860 0.004243 **
## loanstats.dti          -5.815e-03  3.804e-03  -1.529 0.126338
## loanstats.annual_inc    2.141e-06  7.329e-07   2.921 0.003484 **
## loanstats.total_acc     2.438e-03  2.459e-03   0.991 0.321561
## loanstats.total_pymnt   1.353e-03  1.925e-05  70.294  < 2e-16 ***
## loanstats.loan_amnt    -7.820e-04  1.189e-05 -65.743  < 2e-16 ***
## loanstats.total_rec_int -2.022e-03  4.113e-05 -49.167  < 2e-16 ***
## loanstats.grade2       -4.501e-01  1.057e-01  -4.257 2.07e-05 ***
## loanstats.grade3       -5.074e-01  1.521e-01  -3.336 0.000849 ***
## loanstats.grade4       -6.977e-01  1.916e-01  -3.641 0.000272 ***
## loanstats.grade5       -2.351e-01  2.304e-01  -1.020 0.307520
## loanstats.grade6       -5.933e-02  3.000e-01  -0.198 0.843246
## loanstats.grade7       -3.265e-01  3.584e-01  -0.911 0.362257
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 36030  on 42444  degrees of freedom
## Residual deviance: 12618  on 42430  degrees of freedom
## AIC: 12648
##
## Number of Fisher Scoring iterations: 8
```

Vemos que tras hacer el summary se consideran significativas algunas variables.

Procedemos a crear nuestros datasets de entrenamiento y de validación:

```
set.seed(100)
n=nrow(cleandata)
id_train <- sample(1:n , 0.90*n)
cleandata.train = cleandata[id_train,]
cleandata.test = cleandata[-id_train,]
```

Y volvemos a comprobar la regresión logística con todas las variables seleccionadas para nuestro modelo de entrenamiento:

```
cleandata.glm0<-glm(EstadoPrest~loanstats.int_rate+loanstats.dti+loanstats.revol_util+
                      loanstats.dti+loanstats.grade+loanstats.int_rate+loanstats.revol_util+
                      loanstats.annual_inc+loanstats.total_pymnt+loanstats.loan_amnt+
                      loanstats.total_rec_int,family=binomial,cleandata.train)
summary(cleandata.glm0)
```

```
##
## Call:
## glm(formula = EstadoPrest ~ loanstats.int_rate + loanstats.dti +
##     loanstats.revol_util + loanstats.dti + loanstats.grade +
##     loanstats.int_rate + loanstats.revol_util + loanstats.annual_inc +
##     loanstats.total_pymnt + loanstats.loan_amnt + loanstats.total_rec_int,
##     family = binomial, data = cleandata.train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.2935   0.0166   0.1044   0.2791   5.2789
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                1.012e+00  1.975e-01   5.125 2.98e-07 ***
## loanstats.int_rate         1.058e+00  2.362e+00   0.448 0.654243
## loanstats.dti             -4.558e-03  3.735e-03  -1.220 0.222399
## loanstats.revol_util      -3.445e-01  9.720e-02  -3.544 0.000393 ***
## loanstats.grade2          -4.157e-01  1.111e-01  -3.742 0.000183 ***
## loanstats.grade3          -4.786e-01  1.601e-01  -2.989 0.002797 **
## loanstats.grade4          -6.466e-01  2.018e-01  -3.205 0.001352 **
## loanstats.grade5          -1.728e-01  2.422e-01  -0.713 0.475578
## loanstats.grade6           3.423e-03  3.152e-01   0.011 0.991334
## loanstats.grade7          -3.535e-01  3.766e-01  -0.939 0.347812
## loanstats.annual_inc       2.596e-06  7.195e-07   3.608 0.000308 ***
## loanstats.total_pymnt      1.342e-03  2.013e-05  66.678  < 2e-16 ***
## loanstats.loan_amnt       -7.770e-04  1.244e-05 -62.461  < 2e-16 ***
## loanstats.total_rec_int   -1.999e-03  4.326e-05 -46.212  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 32357  on 38199  degrees of freedom
## Residual deviance: 11425  on 38186  degrees of freedom
## AIC: 11453
##
## Number of Fisher Scoring iterations: 8
```

Comprobamos que es bastante similar al modelo inicial con todo el dataset.

Construimos un segundo modelo quitando aquellas variables no significativas:

```r
cleandata.glm1<-glm(EstadoPrest~loanstats.revol_util+loanstats.grade+
                    loanstats.revol_util+loanstats.annual_inc+loanstats.total_pymnt+
                     loanstats.loan_amnt+loanstats.total_rec_int,family=binomial,cleandata.train)
summary(cleandata.glm1)
```

```
##
## Call:
## glm(formula = EstadoPrest ~ loanstats.revol_util + loanstats.grade +
##     loanstats.revol_util + loanstats.annual_inc + loanstats.total_pymnt +
##     loanstats.loan_amnt + loanstats.total_rec_int, family = binomial,
##     data = cleandata.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.2980   0.0166   0.1044   0.2793   5.2774
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                1.039e+00  8.003e-02  12.979  < 2e-16 ***
## loanstats.revol_util      -3.679e-01  9.360e-02  -3.931 8.47e-05 ***
## loanstats.grade2          -3.783e-01  7.607e-02  -4.973 6.58e-07 ***
```

```
## loanstats.grade3        -4.137e-01  8.244e-02  -5.019 5.20e-07 ***
## loanstats.grade4        -5.595e-01  9.283e-02  -6.027 1.67e-09 ***
## loanstats.grade5        -7.068e-02  1.205e-01  -0.587 0.557319
## loanstats.grade6         1.231e-01  1.917e-01   0.642 0.520729
## loanstats.grade7        -2.361e-01  2.631e-01  -0.897 0.369467
## loanstats.annual_inc     2.680e-06  7.158e-07   3.744 0.000181 ***
## loanstats.total_pymnt    1.342e-03  2.011e-05  66.750  < 2e-16 ***
## loanstats.loan_amnt     -7.778e-04  1.242e-05 -62.620  < 2e-16 ***
## loanstats.total_rec_int -1.996e-03  4.216e-05 -47.341  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 32357  on 38199  degrees of freedom
## Residual deviance: 11427  on 38188  degrees of freedom
## AIC: 11451
##
## Number of Fisher Scoring iterations: 8
```

Comprobamos los estadísticos AIC y BIC para ver que modelo predice mejor.

```
AIC(cleandata.glm0)
```

```
## [1] 11453.05
```

```
AIC(cleandata.glm1)
```

```
## [1] 11450.79
```

```
BIC(cleandata.glm0)
```

```
## [1] 11572.76
```

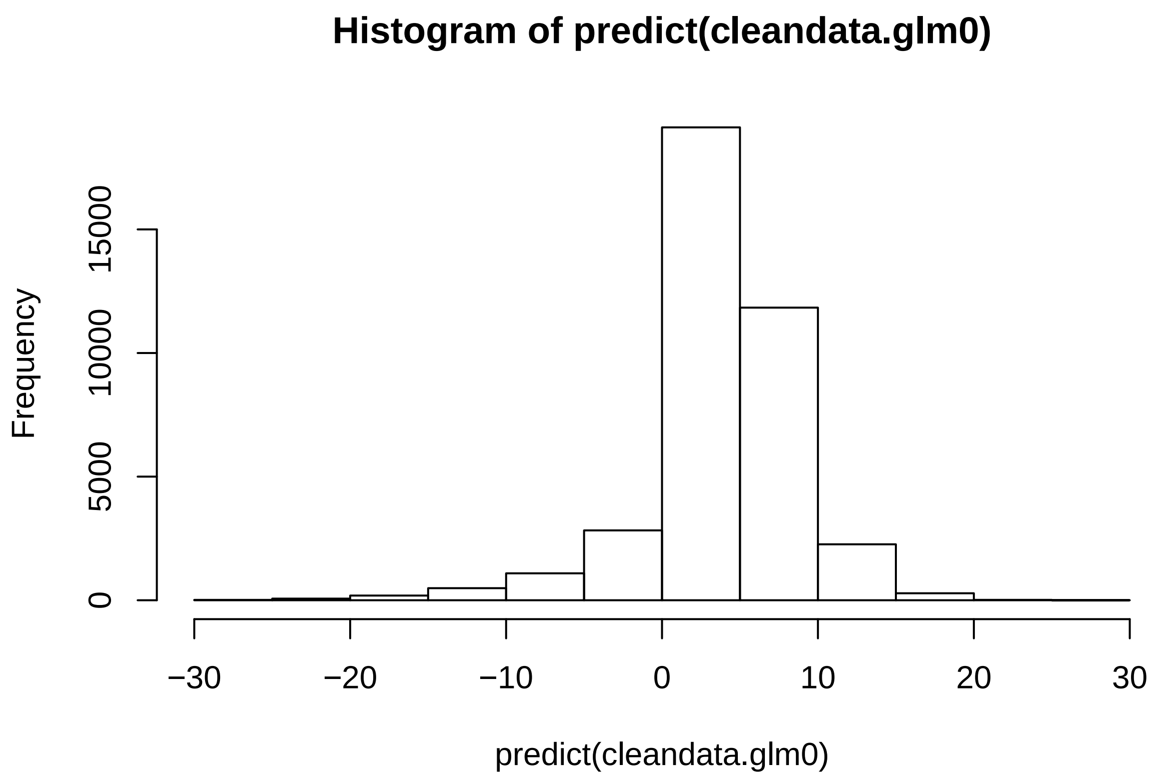
```
BIC(cleandata.glm1)
```

```
## [1] 11553.39
```

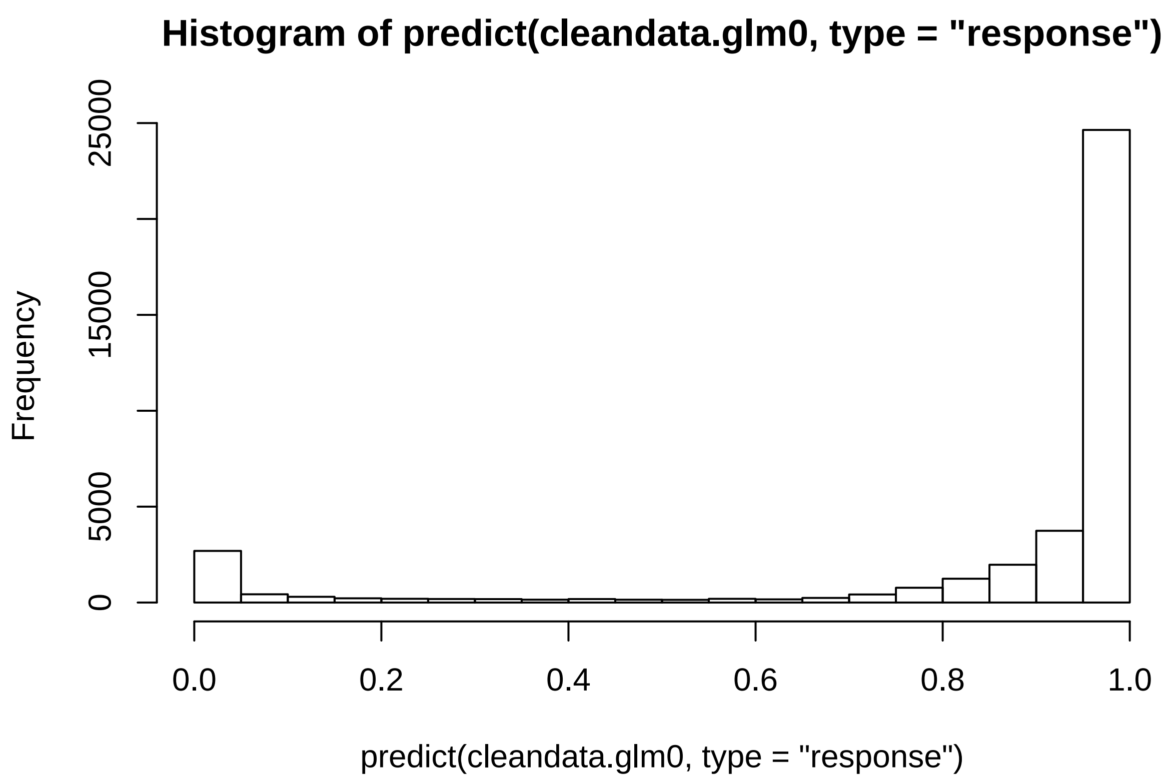
Y vemos que el modelo glm0 con todas las variables iniciales seleccionadas es sensiblemente mejor que el segundo modelo.

Procedemos a representar el primer modelo:

```
hist(predict(cleandata.glm0))
```




```r
hist(predict(cleandata.glm0, type = "response"))
```



Comprobamos varias probabilidades de corte para elegir con cual quedarnos:

```r
table(predict(cleandata.glm0,type="response") > 0.5)
```

```
##
## FALSE  TRUE
##  4674 33526
```

```r
table(predict(cleandata.glm0,type="response") > 0.3)
```

```
##
## FALSE  TRUE
##  4019 34181
```

```r
table(predict(cleandata.glm0,type="response") > 0.2)
```

```
##
## FALSE  TRUE
##  3640 34560
```

Nos quedaremos con la 0.2.

Llevamos a cabo la predicción dentro de la muestra de entrenamiento:

```r
prob.glm0.insample <- predict(cleandata.glm0,type="response")
predicted.glm0.insample <- prob.glm0.insample > 0.2
predicted.glm0.insample <- as.numeric(predicted.glm0.insample)
table(cleandata.train$EstadoPrest, predicted.glm0.insample, dnn=c("Realidad","Predicho"))
```

```
##         Predicho
## Realidad     0     1
##        0  3487  2261
##        1   153 32299
```

```r
mean(ifelse(cleandata.train$EstadoPrest != predicted.glm0.insample, 1, 0))
```

```
## [1] 0.06319372
```

Vemos que dentro de los resultados obtenidos son: - 3487 préstamos no pagados predichos como no pagados. Correcto. - 32299 préstamos pagados predichos como pagados. Correcto. - 2261 préstamos no pagados predichos como pagados. Error. - 153 préstamos pagados predichos como no pagados. Error.

Y a continuación realizamos la predicción dentro de la muestra de validación:

```r
prob.glm0.outsample <- predict(cleandata.glm0,cleandata.test,type="response")
predicted.glm0.outsample <-  prob.glm0.outsample> 0.2
predicted.glm0.outsample <- as.numeric(predicted.glm0.outsample)
table(cleandata.test$EstadoPrest, predicted.glm0.outsample, dnn=c("Realidad","Predicho"))
```

```
##         Predicho
## Realidad    0    1
##        0  399  262
##        1   17 3567
```

```r
mean(ifelse(cleandata.test$EstadoPrest != predicted.glm0.outsample, 1, 0))
```

```
## [1] 0.06572438
```

Obtenemos resultados muy parecidos a los anteriores.

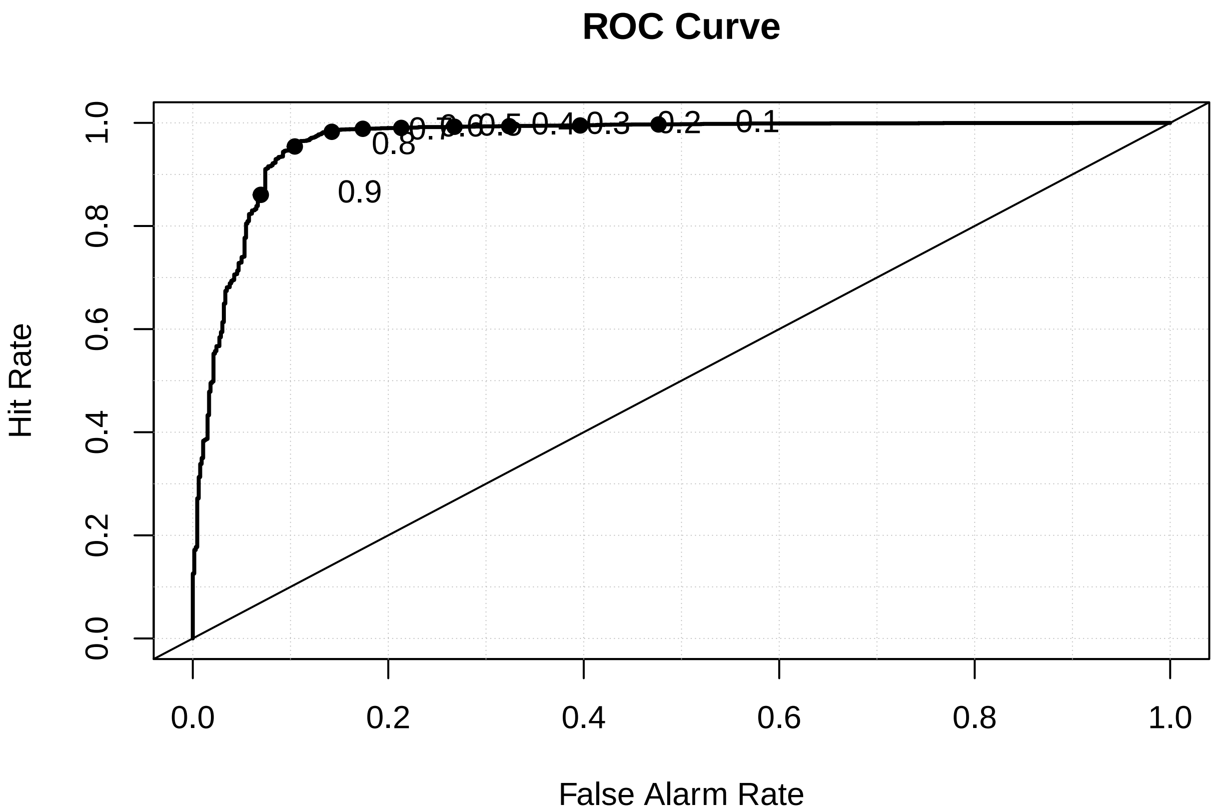
Realizamos la curva ROC:

```r
library(verification)
```

```
## Loading required package: fields

## Loading required package: spam

## Loading required package: dotCall64

## Loading required package: grid

## Spam version 2.1-1 (2017-07-02) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: maps

## Loading required package: boot

## Loading required package: CircStats

## Loading required package: MASS

## Loading required package: dtw

## Loading required package: proxy

## Warning: package 'proxy' was built under R version 3.4.2

##
## Attaching package: 'proxy'

## The following object is masked from 'package:spam':
##
##     as.matrix

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

## Loaded dtw v1.18-1. See ?dtw for help, citation("dtw") for use in publication.
```

```
prob.glm0.outsample <- predict(cleandata.glm0,cleandata.test,type="response")
roc.plot(cleandata.test$EstadoPrest == '1', prob.glm0.outsample)
```

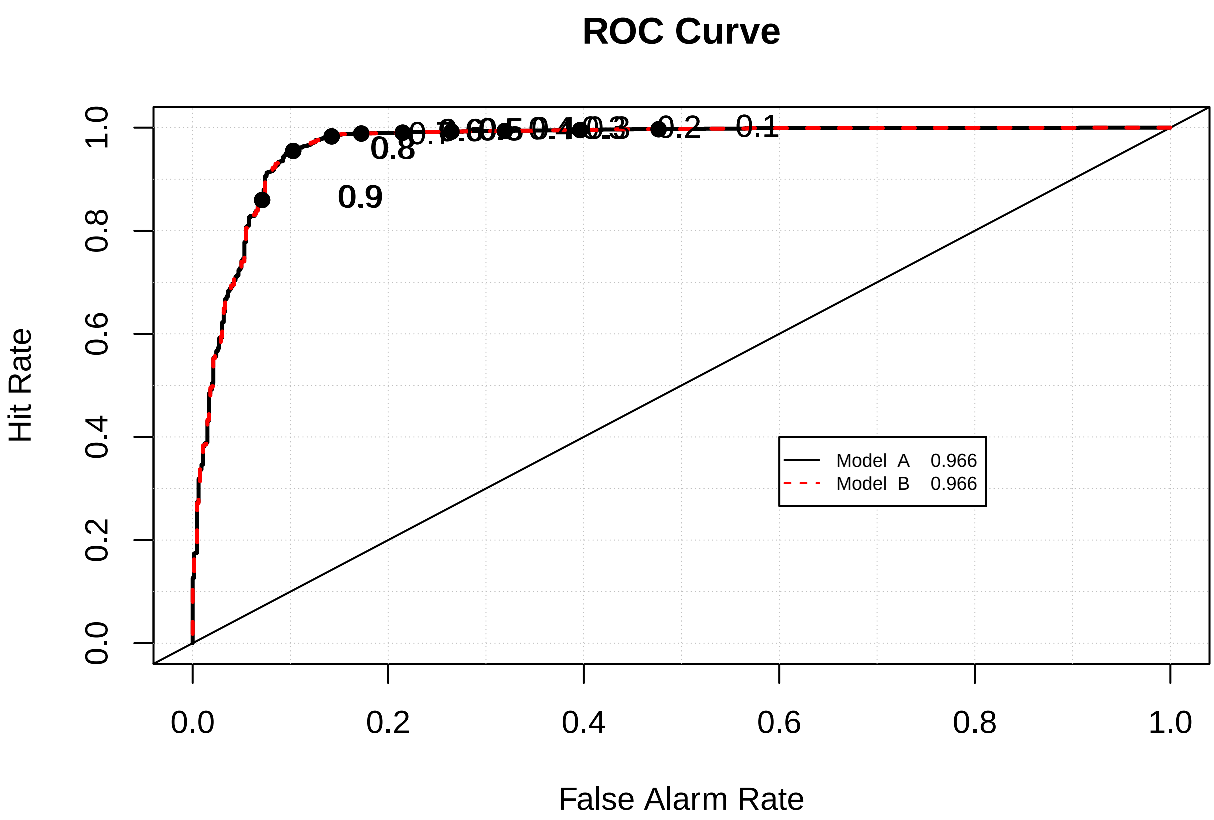
```
## Warning in roc.plot.default(cleandata.test$EstadoPrest == "1",
## prob.glm0.outsample): Large amount of unique predictions used as
## thresholds. Consider specifying thresholds.
```



Comparamos los modelos 1 y 2

```
prob.glm1.outsample <- predict(cleandata.glm1,cleandata.test,type="response")
roc.plot(x= cleandata.test$EstadoPrest == '1',
         pred=cbind(prob.glm1.outsample,prob.glm0.outsample),
         legend=TRUE)$roc.vol
```

```
## Warning in roc.plot.default(x = cleandata.test$EstadoPrest == "1", pred
## = cbind(prob.glm1.outsample, : Large amount of unique predictions used as
## thresholds. Consider specifying thresholds.
```



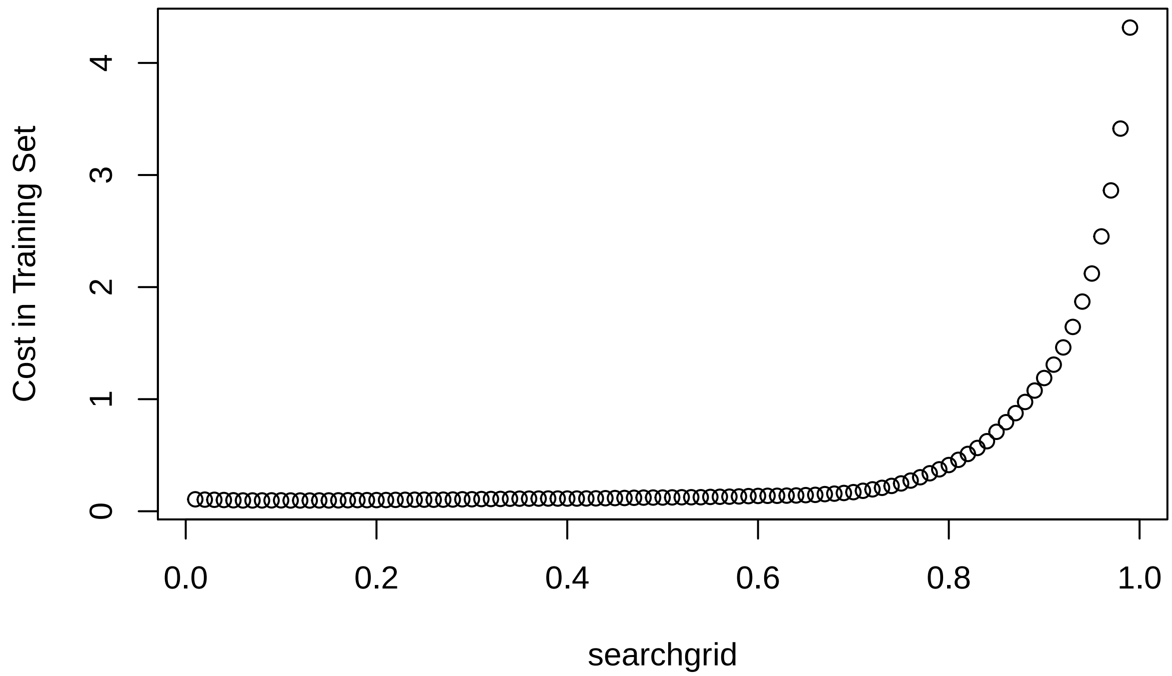

```
##       Model       Area p.value binorm.area
## 1 Model   1 0.9659303       0          NA
## 2 Model   2 0.9659235       0          NA
```

Y comprobamos que ambos modelos son muy similares, realizando predicciones bastante correctas en ambos casos.

Pr último, comprobamos el valor de probabilidad de corte optimo:

```
searchgrid = seq(0.01, 0.99, 0.01)

result = cbind(searchgrid, NA)

cost1 <- function(r, pi){
  weight1 = 10
  weight0 = 1
  c1 = (r==1)&(pi<pcut)
  c0 = (r==0)&(pi>pcut)
  return(mean(weight1*c1+weight0*c0))
}
cleandata.glm0<-glm(EstadoPrest~loanstats.int_rate+loanstats.dti+loanstats.revol_util+
                      loanstats.dti+loanstats.grade+loanstats.int_rate+
                      loanstats.revol_util+loanstats.annual_inc+loanstats.total_pymnt+
                      loanstats.loan_amnt+loanstats.total_rec_int,family=binomial,cleandata.train)
prob <- predict(cleandata.glm1,type="response")
for(i in 1:length(searchgrid))
{
  pcut <- result[i,1]

  result[i,2] <- cost1(cleandata.train$EstadoPrest, prob)
```

```
}
plot(result, ylab="Cost in Training Set")
```



Podemos ver en la grafica que a partir de 0.7 empieza a variar y el error empieza a ser mayor. Podremos considerar como óptimo un 0.6 para nuestro modelo.

```
result[which.min(result[,2]),]
```

```
## searchgrid
## 0.13000000 0.09599476
```

El minimo punto de corte sería 0.07