Caso Analítica de Marketing

Christian Vasquez Hernandez

## Práctica de Analítica de Marketing: Selección de Potenciales

El objetivo de esta práctica es seleccionar un conjunto de clientes que serán el objetivo de la próxima campaña comercial de venta de seguros de ahorro.

Para ello se aporta la siguiente documentación:

* Fichero (“Campaña Historica Seguro Ahorro.csv”) con resultados históricos.
* Fichero (“Campaña Nueva Seguro de Ahorro.csv”) con clientes actuales de la entidad de los que se quiere extraer un conjunto de potenciales.

Como datos significativos indicar que cada llamada a un cliente tiene un coste de 8 euros y la venta de un seguro de ahorro reporta un beneficio de 10 euros.

## Entregables

Para presentar la campaña será necesario aportar la siguiente documentación:

La selección de potenciales tiene que realizarse con el objetivo de maximizar el beneficio de la campaña.

* Script de r con el procedimiento realizado para extraer los potenciales.
* Fichero en formato Word o PDF con la explicación de los procedimientos realizados y las decisiones tomadas.
* Fichero (“Campaña Nueva Seguro de Ahorro.csv”) con las probabilidades asignadas a cada cliente en una variable llamada PROBABILIDAD y una variable llamada POTENCIAL con valor 1 si el cliente es seleccionado para la campaña y 0 si el cliente no es seleccionado.

## Instrucciones y observaciones

La práctica se deberá realizar utilizando R como herramienta.

Se puede utilizar cualquier técnica de clasificación aprendida durante el Master.

Al final de la práctica se publicará un ranking con los resultados de las campañas de los alumnos ordenado por el beneficio obtenido en la campaña.

# Solución

Para la ejecución de este documento es importante que estén cargadas las librerias ROCR y CaTools mediante los comandos library(ROCR) y library(caTools)

Fijamos el directorio de trabajo en la carpeta donde esté contenido el fichero “Campaña Historica Seguro Ahorro.csv” y “Campaña Nueva Seguro de Ahorro.csv” mediante el comando setwd.

Comenzamos con los carga del dataset en dos dataframe uno con informacion historica y otra con la informacion nueva.

df\_segurohistorica=read.csv2("Campaña Historica Seguro Ahorro.csv")  
df\_seguronueva=read.csv2("Campaña Nueva Seguro Ahorro.csv")

df\_segurohistorica$CAMP\_DEPOSITOS=as.factor(df\_segurohistorica$CAMP\_DEPOSITOS)

### Revisamos dataset y variables

Comenzamos analizando la estructura del dataset, y las variables que tienen:

**Información historica**

str(df\_segurohistorica)

## 'data.frame': 37500 obs. of 7 variables:  
## $ COD\_CLIENTE : Factor w/ 37408 levels "CLI0000000450",..: 30264 24487 3362 2106 7876 5344 10398 17120 4711 5963 ...  
## $ CAT\_EDAD : Factor w/ 4 levels "1\_NULL","2\_MINUS30",..: 2 2 4 4 4 4 4 4 4 4 ...  
## $ ESTADO\_CIVIL : Factor w/ 2 levels "CASADO","SOLTERO": 2 2 2 1 1 1 1 2 1 2 ...  
## $ NIVEL\_ESTUDIOS: Factor w/ 4 levels "PRIMARIA","SECUNDARIA",..: 2 1 2 4 4 2 4 2 2 4 ...  
## $ RANGO\_INGRESOS: Factor w/ 4 levels "[1000-2000>",..: 2 1 2 2 4 1 1 2 2 1 ...  
## $ CAMP\_DEPOSITOS: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...  
## $ SEXO : Factor w/ 2 levels "HOMBRE","MUJER": 1 2 1 1 1 2 1 1 2 1 ...

Los primeros 6 registros del dataset para tener una primera impresión del contenido del dataset.

head(df\_segurohistorica)

## COD\_CLIENTE CAT\_EDAD ESTADO\_CIVIL NIVEL\_ESTUDIOS RANGO\_INGRESOS  
## 1 CLI0046780637 2\_MINUS30 SOLTERO SECUNDARIA [600-1000>  
## 2 CLI0043396577 2\_MINUS30 SOLTERO PRIMARIA [1000-2000>  
## 3 CLI0006768677 4\_PLUS40 SOLTERO SECUNDARIA [600-1000>  
## 4 CLI0003847700 4\_PLUS40 CASADO UNIVERSIDAD [600-1000>  
## 5 CLI0010350442 4\_PLUS40 CASADO UNIVERSIDAD >=2000  
## 6 CLI0008678857 4\_PLUS40 CASADO SECUNDARIA [1000-2000>  
## CAMP\_DEPOSITOS SEXO  
## 1 0 HOMBRE  
## 2 0 MUJER  
## 3 0 HOMBRE  
## 4 0 HOMBRE  
## 5 0 HOMBRE  
## 6 0 MUJER

Los ultimos 6 registros del dataset para tener una primera impresión del contenido del dataset.

tail(df\_segurohistorica)

## COD\_CLIENTE CAT\_EDAD ESTADO\_CIVIL NIVEL\_ESTUDIOS RANGO\_INGRESOS  
## 37495 R\_10457703398 2\_MINUS30 SOLTERO SECUNDARIA [1000-2000>  
## 37496 CLI0021809054 4\_PLUS40 CASADO UNIVERSIDAD [600-1000>  
## 37497 CLI0045075719 2\_MINUS30 SOLTERO SECUNDARIA [600-1000>  
## 37498 CLI0048289519 1\_NULL SOLTERO SIN\_ESTUDIOS [600-1000>  
## 37499 CLI0023569318 4\_PLUS40 SOLTERO SECUNDARIA [600-1000>  
## 37500 CLI0010490840 3\_30\_40 CASADO SECUNDARIA [600-1000>  
## CAMP\_DEPOSITOS SEXO  
## 37495 0 HOMBRE  
## 37496 0 HOMBRE  
## 37497 0 HOMBRE  
## 37498 0 MUJER  
## 37499 0 MUJER  
## 37500 1 MUJER

**Información Nueva**

str(df\_seguronueva)

## 'data.frame': 12500 obs. of 6 variables:  
## $ COD\_CLIENTE : Factor w/ 12493 levels "CLI0000001525",..: 8501 11222 8894 11811 1621 7344 7017 3347 8088 11771 ...  
## $ CAT\_EDAD : Factor w/ 4 levels "1\_NULL","2\_MINUS30",..: 2 1 2 2 4 4 3 3 2 2 ...  
## $ ESTADO\_CIVIL : Factor w/ 2 levels "CASADO","SOLTERO": 2 2 2 2 2 2 2 2 2 2 ...  
## $ NIVEL\_ESTUDIOS: Factor w/ 4 levels "PRIMARIA","SECUNDARIA",..: 2 3 1 1 4 2 1 1 2 3 ...  
## $ RANGO\_INGRESOS: Factor w/ 4 levels "[1000-2000>",..: 1 3 1 3 1 1 2 1 2 2 ...  
## $ SEXO : Factor w/ 2 levels "HOMBRE","MUJER": 1 1 2 1 2 2 1 2 2 1 ...

Los primeros 6 registros del dataset para tener una primera impresión del contenido del dataset.

head(df\_seguronueva)

## COD\_CLIENTE CAT\_EDAD ESTADO\_CIVIL NIVEL\_ESTUDIOS RANGO\_INGRESOS  
## 1 CLI0043828980 2\_MINUS30 SOLTERO SECUNDARIA [1000-2000>  
## 2 CLI0062632001 1\_NULL SOLTERO SIN\_ESTUDIOS <600  
## 3 CLI0044539972 2\_MINUS30 SOLTERO PRIMARIA [1000-2000>  
## 4 CLI0074041049 2\_MINUS30 SOLTERO PRIMARIA <600  
## 5 CLI0008226026 4\_PLUS40 SOLTERO UNIVERSIDAD [1000-2000>  
## 6 CLI0041998439 4\_PLUS40 SOLTERO SECUNDARIA [1000-2000>  
## SEXO  
## 1 HOMBRE  
## 2 HOMBRE  
## 3 MUJER  
## 4 HOMBRE  
## 5 MUJER  
## 6 MUJER

Los ultimos 6 registros del dataset para tener una primera impresión del contenido del dataset.

tail(df\_seguronueva)

## COD\_CLIENTE CAT\_EDAD ESTADO\_CIVIL NIVEL\_ESTUDIOS RANGO\_INGRESOS  
## 12495 CLI0044212914 2\_MINUS30 SOLTERO SECUNDARIA [600-1000>  
## 12496 CLI0029558992 4\_PLUS40 SOLTERO UNIVERSIDAD [1000-2000>  
## 12497 CLI0016501677 4\_PLUS40 SOLTERO SECUNDARIA <600  
## 12498 CLI0018192189 4\_PLUS40 SOLTERO SECUNDARIA <600  
## 12499 CLI0046003472 2\_MINUS30 SOLTERO SECUNDARIA <600  
## 12500 CLI0016729552 4\_PLUS40 SOLTERO SECUNDARIA >=2000  
## SEXO  
## 12495 HOMBRE  
## 12496 HOMBRE  
## 12497 HOMBRE  
## 12498 HOMBRE  
## 12499 HOMBRE  
## 12500 MUJER

Revisando los valores estadísticos de las variables para conocer de forma breve características básicas de los datos con los que estamos trabajando.

summary(df\_segurohistorica)

## COD\_CLIENTE CAT\_EDAD ESTADO\_CIVIL   
## CLI0025785429: 3 1\_NULL : 1651 CASADO : 4851   
## R\_10418810039: 3 2\_MINUS30:11600 SOLTERO:32649   
## CLI0002601704: 2 3\_30\_40 :10036   
## CLI0004812180: 2 4\_PLUS40 :14213   
## CLI0006355407: 2   
## CLI0006662190: 2   
## (Other) :37486   
## NIVEL\_ESTUDIOS RANGO\_INGRESOS CAMP\_DEPOSITOS SEXO   
## PRIMARIA : 5710 [1000-2000>: 8150 0:25763 HOMBRE:18689   
## SECUNDARIA :21283 [600-1000> :18781 1:11737 MUJER :18811   
## SIN\_ESTUDIOS: 3057 <600 : 5793   
## UNIVERSIDAD : 7450 >=2000 : 4776   
##   
##   
##

summary(df\_seguronueva)

## COD\_CLIENTE CAT\_EDAD ESTADO\_CIVIL   
## CLI0007476710: 2 1\_NULL : 526 CASADO : 1630   
## CLI0009053069: 2 2\_MINUS30:3832 SOLTERO:10870   
## CLI0009696402: 2 3\_30\_40 :3378   
## CLI0016690510: 2 4\_PLUS40 :4764   
## CLI0023165847: 2   
## CLI0042986753: 2   
## (Other) :12488   
## NIVEL\_ESTUDIOS RANGO\_INGRESOS SEXO   
## PRIMARIA :1951 [1000-2000>:2740 HOMBRE:6156   
## SECUNDARIA :7116 [600-1000> :6278 MUJER :6344   
## SIN\_ESTUDIOS: 968 <600 :1903   
## UNIVERSIDAD :2465 >=2000 :1579   
##   
##   
##

### Bloque de creación de conjuntos de entrenamiento y test

Por este motivo dividimos nuestro conjunto de datos en tres conjuntos: entrenamiento(70%), validación(15%) y test(15%). Para que todo el proceso sea reproducible se ha fijado una semilla mediante el comando set.seed.

Primero se divide el conjunto total en dos conjuntos:

set.seed(1234)   
SAMPLE = sample.split(df\_segurohistorica$CAMP\_DEPOSITOS, SplitRatio = .70)  
df\_segurohistoricaTrain = subset(df\_segurohistorica, SAMPLE == TRUE)  
df\_segurohistoricaValTest = subset(df\_segurohistorica, SAMPLE == FALSE)  
  
SAMPLE = sample.split(df\_segurohistoricaValTest$CAMP\_DEPOSITOS, SplitRatio = .50)  
df\_segurohistoricaVal= subset(df\_segurohistoricaValTest, SAMPLE == TRUE)  
df\_segurohistoricaTest = subset(df\_segurohistoricaValTest, SAMPLE == FALSE)

De forma que tenemos 3 conjuntos df\_segurohistoricaTrain, df\_segurohistoricaVal y df\_segurohistoricaTest con 26’250, 5’624 y 5’624 registros respectivamente.

dim(df\_segurohistorica)

## [1] 37500 7

dim(df\_segurohistoricaTrain)

## [1] 26250 7

dim(df\_segurohistoricaVal)

## [1] 5624 7

dim(df\_segurohistoricaTest)

## [1] 5626 7

El uso del comando sample.split de la librería CaTools nos permite mantener el porcentaje de éxitos por conjunto en el 31.29% aproximadamente como podemos comprobar:

table(df\_segurohistorica$CAMP\_DEPOSITOS)

##   
## 0 1   
## 25763 11737

sum(df\_segurohistorica$CAMP\_DEPOSITOS==1)/length(df\_segurohistorica$CAMP\_DEPOSITOS)

## [1] 0.3129867

table(df\_segurohistoricaTrain$CAMP\_DEPOSITOS)

##   
## 0 1   
## 18034 8216

prior=sum(df\_segurohistoricaTrain$CAMP\_DEPOSITOS==1)/length(df\_segurohistoricaTrain$CAMP\_DEPOSITOS)  
prior

## [1] 0.3129905

table(df\_segurohistoricaVal$CAMP\_DEPOSITOS)

##   
## 0 1   
## 3864 1760

sum(df\_segurohistoricaVal$CAMP\_DEPOSITOS==1)/length(df\_segurohistoricaVal$CAMP\_DEPOSITOS)

## [1] 0.3129445

table(df\_segurohistoricaTest$CAMP\_DEPOSITOS)

##   
## 0 1   
## 3865 1761

sum(df\_segurohistoricaTest$CAMP\_DEPOSITOS==1)/length(df\_segurohistoricaTest$CAMP\_DEPOSITOS)

## [1] 0.313011

### Bloque de análisis del poder predictivo de las variables

En este bloque vamos a analizar la capacidad predictiva individual univariable de cada variable. Esto nos permite conocer qué factores son los que afectan en la contratación del producto.

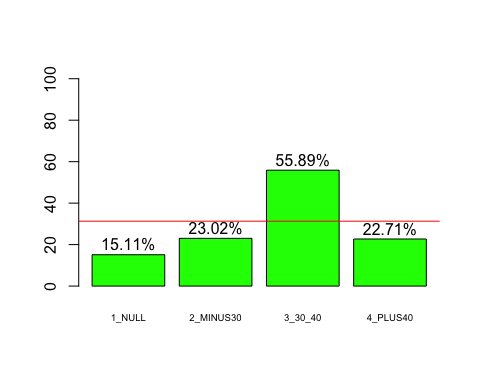
Comenzamos mediante un gráfico que nos permite comparar el porcentaje de éxito de la campaña para las diferentes categorías de una variable (barras) y compararlas con el prior del dataset (línea roja). Para ello creamos la siguiente función:

relevancia=function(Target,VariableCategorica){  
 levels=levels(VariableCategorica)  
 colors=c()  
 for (i in 1:length(levels)){  
 TABLA=table(Target,VariableCategorica==levels[i])  
 chi=chisq.test(TABLA)  
 if (chi$p.value<0.05){  
 colors=c(colors,"green")  
 }else{  
 colors=c(colors,"gray")  
 }  
 }  
 TABLA=table(Target,VariableCategorica)  
 plot=barplot(100\*TABLA[2,]/(TABLA[1,]+TABLA[2,]),ylim=c(0,100),col=colors,cex.names=0.6)  
 text(x=plot, y=5+100\*TABLA[2,]/(TABLA[1,]+TABLA[2,]),labels=paste(round(100\*TABLA[2,]/(TABLA[1,]+TABLA[2,]),2),"%",sep=""))  
 abline(h=100\*prior,col="red")  
}

woe\_iv=function(Target,VariableCategorica){  
 Tabla\_WOE=table(VariableCategorica,Target)  
 DF\_WOE=data.frame(FRACASOS=Tabla\_WOE[,1],EXITOS=Tabla\_WOE[,2])  
 DF\_WOE$EXITOS\_PORC=DF\_WOE$EXITOS/sum(DF\_WOE$EXITOS)  
 DF\_WOE$FRACASOS\_PORC=DF\_WOE$FRACASOS/sum(DF\_WOE$FRACASOS)  
 DF\_WOE$WOE=log(DF\_WOE$EXITOS\_PORC/DF\_WOE$FRACASOS\_PORC)  
 DF\_WOE$IV=(DF\_WOE$EXITOS\_PORC-DF\_WOE$FRACASOS\_PORC)\*DF\_WOE$WOE  
 DF\_WOE  
}

Relevancia por Edad

relevancia(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$CAT\_EDAD)



WOE\_CAT\_EDAD=woe\_iv(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$CAT\_EDAD)  
WOE\_CAT\_EDAD

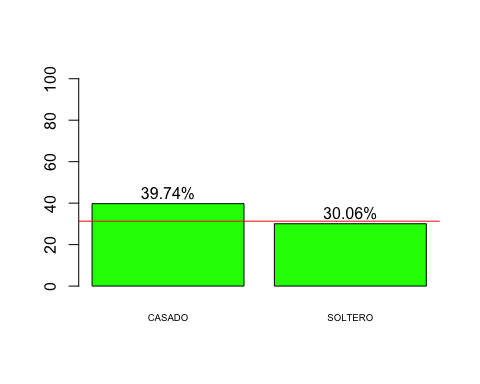
## FRACASOS EXITOS EXITOS\_PORC FRACASOS\_PORC WOE IV  
## 1\_NULL 972 173 0.02105648 0.05389819 -0.9398888 0.03086756  
## 2\_MINUS30 6283 1879 0.22870010 0.34839747 -0.4209325 0.05038451  
## 3\_30\_40 3080 3902 0.47492697 0.17078851 1.0227350 0.31105306  
## 4\_PLUS40 7699 2262 0.27531646 0.42691583 -0.4386657 0.06650144

IV\_CAT\_EDAD=sum(WOE\_CAT\_EDAD$IV)  
IV\_CAT\_EDAD

## [1] 0.4588066

Relevancia por Estado civil

relevancia(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$ESTADO\_CIVIL)



WOE\_ESTADO\_CIVIL=woe\_iv(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$ESTADO\_CIVIL)  
WOE\_ESTADO\_CIVIL

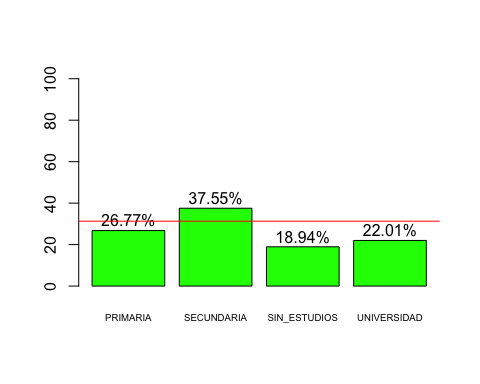
## FRACASOS EXITOS EXITOS\_PORC FRACASOS\_PORC WOE IV  
## CASADO 2029 1338 0.162853 0.1125097 0.36980829 0.018617357  
## SOLTERO 16005 6878 0.837147 0.8874903 -0.05839787 0.002939939

IV\_ESTADO\_CIVIL=sum(WOE\_ESTADO\_CIVIL$IV)  
IV\_ESTADO\_CIVIL

## [1] 0.0215573

Relevancia por NIVEL ESTUDIOS

relevancia(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$NIVEL\_ESTUDIOS)



WOE\_NIVEL\_ESTUDIOS=woe\_iv(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$NIVEL\_ESTUDIO)  
WOE\_NIVEL\_ESTUDIOS

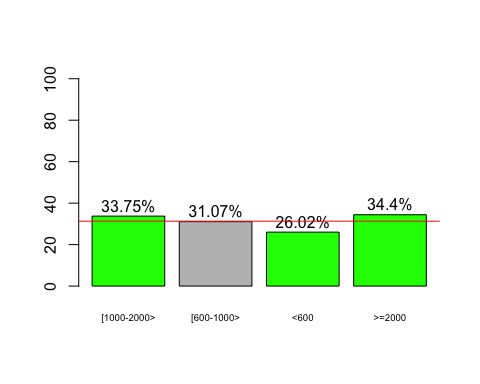
## FRACASOS EXITOS EXITOS\_PORC FRACASOS\_PORC WOE  
## PRIMARIA 2908 1063 0.12938169 0.16125097 -0.2201951  
## SECUNDARIA 9304 5594 0.68086660 0.51591438 0.2774256  
## SIN\_ESTUDIOS 1733 405 0.04929406 0.09609626 -0.6675468  
## UNIVERSIDAD 4089 1154 0.14045764 0.22673838 -0.4788909  
## IV  
## PRIMARIA 0.007017457  
## SECUNDARIA 0.045761963  
## SIN\_ESTUDIOS 0.031242662  
## UNIVERSIDAD 0.041319059

IV\_NIVEL\_ESTUDIOS=sum(WOE\_NIVEL\_ESTUDIOS$IV)  
IV\_NIVEL\_ESTUDIOS

## [1] 0.1253411

Relevancia por RANGO INGRESOS

relevancia(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$RANGO\_INGRESOS)



WOE\_RANGO\_INGRESOS=woe\_iv(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$RANGO\_INGRESOS)  
WOE\_RANGO\_INGRESOS

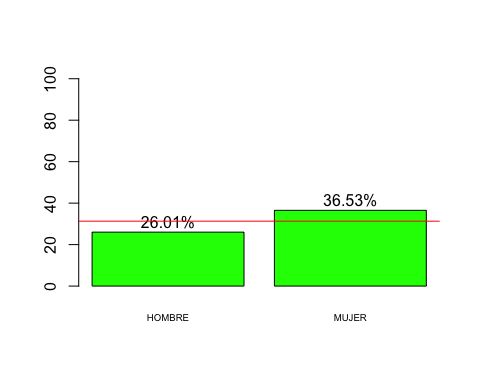
## FRACASOS EXITOS EXITOS\_PORC FRACASOS\_PORC WOE  
## [1000-2000> 3801 1936 0.2356378 0.2107685 0.11153519  
## [600-1000> 9002 4058 0.4939143 0.4991682 -0.01058114  
## <600 3017 1061 0.1291383 0.1672951 -0.25887571  
## >=2000 2214 1161 0.1413096 0.1227681 0.14065626  
## IV  
## [1000-2000> 2.773795e-03  
## [600-1000> 5.559251e-05  
## <600 9.877880e-03  
## >=2000 2.607983e-03

IV\_RANGO\_INGRESOS=sum(WOE\_RANGO\_INGRESOS$IV)  
IV\_RANGO\_INGRESOS

## [1] 0.01531525

Relevancia por SEXO

relevancia(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$SEXO)



WOE\_SEXO=woe\_iv(df\_segurohistoricaTrain$CAMP\_DEPOSITOS,df\_segurohistoricaTrain$SEXO)  
WOE\_SEXO

## FRACASOS EXITOS EXITOS\_PORC FRACASOS\_PORC WOE IV  
## HOMBRE 9662 3397 0.4134615 0.5357658 -0.2591326 0.03169301  
## MUJER 8372 4819 0.5865385 0.4642342 0.2338490 0.02860073

IV\_SEXO=sum(WOE\_SEXO$IV)  
IV\_SEXO

## [1] 0.06029374

Podemos establecer un ranking de capacidad predictiva ordenando las variables en función a su IV:

IV\_CAT\_EDAD

## [1] 0.4588066

IV\_NIVEL\_ESTUDIOS

## [1] 0.1253411

IV\_SEXO

## [1] 0.06029374

IV\_ESTADO\_CIVIL

## [1] 0.0215573

IV\_RANGO\_INGRESOS

## [1] 0.01531525

### Bloque de Construcción de Modelos

Comenzamos con el modelo más sencillo que incluye una sola variable independiente. Hemos seleccionado la variable CAT\_EDAD al ser la de mayor IV.

### Modelo Regresion Logistica:

print("Modelo 1")

## [1] "Modelo 1"

modelo1=glm(CAMP\_DEPOSITOS~CAT\_EDAD, data=df\_segurohistoricaTrain[,-1],family=binomial(link="logit"))  
summary(modelo1)

##   
## Call:  
## glm(formula = CAMP\_DEPOSITOS ~ CAT\_EDAD, family = binomial(link = "logit"),   
## data = df\_segurohistoricaTrain[, -1])  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.2794 -0.7234 -0.7178 1.0788 1.9442   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.72606 0.08252 -20.918 < 2e-16 \*\*\*  
## CAT\_EDAD2\_MINUS30 0.51896 0.08661 5.992 2.07e-09 \*\*\*  
## CAT\_EDAD3\_30\_40 1.96262 0.08597 22.830 < 2e-16 \*\*\*  
## CAT\_EDAD4\_PLUS40 0.50122 0.08591 5.834 5.41e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 32627 on 26249 degrees of freedom  
## Residual deviance: 30035 on 26246 degrees of freedom  
## AIC: 30043  
##   
## Number of Fisher Scoring iterations: 4

print("Modelo 2")

## [1] "Modelo 2"

modelo2=glm(CAMP\_DEPOSITOS~CAT\_EDAD+NIVEL\_ESTUDIOS, data=df\_segurohistoricaTrain[,-1],family=binomial(link="logit"))  
summary(modelo2)

##   
## Call:  
## glm(formula = CAMP\_DEPOSITOS ~ CAT\_EDAD + NIVEL\_ESTUDIOS, family = binomial(link = "logit"),   
## data = df\_segurohistoricaTrain[, -1])  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4338 -0.7694 -0.5960 0.9411 2.0891   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.80336 0.11999 -15.029 < 2e-16 \*\*\*  
## CAT\_EDAD2\_MINUS30 0.16515 0.11498 1.436 0.150896   
## CAT\_EDAD3\_30\_40 1.81591 0.11629 15.615 < 2e-16 \*\*\*  
## CAT\_EDAD4\_PLUS40 0.38139 0.11569 3.297 0.000979 \*\*\*  
## NIVEL\_ESTUDIOSSECUNDARIA 0.57254 0.04212 13.592 < 2e-16 \*\*\*  
## NIVEL\_ESTUDIOSSIN\_ESTUDIOS 0.07729 0.08711 0.887 0.374920   
## NIVEL\_ESTUDIOSUNIVERSIDAD -0.42430 0.05233 -8.109 5.11e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 32627 on 26249 degrees of freedom  
## Residual deviance: 29313 on 26243 degrees of freedom  
## AIC: 29327  
##   
## Number of Fisher Scoring iterations: 4

print("Modelo 3")

## [1] "Modelo 3"

modelo3=glm(CAMP\_DEPOSITOS~CAT\_EDAD+NIVEL\_ESTUDIOS+SEXO, data=df\_segurohistoricaTrain[,-1],family=binomial(link="logit"))  
summary(modelo3)

##   
## Call:  
## glm(formula = CAMP\_DEPOSITOS ~ CAT\_EDAD + NIVEL\_ESTUDIOS + SEXO,   
## family = binomial(link = "logit"), data = df\_segurohistoricaTrain[,   
## -1])  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.5601 -0.8563 -0.6081 1.0481 2.2201   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.10072 0.12198 -17.222 < 2e-16 \*\*\*  
## CAT\_EDAD2\_MINUS30 0.15983 0.11572 1.381 0.16724   
## CAT\_EDAD3\_30\_40 1.84013 0.11711 15.712 < 2e-16 \*\*\*  
## CAT\_EDAD4\_PLUS40 0.38746 0.11648 3.326 0.00088 \*\*\*  
## NIVEL\_ESTUDIOSSECUNDARIA 0.57262 0.04241 13.502 < 2e-16 \*\*\*  
## NIVEL\_ESTUDIOSSIN\_ESTUDIOS 0.06942 0.08783 0.790 0.42933   
## NIVEL\_ESTUDIOSUNIVERSIDAD -0.43461 0.05270 -8.247 < 2e-16 \*\*\*  
## SEXOMUJER 0.55377 0.02898 19.111 < 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: 32627 on 26249 degrees of freedom  
## Residual deviance: 28942 on 26242 degrees of freedom  
## AIC: 28958  
##   
## Number of Fisher Scoring iterations: 4

print("Modelo 4")

## [1] "Modelo 4"

modelo4=glm(CAMP\_DEPOSITOS~CAT\_EDAD+NIVEL\_ESTUDIOS+SEXO+ESTADO\_CIVIL, data=df\_segurohistoricaTrain[,-1],family=binomial(link="logit"))  
summary(modelo4)

##   
## Call:  
## glm(formula = CAMP\_DEPOSITOS ~ CAT\_EDAD + NIVEL\_ESTUDIOS + SEXO +   
## ESTADO\_CIVIL, family = binomial(link = "logit"), data = df\_segurohistoricaTrain[,   
## -1])  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.8994 -0.8504 -0.6419 1.0246 2.2599   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.31099 0.12984 -10.097 <2e-16 \*\*\*  
## CAT\_EDAD2\_MINUS30 0.20442 0.11587 1.764 0.0777 .   
## CAT\_EDAD3\_30\_40 1.84575 0.11737 15.726 <2e-16 \*\*\*  
## CAT\_EDAD4\_PLUS40 0.21917 0.11733 1.868 0.0618 .   
## NIVEL\_ESTUDIOSSECUNDARIA 0.52905 0.04269 12.393 <2e-16 \*\*\*  
## NIVEL\_ESTUDIOSSIN\_ESTUDIOS 0.08940 0.08820 1.014 0.3108   
## NIVEL\_ESTUDIOSUNIVERSIDAD -0.55240 0.05363 -10.301 <2e-16 \*\*\*  
## SEXOMUJER 0.56005 0.02918 19.196 <2e-16 \*\*\*  
## ESTADO\_CIVILSOLTERO -0.81348 0.04556 -17.853 <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: 32627 on 26249 degrees of freedom  
## Residual deviance: 28624 on 26241 degrees of freedom  
## AIC: 28642  
##   
## Number of Fisher Scoring iterations: 4

print("Modelo 5")

## [1] "Modelo 5"

modelo5=glm(CAMP\_DEPOSITOS~., data=df\_segurohistoricaTrain[,-1],family=binomial(link="logit"))  
summary(modelo5)

##   
## Call:  
## glm(formula = CAMP\_DEPOSITOS ~ ., family = binomial(link = "logit"),   
## data = df\_segurohistoricaTrain[, -1])  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9100 -0.8487 -0.6409 1.0245 2.2626   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.33667 0.13425 -9.957 <2e-16 \*\*\*  
## CAT\_EDAD2\_MINUS30 0.20508 0.11588 1.770 0.0768 .   
## CAT\_EDAD3\_30\_40 1.84951 0.11775 15.707 <2e-16 \*\*\*  
## CAT\_EDAD4\_PLUS40 0.22298 0.11759 1.896 0.0579 .   
## ESTADO\_CIVILSOLTERO -0.81232 0.04579 -17.740 <2e-16 \*\*\*  
## NIVEL\_ESTUDIOSSECUNDARIA 0.53025 0.04316 12.285 <2e-16 \*\*\*  
## NIVEL\_ESTUDIOSSIN\_ESTUDIOS 0.08884 0.08820 1.007 0.3138   
## NIVEL\_ESTUDIOSUNIVERSIDAD -0.55328 0.05520 -10.024 <2e-16 \*\*\*  
## RANGO\_INGRESOS[600-1000> 0.02157 0.03787 0.570 0.5689   
## RANGO\_INGRESOS<600 0.03233 0.05181 0.624 0.5327   
## RANGO\_INGRESOS>=2000 0.04502 0.05075 0.887 0.3750   
## SEXOMUJER 0.56000 0.02918 19.192 <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: 32627 on 26249 degrees of freedom  
## Residual deviance: 28623 on 26238 degrees of freedom  
## AIC: 28647  
##   
## Number of Fisher Scoring iterations: 4

Podemos comprobar en el último modelo que para la variable sexo, ambas categorías tienen el mismo comportamiento por lo que no mejora la capacidad predictiva del modelo y no debería incluirse.

De hecho si calculamos los AIC y BIC de todos los modelos podemos comprobar que el modelo4 es el que tiene los valores más bajos para ambas métricas por lo que es el que se ajusta mejor a los datos.

AIC(modelo1)

## [1] 30042.62

AIC(modelo2)

## [1] 29326.93

AIC(modelo3)

## [1] 28957.92

AIC(modelo4)

## [1] 28641.67

AIC(modelo5)

## [1] 28646.76

BIC(modelo1)

## [1] 30075.32

BIC(modelo2)

## [1] 29384.16

BIC(modelo3)

## [1] 29023.32

BIC(modelo4)

## [1] 28715.25

BIC(modelo5)

## [1] 28744.87

Las métricas AIC y BIC equilibran la ganancia de información al meter una nueva variable con la pérdida por complicar el modelo de forma que previenen la inclusión de factores que no aporten capacidad predictiva.

### Modelo Random Forest:

Calculando modelos, en base a lo trabajado anteriormente, ajustamos en base a las variables del modelo 4 y modelo 5

#print("Modelo 1")  
#modeloRF1=randomForest(CAMP\_DEPOSITOS~CAT\_EDAD, data=df\_segurohistoricaTrain[,-1],importance = TRUE, maxnodes=10,mtry=1,ntree=10)  
#summary(modeloRF1)  
  
#print("Modelo 2")  
#modeloRF2=randomForest(CAMP\_DEPOSITOS~CAT\_EDAD+NIVEL\_ESTUDIOS, data=df\_segurohistoricaTrain[,-1],importance = TRUE, maxnodes=10,mtry=2,ntree=10)  
#summary(modeloRF2)  
  
#print("Modelo 3")  
#modeloRF3=randomForest(CAMP\_DEPOSITOS~CAT\_EDAD+NIVEL\_ESTUDIOS+SEXO, data=df\_segurohistoricaTrain[,-1],importance = TRUE, maxnodes=10,mtry=3,ntree=10)  
#summary(modeloRF3)  
  
print("Modelo 4")

## [1] "Modelo 4"

modeloRF4=randomForest(CAMP\_DEPOSITOS~CAT\_EDAD+NIVEL\_ESTUDIOS+SEXO+ESTADO\_CIVIL, data=df\_segurohistoricaTrain[,-1],importance = TRUE, maxnodes=60,mtry=4,ntree=70)  
summary(modeloRF4)

## Length Class Mode   
## call 7 -none- call   
## type 1 -none- character  
## predicted 26250 factor numeric   
## err.rate 210 -none- numeric   
## confusion 6 -none- numeric   
## votes 52500 matrix numeric   
## oob.times 26250 -none- numeric   
## classes 2 -none- character  
## importance 16 -none- numeric   
## importanceSD 12 -none- numeric   
## localImportance 0 -none- NULL   
## proximity 0 -none- NULL   
## ntree 1 -none- numeric   
## mtry 1 -none- numeric   
## forest 14 -none- list   
## y 26250 factor numeric   
## test 0 -none- NULL   
## inbag 0 -none- NULL   
## terms 3 terms call

print("Modelo 5")

## [1] "Modelo 5"

modeloRF5=randomForest(CAMP\_DEPOSITOS~., data=df\_segurohistoricaTrain[,-1],importance = TRUE, maxnodes=50,mtry=5,ntree=50)  
summary(modeloRF5)

## Length Class Mode   
## call 7 -none- call   
## type 1 -none- character  
## predicted 26250 factor numeric   
## err.rate 150 -none- numeric   
## confusion 6 -none- numeric   
## votes 52500 matrix numeric   
## oob.times 26250 -none- numeric   
## classes 2 -none- character  
## importance 20 -none- numeric   
## importanceSD 15 -none- numeric   
## localImportance 0 -none- NULL   
## proximity 0 -none- NULL   
## ntree 1 -none- numeric   
## mtry 1 -none- numeric   
## forest 14 -none- list   
## y 26250 factor numeric   
## test 0 -none- NULL   
## inbag 0 -none- NULL   
## terms 3 terms call

### Bloque de evaluación y selección de modelo.

### Validando su Accuracy del modelo Random Forest

accuracies <-c()  
  
#prediccion=predict(modeloRF1,df\_segurohistoricaTrain)  
#t<-table(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS)  
#print(t)  
#df\_segurohistoricaTrain$Rf1 <- prediccion == df\_segurohistoricaTrain$CAMP\_DEPOSITOS  
#accuracy <- sum(as.numeric(df\_segurohistoricaTrain$Rf1))/nrow(df\_segurohistoricaTrain)  
#accuracies <- c(accuracies,accuracy)  
#print(accuracy)  
  
#accuracies <-c()  
#prediccion=predict(modeloRF2, df\_segurohistoricaTrain)  
#t<-table(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS)  
#print(t)  
#df\_segurohistoricaTrain$Rf2 <- prediccion == df\_segurohistoricaTrain$CAMP\_DEPOSITOS  
#accuracy <- sum(as.numeric(df\_segurohistoricaTrain$Rf2))/nrow(df\_segurohistoricaTrain)  
#accuracies <- c(accuracies,accuracy)  
#print(accuracy)  
  
#prediccion=predict(modeloRF3,df\_segurohistoricaTrain)  
#t<-table(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS)  
#print(t)  
#df\_segurohistoricaTrain$Rf3 <- prediccion == df\_segurohistoricaTrain$CAMP\_DEPOSITOS  
#accuracy <- sum(as.numeric(df\_segurohistoricaTrain$Rf3))/nrow(df\_segurohistoricaTrain)  
#accuracies <- c(accuracies,accuracy)  
#print(accuracy)  
  
prediccion=predict(modeloRF4,df\_segurohistoricaTrain)  
t<-table(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS)  
print(t)

##   
## prediccion 0 1  
## 0 15861 4793  
## 1 2173 3423

df\_segurohistoricaTrain$Rf4 <- prediccion == df\_segurohistoricaTrain$CAMP\_DEPOSITOS  
accuracy <- sum(as.numeric(df\_segurohistoricaTrain$Rf4))/nrow(df\_segurohistoricaTrain)  
accuracies <- c(accuracies,accuracy)  
print(accuracy)

## [1] 0.7346286

prediccion=predict(modeloRF5,df\_segurohistoricaTrain)  
t<-table(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS)  
print(t)

##   
## prediccion 0 1  
## 0 15963 4850  
## 1 2071 3366

df\_segurohistoricaTrain$Rf5 <- prediccion == df\_segurohistoricaTrain$CAMP\_DEPOSITOS  
accuracy <- sum(as.numeric(df\_segurohistoricaTrain$Rf5))/nrow(df\_segurohistoricaTrain)  
accuracies <- c(accuracies,accuracy)  
print(accuracy)

## [1] 0.7363429

### Validando su Accuracy del modelo regression logistica

Para evaluar y seleccionar un modelo vamos a utilizar una métrica ampliamente aceptada en los problemas de clasificación se utilice la técnica que se utilice: AUC (Area Under Curve) o área bajo la curva.

Para el primer modelo vemos que el AUC en entrenamiento 0,7020 es similar al AUC en validación 0,6986. Esto es importante puesto que nos muestra que el modelo tiene la misma capacidad predictiva en el conjunto en el que se ha entrenado que en otro conjunto del cual no se ha entrenado, confirmando que los patrones aprendidos por el modelo son generales, por lo que no se observa sobreajuste u overfitting.

prediccion=predict(modelo1,type="response")  
Pred\_auxiliar= prediction(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo1\_train = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo1, newdata=df\_segurohistoricaVal,type="response")  
Pred\_auxiliar = prediction(prediccion, df\_segurohistoricaVal$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo1\_val = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo2,type="response")  
Pred\_auxiliar= prediction(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo2\_train = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo2, newdata=df\_segurohistoricaVal,type="response")  
Pred\_auxiliar = prediction(prediccion, df\_segurohistoricaVal$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo2\_val = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo3,type="response")  
Pred\_auxiliar= prediction(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo3\_train = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo3, newdata=df\_segurohistoricaVal,type="response")  
Pred\_auxiliar = prediction(prediccion, df\_segurohistoricaVal$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo3\_val = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo4,type="response")  
Pred\_auxiliar= prediction(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo4\_train = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo4, newdata=df\_segurohistoricaVal,type="response")  
Pred\_auxiliar = prediction(prediccion, df\_segurohistoricaVal$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo4\_val = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo5,type="response")  
Pred\_auxiliar= prediction(prediccion, df\_segurohistoricaTrain$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo5\_train = as.numeric(auc.tmp@y.values)  
  
prediccion=predict(modelo5, newdata=df\_segurohistoricaVal,type="response")  
Pred\_auxiliar = prediction(prediccion, df\_segurohistoricaVal$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo5\_val = as.numeric(auc.tmp@y.values)

Representamos los resultados:

## Modelo1 Modelo2 Modelo3 Modelo4 Modelo5  
## auc\_train 0.6583474 0.7069584 0.7218755 0.7296157 0.7300907  
## auc\_val 0.6666602 0.7165237 0.7270413 0.7390733 0.7385212

Podemos apreciar que el modelo con mayor AUC en validación es el modelo 4, por lo que este es el modelo elegido puesto que presenta los mejores resultados en el conjunto de validación.

Una vez elegido el modelo4 sólo nos queda evaluar su capacidad y esto se realiza utilizando el conjunto de test que no ha sido utilizado en ninguna parte del proceso de construcción y selección del modelo.

df\_segurohistoricaTest$prediccion=predict(modelo4, newdata=df\_segurohistoricaTest,type="response")  
Pred\_auxiliar = prediction(df\_segurohistoricaTest$prediccion, df\_segurohistoricaTest$CAMP\_DEPOSITOS, label.ordering = NULL)  
auc.tmp = performance(Pred\_auxiliar, "auc");  
auc\_modelo4\_test = as.numeric(auc.tmp@y.values)  
auc\_modelo4\_test

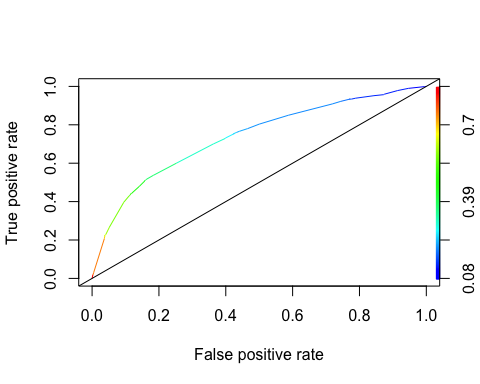
## [1] 0.7357789

head(df\_segurohistoricaTest)

## COD\_CLIENTE CAT\_EDAD ESTADO\_CIVIL NIVEL\_ESTUDIOS RANGO\_INGRESOS  
## 5 CLI0010350442 4\_PLUS40 CASADO UNIVERSIDAD >=2000  
## 10 CLI0009241743 4\_PLUS40 SOLTERO UNIVERSIDAD [1000-2000>  
## 22 CLI0032925889 4\_PLUS40 CASADO UNIVERSIDAD [600-1000>  
## 23 CLI0009004617 4\_PLUS40 SOLTERO SECUNDARIA [600-1000>  
## 36 CLI0080200957 3\_30\_40 SOLTERO SECUNDARIA [600-1000>  
## 45 CLI0042895322 2\_MINUS30 SOLTERO SECUNDARIA [1000-2000>  
## CAMP\_DEPOSITOS SEXO prediccion  
## 5 0 HOMBRE 0.16189196  
## 10 1 HOMBRE 0.07887758  
## 22 1 HOMBRE 0.16189196  
## 23 0 HOMBRE 0.20161330  
## 36 1 HOMBRE 0.56226100  
## 45 0 MUJER 0.30344603

En este caso el modelo resultante tiene un AUC de 0,7357. Gráficamente se puede representar de la siguiente manera para el conjunto de test:

CURVA\_ROC\_modelo4\_train <- performance(Pred\_auxiliar,"tpr","fpr")  
plot(CURVA\_ROC\_modelo4\_train,colorize=TRUE)  
abline(a=0,b=1,col="black")



Otra métrica habitual para representar la capacidad predictiva de un modelo es el Índice de Gini que se puede obtener fácilmente del AUC:

GINI\_train=2\*auc\_modelo4\_train-1  
GINI\_train

## [1] 0.4592314

GINI\_test=2\*auc\_modelo4\_test-1  
GINI\_test

## [1] 0.4715579

Como el Índice de Gini es una combinación lineal positiva del AUC, se podría haber realizado la selección de modelos utilizando el Índice de Gini en lugar del AUC.

Para terminar con la valoración del modelo, podemos mostrar la capacidad del modelo de la siguiente manera:

mean(as.numeric(df\_segurohistoricaTest$CAMP\_DEPOSITOS)-1)

## [1] 0.313011

aggregate(df\_segurohistoricaTest$prediccion~df\_segurohistoricaTest$CAMP\_DEPOSITOS,FUN=mean)

## df\_segurohistoricaTest$CAMP\_DEPOSITOS df\_segurohistoricaTest$prediccion  
## 1 0 0.2641764  
## 2 1 0.4248821

Como podemos apreciar, el éxito medio de la campaña es un 31,29%. Nuestro modelo está asignando una probabilidad media del 42,48% a aquellos que efectivamente fueron éxito de la campaña y un 26,41% a aquellos que no fueron éxito, por lo que el modelo está discriminando.

### Bloque de explotación del modelo

El modelo predictivo elegido nos asigna a cada cliente una probabilidad de éxito de la campaña, es decir, los ordena los clientes por su probabilidad de éxito de la campaña.

ALPHA=0.4248821  
Confusion\_Test=table(df\_segurohistoricaTest$CAMP\_DEPOSITOS,df\_segurohistoricaTest$prediccion>=ALPHA)  
Confusion\_Test

##   
## FALSE TRUE  
## 0 3253 612  
## 1 865 896

En este caso seleccionaríamos 1’508 clientes (2ª columna 612+896), de los que 896 serían éxito. Las métricas asociadas a esta matriz de confusión serían:

Accuracy\_Test= (sum(df\_segurohistoricaTest$CAMP\_DEPOSITOS==1 & df\_segurohistoricaTest$prediccion>=ALPHA)+sum(df\_segurohistoricaTest$CAMP\_DEPOSITOS==0 & df\_segurohistoricaTest$prediccion<ALPHA))/length(df\_segurohistoricaTest$CAMP\_DEPOSITOS)  
Precision\_Test=sum(df\_segurohistoricaTest$CAMP\_DEPOSITOS==1 & df\_segurohistoricaTest$prediccion>=ALPHA)/sum(df\_segurohistoricaTest$prediccion>=ALPHA)  
Cobertura\_Test=sum(df\_segurohistoricaTest$CAMP\_DEPOSITOS==1 & df\_segurohistoricaTest$prediccion>=ALPHA)/sum(df\_segurohistoricaTest$CAMP\_DEPOSITOS==1)  
Accuracy\_Test

## [1] 0.7374689

Precision\_Test

## [1] 0.5941645

Cobertura\_Test

## [1] 0.5088018

Podemos ver que el acierto sería del 74,60% (acierto), pero más importante es que del conjunto seleccionado vamos a acertar en un 65,57% (precisión) y estamos llamando al 39,69% de los éxitos (cobertura).

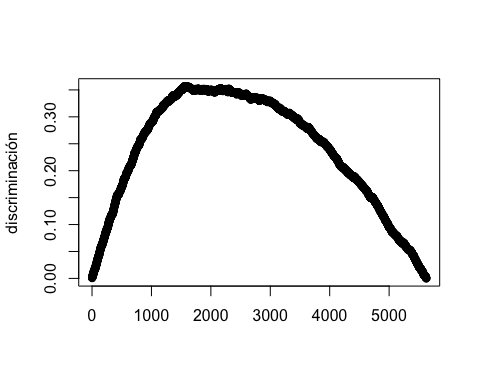
Si modificamos el umbral a otros valores conseguimos seleccionar distintas cantidades de clientes con lo que podemos modificar los parámetros de éxito para adecuarlos a nuestras necesidades.

### Bloque de selección de umbral como el punto de máxima discriminación

Esta técnica calcula el umbral en el que existe la máxima discriminación, la cuantificación de la discriminación es una métrica que recibe el nombre KS y también es utilizada como métrica de capacidad predictiva de un modelo principalmente en modelos de scoring.

Calculamos el punto de máxima discriminación:

BANK\_KS=df\_segurohistoricaTest[order(df\_segurohistoricaTest$prediccion, decreasing=TRUE),c("CAMP\_DEPOSITOS","prediccion")]  
BANK\_KS$N=1:length(BANK\_KS$CAMP\_DEPOSITOS)  
BANK\_KS$EXITOS\_ACUM=cumsum(as.numeric(BANK\_KS$CAMP\_DEPOSITOS)-1)  
BANK\_KS$FRACASOS\_ACUM=BANK\_KS$N-BANK\_KS$EXITOS\_ACUM  
BANK\_KS$EXITOS\_TOT=sum(BANK\_KS$CAMP\_DEPOSITOS==1)  
BANK\_KS$FRACASOS\_TOT=sum(BANK\_KS$CAMP\_DEPOSITOS==0)  
BANK\_KS$TOTAL=BANK\_KS$EXITOS\_TOT+BANK\_KS$FRACASOS\_TOT  
BANK\_KS$TPR=BANK\_KS$EXITOS\_ACUM/BANK\_KS$EXITOS\_TOT  
BANK\_KS$FPR=BANK\_KS$FRACASOS\_ACUM/BANK\_KS$FRACASOS\_TOT  
BANK\_KS$DIFF=BANK\_KS$TPR-BANK\_KS$FPR  
plot(BANK\_KS$DIFF, xlab="",ylab="discriminación")



el valor de la máxima discriminación se llama KS

KS=max(BANK\_KS$DIFF)  
KS

## [1] 0.3565192

y el valor donde se alcanza este valor es el que nos va a permitir seleccionar el umbral

which(BANK\_KS$DIFF==KS)

## [1] 1590

BANK\_KS[1590,c("CAMP\_DEPOSITOS","prediccion")]

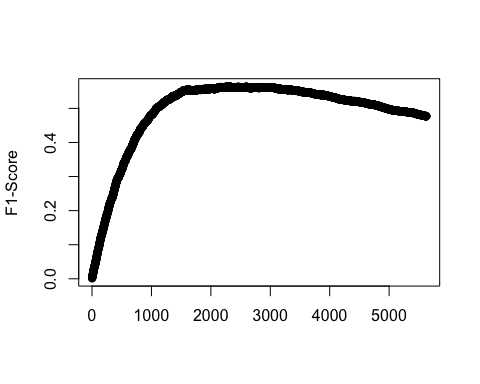
## CAMP\_DEPOSITOS prediccion  
## 16170 1 0.3629083

en este caso el umbral seleccionado de máxima discriminación sería 0.3629083

### Bloque de selección de umbral como el punto óptimo estadístico (F1-Score)

Esta técnica selecciona el umbral como el punto que optimiza la relación entre la precisión y la cobertura maximizando el F1-Score o F-Score.

BANK\_KS$Accuracy=(BANK\_KS$EXITOS\_ACUM+BANK\_KS$FRACASOS\_TOT-BANK\_KS$FRACASOS\_ACUM)/BANK\_KS$TOTAL  
BANK\_KS$Precision=BANK\_KS$EXITOS\_ACUM/BANK\_KS$N  
BANK\_KS$Cobertura=BANK\_KS$EXITOS\_ACUM/BANK\_KS$EXITOS\_TOT  
BANK\_KS$F1Score=2\*(BANK\_KS$Precision\*BANK\_KS$Cobertura)/(BANK\_KS$Precision+BANK\_KS$Cobertura)  
plot(BANK\_KS$F1Score,xlab="",ylab="F1-Score")

 El punto donde se alcanzaría el máximo sería:

max(BANK\_KS$F1Score)

## [1] 0.5644685

y el valor donde se alcanza este valor es el que nos va a permitir seleccionar el umbral

which(BANK\_KS$F1Score==max(BANK\_KS$F1Score))

## [1] 2303

BANK\_KS[2303,c("CAMP\_DEPOSITOS","prediccion")]

## CAMP\_DEPOSITOS prediccion  
## 16427 1 0.303446

en este caso el umbral seleccionado que maximiza el F1-Score es 0.303446.

### Bloque de selección de umbral como el punto que maximiza el Beneficio de la campaña

Si consideramos un coste por llamada y un beneficio por venta

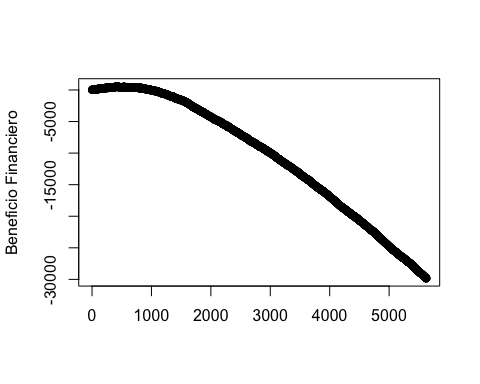
costeLlamada=10  
beneficioVenta=15

Podemos definir el beneficio o pérdida que tendrán nuestras acciones

BANK\_KS$BeneficioTP=beneficioVenta-costeLlamada  
BANK\_KS$BeneficioTN=0  
BANK\_KS$PerdidaFP=-costeLlamada  
BANK\_KS$PerdidaFN=-beneficioVenta

Con estos datos podemos calcular el beneficio financiero en función al umbral de corte para seleccionar el punto máximo que nos defina el umbral a considerar

BANK\_KS$BeneficioFinan=BANK\_KS$EXITOS\_ACUM\*BANK\_KS$BeneficioTP+  
 BANK\_KS$FRACASOS\_ACUM\*BANK\_KS$PerdidaFP  
plot(BANK\_KS$BeneficioFinan,xlab="",ylab="Beneficio Financiero")



max(BANK\_KS$BeneficioFinan)

## [1] 565

which(BANK\_KS$BeneficioFinan==max(BANK\_KS$BeneficioFinan))

## [1] 422 425

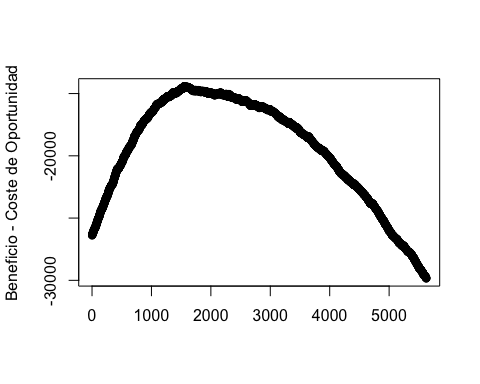
BANK\_KS[422,c("CAMP\_DEPOSITOS","prediccion")]

## CAMP\_DEPOSITOS prediccion  
## 30490 1 0.6921934

En este caso tenemos varios puntos máximos por lo que podemos seleccionar uno, eligiendo el primero, el umbral sería del 0.6921934 y hubiéramos obtenido un beneficio de 565.

Pero podemos considerar que dejar de llamar a un cliente que va a contratar supone un coste de oportunidad en el que incurrimos por lo que podemos calcular el umbral que maximice el beneficio teniendo en cuenta el coste de oportunidad:

BANK\_KS$Oportunidad=BANK\_KS$EXITOS\_ACUM\*BANK\_KS$BeneficioTP+  
 (BANK\_KS$EXITOS\_TOT-BANK\_KS$EXITOS\_ACUM)\*BANK\_KS$PerdidaFN+  
 BANK\_KS$FRACASOS\_ACUM\*BANK\_KS$PerdidaFP+  
 (BANK\_KS$FRACASOS\_TOT-BANK\_KS$FRACASOS\_ACUM)\*BANK\_KS$BeneficioTN  
plot(BANK\_KS$Oportunidad,xlab="",ylab="Beneficio - Coste de Oportunidad")



max(BANK\_KS$Oportunidad)

## [1] -14425

which(BANK\_KS$Oportunidad==max(BANK\_KS$Oportunidad))

## [1] 1546 1549

BANK\_KS[1546,c("CAMP\_DEPOSITOS","prediccion")]

## CAMP\_DEPOSITOS prediccion  
## 4015 1 0.3629083

En este caso, el umbral se fijaría en el 0.3629083 y hubiéramos obtenido una perdida de -14.425.

# Escribiendo del Resultado FINAL

***Revisando los resultados obtenidos, nos quedamos con el modelo de Regresion logistica Modelo 4, ya que tiene mejores resultados en accuracy, por lo que generamos la data propuesta para el ejercicio.***

df\_seguronueva$prediccion=predict(modelo4, newdata=df\_seguronueva,type="response")  
  
ALPHA=0.6921934  
df\_seguronueva$resultado= c(df\_seguronueva$prediccion>=ALPHA)  
df\_seguronueva$resultado=as.integer(df\_seguronueva$resultado)  
  
head(df\_seguronueva, 10)

## COD\_CLIENTE CAT\_EDAD ESTADO\_CIVIL NIVEL\_ESTUDIOS RANGO\_INGRESOS  
## 1 CLI0043828980 2\_MINUS30 SOLTERO SECUNDARIA [1000-2000>  
## 2 CLI0062632001 1\_NULL SOLTERO SIN\_ESTUDIOS <600  
## 3 CLI0044539972 2\_MINUS30 SOLTERO PRIMARIA [1000-2000>  
## 4 CLI0074041049 2\_MINUS30 SOLTERO PRIMARIA <600  
## 5 CLI0008226026 4\_PLUS40 SOLTERO UNIVERSIDAD [1000-2000>  
## 6 CLI0041998439 4\_PLUS40 SOLTERO SECUNDARIA [1000-2000>  
## 7 CLI0041534199 3\_30\_40 SOLTERO PRIMARIA [600-1000>  
## 8 CLI0016745258 3\_30\_40 SOLTERO PRIMARIA [1000-2000>  
## 9 CLI0043147232 2\_MINUS30 SOLTERO SECUNDARIA [600-1000>  
## 10 CLI0073478861 2\_MINUS30 SOLTERO SIN\_ESTUDIOS [600-1000>  
## SEXO prediccion resultado  
## 1 HOMBRE 0.1992495 0  
## 2 HOMBRE 0.1155702 0  
## 3 MUJER 0.2042410 0  
## 4 HOMBRE 0.1278564 0  
## 5 MUJER 0.1303752 0  
## 6 MUJER 0.3065728 0  
## 7 HOMBRE 0.4307699 0  
## 8 MUJER 0.5698746 0  
## 9 MUJER 0.3034460 0  
## 10 HOMBRE 0.1381613 0

write.csv(df\_seguronueva, file = "outputmodel.csv", row.names = FALSE)