Predicción de abandono de compra

Machine Learning Predictivo

Beatriz Cubillas

10/2/2022

---

1 ID Petición

- Número 1: Predicción de los casos de abandono de clientes de un comercio con machine learning

2 Contexto

- El equipo de dirección quiere tener conocimiento sobre la tasa de abanadono de nuestros clientes.
- Conocer cual sería el tamaño óptimo de la campaña de marketing sobre los clientes que posiblemente abandonasen la compañía y el retorno de la inversión (ROI).
- El departamento de marketing ha solicitado al equipo de Data Science una solución para obtener el mayor ROI posible en las campañas comerciales haciendo un estudio de los datos históricos con los que cuenta la compañía.

3 Requirimientos del proyecto

- Mejorar el ROI de las campañas comerciales
- Disminuir el abandono de clientes

---

4 Detalle del proceso

4.1 - Preparación del entorno

- 4.1.1 - Cargamos las librerías que vamos a utilizar

```
paquetes <- c('data.table',
              'dplyr',
              'tidyr',
              'ggplot2',
              'randomForest',
              'ROCR',
              'purrr',
              'smbinning',
              'rpart',
              'rpart.plot')

instalados <- paquetes %in% installed.packages()

if(sum(instalados == FALSE) > 0) {
  install.packages(paquetes[!instalados])
}
lapply(paquetes,require,character.only = TRUE)
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
##
## [[6]]
## [1] TRUE
##
## [[7]]
## [1] TRUE
##
## [[8]]
## [1] TRUE
##
## [[9]]
## [1] TRUE
##
## [[10]]
## [1] TRUE
```

Parámetros - Desactivamos la notación científica:

```
options(scipen=999)
```

- 4.1.2 - Cargamos los datos

Procedentes de la base de datos histórica de la compañía, archivo '.xlsx'.

```
library(readxl)
df <- read_excel("C:/_BEATRIZ/03.- ML Predictivo/Proy2ChurnSale/Data2.xlsx")
```

4.2 - Análisis exploratorio

- 4.2.1 - Análisis exploratorio general y tipo de datos

```
as.data.frame(sort(names(df)))
```

```
##                sort(names(df))
## 1               CashbackAmount
## 2                        Churn
## 3                     CityTier
## 4                     Complain
## 5                   CouponUsed
## 6                   CustomerID
## 7            DaySinceLastOrder
## 8                       Gender
## 9                HourSpendOnApp
## 10                MaritalStatus
## 11              NumberOfAddress
## 12    NumberOfDeviceRegistered
## 13 OrderAmountHikeFromlastYear
## 14                   OrderCount
## 15             PreferedOrderCat
## 16         PreferredLoginDevice
## 17         PreferredPaymentMode
## 18            SatisfactionScore
## 19                      Tenure
## 20              WarehouseToHome
```

```
str(df)
```

```
## tibble [5,630 x 20] (S3: tbl_df/tbl/data.frame)
##  $ CustomerID                : num [1:5630] 50001 50002 50003 50004 50005 ...
##  $ Churn                     : num [1:5630] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Tenure                    : num [1:5630] 4 NA NA 0 0 0 NA NA 13 NA ...
##  $ PreferredLoginDevice      : chr [1:5630] "Mobile Phone" "Phone" "Phone" "Pho
ne" ...
##  $ CityTier                  : num [1:5630] 3 1 1 3 1 1 3 1 3 1 ...
##  $ WarehouseToHome           : num [1:5630] 6 8 30 15 12 22 11 6 9 31 ...
##  $ PreferredPaymentMode      : chr [1:5630] "Debit Card" "UPI" "Debit Card" "De
bit Card" ...
##  $ Gender                    : chr [1:5630] "Female" "Male" "Male" "Male" ...
##  $ HourSpendOnApp            : num [1:5630] 3 3 2 2 NA 3 2 3 NA 2 ...
##  $ NumberOfDeviceRegistered  : num [1:5630] 3 4 4 4 3 5 3 3 4 5 ...
##  $ PreferedOrderCat          : chr [1:5630] "Laptop & Accessory" "Mobile" "Mobi
le" "Laptop & Accessory" ...
##  $ SatisfactionScore         : num [1:5630] 2 3 3 5 5 5 2 2 3 3 ...
##  $ MaritalStatus             : chr [1:5630] "Single" "Single" "Single" "Single"
...
##  $ NumberOfAddress           : num [1:5630] 9 7 6 8 3 2 4 3 2 2 ...
##  $ Complain                  : num [1:5630] 1 1 1 0 0 1 0 1 1 0 ...
##  $ OrderAmountHikeFromlastYear: num [1:5630] 11 15 14 23 11 22 14 16 14 12 ...
##  $ CouponUsed                : num [1:5630] 1 0 0 0 1 4 0 2 0 1 ...
##  $ OrderCount                : num [1:5630] 1 1 1 1 1 6 1 2 1 1 ...
##  $ DaySinceLastOrder         : num [1:5630] 5 0 3 3 3 7 0 0 2 1 ...
##  $ CashbackAmount            : num [1:5630] 160 121 120 134 130 ...
```

```
glimpse(df)
```

```
## Rows: 5,630
## Columns: 20
## $ CustomerID                <dbl> 50001, 50002, 50003, 50004, 50005, 50006, ~
## $ Churn                     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ Tenure                    <dbl> 4, NA, NA, 0, 0, 0, NA, NA, 13, NA, 4, 11,~
## $ PreferredLoginDevice      <chr> "Mobile Phone", "Phone", "Phone", "Phone",~
## $ CityTier                  <dbl> 3, 1, 1, 3, 1, 1, 3, 1, 3, 1, 1, 1, 1, 1, ~
## $ WarehouseToHome           <dbl> 6, 8, 30, 15, 12, 22, 11, 6, 9, 31, 18, 6,~
## $ PreferredPaymentMode      <chr> "Debit Card", "UPI", "Debit Card", "Debit ~
## $ Gender                    <chr> "Female", "Male", "Male", "Male", "Male", ~
## $ HourSpendOnApp            <dbl> 3, 3, 2, 2, NA, 3, 2, 3, NA, 2, 2, 3, 2, 3~
## $ NumberOfDeviceRegistered  <dbl> 3, 4, 4, 4, 3, 5, 3, 3, 4, 5, 3, 4, 3, 4, ~
## $ PreferedOrderCat          <chr> "Laptop & Accessory", "Mobile", "Mobile", ~
## $ SatisfactionScore         <dbl> 2, 3, 3, 5, 5, 5, 2, 2, 3, 3, 3, 3, 3, 3, ~
## $ MaritalStatus             <chr> "Single", "Single", "Single", "Single", "S~
## $ NumberOfAddress           <dbl> 9, 7, 6, 8, 3, 2, 4, 3, 2, 2, 2, 10, 2, 1,~
## $ Complain                  <dbl> 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, ~
## $ OrderAmountHikeFromlastYear <dbl> 11, 15, 14, 23, 11, 22, 14, 16, 14, 12, NA~
## $ CouponUsed                <dbl> 1, 0, 0, 0, 1, 4, 0, 2, 0, 1, 9, 0, 2, 0, ~
## $ OrderCount                <dbl> 1, 1, 1, 1, 1, 6, 1, 2, 1, 1, 15, 1, 2, 1,~
## $ DaySinceLastOrder         <dbl> 5, 0, 3, 3, 3, 7, 0, 0, 2, 1, 8, 0, 2, 0, ~
## $ CashbackAmount            <dbl> 159.93, 120.90, 120.28, 134.07, 129.60, 13~
```

Vemos que tenemos 20 variables, de las que 15 son numéricas que continuarán siendo númericas, las otras 5 son de tipo caracter, vamos a pasarlas a factor.

Hay variables que son categóricas y las vamos a transformar en factores, de momento las guardamos en la variable ‘a_factores’.

```
a_factores <- c("PreferredLoginDevice","PreferredPaymentMode","Gender","PreferedOrd
erCat","MaritalStatus","CityTier","SatisfactionScore","Complain")
a_factores
```

```
## [1] "PreferredLoginDevice" "PreferredPaymentMode" "Gender"
## [4] "PreferedOrderCat"     "MaritalStatus"        "CityTier"
## [7] "SatisfactionScore"    "Complain"
```

- 4.2.2 - Calidad de datos:

Calculamos los estadísticos básicos de cada variable:

```
lapply(df,summary)
```

```
## $CustomerID
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   50001   51408   52816   52816   54223   55630
##
## $Churn
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1684  0.0000  1.0000
##
## $Tenure
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00    2.00    9.00   10.19   16.00   61.00     264
##
## $PreferredLoginDevice
##    Length     Class      Mode
##      5630 character character
##
## $CityTier
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.655   3.000   3.000
##
## $WarehouseToHome
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    5.00    9.00   14.00   15.64   20.00  127.00     251
##
## $PreferredPaymentMode
##    Length     Class      Mode
##      5630 character character
##
## $Gender
##    Length     Class      Mode
##      5630 character character
##
## $HourSpendOnApp
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   2.000   3.000   2.932   3.000   5.000     255
##
## $NumberOfDeviceRegistered
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   4.000   3.689   4.000   6.000
##
## $PreferedOrderCat
##    Length     Class      Mode
##      5630 character character
##
## $SatisfactionScore
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.067   4.000   5.000
##
## $MaritalStatus
##    Length     Class      Mode
##      5630 character character
##
## $NumberOfAddress
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   1.000   2.000   3.000   4.214   6.000  22.000
##
## $Complain
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.2849  1.0000  1.0000
##
## $OrderAmountHikeFromlastYear
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   11.00   13.00   15.00   15.71   18.00   26.00     265
##
## $CouponUsed
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   1.000   1.000   1.751   2.000  16.000     256
##
## $OrderCount
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   1.000   2.000   3.008   3.000  16.000     258
##
## $DaySinceLastOrder
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   2.000   3.000   4.543   7.000  46.000     307
##
## $CashbackAmount
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   145.8   163.3   177.2   196.4   325.0
```

Hemos encontramos datos nulos en siete de las variables.

También vemos que en la permanencia 'Tenure' el valor del tercer cuartil es 16, que es muy bajo en comparación con el máximo, que es 61, profundizaremos en ello más adelante.

La distancia desde el almacén hasta el domicilio 'WarehouseToHome', tiene un tercer cuartil de 20 y un máximo de 127. Lo consideramos normal desde el punto de vista de negocio, ya que puede ser que tengamos pocos clientes que viven a mucha distancia, no los consideremos datos atípicos.

También contemplamos que los días desde el último pedido 'DaySinceLastOrder', también tiene un valor del tercer cuartil de 7 días y un máximo de 46 días. Puede ser un dato real, ya que podemos tener muy pocos clientes que dejen pasar muchos días desde el último pedido, la mayoría de nuestros clientes realiza nuevos pedidos antes de los siete días.

- 4.2.3 - Calidad de datos: Análisis de nulos

```
data.frame(colSums(is.na(df)))
```

```
##                              colSums.is.na.df..
## CustomerID                                    0
## Churn                                         0
## Tenure                                      264
## PreferredLoginDevice                          0
## CityTier                                      0
## WarehouseToHome                             251
## PreferredPaymentMode                          0
## Gender                                        0
## HourSpendOnApp                              255
## NumberOfDeviceRegistered                      0
## PreferedOrderCat                              0
## SatisfactionScore                             0
## MaritalStatus                                 0
## NumberOfAddress                               0
## Complain                                      0
## OrderAmountHikeFromlastYear                 265
## CouponUsed                                  256
## OrderCount                                  258
## DaySinceLastOrder                           307
## CashbackAmount                                0
```

- Confirmamos que hay 7 variables con multiples nulos.

Vamos a sustituir los valores faltantes “NA”, de varias variables por ceros, realizamos esto ya que la cantidad de valores faltantes es del 5% o menor en cada variable, por lo tanto adjudicar ceros a esos valores no trastoca en gran medida la fuente de datos original.

En el caso del aumento de la cantidad de pedidos desde el año pasado ‘OrderAmountHikeFromLastYear’, nos interesa que el dato sea cero sino lo conocemos, así consideramos que no hay incremento de pedidos.

```
df$OrderAmountHikeFromlastYear [is.na (df$OrderAmountHikeFromlastYear)] <- 0
```

Para la variable de la permanencia ‘Tenure’, habiendo comprobado que había muchos clientes nuevos, con permanencia de 0 meses, nosotros consideramos que para valores faltantes en esta variable esos clientes tendrán antigüedad cero meses.

```
df$Tenure [is.na (df$Tenure)] <- 0
```

Consideramos los valores faltantes en la columna del uso de cupones, ‘CouponUsed’ como ceros, ya que es mas favorable considerar que no se ha utilizado ningún cupón en el caso de no tener registro de los datos.

```
df$CouponUsed [is.na (df$CouponUsed)] <- 0
```

Finalmente en el caso del recuento de pedidos ‘OrderCount’, también consideramos que los clientes tendrán cero pedidos anteriores cuando no poseemos datos.

```
df$OrderCount [is.na (df$OrderCount)] <- 0
```

Para el caso de la distancia desde el almacén hasta la casa de los clientes, ‘WarehouseToHome vamos a considerar el valor medio de las distancias para aplicar en los casos en los que no disponemos de datos. Igual operación realizaremos con las horas empleadas en la aplicación ’HourSpendOnApp’ y con los días

desde la última compra ‘DaySinceLastOrder’, ya que considerar ceros en los valores perdidos de estas variables hará que la media de estos valores se vea disminuida, no siendo un valor coherente.

```
df <- mutate_at (df, c("WarehouseToHome", "HourSpendOnApp","DaySinceLastOrder"), ~r
eplace_na(., mean(., na.rm = TRUE)))
```

Hemos eliminado los datos faltantes y volvemos a comprobarlo.

```
data.frame(colSums(is.na(df)))
```

```
##                              colSums.is.na.df..
## CustomerID                                    0
## Churn                                         0
## Tenure                                        0
## PreferredLoginDevice                          0
## CityTier                                      0
## WarehouseToHome                               0
## PreferredPaymentMode                          0
## Gender                                        0
## HourSpendOnApp                                0
## NumberOfDeviceRegistered                      0
## PreferedOrderCat                              0
## SatisfactionScore                             0
## MaritalStatus                                 0
## NumberOfAddress                               0
## Complain                                      0
## OrderAmountHikeFromlastYear                   0
## CouponUsed                                    0
## OrderCount                                    0
## DaySinceLastOrder                             0
## CashbackAmount                                0
```

Vemos que ya no existen datos faltantes, vamos a obtener el resumen estadístico de las variables.

```
lapply(df,summary)
```

```
## $CustomerID
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   50001   51408   52816   52816   54223   55630
##
## $Churn
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1684  0.0000  1.0000
##
## $Tenure
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   8.000   9.712  15.000  61.000
##
## $PreferredLoginDevice
##    Length     Class      Mode
##      5630 character character
##
## $CityTier
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.655   3.000   3.000
##
## $WarehouseToHome
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00    9.00   14.00   15.64   20.00  127.00
##
## $PreferredPaymentMode
##    Length     Class      Mode
##      5630 character character
##
## $Gender
##    Length     Class      Mode
##      5630 character character
##
## $HourSpendOnApp
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   3.000   2.932   3.000   5.000
##
## $NumberOfDeviceRegistered
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   4.000   3.689   4.000   6.000
##
## $PreferedOrderCat
##    Length     Class      Mode
##      5630 character character
##
## $SatisfactionScore
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.067   4.000   5.000
##
## $MaritalStatus
##    Length     Class      Mode
##      5630 character character
##
## $NumberOfAddress
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   1.000   2.000   3.000   4.214   6.000  22.000
##
## $Complain
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.2849  1.0000  1.0000
##
## $OrderAmountHikeFromlastYear
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   13.00   14.00   14.97   18.00   26.00
##
## $CouponUsed
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.671   2.000  16.000
##
## $OrderCount
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    2.00    2.87    3.00   16.00
##
## $DaySinceLastOrder
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   4.000   4.543   7.000  46.000
##
## $CashbackAmount
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   145.8   163.3   177.2   196.4   325.0
```

Comprobamos que las medias y medianas no han cambiado significativamente cuando hemos sustituido los datos faltantes “NA”, bien por la media del resto de los datos de la variable, bien por ceros. Continuamos con el análisis exploratorio de los datos.

- 4.2.4 - Calidad de datos: Análisis de ceros

```
contar_ceros <- function(variable) {
    temp <- transmute(df,if_else(variable==0,1,0))
    sum(temp)
}

num_ceros <- sapply(df,contar_ceros)
num_ceros <- data.frame(VARIABLE=names(num_ceros),CEROS = as.numeric(num_ceros),str
ingsAsFactors = F)
num_ceros <- num_ceros %>%
  arrange(desc(CEROS)) %>%
  mutate(PORCENTAJE = CEROS / nrow(df) * 100)
num_ceros
```

```
##                       VARIABLE CEROS  PORCENTAJE
## 1                        Churn  4682 83.16163410
## 2                      Complain  4026 71.50976909
## 3                    CouponUsed  1286 22.84191829
## 4                        Tenure   772 13.71225577
## 5             DaySinceLastOrder   496  8.80994671
## 6  OrderAmountHikeFromlastYear   265  4.70692718
## 7                    OrderCount   258  4.58259325
## 8                CashbackAmount     4  0.07104796
## 9                HourSpendOnApp     3  0.05328597
## 10                   CustomerID     0  0.00000000
## 11        PreferredLoginDevice     0  0.00000000
## 12                     CityTier     0  0.00000000
## 13             WarehouseToHome     0  0.00000000
## 14        PreferredPaymentMode     0  0.00000000
## 15                      Gender     0  0.00000000
## 16    NumberOfDeviceRegistered     0  0.00000000
## 17             PreferedOrderCat     0  0.00000000
## 18           SatisfactionScore     0  0.00000000
## 19               MaritalStatus     0  0.00000000
## 20             NumberOfAddress     0  0.00000000
```

Vemos que hay dos variales con una cantidad de ceros superior al 70%, que son la variable abandono, que nos indica que el 83% de los clientes no abandonan la compra, y tambien que el 71 % de los clientes no ha realizado una queja, ambos datos son coherentes con la realidad. También vemos que el 78 % de los clientes realizan su compra utilizando un cupón de descuento

Por otro lado mencionar la baja antigüedad que tienen los clientes, ya que un 13 % de los mismos son nuevos del mes en curso. Analizaremos esta variable después.

- 4.2.5 - Calidad de datos: Análisis de atípicos

4.2.5.1 - Analizamos las que son de tipo numérico

Obtenemos un listado de los 20 registros mas altos ordenados de forma decreciente.

```
out <- function(variable){
  t(t(head(sort(variable,decreasing = T),20)))
}
lapply(df,function(x){
  if(is.double(x)) out(x)
})
```

```
## $CustomerID
##        [,1]
##  [1,] 55630
##  [2,] 55629
##  [3,] 55628
##  [4,] 55627
##  [5,] 55626
##  [6,] 55625
##  [7,] 55624
##  [8,] 55623
##  [9,] 55622
## [10,] 55621
## [11,] 55620
## [12,] 55619
## [13,] 55618
## [14,] 55617
## [15,] 55616
## [16,] 55615
## [17,] 55614
## [18,] 55613
## [19,] 55612
## [20,] 55611
##
## $Churn
##       [,1]
##  [1,]    1
##  [2,]    1
##  [3,]    1
##  [4,]    1
##  [5,]    1
##  [6,]    1
##  [7,]    1
##  [8,]    1
##  [9,]    1
## [10,]    1
## [11,]    1
## [12,]    1
## [13,]    1
## [14,]    1
## [15,]    1
## [16,]    1
## [17,]    1
## [18,]    1
## [19,]    1
## [20,]    1
##
## $Tenure
##       [,1]
##  [1,]   61
##  [2,]   60
##  [3,]   51
##  [4,]   50
##  [5,]   31
##  [6,]   31
```

```
##  [7,]   31
##  [8,]   31
##  [9,]   31
## [10,]   31
## [11,]   31
## [12,]   31
## [13,]   31
## [14,]   31
## [15,]   31
## [16,]   31
## [17,]   31
## [18,]   31
## [19,]   31
## [20,]   31
##
## $PreferredLoginDevice
## NULL
##
## $CityTier
##       [,1]
##  [1,]    3
##  [2,]    3
##  [3,]    3
##  [4,]    3
##  [5,]    3
##  [6,]    3
##  [7,]    3
##  [8,]    3
##  [9,]    3
## [10,]    3
## [11,]    3
## [12,]    3
## [13,]    3
## [14,]    3
## [15,]    3
## [16,]    3
## [17,]    3
## [18,]    3
## [19,]    3
## [20,]    3
##
## $WarehouseToHome
##       [,1]
##  [1,]  127
##  [2,]  126
##  [3,]   36
##  [4,]   36
##  [5,]   36
##  [6,]   36
##  [7,]   36
##  [8,]   36
##  [9,]   36
## [10,]   36
## [11,]   36
## [12,]   36
```

```
## [13,]   36
## [14,]   36
## [15,]   36
## [16,]   36
## [17,]   36
## [18,]   36
## [19,]   36
## [20,]   36
##
## $PreferredPaymentMode
## NULL
##
## $Gender
## NULL
##
## $HourSpendOnApp
##       [,1]
##  [1,]    5
##  [2,]    5
##  [3,]    5
##  [4,]    4
##  [5,]    4
##  [6,]    4
##  [7,]    4
##  [8,]    4
##  [9,]    4
## [10,]    4
## [11,]    4
## [12,]    4
## [13,]    4
## [14,]    4
## [15,]    4
## [16,]    4
## [17,]    4
## [18,]    4
## [19,]    4
## [20,]    4
##
## $NumberOfDeviceRegistered
##       [,1]
##  [1,]    6
##  [2,]    6
##  [3,]    6
##  [4,]    6
##  [5,]    6
##  [6,]    6
##  [7,]    6
##  [8,]    6
##  [9,]    6
## [10,]    6
## [11,]    6
## [12,]    6
## [13,]    6
## [14,]    6
## [15,]    6
```

```
## [16,]    6
## [17,]    6
## [18,]    6
## [19,]    6
## [20,]    6
##
## $PreferedOrderCat
## NULL
##
## $SatisfactionScore
##       [,1]
##  [1,]    5
##  [2,]    5
##  [3,]    5
##  [4,]    5
##  [5,]    5
##  [6,]    5
##  [7,]    5
##  [8,]    5
##  [9,]    5
## [10,]    5
## [11,]    5
## [12,]    5
## [13,]    5
## [14,]    5
## [15,]    5
## [16,]    5
## [17,]    5
## [18,]    5
## [19,]    5
## [20,]    5
##
## $MaritalStatus
## NULL
##
## $NumberOfAddress
##       [,1]
##  [1,]   22
##  [2,]   21
##  [3,]   20
##  [4,]   19
##  [5,]   11
##  [6,]   11
##  [7,]   11
##  [8,]   11
##  [9,]   11
## [10,]   11
## [11,]   11
## [12,]   11
## [13,]   11
## [14,]   11
## [15,]   11
## [16,]   11
## [17,]   11
## [18,]   11
```

```
## [19,]   11
## [20,]   11
##
## $Complain
##       [,1]
##  [1,]    1
##  [2,]    1
##  [3,]    1
##  [4,]    1
##  [5,]    1
##  [6,]    1
##  [7,]    1
##  [8,]    1
##  [9,]    1
## [10,]    1
## [11,]    1
## [12,]    1
## [13,]    1
## [14,]    1
## [15,]    1
## [16,]    1
## [17,]    1
## [18,]    1
## [19,]    1
## [20,]    1
##
## $OrderAmountHikeFromlastYear
##       [,1]
##  [1,]   26
##  [2,]   26
##  [3,]   26
##  [4,]   26
##  [5,]   26
##  [6,]   26
##  [7,]   26
##  [8,]   26
##  [9,]   26
## [10,]   26
## [11,]   26
## [12,]   26
## [13,]   26
## [14,]   26
## [15,]   26
## [16,]   26
## [17,]   26
## [18,]   26
## [19,]   26
## [20,]   26
##
## $CouponUsed
##       [,1]
##  [1,]   16
##  [2,]   16
##  [3,]   15
##  [4,]   14
```

```
##  [5,]   14
##  [6,]   14
##  [7,]   14
##  [8,]   14
##  [9,]   13
## [10,]   13
## [11,]   13
## [12,]   13
## [13,]   13
## [14,]   13
## [15,]   13
## [16,]   13
## [17,]   12
## [18,]   12
## [19,]   12
## [20,]   12
##
## $OrderCount
##       [,1]
##  [1,]   16
##  [2,]   16
##  [3,]   16
##  [4,]   16
##  [5,]   16
##  [6,]   16
##  [7,]   16
##  [8,]   16
##  [9,]   16
## [10,]   16
## [11,]   16
## [12,]   16
## [13,]   16
## [14,]   16
## [15,]   16
## [16,]   16
## [17,]   16
## [18,]   16
## [19,]   16
## [20,]   16
##
## $DaySinceLastOrder
##       [,1]
##  [1,]   46
##  [2,]   31
##  [3,]   30
##  [4,]   18
##  [5,]   18
##  [6,]   18
##  [7,]   18
##  [8,]   18
##  [9,]   18
## [10,]   18
## [11,]   18
## [12,]   18
## [13,]   18
```

```
## [14,]   17
## [15,]   17
## [16,]   17
## [17,]   17
## [18,]   17
## [19,]   17
## [20,]   17
##
## $CashbackAmount
##         [,1]
##  [1,] 324.99
##  [2,] 324.99
##  [3,] 324.73
##  [4,] 324.73
##  [5,] 324.43
##  [6,] 324.43
##  [7,] 324.26
##  [8,] 324.26
##  [9,] 323.59
## [10,] 323.59
## [11,] 323.47
## [12,] 323.47
## [13,] 323.45
## [14,] 323.45
## [15,] 323.33
## [16,] 323.33
## [17,] 322.40
## [18,] 322.40
## [19,] 322.17
## [20,] 322.17
```

En la permanencia 'Tenure' vemos que hay cuatro valores muy altos, son atípicos, pero como vamos a discretizar no tenemos que realizar ninguna acción sobre los mismos. En 'WarehouseToHome', la distancia desde el almacén hasta la casa del cliente, hay dos valores que salen de la media, no nos supone problema porque vamos a discretizar esta variable. En 'DaySinceLastOrder' existen también tres valores en los máximos que salen de la media habitual de valores, también discretizaremos esta variable.

4.2.5.2 - Analizamos las que son de tipo integer

Obtenemos un resumen 'table' para cada variable de este tipo.

```
out <- function(variable){
  t(t(table(variable)))
}
lapply(df,function(x){
  if(is.character(x)) out(x)
})
```

```
## $CustomerID
## NULL
##
## $Churn
## NULL
##
## $Tenure
## NULL
##
## $PreferredLoginDevice
##
## variable       [,1]
##   Computer     1634
##   Mobile Phone 2765
##   Phone        1231
##
## $CityTier
## NULL
##
## $WarehouseToHome
## NULL
##
## $PreferredPaymentMode
##
## variable           [,1]
##   Cash on Delivery  149
##   CC                273
##   COD               365
##   Credit Card      1501
##   Debit Card       2314
##   E wallet          614
##   UPI               414
##
## $Gender
##
## variable [,1]
##   Female 2246
##   Male   3384
##
## $HourSpendOnApp
## NULL
##
## $NumberOfDeviceRegistered
## NULL
##
## $PreferedOrderCat
##
## variable             [,1]
##   Fashion             826
##   Grocery             410
##   Laptop & Accessory 2050
##   Mobile              809
##   Mobile Phone       1271
##   Others              264
```

```
##
## $SatisfactionScore
## NULL
##
## $MaritalStatus
##
## variable   [,1]
##   Divorced  848
##   Married  2986
##   Single   1796
##
## $NumberOfAddress
## NULL
##
## $Complain
## NULL
##
## $OrderAmountHikeFromlastYear
## NULL
##
## $CouponUsed
## NULL
##
## $OrderCount
## NULL
##
## $DaySinceLastOrder
## NULL
##
## $CashbackAmount
## NULL
```

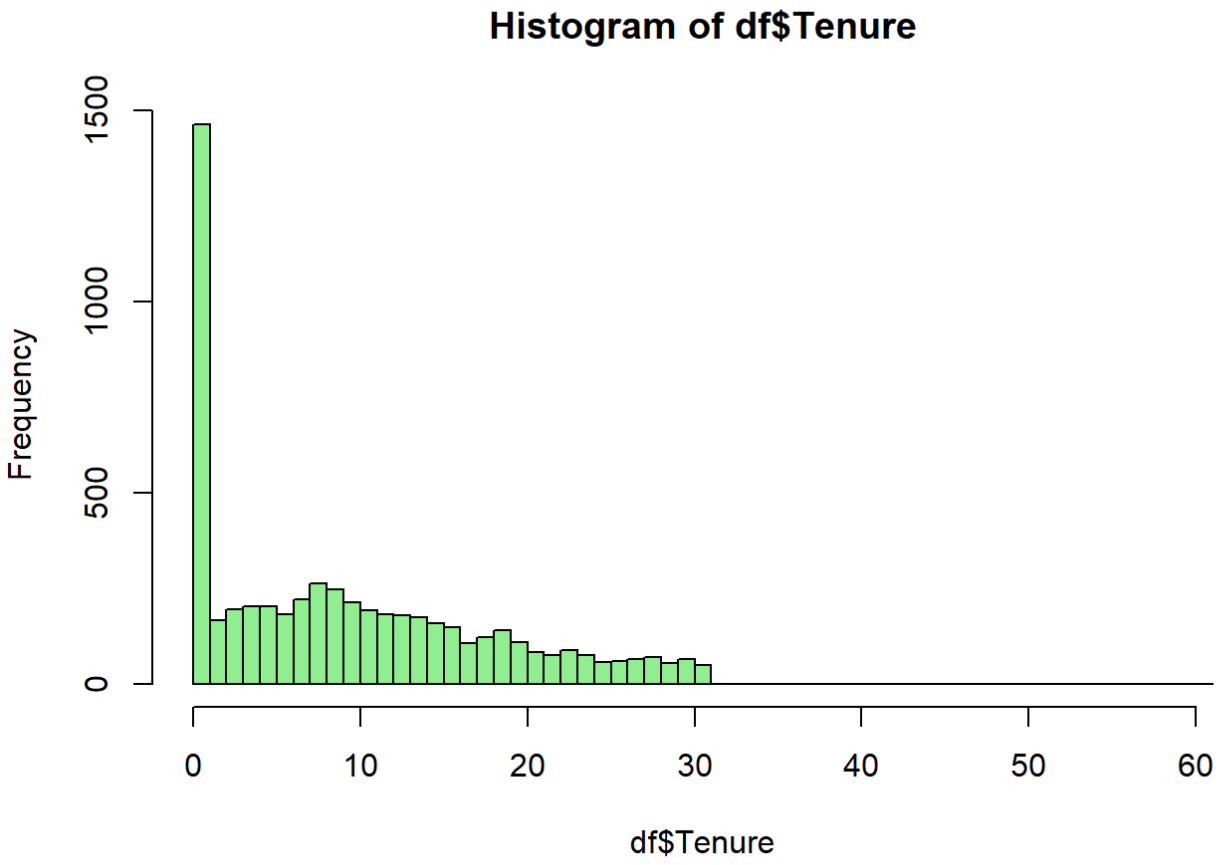
No se aprecia ningún dato llamativo en este resumen.

- 4.2.6 Otros

4.2.6.1 Análisis de la antigüedad

Y ahora analizamos la antigüedad con un gráfico:

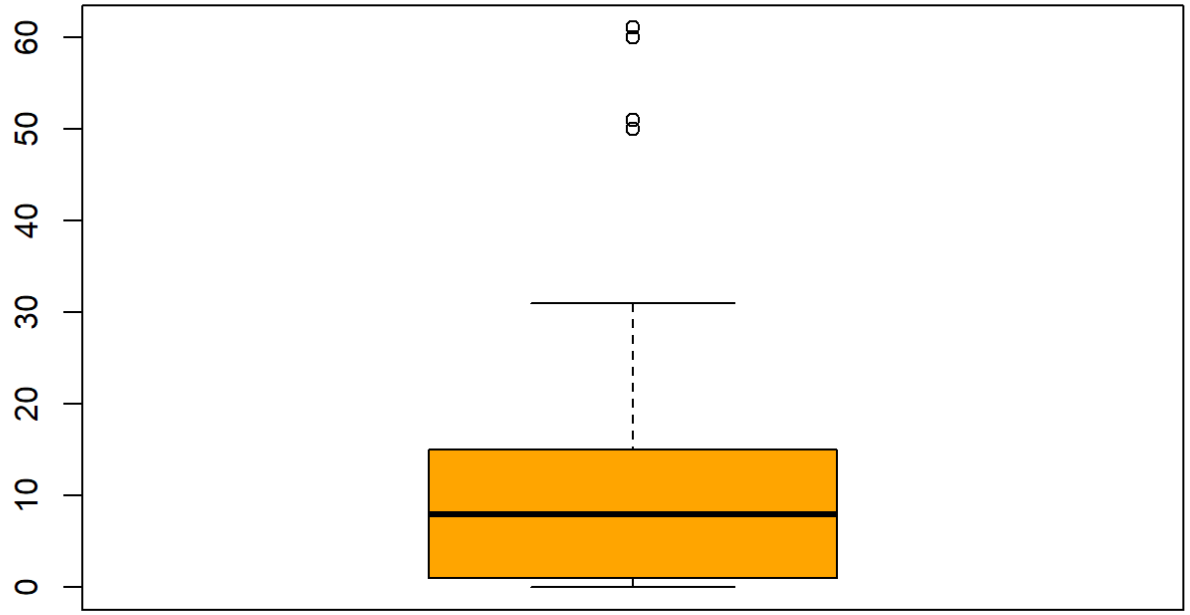
```
hist(df$Tenure,breaks = 50, col='lightgreen')
```



Vemos una frecuencia desproporcionada de clientes con antigüedad de cero meses, eso quiere decir que tenemos muchos clientes nuevos, cuya primera compra se ha realizado en el mes de estudio. Vamos a discretizar la permanencia por lo tanto no supone un problema este hecho.

Vemos los valores outliers con un gráfico boxplot.

```
boxplot(df$Tenure, col= "orange")
```



Decidimos eliminar las filas de esos cuatro valores, ya que han de ser valores incorrectos y nos estan influenciando negativamente el resto de valores. Pasamos de tener 5630 filas en nuestro dataframe a tener 5626.

- 4.2.7 - Acciones resultado del analisis de calidad de datos y exploratorio

Una vez finalizado el análisis de los datos, vamos a transformar a tipo factor las variables contenidas en 'a_factores', exceptuando la variable 'Churn' que va a ser nuestra variable target.

También eliminamos la variable "NumberOfAddress", ya que el número de la calle del cliente no nos va a aportar información.

Eliminamos los cuatro datos atípicos de permanencia 'Tenure' que habíamos localizado, los que son mayores de 32 meses.

```
df <- df %>%
  mutate_at(a_factores,as.factor) %>%
  select(-NumberOfAddress)  %>%
  filter(Tenure < 32)
```

4.3 - Trasformación de datos

- 4.3.1 - Creación de la variable target

Creación de la variable abandono(churn) para el entrenamiento.

```
df <- df %>%
  mutate(TARGET1 = as.factor(as.numeric(ifelse((Churn == 1),1,0)
        ))) %>%
  select(-Churn)
```

- 4.3.2 - Preparación de las variables independientes

4.3.2.1 - Preselección de variables independientes

Creamos una lista larga con todas las variables independientes, tenemos 17 variables.

```
ind_larga<-names(df)
no_usar <- c('CustomerID','TARGET1')
ind_larga<-setdiff(ind_larga,no_usar)
ind_larga
```

```
##  [1] "Tenure"                       "PreferredLoginDevice"
##  [3] "CityTier"                     "WarehouseToHome"
##  [5] "PreferredPaymentMode"         "Gender"
##  [7] "HourSpendOnApp"               "NumberOfDeviceRegistered"
##  [9] "PreferedOrderCat"             "SatisfactionScore"
## [11] "MaritalStatus"                "Complain"
## [13] "OrderAmountHikeFromlastYear" "CouponUsed"
## [15] "OrderCount"                   "DaySinceLastOrder"
## [17] "CashbackAmount"
```

4.3.2.1.1 - Preselección con RandomForest

Con las 17 variables que tenemos ahora, generamos un listado de variables ordenadas por importancia.

```
pre_rf <- randomForest(formula = reformulate(ind_larga,'TARGET1'), data= df,mtry=2,
ntree=50, importance = T)
imp_rf <- importance(pre_rf)[,4]
imp_rf <- data.frame(VARIABLE = names(imp_rf), IMP_RF = imp_rf)
imp_rf <- imp_rf %>% arrange(desc(IMP_RF)) %>% mutate(RANKING_RF = 1:nrow(imp_rf))
imp_rf
```

```
##                                                VARIABLE    IMP_RF RANKING_RF
## Tenure                                           Tenure 314.85745          1
## CashbackAmount                             CashbackAmount 135.08160          2
## WarehouseToHome                           WarehouseToHome 111.22801          3
## DaySinceLastOrder                       DaySinceLastOrder  97.15845          4
## OrderAmountHikeFromlastYear OrderAmountHikeFromlastYear  92.67672          5
## Complain                                       Complain  91.85636          6
## SatisfactionScore                     SatisfactionScore  88.64576          7
## PreferredPaymentMode               PreferredPaymentMode  85.51116          8
## PreferedOrderCat                       PreferedOrderCat  75.69279          9
## NumberOfDeviceRegistered       NumberOfDeviceRegistered  62.18551         10
## MaritalStatus                             MaritalStatus  59.16027         11
## OrderCount                                   OrderCount  55.60082         12
## CouponUsed                                   CouponUsed  45.52453         13
## PreferredLoginDevice               PreferredLoginDevice  43.71671         14
## CityTier                                       CityTier  38.56612         15
## HourSpendOnApp                           HourSpendOnApp  36.84279         16
## Gender                                           Gender  26.91989         17
```

4.3.2.1.2 - Preselección con Information Value

Comprobamos con otro método como quedaría el listado de variables ordenadas por importancia.

```
temp <- mutate(df,TARGET1 = as.numeric(as.character(TARGET1))) %>% as.data.frame()
imp_iv <- smbinning.sumiv(temp[c(ind_larga,'TARGET1')],y="TARGET1")
```

```
##
##
  |
  |                                                      |   0%
  |
  |---                                                   |   6%
  |
  |------                                                |  11%
  |
  |--------                                              |  17%
  |
  |-----------                                           |  22%
  |
  |--------------                                        |  28%
  |
  |-----------------                                     |  33%
  |
  |-------------------                                   |  39%
  |
  |----------------------                                |  44%
  |
  |-------------------------                             |  50%
  |
  |---------------------------                           |  56%
  |
  |-------------------------------                       |  61%
  |
  |---------------------------------                     |  67%
  |
  |-----------------------------------                   |  72%
  |
  |---------------------------------------               |  78%
  |
  |-----------------------------------------             |  83%
  |
  |-------------------------------------------           |  89%
  |
  |-----------------------------------------------       |  94%
  |
  |--------------------------------------------------| 100%
##
```

```
imp_iv <- imp_iv %>% mutate(Ranking = 1:nrow(imp_iv), IV = ifelse(is.na(.$IV),0,I
V)) %>% select(-Process)
names(imp_iv) <- c('VARIABLE','IMP_IV','RANKING_IV')
imp_iv
```

```
##                       VARIABLE IMP_IV RANKING_IV
## 1                       Tenure 1.3065          1
## 12                     Complain 0.4018          2
## 9               PreferedOrderCat 0.3856          3
## 17                CashbackAmount 0.3380          4
## 16             DaySinceLastOrder 0.2806          5
## 11                 MaritalStatus 0.2251          6
## 2           PreferredLoginDevice 0.0940          7
## 5           PreferredPaymentMode 0.0897          8
## 10             SatisfactionScore 0.0881          9
## 8       NumberOfDeviceRegistered 0.0815         10
## 4                WarehouseToHome 0.0678         11
## 3                       CityTier 0.0502         12
## 13 OrderAmountHikeFromlastYear 0.0104         13
## 6                         Gender 0.0062         14
## 7                 HourSpendOnApp 0.0000         15
## 14                    CouponUsed 0.0000         16
## 15                    OrderCount 0.0000         17
```

4.3.2.1.3 - Preselección final

Comparamos los resultados obtenidos por los dos métodos y obtenemos el listado final de variables ordenadas por importancia.

```
imp_final <- inner_join(imp_rf,imp_iv,by='VARIABLE') %>%
  select(VARIABLE,IMP_RF,IMP_IV,RANKING_RF,RANKING_IV) %>%
  mutate(RANKING_TOT = RANKING_RF + RANKING_IV) %>%
  arrange(RANKING_TOT)
imp_final
```

```
##                       VARIABLE    IMP_RF IMP_IV RANKING_RF RANKING_IV
## 1                       Tenure 314.85745 1.3065          1          1
## 2                 CashbackAmount 135.08160 0.3380          2          4
## 3                       Complain  91.85636 0.4018          6          2
## 4              DaySinceLastOrder  97.15845 0.2806          4          5
## 5               PreferedOrderCat  75.69279 0.3856          9          3
## 6                WarehouseToHome 111.22801 0.0678          3         11
## 7              SatisfactionScore  88.64576 0.0881          7          9
## 8           PreferredPaymentMode  85.51116 0.0897          8          8
## 9                  MaritalStatus  59.16027 0.2251         11          6
## 10 OrderAmountHikeFromlastYear  92.67672 0.0104          5         13
## 11    NumberOfDeviceRegistered  62.18551 0.0815         10         10
## 12         PreferredLoginDevice  43.71671 0.0940         14          7
## 13                     CityTier  38.56612 0.0502         15         12
## 14                   OrderCount  55.60082 0.0000         12         17
## 15                   CouponUsed  45.52453 0.0000         13         16
## 16               HourSpendOnApp  36.84279 0.0000         16         15
## 17                       Gender  26.91989 0.0062         17         14
##    RANKING_TOT
## 1            2
## 2            6
## 3            8
## 4            9
## 5           12
## 6           14
## 7           16
## 8           16
## 9           17
## 10          18
## 11          20
## 12          21
## 13          27
## 14          29
## 15          29
## 16          31
## 17          31
```

Ahora vamos a hacer una correlación entre ellos a ver si ambos métodos son fiables

```
cor(imp_final$IMP_RF,imp_final$IMP_IV)
```

```
## [1] 0.9104204
```

Los métodos nos ofrecen una fiabilidad en los resultados muy alta. Decidimos, por lo tanto, utilizar por su importancia, las 13 primeras variables para la modelización.

Las incluimos en la siguiente lista:

```
ind_corta <- c("Tenure","PreferredLoginDevice","CityTier","WarehouseToHome","Prefer
redPaymentMode","NumberOfDeviceRegistered","PreferedOrderCat","SatisfactionScore","
MaritalStatus","Complain","OrderAmountHikeFromlastYear","DaySinceLastOrder","Cashba
ckAmount")
```

Estas son las variables predictoras con las que vamos a trabajar finalmente

```
ind_corta
```

```
##  [1] "Tenure"                       "PreferredLoginDevice"
##  [3] "CityTier"                     "WarehouseToHome"
##  [5] "PreferredPaymentMode"         "NumberOfDeviceRegistered"
##  [7] "PreferedOrderCat"             "SatisfactionScore"
##  [9] "MaritalStatus"                "Complain"
## [11] "OrderAmountHikeFromlastYear" "DaySinceLastOrder"
## [13] "CashbackAmount"
```

4.3.2.2 - Seleccionar la lista de variables finales del proyecto

Una vez que ya hemos identificado las variables importantes tenemos que volver a meter las de identificacion de clientes ‘CustomerID’ y la variable Target ‘Churn’.

```
lista <- ind_corta
lista
```

```
##  [1] "Tenure"                       "PreferredLoginDevice"
##  [3] "CityTier"                     "WarehouseToHome"
##  [5] "PreferredPaymentMode"         "NumberOfDeviceRegistered"
##  [7] "PreferedOrderCat"             "SatisfactionScore"
##  [9] "MaritalStatus"                "Complain"
## [11] "OrderAmountHikeFromlastYear" "DaySinceLastOrder"
## [13] "CashbackAmount"
```

y ahora vamos a buscar en las variables iniciales las variable de esta lista y crear un nuevo dataframe, incluyendo la identificación de clientes y nuestra variable elegida como target, que es el abandono (‘churn’).

```
iniciales <- names(df)
patron <- paste(lista,collapse='|')
intermedias <- iniciales[grepl(patron,iniciales)]
finales <- union(intermedias,c('CustomerID','TARGET1'))
finales
```

```
##  [1] "Tenure"                       "PreferredLoginDevice"
##  [3] "CityTier"                     "WarehouseToHome"
##  [5] "PreferredPaymentMode"         "NumberOfDeviceRegistered"
##  [7] "PreferedOrderCat"             "SatisfactionScore"
##  [9] "MaritalStatus"                "Complain"
## [11] "OrderAmountHikeFromlastYear" "DaySinceLastOrder"
## [13] "CashbackAmount"              "CustomerID"
## [15] "TARGET1"
```

Vemos que ahora nuestro data frame tiene 15 variables.

- 4.3.3 - Fichero final y limpieza del entorno

4.3.3.1 - Fichero final

```
dim(df)
```

```
## [1] 5626   19
```

```
df <- df %>%
  select(one_of(finales))
dim(df)
```

```
## [1] 5626   15
```

Nuestro data frame tiene ahora 5626 filas y 15 variables.

4.3.3.2 - Limpieza del entorno

Realizamos una limpieza de ficheros temporales que hemos creado, así como crear la variable TARGET1 con la que vamos a trabajar.

```
ls()
```

```
##  [1] "a_factores"   "contar_ceros" "df"           "finales"      "imp_final"
##  [6] "imp_iv"       "imp_rf"       "ind_corta"    "ind_larga"    "iniciales"
## [11] "instalados"   "intermedias"  "lista"        "no_usar"      "num_ceros"
## [16] "out"          "paquetes"     "patron"       "pre_rf"       "temp"
```

```
rm(list=setdiff(ls(),'df'))
target <- 'TARGET1'
indep <- setdiff(names(df),c(target,'customerID'))
```

Vamos a guardar una copia temporal del data frame con los cambios realizados hasta este momento.

```
saveRDS(df,'cache1.rds')
```

4.4 - Creación de variables sintéticas

Como no tenemos datos historicos no podemos realizar la creación de variables sintéticas de tenencia, contratación, cancelación, medias y tendencias.

- 4.4.1 - Discretización

Mediante una función vamos a discretizar las variables numéricas de forma automática con la regresión logística, para que la discretización sea monotónica.

```
discretizar <- function(vi,target){
  temp_df <- data.frame(vi = vi, target = target)
  temp_df$target <- as.numeric(as.character(temp_df$target))
  disc <- smbinning(temp_df, y = 'target', x = 'vi')
  return(disc)
}
```

Aplicamos la función a las variables susceptibles de discretizar:

```
#Tenure:
disc_temp_Tenure <- discretizar(df$Tenure,df$TARGET1)
df_temp <- select(df,Tenure,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_Tenure,chrname = 'TENURE_DISC')
df <- cbind(df,df_temp[3]) %>% select(-Tenure)

#WarehouseToHome:
disc_temp_WarehouseToHome <- discretizar(df$WarehouseToHome,df$TARGET1)
df_temp <- select(df,WarehouseToHome,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_WarehouseToHome,chrname = 'WAREHOUSETOHO
ME_DISC')
df <- cbind(df,df_temp[3]) %>% select(-WarehouseToHome)

#NumberOfDeviceRegistered:
disc_temp_NumberOfDeviceRegistered <- discretizar(df$NumberOfDeviceRegistered,df$TA
RGET1)
df_temp <- select(df,NumberOfDeviceRegistered,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_NumberOfDeviceRegistered,chrname = 'NUMB
EROFDEVICEREGISTERED_DISC')
df <- cbind(df,df_temp[3]) %>% select(-NumberOfDeviceRegistered)

#OrderAmountHikeFromlastYear:
disc_temp_OrderAmountHikeFromlastYear <- discretizar(df$OrderAmountHikeFromlastYea
r,df$TARGET1)
df_temp <- select(df,OrderAmountHikeFromlastYear,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_OrderAmountHikeFromlastYear,chrname = 'O
RDERAMOUNTHIKEFROMLASTYEAR_DISC')
df <- cbind(df,df_temp[3]) %>% select(-OrderAmountHikeFromlastYear)

#DaySinceLastOrder:
disc_temp_DaySinceLastOrder <- discretizar(df$DaySinceLastOrder,df$TARGET1)
df_temp <- select(df,DaySinceLastOrder,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_DaySinceLastOrder,chrname = 'DAYSINCELAS
TORDER_DISC')
df <- cbind(df,df_temp[3]) %>% select(-DaySinceLastOrder)

#CashbackAmount:
disc_temp_CashbackAmount <- discretizar(df$CashbackAmount,df$TARGET1)
df_temp <- select(df,CashbackAmount,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_CashbackAmount,chrname = 'CASHBACKAMOUNT
_DISC')
df <- cbind(df,df_temp[3]) %>% select(-CashbackAmount)
```

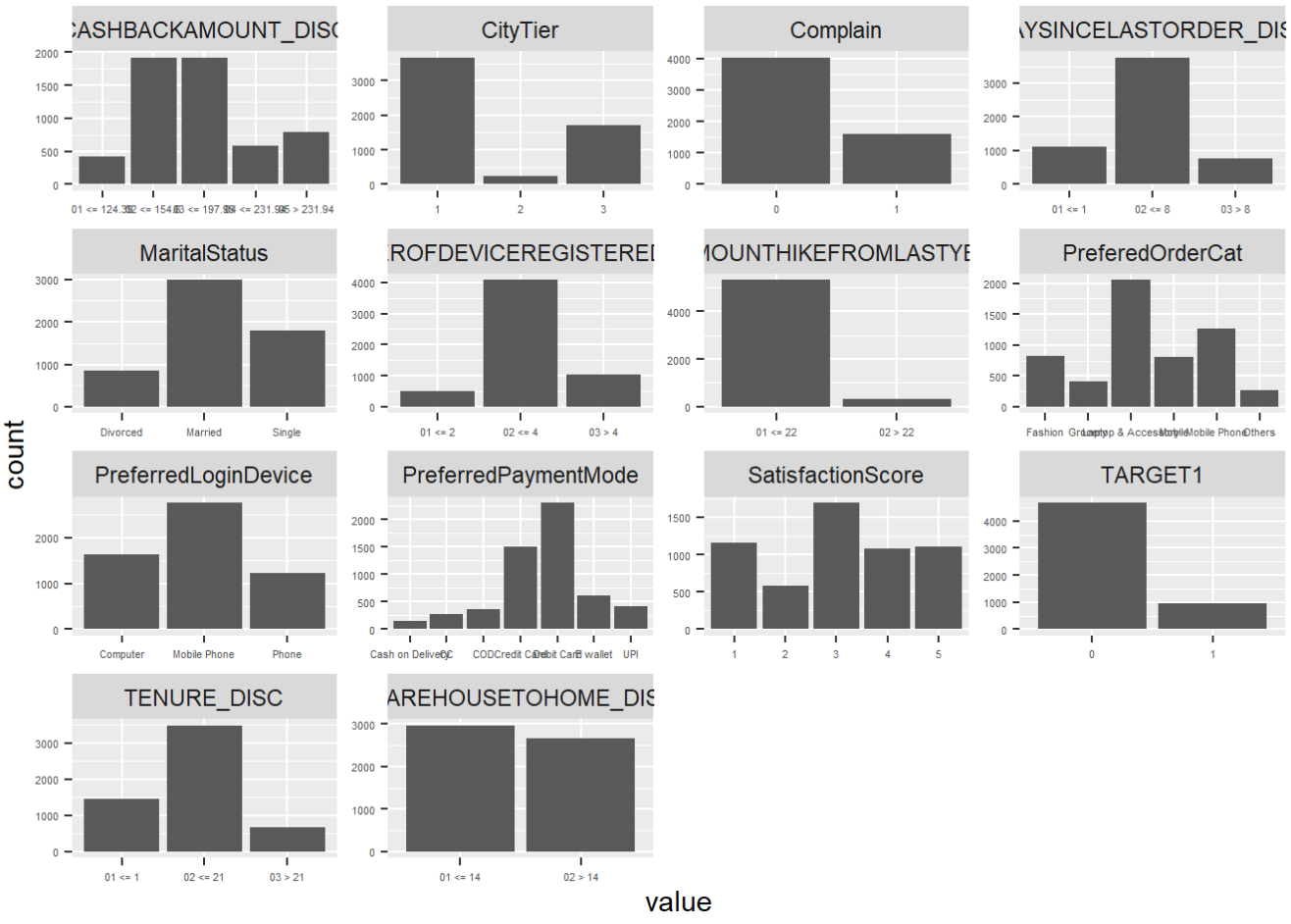
Obtebemos un resumen para ver las variables discretizadas que hemos obtenido.

```
glimpse(df)
```

```
## Rows: 5,626
## Columns: 15
## $ PreferredLoginDevice             <fct> Mobile Phone, Phone, Phone, Phone, Ph~
## $ CityTier                         <fct> 3, 1, 1, 3, 1, 1, 3, 1, 3, 1, 1, 1, 1~
## $ PreferredPaymentMode             <fct> Debit Card, UPI, Debit Card, Debit Ca~
## $ PreferedOrderCat                 <fct> Laptop & Accessory, Mobile, Mobile, L~
## $ SatisfactionScore                <fct> 2, 3, 3, 5, 5, 5, 2, 2, 3, 3, 3, 3, 3~
## $ MaritalStatus                    <fct> Single, Single, Single, Single, Singl~
## $ Complain                         <fct> 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1~
## $ CustomerID                       <dbl> 50001, 50002, 50003, 50004, 50005, 50~
## $ TARGET1                          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ TENURE_DISC                      <fct> 02 <= 21, 01 <= 1, 01 <= 1, 01 <= 1, ~
## $ WAREHOUSETOHOME_DISC             <fct> 01 <= 14, 01 <= 14, 02 > 14, 02 > 14,~
## $ NUMBEROFDEVICEREGISTERED_DISC    <fct> 02 <= 4, 02 <= 4, 02 <= 4, 02 <= 4, 0~
## $ ORDERAMOUNTHIKEFROMLASTYEAR_DISC <fct> 01 <= 22, 01 <= 22, 01 <= 22, 02 > 22~
## $ DAYSINCELASTORDER_DISC           <fct> 02 <= 8, 01 <= 1, 02 <= 8, 02 <= 8, 0~
## $ CASHBACKAMOUNT_DISC              <fct> 03 <= 197.95, 01 <= 124.39, 01 <= 124~
```

Vamos a hacer una inspección visual de todas las variables a ver si han salido bien.

```
df %>%
  select_if(is.factor) %>%
  gather() %>%
  ggplot(aes(value)) +
    geom_bar() +
    facet_wrap(~ key, scales = "free") +
    theme(axis.text=element_text(size=4))
```



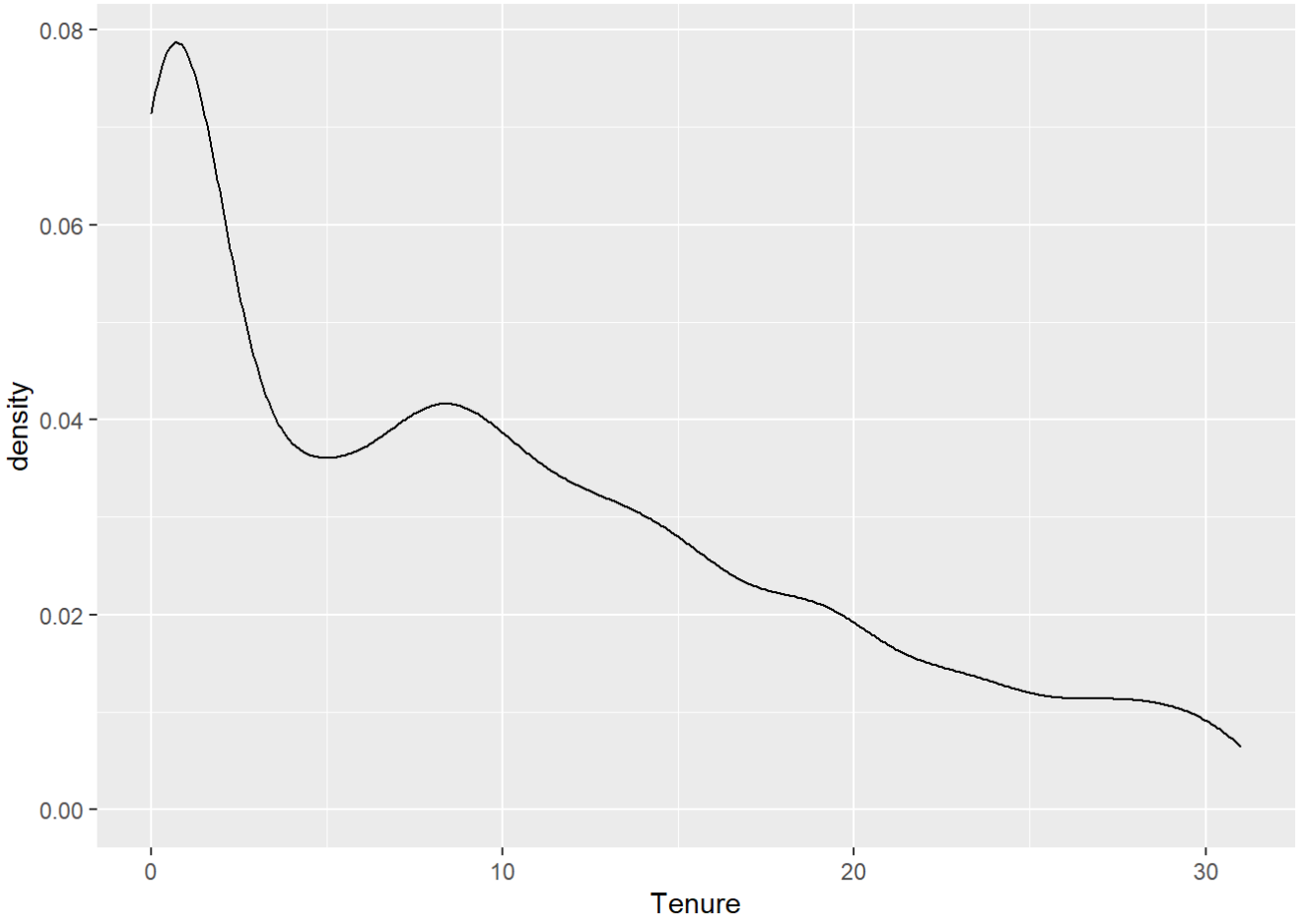
Vemos que las variables mas significativas desde el punto de vista de negocio, como son “Tenure” y “CahsbackAmount”, no siguen una tendencia monotónicas después de la discretización automática por lo tanto vamos a discretizarlas manualmente.

Volvemos a cargar los datos ya revisada su calidad.

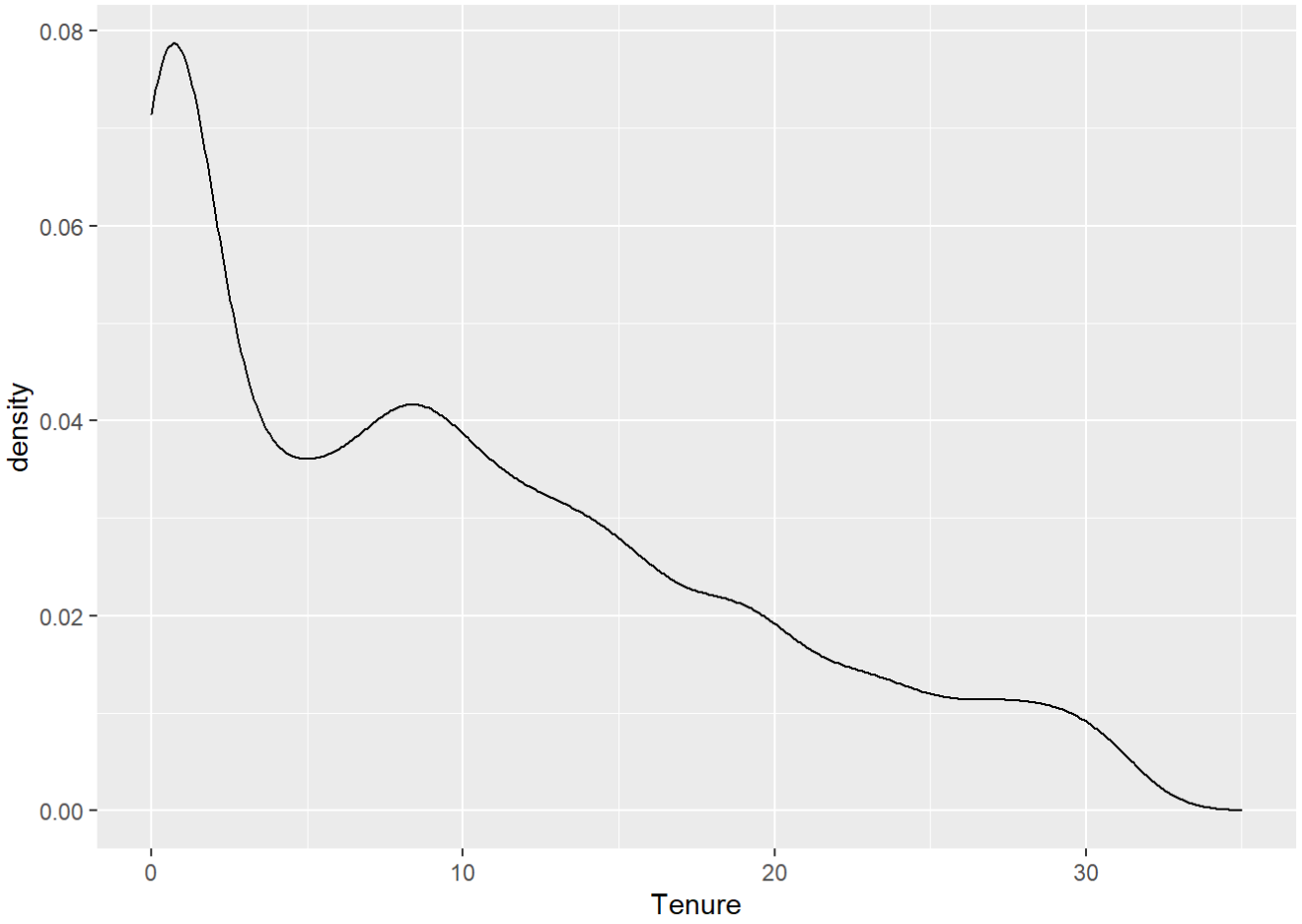
```
df <- readRDS(file = 'cache1.rds')
```

Empezamos con ‘Tenure’. Lo primero es ver que distribucion tiene la variable target con la variable a discretizar.

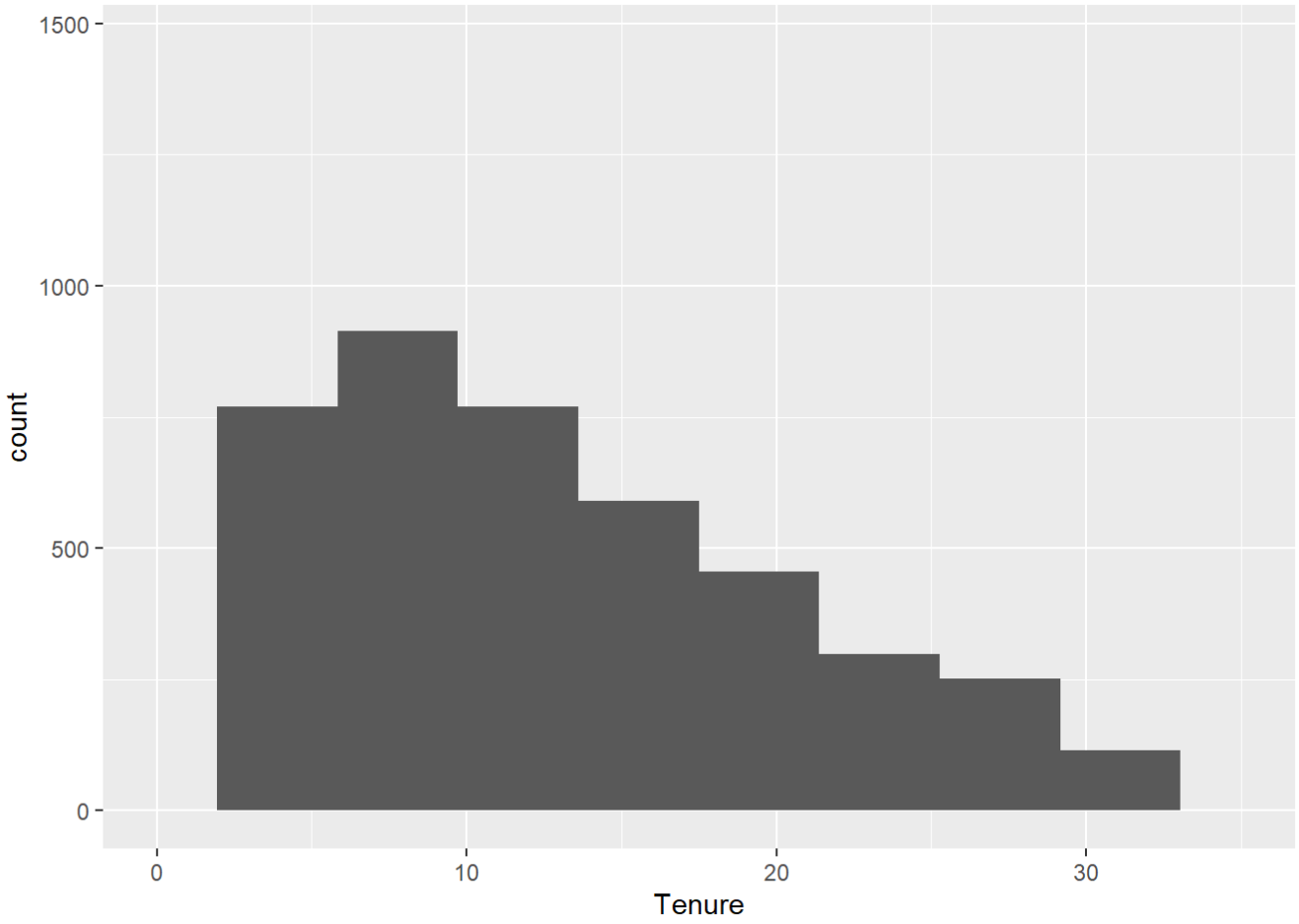
```
ggplot(df,aes(Tenure)) + geom_density()
```



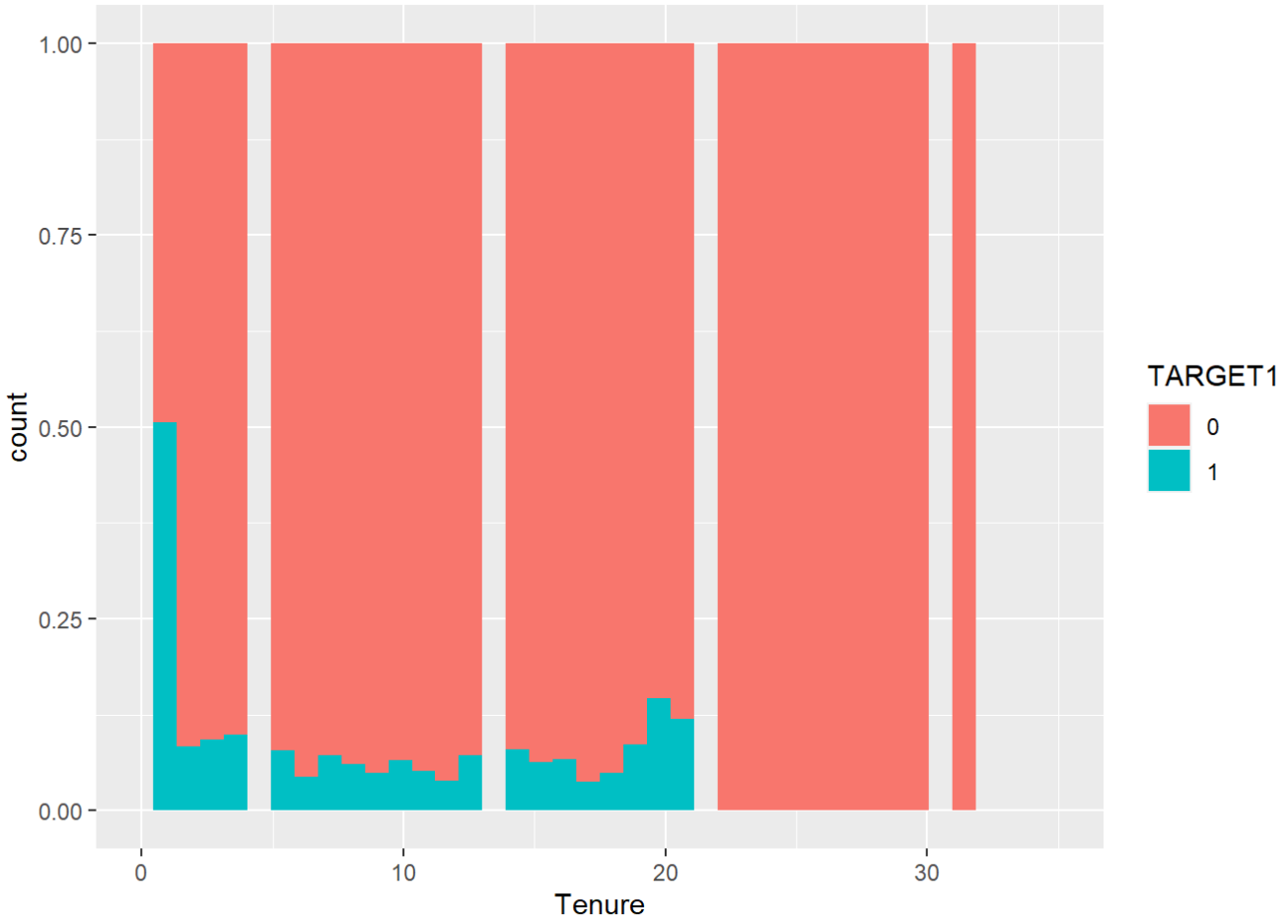

```
# Vamos a limitar el eje x para obtener la gráfica completa.
ggplot(df,aes(Tenure)) + geom_density() + scale_x_continuous(limits = c(0, 35))
```




```
# Hacemos un histograma para aproximar mejor la forma que queremos conseguir
ggplot(df,aes(Tenure)) + geom_histogram(bins = 10) + scale_x_continuous(limits = c
(0, 35))
```




```
# Ya sabemos que queremos una forma decreciente ahora veamos como se comporta la va
riable target para ver si podremos generar un perfil monotónico
ggplot(df,aes(Tenure,fill=TARGET1)) + geom_histogram(bins = 40,position='fill') + s
cale_x_continuous(limits = c(0, 35))
```



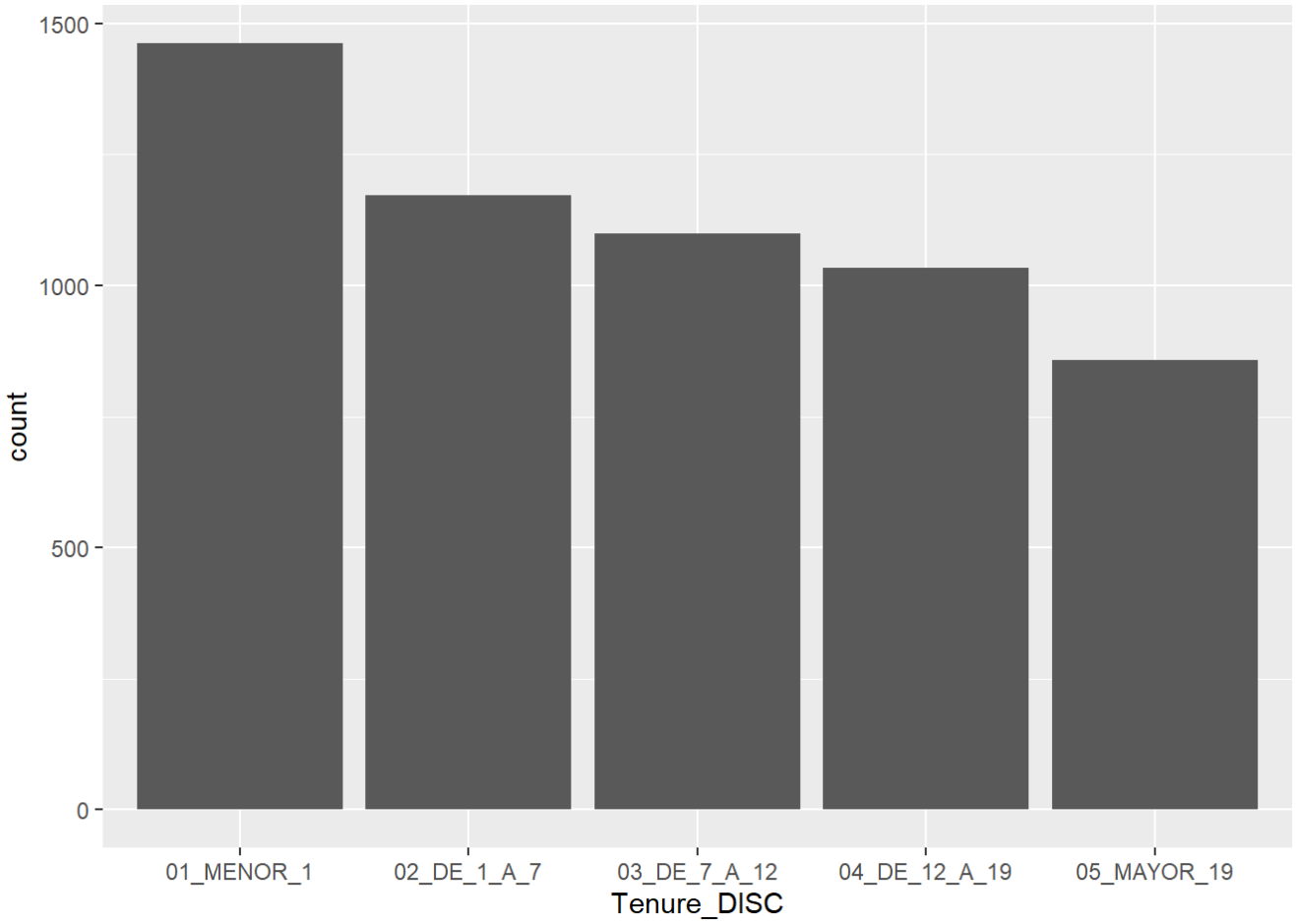

```
# Sabiendo ambas cosas vamos a apoyarnos en los deciles para intuir donde podemos h
acer buenos cortes
as.data.frame(quantile(df$Tenure,prob = seq(0, 1, length = 11)))
```

```
##      quantile(df$Tenure, prob = seq(0, 1, length = 11))
## 0%                                                    0
## 10%                                                   0
## 20%                                                   1
## 30%                                                   3
## 40%                                                   6
## 50%                                                   8
## 60%                                                  11
## 70%                                                  14
## 80%                                                  17
## 90%                                                  23
## 100%                                                 31
```

```
df <- df %>% mutate(Tenure_DISC = as.factor(case_when(
      Tenure <= 1 ~ '01_MENOR_1',
      Tenure > 1 & Tenure <= 7 ~ '02_DE_1_A_7',
      Tenure > 7 & Tenure <= 12 ~ '03_DE_7_A_12',
      Tenure > 12 & Tenure <= 19 ~ '04_DE_12_A_19',
      Tenure > 19 ~ '05_MAYOR_19',
      TRUE ~ '00_ERROR'))
)
```

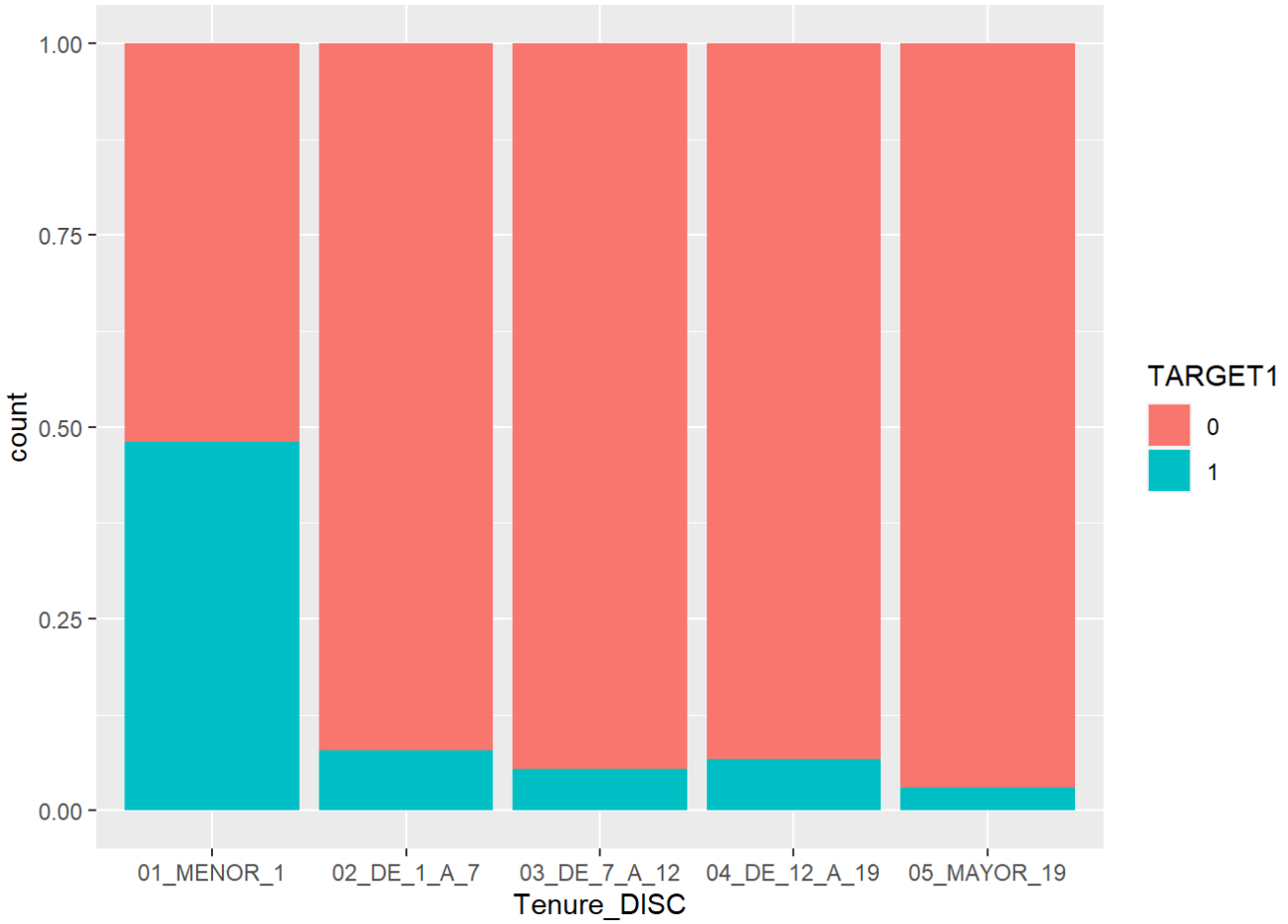
Veamos si la distribución ha quedado similar a la original.

```
ggplot(df,aes(Tenure_DISC)) + geom_bar()
```



Y ahora vamos a comprobar si la penetración de la target es monotónica.

```
ggplot(df,aes(Tenure_DISC,fill=TARGET1)) + geom_bar(position='fill')
```

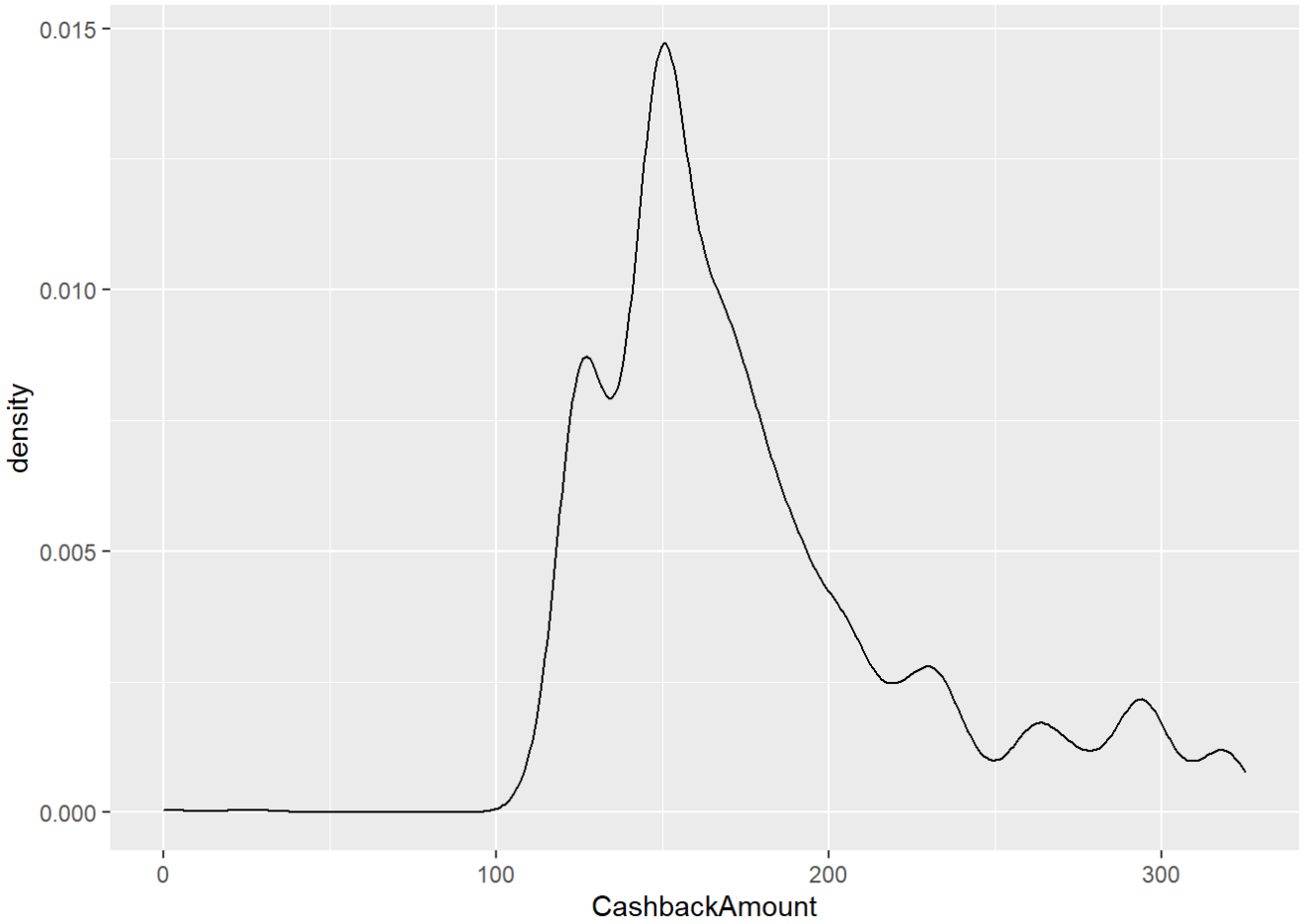


Vemos como a menor permanencia la posibilidad de abandon es mayor, pero cuando la permanencia es mayor, vemos como la probabilidad de abandono es mucho menor.

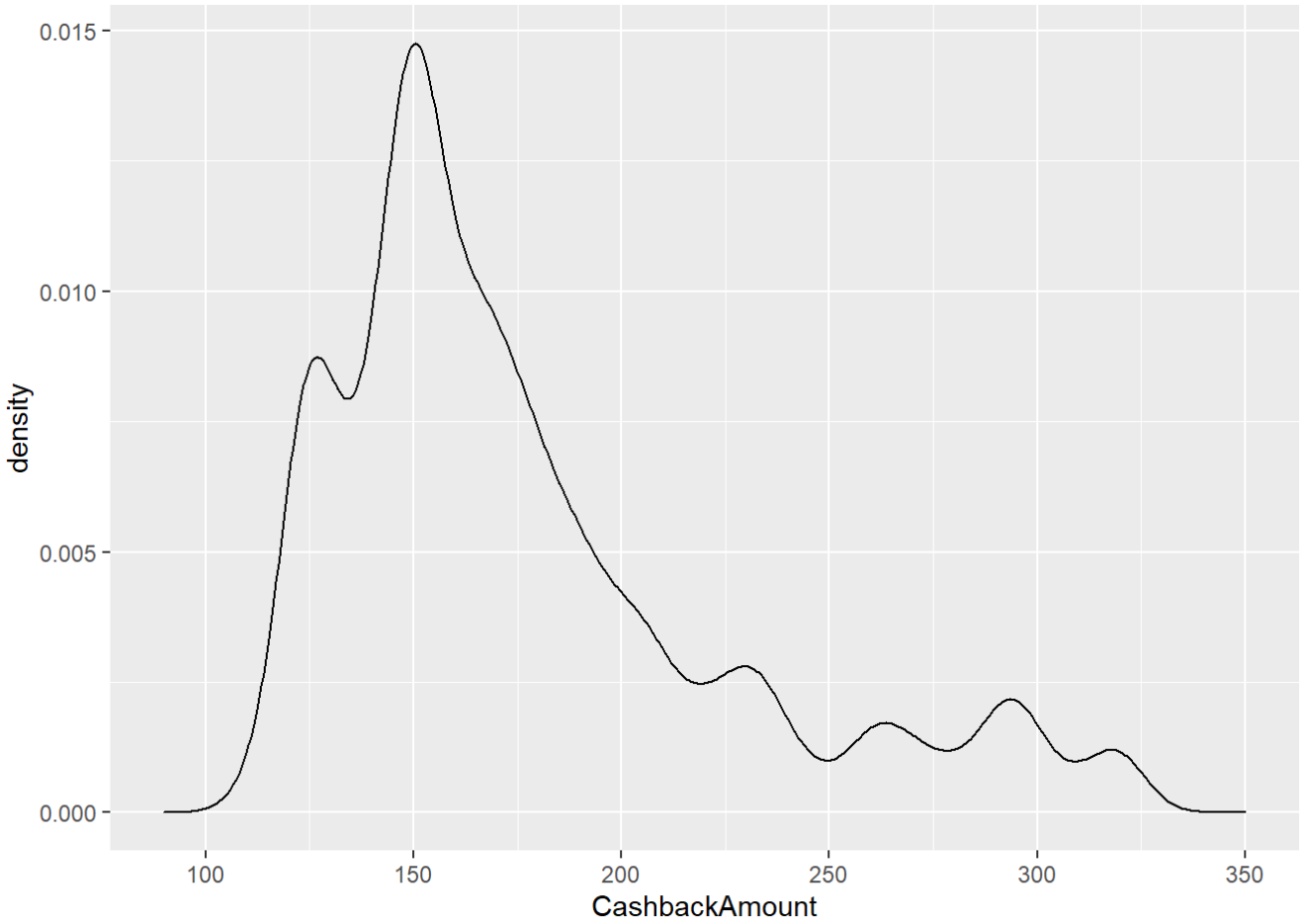
Continuamos con 'CashbackAmount'.

Lo primero es ver que distribucion tiene la variable.

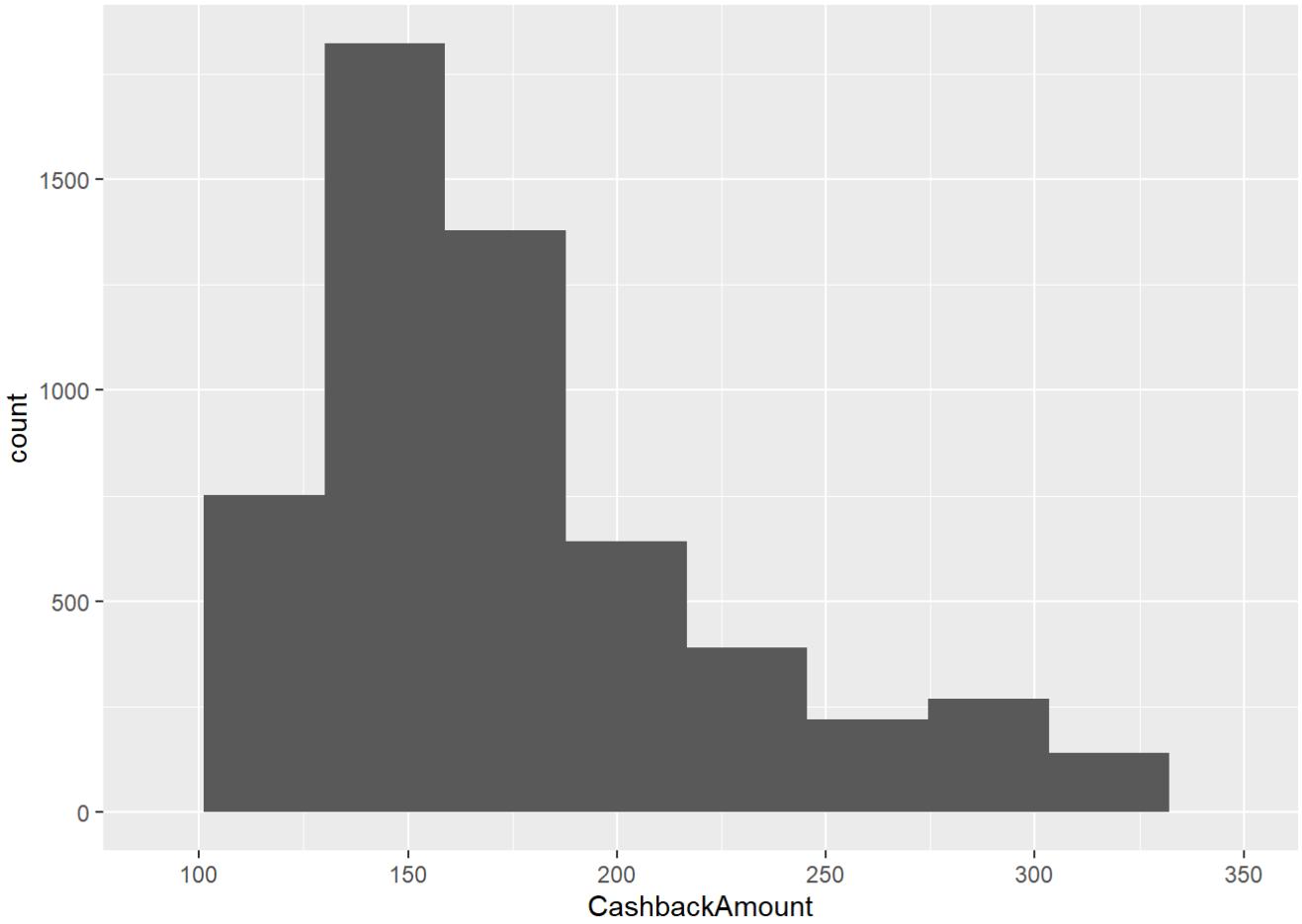
```
ggplot(df,aes(CashbackAmount)) + geom_density()
```



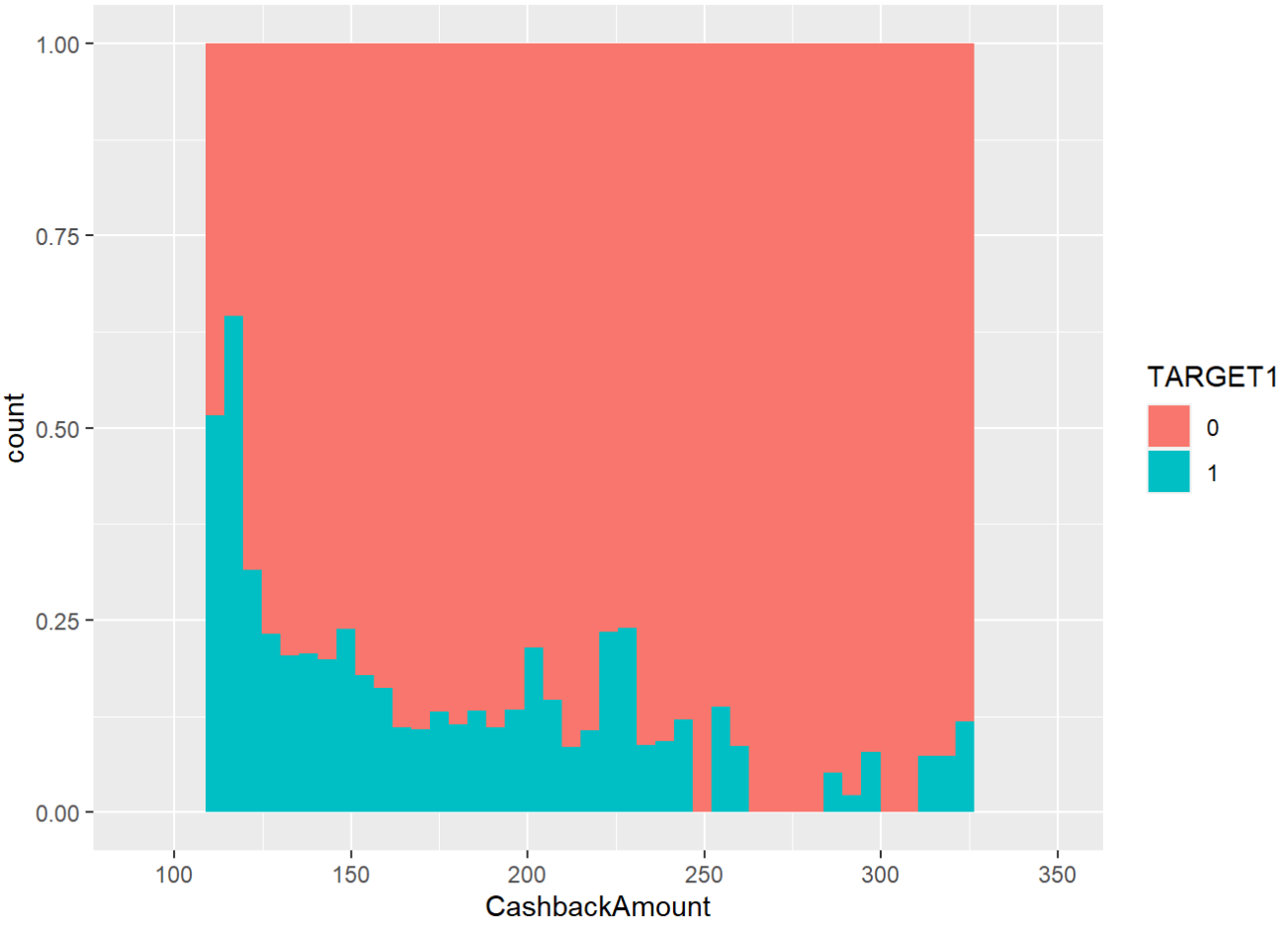

```
# Vamos a limitar el eje x
ggplot(df,aes(CashbackAmount)) + geom_density() + scale_x_continuous(limits = c(90,
350))
```




```
# Pedimos un histograma para aproximar mejor la forma que queremos conseguir
ggplot(df,aes(CashbackAmount)) + geom_histogram(bins = 10) + scale_x_continuous(lim
its = c(90, 350))
```




```
# Ya sabemos que queremos una forma decreciente, ahora veamos como se comporta la v
ariable target para ver si podremos generar un perfil monotónico.
ggplot(df,aes(CashbackAmount,fill=TARGET1)) + geom_histogram(bins = 50,position='fi
ll') + scale_x_continuous(limits = c(90, 350))
```




```
#Sabiendo ambas cosas vamos a apoyarnos en los deciles para intuir donde podemos ha
cer buenos cortes
as.data.frame(quantile(df$CashbackAmount,prob = seq(0, 1, length = 11)))
```

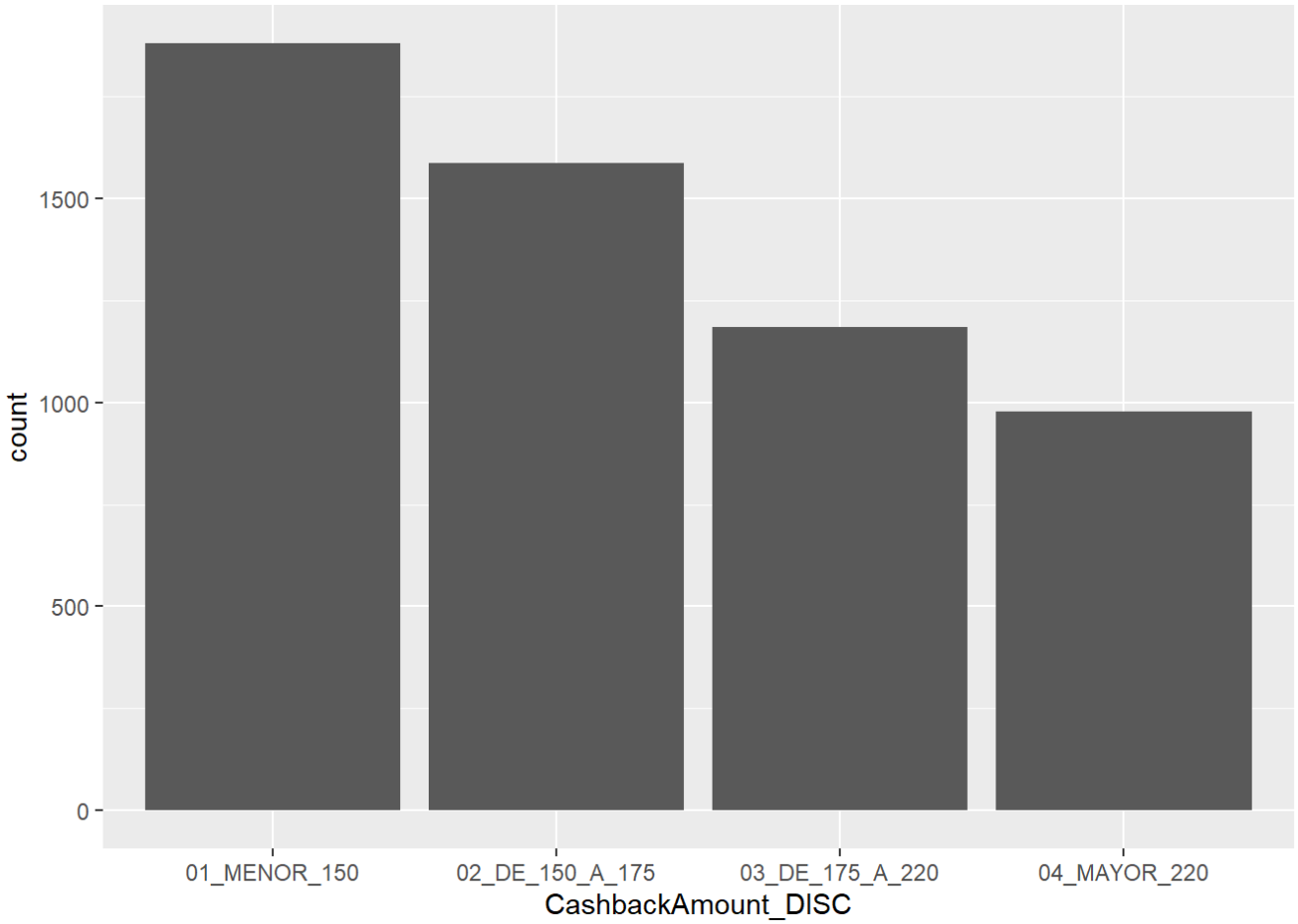
```
##      quantile(df$CashbackAmount, prob = seq(0, 1, length = 11))
## 0%                                                         0.00
## 10%                                                      126.86
## 20%                                                      140.78
## 30%                                                      148.32
## 40%                                                      153.63
## 50%                                                      163.23
## 60%                                                      173.05
## 70%                                                      187.05
## 80%                                                      208.81
## 90%                                                      259.64
## 100%                                                     324.99
```

Procedemos a discretizarla manualmente

```
df <- df %>% mutate(CashbackAmount_DISC = as.factor(case_when(
      CashbackAmount <= 150 ~ '01_MENOR_150',
      CashbackAmount > 150 & CashbackAmount <= 175 ~ '02_DE_150_A_175',
      CashbackAmount > 175 & CashbackAmount <= 220 ~ '03_DE_175_A_220',
      CashbackAmount > 220 ~ '04_MAYOR_220',
      TRUE ~ '00_ERROR'))
)
```

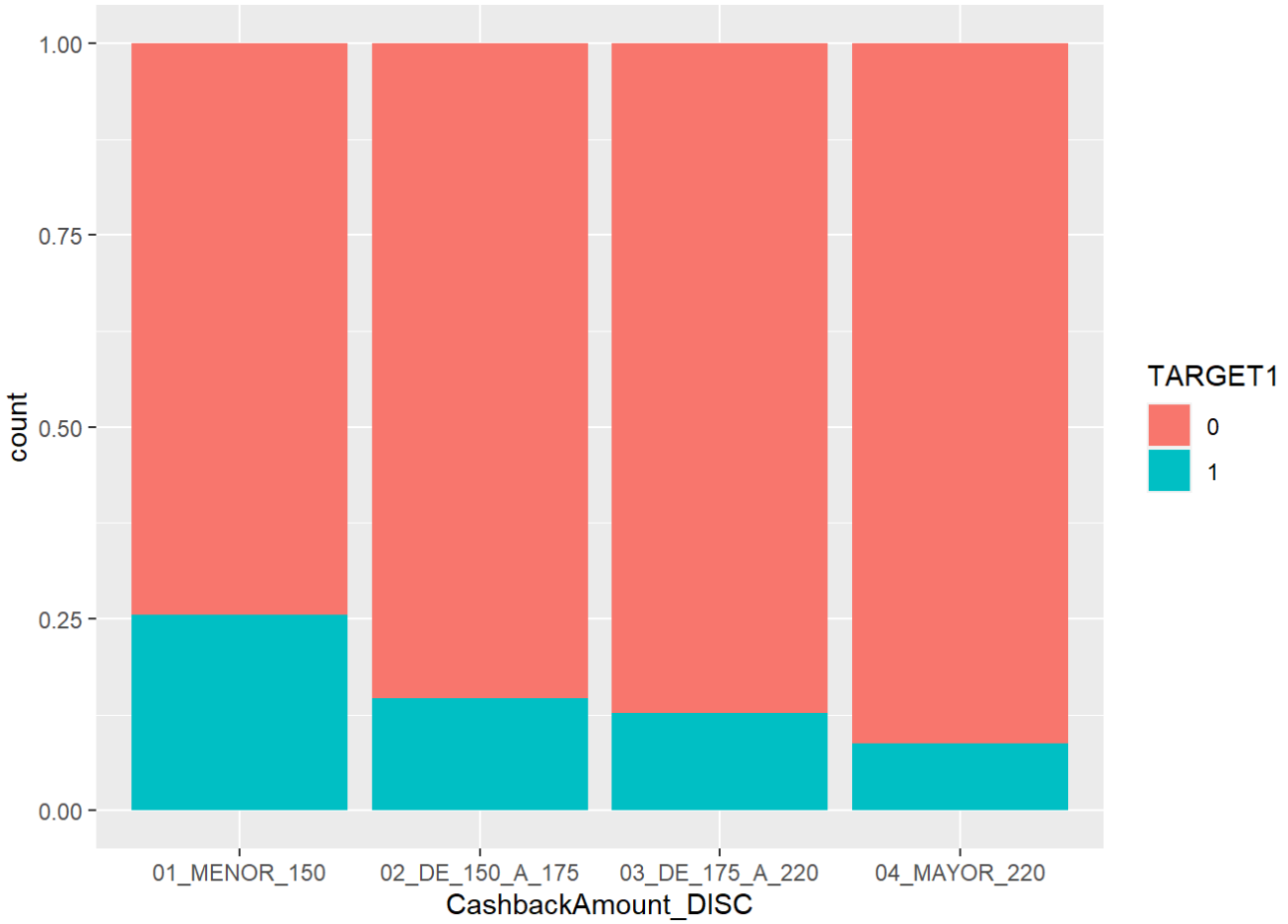
Vemos que la distribución ha quedado similar a la original.

```
ggplot(df,aes(CashbackAmount_DISC)) + geom_bar()
```



Y ahora vamos a comprobar si la penetración de la target es monotónica.

```
ggplot(df,aes(CashbackAmount_DISC,fill=TARGET1)) + geom_bar(position='fill')
```



Vemos como los clientes con cantidad de compra menor tiene mayor posibilidad de abandono.

Eliminamos las variables originales de las que hemos discretizado manualmente.

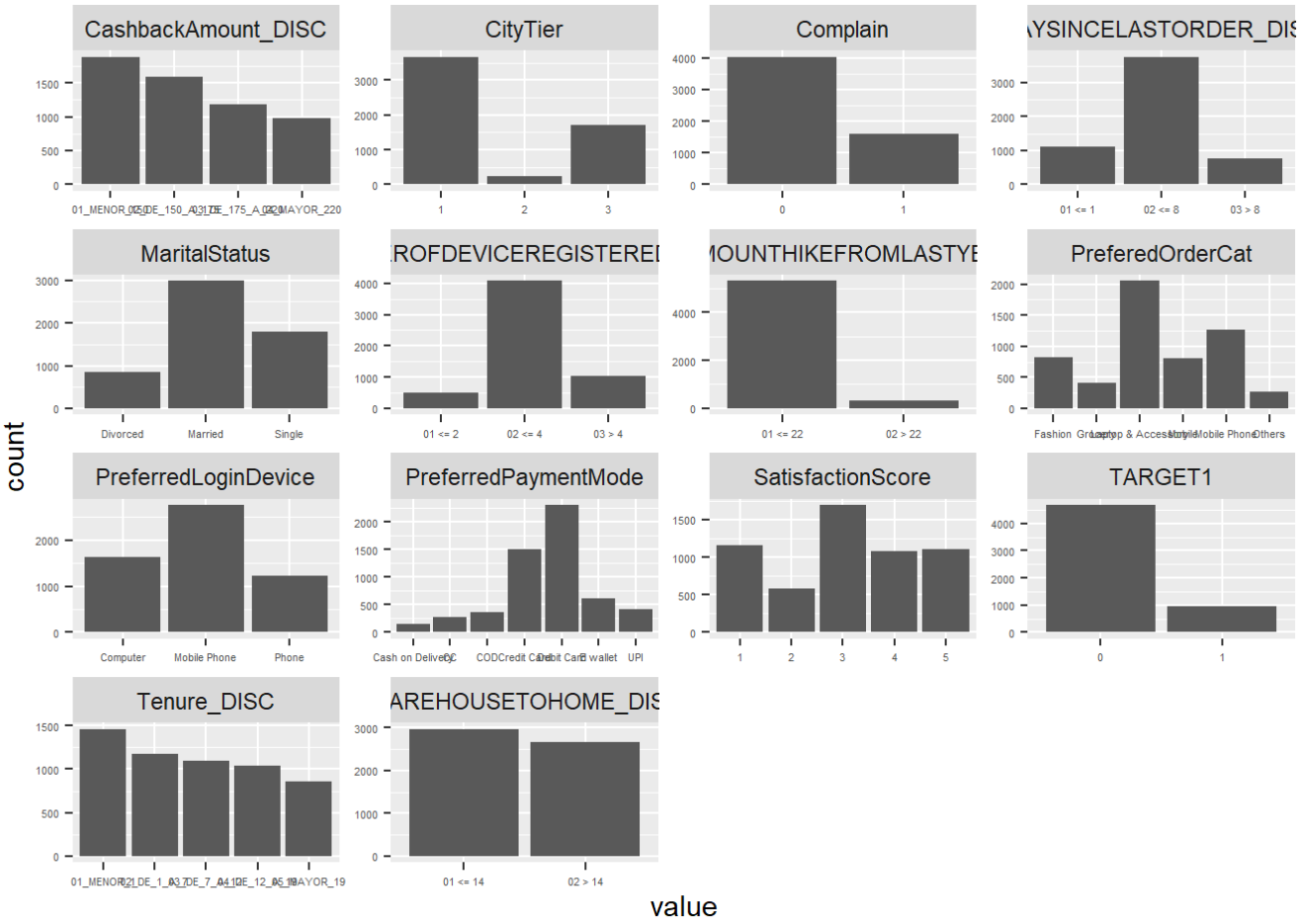
```
df <- select(df,-Tenure)
df <- select(df,-CashbackAmount)
```

Discretizamos automáticamente las otras variables

```
#WarehouseToHome:
disc_temp_WarehouseToHome <- discretizar(df$WarehouseToHome,df$TARGET1)
df_temp <- select(df,WarehouseToHome,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_WarehouseToHome,chrname = 'WAREHOUSETOHO
ME_DISC')
df <- cbind(df,df_temp[3]) %>% select(-WarehouseToHome)

#NumberOfDeviceRegistered:
disc_temp_NumberOfDeviceRegistered <- discretizar(df$NumberOfDeviceRegistered,df$TA
RGET1)
df_temp <- select(df,NumberOfDeviceRegistered,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_NumberOfDeviceRegistered,chrname = 'NUMB
EROFDEVICEREGISTERED_DISC')
df <- cbind(df,df_temp[3]) %>% select(-NumberOfDeviceRegistered)

#OrderAmountHikeFromlastYear:
disc_temp_OrderAmountHikeFromlastYear <- discretizar(df$OrderAmountHikeFromlastYea
r,df$TARGET1)
df_temp <- select(df,OrderAmountHikeFromlastYear,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_OrderAmountHikeFromlastYear,chrname = 'O
RDERAMOUNTHIKEFROMLASTYEAR_DISC')
df <- cbind(df,df_temp[3]) %>% select(-OrderAmountHikeFromlastYear)

#DaySinceLastOrder:
disc_temp_DaySinceLastOrder <- discretizar(df$DaySinceLastOrder,df$TARGET1)
df_temp <- select(df,DaySinceLastOrder,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_DaySinceLastOrder,chrname = 'DAYSINCELAS
TORDER_DISC')
df <- cbind(df,df_temp[3]) %>% select(-DaySinceLastOrder)
```

Vamos a hacer una inspección visual de todas las variables a ver si han salido bien

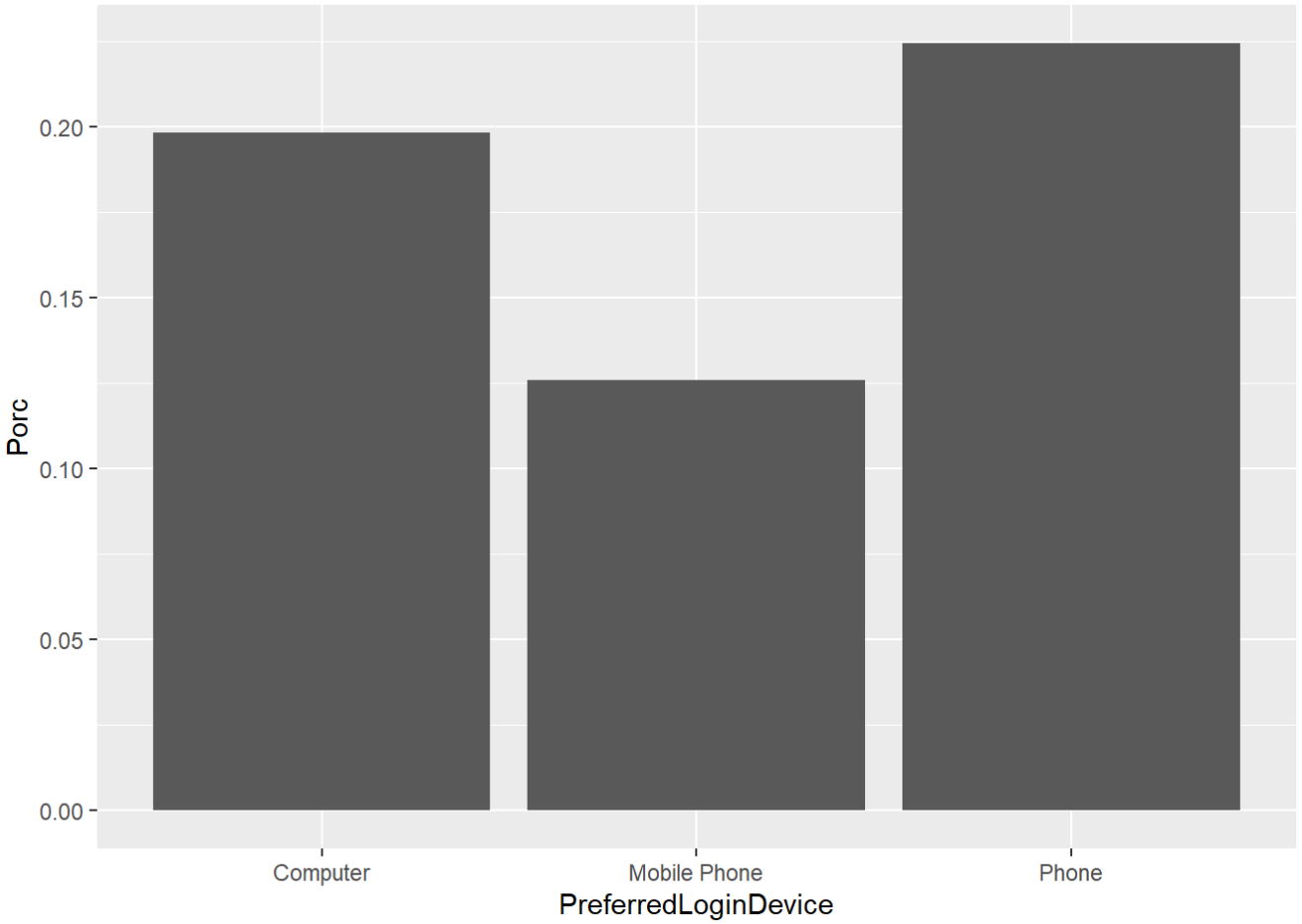
```
df %>%
  select_if(is.factor) %>%
  gather() %>%
  ggplot(aes(value)) +
    geom_bar() +
    facet_wrap(~ key, scales = "free") +
    theme(axis.text=element_text(size=4))
```



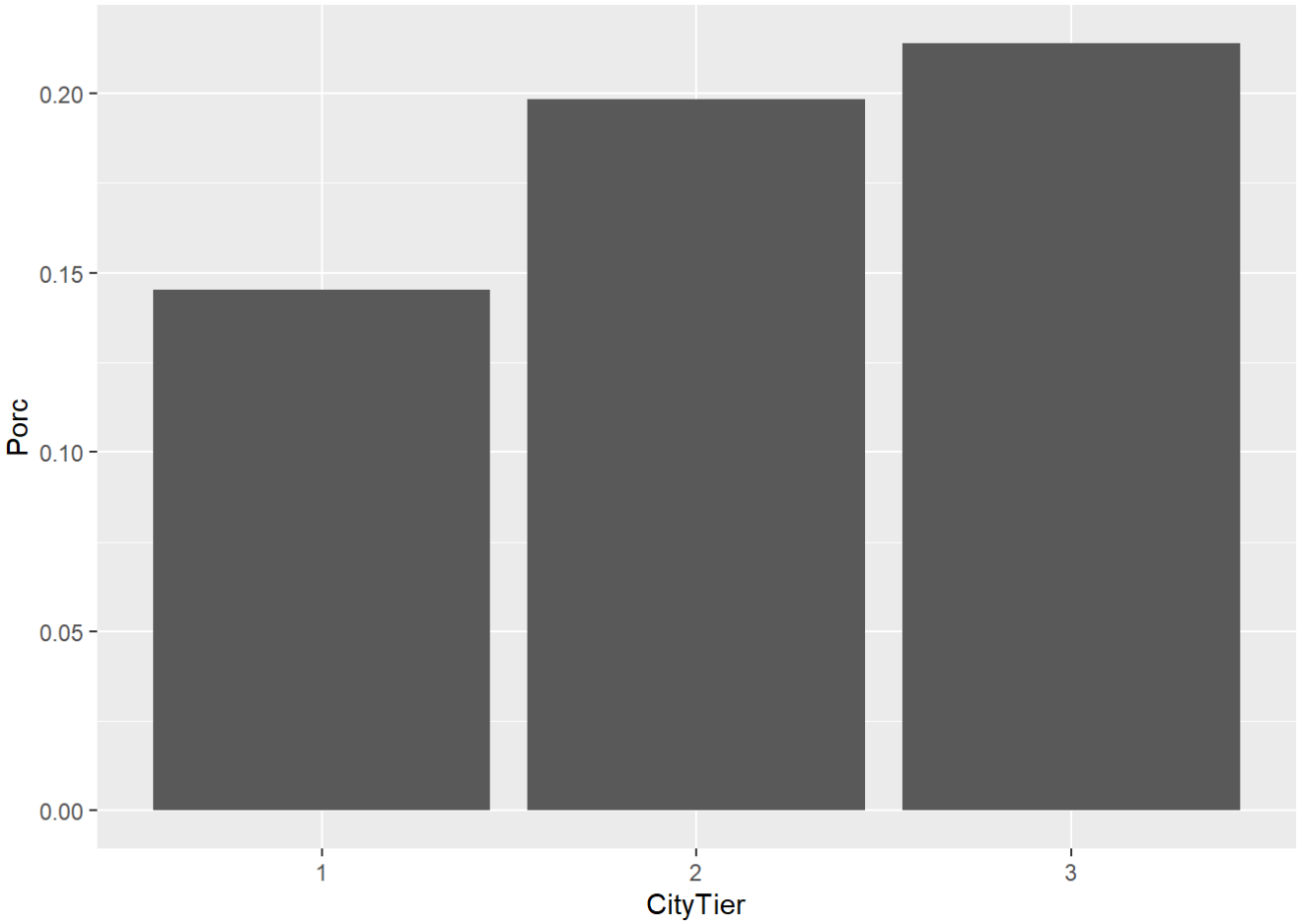
Ahora vamos a analizar la penetración de la target en cada categoría para ver si las variables han salido monotónicas

```
a <- function(var1,var2) {
  df_temp <- data.frame(var1 = df[[var1]],var2 = df[[var2]])
  df_temp %>%
    group_by(var1) %>%
    summarise(Conteo = n(), Porc = mean(as.numeric(as.character(var2)))) %>%
  ggplot(aes(var1,Porc)) + geom_bar(stat='identity') + xlab(var1)
}
df2_nombres <- df %>% select_if(is.factor) %>% names()
lapply(df2_nombres,function(x){a(x,'TARGET1')})
```

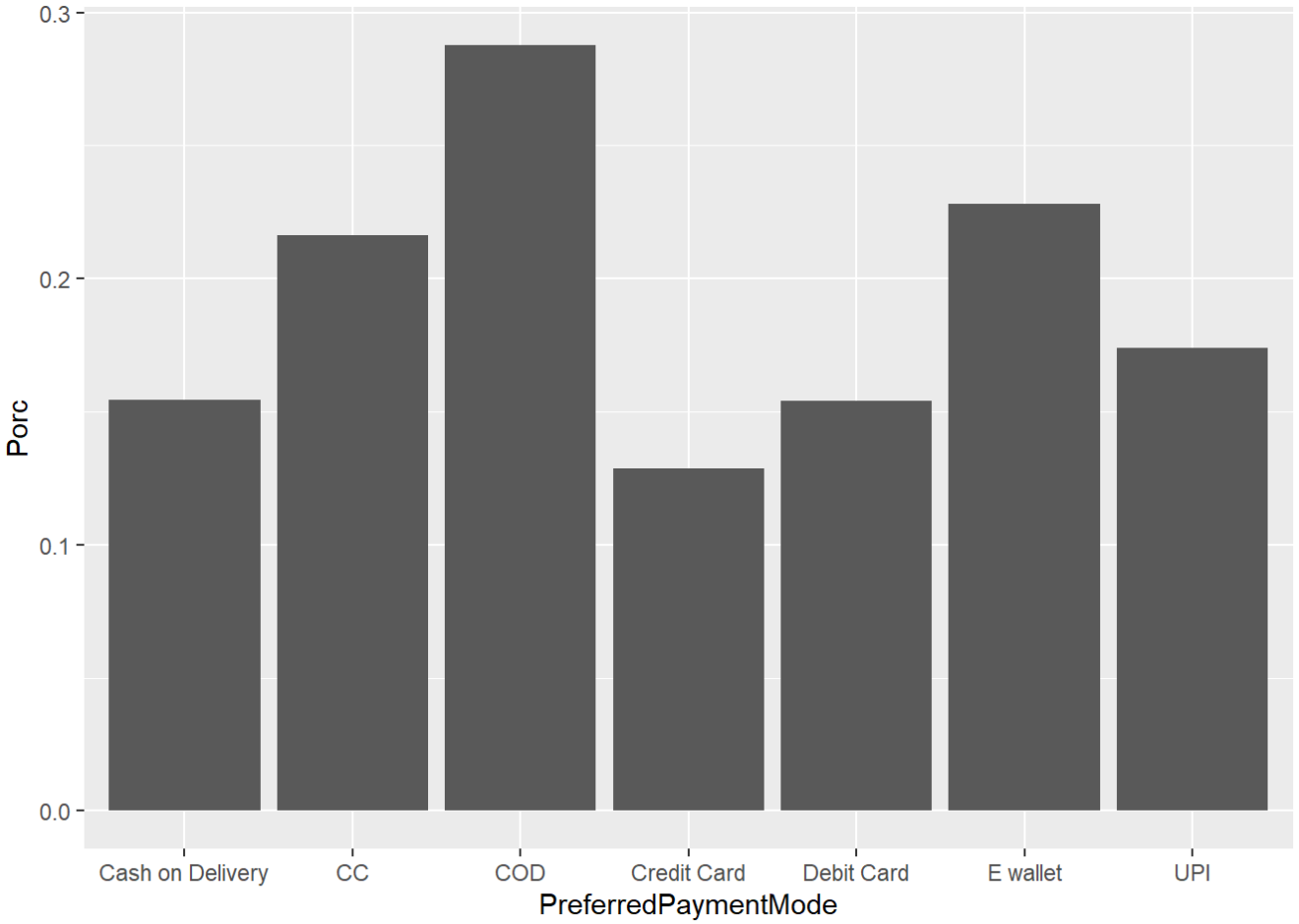
```
## [[1]]
```


0.20
0.15
0.10
0.05
0.00
Porc
Computer
Mobile Phone
Phone
PreferredLoginDevice

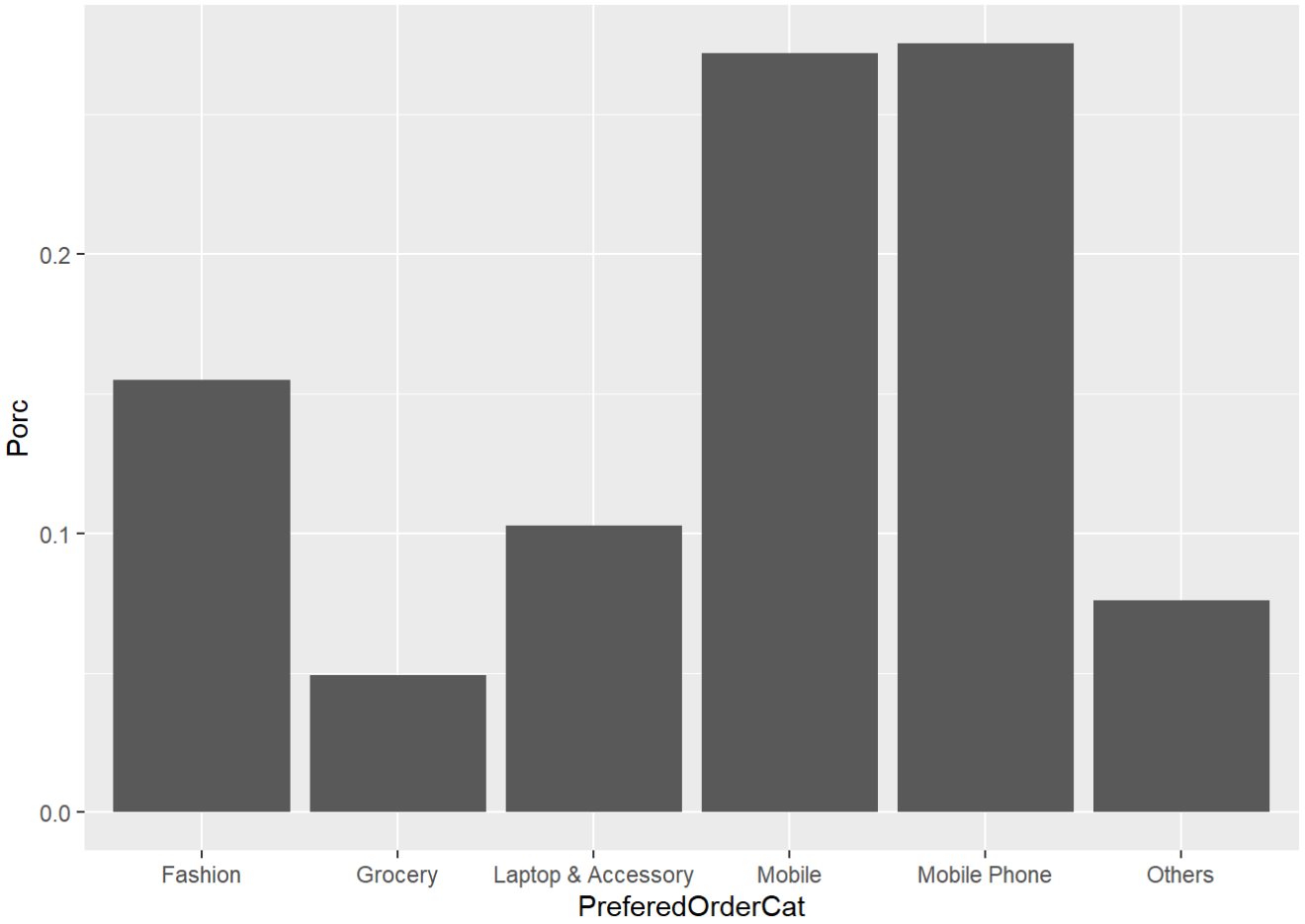
```
##
## [[2]]
```



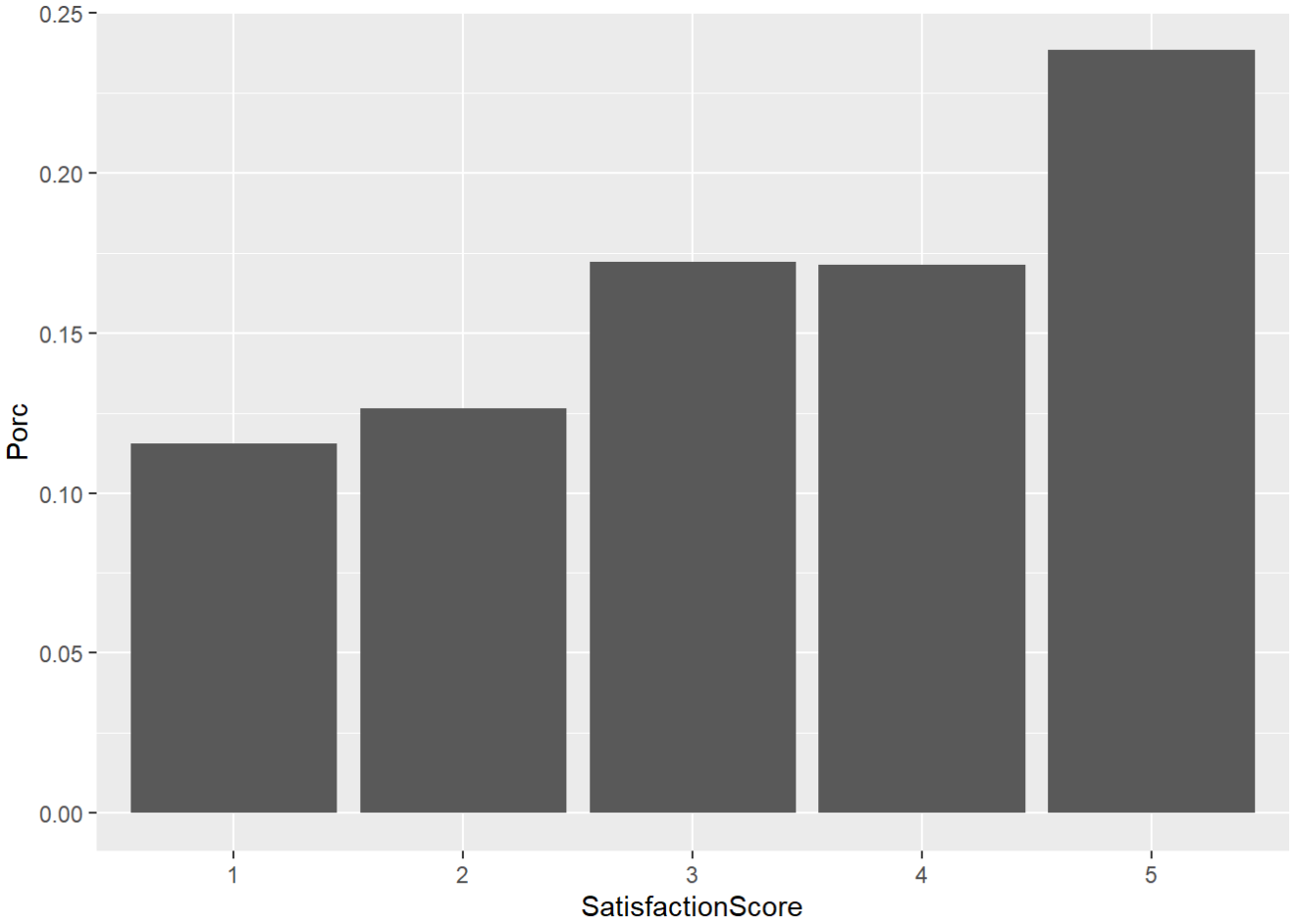

```
##
## [[3]]
```


0.3
0.2
0.1
0.0
Porc
Cash on Delivery
CC
COD
Credit Card
Debit Card
E wallet
UPI
PreferredPaymentMode

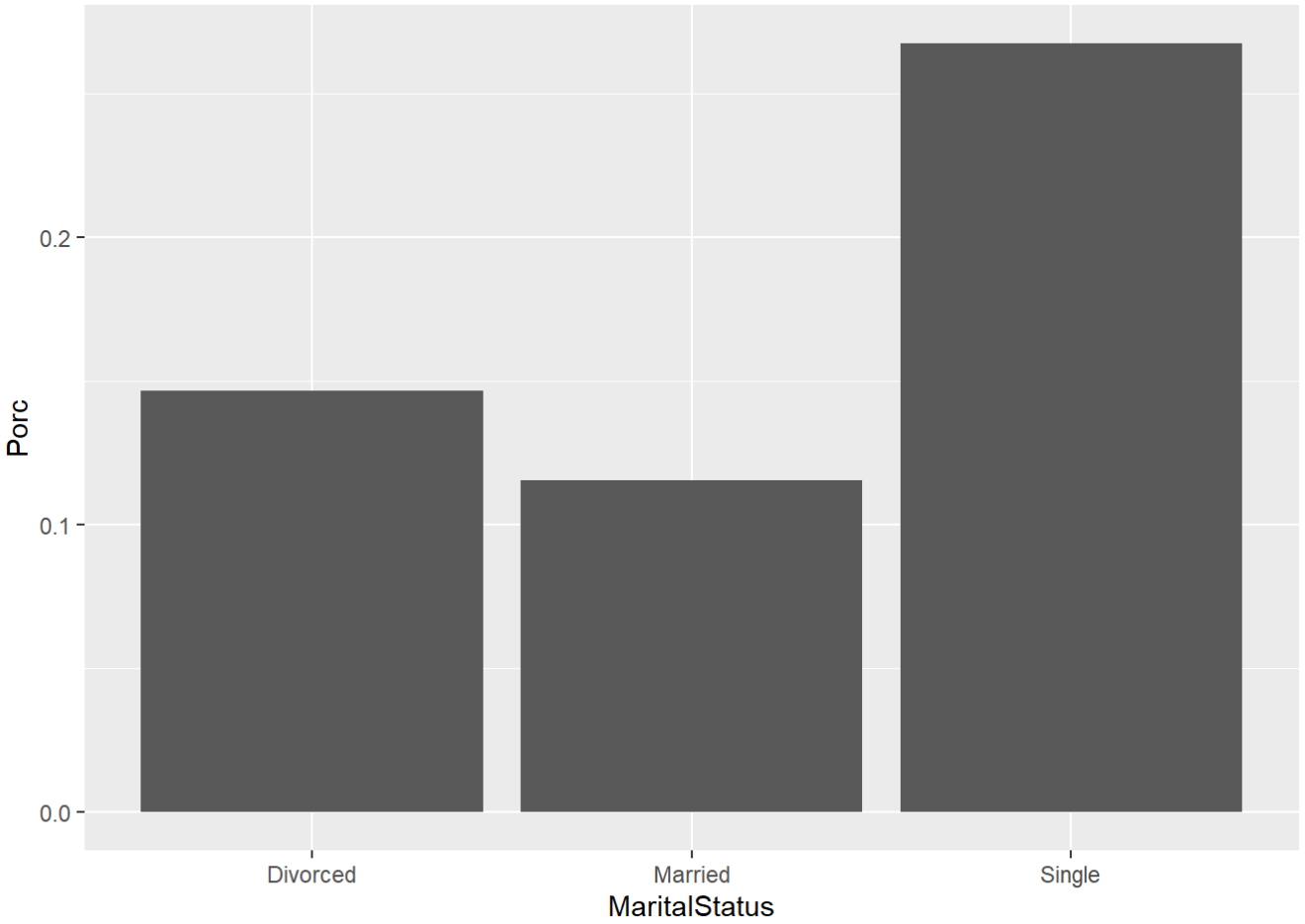
```
##
## [[4]]
```


0.2
0.1
0.0
Porc
Fashion
Grocery
Laptop & Accessory
Mobile
Mobile Phone
Others
PreferedOrderCat

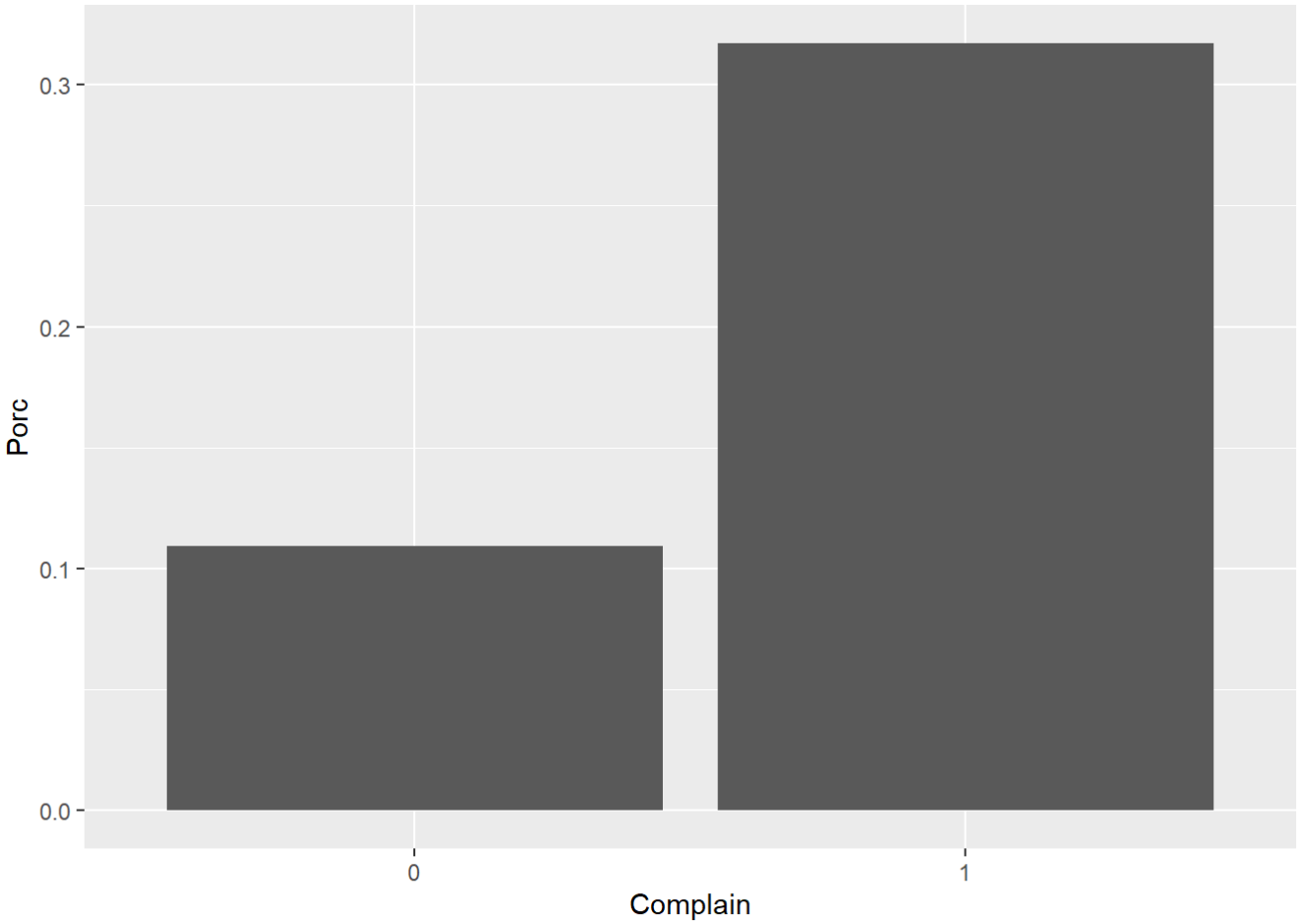
```
##
## [[5]]
```



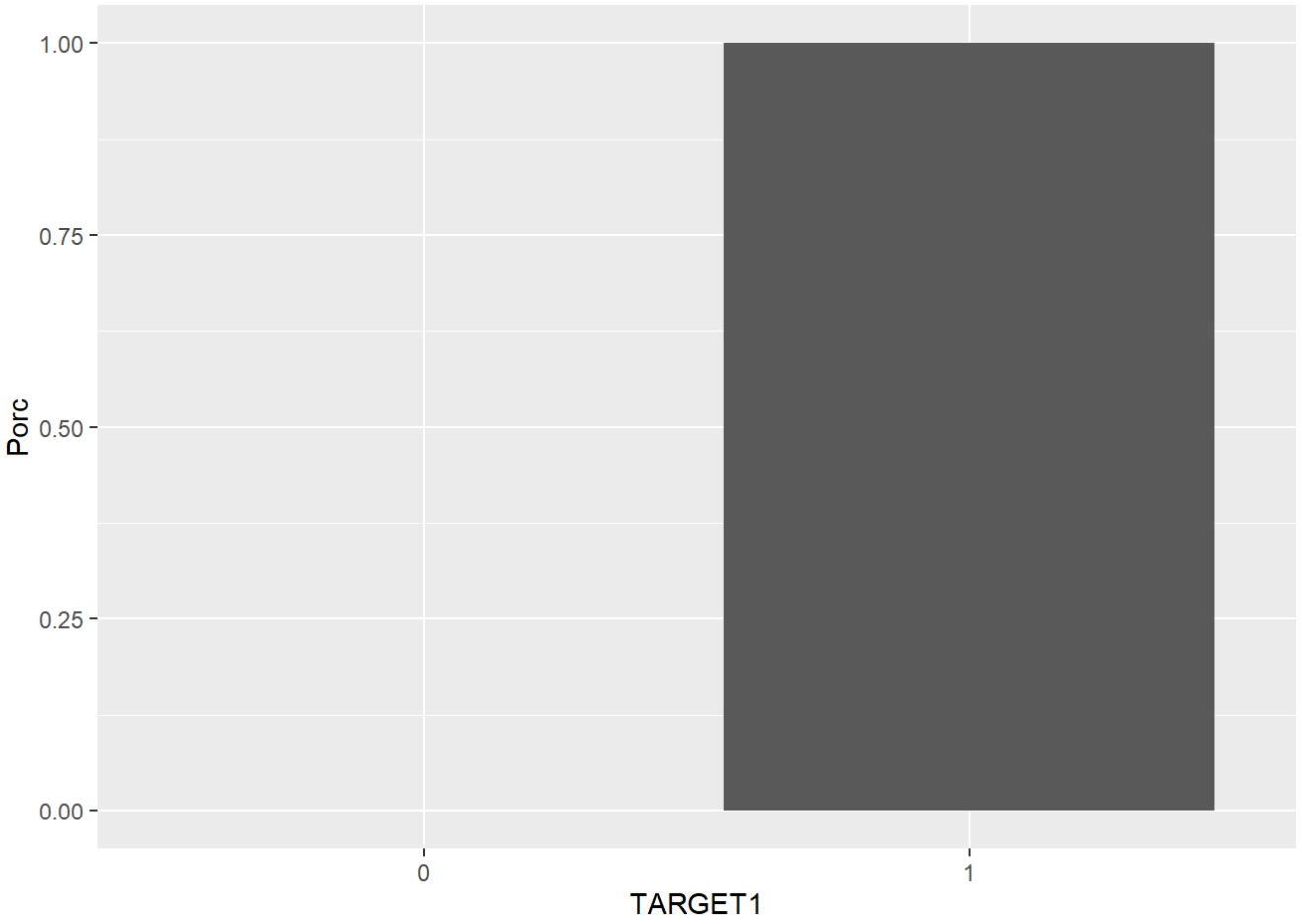

```
##
## [[6]]
```



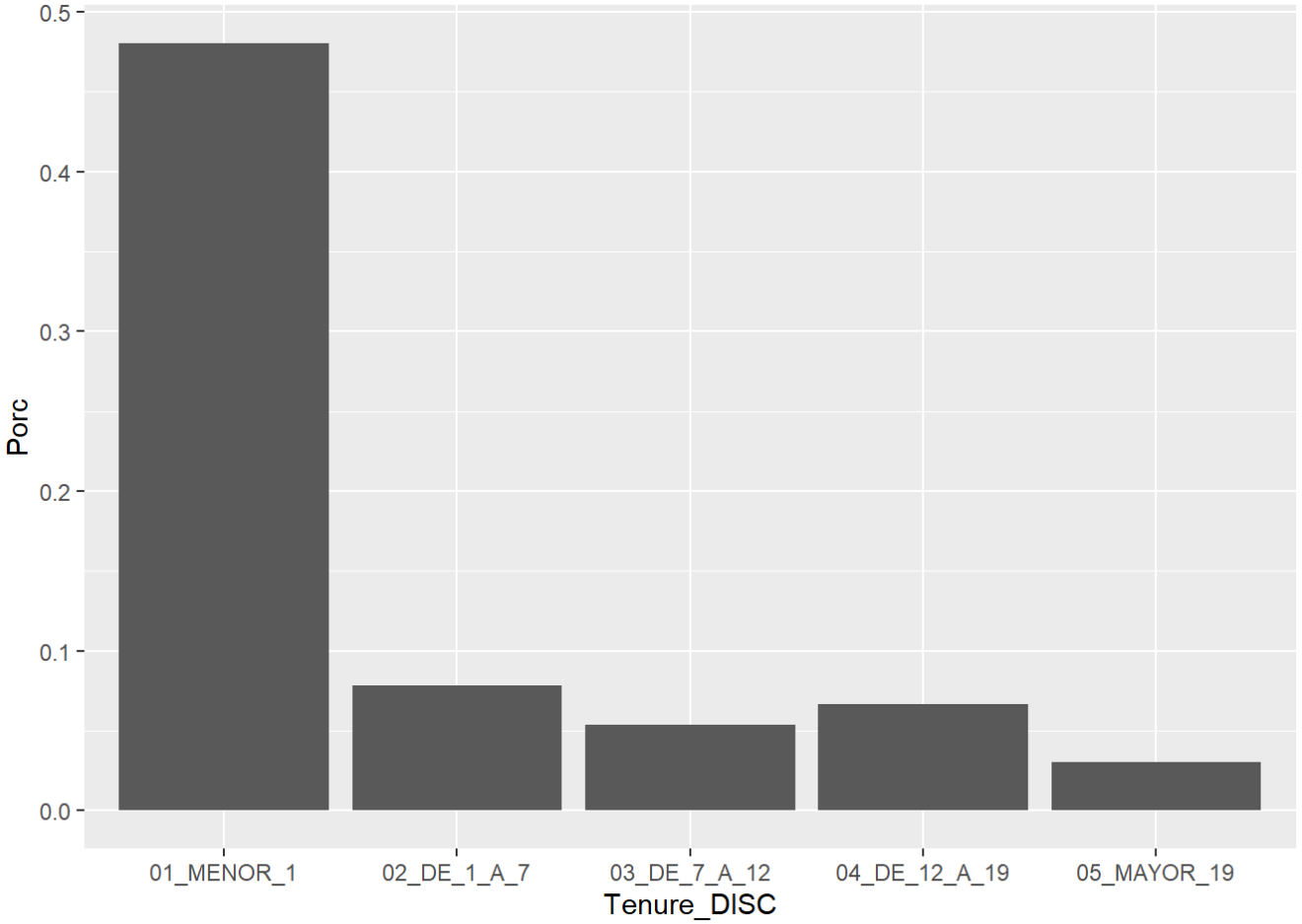

```
##
## [[7]]
```



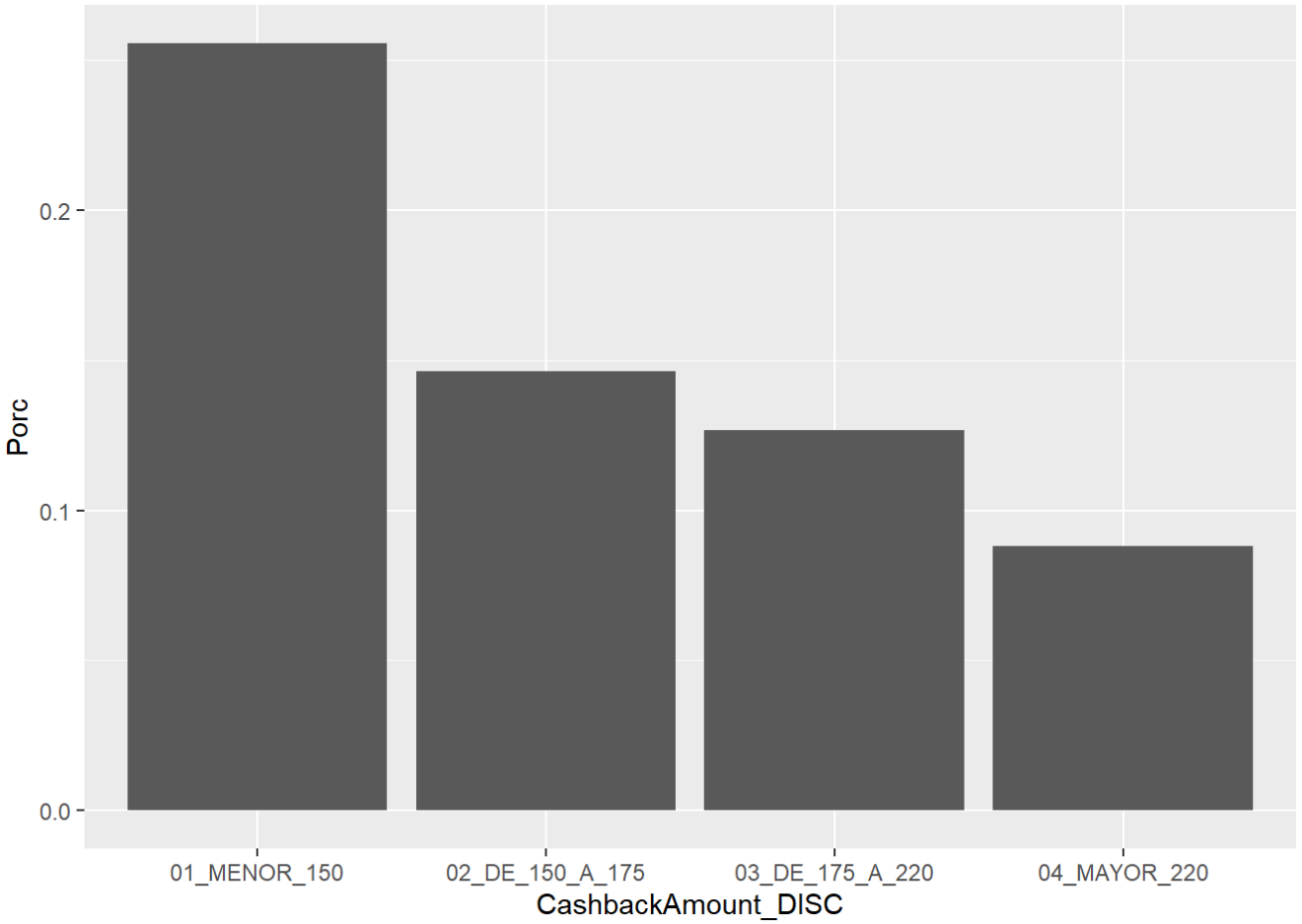

```
##
## [[8]]
```



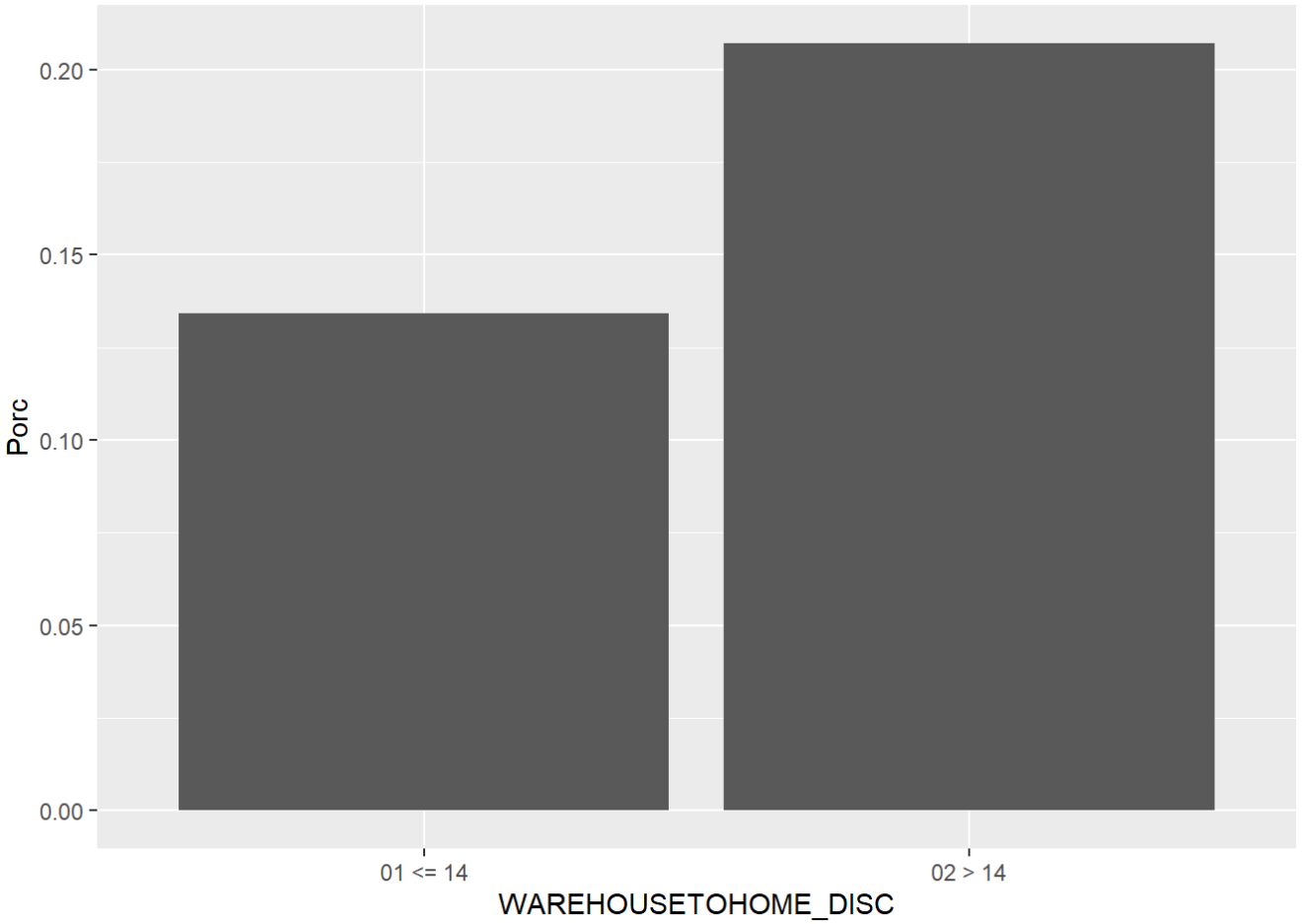

```
##
## [[9]]
```


0.5
0.4
0.3
0.2
0.1
0.0
Porc
01_MENOR_1
02_DE_1_A_7
03_DE_7_A_12
04_DE_12_A_19
05_MAYOR_19
Tenure_DISC

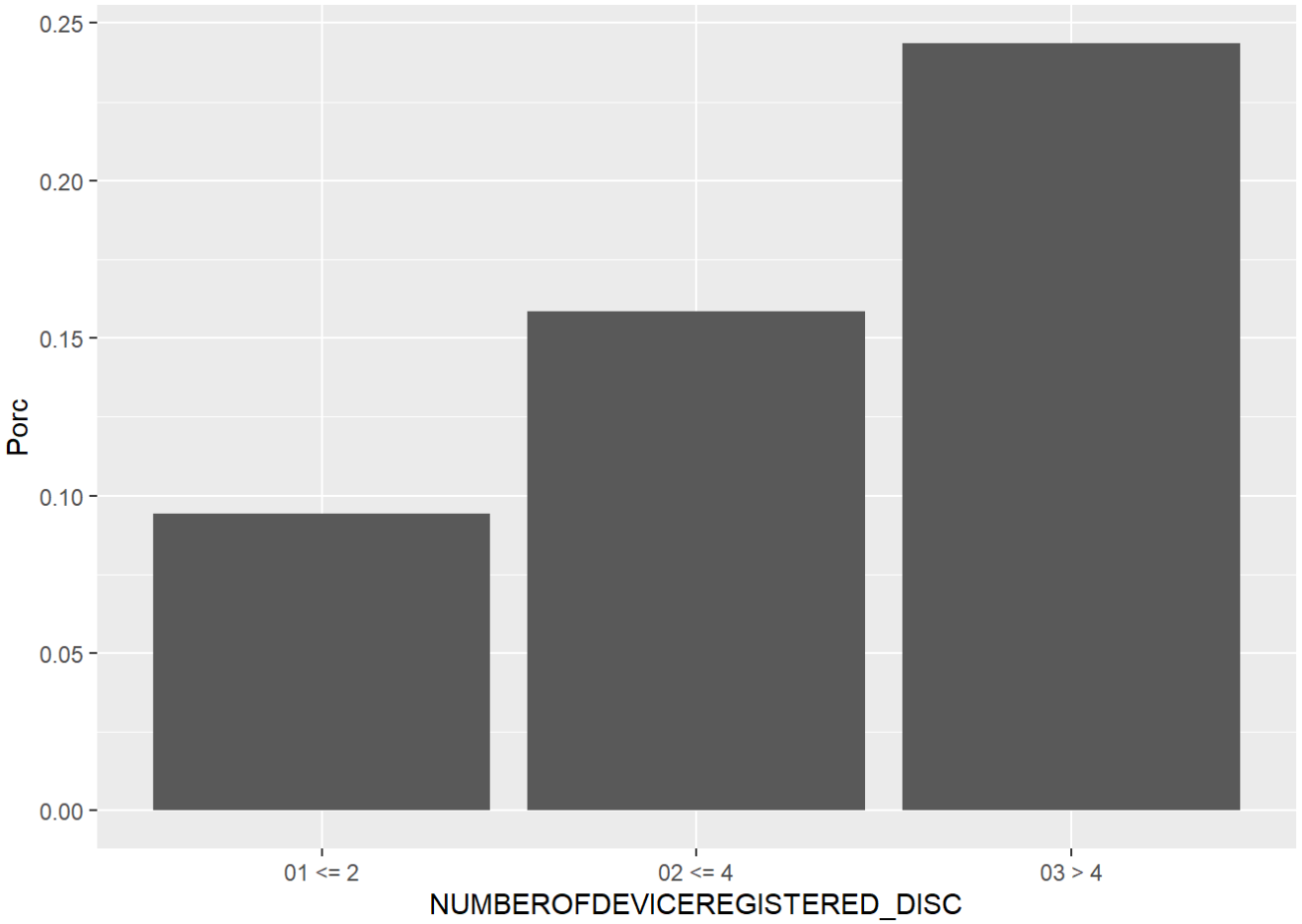
```
##
## [[10]]
```


0.2
0.1
0.0
Porc
01_MENOR_150
02_DE_150_A_175
03_DE_175_A_220
04_MAYOR_220
CashbackAmount_DISC

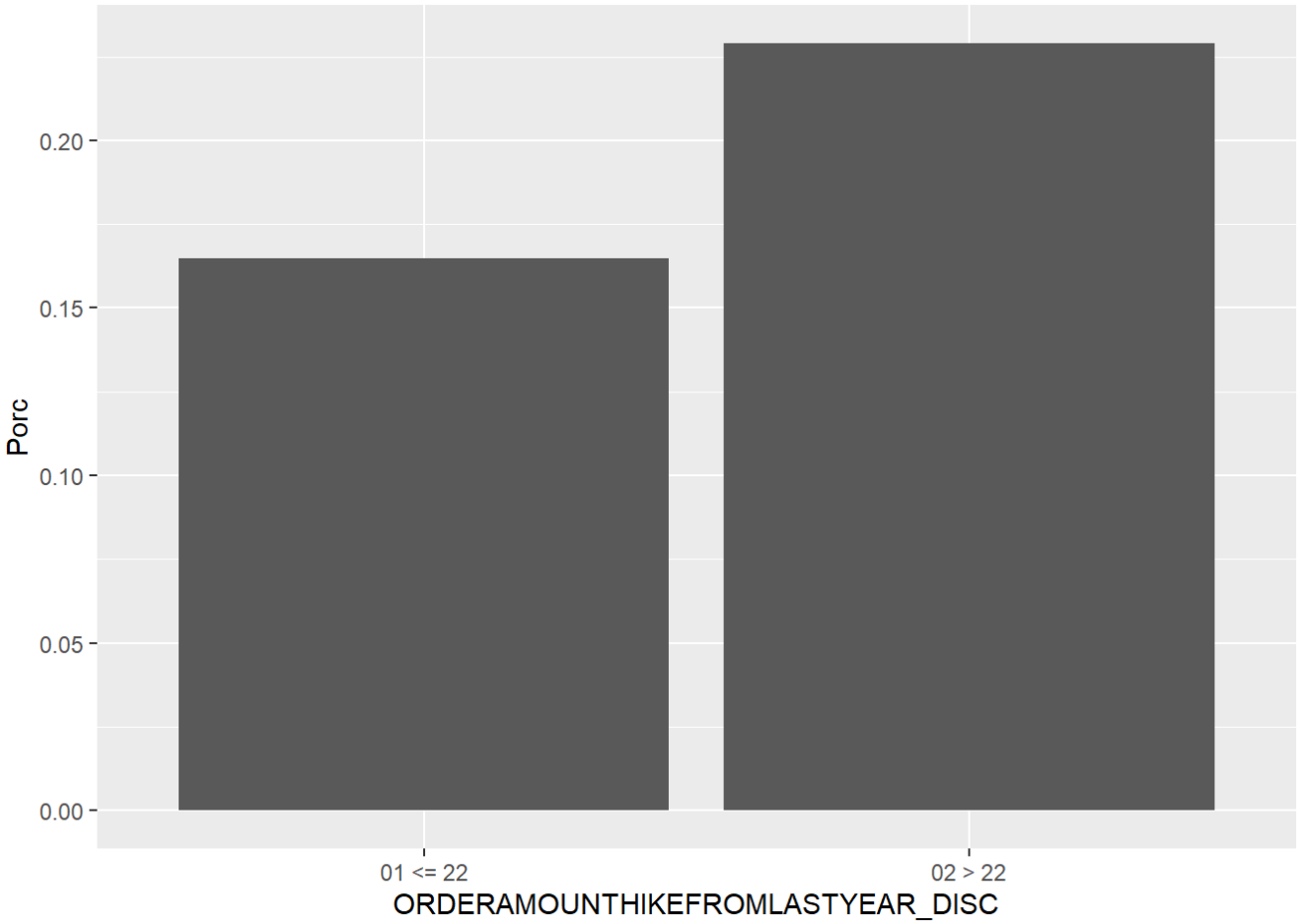
```
##
## [[11]]
```



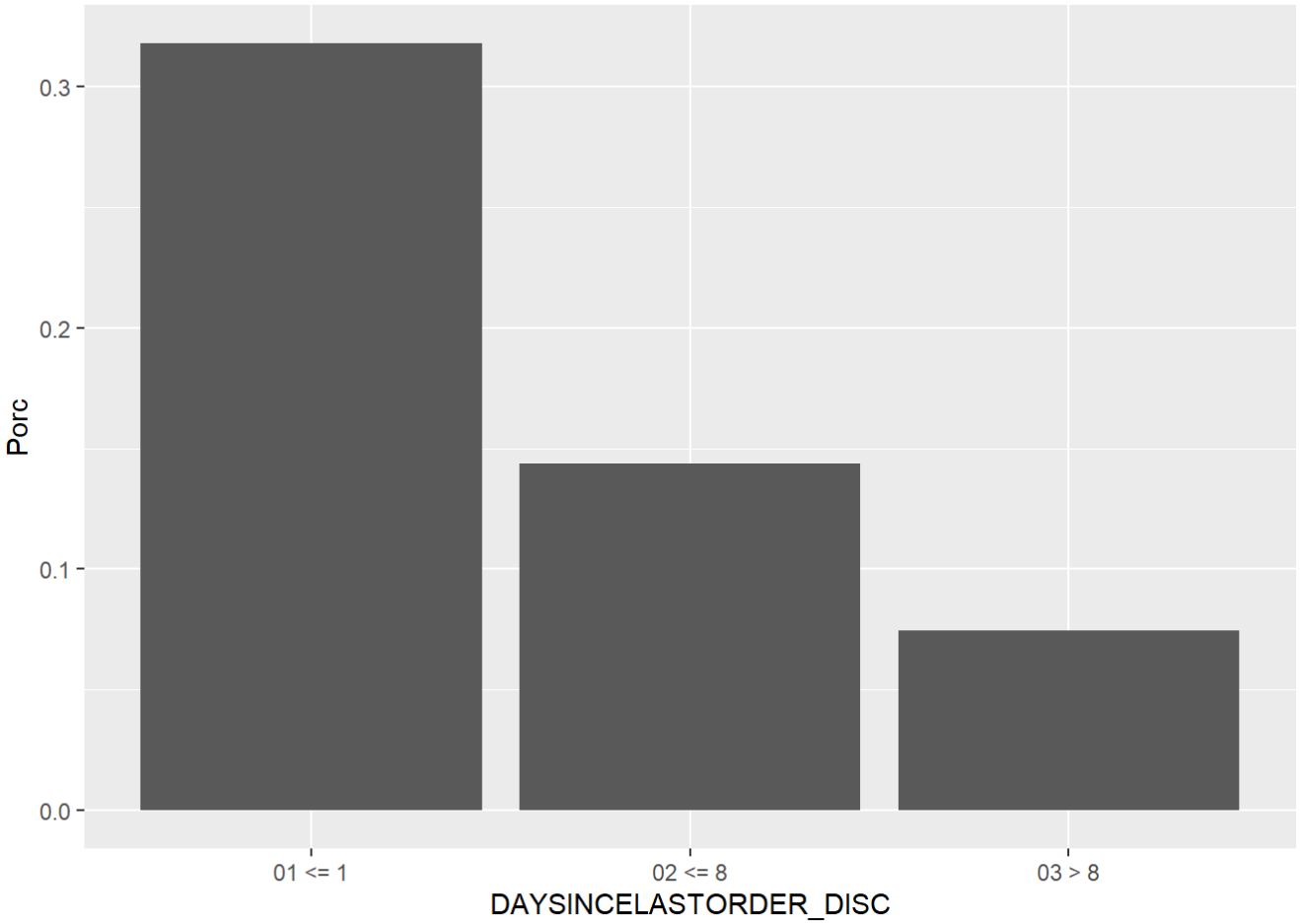

```
##
## [[12]]
```


0.25
0.20
0.15
0.10
0.05
0.00
Porc
01 <= 2
02 <= 4
03 > 4
NUMBEROFDEVICEREGISTERED_DISC

```
##
## [[13]]
```


0.20
0.15
0.10
0.05
0.00
Porc
01 <= 22
02 > 22
ORDERAMOUNTHIKEFROMLASTYEAR_DISC

```
##
## [[14]]
```



La mayoria han salido monotónicas.

Antes de continuar vamos a guardar en un objeto de R las discretizaciones, será una lista.

```
discretizaciones <- list(
 disc_temp_Tenure = disc_temp_Tenure,
 disc_temp_CashbackAmount = disc_temp_CashbackAmount,
 disc_temp_WarehouseToHome = disc_temp_WarehouseToHome,
 disc_temp_NumberOfDeviceRegistered = disc_temp_NumberOfDeviceRegistered,
 disc_temp_OrderAmountHikeFromlastYear = disc_temp_OrderAmountHikeFromlastYear,
 disc_temp_DaySinceLastOrder = disc_temp_DaySinceLastOrder)
saveRDS(discretizaciones,'02_CortesDiscretizaciones.rds')
```

Vamos a ver como ha quedado nuestro fichero antes de pasar a la fase de modelización.

```
glimpse(df)
```

```
## Rows: 5,626
## Columns: 15
## $ PreferredLoginDevice             <fct> Mobile Phone, Phone, Phone, Phone, Ph~
## $ CityTier                         <fct> 3, 1, 1, 3, 1, 1, 3, 1, 3, 1, 1, 1, 1~
## $ PreferredPaymentMode             <fct> Debit Card, UPI, Debit Card, Debit Ca~
## $ PreferedOrderCat                 <fct> Laptop & Accessory, Mobile, Mobile, L~
## $ SatisfactionScore                <fct> 2, 3, 3, 5, 5, 5, 2, 2, 3, 3, 3, 3, 3~
## $ MaritalStatus                    <fct> Single, Single, Single, Single, Singl~
## $ Complain                         <fct> 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1~
## $ CustomerID                       <dbl> 50001, 50002, 50003, 50004, 50005, 50~
## $ TARGET1                          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ Tenure_DISC                      <fct> 02_DE_1_A_7, 01_MENOR_1, 01_MENOR_1, ~
## $ CashbackAmount_DISC              <fct> 02_DE_150_A_175, 01_MENOR_150, 01_MEN~
## $ WAREHOUSETOHOME_DISC             <fct> 01 <= 14, 01 <= 14, 02 > 14, 02 > 14,~
## $ NUMBEROFDEVICEREGISTERED_DISC    <fct> 02 <= 4, 02 <= 4, 02 <= 4, 02 <= 4, 0~
## $ ORDERAMOUNTHIKEFROMLASTYEAR_DISC <fct> 01 <= 22, 01 <= 22, 01 <= 22, 02 > 22~
## $ DAYSINCELASTORDER_DISC           <fct> 02 <= 8, 01 <= 1, 02 <= 8, 02 <= 8, 0~
```

Ordenamos las variables, con la identificación del cliente al principio de la lista y la Target al final.

```
centrales <- setdiff(names(df),c('CustomerID','TARGET1'))
df <- df %>% select(
  CustomerID,
  one_of(centrales),
  TARGET1)
```

Comprobamos de nuevo.

```
glimpse(df)
```

```
## Rows: 5,626
## Columns: 15
## $ CustomerID                       <dbl> 50001, 50002, 50003, 50004, 50005, 50~
## $ PreferredLoginDevice             <fct> Mobile Phone, Phone, Phone, Phone, Ph~
## $ CityTier                         <fct> 3, 1, 1, 3, 1, 1, 3, 1, 3, 1, 1, 1, 1~
## $ PreferredPaymentMode             <fct> Debit Card, UPI, Debit Card, Debit Ca~
## $ PreferedOrderCat                 <fct> Laptop & Accessory, Mobile, Mobile, L~
## $ SatisfactionScore                <fct> 2, 3, 3, 5, 5, 5, 2, 2, 3, 3, 3, 3, 3~
## $ MaritalStatus                    <fct> Single, Single, Single, Single, Singl~
## $ Complain                         <fct> 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1~
## $ Tenure_DISC                      <fct> 02_DE_1_A_7, 01_MENOR_1, 01_MENOR_1, ~
## $ CashbackAmount_DISC              <fct> 02_DE_150_A_175, 01_MENOR_150, 01_MEN~
## $ WAREHOUSETOHOME_DISC             <fct> 01 <= 14, 01 <= 14, 02 > 14, 02 > 14,~
## $ NUMBEROFDEVICEREGISTERED_DISC    <fct> 02 <= 4, 02 <= 4, 02 <= 4, 02 <= 4, 0~
## $ ORDERAMOUNTHIKEFROMLASTYEAR_DISC <fct> 01 <= 22, 01 <= 22, 01 <= 22, 02 > 22~
## $ DAYSINCELASTORDER_DISC           <fct> 02 <= 8, 01 <= 1, 02 <= 8, 02 <= 8, 0~
## $ TARGET1                          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
```

- 4.3 - Limpieza

Limpiamos el entorno de variables temporales.

```
a_borrar <- setdiff(ls(),'df')
rm(list=c(a_borrar,'a_borrar'))
```

Guardamos una copia temporal con los cambios realizados hasta ahora, tenermos 15 variables, 6 de ellas discretizadas, y todas de tipo factor, excepto CustomerID.

```
saveRDS(df,'cache2.rds')
```

4.5 - Modelización

- 4.5.1 - Preparar las funciones que vamos a necesitar

Función para crear una matriz de confusión.

```
confusion<-function(real,scoring,umbral){
  conf<-table(real,scoring>=umbral)
  if(ncol(conf)==2) return(conf) else return(NULL)
}
```

Funcion para calcular las métricas de los modelos: acierto, precisión, cobertura y F1.

```
metricas<-function(matriz_conf){
  acierto <- (matriz_conf[1,1] + matriz_conf[2,2]) / sum(matriz_conf) *100
  precision <- matriz_conf[2,2] / (matriz_conf[2,2] + matriz_conf[1,2]) *100
  cobertura <- matriz_conf[2,2] / (matriz_conf[2,2] + matriz_conf[2,1]) *100
  F1 <- 2*precision*cobertura/(precision+cobertura)
  salida<-c(acierto,precision,cobertura,F1)
  return(salida)
}
```

Función para probar distintos umbrales y ver el efecto sobre precisión y cobertura.

```
umbrales<-function(real,scoring){
  umbrales<-data.frame(umbral=rep(0,times=19),acierto=rep(0,times=19),precision=rep
(0,times=19),cobertura=rep(0,times=19),F1=rep(0,times=19))
  cont <- 1
  for (cada in seq(0.05,0.95,by = 0.05)){
    datos<-metricas(confusion(real,scoring,cada))
    registro<-c(cada,datos)
    umbrales[cont,]<-registro
    cont <- cont + 1
  }
  return(umbrales)
}
```

Funciones que calculan la curva ROC y el AUC.

```
roc<-function(prediction){
  r<-performance(prediction,'tpr','fpr')
  plot(r, col='darkgreen')
}

auc<-function(prediction){
  a<-performance(prediction,'auc')
  return(a@y.values[[1]])
}
```

- 4.5.2 - Creamos las particiones de training (70%) y test (30%)

Generamos una variable aleatoria con una distribución 70-30

```
df$random<-sample(0:1,size = nrow(df),replace = T,prob = c(0.3,0.7))
```

Creamos los dos dataframes. Y eliminamos la random generada.

```
train<-filter(df,random==1)
test<-filter(df,random==0)

df$random <- NULL
```

- 4.5.3 - Creación del modelo de propensión

Vamos a realizar la modelización con tres algoritmos diferentes con el fin de compararlos y así elegir el que mejor funcione: Regresión Logística, Árboles de decisión y Random Forest.

4.5.3.1 - Identificamos las variables

Todas las variables son independientes excepto la identificación del cliente y la target.

```
independientes <- setdiff(names(df),c('CustomerID','TARGET1'))
target <- 'TARGET1'
independientes
```

```
##  [1] "PreferredLoginDevice"             "CityTier"
##  [3] "PreferredPaymentMode"             "PreferedOrderCat"
##  [5] "SatisfactionScore"                "MaritalStatus"
##  [7] "Complain"                         "Tenure_DISC"
##  [9] "CashbackAmount_DISC"              "WAREHOUSETOHOME_DISC"
## [11] "NUMBEROFDEVICEREGISTERED_DISC"    "ORDERAMOUNTHIKEFROMLASTYEAR_DISC"
## [13] "DAYSINCELASTORDER_DISC"
```

4.5.3.2 - Creamos la formula para usar en el modelo

```
formula <- reformulate(independientes,target)
```

- 4.5.4 - Modelizamos con regresión logística

Primero vamos a hacer un modelo con todas las variables.

```
formula_rl <- formula
rl<- glm(formula_rl,train,family=binomial(link='logit'))
summary(rl)
```

```
##
## Call:
## glm(formula = formula_rl, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1349  -0.3845  -0.2102  -0.1084   3.3713
##
## Coefficients:
##                                          Estimate Std. Error z value
## (Intercept)                             -1.063335   0.601307  -1.768
## PreferredLoginDeviceMobile Phone        -0.145061   0.155209  -0.935
## PreferredLoginDevicePhone               -0.471562   0.186079  -2.534
## CityTier2                                0.795835   0.289770   2.746
## CityTier3                                0.753978   0.150417   5.013
## PreferredPaymentModeCC                  -0.122606   0.466492  -0.263
## PreferredPaymentModeCOD                  0.738732   0.454561   1.625
## PreferredPaymentModeCredit Card         -0.074691   0.423118  -0.177
## PreferredPaymentModeDebit Card           0.278547   0.410672   0.678
## PreferredPaymentModeE wallet             0.497925   0.441213   1.129
## PreferredPaymentModeUPI                 -0.085335   0.467294  -0.183
## PreferedOrderCatGrocery                 -0.435403   0.426088  -1.022
## PreferedOrderCatLaptop & Accessory      -1.240917   0.244648  -5.072
## PreferedOrderCatMobile                  -0.824388   0.339590  -2.428
## PreferedOrderCatMobile Phone            -0.466286   0.301665  -1.546
## PreferedOrderCatOthers                   0.529187   0.434568   1.218
## SatisfactionScore2                       0.004981   0.253474   0.020
## SatisfactionScore3                       0.501879   0.176225   2.848
## SatisfactionScore4                       0.608394   0.194359   3.130
## SatisfactionScore5                       1.076932   0.186873   5.763
## MaritalStatusMarried                    -0.337916   0.178425  -1.894
## MaritalStatusSingle                      0.545907   0.176425   3.094
## Complain1                                1.766493   0.123128  14.347
## Tenure_DISC02_DE_1_A_7                  -2.378459   0.171752 -13.848
## Tenure_DISC03_DE_7_A_12                 -3.269678   0.219265 -14.912
## Tenure_DISC04_DE_12_A_19                -2.874314   0.208016 -13.818
## Tenure_DISC05_MAYOR_19                  -3.740235   0.312774 -11.958
## CashbackAmount_DISC02_DE_150_A_175      -0.372110   0.220482  -1.688
## CashbackAmount_DISC03_DE_175_A_220      -0.100524   0.296503  -0.339
## CashbackAmount_DISC04_MAYOR_220         -0.541712   0.406427  -1.333
## WAREHOUSETOHOME_DISC02 > 14              0.517277   0.117876   4.388
## NUMBEROFDEVICEREGISTERED_DISC02 <= 4     0.632410   0.243486   2.597
## NUMBEROFDEVICEREGISTERED_DISC03 > 4      1.370125   0.264861   5.173
## ORDERAMOUNTHIKEFROMLASTYEAR_DISC02 > 22  0.109567   0.238441   0.460
## DAYSINCELASTORDER_DISC02 <= 8           -0.857630   0.131719  -6.511
## DAYSINCELASTORDER_DISC03 > 8            -0.891645   0.249549  -3.573
##                                                    Pr(>|z|)
## (Intercept)                                        0.076998 .
## PreferredLoginDeviceMobile Phone                   0.349987
## PreferredLoginDevicePhone                          0.011270 *
## CityTier2                                          0.006025 **
## CityTier3                                   0.0000005370275 ***
## PreferredPaymentModeCC                             0.792686
```

```
## PreferredPaymentModeCOD                                0.104130
## PreferredPaymentModeCredit Card                        0.859882
## PreferredPaymentModeDebit Card                         0.497600
## PreferredPaymentModeE wallet                           0.259093
## PreferredPaymentModeUPI                                0.855100
## PreferedOrderCatGrocery                                0.306846
## PreferedOrderCatLaptop & Accessory           0.0000003931199 ***
## PreferedOrderCatMobile                                 0.015199 *
## PreferedOrderCatMobile Phone                           0.122176
## PreferedOrderCatOthers                                 0.223326
## SatisfactionScore2                                     0.984321
## SatisfactionScore3                                     0.004400 **
## SatisfactionScore4                                     0.001747 **
## SatisfactionScore5                           0.0000000082677 ***
## MaritalStatusMarried                                   0.058241 .
## MaritalStatusSingle                                    0.001973 **
## Complain1                                 < 0.0000000000000002 ***
## Tenure_DISC02_DE_1_A_7                    < 0.0000000000000002 ***
## Tenure_DISC03_DE_7_A_12                   < 0.0000000000000002 ***
## Tenure_DISC04_DE_12_A_19                  < 0.0000000000000002 ***
## Tenure_DISC05_MAYOR_19                    < 0.0000000000000002 ***
## CashbackAmount_DISC02_DE_150_A_175                     0.091467 .
## CashbackAmount_DISC03_DE_175_A_220                     0.734585
## CashbackAmount_DISC04_MAYOR_220                        0.182576
## WAREHOUSETOHOME_DISC02 > 14                  0.0000114222309 ***
## NUMBEROFDEVICEREGISTERED_DISC02 <= 4                   0.009396 **
## NUMBEROFDEVICEREGISTERED_DISC03 > 4          0.0000002303670 ***
## ORDERAMOUNTHIKEFROMLASTYEAR_DISC02 > 22                0.645864
## DAYSINCELASTORDER_DISC02 <= 8                0.0000000000746 ***
## DAYSINCELASTORDER_DISC03 > 8                           0.000353 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3548.0  on 3956  degrees of freedom
## Residual deviance: 2054.4  on 3921  degrees of freedom
## AIC: 2126.4
##
## Number of Fisher Scoring iterations: 6
```

Revisamos la significatividad y mantenemos todas las variables que tengan tres estrellas en alguna categoría.

```
a_mantener <- c(
  'CityTier',
  'PreferedOrderCat',
  'SatisfactionScore',
  'Complain',
  'Tenure_DISC',
  'WAREHOUSETOHOME_DISC',
  'NUMBEROFDEVICEREGISTERED_DISC',
  'DAYSINCELASTORDER_DISC'
)
a_mantener
```

```
## [1] "CityTier"                      "PreferedOrderCat"
## [3] "SatisfactionScore"             "Complain"
## [5] "Tenure_DISC"                   "WAREHOUSETOHOME_DISC"
## [7] "NUMBEROFDEVICEREGISTERED_DISC" "DAYSINCELASTORDER_DISC"
```

Volvemos a modelizar de nuevo, con estas variables seleccionadas.

```
formula_rl <- reformulate(a_mantener,target)
rl<- glm(formula_rl,train,family=binomial(link='logit'))
summary(rl)
```

```
##
## Call:
## glm(formula = formula_rl, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2168  -0.3953  -0.2209  -0.1174   3.1817
##
## Coefficients:
##                                    Estimate Std. Error z value
## (Intercept)                        -1.28014    0.34969  -3.661
## CityTier2                           0.57291    0.27168   2.109
## CityTier3                           0.95389    0.13082   7.291
## PreferedOrderCatGrocery            -0.57948    0.36528  -1.586
## PreferedOrderCatLaptop & Accessory -1.14478    0.18933  -6.046
## PreferedOrderCatMobile             -0.69619    0.21945  -3.172
## PreferedOrderCatMobile Phone       -0.30157    0.20096  -1.501
## PreferedOrderCatOthers              0.32888    0.37687   0.873
## SatisfactionScore2                  0.09988    0.23310   0.428
## SatisfactionScore3                  0.46745    0.16948   2.758
## SatisfactionScore4                  0.60603    0.18609   3.257
## SatisfactionScore5                  1.04655    0.17843   5.865
## Complain1                           1.71631    0.11874  14.455
## Tenure_DISC02_DE_1_A_7             -2.36531    0.16605 -14.245
## Tenure_DISC03_DE_7_A_12            -3.21683    0.21082 -15.259
## Tenure_DISC04_DE_12_A_19           -2.95117    0.20631 -14.305
## Tenure_DISC05_MAYOR_19             -3.81666    0.30882 -12.359
## WAREHOUSETOHOME_DISC02 > 14         0.45430    0.11352   4.002
## NUMBEROFDEVICEREGISTERED_DISC02 <= 4  0.75184    0.23520   3.197
## NUMBEROFDEVICEREGISTERED_DISC03 > 4   1.39140    0.25611   5.433
## DAYSINCELASTORDER_DISC02 <= 8      -0.96649    0.12620  -7.659
## DAYSINCELASTORDER_DISC03 > 8       -0.87616    0.24324  -3.602
##                                              Pr(>|z|)
## (Intercept)                                  0.000251 ***
## CityTier2                                    0.034968 *
## CityTier3                          0.0000000000003068 ***
## PreferedOrderCatGrocery                      0.112644
## PreferedOrderCatLaptop & Accessory 0.0000000014814382 ***
## PreferedOrderCatMobile                       0.001512 **
## PreferedOrderCatMobile Phone                 0.133437
## PreferedOrderCatOthers                       0.382835
## SatisfactionScore2                           0.668298
## SatisfactionScore3                           0.005814 **
## SatisfactionScore4                           0.001127 **
## SatisfactionScore5                 0.0000000044839075 ***
## Complain1                         < 0.0000000000000002 ***
## Tenure_DISC02_DE_1_A_7            < 0.0000000000000002 ***
## Tenure_DISC03_DE_7_A_12           < 0.0000000000000002 ***
## Tenure_DISC04_DE_12_A_19          < 0.0000000000000002 ***
## Tenure_DISC05_MAYOR_19            < 0.0000000000000002 ***
## WAREHOUSETOHOME_DISC02 > 14        0.0000628121611314 ***
## NUMBEROFDEVICEREGISTERED_DISC02 <= 4            0.001391 **
## NUMBEROFDEVICEREGISTERED_DISC03 > 4  0.0000000554880835 ***
```

```
## DAYSINCELASTORDER_DISC02 <= 8          0.0000000000000188 ***
## DAYSINCELASTORDER_DISC03 > 8                    0.000316 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3548.0  on 3956  degrees of freedom
## Residual deviance: 2137.3  on 3935  degrees of freedom
## AIC: 2181.3
##
## Number of Fisher Scoring iterations: 6
```

Vemos que ahora ya todas las variables tienen al menos una categoría con 3 estrellas de significación.

El valor de Estimate visto desde el punto de negocio, nos dice que las valores mas altos en negativo, como la permancia, nos indican una menor tasa de abandono a mayor permanencia.

Y calculamos el pseudo R cuadrado:

```
pr2_rl <- 1 -(rl$deviance / rl$null.deviance)
pr2_rl
```

```
## [1] 0.3976048
```

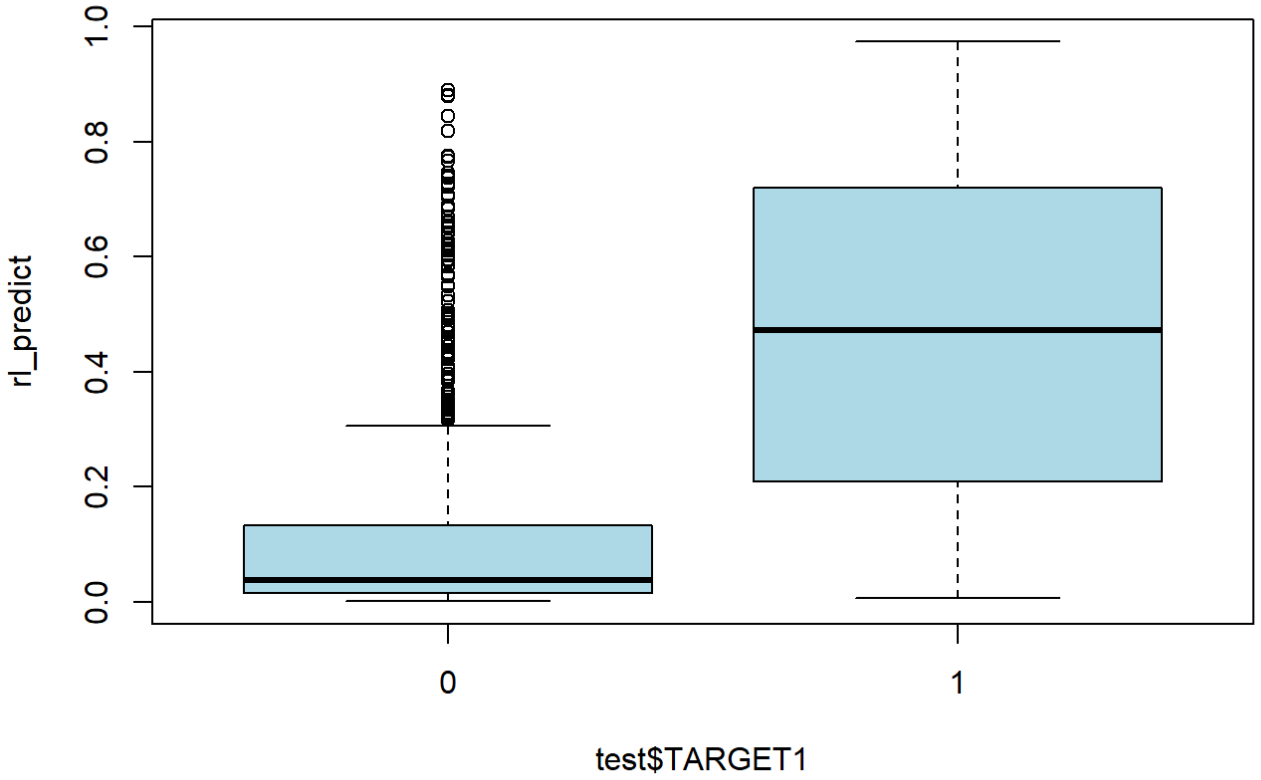
Aplicamos el modelo al conjunto de test, generando un vector con las probabilidades

```
rl_predict<-predict(rl,test,type = 'response')
head(rl_predict)
```

```
##          1          2          3          4          5          6
## 0.09666855 0.45392269 0.34993705 0.06853165 0.17345084 0.75855757
```

Comprobamos en el gráfico:

```
plot(rl_predict~test$TARGET1, col='lightblue')
```



El modelo está funcionando bien, ya que a los clientes que no abandonan las empresa el modelo les da un porcentaje de abandono muy bajo, en cambio a los clientes que si abandonan la empresa, el modelo nos da un 50 % de scoring.

Ahora tenemos que transformar la probabilidad en una decisión de si el cliente va a abandonar o no.

Con la función umbrales probamos diferentes cortes:

```
umb_rl<-umbrales(test$TARGET1,rl_predict)
umb_rl
```

```
##    umbral  acierto precision cobertura        F1
## 1    0.05 62.97184  31.11888 90.816327 46.354167
## 2    0.10 73.21750  38.06552 82.993197 52.192513
## 3    0.15 78.13062  43.48624 80.612245 56.495828
## 4    0.20 81.36609  48.18763 76.870748 59.239843
## 5    0.25 83.70282  52.82051 70.068027 60.233918
## 6    0.30 84.36189  54.54545 67.346939 60.273973
## 7    0.35 85.97963  59.86842 61.904762 60.869565
## 8    0.40 86.69862  62.76596 60.204082 61.458333
## 9    0.45 87.11803  66.39004 54.421769 59.813084
## 10   0.50 86.69862  67.64706 46.938776 55.421687
## 11   0.55 86.69862  69.56522 43.537415 53.556485
## 12   0.60 86.57879  72.72727 38.095238 50.000000
## 13   0.65 85.85980  75.00000 29.591837 42.439024
## 14   0.70 86.03954  81.44330 26.870748 40.409207
## 15   0.75 85.26064  85.29412 19.727891 32.044199
## 16   0.80 84.84122  88.67925 15.986395 27.089337
## 17   0.85 84.06231  88.88889 10.884354 19.393939
## 18   0.90 83.58298 100.00000  6.802721 12.738854
## 19   0.95 82.62433 100.00000  1.360544  2.684564
```

Seleccionamos el umbral que maximiza la F1

```
umbral_final_rl<-umb_rl[which.max(umb_rl$F1),1]
umbral_final_rl
```

```
## [1] 0.4
```

Evaluamos la matriz de confusión y las métricas con el umbral optimizado

```
confusion(test$TARGET1,rl_predict,umbral_final_rl)
```

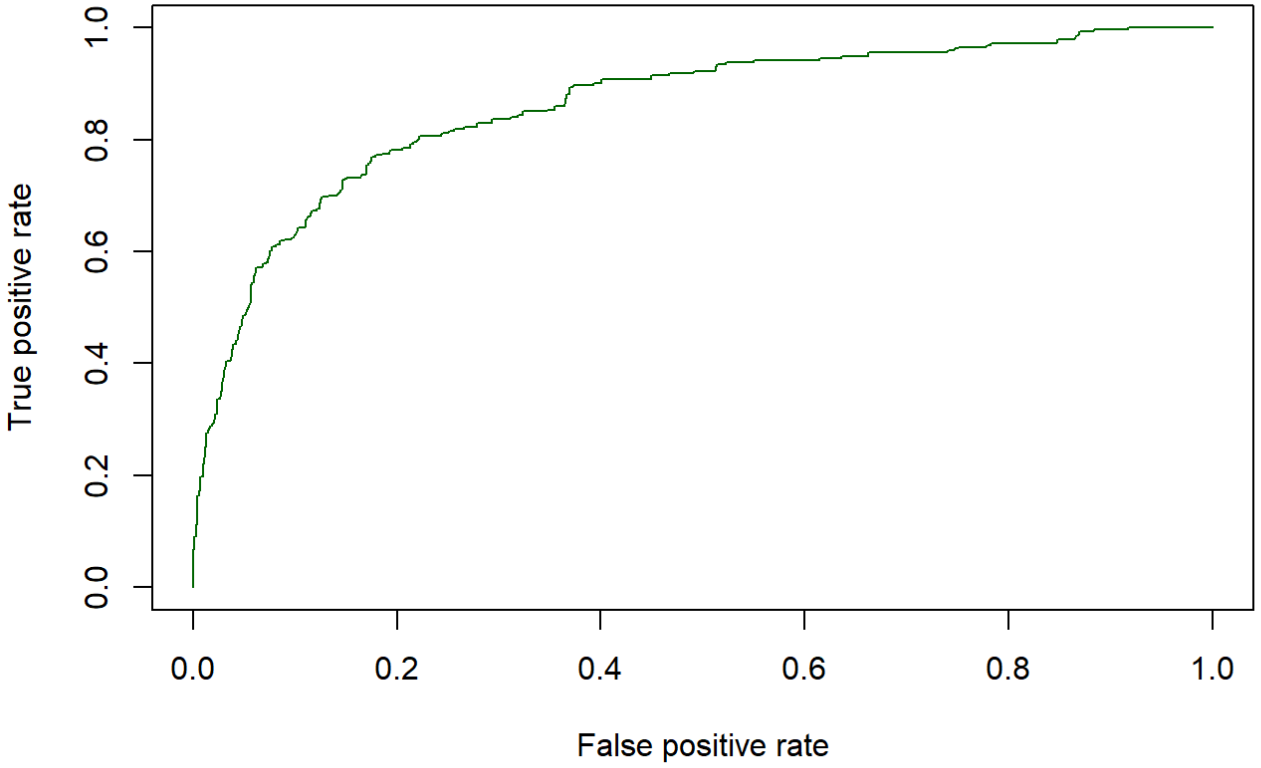
```
##
## real FALSE TRUE
##    0  1270  105
##    1   117  177
```

```
rl_metricas<-filter(umb_rl,umbral==umbral_final_rl)
rl_metricas
```

```
##   umbral  acierto precision cobertura       F1
## 1    0.4 86.69862  62.76596  60.20408 61.45833
```

Evaluamos la ROC

```
rl_prediction<-prediction(rl_predict,test$TARGET1)
roc(rl_prediction)
```



Sacamos las métricas definitivas incluyendo el AUC

```
rl_metricas<-cbind(rl_metricas,AUC=round(auc(rl_prediction),2)*100)
print(t(rl_metricas))
```

```
##                [,1]
## umbral      0.40000
## acierto    86.69862
## precision  62.76596
## cobertura  60.20408
## F1         61.45833
## AUC        86.00000
```

- 4.5.5 - Modelizamos con Árboles de decisión

Creamos el primer modelo

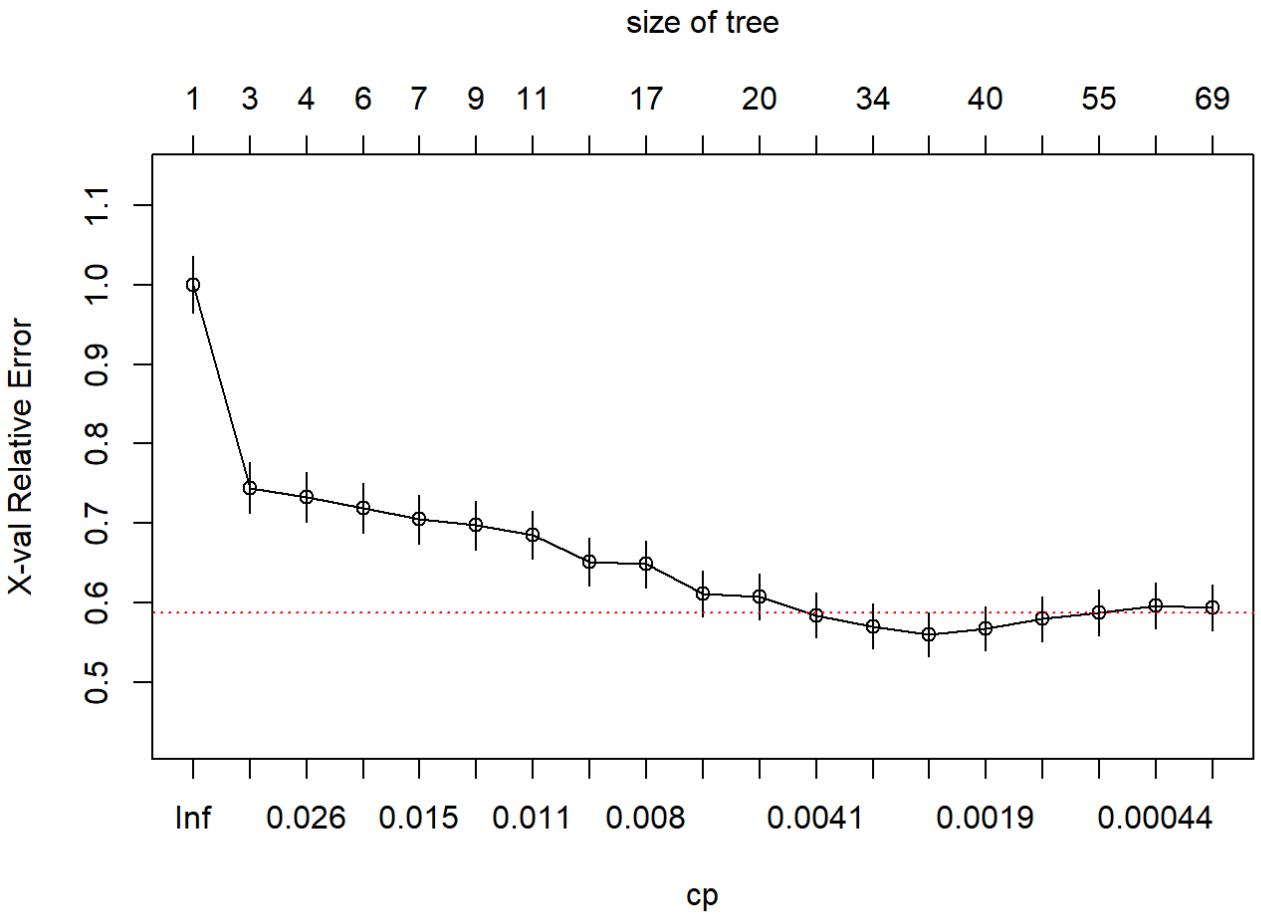
```
formula_ar <- formula
ar<-rpart(formula_ar, train, method = 'class', parms = list(
  split = "information"),
  control = rpart.control(cp = 0.00001))
```

Revisamos donde el error de validación cruzada empieza a crecer.

```
printcp(ar)
```

```
##
## Classification tree:
## rpart(formula = formula_ar, data = train, method = "class", parms = list(split =
"information"),
##     control = rpart.control(cp = 0.00001))
##
## Variables actually used in tree construction:
##  [1] CashbackAmount_DISC              CityTier
##  [3] Complain                         DAYSINCELASTORDER_DISC
##  [5] MaritalStatus                    NUMBEROFDEVICEREGISTERED_DISC
##  [7] ORDERAMOUNTHIKEFROMLASTYEAR_DISC PreferedOrderCat
##  [9] PreferredLoginDevice             PreferredPaymentMode
## [11] SatisfactionScore                Tenure_DISC
## [13] WAREHOUSETOHOME_DISC
##
## Root node error: 654/3957 = 0.16528
##
## n= 3957
##
##            CP nsplit rel error  xerror     xstd
## 1  0.12767584      0   1.00000 1.00000 0.035726
## 2  0.03058104      2   0.74465 0.74465 0.031599
## 3  0.02140673      3   0.71407 0.73242 0.031374
## 4  0.01681957      5   0.67125 0.71865 0.031118
## 5  0.01299694      6   0.65443 0.70489 0.030859
## 6  0.01223242      8   0.62844 0.69725 0.030713
## 7  0.01070336     10   0.60398 0.68502 0.030477
## 8  0.00840979     12   0.58257 0.65138 0.029812
## 9  0.00764526     16   0.54740 0.64832 0.029750
## 10 0.00535168     17   0.53976 0.61162 0.028994
## 11 0.00502403     19   0.52905 0.60703 0.028897
## 12 0.00336391     28   0.48165 0.58410 0.028406
## 13 0.00305810     33   0.46483 0.57034 0.028105
## 14 0.00229358     37   0.45260 0.55963 0.027867
## 15 0.00152905     39   0.44801 0.56728 0.028037
## 16 0.00114679     50   0.43119 0.57951 0.028306
## 17 0.00076453     54   0.42661 0.58716 0.028472
## 18 0.00025484     62   0.41896 0.59633 0.028670
## 19 0.00001000     68   0.41743 0.59327 0.028604
```

```
plotcp(ar, col='red')
```



Parece que minimiza en un nivel muy bajo de complejidad, cp = 0.002. Generamos un nuevo árbol con ese parámetro. Ademas vamos a incluir un nuevo parametro para que el árbol no tenga mas de 7 niveles.

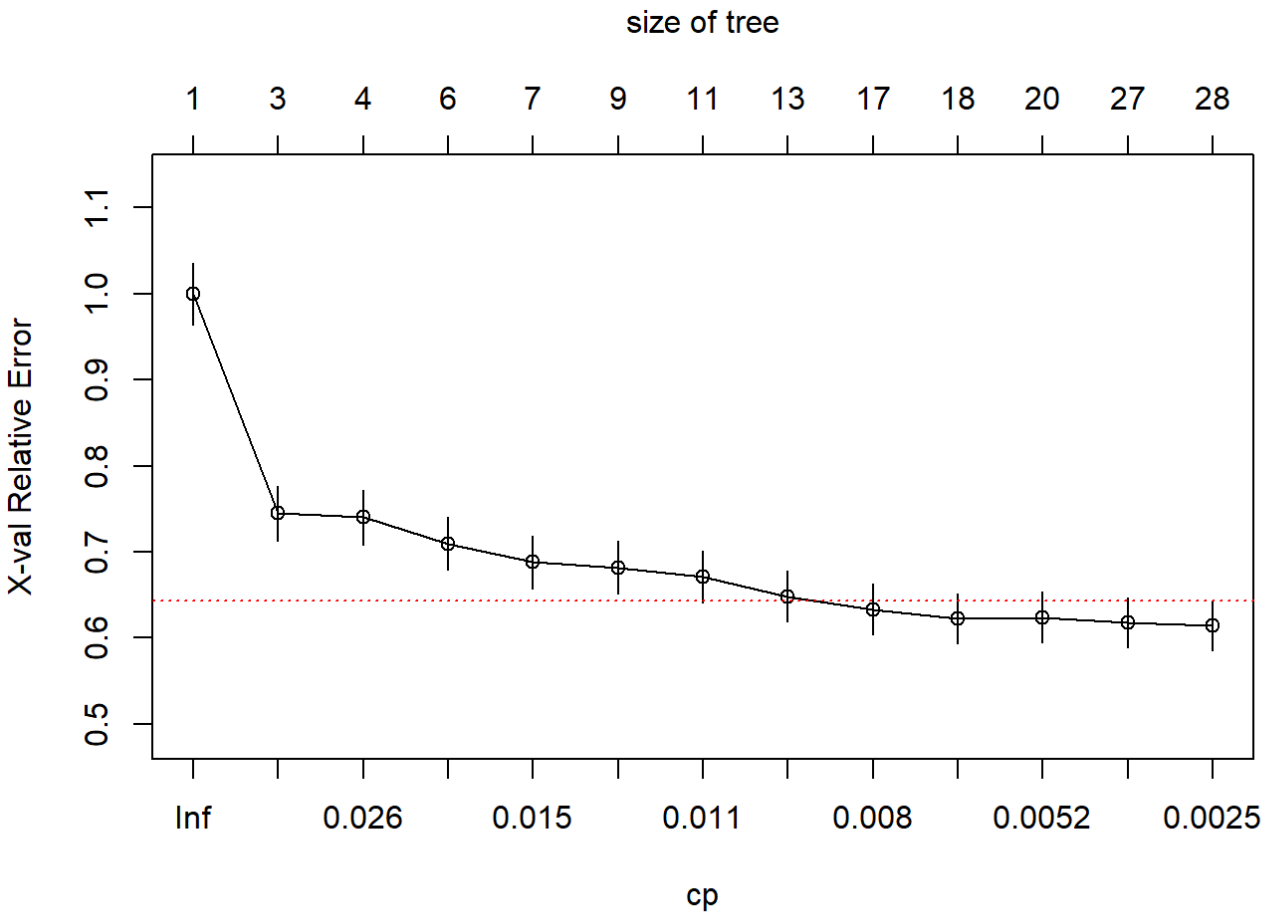
```
ar<-rpart(formula, train, method = 'class', parms = list(
  split = "information"),
  control = rpart.control(cp = 0.002,maxdepth = 7))
```

Revisamos de nuevo la complejidad.

```
printcp(ar)
```

```
##
## Classification tree:
## rpart(formula = formula, data = train, method = "class", parms = list(split = "i
nformation"),
##     control = rpart.control(cp = 0.002, maxdepth = 7))
##
## Variables actually used in tree construction:
##  [1] CityTier                      Complain
##  [3] DAYSINCELASTORDER_DISC        MaritalStatus
##  [5] NUMBEROFDEVICEREGISTERED_DISC PreferedOrderCat
##  [7] PreferredLoginDevice          PreferredPaymentMode
##  [9] SatisfactionScore             Tenure_DISC
##
## Root node error: 654/3957 = 0.16528
##
## n= 3957
##
##           CP nsplit rel error  xerror     xstd
## 1  0.1276758      0   1.00000 1.00000 0.035726
## 2  0.0305810      2   0.74465 0.74465 0.031599
## 3  0.0214067      3   0.71407 0.74006 0.031515
## 4  0.0168196      5   0.67125 0.70948 0.030946
## 5  0.0129969      6   0.65443 0.68807 0.030536
## 6  0.0122324      8   0.62844 0.68196 0.030417
## 7  0.0107034     10   0.60398 0.67125 0.030208
## 8  0.0084098     12   0.58257 0.64832 0.029750
## 9  0.0076453     16   0.54740 0.63303 0.029439
## 10 0.0053517     17   0.53976 0.62232 0.029218
## 11 0.0050240     19   0.52905 0.62385 0.029250
## 12 0.0030581     26   0.49388 0.61774 0.029122
## 13 0.0020000     27   0.49083 0.61468 0.029058
```

```
plotcp(ar, col='red')
```

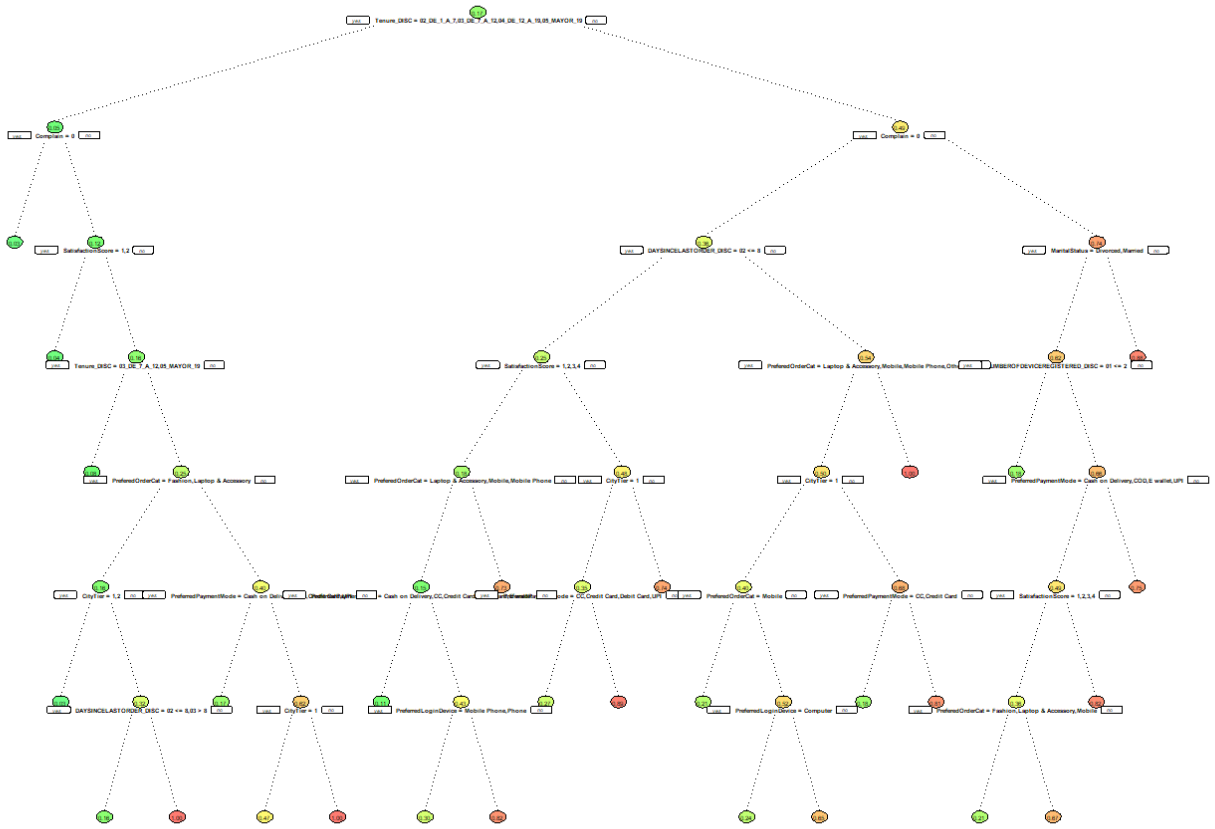


Vemos que ahora se ha estabilizado.

Vamos a crear el gráfico del árbol para analizarlo

```
rpart.plot(ar,type=2,extra = 7, under = TRUE,under.cex = 0.7,fallen.leaves=F,gap =
0,cex=0.2,yesno = 2,box.palette = "GnYlRd",branch.lty = 3)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Vamos a sacar las reglas que podrían ser utilizadas por ejemplo para hacer una implantación del árbol.

```
rpart.rules(ar,style = 'tall',cover = T)
```

```
## TARGET1 is 0.03 with cover 2% when
##     Tenure_DISC is 02_DE_1_A_7 or 04_DE_12_A_19
##     Complain is 1
##     PreferedOrderCat is Fashion or Laptop & Accessory
##     SatisfactionScore is 3 or 4 or 5
##     CityTier is 1 or 2
##
## TARGET1 is 0.03 with cover 56% when
##     Tenure_DISC is 02_DE_1_A_7 or 03_DE_7_A_12 or 04_DE_12_A_19 or 05_MAYOR_19
##     Complain is 0
##
## TARGET1 is 0.04 with cover 7% when
##     Tenure_DISC is 02_DE_1_A_7 or 03_DE_7_A_12 or 04_DE_12_A_19 or 05_MAYOR_19
##     Complain is 1
##     SatisfactionScore is 1 or 2
##
## TARGET1 is 0.08 with cover 6% when
##     Tenure_DISC is 03_DE_7_A_12 or 05_MAYOR_19
##     Complain is 1
##     SatisfactionScore is 3 or 4 or 5
##
## TARGET1 is 0.11 with cover 6% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Laptop & Accessory or Mobile or Mobile Phone
##     SatisfactionScore is 1 or 2 or 3 or 4
##     DAYSINCELASTORDER_DISC is 02 <= 8
##     PreferredPaymentMode is Cash on Delivery or CC or Credit Card or Debit Card
or E wallet
##
## TARGET1 is 0.16 with cover 1% when
##     Tenure_DISC is 02_DE_1_A_7 or 04_DE_12_A_19
##     Complain is 1
##     PreferedOrderCat is Fashion or Laptop & Accessory
##     SatisfactionScore is 3 or 4 or 5
##     DAYSINCELASTORDER_DISC is 02 <= 8 or 03 > 8
##     CityTier is 3
##
## TARGET1 is 0.17 with cover 1% when
##     Tenure_DISC is 02_DE_1_A_7 or 04_DE_12_A_19
##     Complain is 1
##     PreferedOrderCat is Grocery or Mobile or Mobile Phone or Others
##     SatisfactionScore is 3 or 4 or 5
##     PreferredPaymentMode is Cash on Delivery or COD or Credit Card or UPI
##
## TARGET1 is 0.18 with cover 0% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Laptop & Accessory or Mobile or Mobile Phone or Others
##     DAYSINCELASTORDER_DISC is 01 <= 1 or 03 > 8
##     PreferredPaymentMode is CC or Credit Card
##     CityTier is 2 or 3
##
## TARGET1 is 0.18 with cover 0% when
```

```
##     Tenure_DISC is 01_MENOR_1
##     Complain is 1
##     MaritalStatus is Divorced or Married
##     NUMBEROFDEVICEREGISTERED_DISC is 01 <= 2
##
## TARGET1 is 0.21 with cover 1% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 1
##     PreferedOrderCat is Fashion or Laptop & Accessory or Mobile
##     SatisfactionScore is 1 or 2 or 3 or 4
##     PreferredPaymentMode is Cash on Delivery or COD or E wallet or UPI
##     MaritalStatus is Divorced or Married
##     NUMBEROFDEVICEREGISTERED_DISC is 02 <= 4 or 03 > 4
##
## TARGET1 is 0.21 with cover 1% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Mobile
##     DAYSINCELASTORDER_DISC is 01 <= 1 or 03 > 8
##     CityTier is 1
##
## TARGET1 is 0.24 with cover 1% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Laptop & Accessory or Mobile Phone
##     DAYSINCELASTORDER_DISC is 01 <= 1 or 03 > 8
##     CityTier is 1
##     PreferredLoginDevice is Computer
##
## TARGET1 is 0.27 with cover 2% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     SatisfactionScore is 5
##     DAYSINCELASTORDER_DISC is 02 <= 8
##     PreferredPaymentMode is CC or Credit Card or Debit Card or UPI
##     CityTier is 1
##
## TARGET1 is 0.30 with cover 1% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Laptop & Accessory or Mobile or Mobile Phone
##     SatisfactionScore is 1 or 2 or 3 or 4
##     DAYSINCELASTORDER_DISC is 02 <= 8
##     PreferredPaymentMode is COD or UPI
##     PreferredLoginDevice is Mobile Phone or Phone
##
## TARGET1 is 0.47 with cover 1% when
##     Tenure_DISC is 02_DE_1_A_7 or 04_DE_12_A_19
##     Complain is 1
##     PreferedOrderCat is Grocery or Mobile or Mobile Phone or Others
##     SatisfactionScore is 3 or 4 or 5
##     PreferredPaymentMode is CC or Debit Card or E wallet
##     CityTier is 1
##
## TARGET1 is 0.65 with cover 1% when
```

```
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Laptop & Accessory or Mobile Phone
##     DAYSINCELASTORDER_DISC is 01 <= 1 or 03 > 8
##     CityTier is 1
##     PreferredLoginDevice is Mobile Phone or Phone
##
## TARGET1 is 0.67 with cover 0% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 1
##     PreferedOrderCat is Mobile Phone
##     SatisfactionScore is 1 or 2 or 3 or 4
##     PreferredPaymentMode is Cash on Delivery or COD or E wallet or UPI
##     MaritalStatus is Divorced or Married
##     NUMBEROFDEVICEREGISTERED_DISC is 02 <= 4 or 03 > 4
##
## TARGET1 is 0.73 with cover 0% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Fashion
##     SatisfactionScore is 1 or 2 or 3 or 4
##     DAYSINCELASTORDER_DISC is 02 <= 8
##
## TARGET1 is 0.74 with cover 1% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     SatisfactionScore is 5
##     DAYSINCELASTORDER_DISC is 02 <= 8
##     CityTier is 3
##
## TARGET1 is 0.75 with cover 3% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 1
##     PreferredPaymentMode is CC or Credit Card or Debit Card
##     MaritalStatus is Divorced or Married
##     NUMBEROFDEVICEREGISTERED_DISC is 02 <= 4 or 03 > 4
##
## TARGET1 is 0.81 with cover 2% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Laptop & Accessory or Mobile or Mobile Phone or Others
##     DAYSINCELASTORDER_DISC is 01 <= 1 or 03 > 8
##     PreferredPaymentMode is Cash on Delivery or COD or Debit Card or E wallet or
UPI
##     CityTier is 2 or 3
##
## TARGET1 is 0.82 with cover 0% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Laptop & Accessory or Mobile or Mobile Phone
##     SatisfactionScore is 1 or 2 or 3 or 4
##     DAYSINCELASTORDER_DISC is 02 <= 8
##     PreferredPaymentMode is COD or UPI
##     PreferredLoginDevice is Computer
##
```

```
## TARGET1 is 0.82 with cover 0% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 1
##     SatisfactionScore is 5
##     PreferredPaymentMode is Cash on Delivery or COD or E wallet or UPI
##     MaritalStatus is Divorced or Married
##     NUMBEROFDEVICEREGISTERED_DISC is 02 <= 4 or 03 > 4
##
## TARGET1 is 0.88 with cover 4% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 1
##     MaritalStatus is Single
##
## TARGET1 is 0.89 with cover 0% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     SatisfactionScore is 5
##     DAYSINCELASTORDER_DISC is 02 <= 8
##     PreferredPaymentMode is Cash on Delivery or COD
##     CityTier is 1
##
## TARGET1 is 1.00 with cover 0% when
##     Tenure_DISC is 02_DE_1_A_7 or 04_DE_12_A_19
##     Complain is 1
##     PreferedOrderCat is Fashion or Laptop & Accessory
##     SatisfactionScore is 3 or 4 or 5
##     DAYSINCELASTORDER_DISC is 01 <= 1
##     CityTier is 3
##
## TARGET1 is 1.00 with cover 0% when
##     Tenure_DISC is 02_DE_1_A_7 or 04_DE_12_A_19
##     Complain is 1
##     PreferedOrderCat is Grocery or Mobile or Mobile Phone or Others
##     SatisfactionScore is 3 or 4 or 5
##     PreferredPaymentMode is CC or Debit Card or E wallet
##     CityTier is 3
##
## TARGET1 is 1.00 with cover 0% when
##     Tenure_DISC is 01_MENOR_1
##     Complain is 0
##     PreferedOrderCat is Fashion
##     DAYSINCELASTORDER_DISC is 01 <= 1 or 03 > 8
```

Podemos llevarnos el nodo final de cada cliente a un data.frame para poder hacer una explotacion posterior.

```
ar_numnodos<-rpart.predict(ar,test,nn = T)
head(ar_numnodos)
```

```
##           0          1  nn
## 1 0.9624060 0.03759398  10
## 2 0.2647059 0.73529412  51
## 3 0.1940299 0.80597015 107
## 4 0.9672876 0.03271240   4
## 5 0.9249012 0.07509881  22
## 6 0.1764706 0.82352941 117
```

Nos da una tabla donde vemos por cliente, la probabilidad de que no abandone (0) nuestra empresa, y la probabilidad de que si lo haga (1) y el número de nodo en el que identificamos a ese cliente.

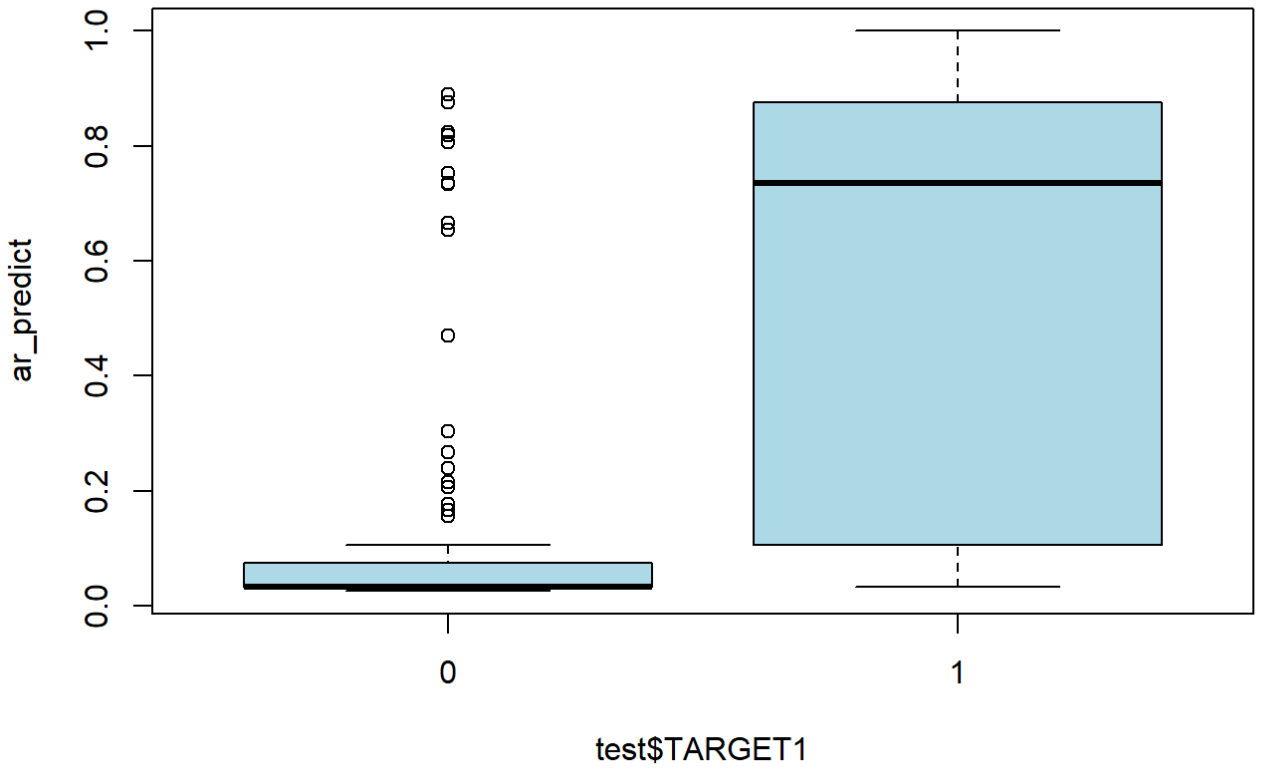
Vamos a calcular los scorings y evaluar el modelo.

```
ar_predict<-predict(ar,test,type = 'prob')[,2]
head(ar_predict)
```

```
##          1          2          3          4          5          6
## 0.03759398 0.73529412 0.80597015 0.03271240 0.07509881 0.82352941
```

Vemos el gráfico boxplot para evaluarlo:

```
plot(ar_predict~test$TARGET1, col='lightblue')
```



Con la función umbrales probamos diferentes cortes.

```
umb_ar<-umbrales(test$TARGET1,ar_predict)
umb_ar
```

```
##    umbral  acierto precision cobertura       F1
## 1    0.05 74.29599  39.50233 86.394558 54.21558
## 2    0.10 79.32894  45.14286 80.612245 57.87546
## 3    0.15 83.46315  52.13270 74.829932 61.45251
## 4    0.20 85.32055  56.71233 70.408163 62.82246
## 5    0.25 86.27921  60.65574 62.925170 61.76962
## 6    0.30 87.47753  65.79926 60.204082 62.87744
## 7    0.35 88.07669  68.62745 59.523810 63.75228
## 8    0.40 88.07669  68.62745 59.523810 63.75228
## 9    0.45 88.07669  68.62745 59.523810 63.75228
## 10   0.50 88.25644  70.41667 57.482993 63.29588
## 11   0.55 88.25644  70.41667 57.482993 63.29588
## 12   0.60 88.25644  70.41667 57.482993 63.29588
## 13   0.65 88.25644  70.41667 57.482993 63.29588
## 14   0.70 88.37627  73.36449 53.401361 61.81102
## 15   0.75 88.13661  75.53191 48.299320 58.92116
## 16   0.80 86.75854  76.64234 35.714286 48.72390
## 17   0.85 85.56022  77.89474 25.170068 38.04627
## 18   0.90 83.34332 100.00000  5.442177 10.32258
## 19   0.95 83.34332 100.00000  5.442177 10.32258
```

Seleccionamos automáticamente el mejor umbral.

```
umbral_final_ar<-umb_ar[which.max(umb_ar$F1),1]
umbral_final_ar
```

```
## [1] 0.35
```

Evaluamos la matriz de confusión y las métricas con el umbral optimizado.

```
confusion(test$TARGET1,ar_predict,umbral_final_ar)
```

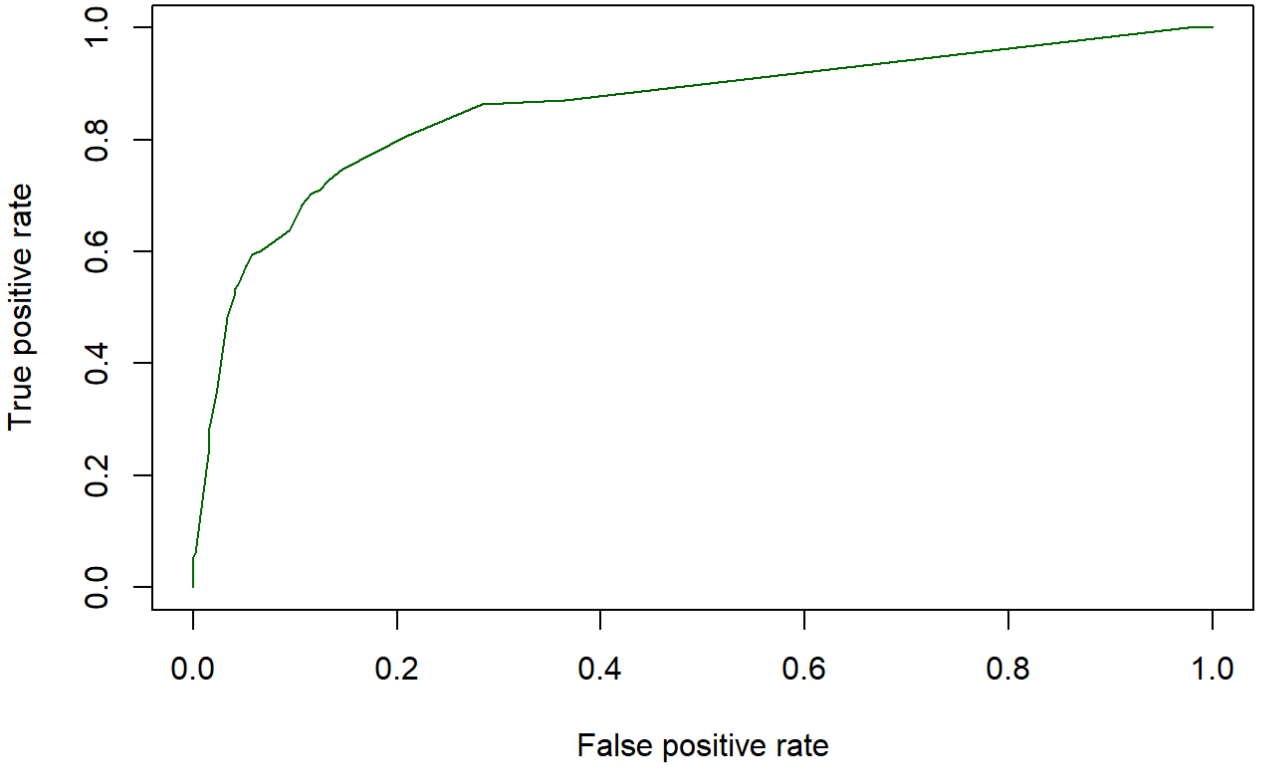
```
##
## real FALSE TRUE
##    0  1295   80
##    1   119  175
```

```
ar_metricas<-filter(umb_ar,umbral==umbral_final_ar)
ar_metricas
```

```
##   umbral  acierto precision cobertura       F1
## 1   0.35 88.07669  68.62745  59.52381 63.75228
```

Evaluamos la ROC.

```
ar_prediction<-prediction(ar_predict,test$TARGET1)
roc(ar_prediction)
```



Sacamos las métricas definitivas incluyendo el AUC.

```
ar_metricas<-cbind(ar_metricas,AUC=round(auc(ar_prediction),2)*100)
print(t(ar_metricas))
```

```
##                [,1]
## umbral      0.35000
## acierto    88.07669
## precision  68.62745
## cobertura  59.52381
## F1         63.75228
## AUC        86.00000
```

- 4.5.6 - Modelizamos con Random Forest

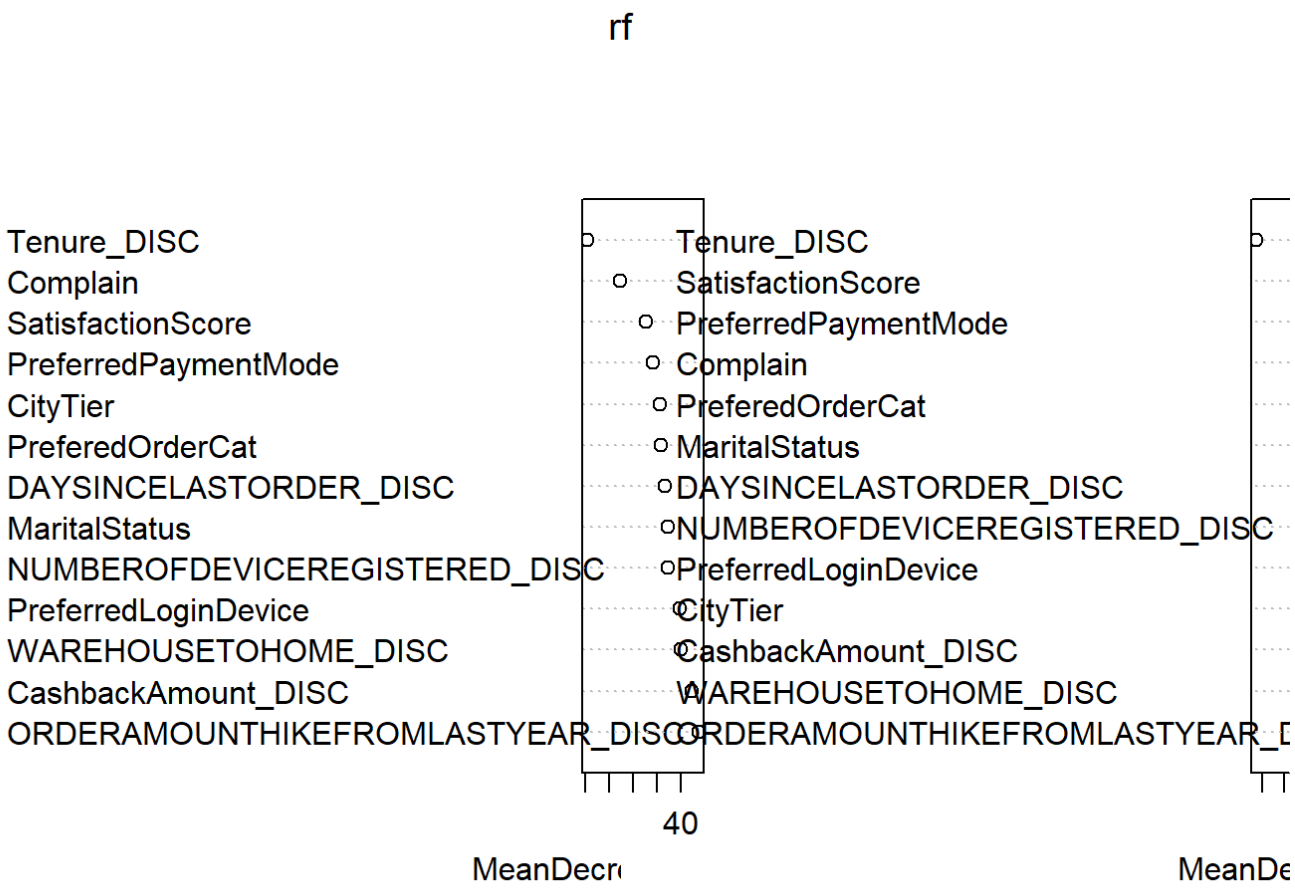
Creamos el modelo

```
formula_rf <- formula
rf<-randomForest(formula_rf,train,importance=T)
rf
```

```
##
## Call:
##  randomForest(formula = formula_rf, data = train, importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 5.99%
## Confusion matrix:
##      0   1 class.error
## 0 3246  57  0.01725704
## 1  180 474  0.27522936
```

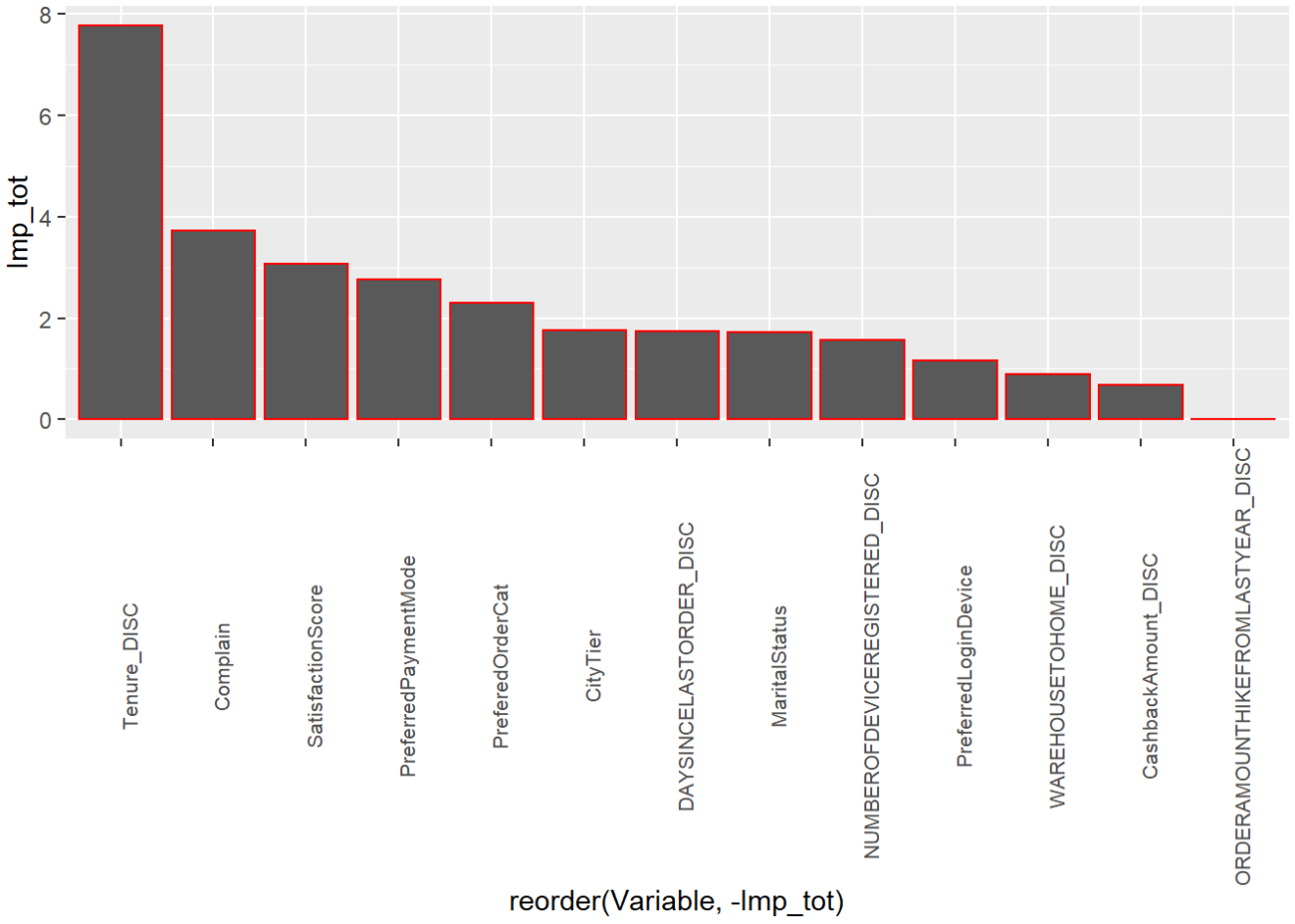
Visualizamos las variables más importantes.

```
varImpPlot(rf)
```



Como hay dos criterios vamos a crear una única variable agregada y visualizarla para tener una mejor idea de la importancia de cada variable. Lo vemos en la gráfica y obtenemos la tabla con los valores.

```
importancia <- importance(rf)[,3:4]
importancia_norm <- as.data.frame(scale(importancia))
importancia_norm <- importancia_norm %>% mutate(
  Variable = rownames(importancia_norm),
  Imp_tot = MeanDecreaseAccuracy + MeanDecreaseGini) %>%
  mutate(Imp_tot = Imp_tot + abs(min(Imp_tot))) %>%
  arrange(desc(Imp_tot)) %>%
  select(Variable,Imp_tot,MeanDecreaseAccuracy,MeanDecreaseGini)
ggplot(importancia_norm, aes(reorder(Variable,-Imp_tot),Imp_tot)) + geom_bar(stat =
"identity", col='red') + theme(axis.text.x = element_text(angle = 90,size = 8))
```




```
importancia_norm
```

```
##                                                          Variable   Imp_tot
## Tenure_DISC                                           Tenure_DISC 7.7678392
## Complain                                                 Complain 3.7303178
## SatisfactionScore                                 SatisfactionScore 3.0694100
## PreferredPaymentMode                           PreferredPaymentMode 2.7565747
## PreferedOrderCat                                   PreferedOrderCat 2.2922722
## CityTier                                                 CityTier 1.7505796
## DAYSINCELASTORDER_DISC                     DAYSINCELASTORDER_DISC 1.7364986
## MaritalStatus                                       MaritalStatus 1.7230170
## NUMBEROFDEVICEREGISTERED_DISC       NUMBEROFDEVICEREGISTERED_DISC 1.5681142
## PreferredLoginDevice                         PreferredLoginDevice 1.1605651
## WAREHOUSETOHOME_DISC                         WAREHOUSETOHOME_DISC 0.8905847
## CashbackAmount_DISC                           CashbackAmount_DISC 0.6802230
## ORDERAMOUNTHIKEFROMLASTYEAR_DISC ORDERAMOUNTHIKEFROMLASTYEAR_DISC 0.0000000
##                                  MeanDecreaseAccuracy MeanDecreaseGini
## Tenure_DISC                                 2.44770996       3.07966802
## Complain                                    1.33294215       0.15691443
## SatisfactionScore                           0.45527519       0.37367357
## PreferredPaymentMode                        0.25291877       0.26319471
## PreferedOrderCat                           -0.03151031       0.08332125
## CityTier                                   -0.01125141      -0.47863021
## DAYSINCELASTORDER_DISC                     -0.16293174      -0.34103087
## MaritalStatus                              -0.26777186      -0.24967239
## NUMBEROFDEVICEREGISTERED_DISC              -0.27330520      -0.39904184
## PreferredLoginDevice                       -0.66498465      -0.41491148
## WAREHOUSETOHOME_DISC                       -0.69755330      -0.65232323
## CashbackAmount_DISC                        -1.07403366      -0.48620464
## ORDERAMOUNTHIKEFROMLASTYEAR_DISC           -1.30550394      -0.93495732
```

La caída es bastante gradual, así que no hay corte claro. Podemos coger por ejemplo hasta WAREHOUSETOHOME_DISC incluido, que tiene una importancia total mayor de 0,8.

```
a_mantener <- importancia_norm %>%
  filter(Imp_tot > 0.8) %>%
  select(Variable)
a_mantener <- as.character((a_mantener$Variable))
```

Creamos de nuevo el modelo con las nuevas variables:

```
formula_rf <- reformulate(a_mantener,target)
rf<-randomForest(formula_rf,train,importance=T)
rf
```

```
##
## Call:
##  randomForest(formula = formula_rf, data = train, importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 5.69%
## Confusion matrix:
##      0   1 class.error
## 0 3240  63  0.01907357
## 1  162 492  0.24770642
```

Aplicamos el modelo al conjunto de test, generando un vector con las probabilidades.

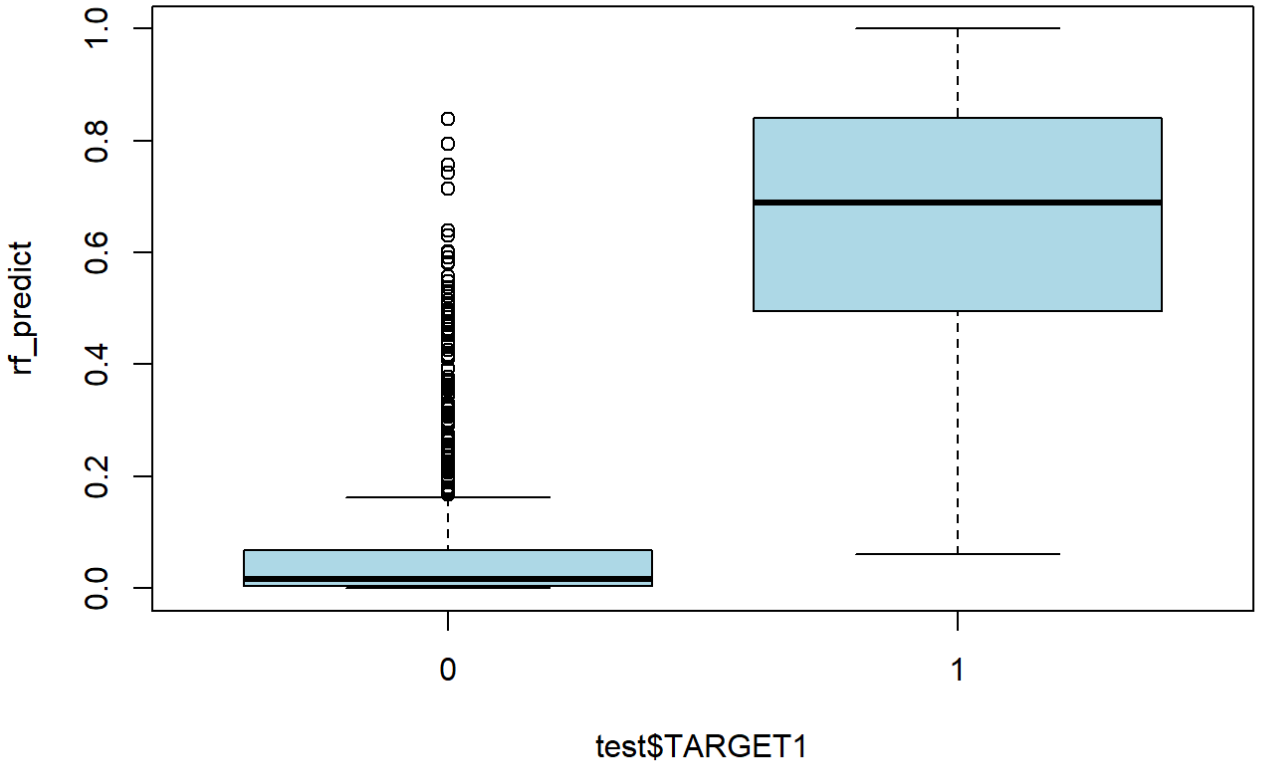
Vemos en la segunda columna de la matriz obtenida la clasificación del error.

```
rf_predict<-predict(rf,test,type = 'prob')[,2]
head(rf_predict)
```

```
##     1     2     3     4     5     6
## 0.260 0.428 0.472 0.492 0.130 0.684
```

Observamos el gráfico:

```
plot(rf_predict~test$TARGET1, col='lightblue')
```



Con la función umbrales probamos diferentes cortes:

```
umb_rf<-umbrales(test$TARGET1,rf_predict)
umb_rf
```

```
##    umbral  acierto precision  cobertura       F1
## 1    0.05 74.95506  41.29213 100.000000 58.44930
## 2    0.10 83.76273  52.11786  96.258503 67.62246
## 3    0.15 87.83703  60.00000  92.857143 72.89720
## 4    0.20 89.63451  64.43914  91.836735 75.73633
## 5    0.25 91.67166  70.55703  90.476190 79.28465
## 6    0.30 92.81007  74.57627  89.795918 81.48148
## 7    0.35 93.40923  78.04878  87.074830 82.31511
## 8    0.40 93.82864  81.51815  84.013605 82.74707
## 9    0.45 93.58898  83.51254  79.251701 81.32635
## 10   0.50 93.82864  88.35341  74.829932 81.03131
## 11   0.55 92.92990  91.12150  66.326531 76.77165
## 12   0.60 92.09107  92.63158  59.863946 72.72727
## 13   0.65 91.67166  94.79769  55.782313 70.23555
## 14   0.70 90.29359  94.00000  47.959184 63.51351
## 15   0.75 89.57460  95.45455  42.857143 59.15493
## 16   0.80 88.43619  98.09524  35.034014 51.62907
## 17   0.85 86.39904 100.00000  22.789116 37.11911
## 18   0.90 84.96105 100.00000  14.625850 25.51929
## 19   0.95 83.58298 100.00000   6.802721 12.73885
```

Seleccionamos automáticamente el mejor umbral.

```
umbral_final_rf<-umb_rf[which.max(umb_rf$F1),1]
umbral_final_rf
```

```
## [1] 0.4
```

Evaluamos la matriz de confusión y las métricas con el umbral optimizado.

```
confusion(test$TARGET1,rf_predict,umbral_final_rf)
```

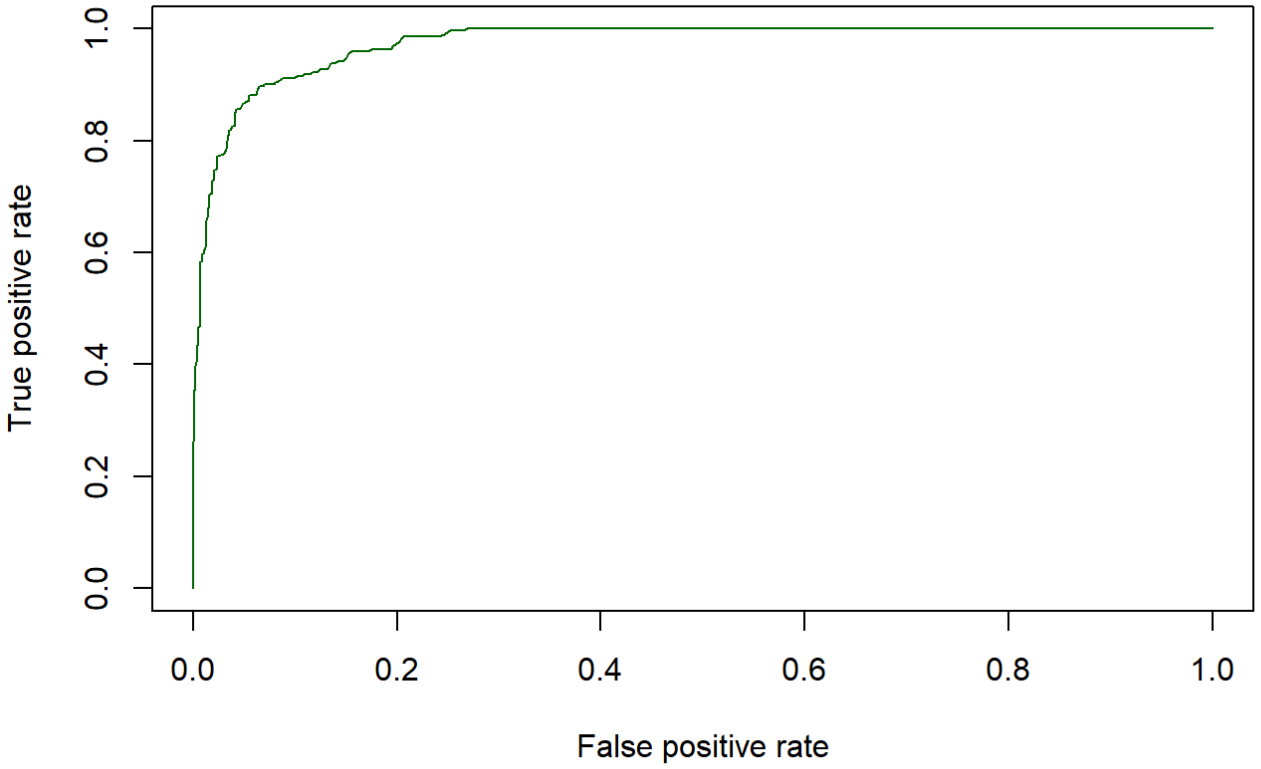
```
##
## real FALSE TRUE
##    0  1319   56
##    1    47  247
```

```
rf_metricas<-filter(umb_rf,umbral==umbral_final_rf)
rf_metricas
```

```
##   umbral  acierto precision cobertura       F1
## 1    0.4 93.82864  81.51815  84.01361 82.74707
```

Evaluamos la ROC

```
rf_prediction<-prediction(rf_predict,test$TARGET1)
roc(rf_prediction)
```



Sacamos las métricas definitivas incluyendo el AUC

```
rf_metricas<-cbind(rf_metricas,AUC=round(auc(rf_prediction),2)*100)
print(t(rf_metricas))
```

```
##                [,1]
## umbral      0.40000
## acierto    93.82864
## precision  81.51815
## cobertura  84.01361
## F1         82.74707
## AUC        97.00000
```

- 4.5.7 - Comparamos los 3 métodos

En esta tabla vemos los valores obtenidos para cada método.

```
comparativa <- rbind(rl_metricas,ar_metricas,rf_metricas)
rownames(comparativa) <- c('Regresion Logistica','Arbol Decision','Random Forest')
t(comparativa)
```

```
##           Regresion Logistica Arbol Decision Random Forest
## umbral                0.40000        0.35000       0.40000
## acierto              86.69862       88.07669      93.82864
## precision            62.76596       68.62745      81.51815
## cobertura            60.20408       59.52381      84.01361
## F1                   61.45833       63.75228      82.74707
## AUC                  86.00000       86.00000      97.00000
```

Como vemos en los valores de AUC, la modelización con Random Forest sería el método más predictivo para este caso.

- 4.5.8 - Escribimos el scoring final en el dataset y guardamos el modelo

```
df$SCORING_CHURN <- predict(rf,df,type = 'prob')[,2]
df %>% select(CustomerID,SCORING_CHURN) %>% arrange(desc(SCORING_CHURN)) %>% slice
(1:30)
```

```
##    CustomerID SCORING_CHURN
## 1       50333         1.000
## 2       51803         1.000
## 3       53148         1.000
## 4       54618         1.000
## 5       52257         0.996
## 6       53583         0.996
## 7       55053         0.996
## 8       55072         0.996
## 9       50787         0.992
## 10      52833         0.992
## 11      53602         0.992
## 12      54023         0.992
## 13      54268         0.992
## 14      55493         0.992
## 15      52838         0.990
## 16      54288         0.990
## 17      50023         0.988
## 18      51473         0.988
## 19      50239         0.986
## 20      51709         0.986
## 21      53379         0.986
## 22      53688         0.986
## 23      54849         0.986
## 24      55158         0.986
## 25      50607         0.984
## 26      51029         0.984
## 27      52077         0.984
## 28      52499         0.984
## 29      52818         0.984
## 30      53402         0.984
```

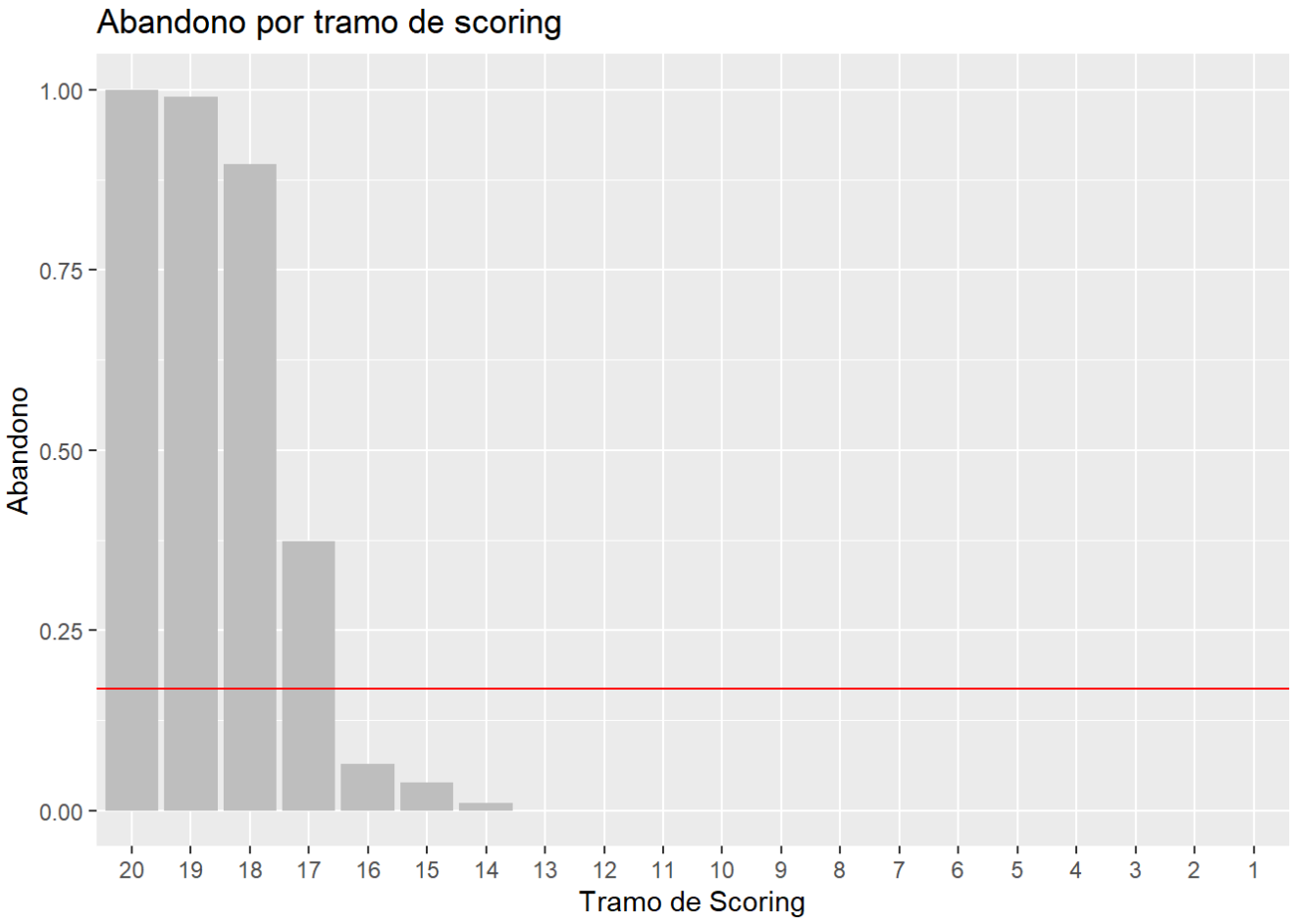
```
saveRDS(rf,'03_modelo_final.rds')
```

```
saveRDS(df,'cache3.rds')
```

4.6 - Evaluación y análisis de negocio

Vamos a visualizar la contratación real por tramos de scoring. Este gráfico es importante para ver que el modelo es consistente, ya que debe presentar una linea descendente en la tasa de no abandono conforme se desciende en el scoring.

```
vis <- function(scoring,real) {
    vis_df <- data.frame(Scoring = scoring, Perc_Scoring = ntile(scoring, 20), Real
= real)
    levels(vis_df$Perc_Scoring) <- seq(from = 100,to = 5,by = -5)
    vis_gr <- vis_df %>% group_by(Perc_Scoring) %>% summarise(Tasa_Contr = mean(as.
numeric(as.character(Real)))) %>% arrange(Perc_Scoring)
    vis_gr$Perc_Scoring <- factor(vis_gr$Perc_Scoring, levels = vis_gr$Perc_Scoring
[order(vis_gr$Perc_Scoring, decreasing = T)])
    ggplot(vis_gr,aes(Perc_Scoring, Tasa_Contr)) +
      geom_col(fill='grey') +
      geom_hline(aes(yintercept = mean(as.numeric(as.character(vis_df$Real)))), col
= 'red') +
      labs(title = 'Abandono por tramo de scoring', x = 'Tramo de Scoring', y = 'Ab
andono')
}
vis(df$SCORING_CHURN,df$TARGET1)
```



Vamos a calcular un tamaño máximo de campaña para evitar el abanadono de la compañía, con la premisa de que resulte rentable, teniendo en cuenta el ingreso medio previsto y el coste medio por accion comercial, que será una campaña de email ya que al tratarse de un comercio electrónico, disponemos de las direcciones de correo de todos los clientes.

Según los datos que nos ha facilitado el departamento de marketing el margen medio por cliente que no abandona la compañía de 40 €.

El coste medio por acción comercial (cliente contactado por email) = 1,20 €.

Vamos a calcular: - Margen esperado = probabilidad de evento * margen evento - Margen neto= margen esperado - coste medio

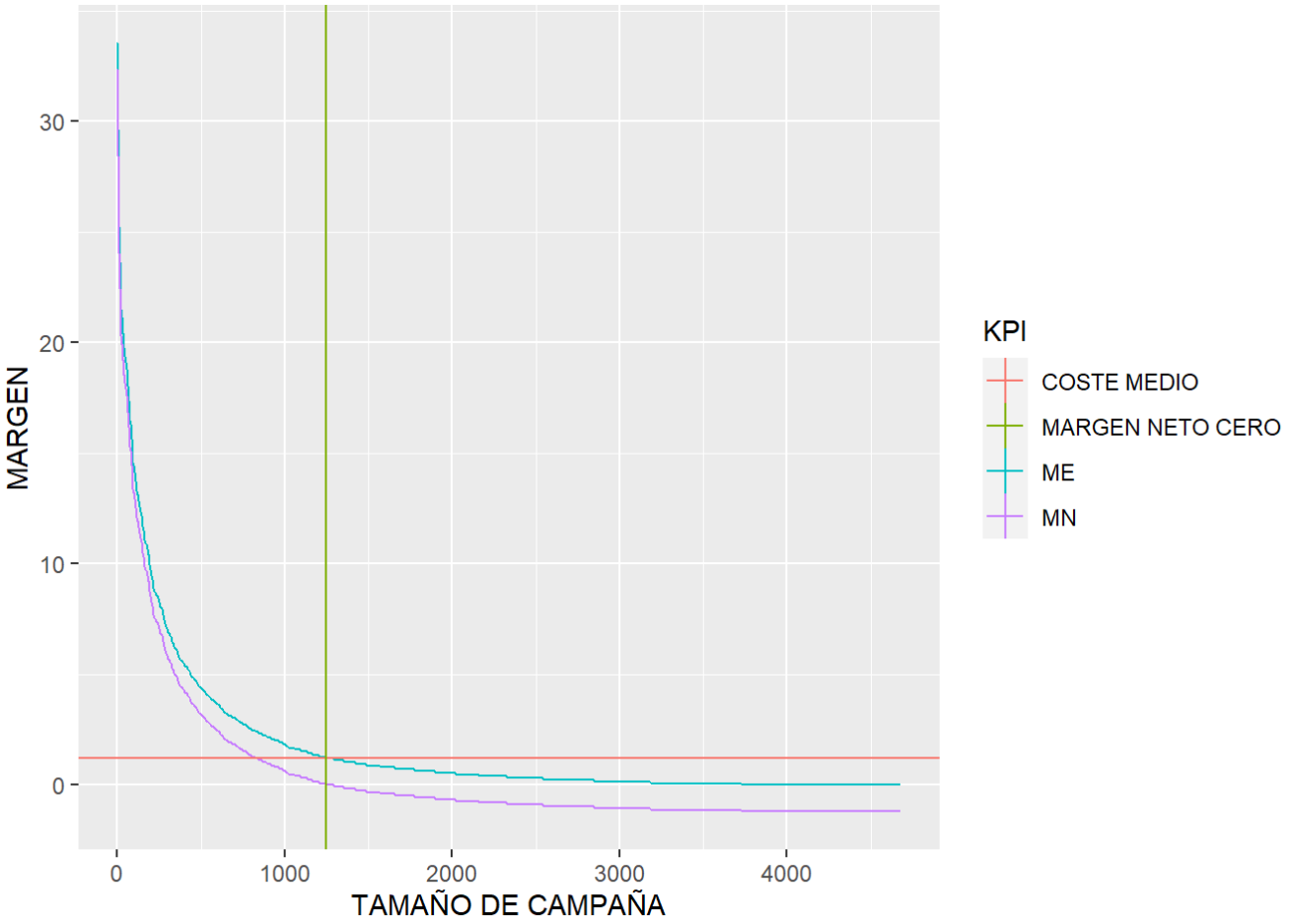
Con estos datos podemos calcular el punto de equilibrio, en el que los ingresos van a ser igual que los gastos y por tanto el margen neto generado por la acción comercial va a ser cero (y a partir de ahí comenzaríamos a perder dinero):

```
mar_medio <- 40
coste_medio <- 1.20
df_campaña <- df %>%
  filter(TARGET1==0) %>%
  mutate(
    ME = SCORING_CHURN * mar_medio,
    MN = ME - coste_medio) %>%
  arrange(desc(MN)) %>%
  mutate(INDICE = 1:nrow(.)) %>%
  select(CustomerID,INDICE,ME,MN)
head(df_campaña,50)
```

```
##    CustomerID INDICE    ME    MN
## 1       53361      1 33.52 32.32
## 2       54831      2 33.52 32.32
## 3       54558      3 31.76 30.56
## 4       51312      4 30.24 29.04
## 5       52782      5 30.24 29.04
## 6       53088      6 30.24 29.04
## 7       51762      7 29.68 28.48
## 8       53888      8 28.56 27.36
## 9       55358      9 28.56 27.36
## 10      53544     10 25.68 24.48
## 11      55014     11 25.68 24.48
## 12      50771     12 25.60 24.40
## 13      50245     13 25.20 24.00
## 14      51715     14 25.20 24.00
## 15      51660     15 24.08 22.88
## 16      54475     16 24.08 22.88
## 17      52205     17 24.00 22.80
## 18      52258     18 23.68 22.48
## 19      50194     19 23.28 22.08
## 20      51664     20 23.28 22.08
## 21      51349     21 23.20 22.00
## 22      53687     22 22.32 21.12
## 23      54331     23 21.92 20.72
## 24      54215     24 21.52 20.32
## 25      52173     25 21.44 20.24
## 26      55157     26 21.28 20.08
## 27      51369     27 21.20 20.00
## 28      51013     28 21.12 19.92
## 29      51743     29 21.12 19.92
## 30      52483     30 21.12 19.92
## 31      52275     31 20.96 19.76
## 32      51027     32 20.88 19.68
## 33      50381     33 20.56 19.36
## 34      53178     34 20.56 19.36
## 35      54648     35 20.56 19.36
## 36      53147     36 20.40 19.20
## 37      54164     37 20.24 19.04
## 38      50196     38 19.92 18.72
## 39      50292     39 19.92 18.72
## 40      51666     40 19.92 18.72
## 41      50332     41 19.84 18.64
## 42      54799     42 19.76 18.56
## 43      51796     43 19.60 18.40
## 44      53011     44 19.60 18.40
## 45      53580     45 19.60 18.40
## 46      54481     46 19.60 18.40
## 47      54611     47 19.60 18.40
## 48      51340     48 19.36 18.16
## 49      52810     49 19.36 18.16
## 50      50691     50 19.12 17.92
```

Visualizamos las curvas obtenidas:

```
MN_cero <- df_campaña %>% filter(MN <= 0 ) %>% slice(1) %>% select(INDICE)
MN_cero <- MN_cero$INDICE
ggplot(df_campaña,aes(x = INDICE)) +
    geom_line(aes(y = ME, col = "ME")) +
    geom_line(aes(y = MN, col = "MN")) +
    geom_hline(aes(yintercept = coste_medio, col = 'COSTE MEDIO')) +
    geom_vline(aes(xintercept = MN_cero, col = 'MARGEN NETO CERO')) +
    labs(x = 'TAMAÑO DE CAMPAÑA', y = 'MARGEN', colour = 'KPI')
```




```
print(paste('Tamaño maximo de campaña rentable: ',MN_cero))
```

```
## [1] "Tamaño maximo de campaña rentable:  1245"
```

Este máximo de campaña rentable, corresponde al punto en el que el margen neto pasa a ser cero o menos.

Ahora vamos a calcular el punto óptimo de retorno de la inversión.

Para ello generamos dos nuevas variables que sean un agregado de los ingresos agregados y de los gastos agregados en cada potencial tamaño de campaña, y el ROI como diferencia de las anteriores, y vamos a localizar el tamaño de la campaña que va a maximizar ese ROI (Retorno óptimo de la inversión) y también cuanto vamos a ganar previsiblemente

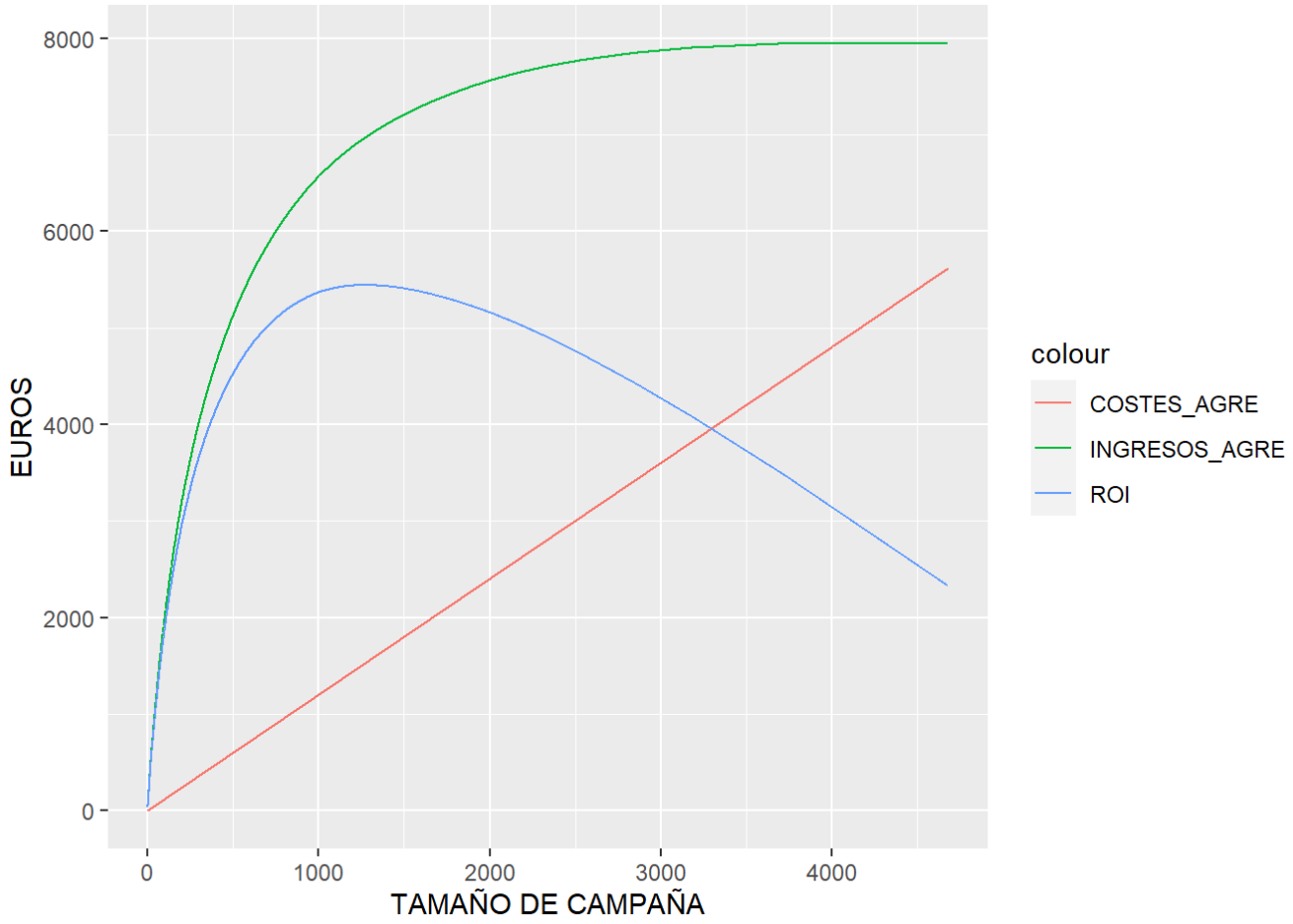
```
df_campaña <- df_campaña %>%
  mutate(
    INGRESOS_AGRE = cumsum(ME),
    COSTES_AGRE = INDICE * coste_medio,
    ROI = INGRESOS_AGRE - COSTES_AGRE)
head(df_campaña,50)
```

```
##    CustomerID INDICE    ME    MN INGRESOS_AGRE COSTES_AGRE     ROI
## 1       53361      1 33.52 32.32         33.52         1.2   32.32
## 2       54831      2 33.52 32.32         67.04         2.4   64.64
## 3       54558      3 31.76 30.56         98.80         3.6   95.20
## 4       51312      4 30.24 29.04        129.04         4.8  124.24
## 5       52782      5 30.24 29.04        159.28         6.0  153.28
## 6       53088      6 30.24 29.04        189.52         7.2  182.32
## 7       51762      7 29.68 28.48        219.20         8.4  210.80
## 8       53888      8 28.56 27.36        247.76         9.6  238.16
## 9       55358      9 28.56 27.36        276.32        10.8  265.52
## 10      53544     10 25.68 24.48        302.00        12.0  290.00
## 11      55014     11 25.68 24.48        327.68        13.2  314.48
## 12      50771     12 25.60 24.40        353.28        14.4  338.88
## 13      50245     13 25.20 24.00        378.48        15.6  362.88
## 14      51715     14 25.20 24.00        403.68        16.8  386.88
## 15      51660     15 24.08 22.88        427.76        18.0  409.76
## 16      54475     16 24.08 22.88        451.84        19.2  432.64
## 17      52205     17 24.00 22.80        475.84        20.4  455.44
## 18      52258     18 23.68 22.48        499.52        21.6  477.92
## 19      50194     19 23.28 22.08        522.80        22.8  500.00
## 20      51664     20 23.28 22.08        546.08        24.0  522.08
## 21      51349     21 23.20 22.00        569.28        25.2  544.08
## 22      53687     22 22.32 21.12        591.60        26.4  565.20
## 23      54331     23 21.92 20.72        613.52        27.6  585.92
## 24      54215     24 21.52 20.32        635.04        28.8  606.24
## 25      52173     25 21.44 20.24        656.48        30.0  626.48
## 26      55157     26 21.28 20.08        677.76        31.2  646.56
## 27      51369     27 21.20 20.00        698.96        32.4  666.56
## 28      51013     28 21.12 19.92        720.08        33.6  686.48
## 29      51743     29 21.12 19.92        741.20        34.8  706.40
## 30      52483     30 21.12 19.92        762.32        36.0  726.32
## 31      52275     31 20.96 19.76        783.28        37.2  746.08
## 32      51027     32 20.88 19.68        804.16        38.4  765.76
## 33      50381     33 20.56 19.36        824.72        39.6  785.12
## 34      53178     34 20.56 19.36        845.28        40.8  804.48
## 35      54648     35 20.56 19.36        865.84        42.0  823.84
## 36      53147     36 20.40 19.20        886.24        43.2  843.04
## 37      54164     37 20.24 19.04        906.48        44.4  862.08
## 38      50196     38 19.92 18.72        926.40        45.6  880.80
## 39      50292     39 19.92 18.72        946.32        46.8  899.52
## 40      51666     40 19.92 18.72        966.24        48.0  918.24
## 41      50332     41 19.84 18.64        986.08        49.2  936.88
## 42      54799     42 19.76 18.56       1005.84        50.4  955.44
## 43      51796     43 19.60 18.40       1025.44        51.6  973.84
## 44      53011     44 19.60 18.40       1045.04        52.8  992.24
## 45      53580     45 19.60 18.40       1064.64        54.0 1010.64
## 46      54481     46 19.60 18.40       1084.24        55.2 1029.04
## 47      54611     47 19.60 18.40       1103.84        56.4 1047.44
## 48      51340     48 19.36 18.16       1123.20        57.6 1065.60
## 49      52810     49 19.36 18.16       1142.56        58.8 1083.76
## 50      50691     50 19.12 17.92       1161.68        60.0 1101.68
```

Conocido el punto de equilibrio ya podemos crear la curva agregada de la campaña, que nos va a permitir calcular su tamaño óptimo:

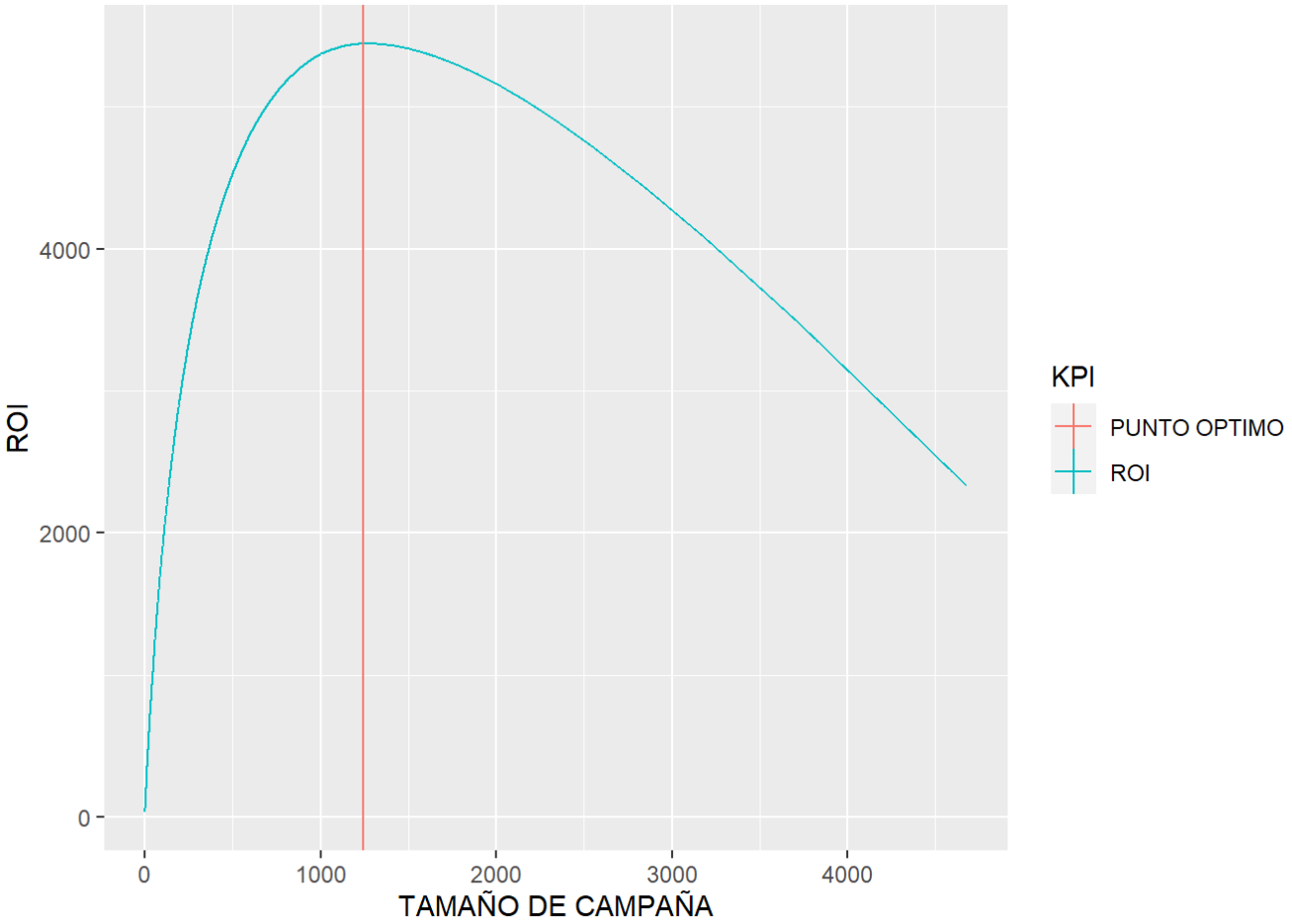
Visualizamos las curvas

```
ggplot(df_campaña,aes(x = INDICE)) +
  geom_line(aes(y = INGRESOS_AGRE, col='INGRESOS_AGRE')) +
  geom_line(aes(y = COSTES_AGRE, col='COSTES_AGRE')) +
  geom_line(aes(y = ROI, col='ROI')) +
  labs(y='EUROS', x = 'TAMAÑO DE CAMPAÑA')
```



Y ahora vamos a visualizar un zoom sobre el ROI solo en los tamaños de campaña que son positivos para localizar el punto optimo

```
df_campaña %>%
  filter(ROI > 0) %>%
  ggplot(aes(x = INDICE)) +
  geom_line(aes(y = ROI, col='ROI')) +
  geom_vline(aes(xintercept = MN_cero, col = 'PUNTO OPTIMO')) +
  labs(x = 'TAMAÑO DE CAMPAÑA', y = 'ROI', colour = 'KPI')
```



Finalmente, este es el resultado obtenido:

```
cat(
  paste(
    'El tamaño óptimo de campaña para el ROI es de:', MN_cero, 'clientes',
    '\nCon unos ingresos esperados de margen neto acumulado de:',   round(df_campañ
a[which(df_campaña$INDICE == MN_cero),'INGRESOS_AGRE']), '€',
    '\nY unos costes agregados de:',
    df_campaña[which(df_campaña$INDICE == MN_cero),'COSTES_AGRE'], '€',
    '\nQue van a generar un Retorno Neto de la Inversión de:',
    round(df_campaña[which(df_campaña$INDICE == MN_cero),'ROI']),'€'
    )
  )
```

```
## El tamaño óptimo de campaña para el ROI es de: 1245 clientes
## Con unos ingresos esperados de margen neto acumulado de: 6938 €
## Y unos costes agregados de: 1494 €
## Que van a generar un Retorno Neto de la Inversión de: 5444 €
```

Vemos el fichero generado con los datos de los clientes, para poder realizar la campaña de marketing.

```
head(df_campaña)
```

```
##   CustomerID INDICE    ME    MN INGRESOS_AGRE COSTES_AGRE    ROI
## 1      53361      1 33.52 32.32         33.52         1.2  32.32
## 2      54831      2 33.52 32.32         67.04         2.4  64.64
## 3      54558      3 31.76 30.56         98.80         3.6  95.20
## 4      51312      4 30.24 29.04        129.04         4.8 124.24
## 5      52782      5 30.24 29.04        159.28         6.0 153.28
## 6      53088      6 30.24 29.04        189.52         7.2 182.32
```

5 Conclusiones

Se ha trabajado sobre un histórico de clientes y se ha obtenido un modelo predictivo de alta calidad.

El modelo es estable y consigue plasmar las características de los clientes que no abandonarán la compañía.

1. LIMITAR EL TAMAÑO DE CAMPAÑA: El **tamaño óptimo de campaña,** el ROI es de: 1.245 clientes
2. CON LO QUE SE GENERARÁN UNOS **ingresos esperados** de margen neto acumulado de: 6.938 €
3. DISMINUYENDO LOS **costes agregados** HASTA: 1.494 €
4. Y PRODUCIENDO UN **Retorno Neto de la Inversión** de: 5.444 €

6 Resultados

Nuestra previsión es que usando este modelo se puede conseguir un ROI en la próxima campaña de 5.444 €.

En conjunto con el departamento de marketing se diseñará un campaña de fidelización mediante email dirigida a estos clientes que fomentará su conformidad con la empresa, y a su vez disminuirá la tasa de abanadono de la misma.