Predicción de abandono de clientes en compañía de telecomunicaciones

Machine Learning Predictivo

Beatriz Cubillas García

26/01/2022

---

1 ID Petición

- Número 1: Predicción de los casos de abandono de clientes 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',#para leer y escribir datos de forma rapida
              '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 '.csv'.

```
df <- fread('datos1.csv')
```

4.2 - Análisis exploratorio

- 4.2.1 - Análisis exploratorio general y tipo de datos

```
glimpse(df)
```

```
## Rows: 7,043
## Columns: 21
## $ customerID       <chr> "7590-VHVEG", "5575-GNVDE", "3668-QPYBK", "7795-CFOCW~
## $ gender           <chr> "Female", "Male", "Male", "Male", "Female", "Female",~
## $ SeniorCitizen    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Partner          <chr> "Yes", "No", "No", "No", "No", "No", "No", "No", "Yes~
## $ Dependents       <chr> "No", "No", "No", "No", "No", "No", "Yes", "No", "No"~
## $ tenure           <int> 1, 34, 2, 45, 2, 8, 22, 10, 28, 62, 13, 16, 58, 49, 2~
## $ PhoneService     <chr> "No", "Yes", "Yes", "No", "Yes", "Yes", "Yes", "No", ~
## $ MultipleLines    <chr> "No phone service", "No", "No", "No phone service", "~
## $ InternetService  <chr> "DSL", "DSL", "DSL", "DSL", "Fiber optic", "Fiber opt~
## $ OnlineSecurity   <chr> "No", "Yes", "Yes", "Yes", "No", "No", "No", "Yes", "~
## $ OnlineBackup     <chr> "Yes", "No", "Yes", "No", "No", "No", "Yes", "No", "N~
## $ DeviceProtection <chr> "No", "Yes", "No", "Yes", "No", "Yes", "No", "No", "Y~
## $ TechSupport      <chr> "No", "No", "No", "Yes", "No", "No", "No", "No", "Yes~
## $ StreamingTV      <chr> "No", "No", "No", "No", "No", "Yes", "Yes", "No", "Ye~
## $ StreamingMovies  <chr> "No", "No", "No", "No", "No", "Yes", "No", "No", "Yes~
## $ Contract         <chr> "Month-to-month", "One year", "Month-to-month", "One ~
## $ PaperlessBilling <chr> "Yes", "No", "Yes", "No", "Yes", "Yes", "Yes", "No", ~
## $ PaymentMethod    <chr> "Electronic check", "Mailed check", "Mailed check", "~
## $ MonthlyCharges   <dbl> 29.85, 56.95, 53.85, 42.30, 70.70, 99.65, 89.10, 29.7~
## $ TotalCharges     <dbl> 29.85, 1889.50, 108.15, 1840.75, 151.65, 820.50, 1949~
## $ Churn            <chr> "No", "No", "Yes", "No", "Yes", "Yes", "No", "No", "Y~
```

Vemos que tenemos 21 variables, de las que dos son numéricas que continuarán siendo números (MonthlyCharge, TotalCharge), una de tipo entero que es la permanencia (tenure) y el resto excepto el identificador de cliente, vamos a pasarlas a factor.

De momento las guardamos en la variable 'a_factores'.

```
a_factores <- c("gender","SeniorCitizen","Partner","Dependents","PhoneService","Multiple
Lines","InternetService","OnlineSecurity","OnlineBackup","DeviceProtection","TechSuppor
t","StreamingTV","StreamingMovies","Contract","PaperlessBilling","PaymentMethod")
a_factores
```

```
##  [1] "gender"           "SeniorCitizen"    "Partner"          "Dependents"
##  [5] "PhoneService"     "MultipleLines"    "InternetService"  "OnlineSecurity"
##  [9] "OnlineBackup"     "DeviceProtection" "TechSupport"      "StreamingTV"
## [13] "StreamingMovies"  "Contract"         "PaperlessBilling" "PaymentMethod"
```

- 4.2.2 - Calidad de datos:

Calculamos los estadísticos básicos de cada variable

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

```
## $customerID
##    Length     Class      Mode
##      7043 character character
##
## $gender
##    Length     Class      Mode
##      7043 character character
##
## $SeniorCitizen
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1621  0.0000  1.0000
##
## $Partner
##    Length     Class      Mode
##      7043 character character
##
## $Dependents
##    Length     Class      Mode
##      7043 character character
##
## $tenure
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    9.00   29.00   32.37   55.00   72.00
##
## $PhoneService
##    Length     Class      Mode
##      7043 character character
##
## $MultipleLines
##    Length     Class      Mode
##      7043 character character
##
## $InternetService
##    Length     Class      Mode
##      7043 character character
##
## $OnlineSecurity
##    Length     Class      Mode
##      7043 character character
##
## $OnlineBackup
##    Length     Class      Mode
##      7043 character character
##
## $DeviceProtection
```

```
##    Length     Class      Mode
##      7043 character character
##
## $TechSupport
##    Length     Class      Mode
##      7043 character character
##
## $StreamingTV
##    Length     Class      Mode
##      7043 character character
##
## $StreamingMovies
##    Length     Class      Mode
##      7043 character character
##
## $Contract
##    Length     Class      Mode
##      7043 character character
##
## $PaperlessBilling
##    Length     Class      Mode
##      7043 character character
##
## $PaymentMethod
##    Length     Class      Mode
##      7043 character character
##
## $MonthlyCharges
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.25   35.50   70.35   64.76   89.85  118.75
##
## $TotalCharges
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    18.8   401.4  1397.5  2283.3  3794.7  8684.8      11
##
## $Churn
##    Length     Class      Mode
##      7043 character character
```

Hemos encontramos 11 nulos en la variable TotalCharges y nos llama la atención la antigüedad que tiene un máximo de 72 meses (6 años) para muchos clientes.

- 4.2.3 - Calidad de datos: Análisis de nulos

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

```
##                  colSums.is.na.df..
## customerID                        0
## gender                            0
## SeniorCitizen                     0
## Partner                           0
## Dependents                        0
## tenure                            0
## PhoneService                      0
## MultipleLines                     0
## InternetService                   0
## OnlineSecurity                    0
## OnlineBackup                      0
## DeviceProtection                  0
## TechSupport                       0
## StreamingTV                       0
## StreamingMovies                   0
## Contract                          0
## PaperlessBilling                  0
## PaymentMethod                     0
## MonthlyCharges                    0
## TotalCharges                     11
## Churn                             0
```

- Confirmamos que hay 11 nulos en la variable 'TotalCharges'

Decidimos eliminar las once filas donde aparecen los nulos ya que supone una perdida de datos mínima en el total de la base de datos.

```
df<-na.omit(df)
```

- 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),stringsA
sFactors = F)
num_ceros <- num_ceros %>%
  arrange(desc(CEROS)) %>%
  mutate(PORCENTAJE = CEROS / nrow(df) * 100)
num_ceros
```

```
##            VARIABLE CEROS PORCENTAJE
## 1     SeniorCitizen  5890   83.75995
## 2        customerID     0    0.00000
## 3            gender     0    0.00000
## 4           Partner     0    0.00000
## 5        Dependents     0    0.00000
## 6            tenure     0    0.00000
## 7      PhoneService     0    0.00000
## 8     MultipleLines     0    0.00000
## 9   InternetService     0    0.00000
## 10   OnlineSecurity     0    0.00000
## 11     OnlineBackup     0    0.00000
## 12 DeviceProtection     0    0.00000
## 13      TechSupport     0    0.00000
## 14      StreamingTV     0    0.00000
## 15  StreamingMovies     0    0.00000
## 16         Contract     0    0.00000
## 17 PaperlessBilling     0    0.00000
## 18    PaymentMethod     0    0.00000
## 19   MonthlyCharges     0    0.00000
## 20     TotalCharges     0    0.00000
## 21            Churn     0    0.00000
```

Vemos que en la variable 'SeniorCitizen' hay una proporción muy alta de ceros, pero eso significa que en el total de nuestros clientes la proporción de personas jubiladas es menor al 17 %, datos que es coherente con la realidad.

- 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
## NULL
##
## $gender
## NULL
##
## $SeniorCitizen
## NULL
##
## $Partner
## NULL
##
## $Dependents
## NULL
##
## $tenure
## NULL
##
## $PhoneService
## NULL
##
## $MultipleLines
## NULL
##
## $InternetService
## NULL
##
## $OnlineSecurity
## NULL
##
## $OnlineBackup
## NULL
##
## $DeviceProtection
## NULL
##
## $TechSupport
## NULL
##
## $StreamingTV
## NULL
##
## $StreamingMovies
## NULL
##
```

```
## $Contract
## NULL
##
## $PaperlessBilling
## NULL
##
## $PaymentMethod
## NULL
##
## $MonthlyCharges
##         [,1]
##  [1,] 118.75
##  [2,] 118.65
##  [3,] 118.60
##  [4,] 118.60
##  [5,] 118.35
##  [6,] 118.20
##  [7,] 117.80
##  [8,] 117.60
##  [9,] 117.50
## [10,] 117.45
## [11,] 117.35
## [12,] 117.20
## [13,] 117.15
## [14,] 116.95
## [15,] 116.85
## [16,] 116.80
## [17,] 116.75
## [18,] 116.60
## [19,] 116.60
## [20,] 116.55
##
## $TotalCharges
##          [,1]
##  [1,] 8684.80
##  [2,] 8672.45
##  [3,] 8670.10
##  [4,] 8594.40
##  [5,] 8564.75
##  [6,] 8547.15
##  [7,] 8543.25
##  [8,] 8529.50
##  [9,] 8496.70
## [10,] 8477.70
## [11,] 8477.60
## [12,] 8476.50
```

```
## [13,] 8468.20
## [14,] 8456.75
## [15,] 8443.70
## [16,] 8436.25
## [17,] 8425.30
## [18,] 8425.15
## [19,] 8424.90
## [20,] 8405.00
##
## $Churn
## NULL
```

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.integer(x)) out(x)
})
```

```
## $customerID
## NULL
##
## $gender
## NULL
##
## $SeniorCitizen
##
## variable [,1]
##        0 5890
##        1 1142
##
## $Partner
## NULL
##
## $Dependents
## NULL
##
## $tenure
##
## variable [,1]
##       1   613
##       2   238
##       3   200
##       4   176
##       5   133
##       6   110
##       7   131
##       8   123
##       9   119
##       10  116
##       11   99
##       12  117
##       13  109
##       14   76
##       15   99
##       16   80
##       17   87
##       18   97
##       19   73
##       20   71
##       21   63
##       22   90
##       23   85
##       24   94
```

```
##       25   79
##       26   79
##       27   72
##       28   57
##       29   72
##       30   72
##       31   65
##       32   69
##       33   64
##       34   65
##       35   88
##       36   50
##       37   65
##       38   59
##       39   56
##       40   64
##       41   70
##       42   65
##       43   65
##       44   51
##       45   61
##       46   74
##       47   68
##       48   64
##       49   66
##       50   68
##       51   68
##       52   80
##       53   70
##       54   68
##       55   64
##       56   80
##       57   65
##       58   67
##       59   60
##       60   76
##       61   76
##       62   70
##       63   72
##       64   80
##       65   76
##       66   89
##       67   98
##       68  100
##       69   95
##       70  119
```

```
##       71  170
##       72  362
##
## $PhoneService
## NULL
##
## $MultipleLines
## NULL
##
## $InternetService
## NULL
##
## $OnlineSecurity
## NULL
##
## $OnlineBackup
## NULL
##
## $DeviceProtection
## NULL
##
## $TechSupport
## NULL
##
## $StreamingTV
## NULL
##
## $StreamingMovies
## NULL
##
## $Contract
## NULL
##
## $PaperlessBilling
## NULL
##
## $PaymentMethod
## NULL
##
## $MonthlyCharges
## NULL
##
## $TotalCharges
## NULL
##
## $Churn
```

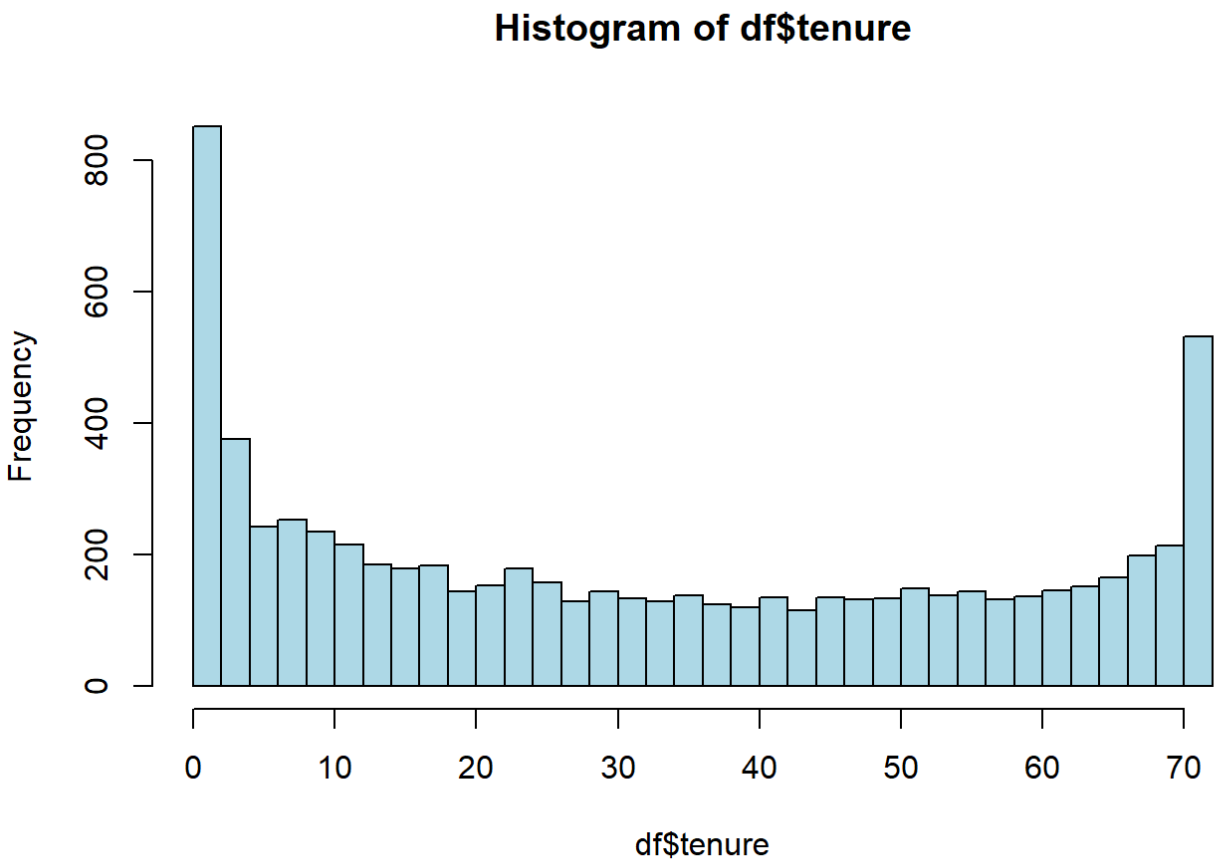
```
## NULL
```

- 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='lightblue')
```



Vemos una frecuencia desproporcionada de clientes con valores menores de 2 meses y mayores de 70 meses en este gráfico de la antigüedad. Eso es aproximadamente 6 años. Este es un efecto muy frecuente en las grandes compañías. Suele ser porque seguramente en esa fecha se implantaría o se cambiaría el sistema de gestión, y antes no se estaría recogiendo la fecha de alta o simplemente se perdió en la migración de sistema. Lo que pasa es que todos los clientes más antiguos aparecen dados de alta en esa fecha. Como vamos a discretizar esta variable, no supone problema este hecho.

- 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.

```
df <- df %>%
  mutate_at(a_factores,as.factor)
```

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 == 'Yes'),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.

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

```
##  [1] "gender"           "SeniorCitizen"    "Partner"          "Dependents"
##  [5] "tenure"           "PhoneService"     "MultipleLines"    "InternetService"
##  [9] "OnlineSecurity"   "OnlineBackup"     "DeviceProtection" "TechSupport"
## [13] "StreamingTV"      "StreamingMovies"  "Contract"         "PaperlessBilling"
## [17] "PaymentMethod"    "MonthlyCharges"   "TotalCharges"
```

4.3.2.1.1 - Preselección con RandomForest

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 236.381943          1
## TotalCharges         TotalCharges 235.241598          2
## MonthlyCharges     MonthlyCharges 183.791079          3
## Contract                 Contract 147.971206          4
## PaymentMethod       PaymentMethod 108.449539          5
## TechSupport           TechSupport  96.668796          6
## OnlineSecurity     OnlineSecurity  90.587588          7
## InternetService   InternetService  72.192209          8
## OnlineBackup         OnlineBackup  56.391452          9
## DeviceProtection DeviceProtection  47.083949         10
## PaperlessBilling PaperlessBilling  36.428073         11
## StreamingMovies   StreamingMovies  36.240288         12
## MultipleLines       MultipleLines  35.602551         13
## StreamingTV           StreamingTV  30.431791         14
## SeniorCitizen       SeniorCitizen  27.369302         15
## gender                     gender  27.187502         16
## Partner                   Partner  26.906220         17
## Dependents              Dependents  26.651402         18
## PhoneService         PhoneService   9.510707         19
```

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%
  |
  |--                                                    |   5%
  |
  |-----                                                 |  10%
  |
  |--------                                              |  15%
  |
  |----------                                            |  20%
  |
  |------------                                          |  25%
  |
  |---------------                                       |  30%
  |
  |------------------                                    |  35%
  |
  |--------------------                                  |  40%
  |
  |----------------------                                |  45%
  |
  |--------------------------                            |  50%
  |
  |----------------------------                          |  55%
  |
  |------------------------------                        |  60%
  |
  |--------------------------------                      |  65%
  |
  |----------------------------------                    |  70%
  |
  |--------------------------------------                |  75%
  |
  |----------------------------------------              |  80%
  |
  |------------------------------------------            |  85%
  |
  |--------------------------------------------          |  90%
  |
  |----------------------------------------------        |  95%
  |
  |----------------------------------------------------| 100%
##
```

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

```
##            VARIABLE IMP_IV RANKING_IV
## 15         Contract 1.2332          1
## 5            tenure 0.8773          2
## 9    OnlineSecurity 0.7153          3
## 12      TechSupport 0.6971          4
## 8   InternetService 0.6152          5
## 10     OnlineBackup 0.5265          6
## 11 DeviceProtection 0.4976          7
## 18   MonthlyCharges 0.4824          8
## 17    PaymentMethod 0.4557          9
## 14  StreamingMovies 0.3799         10
## 13      StreamingTV 0.3787         11
## 19     TotalCharges 0.3202         12
## 16 PaperlessBilling 0.2020         13
## 4        Dependents 0.1532         14
## 3           Partner 0.1179         15
## 2     SeniorCitizen 0.1051         16
## 7     MultipleLines 0.0081         17
## 6      PhoneService 0.0007         18
## 1            gender 0.0004         19
```

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 RANKING_TOT
## 1            tenure 236.381943 0.8773          1          2           3
## 2          Contract 147.971206 1.2332          4          1           5
## 3       TechSupport  96.668796 0.6971          6          4          10
## 4    OnlineSecurity  90.587588 0.7153          7          3          10
## 5    MonthlyCharges 183.791079 0.4824          3          8          11
## 6   InternetService  72.192209 0.6152          8          5          13
## 7      TotalCharges 235.241598 0.3202          2         12          14
## 8     PaymentMethod 108.449539 0.4557          5          9          14
## 9      OnlineBackup  56.391452 0.5265          9          6          15
## 10 DeviceProtection  47.083949 0.4976         10          7          17
## 11  StreamingMovies  36.240288 0.3799         12         10          22
## 12 PaperlessBilling  36.428073 0.2020         11         13          24
## 13      StreamingTV  30.431791 0.3787         14         11          25
## 14    MultipleLines  35.602551 0.0081         13         17          30
## 15    SeniorCitizen  27.369302 0.1051         15         16          31
## 16          Partner  26.906220 0.1179         17         15          32
## 17       Dependents  26.651402 0.1532         18         14          32
## 18           gender  27.187502 0.0004         16         19          35
## 19     PhoneService   9.510707 0.0007         19         18          37
```

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] 0.5951327
```

Los métodos nos ofrecen una fiabilidad en los resultados de 0.64

A la vista de los resultados, vamos a descartar aquellas variables que no hayan salido entre las 13 mas importantes en el ranking total.

Incluimos las 13 primeras en la siguiente lista

```
ind_corta <- c('tenure','Contract','TechSupport','MonthlyCharges','OnlineSecurity','Inte
rnetService','TotalCharges','PaymentMethod','OnlineBackup','DeviceProtection','Streaming
Movies','StreamingTV','PaperlessBilling')
```

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

```
ind_corta
```

```
##  [1] "tenure"           "Contract"         "TechSupport"      "MonthlyCharges"
##  [5] "OnlineSecurity"   "InternetService"  "TotalCharges"     "PaymentMethod"
##  [9] "OnlineBackup"     "DeviceProtection" "StreamingMovies"  "StreamingTV"
## [13] "PaperlessBilling"
```

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 y la TARGET1

```
lista <- ind_corta
lista
```

```
##  [1] "tenure"           "Contract"         "TechSupport"      "MonthlyCharges"
##  [5] "OnlineSecurity"   "InternetService"  "TotalCharges"     "PaymentMethod"
##  [9] "OnlineBackup"     "DeviceProtection" "StreamingMovies"  "StreamingTV"
## [13] "PaperlessBilling"
```

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 varaible 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"           "InternetService"  "OnlineSecurity"   "OnlineBackup"
##  [5] "DeviceProtection" "TechSupport"      "StreamingTV"      "StreamingMovies"
##  [9] "Contract"         "PaperlessBilling" "PaymentMethod"    "MonthlyCharges"
## [13] "TotalCharges"     "customerID"       "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] 7032   21
```

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

```
## [1] 7032   15
```

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 'tenure', 'MonthlyCharges'y 'TotalCharges' 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)
}
```

```
#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)

#MonthlyCharges:
disc_temp_MonthlyCharges <- discretizar(df$MonthlyCharges,df$TARGET1)
df_temp <- select(df,MonthlyCharges,TARGET1)
df_temp <- smbinning.gen(df_temp,disc_temp_MonthlyCharges,chrname = 'MONTHLYCHARGES_DISC
')
df <- cbind(df,df_temp[,3]) %>% select(-MonthlyCharges)

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

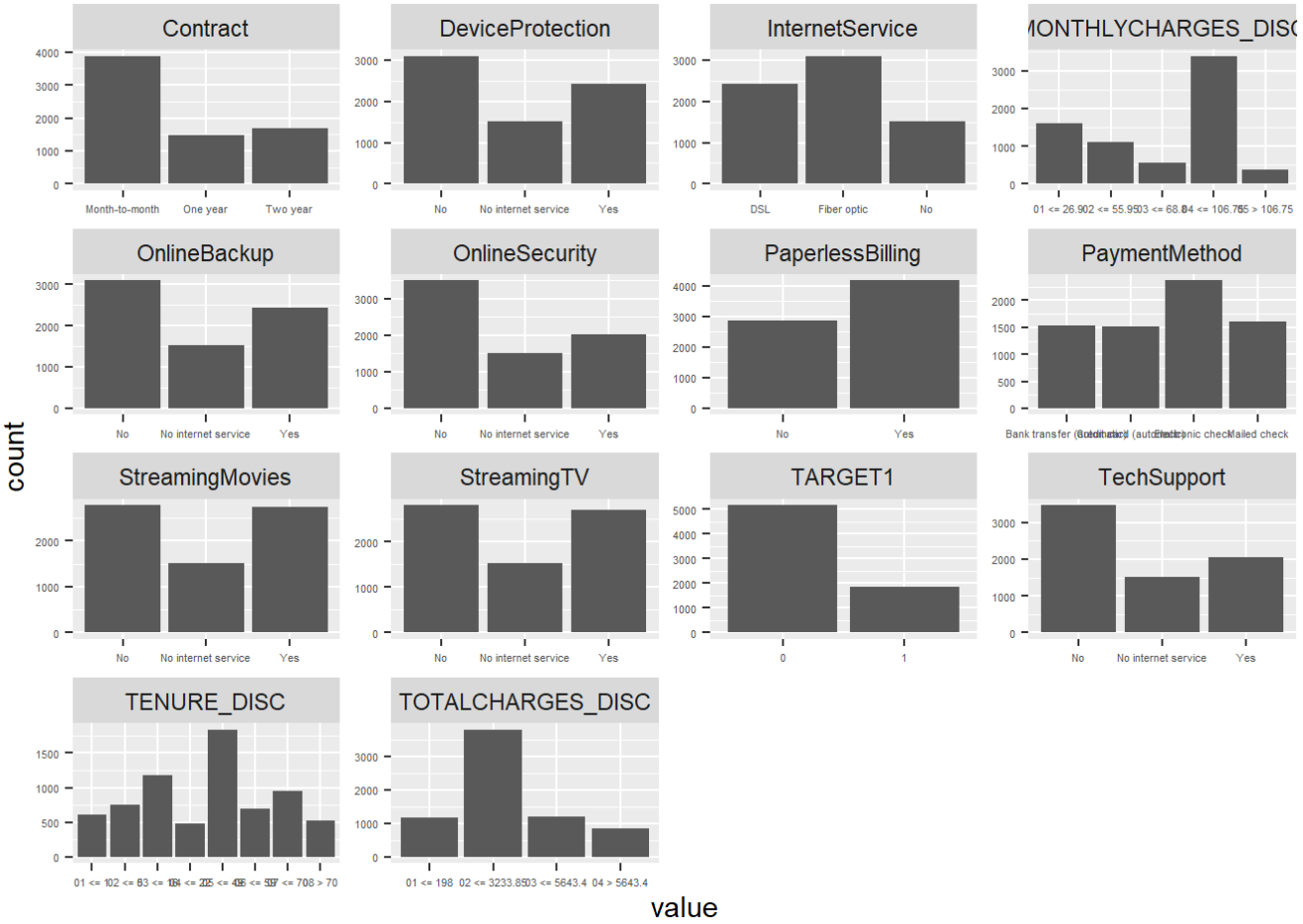
Obtebemos un resumen para ver las variables discretizadas que hemos obtenido.

```
glimpse(df)
```

```
## Rows: 7,032
## Columns: 15
## $ InternetService     <fct> DSL, DSL, DSL, DSL, Fiber optic, Fiber optic, Fibe~
## $ OnlineSecurity      <fct> No, Yes, Yes, Yes, No, No, No, Yes, No, Yes, Yes, ~
## $ OnlineBackup        <fct> Yes, No, Yes, No, No, No, Yes, No, No, Yes, No, No~
## $ DeviceProtection    <fct> No, Yes, No, Yes, No, Yes, No, No, Yes, No, No, No~
## $ TechSupport         <fct> No, No, No, Yes, No, No, No, No, Yes, No, No, No i~
## $ StreamingTV         <fct> No, No, No, No, No, Yes, Yes, No, Yes, No, No, No ~
## $ StreamingMovies     <fct> No, No, No, No, No, Yes, No, No, Yes, No, No, No i~
## $ Contract            <fct> Month-to-month, One year, Month-to-month, One year~
## $ PaperlessBilling    <fct> Yes, No, Yes, No, Yes, Yes, Yes, No, Yes, No, Yes,~
## $ PaymentMethod       <fct> Electronic check, Mailed check, Mailed check, Bank~
## $ customerID          <chr> "7590-VHVEG", "5575-GNVDE", "3668-QPYBK", "7795-CF~
## $ TARGET1             <fct> 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,~
## $ TENURE_DISC         <fct> 01 <= 1, 05 <= 49, 02 <= 5, 05 <= 49, 02 <= 5, 03 ~
## $ MONTHLYCHARGES_DISC <fct> 02 <= 55.95, 03 <= 68.8, 02 <= 55.95, 02 <= 55.95,~
## $ TOTALCHARGES_DISC   <fct> 01 <= 198, 02 <= 3233.85, 01 <= 198, 02 <= 3233.85~
```

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))
```



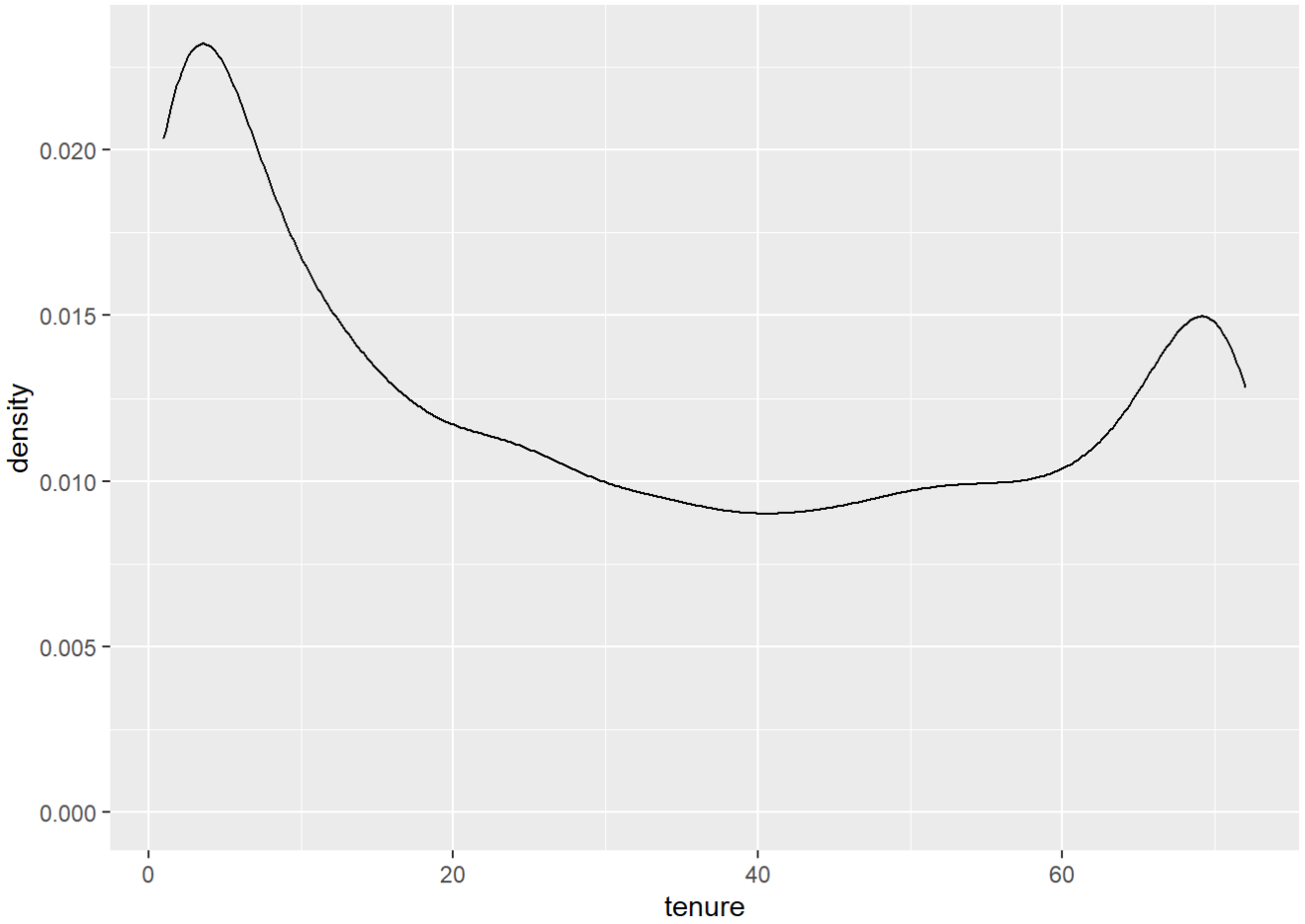
Comprobamos en los gráficos que las tres variables discretizadas, MonthlyCharges, TotalCharges y Tenure, no salen monotónicas, así que vamos a discretizarlas a mano.

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

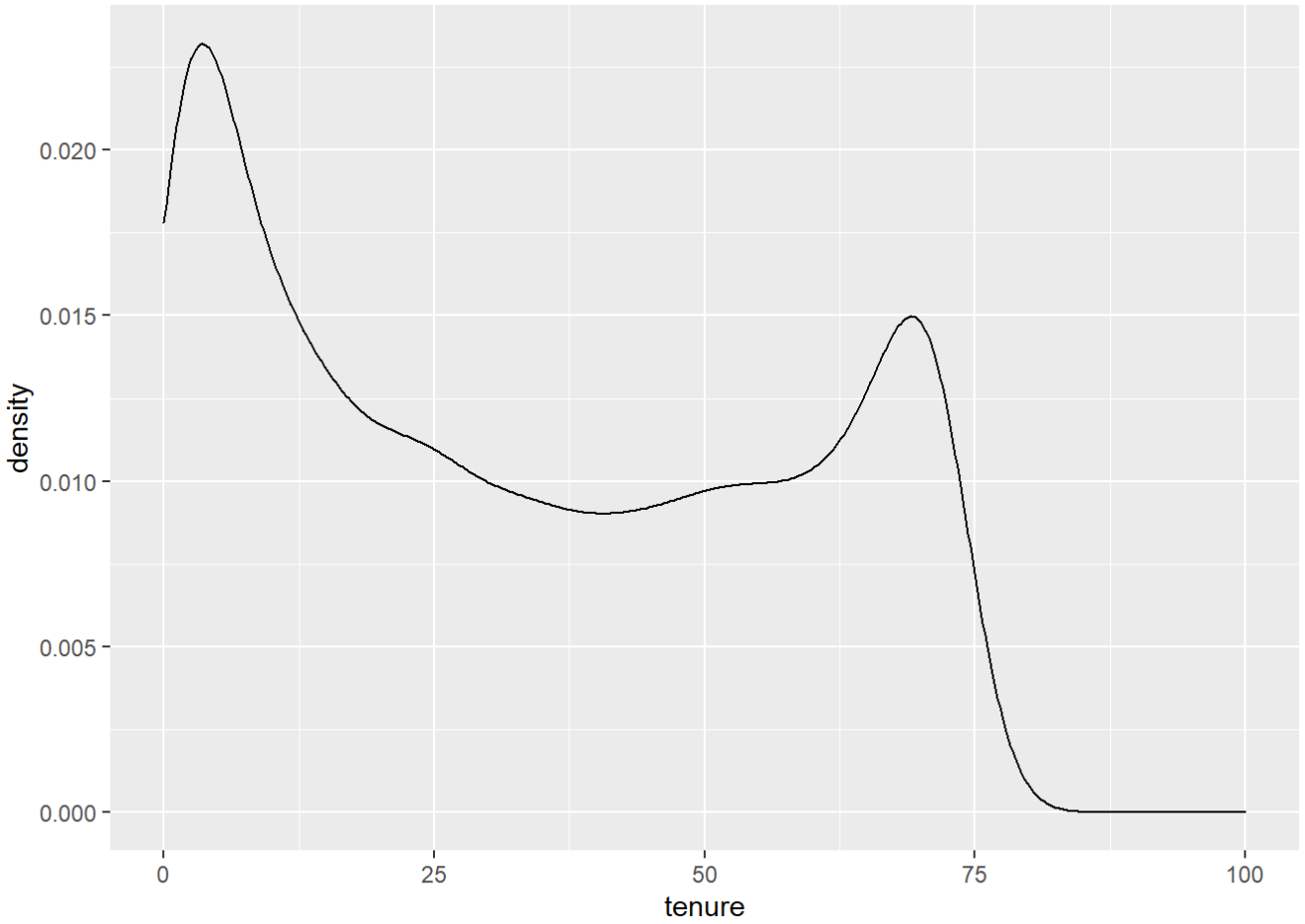
Procedemos a discretizarla manualmente.

Empezamos con 'tenure'. Lo primero es ver que distribucion tiene la variable target con la variable a discretizar.

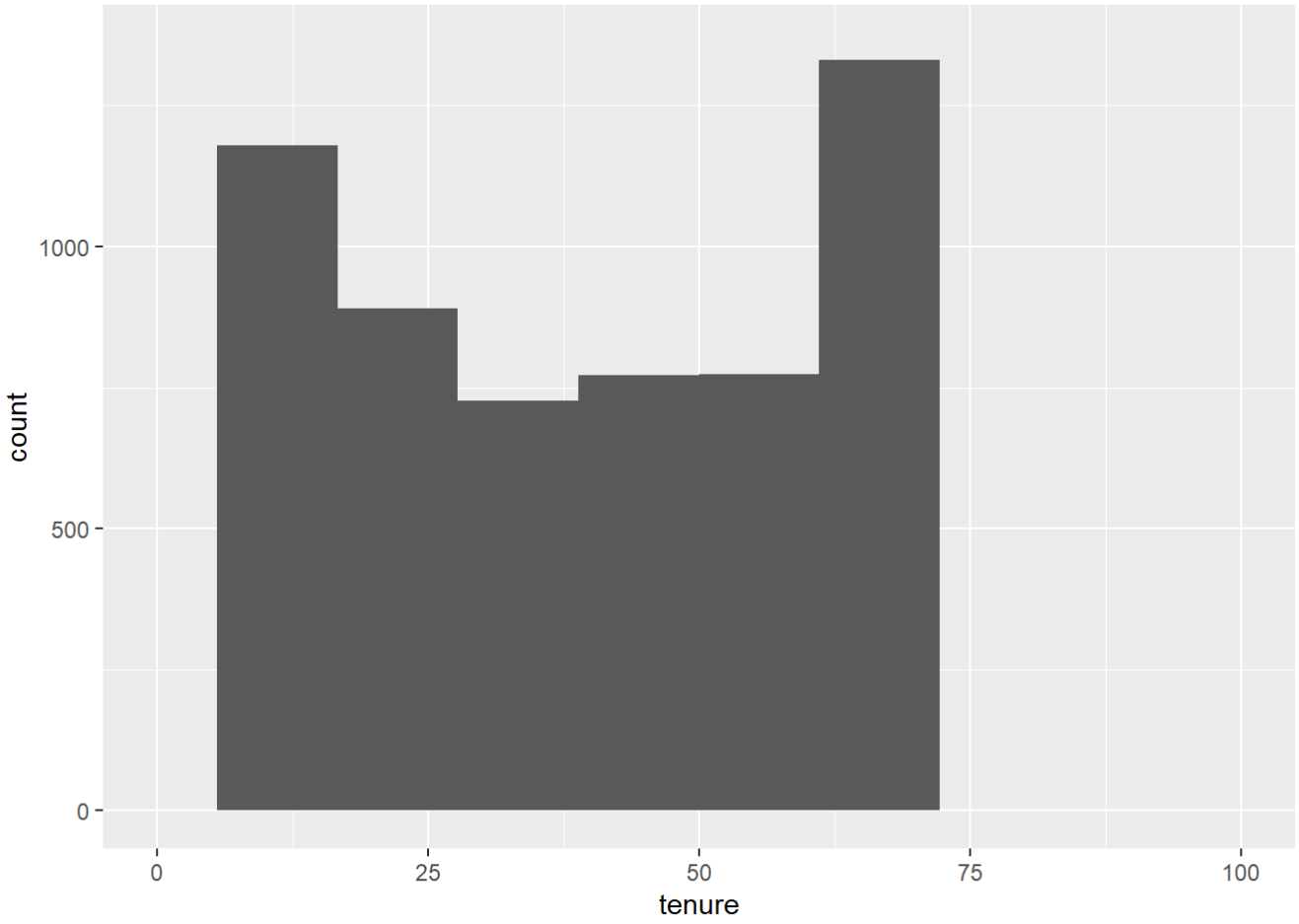
```
ggplot(df,aes(tenure)) + geom_density()
```



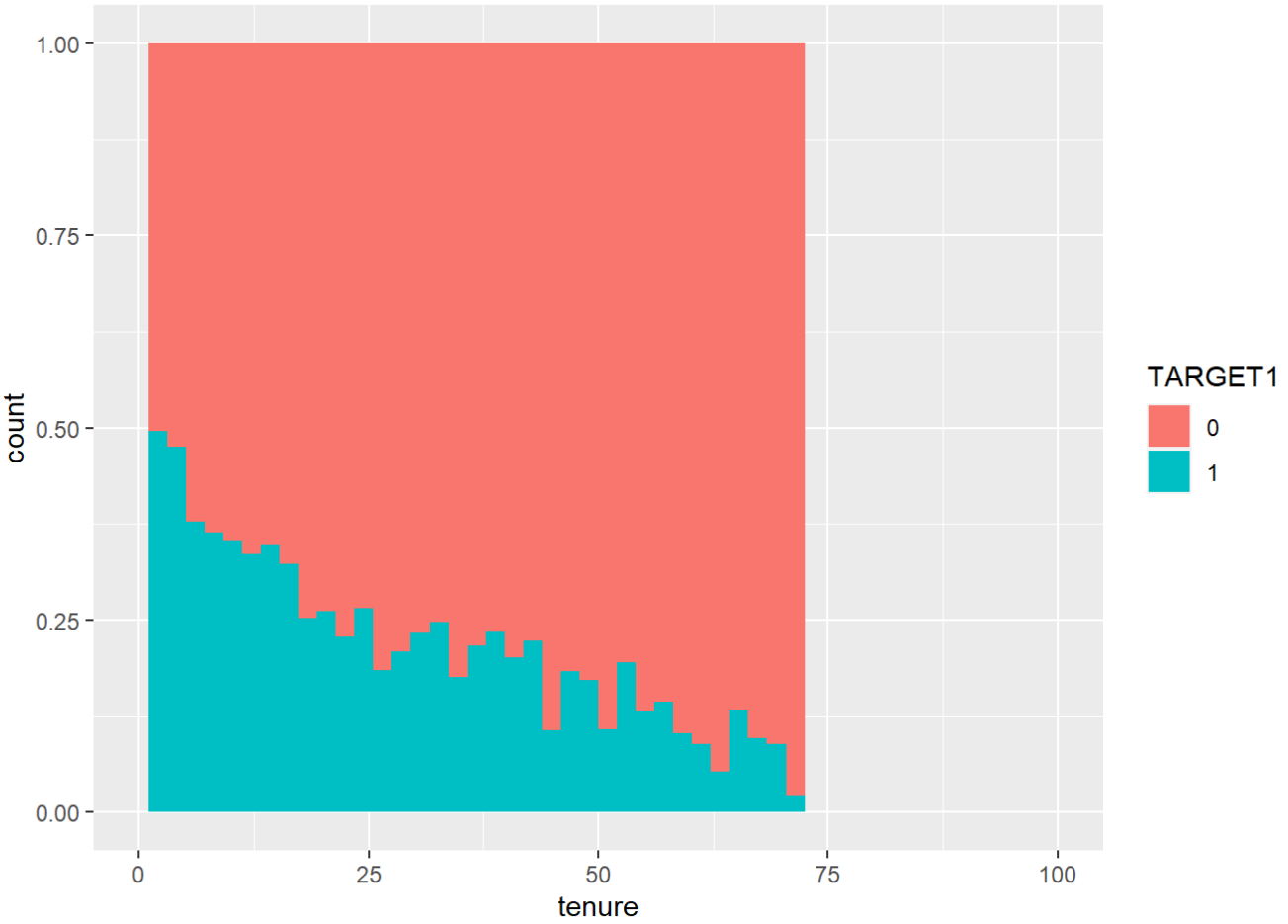

```
# Vamos a limitar el eje x para obtener la gráfica completa.
ggplot(df,aes(tenure)) + geom_density() + scale_x_continuous(limits = c(0, 100))
```




```
# 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, 10
0))
```




```
# Ya sabemos que queremos una forma decreciente ahora veamos como se comporta la variabl
e target para ver si podremos generar un perfil monotónico
ggplot(df,aes(tenure,fill=TARGET1)) + geom_histogram(bins = 50,position='fill') + scale_
x_continuous(limits = c(0, 100))
```



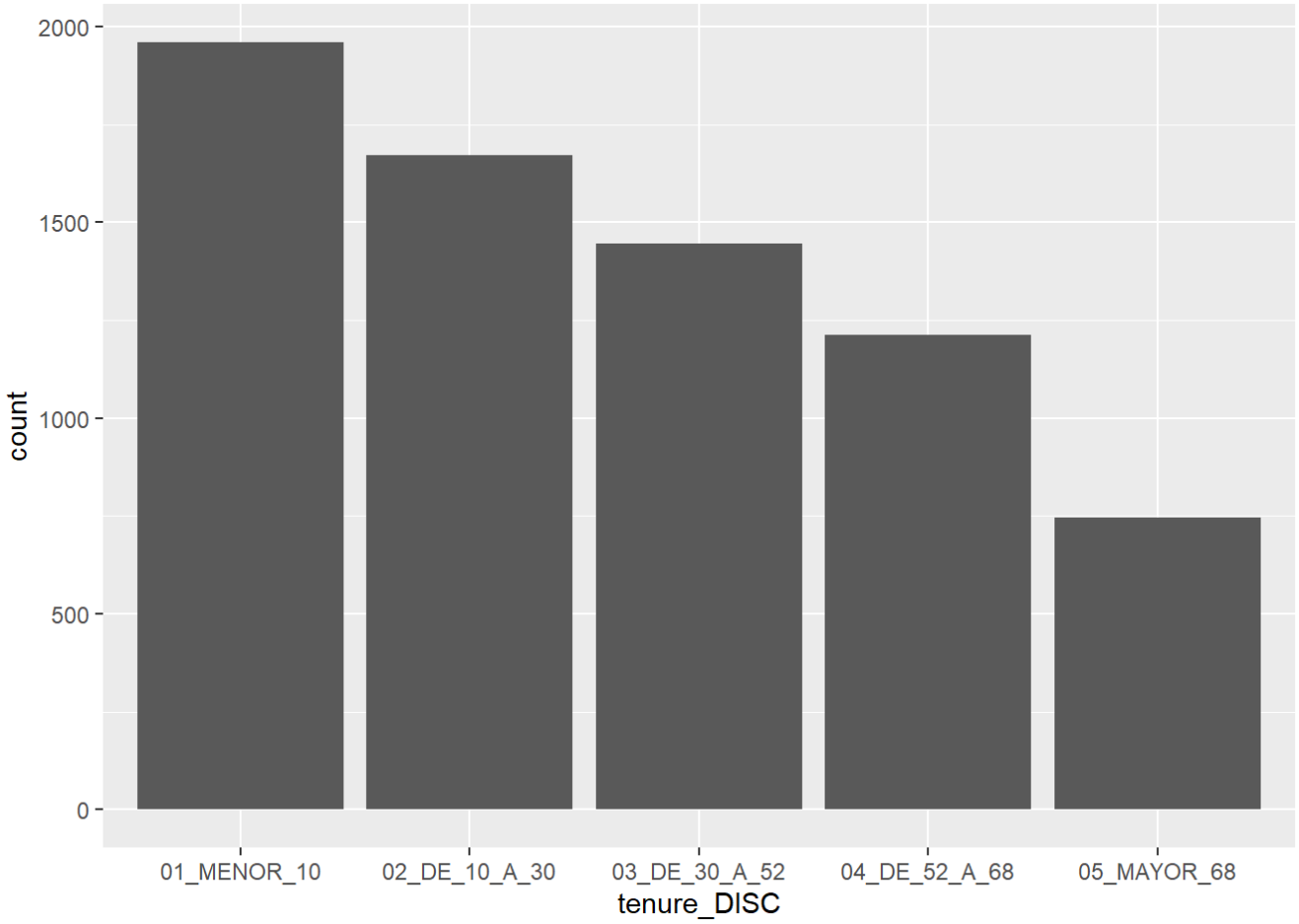

```
# Sabiendo ambas cosas vamos a apoyarnos en los deciles para intuir donde podemos hacer buenos cortes
as.data.frame(quantile(df$tenure,prob = seq(0, 1, length = 11)))
```

```
##      quantile(df$tenure, prob = seq(0, 1, length = 11))
## 0%                                                  1.0
## 10%                                                 2.0
## 20%                                                 6.0
## 30%                                                12.0
## 40%                                                20.0
## 50%                                                29.0
## 60%                                                40.0
## 70%                                                50.0
## 80%                                                60.8
## 90%                                                69.0
## 100%                                               72.0
```

```
df <- df %>% mutate(tenure_DISC = as.factor(case_when(
      tenure <= 10 ~ '01_MENOR_10',
      tenure > 10 & tenure <= 30 ~ '02_DE_10_A_30',
      tenure > 30 & tenure <= 52 ~ '03_DE_30_A_52',
      tenure > 52 & tenure <= 68 ~ '04_DE_52_A_68',
      tenure > 68 ~ '05_MAYOR_68',
      TRUE ~ '00_ERROR'))
)
```

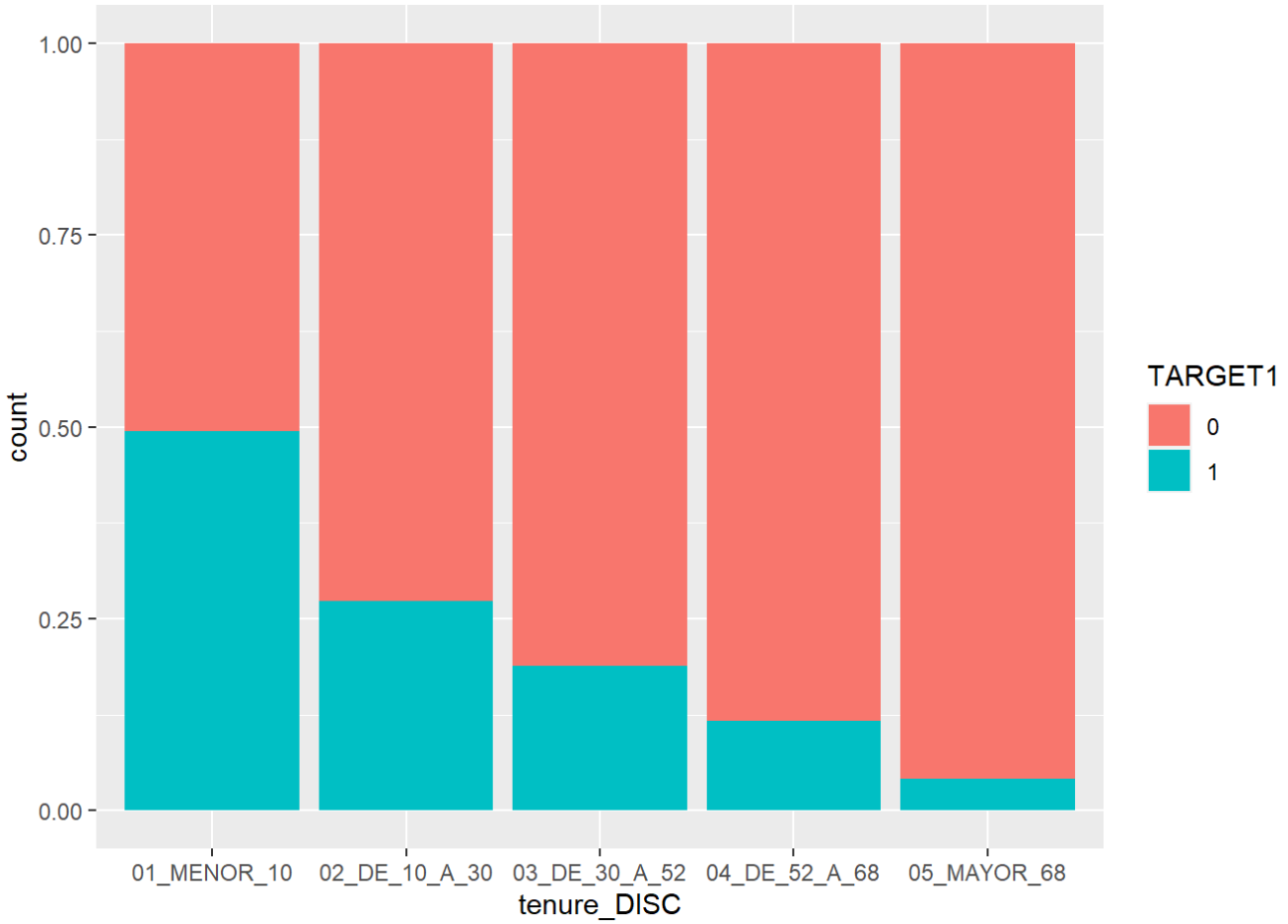
Veamos si la distribución ha quedado similar a la original.

```
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')
```

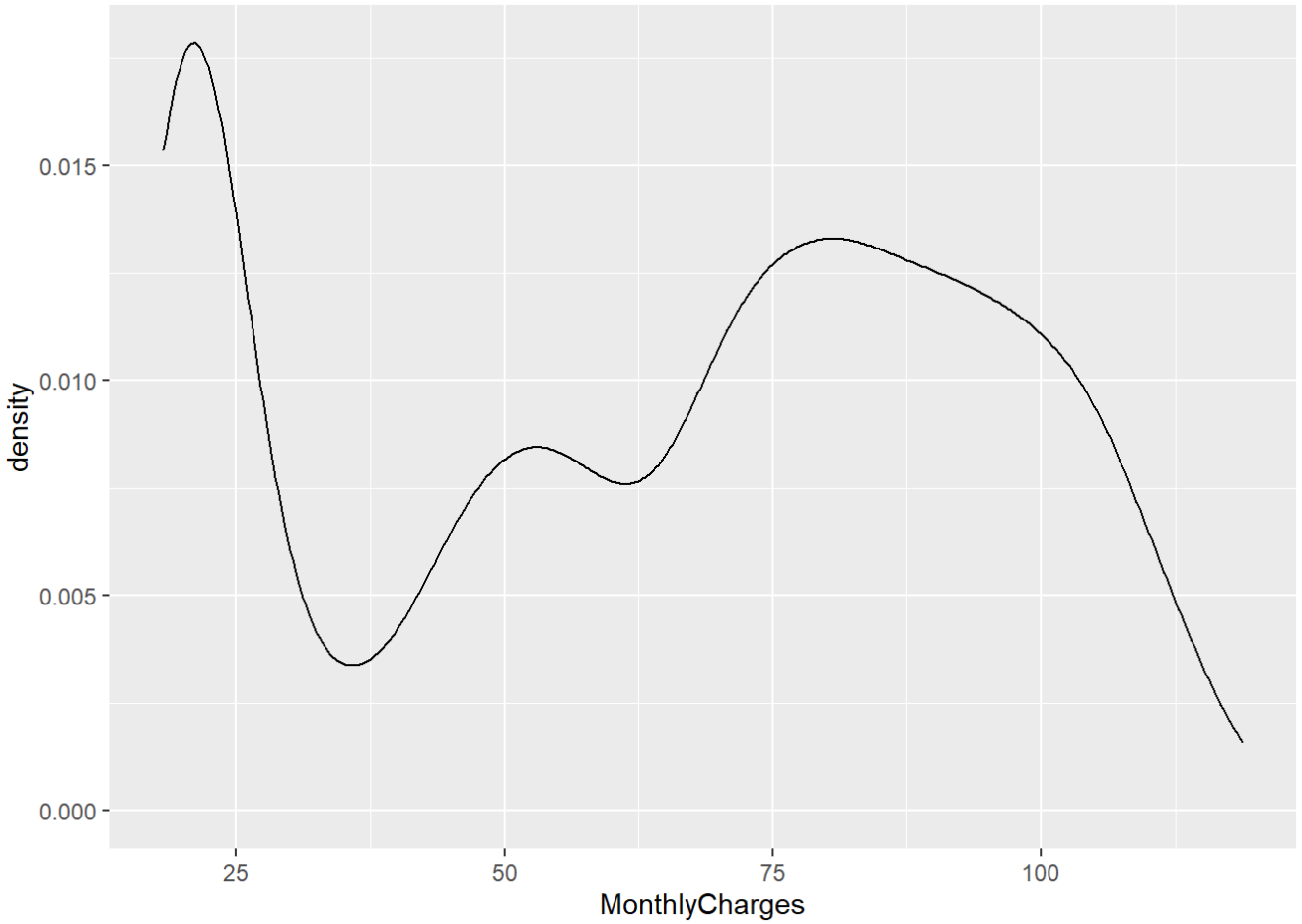


Vemos como a menor permanencia la posibilidad de abandon es menor, pero cuando la permanencia es mayor, vemos como la probabilidad de abandono es mucho mayor.

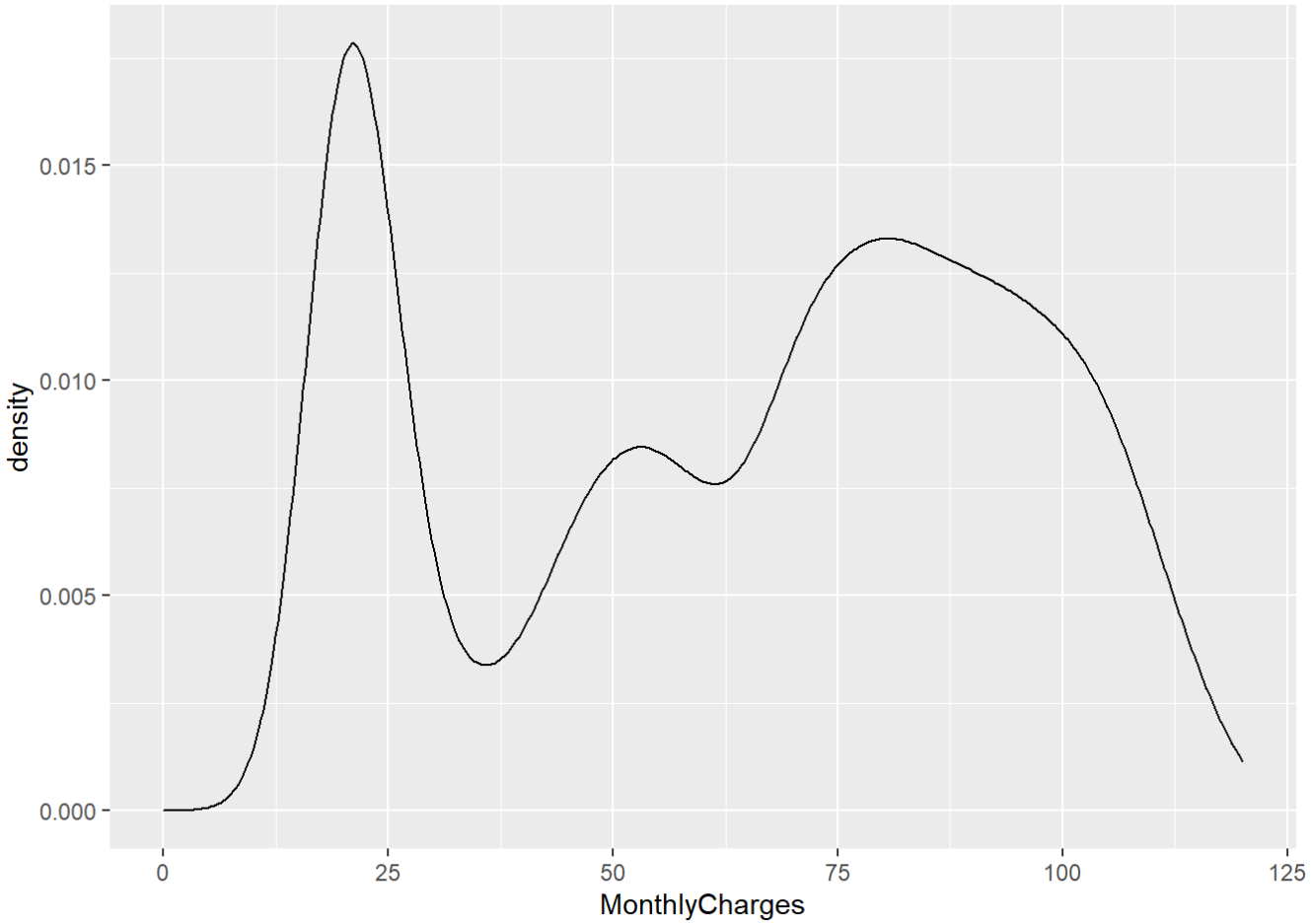
Continuamos con 'MonthlyCharges'.

Lo primero es ver que distribucion tiene la variable.

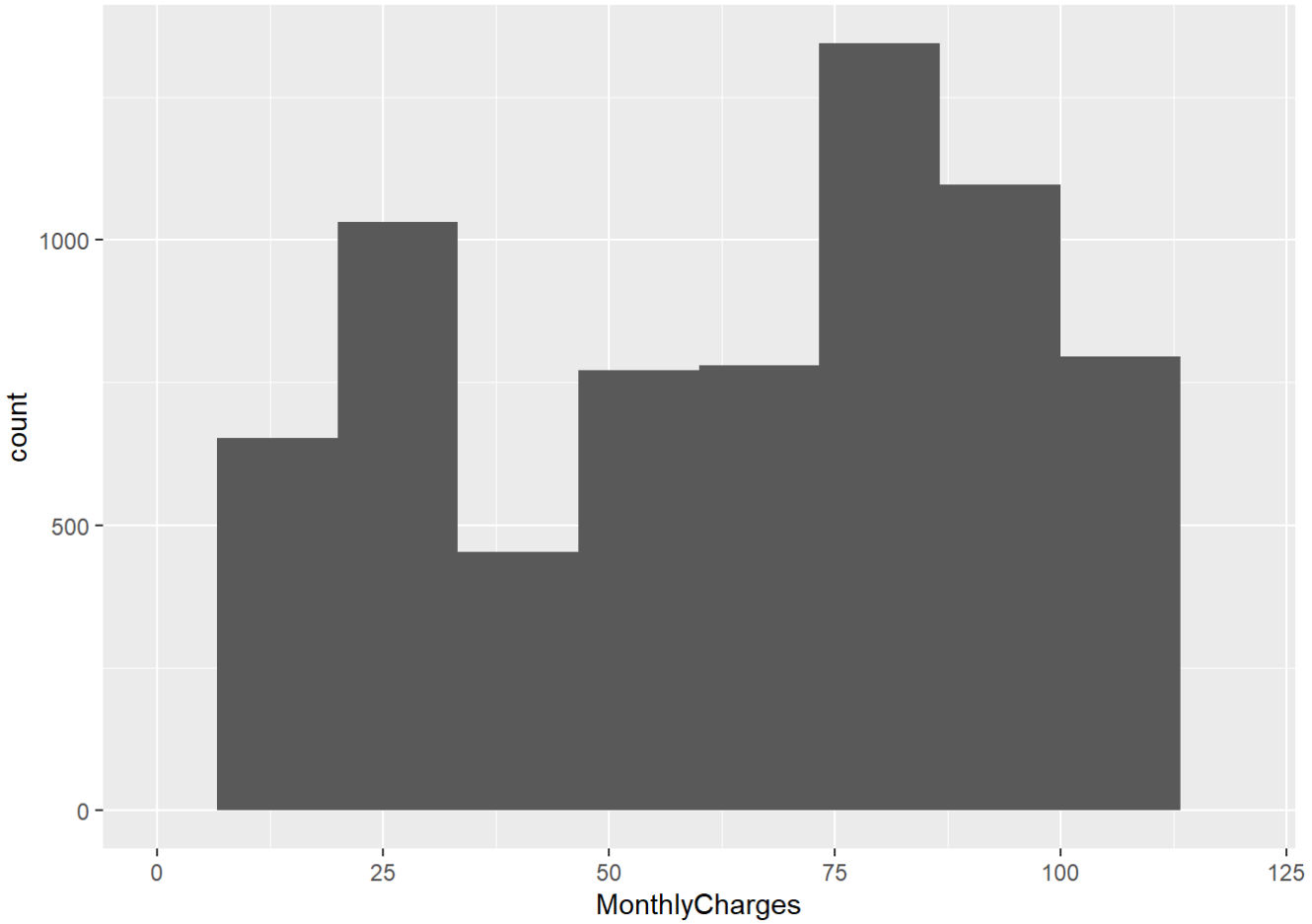
```
ggplot(df,aes(MonthlyCharges)) + geom_density()
```



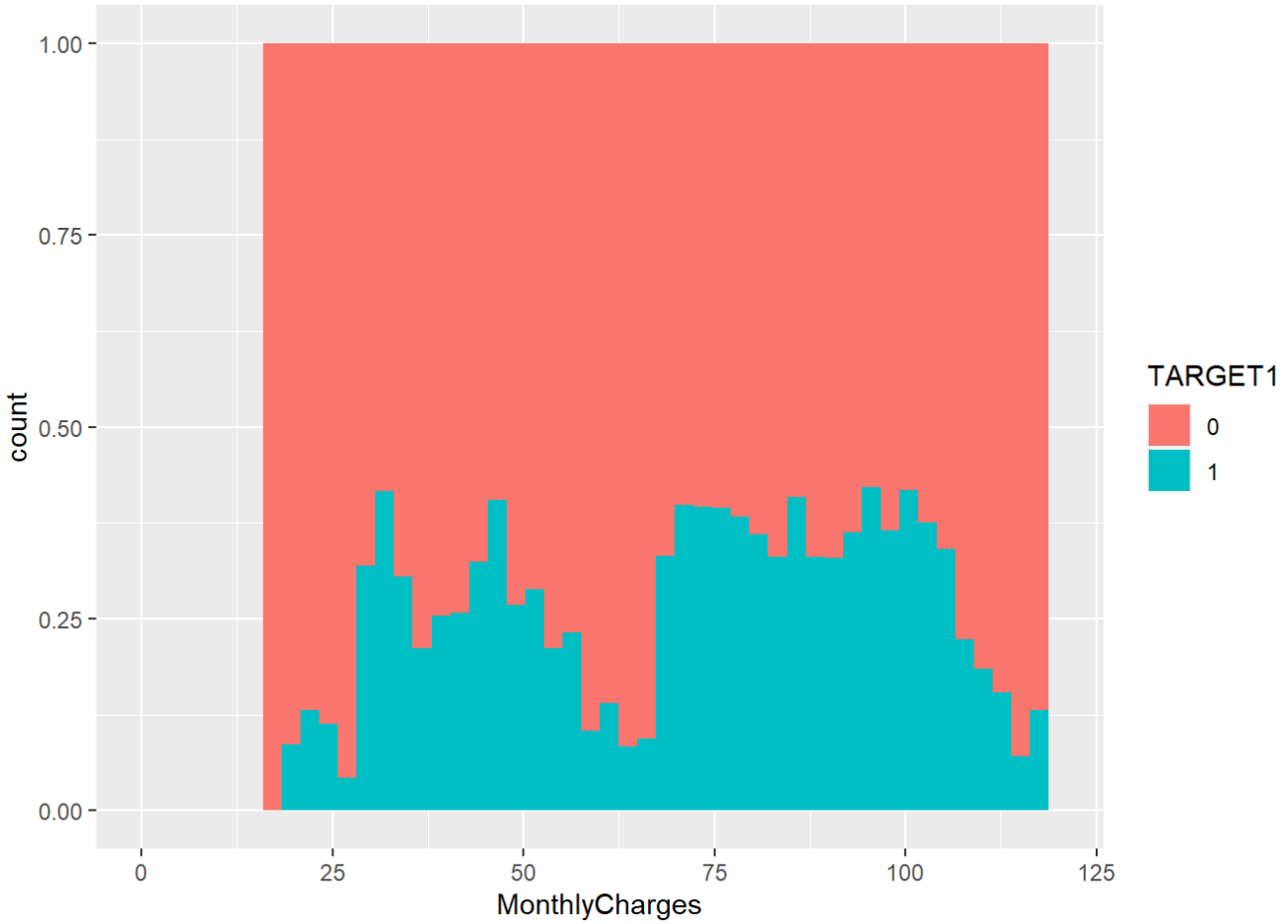

```
# Vamos a limitar el eje x
ggplot(df,aes(MonthlyCharges)) + geom_density() + scale_x_continuous(limits = c(0, 120))
```




```
# Pedimos un histograma para aproximar mejor la forma que queremos conseguir
ggplot(df,aes(MonthlyCharges)) + geom_histogram(bins = 10) + scale_x_continuous(limits =
c(0, 120))
```




```
# Ya sabemos que queremos una forma creciente, ahora veamos como se comporta la variable target para ver si podremos generar un perfil monotónico.
ggplot(df,aes(MonthlyCharges,fill=TARGET1)) + geom_histogram(bins = 50,position='fill') + scale_x_continuous(limits = c(0, 120))
```




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

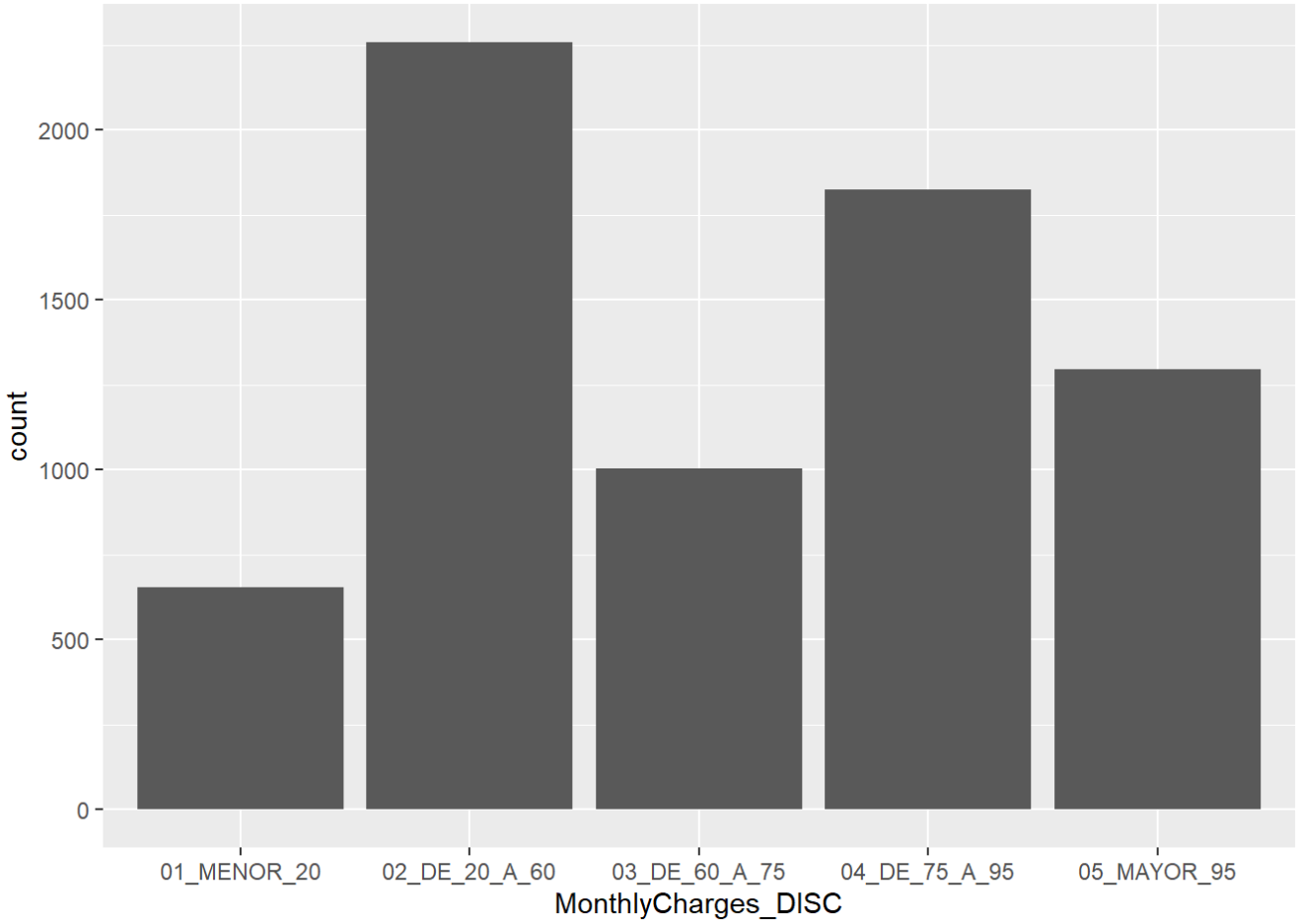
```
##      quantile(df$MonthlyCharges, prob = seq(0, 1, length = 11))
## 0%                                                       18.250
## 10%                                                      20.050
## 20%                                                      25.050
## 30%                                                      45.900
## 40%                                                      58.920
## 50%                                                      70.350
## 60%                                                      79.150
## 70%                                                      85.535
## 80%                                                      94.300
## 90%                                                     102.645
## 100%                                                    118.750
```

Procedemos a discretizarla manualmente

```
df <- df %>% mutate(MonthlyCharges_DISC = as.factor(case_when(
      MonthlyCharges <= 20 ~ '01_MENOR_20',
      MonthlyCharges > 20 & MonthlyCharges <= 60 ~ '02_DE_20_A_60',
      MonthlyCharges > 60 & MonthlyCharges <= 75 ~ '03_DE_60_A_75',
      MonthlyCharges > 75 & MonthlyCharges <= 95 ~ '04_DE_75_A_95',
      MonthlyCharges > 95 ~ '05_MAYOR_95',
      TRUE ~ '00_ERROR'))
)
```

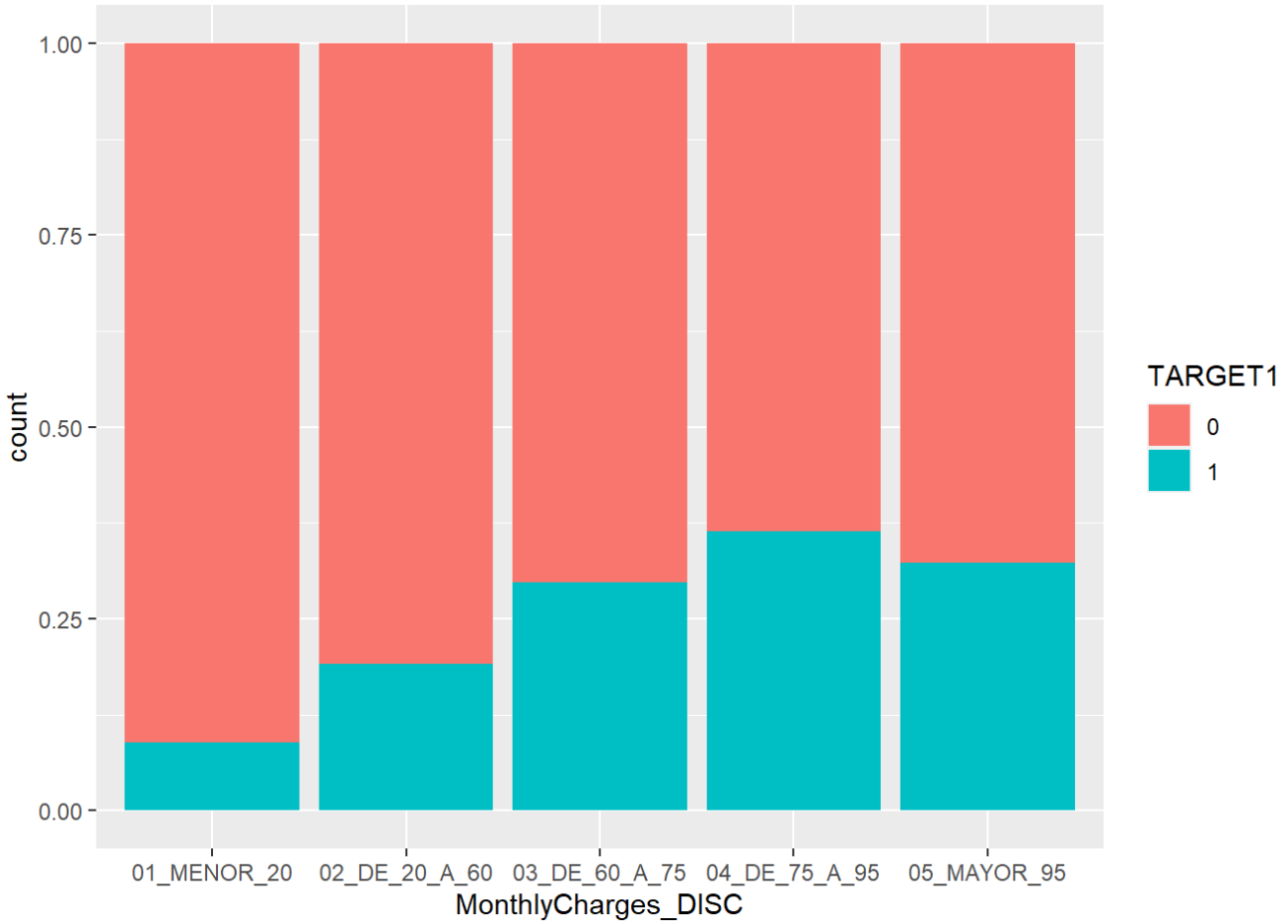
Vemos que la distribución ha quedado similar a la original.

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



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

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

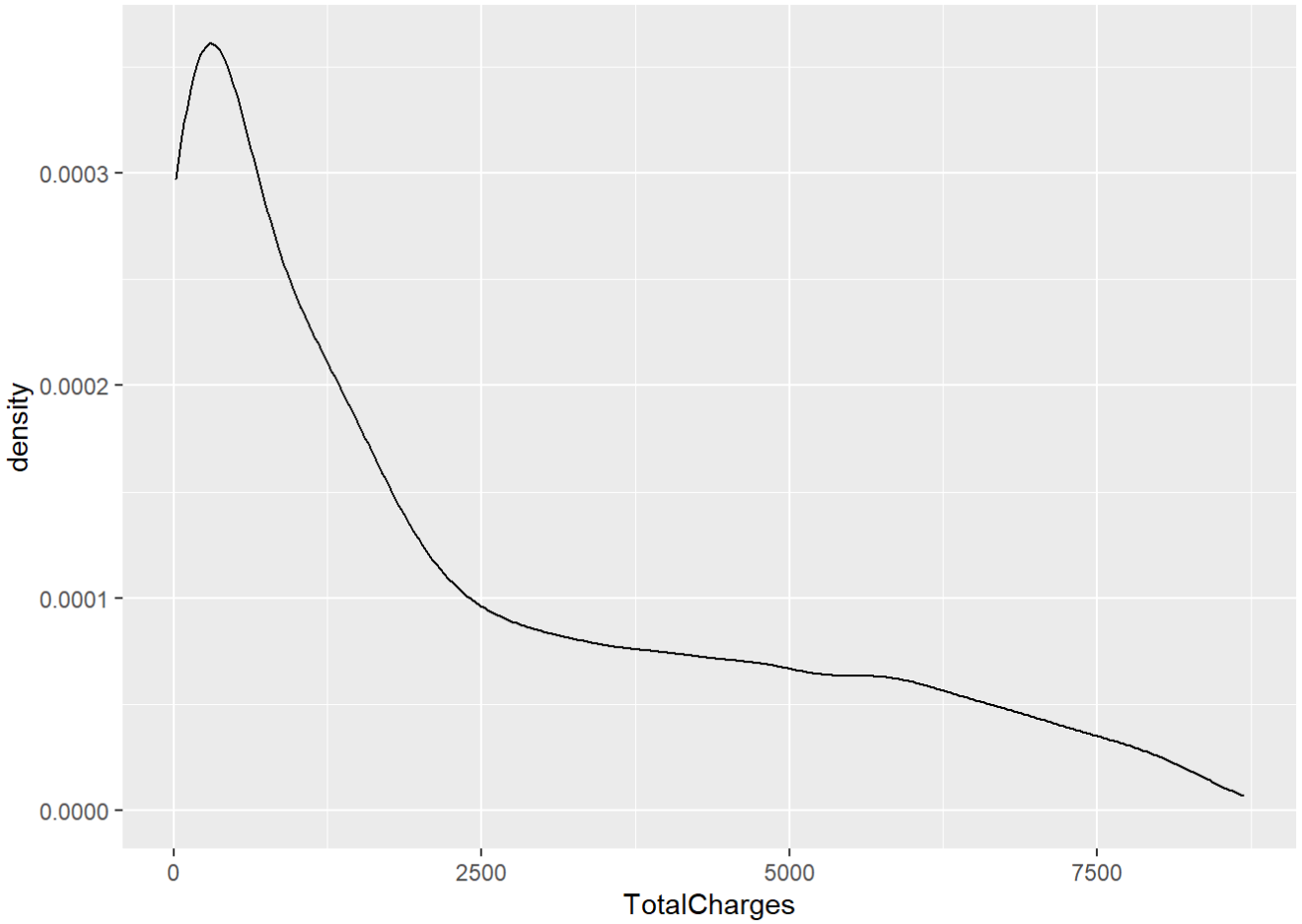


Vemos como los clientes con facturación mensual mayor de 60 € tienen menos probabilidad de abandono.

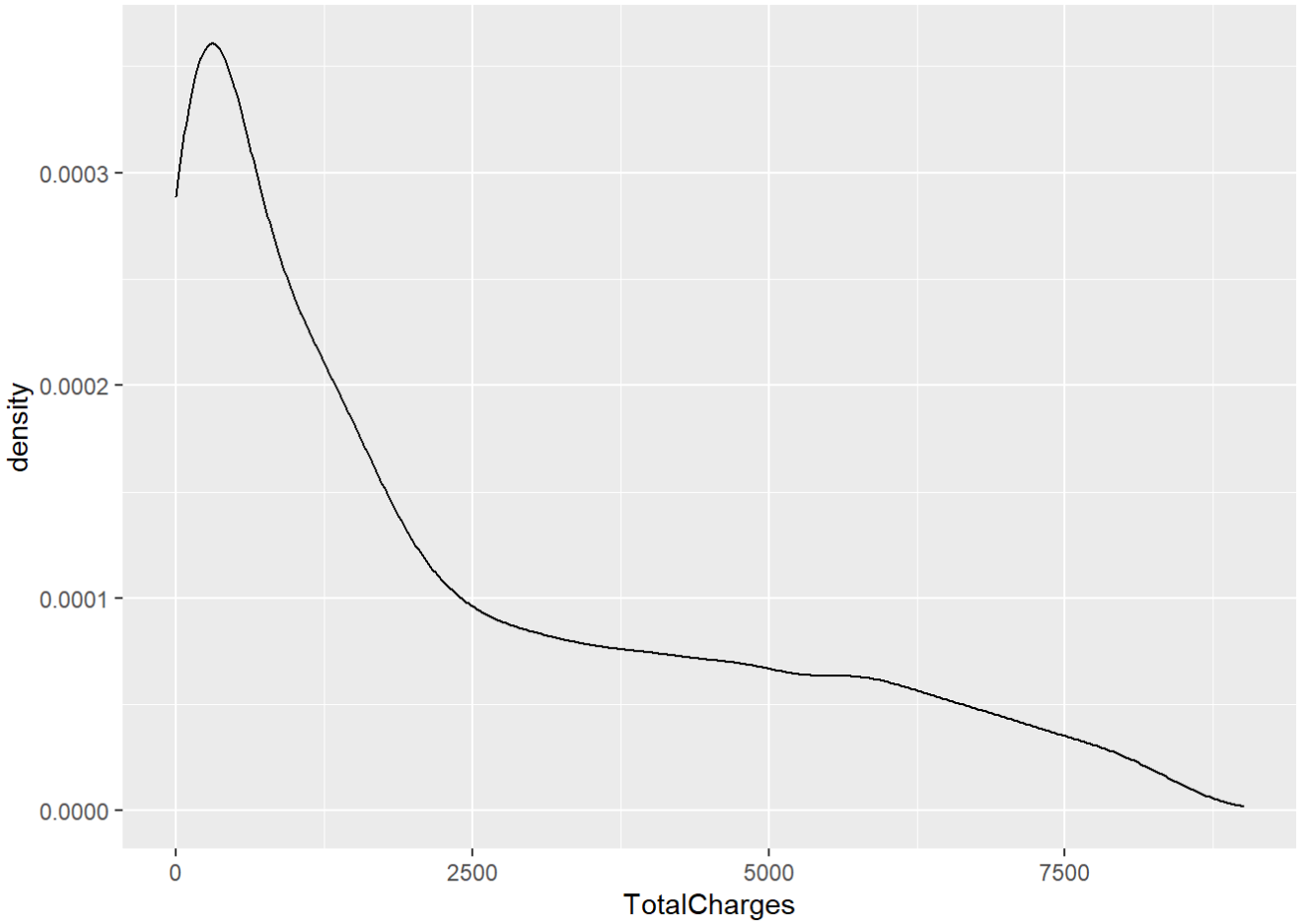
Finalizamos con 'TotalCharges'

Lo primero es ver que distribucion tiene la variable

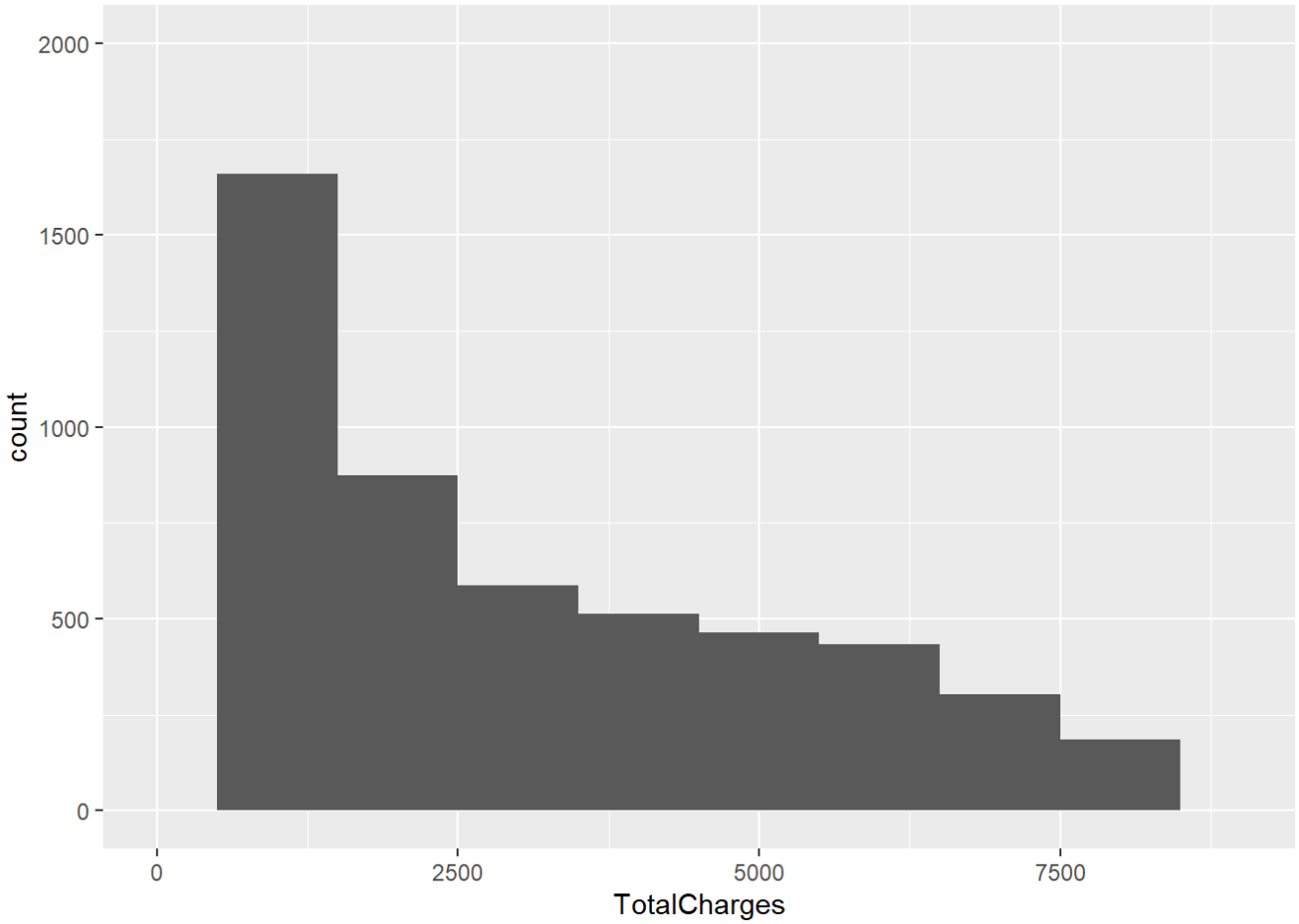
```
ggplot(df,aes(TotalCharges)) + geom_density()
```



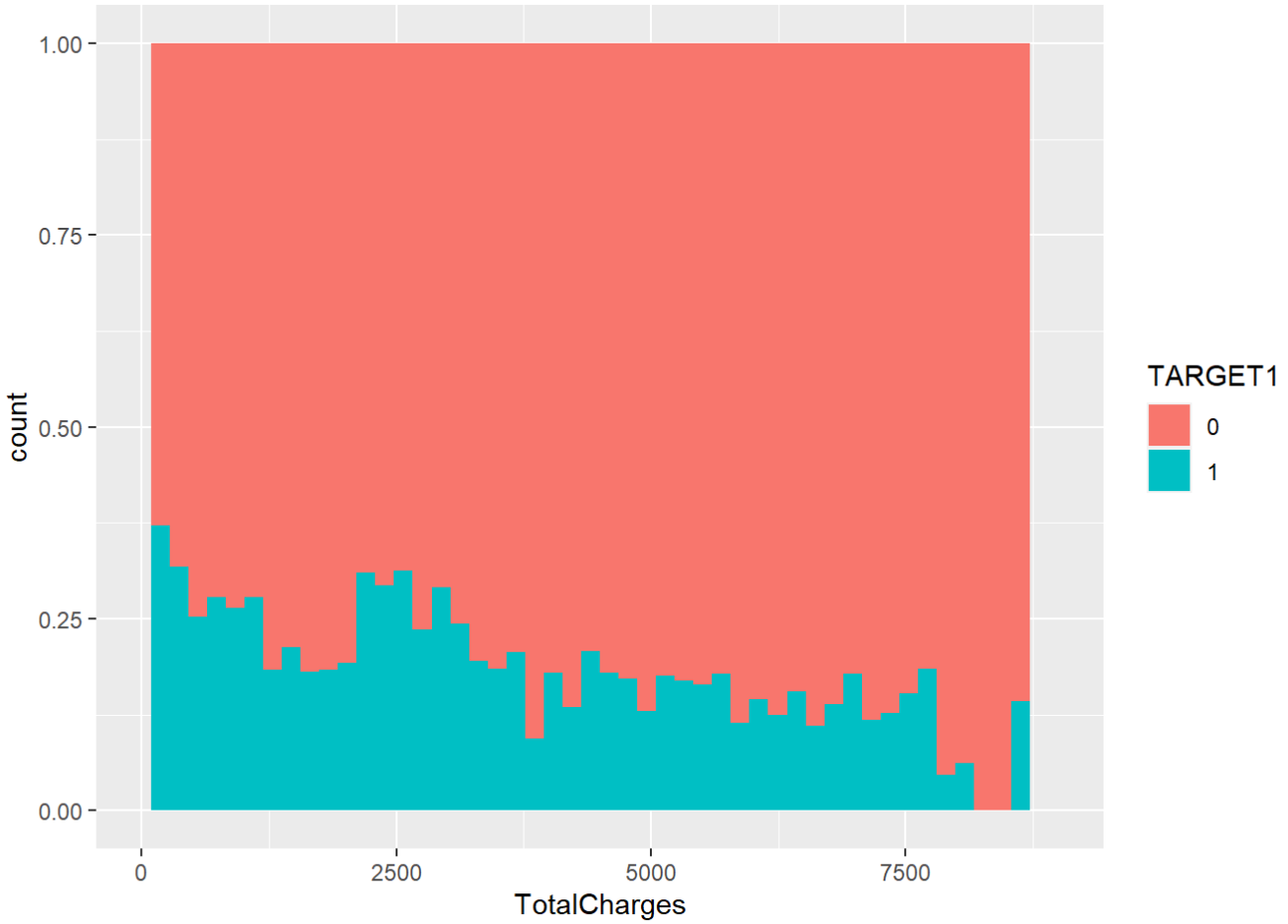

```
# Vamos a limitar el eje x
ggplot(df,aes(TotalCharges)) + geom_density() + scale_x_continuous(limits = c(0, 9000))
```




```
#Pedimos un histograma para aproximar mejor la forma que queremos conseguir
ggplot(df,aes(TotalCharges)) + geom_histogram(bins = 10) + scale_x_continuous(limits = c
(0, 9000))
```




```
#Ya sabemos que queremos una forma decreciente ahora veamos como se comporta la variable target para ver si podremos generar un perfil monotónico.
ggplot(df,aes(TotalCharges,fill=TARGET1)) + geom_histogram(bins = 50,position='fill') + scale_x_continuous(limits = c(0, 9000))
```




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

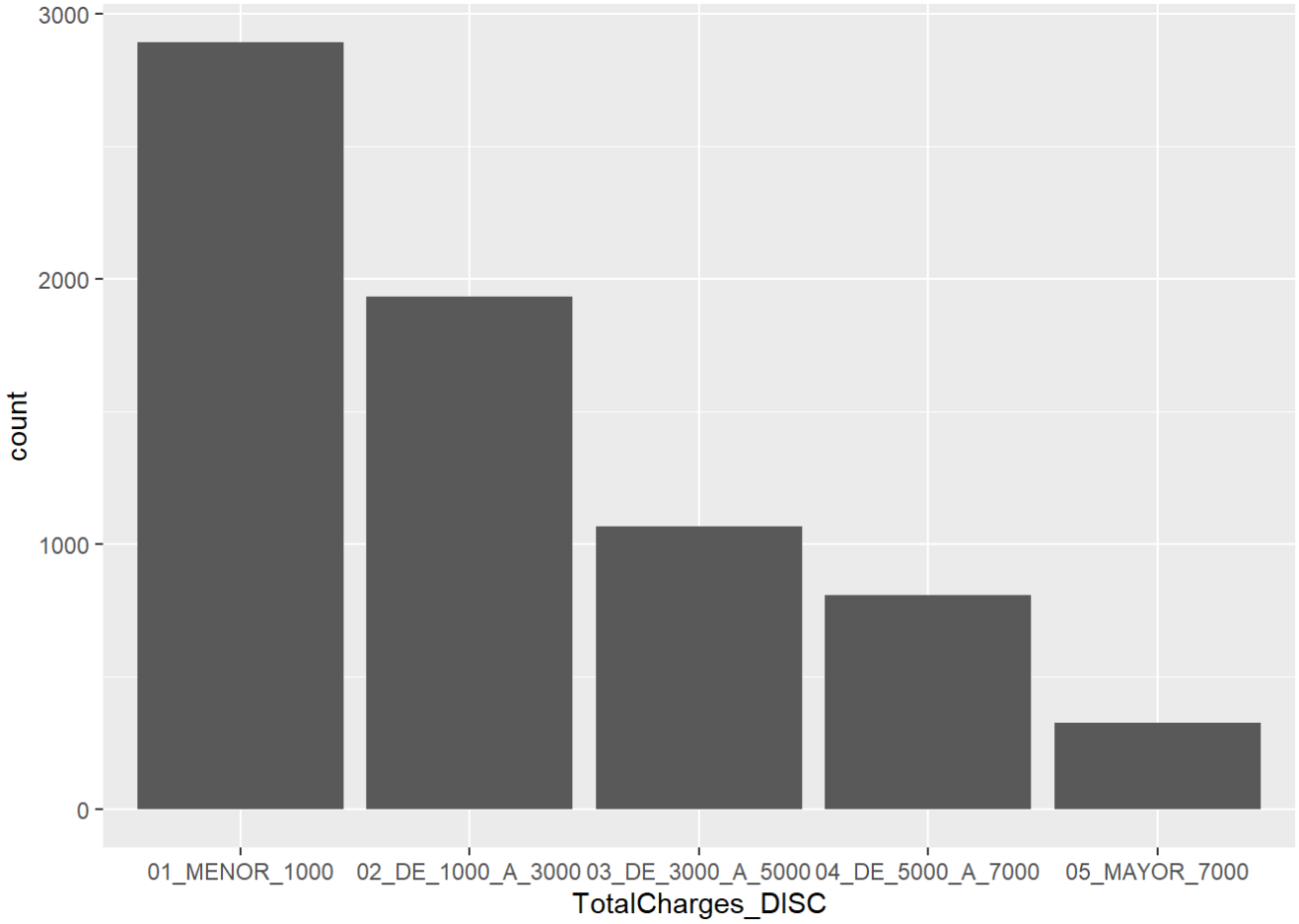
```
##      quantile(df$TotalCharges, prob = seq(0, 1, length = 11))
## 0%                                                     18.800
## 10%                                                    84.600
## 20%                                                   267.070
## 30%                                                   551.995
## 40%                                                   944.170
## 50%                                                  1397.475
## 60%                                                  2048.950
## 70%                                                  3141.130
## 80%                                                  4475.410
## 90%                                                  5976.640
## 100%                                                 8684.800
```

Procedemos a discretizarla manualmente

```
df <- df %>% mutate(TotalCharges_DISC = as.factor(case_when(
      TotalCharges <= 1000 ~ '01_MENOR_1000',
      TotalCharges > 1000 & TotalCharges <= 3000 ~ '02_DE_1000_A_3000',
      TotalCharges > 3000 & TotalCharges <= 5000 ~ '03_DE_3000_A_5000',
      TotalCharges > 5000 & TotalCharges <= 7000 ~ '04_DE_5000_A_7000',
      TotalCharges > 7000 ~ '05_MAYOR_7000',
      TRUE ~ '00_ERROR'))
)
```

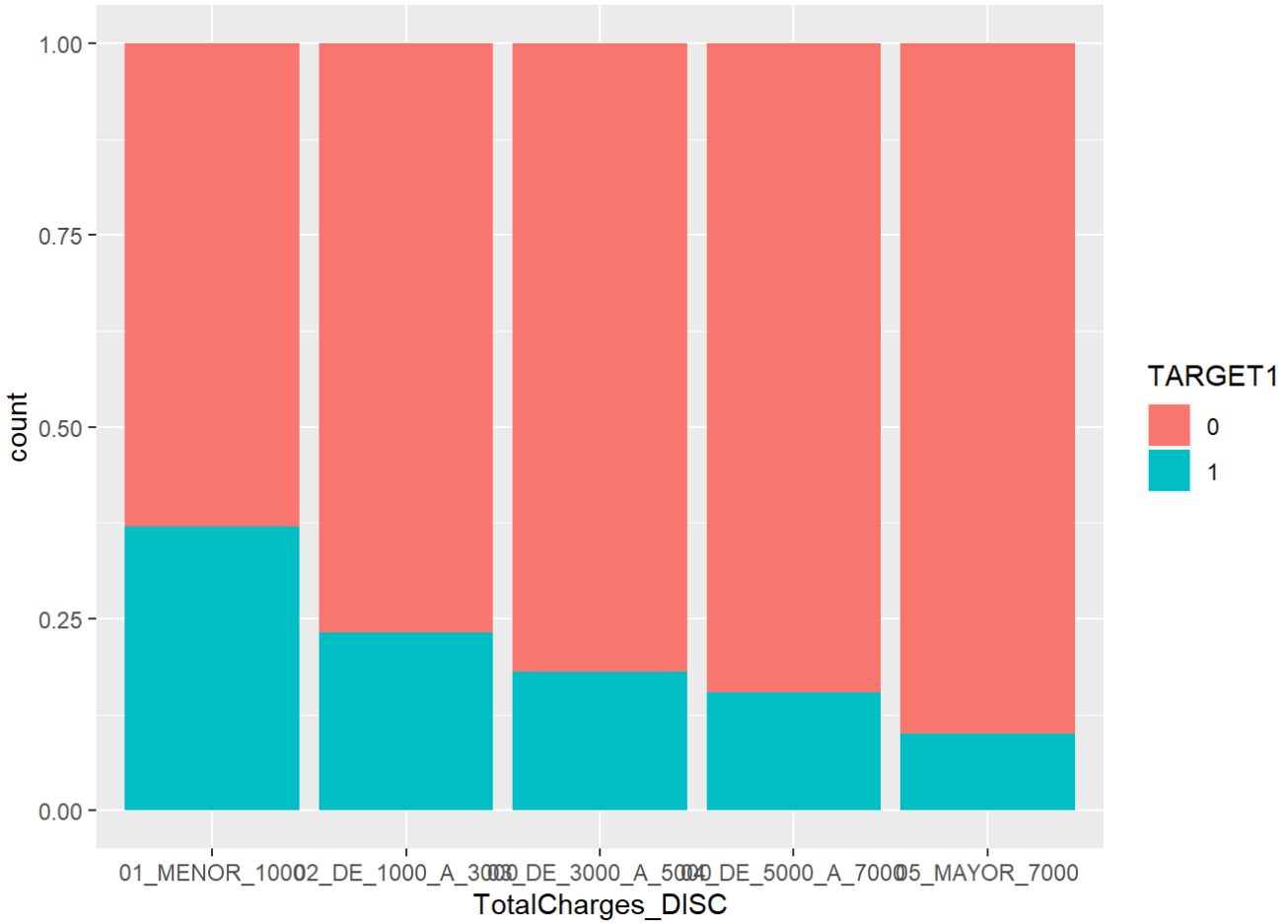
Veamos si la distribución ha quedado similar a la original

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



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

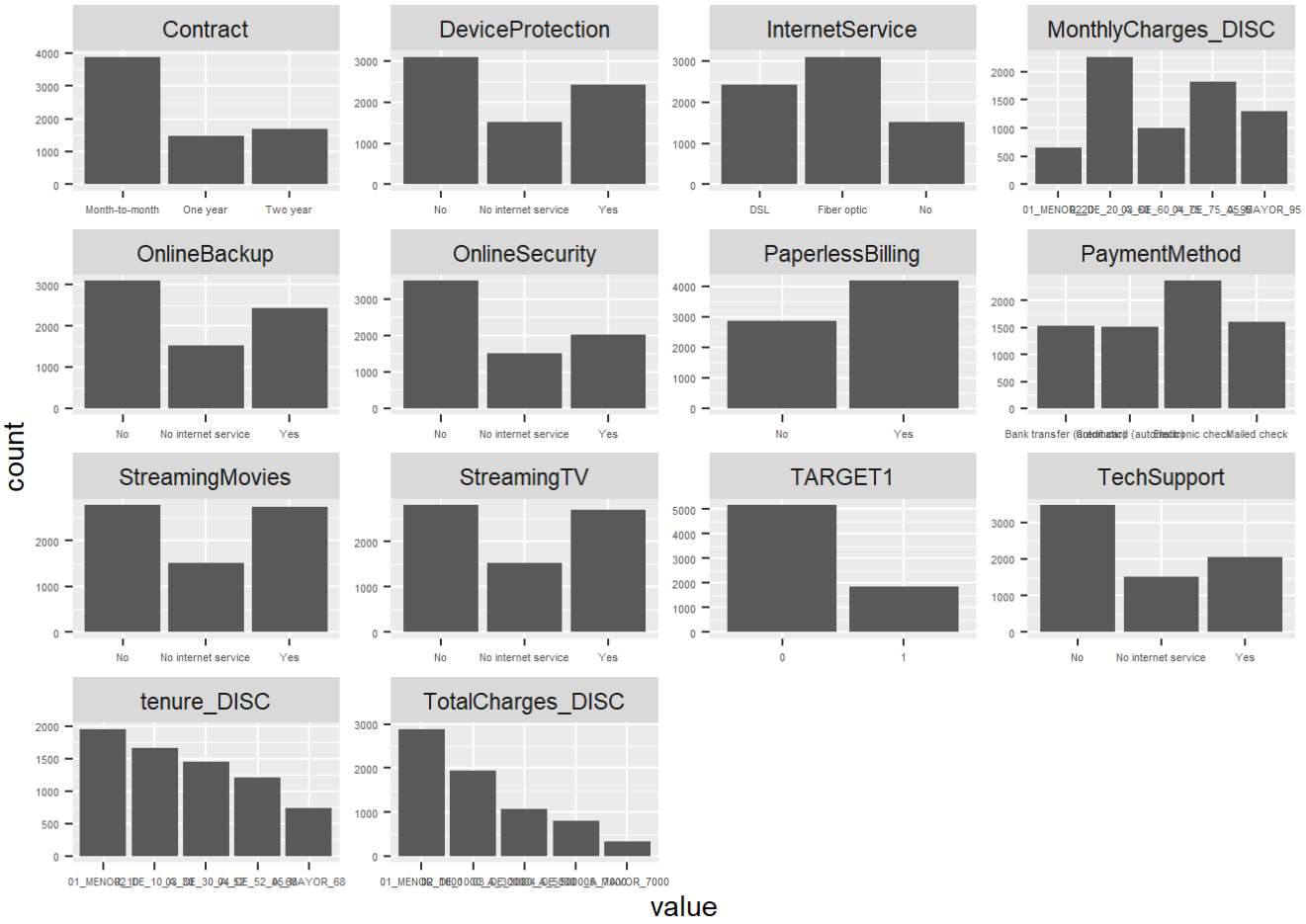
```
ggplot(df,aes(TotalCharges_DISC,fill=TARGET1)) + geom_bar(position='fill')
```



Vemos como los clientes con cargos totales de menor importe tienen menos probabilidad de abandono de la compañía.

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

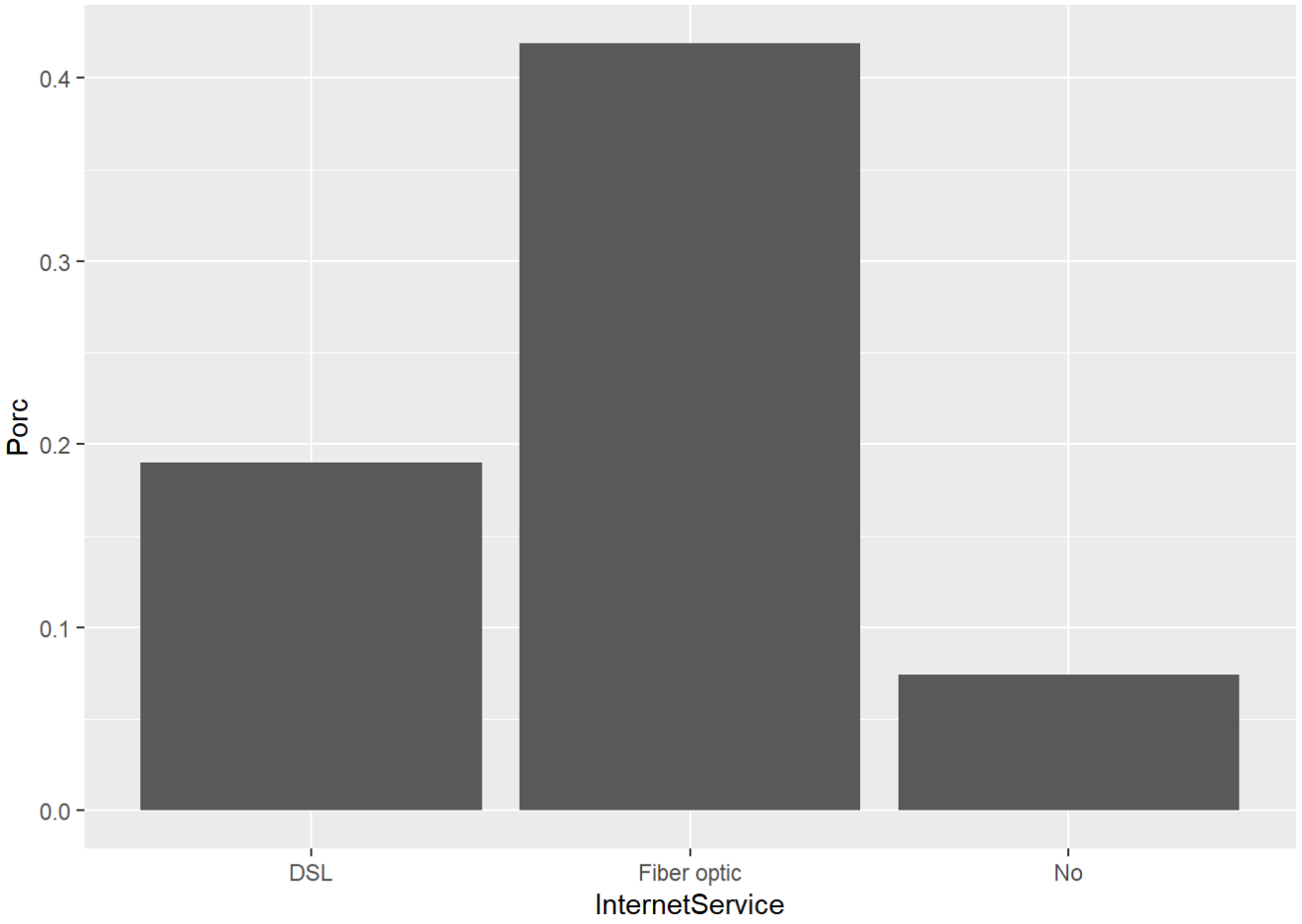
```
df %>%
  select_if(is.factor) %>%
  gather() %>%
  ggplot(aes(value)) +
    geom_bar() +
    facet_wrap(~ key, scales = "free") +
    theme(axis.text=element_text(size=4))
```



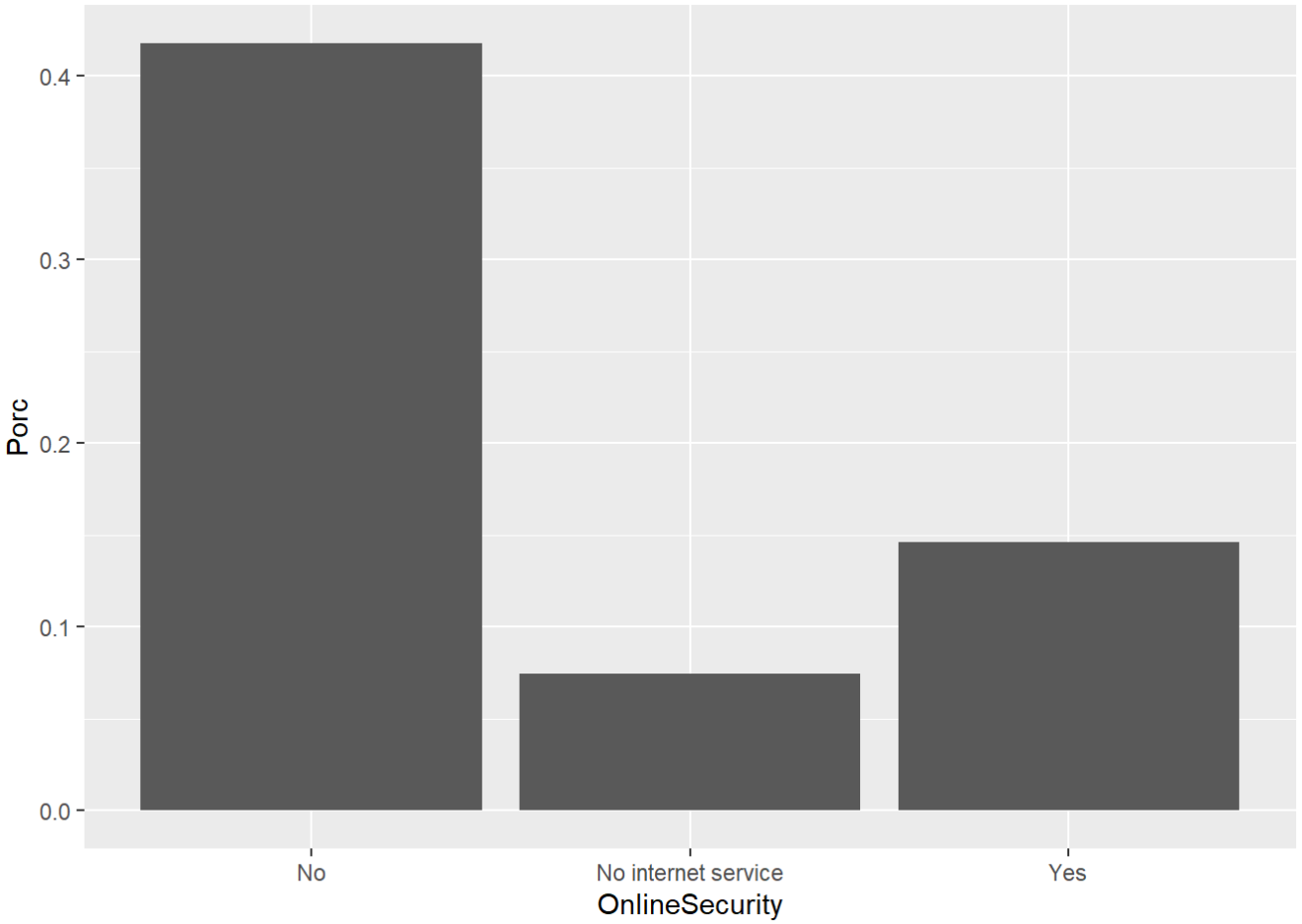
Ahora vamos a analizar la penetración de la target en cada categoría para ver si las variables han salido monotónicas

```
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')})
```

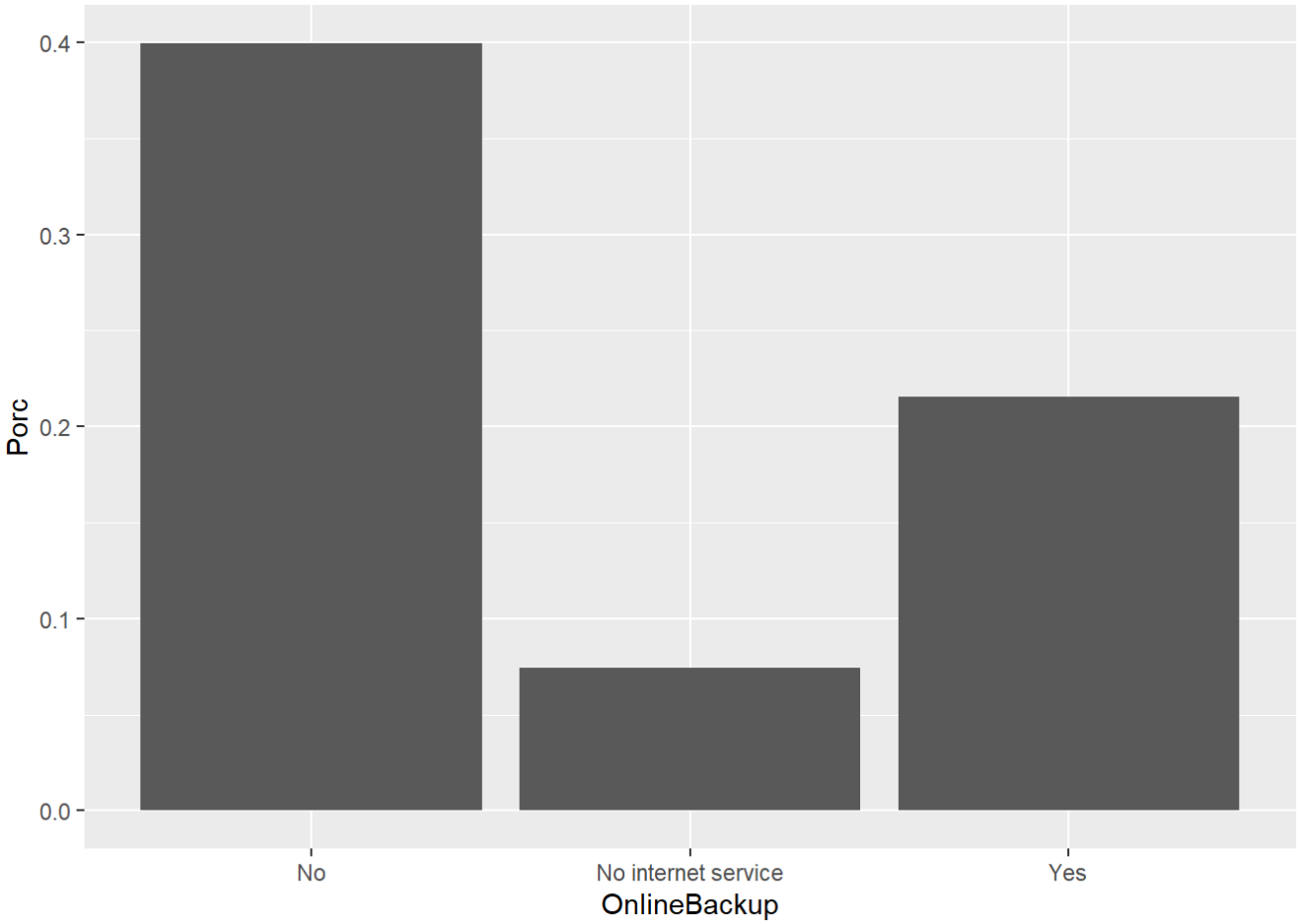
```
## [[1]]
```



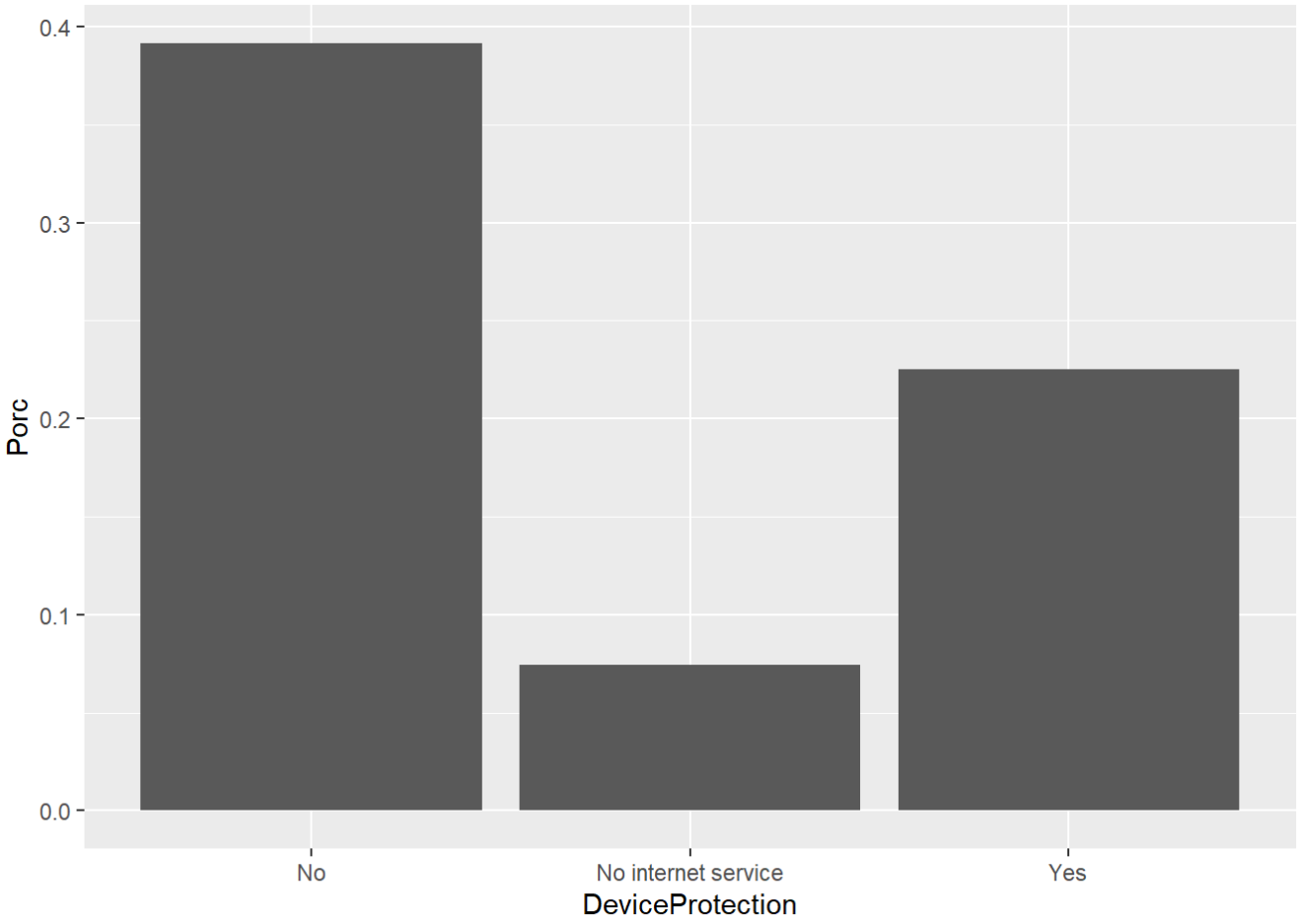

```
##
## [[2]]
```


0.4
0.3
0.2
0.1
0.0
Porc
No
No internet service
Yes
OnlineSecurity

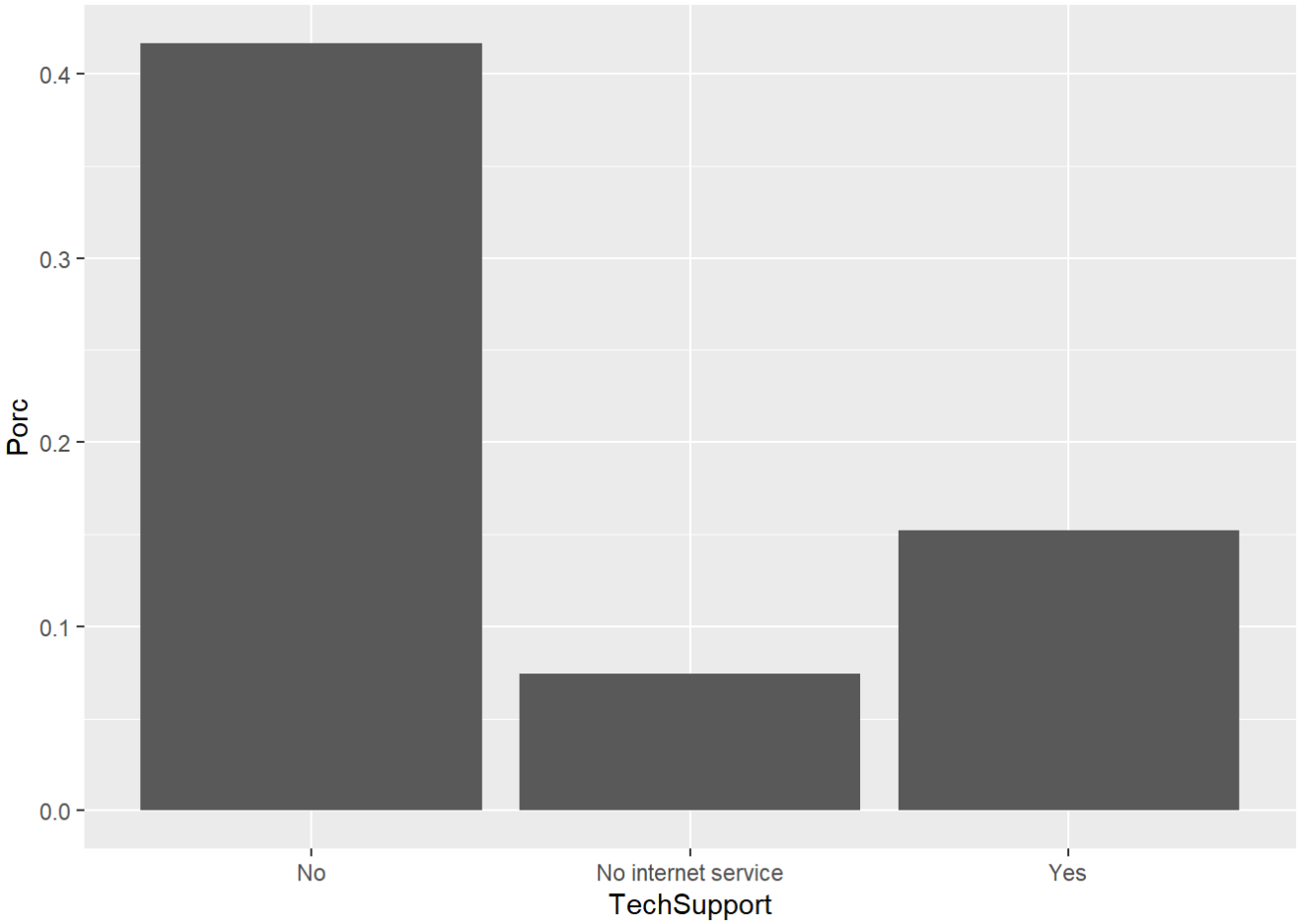
```
##
## [[3]]
```



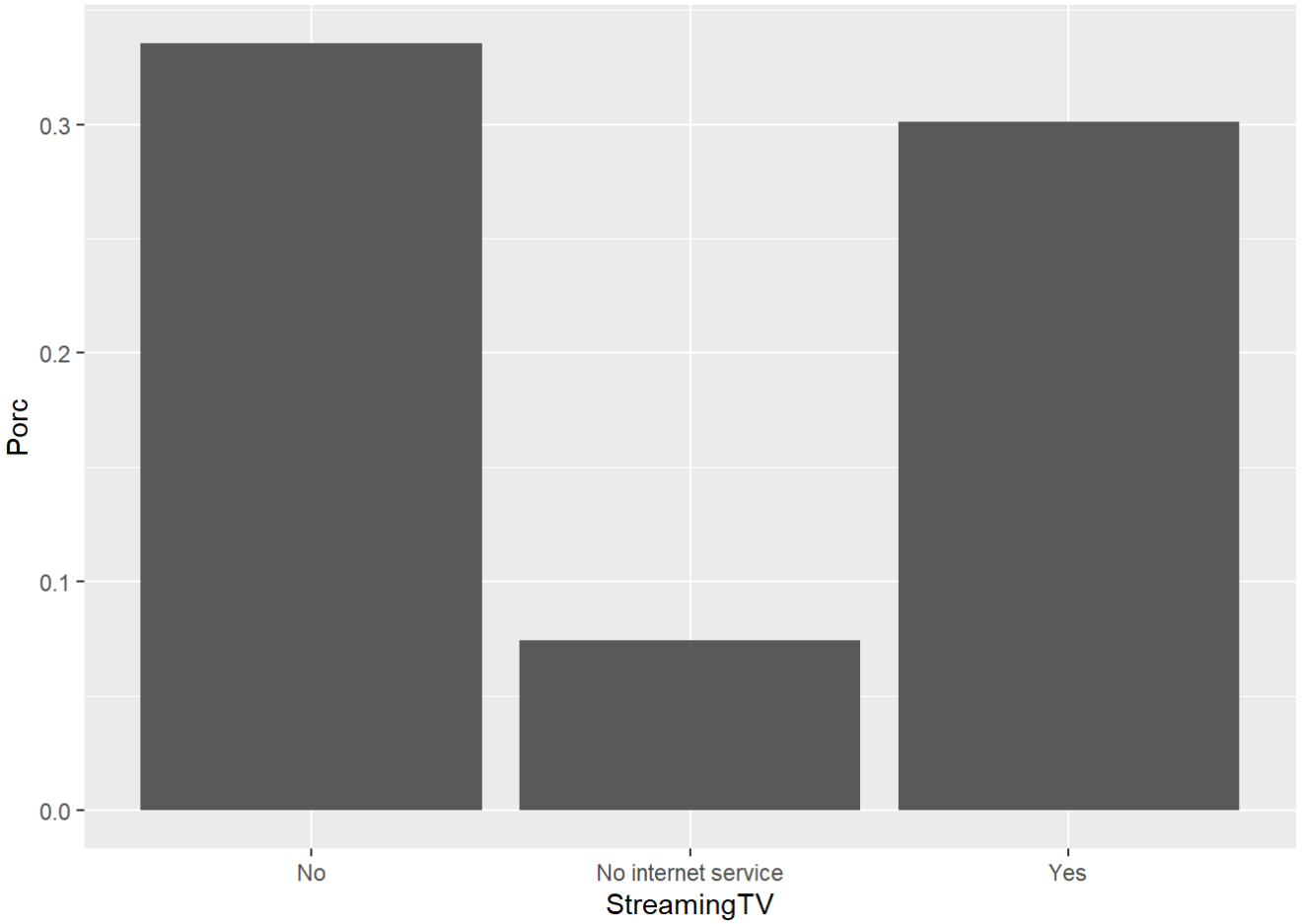

```
##
## [[4]]
```


0.4
0.3
0.2
0.1
0.0
Porc
No
No internet service
Yes
DeviceProtection

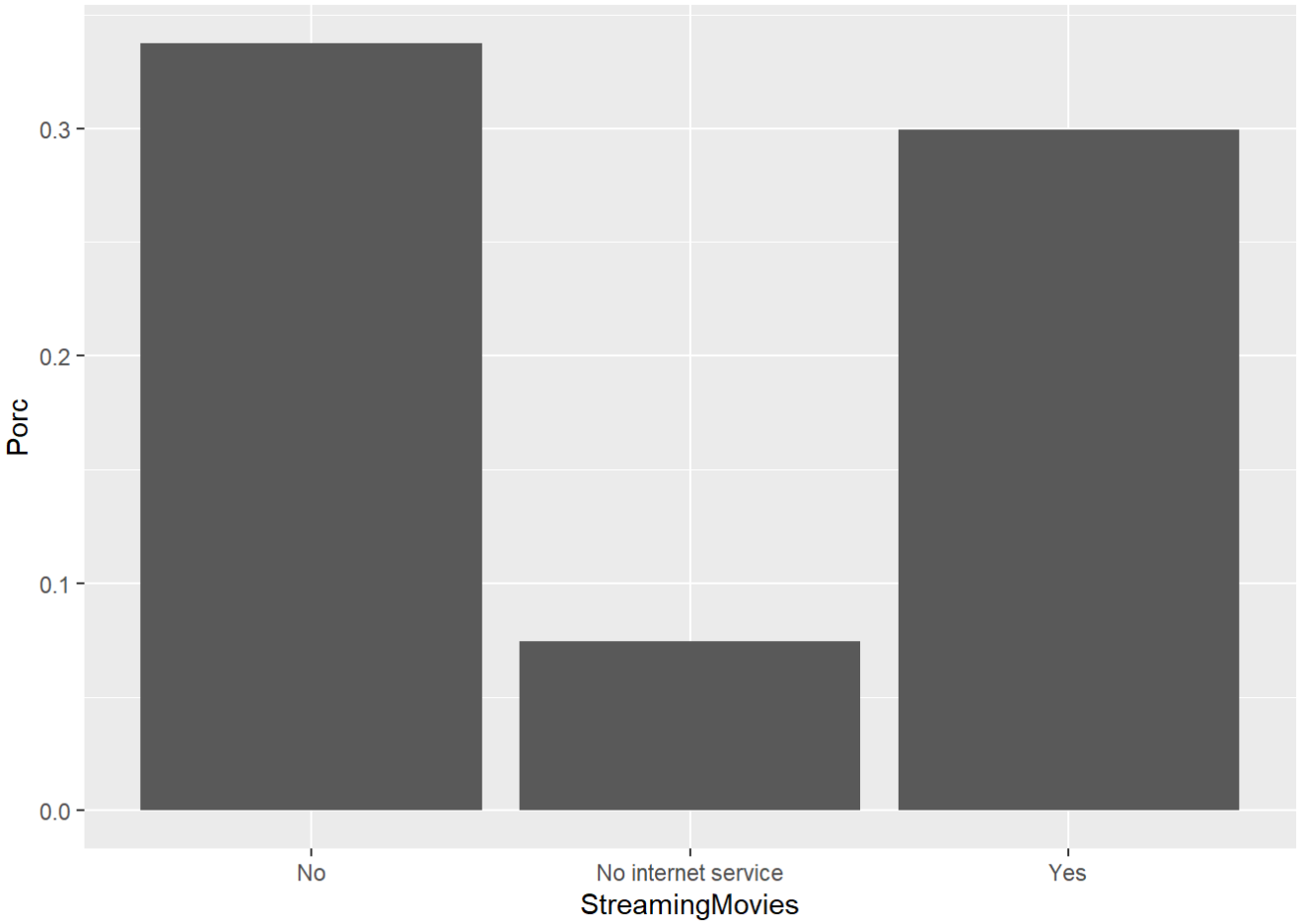
```
##
## [[5]]
```



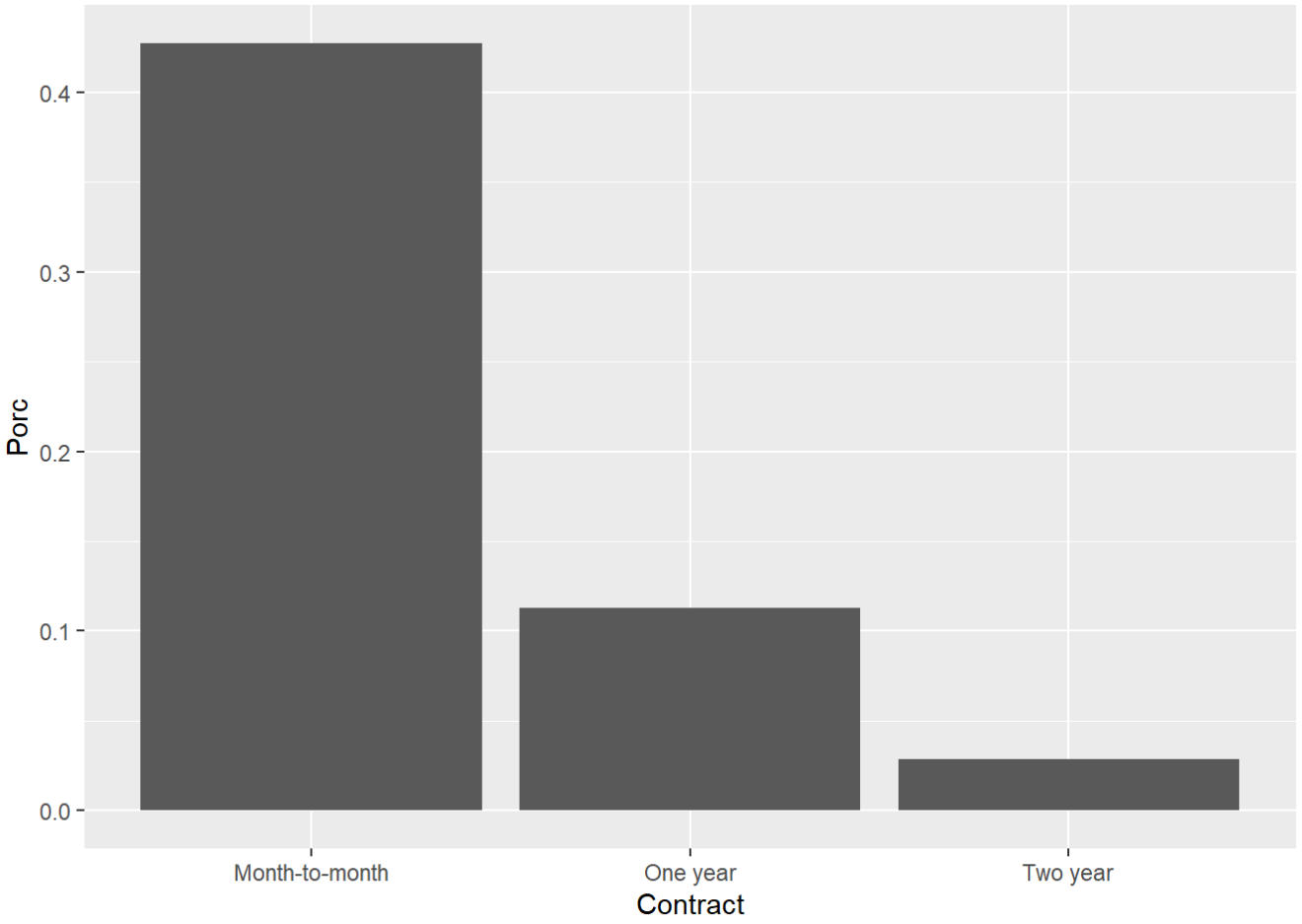

```
##
## [[6]]
```



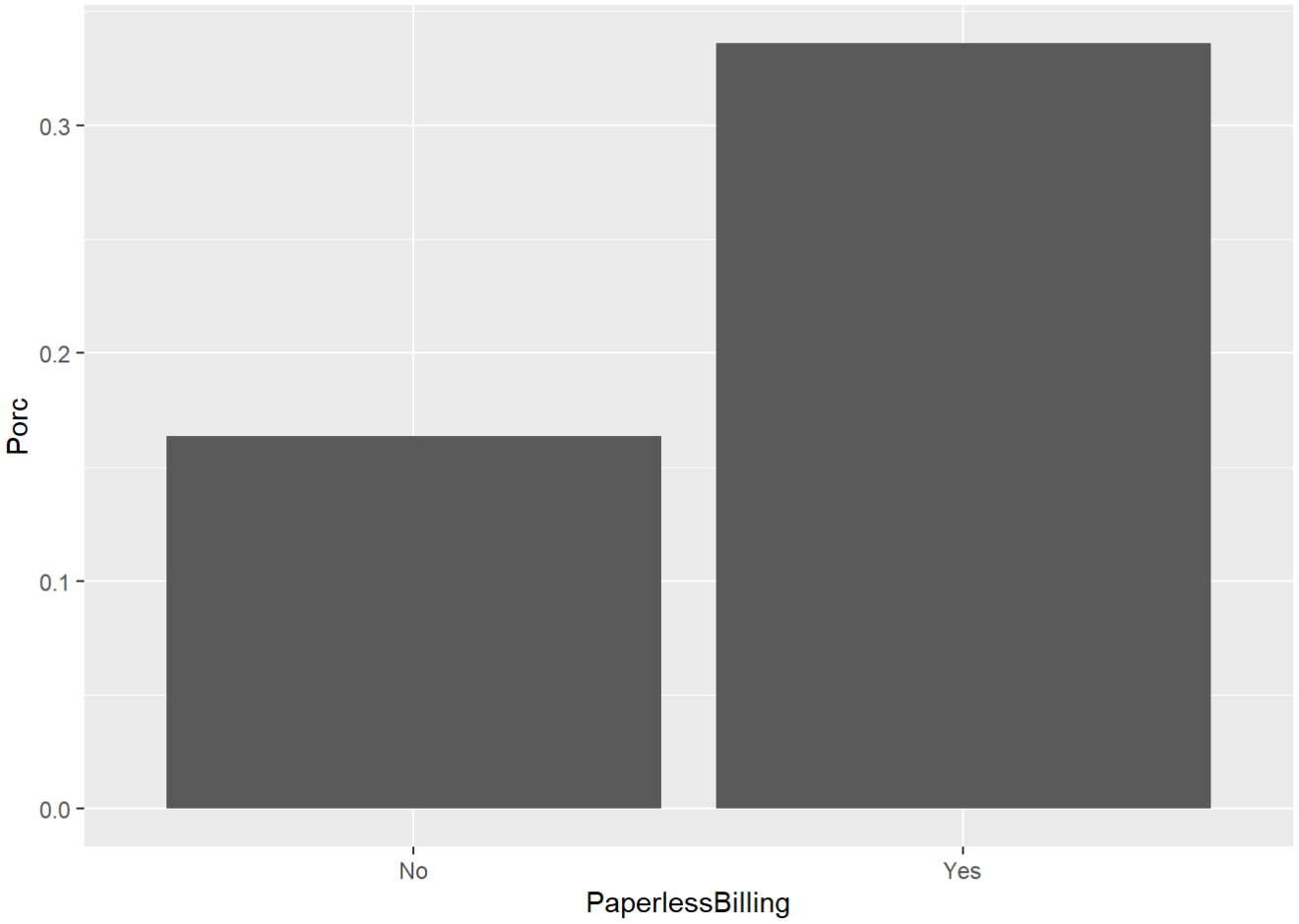

```
##
## [[7]]
```



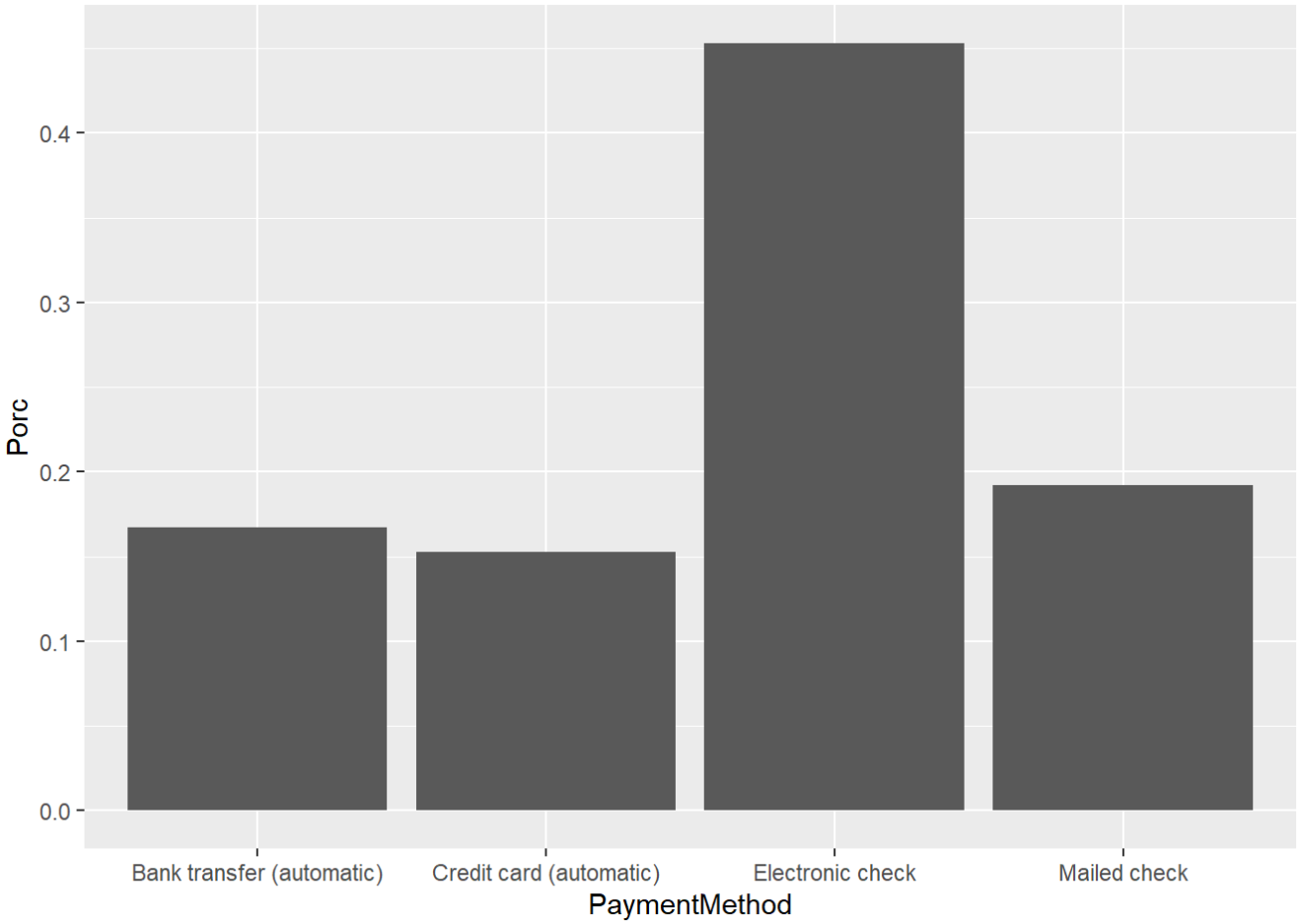

```
##
## [[8]]
```



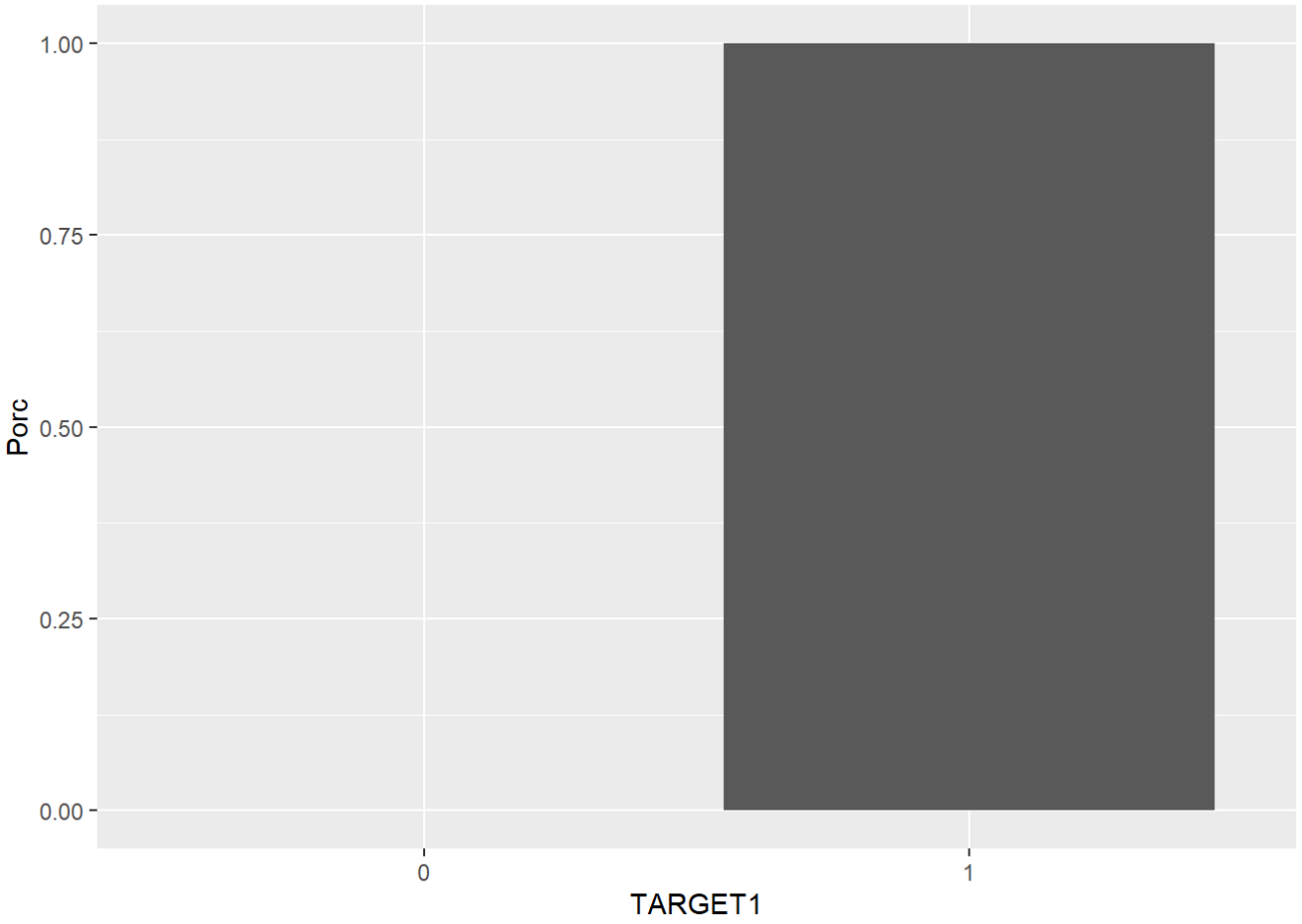

```
##
## [[9]]
```



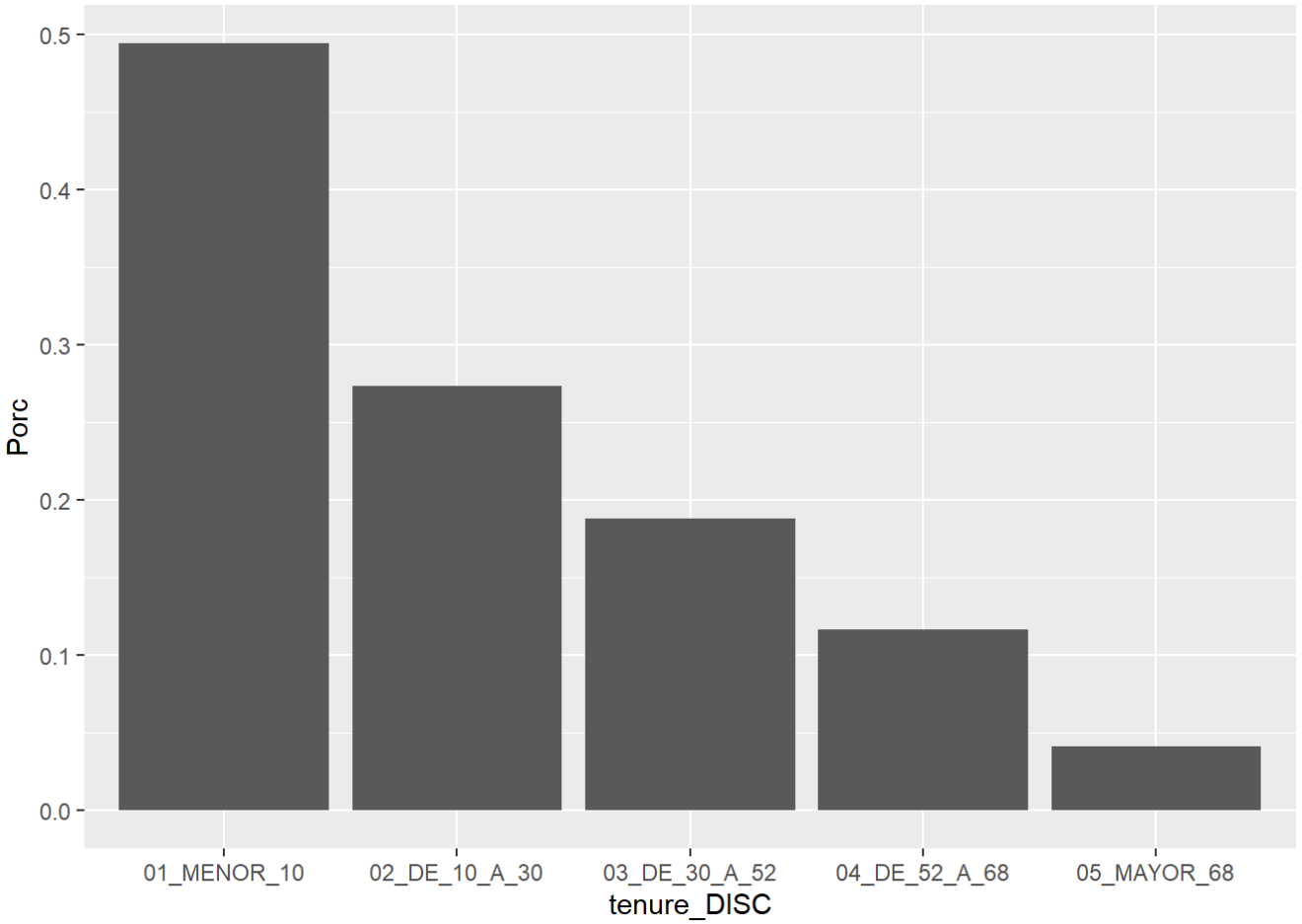

```
##
## [[10]]
```


0.4
0.3
0.2
0.1
0.0
Porc
Bank transfer (automatic)
Credit card (automatic)
Electronic check
Mailed check
PaymentMethod

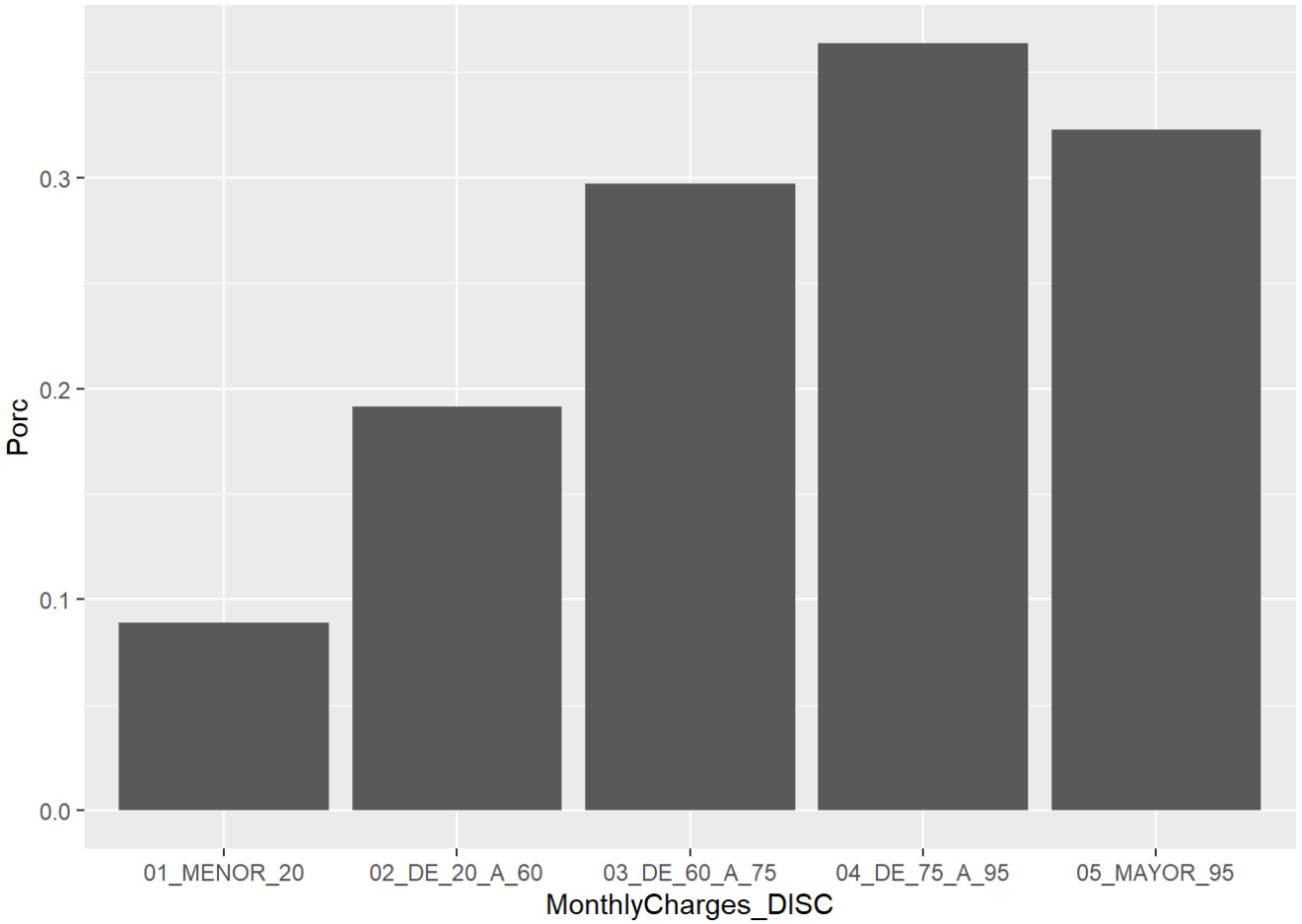
```
##
## [[11]]
```



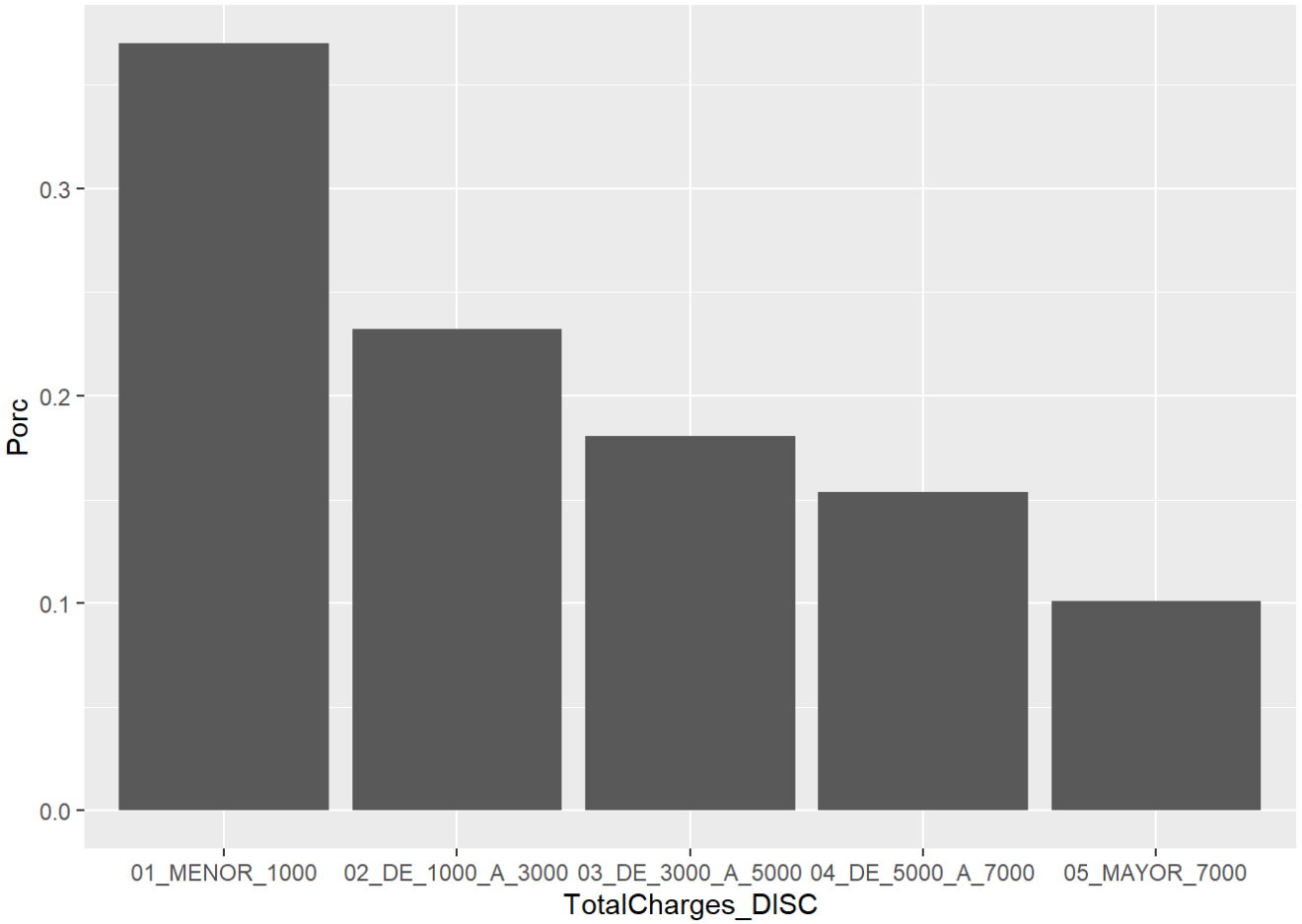

```
##
## [[12]]
```


0.5
0.4
0.3
0.2
0.1
0.0
Porc
01_MENOR_10
02_DE_10_A_30
03_DE_30_A_52
04_DE_52_A_68
05_MAYOR_68
tenure_DISC

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


0.3
0.2
0.1
0.0
Porc
01_MENOR_20
02_DE_20_A_60
03_DE_60_A_75
04_DE_75_A_95
05_MAYOR_95
MonthlyCharges_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_MonthlyCharges = disc_temp_MonthlyCharges,
 disc_temp_TotalCharges = disc_temp_TotalCharges)
saveRDS(discretizaciones,'02_CortesDiscretizaciones.rds')
```

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

```
glimpse(df)
```

```
## Rows: 7,032
## Columns: 18
## $ tenure              <int> 1, 34, 2, 45, 2, 8, 22, 10, 28, 62, 13, 16, 58, 49~
## $ InternetService     <fct> DSL, DSL, DSL, DSL, Fiber optic, Fiber optic, Fibe~
## $ OnlineSecurity      <fct> No, Yes, Yes, Yes, No, No, No, Yes, No, Yes, Yes, ~
## $ OnlineBackup        <fct> Yes, No, Yes, No, No, No, Yes, No, No, Yes, No, No~
## $ DeviceProtection    <fct> No, Yes, No, Yes, No, Yes, No, No, Yes, No, No, No~
## $ TechSupport         <fct> No, No, No, Yes, No, No, No, No, Yes, No, No, No i~
## $ StreamingTV         <fct> No, No, No, No, No, Yes, Yes, No, Yes, No, No, No ~
## $ StreamingMovies     <fct> No, No, No, No, No, Yes, No, No, Yes, No, No, No i~
## $ Contract            <fct> Month-to-month, One year, Month-to-month, One year~
## $ PaperlessBilling    <fct> Yes, No, Yes, No, Yes, Yes, Yes, No, Yes, No, Yes,~
## $ PaymentMethod       <fct> Electronic check, Mailed check, Mailed check, Bank~
## $ MonthlyCharges      <dbl> 29.85, 56.95, 53.85, 42.30, 70.70, 99.65, 89.10, 2~
## $ TotalCharges        <dbl> 29.85, 1889.50, 108.15, 1840.75, 151.65, 820.50, 1~
## $ customerID          <chr> "7590-VHVEG", "5575-GNVDE", "3668-QPYBK", "7795-CF~
## $ TARGET1             <fct> 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,~
## $ tenure_DISC         <fct> 01_MENOR_10, 03_DE_30_A_52, 01_MENOR_10, 03_DE_30_~
## $ MonthlyCharges_DISC <fct> 02_DE_20_A_60, 02_DE_20_A_60, 02_DE_20_A_60, 02_DE~
## $ TotalCharges_DISC   <fct> 01_MENOR_1000, 02_DE_1000_A_3000, 01_MENOR_1000, 0~
```

Eliminamos las variables originales de las que hemos discretizado.

```
df <- select(df,-tenure)
df <- select(df,-MonthlyCharges)
df <- select(df,-TotalCharges)
glimpse(df)
```

```
## Rows: 7,032
## Columns: 15
## $ InternetService     <fct> DSL, DSL, DSL, DSL, Fiber optic, Fiber optic, Fibe~
## $ OnlineSecurity      <fct> No, Yes, Yes, Yes, No, No, No, Yes, No, Yes, Yes, ~
## $ OnlineBackup        <fct> Yes, No, Yes, No, No, No, Yes, No, No, Yes, No, No~
## $ DeviceProtection    <fct> No, Yes, No, Yes, No, Yes, No, No, Yes, No, No, No~
## $ TechSupport         <fct> No, No, No, Yes, No, No, No, No, Yes, No, No, No i~
## $ StreamingTV         <fct> No, No, No, No, No, Yes, Yes, No, Yes, No, No, No ~
## $ StreamingMovies     <fct> No, No, No, No, No, Yes, No, No, Yes, No, No, No i~
## $ Contract            <fct> Month-to-month, One year, Month-to-month, One year~
## $ PaperlessBilling    <fct> Yes, No, Yes, No, Yes, Yes, Yes, No, Yes, No, Yes,~
## $ PaymentMethod       <fct> Electronic check, Mailed check, Mailed check, Bank~
## $ customerID          <chr> "7590-VHVEG", "5575-GNVDE", "3668-QPYBK", "7795-CF~
## $ TARGET1             <fct> 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,~
## $ tenure_DISC         <fct> 01_MENOR_10, 03_DE_30_A_52, 01_MENOR_10, 03_DE_30_~
## $ MonthlyCharges_DISC <fct> 02_DE_20_A_60, 02_DE_20_A_60, 02_DE_20_A_60, 02_DE~
## $ TotalCharges_DISC   <fct> 01_MENOR_1000, 02_DE_1000_A_3000, 01_MENOR_1000, 0~
```

Ordenamos las variables.

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

Comprobamos de nuevo.

```
glimpse(df)
```

```
## Rows: 7,032
## Columns: 15
## $ customerID          <chr> "7590-VHVEG", "5575-GNVDE", "3668-QPYBK", "7795-CF~
## $ InternetService     <fct> DSL, DSL, DSL, DSL, Fiber optic, Fiber optic, Fibe~
## $ OnlineSecurity      <fct> No, Yes, Yes, Yes, No, No, No, Yes, No, Yes, Yes, ~
## $ OnlineBackup        <fct> Yes, No, Yes, No, No, No, Yes, No, No, Yes, No, No~
## $ DeviceProtection    <fct> No, Yes, No, Yes, No, Yes, No, No, Yes, No, No, No~
## $ TechSupport         <fct> No, No, No, Yes, No, No, No, No, Yes, No, No, No i~
## $ StreamingTV         <fct> No, No, No, No, No, Yes, Yes, No, Yes, No, No, No ~
## $ StreamingMovies     <fct> No, No, No, No, No, Yes, No, No, Yes, No, No, No i~
## $ Contract            <fct> Month-to-month, One year, Month-to-month, One year~
## $ PaperlessBilling    <fct> Yes, No, Yes, No, Yes, Yes, Yes, No, Yes, No, Yes,~
## $ PaymentMethod       <fct> Electronic check, Mailed check, Mailed check, Bank~
## $ tenure_DISC         <fct> 01_MENOR_10, 03_DE_30_A_52, 01_MENOR_10, 03_DE_30_~
## $ MonthlyCharges_DISC <fct> 02_DE_20_A_60, 02_DE_20_A_60, 02_DE_20_A_60, 02_DE~
## $ TotalCharges_DISC   <fct> 01_MENOR_1000, 02_DE_1000_A_3000, 01_MENOR_1000, 0~
## $ TARGET1             <fct> 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,~
```

- 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.

```
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,ti
mes=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.

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] "InternetService"     "OnlineSecurity"      "OnlineBackup"
##  [4] "DeviceProtection"    "TechSupport"         "StreamingTV"
##  [7] "StreamingMovies"     "Contract"            "PaperlessBilling"
## [10] "PaymentMethod"       "tenure_DISC"         "MonthlyCharges_DISC"
## [13] "TotalCharges_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
## -1.8446  -0.6833  -0.2909   0.7037   3.1585
##
## Coefficients: (6 not defined because of singularities)
##                                     Estimate Std. Error z value
## (Intercept)                         -0.31208    0.27873  -1.120
## InternetServiceFiber optic           1.07259    0.21864   4.906
## InternetServiceNo                   -1.10999    0.19568  -5.672
## OnlineSecurityNo internet service          NA         NA      NA
## OnlineSecurityYes                   -0.33907    0.10376  -3.268
## OnlineBackupNo internet service            NA         NA      NA
## OnlineBackupYes                     -0.17675    0.09312  -1.898
## DeviceProtectionNo internet service        NA         NA      NA
## DeviceProtectionYes                  0.06991    0.09660   0.724
## TechSupportNo internet service             NA         NA      NA
## TechSupportYes                      -0.37192    0.10540  -3.529
## StreamingTVNo internet service             NA         NA      NA
## StreamingTVYes                       0.26927    0.10797   2.494
## StreamingMoviesNo internet service         NA         NA      NA
## StreamingMoviesYes                   0.16021    0.10790   1.485
## ContractOne year                    -0.83324    0.12578  -6.625
## ContractTwo year                    -1.66446    0.21715  -7.665
## PaperlessBillingYes                  0.28493    0.08835   3.225
## PaymentMethodCredit card (automatic) -0.07360    0.13571  -0.542
## PaymentMethodElectronic check        0.36740    0.11261   3.263
## PaymentMethodMailed check            0.04098    0.13504   0.303
## tenure_DISC02_DE_10_A_30            -1.02530    0.14899  -6.882
## tenure_DISC03_DE_30_A_52            -1.11311    0.21759  -5.116
## tenure_DISC04_DE_52_A_68            -1.27614    0.29092  -4.387
## tenure_DISC05_MAYOR_68              -1.80875    0.42789  -4.227
## MonthlyCharges_DISC02_DE_20_A_60    -0.01577    0.23570  -0.067
## MonthlyCharges_DISC03_DE_60_A_75    -0.45596    0.30191  -1.510
## MonthlyCharges_DISC04_DE_75_A_95    -0.41244    0.34213  -1.206
## MonthlyCharges_DISC05_MAYOR_95      -0.10205    0.40715  -0.251
## TotalCharges_DISC02_DE_1000_A_3000   0.06422    0.16992   0.378
## TotalCharges_DISC03_DE_3000_A_5000  -0.28027    0.25913  -1.082
## TotalCharges_DISC04_DE_5000_A_7000  -0.32145    0.33843  -0.950
## TotalCharges_DISC05_MAYOR_7000      -0.44840    0.46295  -0.969
##                                            Pr(>|z|)
```

```
## (Intercept)                                    0.262866
## InternetServiceFiber optic           0.0000009310375077 ***
## InternetServiceNo                    0.0000000140836863 ***
## OnlineSecurityNo internet service                   NA
## OnlineSecurityYes                              0.001084 **
## OnlineBackupNo internet service                     NA
## OnlineBackupYes                                0.057685 .
## DeviceProtectionNo internet service                 NA
## DeviceProtectionYes                            0.469266
## TechSupportNo internet service                      NA
## TechSupportYes                                 0.000418 ***
## StreamingTVNo internet service                      NA
## StreamingTVYes                                 0.012631 *
## StreamingMoviesNo internet service                  NA
## StreamingMoviesYes                             0.137598
## ContractOne year                     0.0000000000347894 ***
## ContractTwo year                     0.0000000000000179 ***
## PaperlessBillingYes                            0.001260 **
## PaymentMethodCredit card (automatic)           0.587606
## PaymentMethodElectronic check                  0.001104 **
## PaymentMethodMailed check                      0.761533
## tenure_DISC02_DE_10_A_30             0.0000000000059185 ***
## tenure_DISC03_DE_30_A_52             0.0000003127468797 ***
## tenure_DISC04_DE_52_A_68             0.0000115152755063 ***
## tenure_DISC05_MAYOR_68               0.0000236630559442 ***
## MonthlyCharges_DISC02_DE_20_A_60               0.946669
## MonthlyCharges_DISC03_DE_60_A_75               0.130986
## MonthlyCharges_DISC04_DE_75_A_95               0.228009
## MonthlyCharges_DISC05_MAYOR_95                 0.802095
## TotalCharges_DISC02_DE_1000_A_3000             0.705486
## TotalCharges_DISC03_DE_3000_A_5000             0.279450
## TotalCharges_DISC04_DE_5000_A_7000             0.342209
## TotalCharges_DISC05_MAYOR_7000                 0.332757
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5727.0  on 4923  degrees of freedom
## Residual deviance: 4159.5  on 4897  degrees of freedom
## AIC: 4213.5
##
## Number of Fisher Scoring iterations: 6
```

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

```
a_mantener <- c(
  'InternetService',
  'OnlineSecurity',
  'StreamingMovies',
  'Contract',
  'PaperlessBilling',
  'tenure_DISC'
)
a_mantener
```

```
## [1] "InternetService"  "OnlineSecurity"   "StreamingMovies"  "Contract"
## [5] "PaperlessBilling" "tenure_DISC"
```

Volvemos a modelizar 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
## -1.7120  -0.6996  -0.2858   0.7247   3.0708
##
## Coefficients: (2 not defined because of singularities)
##                                   Estimate Std. Error z value
## (Intercept)                       -0.34041    0.09772  -3.483
## InternetServiceFiber optic         0.93539    0.08941  10.462
## InternetServiceNo                 -0.98711    0.14411  -6.850
## OnlineSecurityNo internet service        NA         NA      NA
## OnlineSecurityYes                 -0.43967    0.09882  -4.449
## StreamingMoviesNo internet service       NA         NA      NA
## StreamingMoviesYes                 0.28970    0.08651   3.349
## ContractOne year                  -0.94514    0.12198  -7.749
## ContractTwo year                  -1.85847    0.21226  -8.756
## PaperlessBillingYes                0.31826    0.08664   3.673
## tenure_DISC02_DE_10_A_30          -1.00629    0.09765 -10.305
## tenure_DISC03_DE_30_A_52          -1.26061    0.11762 -10.718
## tenure_DISC04_DE_52_A_68          -1.51990    0.15184 -10.010
## tenure_DISC05_MAYOR_68            -2.14414    0.29714  -7.216
##                                             Pr(>|z|)
## (Intercept)                                 0.000495 ***
## InternetServiceFiber optic         < 0.0000000000000002 ***
## InternetServiceNo                   0.00000000000739853 ***
## OnlineSecurityNo internet service                   NA
## OnlineSecurityYes                   0.00000862014571325 ***
## StreamingMoviesNo internet service                  NA
## StreamingMoviesYes                          0.000812 ***
## ContractOne year                    0.00000000000000929 ***
## ContractTwo year                   < 0.0000000000000002 ***
## PaperlessBillingYes                         0.000239 ***
## tenure_DISC02_DE_10_A_30           < 0.0000000000000002 ***
## tenure_DISC03_DE_30_A_52           < 0.0000000000000002 ***
## tenure_DISC04_DE_52_A_68           < 0.0000000000000002 ***
## tenure_DISC05_MAYOR_68              0.00000000000053591 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 5727.0  on 4923  degrees of freedom
## Residual deviance: 4232.7  on 4912  degrees of freedom
## AIC: 4256.7
##
## Number of Fisher Scoring iterations: 6
```

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

Y calculamos el pseudo R cuadrado:

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

```
## [1] 0.2609168
```

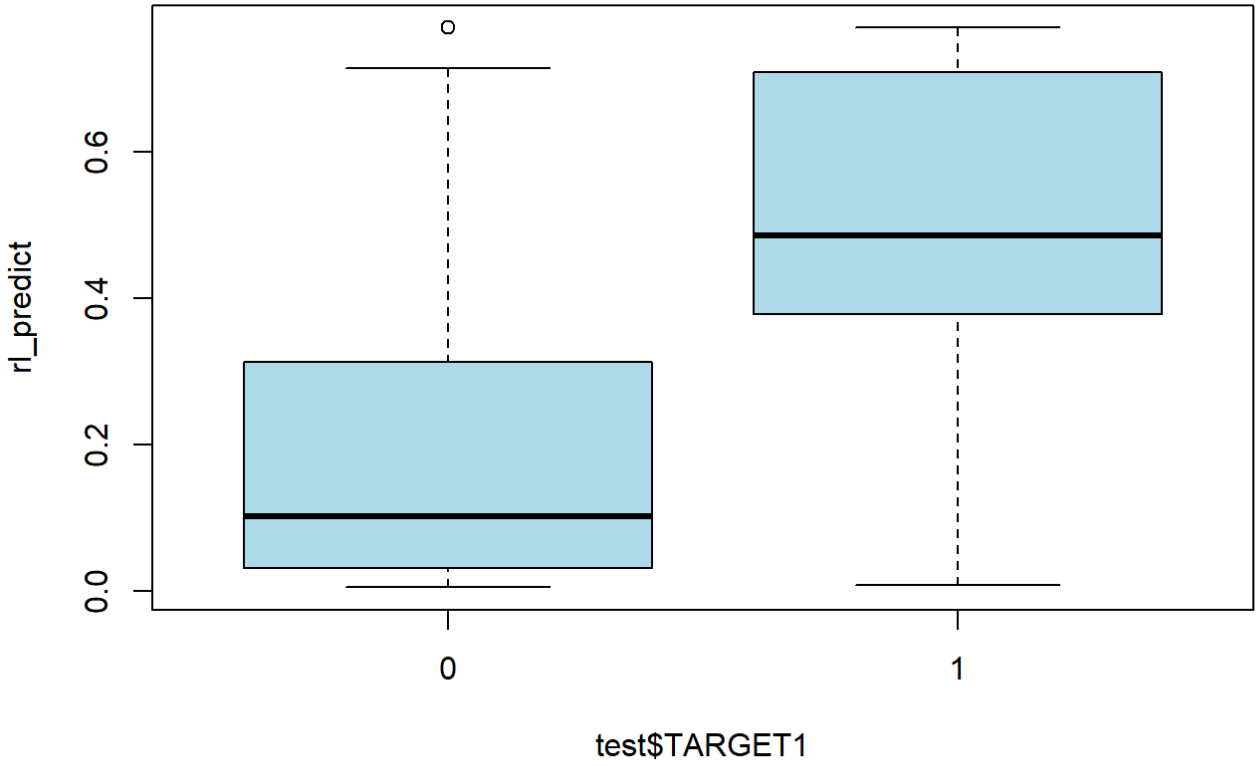
Aplicamos el modelo al conjunto de test, generando un vector con las probabilidades

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

```
##          1          2          3          4          5          6
## 0.04807015 0.54900342 0.01488682 0.48558675 0.03227923 0.48558675
```

Comprobamos en el gráfico:

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



Vemos en el gráfico que el scoring de los clientes que abandonan la empresa es muy baja en torno al 10 %, en contraste con los clientes que no abanadonan que ronda el 50 %.

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 51.80266  34.88372 98.540146 51.52672
## 2    0.10 61.57495  39.92308 94.708029 56.16883
## 3    0.15 64.84820  42.00497 92.518248 57.77778
## 4    0.20 67.78937  44.06165 88.686131 58.87341
## 5    0.25 72.96015  48.83475 84.124088 61.79625
## 6    0.30 76.04364  52.59349 79.562044 63.32607
## 7    0.35 78.27324  56.00000 76.642336 64.71495
## 8    0.40 79.88615  59.19881 72.810219 65.30278
## 9    0.45 80.17078  62.12687 60.766423 61.43911
## 10   0.50 80.45541  70.00000 43.430657 53.60360
## 11   0.55 79.41176  72.09302 33.941606 46.15385
## 12   0.60 78.98482  71.96653 31.386861 43.71029
## 13   0.65 78.70019  75.38462 26.824818 39.56931
## 14   0.70 78.36812  75.00000 25.182482 37.70492
## 15   0.75 75.56926  81.13208  7.846715 14.30948
## 16   0.80  0.80000   0.80000  0.800000  0.80000
## 17   0.85  0.85000   0.85000  0.850000  0.85000
## 18   0.90  0.90000   0.90000  0.900000  0.90000
## 19   0.95  0.95000   0.95000  0.950000  0.95000
```

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)
```

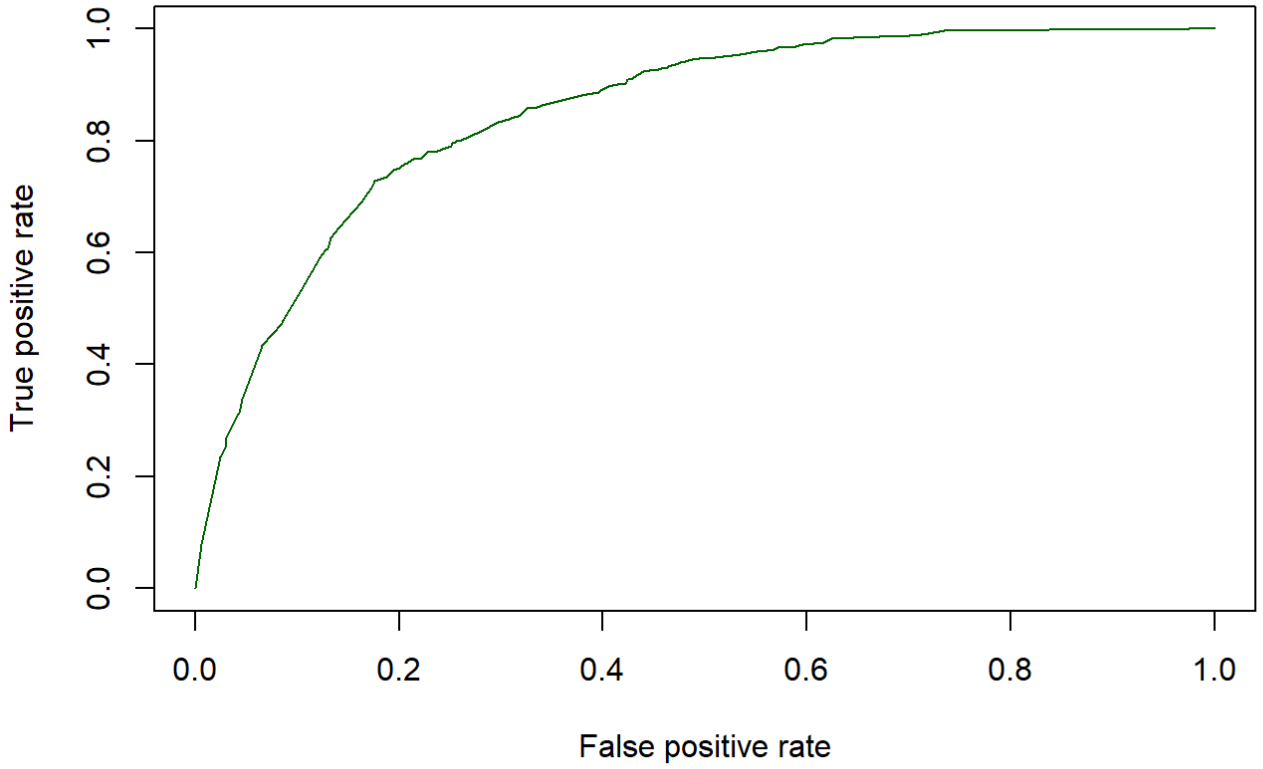
```
##
## real FALSE TRUE
##    0  1285  275
##    1   149  399
```

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

```
##   umbral  acierto precision cobertura       F1
## 1    0.4 79.88615  59.19881  72.81022 65.30278
```

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   79.88615
## precision 59.19881
## cobertura 72.81022
## F1        65.30278
## AUC       85.00000
```

- 4.5.5 - Modelizamos con Arboles de decisión

Creamos el primer modelo

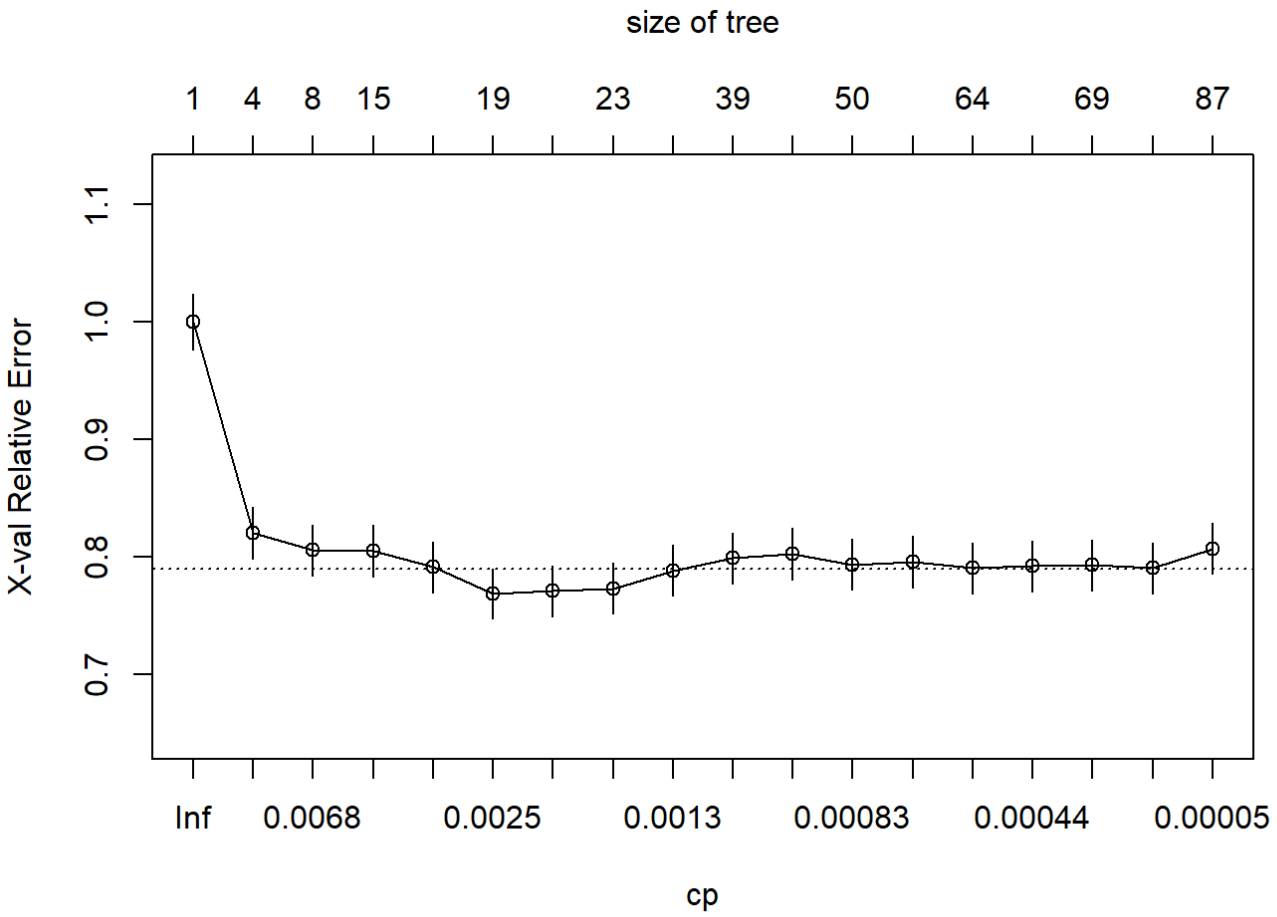
```
formula_ar <- formula
ar<-rpart(formula_ar, train, method = 'class', parms = list(
  split = "information"),
  control = rpart.control(cp = 0.00001))
```

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 = "inf
ormation"),
##     control = rpart.control(cp = 0.00001))
##
## Variables actually used in tree construction:
##  [1] Contract            DeviceProtection    InternetService
##  [4] MonthlyCharges_DISC OnlineBackup        OnlineSecurity
##  [7] PaperlessBilling    PaymentMethod       StreamingMovies
## [10] StreamingTV         TechSupport         tenure_DISC
## [13] TotalCharges_DISC
##
## Root node error: 1321/4924 = 0.26828
##
## n= 4924
##
##            CP nsplit rel error  xerror     xstd
## 1  0.05109765      0   1.00000 1.00000 0.023535
## 2  0.01021953      3   0.80924 0.81983 0.022003
## 3  0.00454201      7   0.76609 0.80545 0.021863
## 4  0.00378501     14   0.73126 0.80469 0.021855
## 5  0.00264951     16   0.72369 0.79107 0.021720
## 6  0.00227101     18   0.71840 0.76836 0.021488
## 7  0.00189251     20   0.71385 0.77063 0.021512
## 8  0.00151400     22   0.71007 0.77290 0.021535
## 9  0.00113550     25   0.70553 0.78804 0.021689
## 10 0.00100934     38   0.69039 0.79864 0.021795
## 11 0.00090840     44   0.68433 0.80242 0.021833
## 12 0.00075700     49   0.67979 0.79334 0.021743
## 13 0.00056775     57   0.67373 0.79561 0.021765
## 14 0.00050467     63   0.66995 0.79031 0.021712
## 15 0.00037850     66   0.66843 0.79182 0.021727
## 16 0.00033645     68   0.66768 0.79258 0.021735
## 17 0.00025233     81   0.66238 0.79031 0.021712
## 18 0.00001000     86   0.66086 0.80696 0.021878
```

```
plotcp(ar)
```



Parece que minimiza aprox en 0.003 de complejidad Generamos un nuevo árbol con ese parámetro Ademas vamos a incluir un nuevo parametro para que el árbol no tenga mas de 7 niveles

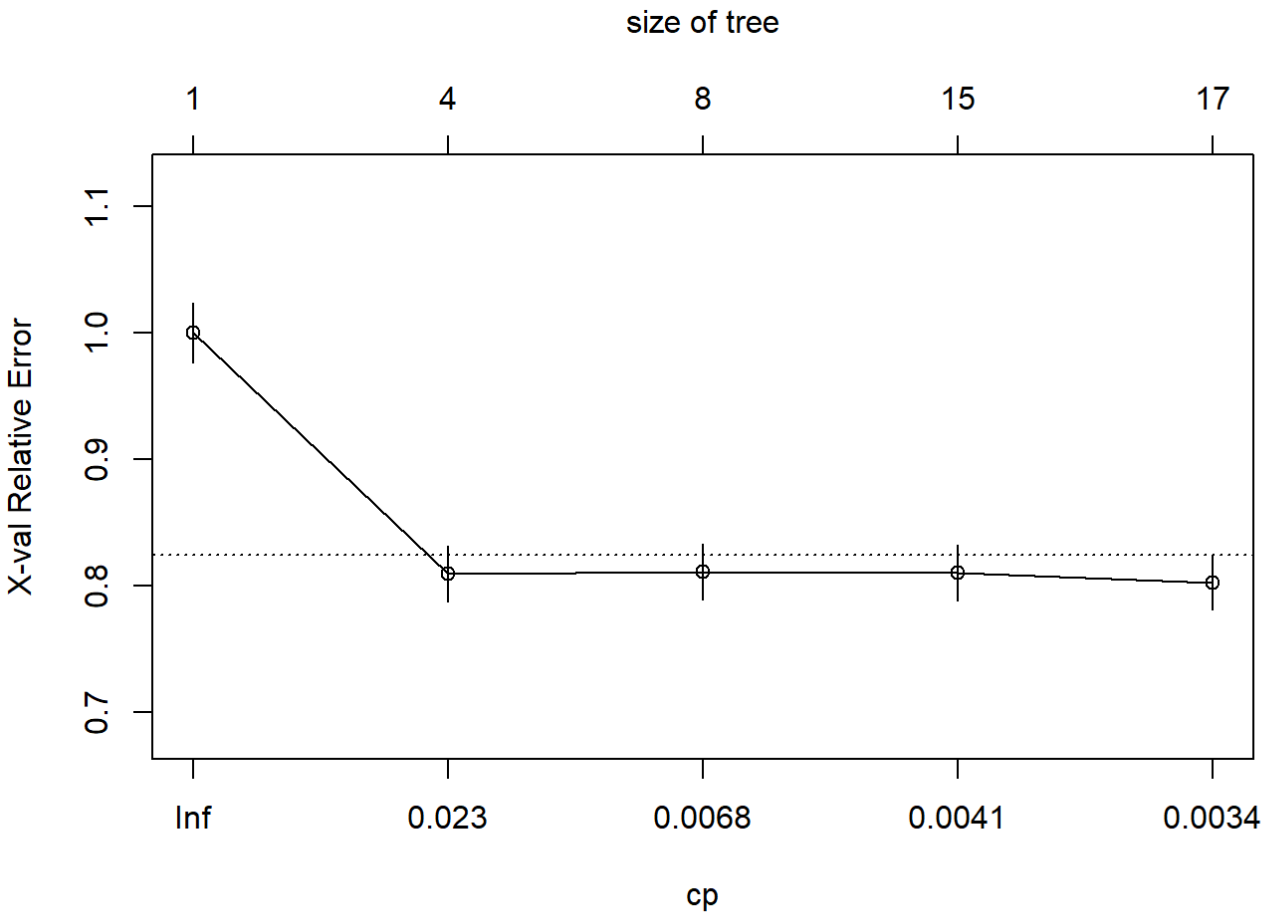
```
ar<-rpart(formula, train, method = 'class', parms = list(
  split = "information"),
  control = rpart.control(cp = 0.003,maxdepth = 7))
```

Revisamos de nuevo la complejidad.

```
printcp(ar)
```

```
##
## Classification tree:
## rpart(formula = formula, data = train, method = "class", parms = list(split = "inform
ation"),
##     control = rpart.control(cp = 0.003, maxdepth = 7))
##
## Variables actually used in tree construction:
##  [1] Contract            InternetService     MonthlyCharges_DISC
##  [4] OnlineBackup        OnlineSecurity      PaperlessBilling
##  [7] PaymentMethod       StreamingMovies     TechSupport
## [10] tenure_DISC
##
## Root node error: 1321/4924 = 0.26828
##
## n= 4924
##
##         CP nsplit rel error  xerror     xstd
## 1 0.051098      0   1.00000 1.00000 0.023535
## 2 0.010220      3   0.80924 0.80924 0.021900
## 3 0.004542      7   0.76609 0.81075 0.021915
## 4 0.003785     14   0.73126 0.80999 0.021907
## 5 0.003000     16   0.72369 0.80242 0.021833
```

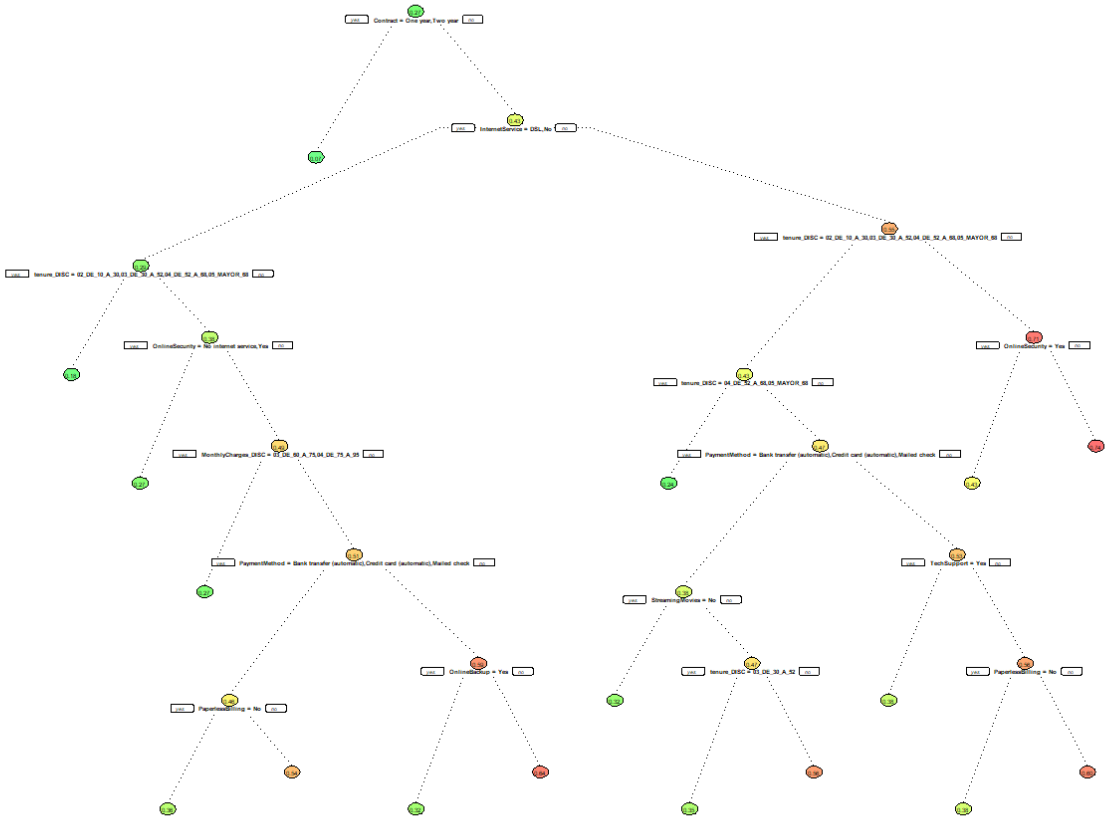
```
plotcp(ar)
```



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)
```



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 0.07 with cover 45% when
##     Contract is One year or Two year
##
## TARGET1 is 0.18 with cover 11% when
##     Contract is Month-to-month
##     tenure_DISC is 02_DE_10_A_30 or 03_DE_30_A_52 or 04_DE_52_A_68 or 05_MAYOR_68
##     InternetService is DSL or No
##
## TARGET1 is 0.24 with cover 3% when
##     Contract is Month-to-month
##     tenure_DISC is 04_DE_52_A_68 or 05_MAYOR_68
##     InternetService is Fiber optic
##
## TARGET1 is 0.27 with cover 1% when
##     Contract is Month-to-month
##     tenure_DISC is 01_MENOR_10
##     InternetService is DSL or No
##     OnlineSecurity is No
##     MonthlyCharges_DISC is 03_DE_60_A_75 or 04_DE_75_A_95
##
## TARGET1 is 0.27 with cover 7% when
##     Contract is Month-to-month
##     tenure_DISC is 01_MENOR_10
##     InternetService is DSL or No
##     OnlineSecurity is No internet service or Yes
##
## TARGET1 is 0.32 with cover 4% when
##     Contract is Month-to-month
##     tenure_DISC is 02_DE_10_A_30 or 03_DE_30_A_52
##     InternetService is Fiber optic
##     PaymentMethod is Bank transfer (automatic) or Credit card (automatic) or Mailed c
heck
##     StreamingMovies is No
##
## TARGET1 is 0.32 with cover 0% when
##     Contract is Month-to-month
##     tenure_DISC is 01_MENOR_10
##     InternetService is DSL or No
##     PaymentMethod is Electronic check
##     OnlineSecurity is No
##     MonthlyCharges_DISC is 02_DE_20_A_60
##     OnlineBackup is Yes
##
## TARGET1 is 0.35 with cover 1% when
##     Contract is Month-to-month
```

```
##     tenure_DISC is 03_DE_30_A_52
##     InternetService is Fiber optic
##     PaymentMethod is Bank transfer (automatic) or Credit card (automatic) or Mailed c
heck
##     StreamingMovies is Yes
##
## TARGET1 is 0.36 with cover 2% when
##     Contract is Month-to-month
##     tenure_DISC is 01_MENOR_10
##     InternetService is DSL or No
##     PaymentMethod is Bank transfer (automatic) or Credit card (automatic) or Mailed c
heck
##     OnlineSecurity is No
##     MonthlyCharges_DISC is 02_DE_20_A_60
##     PaperlessBilling is No
##
## TARGET1 is 0.38 with cover 1% when
##     Contract is Month-to-month
##     tenure_DISC is 02_DE_10_A_30 or 03_DE_30_A_52
##     InternetService is Fiber optic
##     PaymentMethod is Electronic check
##     TechSupport is Yes
##
## TARGET1 is 0.38 with cover 1% when
##     Contract is Month-to-month
##     tenure_DISC is 02_DE_10_A_30 or 03_DE_30_A_52
##     InternetService is Fiber optic
##     PaymentMethod is Electronic check
##     PaperlessBilling is No
##     TechSupport is No
##
## TARGET1 is 0.43 with cover 1% when
##     Contract is Month-to-month
##     tenure_DISC is 01_MENOR_10
##     InternetService is Fiber optic
##     OnlineSecurity is Yes
##
## TARGET1 is 0.54 with cover 2% when
##     Contract is Month-to-month
##     tenure_DISC is 01_MENOR_10
##     InternetService is DSL or No
##     PaymentMethod is Bank transfer (automatic) or Credit card (automatic) or Mailed c
heck
##     OnlineSecurity is No
##     MonthlyCharges_DISC is 02_DE_20_A_60
##     PaperlessBilling is Yes
```

```
##
## TARGET1 is 0.56 with cover 2% when
##     Contract is Month-to-month
##     tenure_DISC is 02_DE_10_A_30
##     InternetService is Fiber optic
##     PaymentMethod is Bank transfer (automatic) or Credit card (automatic) or Mailed c
heck
##     StreamingMovies is Yes
##
## TARGET1 is 0.60 with cover 6% when
##     Contract is Month-to-month
##     tenure_DISC is 02_DE_10_A_30 or 03_DE_30_A_52
##     InternetService is Fiber optic
##     PaymentMethod is Electronic check
##     PaperlessBilling is Yes
##     TechSupport is No
##
## TARGET1 is 0.64 with cover 2% when
##     Contract is Month-to-month
##     tenure_DISC is 01_MENOR_10
##     InternetService is DSL or No
##     PaymentMethod is Electronic check
##     OnlineSecurity is No
##     MonthlyCharges_DISC is 02_DE_20_A_60
##     OnlineBackup is No
##
## TARGET1 is 0.74 with cover 11% when
##     Contract is Month-to-month
##     tenure_DISC is 01_MENOR_10
##     InternetService is Fiber optic
##     OnlineSecurity is No
```

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 0.9308318 0.06916817   2
## 2 0.6176471 0.38235294 118
## 3 0.9308318 0.06916817   2
## 4 0.6451613 0.35483871 234
## 5 0.9308318 0.06916817   2
## 6 0.4006969 0.59930314 239
```

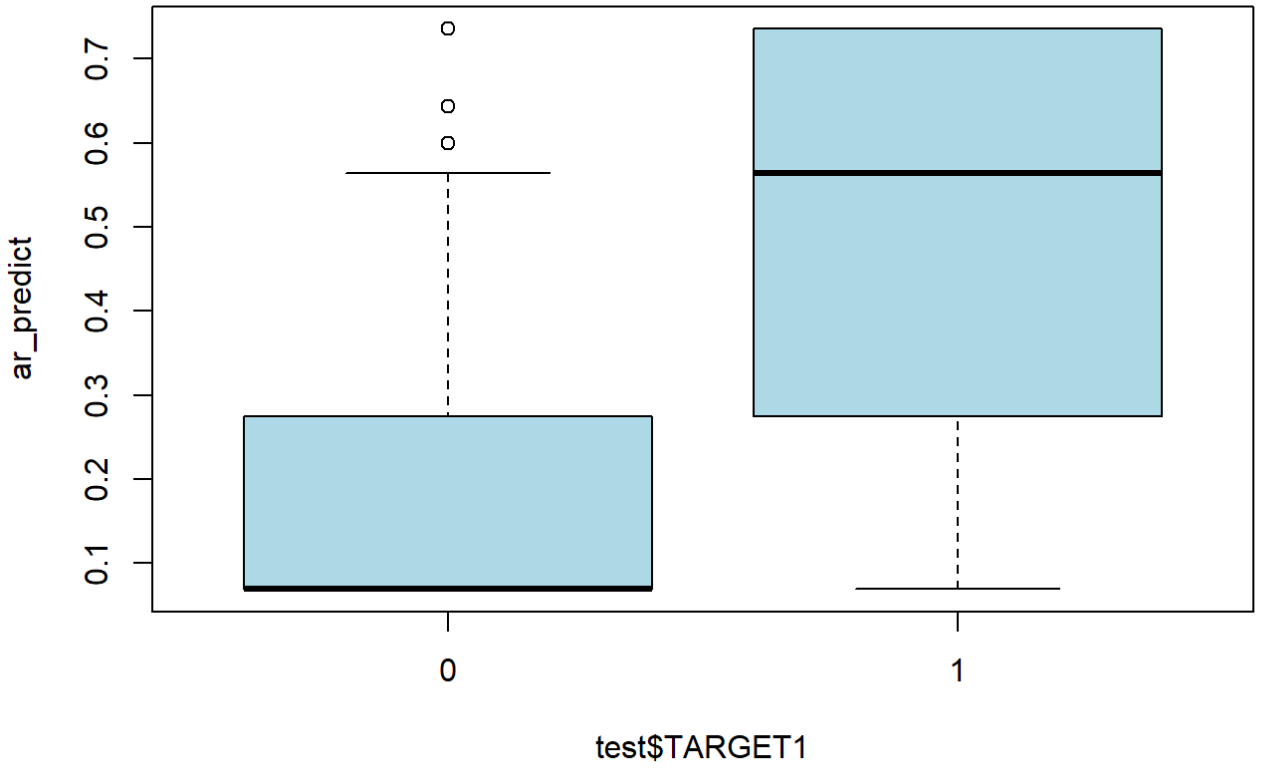
Vamos a calcular los scorings y evaluar el modelo.

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

```
##         1         2         3         4         5         6
## 0.06916817 0.38235294 0.06916817 0.35483871 0.06916817 0.59930314
```

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  0.05000   0.05000   0.05000  0.05000
## 2    0.10 65.03795  41.87446  88.86861 56.92577
## 3    0.15 65.03795  41.87446  88.86861 56.92577
## 4    0.20 72.43833  48.25027  83.02920 61.03286
## 5    0.25 73.71917  49.65909  79.74453 61.20448
## 6    0.30 77.94118  55.77191  73.17518 63.29913
## 7    0.35 79.64896  59.39968  68.61314 63.67485
## 8    0.40 80.36053  63.56275  57.29927 60.26871
## 9    0.45 79.93359  63.32623  54.19708 58.40708
## 10   0.50 79.93359  63.32623  54.19708 58.40708
## 11   0.55 80.12334  64.75973  51.64234 57.46193
## 12   0.60 78.74763  68.93939  33.21168 44.82759
## 13   0.65 78.60531  72.55814  28.46715 40.89122
## 14   0.70 78.60531  72.55814  28.46715 40.89122
## 15   0.75  0.75000   0.75000   0.75000  0.75000
## 16   0.80  0.80000   0.80000   0.80000  0.80000
## 17   0.85  0.85000   0.85000   0.85000  0.85000
## 18   0.90  0.90000   0.90000   0.90000  0.90000
## 19   0.95  0.95000   0.95000   0.95000  0.95000
```

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)
```

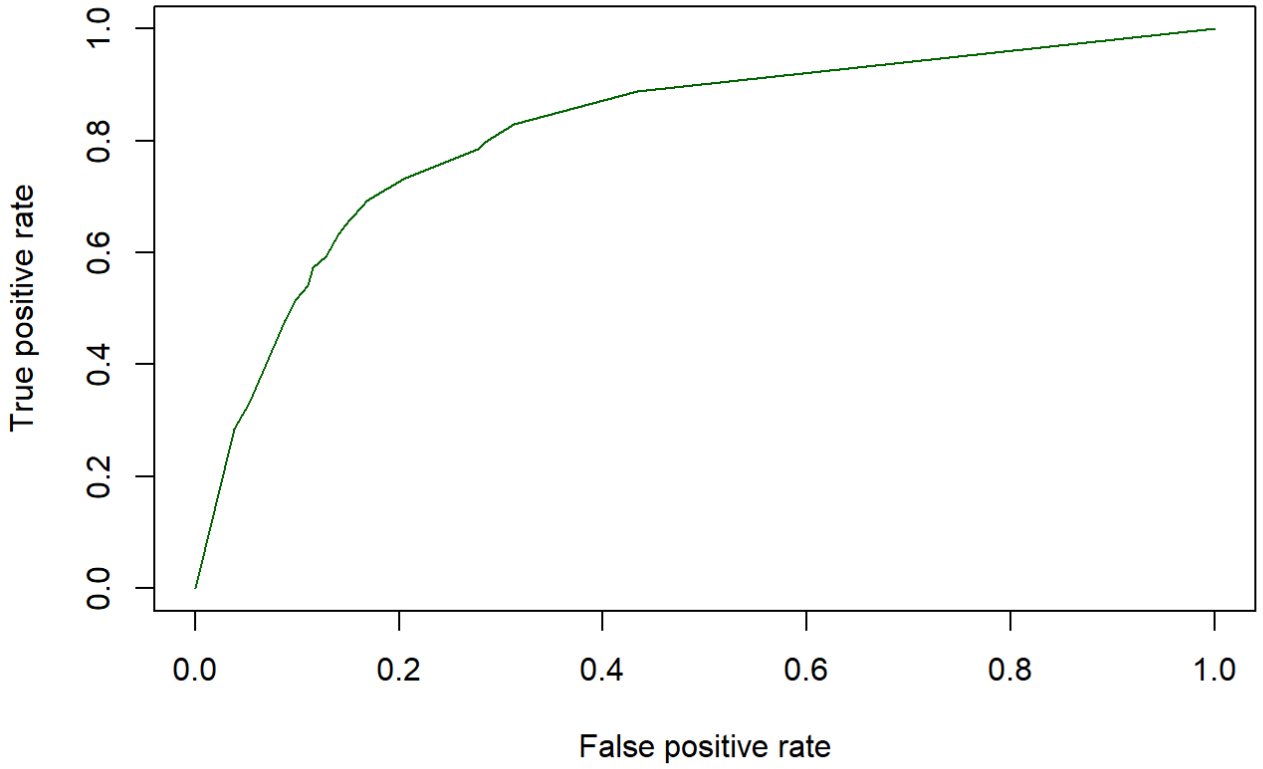
```
##
## real FALSE TRUE
##    0  1303  257
##    1   172  376
```

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

```
##   umbral  acierto precision cobertura       F1
## 1   0.35 79.64896  59.39968  68.61314 63.67485
```

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   79.64896
## precision 59.39968
## cobertura 68.61314
## F1        63.67485
## AUC       82.00000
```

- 4.5.6 - Modelizamos con Random Forest

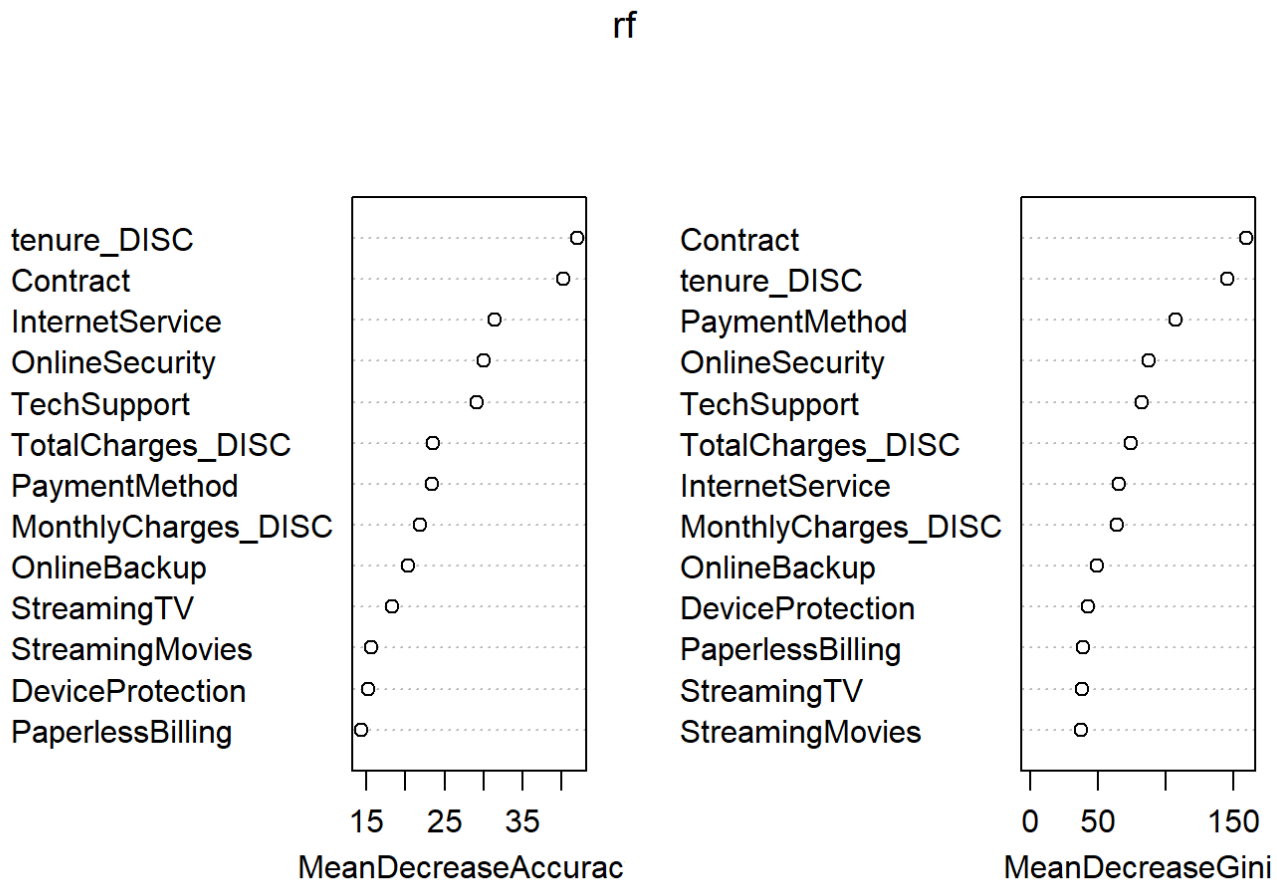
Creamos el modelo

```
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: 20.09%
## Confusion matrix:
##      0   1 class.error
## 0 3236 367   0.1018596
## 1  622 699   0.4708554
```

Visualizamos las variables más importantes.

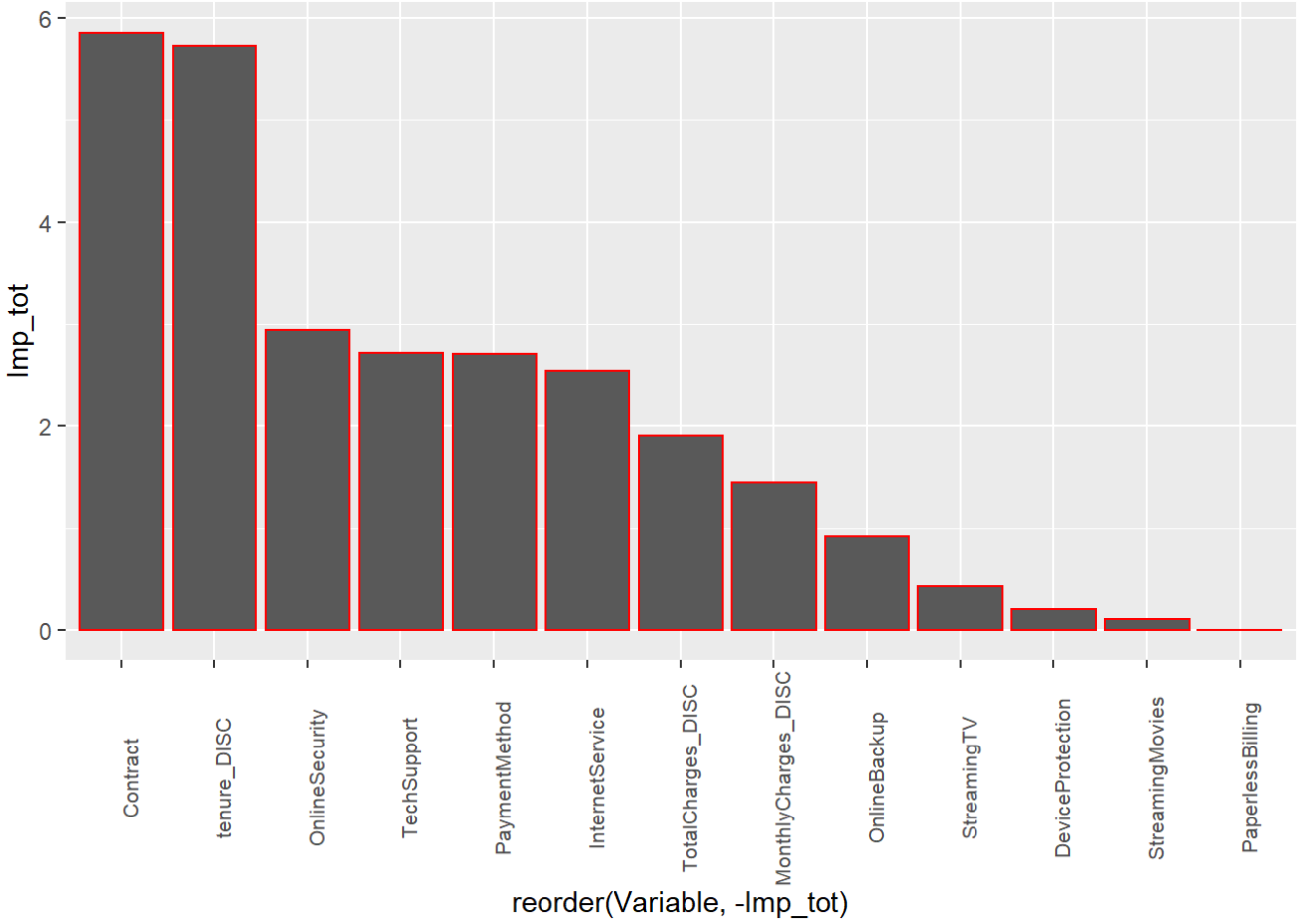
```
varImpPlot(rf)
```



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 = "ide
ntity", col='red') + theme(axis.text.x = element_text(angle = 90,size = 8))
```




```
importancia_norm
```

```
##                                Variable   Imp_tot MeanDecreaseAccuracy
## Contract                       Contract 5.8632148            1.6727295
## tenure_DISC                 tenure_DISC 5.7240946            1.8800395
## OnlineSecurity           OnlineSecurity 2.9393599            0.5507596
## TechSupport                 TechSupport 2.7134515            0.4524544
## PaymentMethod            PaymentMethod 2.7057872           -0.1817777
## InternetService         InternetService 2.5453420            0.7080895
## TotalCharges_DISC     TotalCharges_DISC 1.9051817           -0.1609506
## MonthlyCharges_DISC MonthlyCharges_DISC 1.4460784           -0.3534465
## OnlineBackup               OnlineBackup 0.9136369           -0.5213627
## StreamingTV                 StreamingTV 0.4353256           -0.7415748
## DeviceProtection       DeviceProtection 0.2048495           -1.0827744
## StreamingMovies         StreamingMovies 0.1089107           -1.0372497
## PaperlessBilling       PaperlessBilling 0.0000000           -1.1849360
##                     MeanDecreaseGini
## Contract                  2.07469826
## tenure_DISC               1.72826795
## OnlineSecurity            0.27281320
## TechSupport               0.14520992
## PaymentMethod             0.77177779
## InternetService          -0.27853466
## TotalCharges_DISC        -0.04965485
## MonthlyCharges_DISC      -0.31626227
## OnlineBackup             -0.68078749
## StreamingTV              -0.93888681
## DeviceProtection         -0.82816315
## StreamingMovies          -0.96962677
## PaperlessBilling         -0.93085111
```

La caída es bastante gradual, así que no hay corte claro. Podemos coger por ejemplo hasta MonthlyCharges_DISC incluido, que tiene una importancia total mayor de 1.

```
a_mantener <- importancia_norm %>%
  filter(Imp_tot > 1) %>%
  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: 2
##
##         OOB estimate of  error rate: 20.37%
## Confusion matrix:
##      0   1 class.error
## 0 3280 323  0.08964752
## 1  680 641  0.51476154
```

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

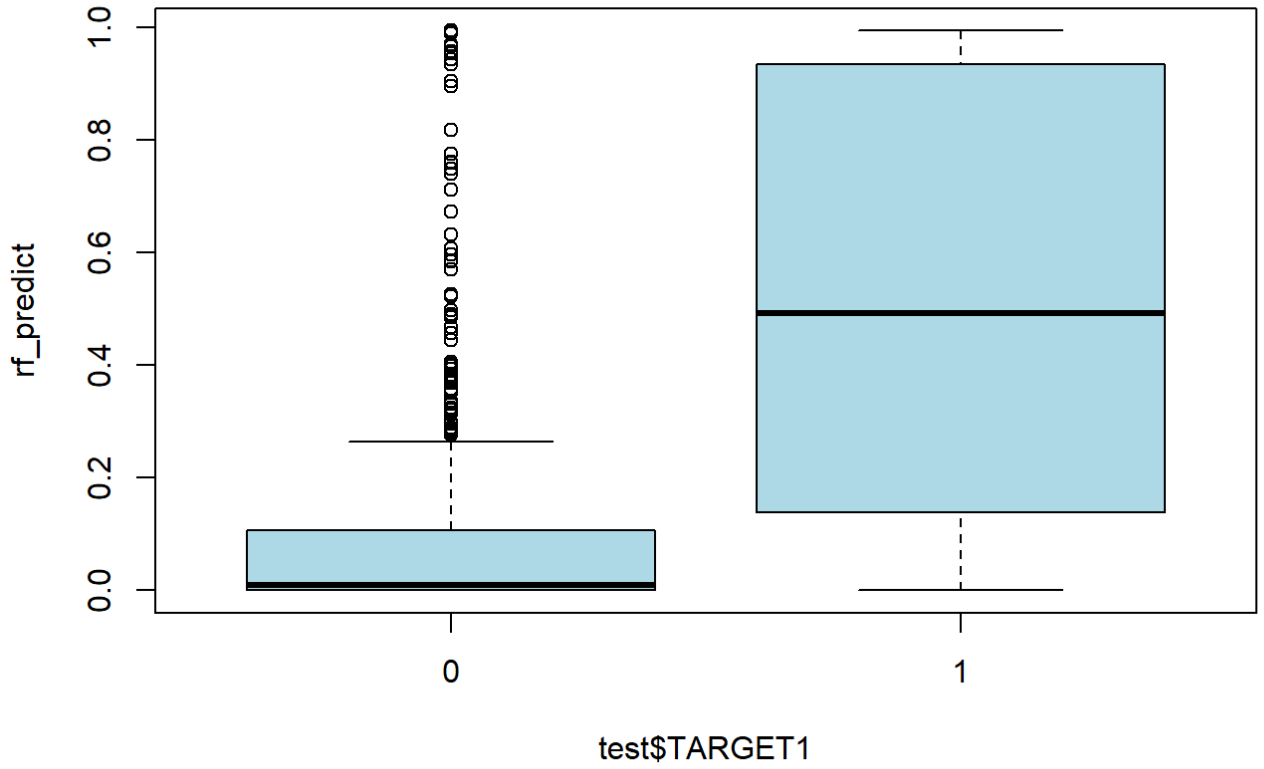
Vemos en la segunda columna de la matriz obtenida la clasificación del error.

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

```
##     1     2     3     4     5     6
## 0.002 0.276 0.000 0.360 0.000 0.632
```

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 71.15750  46.93878  83.94161 60.20942
## 2    0.10 75.52182  51.90931  79.37956 62.77056
## 3    0.15 77.75142  55.38881  74.08759 63.38798
## 4    0.20 78.88994  57.67511  70.62044 63.49467
## 5    0.25 79.60152  59.45513  67.70073 63.31058
## 6    0.30 79.93359  60.79447  64.23358 62.46673
## 7    0.35 80.07590  62.80000  57.29927 59.92366
## 8    0.40 80.21822  64.58797  52.91971 58.17452
## 9    0.45 80.17078  65.11628  51.09489 57.25971
## 10   0.50 80.21822  65.85956  49.63504 56.60770
## 11   0.55 80.02846  66.15776  47.44526 55.26036
## 12   0.60 80.12334  66.92913  46.53285 54.89774
## 13   0.65 80.12334  68.16901  44.16058 53.59911
## 14   0.70 79.93359  67.90831  43.24818 52.84281
## 15   0.75 79.50664  69.72789  37.40876 48.69359
## 16   0.80 79.45920  70.31802  36.31387 47.89410
## 17   0.85 79.03226  73.87387  29.92701 42.59740
## 18   0.90 78.41556  73.60406  26.45985 38.92617
## 19   0.95 78.13093  74.85714  23.90511 36.23790
```

Seleccionamos automáticamente el mejor umbral.

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

```
## [1] 0.2
```

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

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

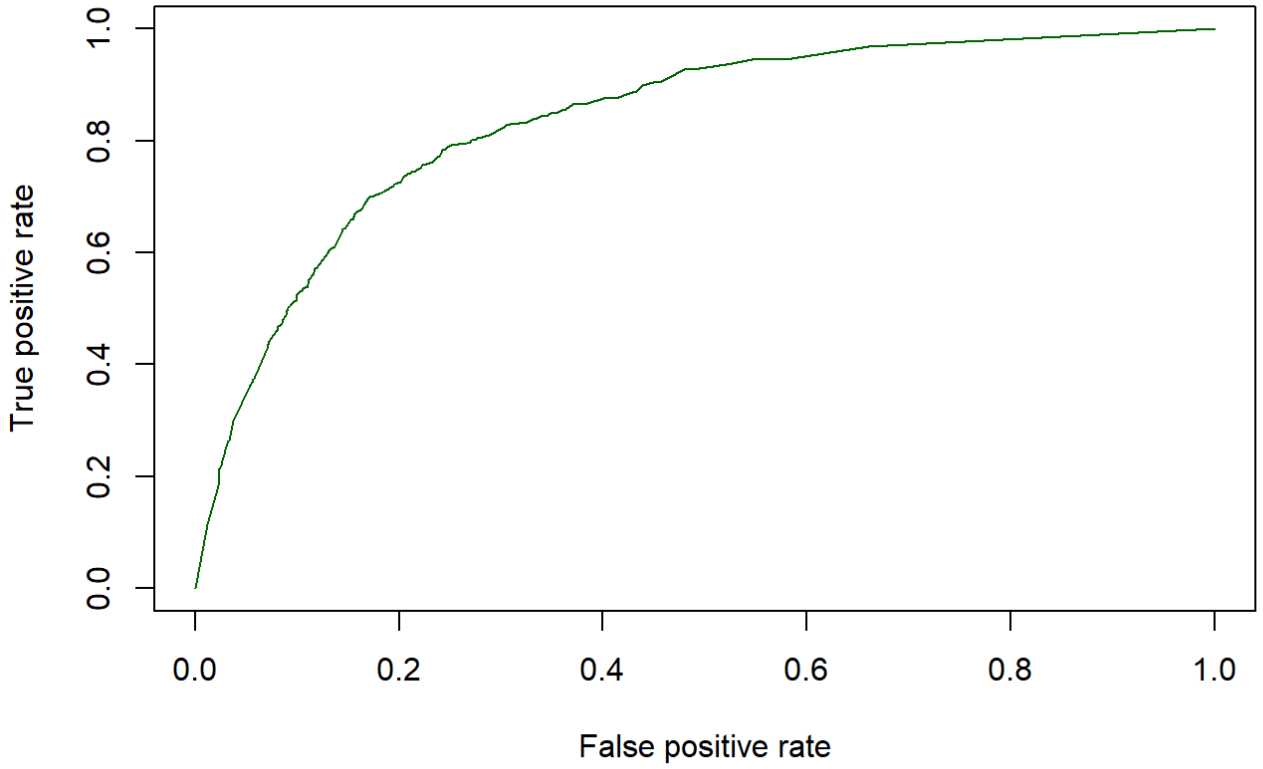
```
##
## real FALSE TRUE
##    0  1276  284
##    1   161  387
```

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

```
##   umbral  acierto precision cobertura       F1
## 1    0.2 78.88994  57.67511  70.62044 63.49467
```

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     0.20000
## acierto   78.88994
## precision 57.67511
## cobertura 70.62044
## F1        63.49467
## AUC       84.00000
```

- 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       0.20000
## acierto              79.88615       79.64896      78.88994
## precision            59.19881       59.39968      57.67511
## cobertura            72.81022       68.61314      70.62044
## F1                   65.30278       63.67485      63.49467
## AUC                  85.00000       82.00000      84.00000
```

Como vemos en los valores de AUC, la regresión logística 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(rl,df,type = 'response')
df %>% select(customerID,SCORING_CHURN) %>% arrange(desc(SCORING_CHURN)) %>% slice(1:30)
```

```
##     customerID SCORING_CHURN
##  1: 9305-CDSKC      0.769047
##  2: 4929-XIHVW      0.769047
##  3: 4445-ZJNMU      0.769047
##  4: 1875-QIVME      0.769047
##  5: 2472-OVKUP      0.769047
##  6: 4847-TAJYI      0.769047
##  7: 8098-LLAZX      0.769047
##  8: 8266-VBFQL      0.769047
##  9: 2034-GDRCN      0.769047
## 10: 4115-NZRKS      0.769047
## 11: 4572-DVCGN      0.769047
## 12: 5167-ZFFMM      0.769047
## 13: 5168-MSWXT      0.769047
## 14: 3811-VBYBZ      0.769047
## 15: 0306-JAELE      0.769047
## 16: 7218-HKQFK      0.769047
## 17: 9282-IZGQK      0.769047
## 18: 0195-IESCP      0.769047
## 19: 0970-ETWGE      0.769047
## 20: 5183-SNMJQ      0.769047
## 21: 4952-YSOGZ      0.769047
## 22: 1450-GALXR      0.769047
## 23: 6275-YDUVO      0.769047
## 24: 3643-AHCFP      0.769047
## 25: 3027-ZTDHO      0.769047
## 26: 9944-HKVVB      0.769047
## 27: 0781-LKXBR      0.769047
## 28: 4750-ZRXIU      0.769047
## 29: 3158-MOERK      0.769047
## 30: 8603-IJWDN      0.769047
##     customerID SCORING_CHURN
```

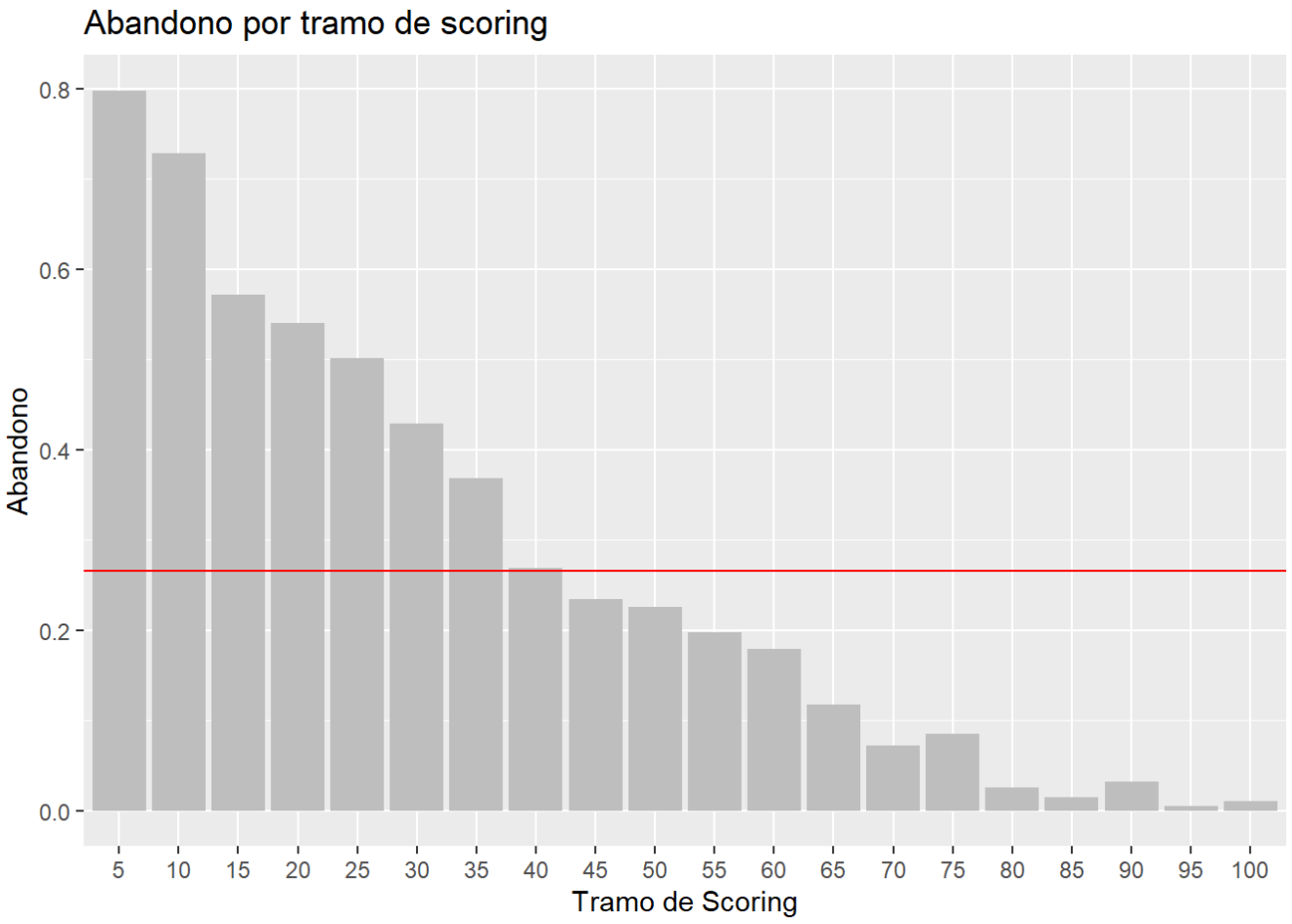
```
saveRDS(rl,'03_modelo_final.rds')
```

```
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 = cut_number(scoring, 20), Real
= real)
    levels(vis_df$Perc_Scoring) <- seq(from = 100,to = 5,by = -5)
    vis_gr <- vis_df %>% group_by(Perc_Scoring) %>% summarise(Tasa_Contr = mean(as.numer
ic(as.character(Real)))) %>% arrange(Perc_Scoring)
    vis_gr$Perc_Scoring <- factor(vis_gr$Perc_Scoring, levels = vis_gr$Perc_Scoring[orde
r(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 = 'Abandon
o')
}
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.

Según los datos que nos ha facilitado el departamento de marketing el margen medio por cliente que no abandona la compañía de 100 €.

El coste medio por acción comercial (cliente contactado por venta telefónica) = 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):

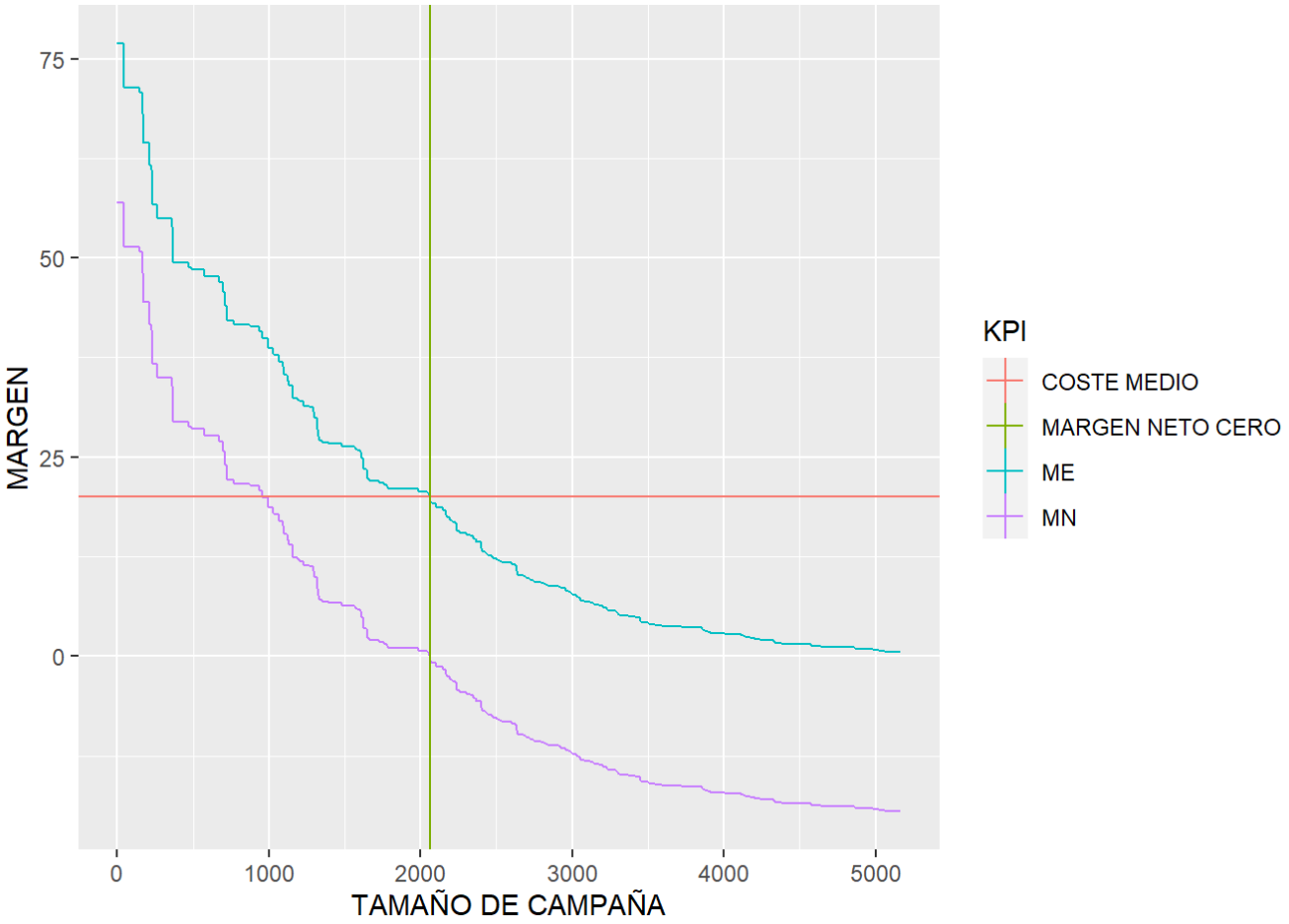
```
mar_medio <- 100
coste_medio <- 20
df_campaña <- df %>%
  filter(TARGET1==0) %>%
  mutate(
    ME = SCORING_CHURN * mar_medio,
    MN = ME - coste_medio) %>%
  arrange(desc(MN)) %>%
  mutate(INDICE = 1:nrow(.)) %>%
  select(customerID,INDICE,ME,MN)
head(df_campaña,50)
```

```
##     customerID INDICE      ME      MN
##  1: 4929-XIHVW      1 76.9047 56.9047
##  2: 4445-ZJNMU      2 76.9047 56.9047
##  3: 4847-TAJYI      3 76.9047 56.9047
##  4: 8266-VBFQL      4 76.9047 56.9047
##  5: 4115-NZRKS      5 76.9047 56.9047
##  6: 5168-MSWXT      6 76.9047 56.9047
##  7: 5183-SNMJQ      7 76.9047 56.9047
##  8: 7426-RHZGU      8 76.9047 56.9047
##  9: 7363-QTBIW      9 76.9047 56.9047
## 10: 8714-CTZJW     10 76.9047 56.9047
## 11: 2081-VEYEH     11 76.9047 56.9047
## 12: 3318-NMQXL     12 76.9047 56.9047
## 13: 2018-QKYGT     13 76.9047 56.9047
## 14: 1960-UYCNN     14 76.9047 56.9047
## 15: 8087-LGYHQ     15 76.9047 56.9047
## 16: 1393-IMKZG     16 76.9047 56.9047
## 17: 2754-SDJRD     17 76.9047 56.9047
## 18: 4813-HQMGZ     18 76.9047 56.9047
## 19: 2789-HQBOU     19 76.9047 56.9047
## 20: 7994-UYIVZ     20 76.9047 56.9047
## 21: 8622-ZLFKO     21 76.9047 56.9047
## 22: 0722-SVSFK     22 76.9047 56.9047
## 23: 5150-ITWWB     23 76.9047 56.9047
## 24: 4132-KALRO     24 76.9047 56.9047
## 25: 2545-EBUPK     25 76.9047 56.9047
## 26: 4633-MKHYU     26 76.9047 56.9047
## 27: 8161-QYMTT     27 76.9047 56.9047
## 28: 2688-BHGOG     28 76.9047 56.9047
## 29: 0187-QSXOE     29 76.9047 56.9047
## 30: 9741-YLNTD     30 76.9047 56.9047
## 31: 4395-PZMSN     31 76.9047 56.9047
## 32: 6350-XFYGW     32 76.9047 56.9047
## 33: 8118-TJAFG     33 76.9047 56.9047
## 34: 7577-SWIFR     34 76.9047 56.9047
## 35: 1941-HOSAM     35 76.9047 56.9047
## 36: 4273-MBHYA     36 76.9047 56.9047
## 37: 2657-VPXTA     37 76.9047 56.9047
## 38: 8132-YPVBX     38 76.9047 56.9047
## 39: 1628-BIZYP     39 76.9047 56.9047
## 40: 8035-PWSEV     40 76.9047 56.9047
## 41: 2959-FENLU     41 76.9047 56.9047
## 42: 9547-ITEFG     42 76.9047 56.9047
## 43: 4482-EWFMI     43 71.3662 51.3662
## 44: 6496-JDSSB     44 71.3662 51.3662
```

```
## 45: 2799-ARNLO     45 71.3662 51.3662
## 46: 0021-IKXGC     46 71.3662 51.3662
## 47: 5605-IYGFG     47 71.3662 51.3662
## 48: 1452-VOQCH     48 71.3662 51.3662
## 49: 4234-XTNEA     49 71.3662 51.3662
## 50: 8270-RKSAP     50 71.3662 51.3662
##     customerID INDICE      ME      MN
```

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:  2061"
```

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)
```

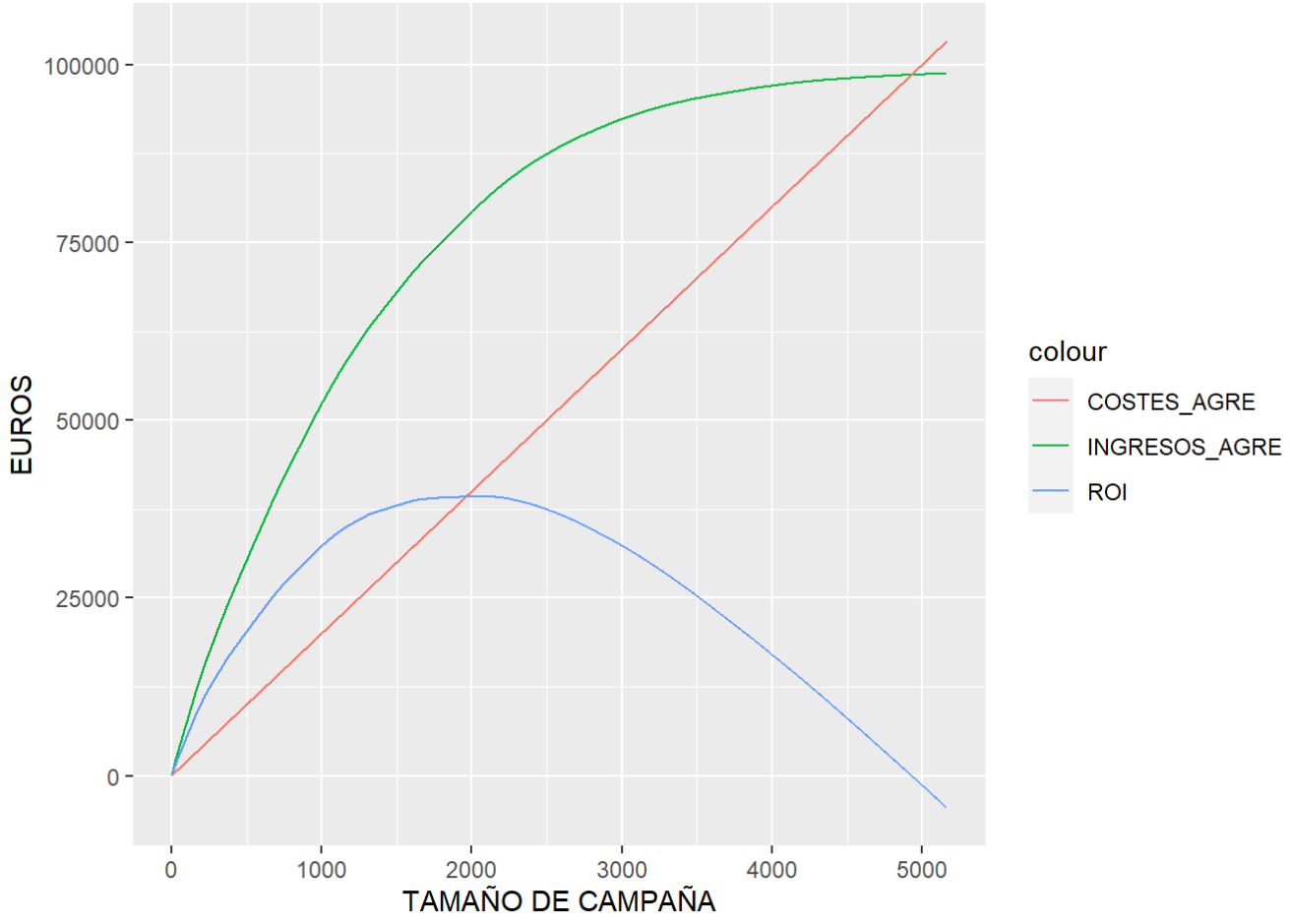
```
##     customerID INDICE      ME      MN INGRESOS_AGRE COSTES_AGRE       ROI
##  1: 4929-XIHVW      1 76.9047 56.9047       76.9047          20   56.9047
##  2: 4445-ZJNMU      2 76.9047 56.9047      153.8094          40  113.8094
##  3: 4847-TAJYI      3 76.9047 56.9047      230.7141          60  170.7141
##  4: 8266-VBFQL      4 76.9047 56.9047      307.6188          80  227.6188
##  5: 4115-NZRKS      5 76.9047 56.9047      384.5235         100  284.5235
##  6: 5168-MSWXT      6 76.9047 56.9047      461.4282         120  341.4282
##  7: 5183-SNMJQ      7 76.9047 56.9047      538.3329         140  398.3329
##  8: 7426-RHZGU      8 76.9047 56.9047      615.2376         160  455.2376
##  9: 7363-QTBIW      9 76.9047 56.9047      692.1423         180  512.1423
## 10: 8714-CTZJW     10 76.9047 56.9047      769.0470         200  569.0470
## 11: 2081-VEYEH     11 76.9047 56.9047      845.9517         220  625.9517
## 12: 3318-NMQXL     12 76.9047 56.9047      922.8564         240  682.8564
## 13: 2018-QKYGT     13 76.9047 56.9047      999.7611         260  739.7611
## 14: 1960-UYCNN     14 76.9047 56.9047     1076.6658         280  796.6658
## 15: 8087-LGYHQ     15 76.9047 56.9047     1153.5705         300  853.5705
## 16: 1393-IMKZG     16 76.9047 56.9047     1230.4752         320  910.4752
## 17: 2754-SDJRD     17 76.9047 56.9047     1307.3799         340  967.3799
## 18: 4813-HQMGZ     18 76.9047 56.9047     1384.2846         360 1024.2846
## 19: 2789-HQBOU     19 76.9047 56.9047     1461.1893         380 1081.1893
## 20: 7994-UYIVZ     20 76.9047 56.9047     1538.0940         400 1138.0940
## 21: 8622-ZLFKO     21 76.9047 56.9047     1614.9987         420 1194.9987
## 22: 0722-SVSFK     22 76.9047 56.9047     1691.9034         440 1251.9034
## 23: 5150-ITWWB     23 76.9047 56.9047     1768.8081         460 1308.8081
## 24: 4132-KALRO     24 76.9047 56.9047     1845.7128         480 1365.7128
## 25: 2545-EBUPK     25 76.9047 56.9047     1922.6175         500 1422.6175
## 26: 4633-MKHYU     26 76.9047 56.9047     1999.5222         520 1479.5222
## 27: 8161-QYMTT     27 76.9047 56.9047     2076.4269         540 1536.4269
## 28: 2688-BHGOG     28 76.9047 56.9047     2153.3316         560 1593.3316
## 29: 0187-QSXOE     29 76.9047 56.9047     2230.2363         580 1650.2363
## 30: 9741-YLNTD     30 76.9047 56.9047     2307.1410         600 1707.1410
## 31: 4395-PZMSN     31 76.9047 56.9047     2384.0457         620 1764.0457
## 32: 6350-XFYGW     32 76.9047 56.9047     2460.9504         640 1820.9504
## 33: 8118-TJAFG     33 76.9047 56.9047     2537.8551         660 1877.8551
## 34: 7577-SWIFR     34 76.9047 56.9047     2614.7598         680 1934.7598
## 35: 1941-HOSAM     35 76.9047 56.9047     2691.6645         700 1991.6645
## 36: 4273-MBHYA     36 76.9047 56.9047     2768.5692         720 2048.5692
## 37: 2657-VPXTA     37 76.9047 56.9047     2845.4739         740 2105.4739
## 38: 8132-YPVBX     38 76.9047 56.9047     2922.3786         760 2162.3786
## 39: 1628-BIZYP     39 76.9047 56.9047     2999.2833         780 2219.2833
## 40: 8035-PWSEV     40 76.9047 56.9047     3076.1880         800 2276.1880
## 41: 2959-FENLU     41 76.9047 56.9047     3153.0927         820 2333.0927
## 42: 9547-ITEFG     42 76.9047 56.9047     3229.9974         840 2389.9974
## 43: 4482-EWFMI     43 71.3662 51.3662     3301.3636         860 2441.3636
## 44: 6496-JDSSB     44 71.3662 51.3662     3372.7298         880 2492.7298
```

```
## 45: 2799-ARNLO     45 71.3662 51.3662     3444.0960          900 2544.0960
## 46: 0021-IKXGC     46 71.3662 51.3662     3515.4622          920 2595.4622
## 47: 5605-IYGFG     47 71.3662 51.3662     3586.8284          940 2646.8284
## 48: 1452-VOQCH     48 71.3662 51.3662     3658.1946          960 2698.1946
## 49: 4234-XTNEA     49 71.3662 51.3662     3729.5608          980 2749.5608
## 50: 8270-RKSAP     50 71.3662 51.3662     3800.9270         1000 2800.9270
##     customerID INDICE      ME      MN INGRESOS_AGRE COSTES_AGRE       ROI
```

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

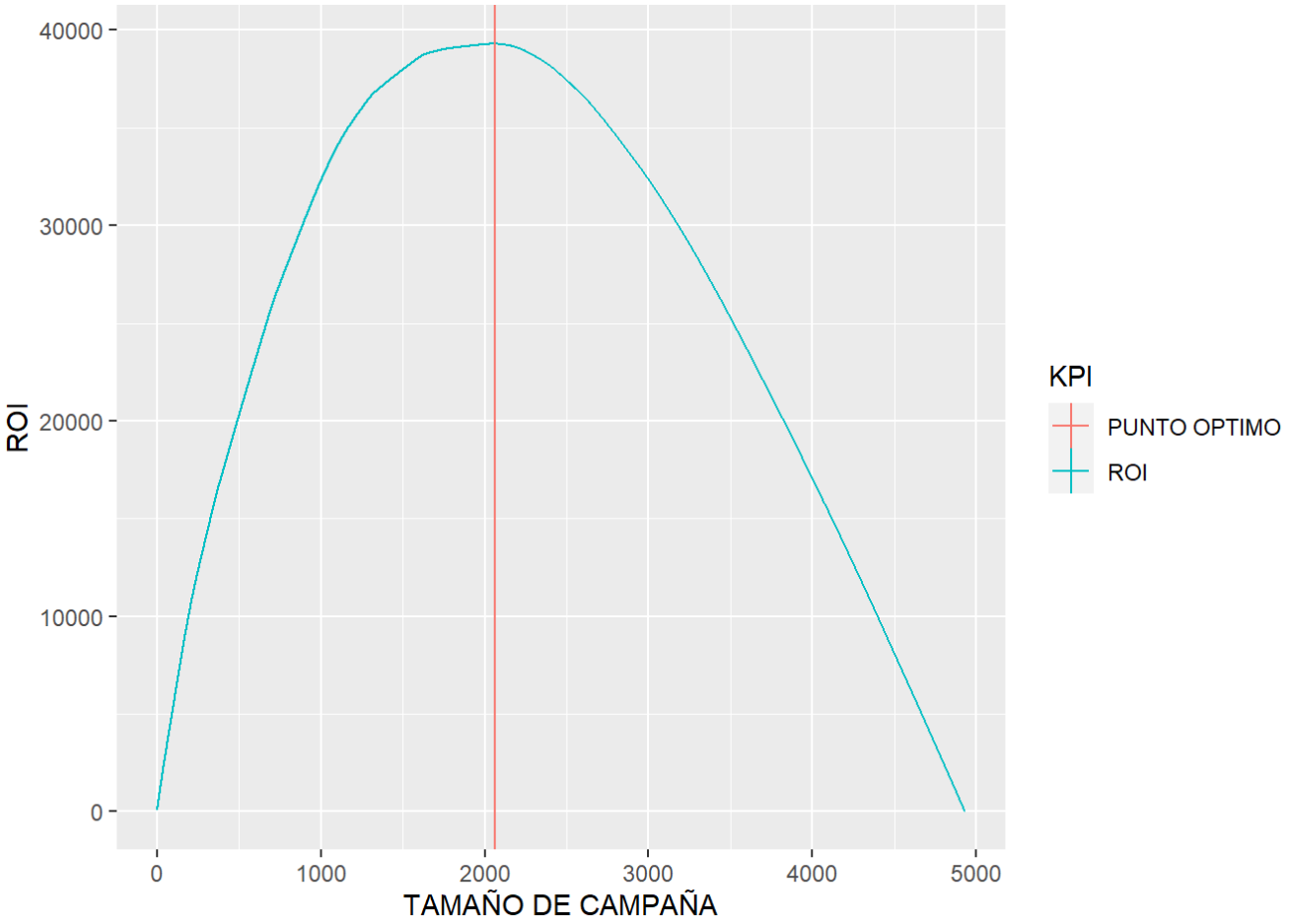
Visualizamos las curvas

```
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_campaña[whi
ch(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: 2061 clientes
## Con unos ingresos esperados de margen neto acumulado de: 80525 €
## Y unos costes agregados de: 41220 €
## Que van a generar un Retorno Neto de la Inversión de: 39305 €
```

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: 4929-XIHVW      1 76.9047 56.9047       76.9047          20  56.9047
## 2: 4445-ZJNMU      2 76.9047 56.9047      153.8094          40 113.8094
## 3: 4847-TAJYI      3 76.9047 56.9047      230.7141          60 170.7141
## 4: 8266-VBFQL      4 76.9047 56.9047      307.6188          80 227.6188
## 5: 4115-NZRKS      5 76.9047 56.9047      384.5235         100 284.5235
## 6: 5168-MSWXT      6 76.9047 56.9047      461.4282         120 341.4282
```

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: 2.061 clientes
2. CON LO QUE SE GENERARÁN UNOS **ingresos esperados** de margen neto acumulado de: 80.525 €
3. DISMINUYENDO LOS **costes agregados** HASTA: 41.220 €
4. Y PRODUCIENDO UN **Retorno Neto de la Inversión** de: 39.305 €

6 Resultados

Nuestra previsión es que usando este modelo se puede conseguir un ROI en la próxima campaña de 39.305 €.

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