MÁSTER BIG DATA Y DATA SCIENCE

DEPURACIÓN DE DATOS Y MODELIZACION PREDICTIVA

Análisis sobre la fuga de clientes en una compañía telefónica

MÓDULO: Minería de Datos y Modelización Predictiva

PROFESOR: Guillermo Villarino

AUTOR: Hassan Chafi Xavier

Preliminar

La preparación de las herramientas para el siguiente proyecto supone, en primer lugar, fijar el directorio de trabajo donde se encuentran los distintos archivos que se utilizarán y el paquete de funciones que se cargan a continuación. De igual forma, se cargan las librerías de R que se usarán.

```
# Fijar el directorio de trabajo
setwd("C:/Users/96has/Documents/NTIC/6. Documentación minería de Datos y Modelización Predictiva-20211212")

# Cargar las funciones
source("Funciones_R.R")

# Instalar/cargar las librerías de R
paquetes(c("questionr","psych","car","corrplot","ggplot2","gridExtra","kableExtra","dplyr","DMwR2",“caret”,“lmSupport”,“glmnet”,“epiDisplay”,“pROC”))
```

Parte I

Esta primera fase del proyecto consta de la lectura del archivo que contiene los datos a depurar y la inspección de estos datos para observar los tipos de variables, sus distribuciones y detectar presencia de outliers (datos atípicos) y missings (datos perdidos).

```
# Lectura del archivo de datos.
datos <- readRDS("C:/Users/96has/Documents/NTIC/6. Documentación minería de Datos y Modelización Predictiva-20211212/Tarea/FugaClientes_Training.RDS")
```

A continuación, se realiza una consulta para observar en detalle en qué tipo de variable están consideradas las variables del archivo de los datos. Se puede observar que, en general, los datos han sido leídos correctamente a excepción de la variable **ID** que es una variable que no aportará valor en la modelización predictiva de los datos y la mejor opción es convertirla en un tipo de variable character para posteriormente poder trabajar diferenciadamente con las variables continuas y las variables categóricas que sí aportarán valor en el modelado de los datos.

```
# Observamos los detalles de las variables
str(datos)
```

```
## 'data.frame':    6353 obs. of  21 variables:
##  $ ID              : Factor w/ 6353 levels "0002-ORFBO","0003-MKNFE",..: 4858 3587 2338 4997 587
9 5915 911 4316 5063 4102 ...
##  $ Genero          : Factor w/ 2 levels "Female","Male": 1 2 2 NA 1 1 2 1 NA 2 ...
##  $ Mayor65         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Conyuge         : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 1 2 1 ...
##  $ PersCargo       : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 2 ...
##  $ Antig.fc.edad   : num  1 34 2 45 NA 8 22 10 28 62 ...
##  $ Telf_serv       : Factor w/ 2 levels "No","Yes": 1 2 2 1 2 2 2 1 2 2 ...
##  $ VariasLineas    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 2 1 2 1 ...
##  $ Int_serv        : Factor w/ 3 levels "DSL","Fiber optic",..: 1 1 1 1 2 2 2 NA 2 1 ...
##  $ Seguridad       : Factor w/ 2 levels "No","Yes": 1 2 2 2 1 1 1 2 1 2 ...
##  $ CopiaSeguridad  : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 2 1 1 2 ...
##  $ Antivirus_disp  : Factor w/ 2 levels "No","Yes": 1 2 1 2 1 2 1 1 2 1 ...
##  $ Soporte_tecnico: Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 1 1 2 1 ...
##  $ TV_streaming    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 2 1 2 1 ...
##  $ Peliculas       : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 2 1 ...
##  $ Contrato        : Factor w/ 3 levels "Month-to-month",..: 1 2 1 2 1 1 1 1 1 NA ...
##  $ Fact_sinPapel   : Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 2 1 2 1 ...
##  $ MetodoPago      : Factor w/ 4 levels "Bank transfer (automatic)",..: NA 4 4 NA 3 NA 2 4 3 NA
...
##  $ FacturaMes      : num  NA NA 53.9 42.3 70.7 ...
```

```
##  $ FacturaTotal   : num  29.9 1889.5 108.2 1840.8 151.7 ...
##  $ Fuga           : Factor w/ 2 levels "0","1": 1 1 2 1 2 2 1 1 2 1 ...
```

Asimismo, es posible consultar en mayor detalle las variables conociendo los estadísticos básicos de cada una de ellas con los que podemos comenzar a sospechar de la posible presencia de valores extraños y observar si existen valores perdidos en alguna variable. En este caso, podemos afirmar ya que los datos a modelar tienen valores perdidos puesto que existen varias variables que confirman la presencia de estos valores NA´s. En cuanto a la presencia de valores extraños, puede sospecharse de este hecho observando para la variable **FacturaTotal** que sus valores de media y mediana se encuentran muy distantes entre si y podría estar advirtiendo de la existencia de outliers que estén arrastrando a la media hacia la derecha y que provoca que la media tenga un valor tan superior a la mediana.

```
# Observamos a detalle los valores de las variables
summary(datos)
```

```
##           ID          Genero     Mayor65     Conyuge    PersCargo
##  0002-ORFBO:   1   Female:2963   0   :5045   No :3287   No :4439
##  0003-MKNFE:   1   Male  :3038   1   : 956   Yes:3066   Yes:1914
##  0004-TLHLJ:   1   NA's  : 352   NA's: 352
##  0011-IGKFF:   1
##  0013-EXCHZ:   1
##  0013-MHZWF:   1
##  (Other)   :6347
##  Antig.fc.edad   Telf_serv   VariasLineas        Int_serv    Seguridad
##  Min.   : 0.00   No  : 605   No :3687     DSL        :2122   No :4532
##  1st Qu.: 9.00   Yes :5656   Yes:2666     Fiber optic:2702   Yes:1821
##  Median :29.00   NA's:  92                No         :1339
##  Mean   :32.38                            NA's       : 190
##  3rd Qu.:55.00
##  Max.   :72.00
##  NA's   :394
##  CopiaSeguridad Antivirus_disp Soporte_tecnico TV_streaming Peliculas
##  No :4180       No :4180       No :4509        No :3936     No :3902
##  Yes:2173       Yes:2173       Yes:1844        Yes:2417     Yes:2451
##
##
##
##
##
##            Contrato    Fact_sinPapel                     MetodoPago
##  Month-to-month:3285   No  :2483     Bank transfer (automatic):1283
##  One year      :1261   Yes :3609     Credit card (automatic)  :1290
##  Two year      :1413   NA's: 261     Electronic check         :1972
##  NA's          : 394                 Mailed check             :1336
##                                      NA's                     : 472
##
##
##    FacturaMes      FacturaTotal    Fuga
##  Min.   : 18.25   Min.   :  18.8   0:4667
##  1st Qu.: 35.55   1st Qu.: 394.5   1:1686
##  Median : 70.30   Median :1384.2
##  Mean   : 64.68   Mean   :2268.7
##  3rd Qu.: 89.80   3rd Qu.:3781.5
##  Max.   :118.75   Max.   :8672.5
##  NA's   :394      NA's   :11
```

Las variables que están consideradas como variables de tipo numérico podrían pasar a ser consideradas de tipo factor en el caso que la cantidad de valores distintos que adopta la variable no supere los diez valores, se sigue este criterio con el objetivo de no consumir excesivamente números de parámetros en el posterior modelado de los datos. Las tres variables numéricas que tenenos no parecen estar próximas

a ser consideradas como variables factor ya que sus cantidades de valores distintos superan el criterio fijado.

```
# Observamos los valores distintos que tienen las variables numéricas del archivo de datos
sapply(Filter(is.numeric, datos), FUN = function(x) length(unique(x)))

## Antig.fc.edad    FacturaMes  FacturaTotal
##            74          1523          5924
```

Los errores que se han detectado hasta este punto y que se va a proceder a su corrección son:

- La variable **ID** que está considerada como un factor se convertirá en un tipo de variable character.
- De cara a trabajar con los mismos valores que la variable objetivo donde la afirmación de fuga toma los valores 1 y la negación de fuga toma los valores 0; en las variables **Conyuge**, **PersCargo**, **Telf_serv**, **VariasLineas**, **Seguridad**, **CopiaSeguridad**, **Antivirus_disp**, **Soporte_tecnico**, **TV_streaming**, **Peliculas** y **Fact_sinPapel** que son variables dicotómicas que adoptan los valores 'Yes'/'No' se representará los valores 'Yes' con un 1 y los valores 'No' con un 0

```
# Modificación del tipo de variable de ID
datos$ID <- as.character(datos$ID)

#Recodificación de los valores de las variables dicotómicas
datos$Conyuge <- car::recode(datos$Conyuge, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$PersCargo <- car::recode(datos$PersCargo, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$Telf_serv <- car::recode(datos$Telf_serv, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$VariasLineas <- car::recode(datos$VariasLineas, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$Seguridad <- car::recode(datos$Seguridad, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$CopiaSeguridad <- car::recode(datos$CopiaSeguridad, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$Antivirus_disp <- car::recode(datos$Antivirus_disp, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$Soporte_tecnico<-car::recode(datos$Soporte_tecnico, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$TV_streaming <- car::recode(datos$TV_streaming, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$Peliculas <- car::recode(datos$Peliculas, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datos$Fact_sinPapel <- car::recode(datos$Fact_sinPapel, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
```

Unificar valores dentro de las variables categóricas requeriría de la presencia de valores poco representados dentro de estas variables. Observando la frecuencia de aparición de los valores en las variables categóricas nos damos cuenta que tienen una representación superior al 20% por lo que no parece indicado unificar estos valores ya que sin duda constituyen una muestra representativa con capacidad de influencia. **(CONSULTAR DETALLES EN ANEXO)**

```
# Comprobación de la frecuencia de aparición de los valores de las variables categóricas
lapply(Filter(is.factor, datos), FUN = freq)
```

Con estas modificaciones realizadas sobre el conjunto de datos, ya se puede comenzar a tratar los valores atípicos (en el caso de que los hubiera) entendiendo por valores atípicos o outliers aquellos valores extremos que son muy distintos a los demás valores, que se alejan del centro y que suelen tener influencia sobre la media haciendo que ésta se aleje de la mediana. La variable **FacturaTotal** tiene una simetría positiva que podría indicar la presencia de estos valores atípicos, pero solo mediante un criterio que se seguirá posteriormente, podremos saber si existen estos valores en la variable.

```
# Distribución de las variables numéricas

psych::describe(Filter(is.numeric, input))

##               vars    n    mean      sd  median trimmed     mad   min     max
## Antig.fc.edad    1 5959   32.38   24.54   29.00   31.44   32.62  0.00   72.00
## FacturaMes       2 5959   64.68   30.01   70.30   64.87   35.58 18.25  118.75
## FacturaTotal     3 6342 2268.73 2258.84 1384.17 1954.72 1801.28 18.80 8672.45
##                 range  skew kurtosis    se
```

```
## Antig.fc.edad   72.00  0.24    -1.39  0.32
## FacturaMes     100.50 -0.22    -1.25  0.39
## FacturaTotal  8653.65  0.97    -0.22 28.36
```

El criterio que se va a seguir para determiner la presencia de outliers (criterio 3sd + criterio de 3 IQR en las variables con distribuciones simétricas y el criterio del MAD con valor 8 + criterio de 3 IQR en las variables con distribuciones asimétricas) no revela la existencia de estos valores como se puede ver a continuación donde se representa el porcentaje de estos valores atípicos sobre el total de valores por cada variable.

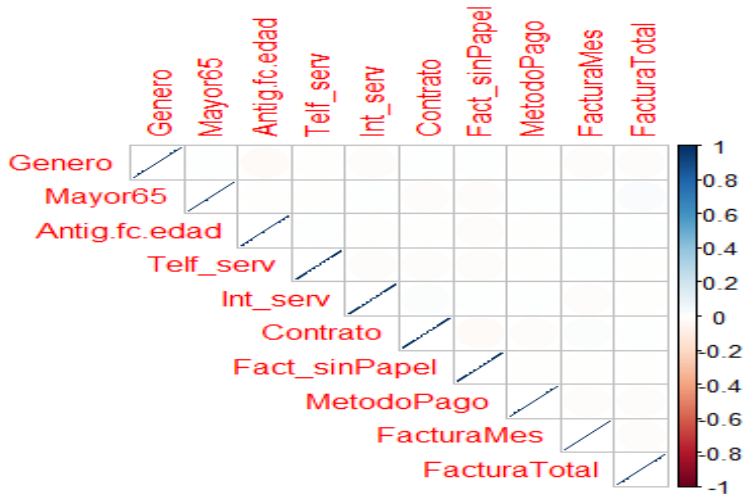
```
# Tabla con porcentaje de outliers por variable
outliersV <- data.frame(sort(
  round(sapply(Filter(
    is.numeric, input), function(nOut) atipicosAmissing(
      nOut)[[2]]) / nrow(input) * 100, 3), decreasing = TRUE))
names(outliersV) <- "% Outliers por variable"
outliersV

##               % Outliers por variable
## Antig.fc.edad                       0
## FacturaMes                          0
## FacturaTotal                        0
```

Así pues, se puede comenzar a analizar y tratar los valores perdidos o missings en las variables. Para comenzar con ello, trataremos primero de descubrir la presencia de patrones de coexistencia entre las variables. La presencia de un patron de coexistencia de missings entre dos variables desvelaría que los registros que tienen valores perdidos en una variable lo tienden a tener también en otra variable para la que existe dicho patron de coexistencia de missings.

```
# Diagrama de patrones de coexistencia de missings

corrplot(cor(is.na(input[colnames(input)[colSums(is.na(input)) > 0]])),method = "ellipse",type = "
upper")
```



A continuación se puede realizar una consulta donde podemos ver el porcentaje de valores perdidos que tienen cada una de las variables sobre el total de valores que tiene cada una de esas variables. En nuestro caso, las variables no parecen que tengan una cantidad significativa de missings por lo que posteriormente se pasará a su imputación; siendo las variables cuantitativas imputadas con su media, mediana o aleatoriamente y las variables cualitativas siendo imputadas con su moda o aleatoriamente.

```r
# Variable que recoge los missings por cada variable

prop_missingsV <- apply(is.na(input), 2, mean)



# Tabla con porcentaje de missings por variable
missingsV <- data.frame(sort(prop_missingsV * 100, decreasing = TRUE))
names(missingsV) <- "% Missings por Variable"
missingsV
```

```
##                 % Missings por Variable
## MetodoPago                    7.4295608
## Antig.fc.edad                 6.2017944
## Contrato                      6.2017944
## FacturaMes                    6.2017944
## Genero                        5.5406894
## Mayor65                       5.5406894
## Fact_sinPapel                 4.1082953
## Int_serv                      2.9907130
## Telf_serv                     1.4481347
## FacturaTotal                  0.1731465
## Conyuge                       0.0000000
## PersCargo                     0.0000000
## VariasLineas                  0.0000000
## Seguridad                     0.0000000
## CopiaSeguridad                0.0000000
## Antivirus_disp                0.0000000
## Soporte_tecnico               0.0000000
## TV_streaming                  0.0000000
## Peliculas                     0.0000000
```

Llegado el momento de imputar los missings de las variables cuantitativas y cualitativas; se realizará con su media y su moda respectivamente. Se decide realizar de esta forma y no de manera aleatoria para no tener una gran dependencia del azar y no alterar las distribuciones de las variables. Sin embargo, imputar los valores de la manera en la que se procederá a continuación, no está exento de inconvenientes y el más relevante podría ocasionar una importante alteración de la varianza de las variables.

```r
# Imputación de las variables cuantitativas
input[ ,as.vector(which(sapply(input, class) == "numeric"))] <- sapply(Filter(is.numeric, input),
function(x) ImputacionCuant(x, "media"))

# Inputación de las variables cualitativas
input[ ,as.vector(which(sapply(input, class) == "factor"))] <- sapply(Filter(is.factor, input), fu
nction(x) ImputacionCuali(x, "moda"))
```

Se puede considerer que la fase de depuración de los datos se ha realizado adecuadamente y por ello, podemos observar los estadísticos básicos de las variables para comprobar que todos los cambios se han aplicado y realizar una inspección gráfica final. **(CONSULTAR DETALLES EN ANEXO)**

```r
# Observamos a detalle los valores de las variables

summary(input)

# Inspección gráfica final
par(mfrow = c(3,3))
lista_his <- dfplot_his(input)
gridExtra::marrangeGrob(lista_his, nrow=3, ncol=2)
```

Finalmente, se procede a guardar los datos depurados junto con la variable objetivo que había sido extraída para llevar a cabo el proceso de depuración únicamente con las variables predictoras.

```
# Se guardan los datos
saveRDS(cbind(varObj, input),"datosTelefoniaDep.RDS")
```

Parte II

Se comienza realizando la lectura de los datos depurados guardados anteriormente con los que se llevará a cabo las tareas de observar sus influencias sobre la variable objetivo y todo el proceso de modelado.

```
# Cargar el archivo de datos depurados
datos <- readRDS("C:/Users/96has/Documents/NTIC/6. Documentación minería de Datos y Modelización P
redictiva-20211212/Tarea/datosTelefoniaDep.RDS")
```

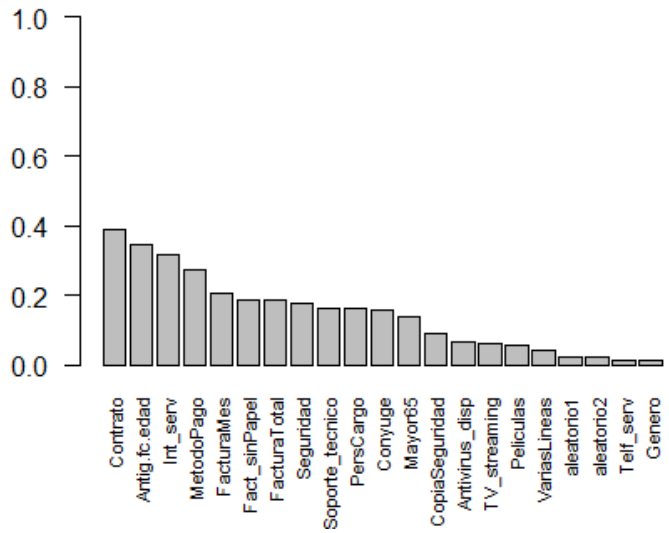
Se separa la variable objetivo de las variables predictoras para trabajar sobre estas últimas y se crean dos variables aleatorias que ejercerán una función de control que nos ayude a identificar qué variables tienen mayor capacidad de influencia sobre la respuesta, las variables predictoras con menos capacidad de discriminación que las variables aleatorias serán variables que no tendrán gran relevancia en la variabilidad de la variable objetivo.

```
# Separación de la variable objetivo de las variables predictoras
varObj <- datos$varObj
input <- datos[ , -1]

# Creación de las variables aleatorias de control
input$aleatorio1 <- runif(nrow(input))
input$aleatorio2 <- runif(nrow(input))
```

Mediante el gráfico V de Cramer se puede descubrir las relaciones marginales de las variables predictoras con la variable objetivo para tener una aproximación visual de cuáles serán las variables influyentes en los modelos de regresión que se ajustarán.

```
# Evaluación sobre la importancia de las variables predictoras sobre la variable objetivo
graficoVcramer(input, varObj)
```

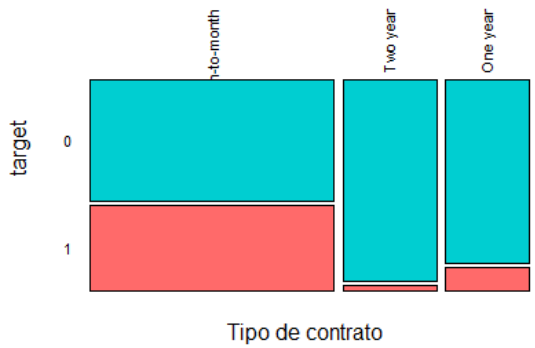


El gráfico V de Cramer revela un ranking de influencia de las variables predictoras sobre la variable objetivo. Las variables posicionadas por debajo de las dos variables aleatorias de control generadas no tendrán la necesaria influencia sobre la variable objetivo para explicar su variabilidad, estas variables son: **Telf_serv** y **Genero**. De este modo, parece que las variables que más influyen en la variabilidad de la variable objetivo binaria **Fuga** son:

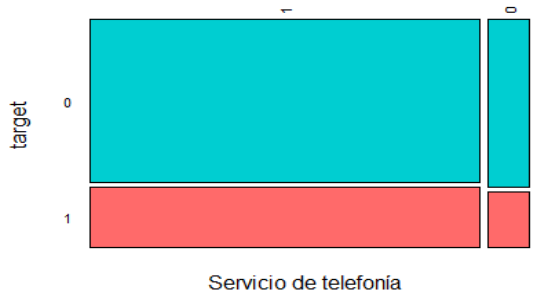
- Contrato
- Antig.fc.edad
- Int_serv
- MetodoPago
- FacturaMes

De este modo, se puede observar gráficamente las relaciones de las variables cualitativas sobre la variable objetivo:

```
#Analisis gráfico sobre la influencia de algunas variables categóricas sobre la variable objetivo
m1 <- mosaico_targetbinaria(input$Contrato, varObj, "Tipo de contrato") #Esta sí influye
```



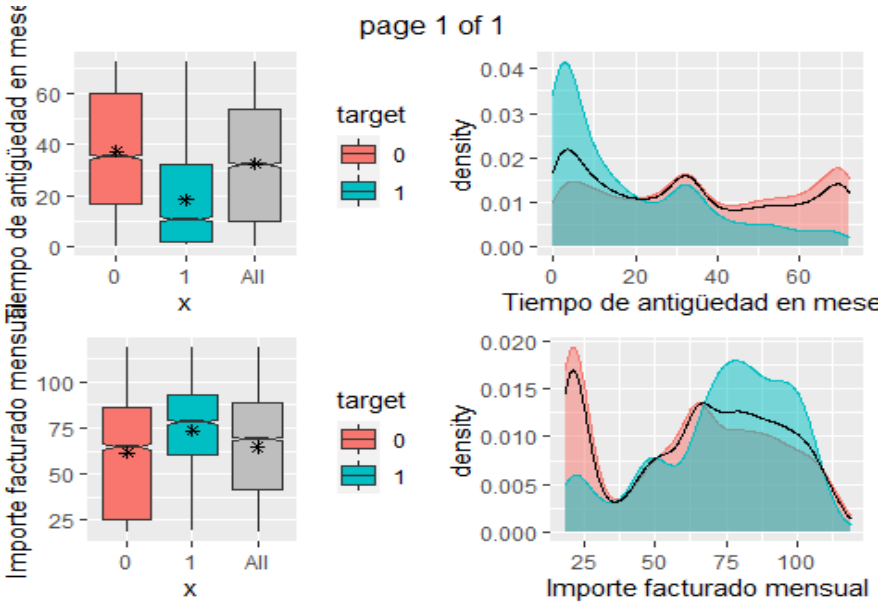

```
m2 <- mosaico_targetbinaria(input$Telf_serv, varObj, "Servicio de telefonía") #Esta no influye
```



Asimismo, se puede observar gráficamente las relaciones de las variables cuantitativas sobre la variable objetivo:

```
#Analisis gráfico sobre la influencia de algunas variables continuas sobre la variable objetivo
bx1 <- boxplot_targetbinaria(input$Antig.fc.edad, varObj,"Tiempo de antigüedad en meses") #Esta sí influye
h1 <- hist_targetbinaria(input$Antig.fc.edad, varObj, "Tiempo de antigüedad en meses") #Esta sí influye
bx2 <- boxplot_targetbinaria(input$FacturaMes, varObj, "Importe facturado mensual") #Esta sí influye
h2 <- hist_targetbinaria(input$FacturaMes, varObj, "Importe facturado mensual") #Esta sí influye

gridExtra::marrangeGrob(list(bx1, bx2, h1, h2),nrow = 2,ncol=2)
```



Se lleva a cabo el proceso de transformación de las variables cuantitativas con el objetivo de maximizar la influencia que tienen sobre la respuesta y quedarnos con las mejores transformaciones mediante un proceso automático. Asimismo, se guardan estas transformaciones para disponer de estos datos transformados, esto es un proceso necesario previo a la selección de variables clásicas que se llevará a cabo posteriormente en este proyecto.

```
# Se buscan las mejores transformaciones
input_obj <- cbind(input, Transf_Auto(Filter(is.numeric, input), varObj))

# Se guarda el dataset con las transformaciones
todo_obj <- data.frame(input_obj, varObj)
saveRDS(todo_obj, "transf_obj_Telefonia.RDS")
```

Parece ser que las tres variables cuantitativas (**Antig.fd.edad**, **FacturaMes** y **FacturaTotal**) han visto mejorada su influencia sobre la variable objetivo tras este paso de transformación como se puede comprobar en el gráfico V de Cramer. **(CONSULTAR DETALLES EN ANEXO)**

```
# Evaluación sobre la importancia de las variables predictoras sobre la variable objetivo
graficoVcramer(input_obj, varObj)
```

A continuación se ajustan los distintos modelos de regresión logística para predecir la fuga de clientes de la compañía telefónica. Para ello, se toma la partición training donde se ajusta el modelo y la partición test donde se prueba su capacidad.

```
# Posiciones de las variables
names(todo)
```

```
##  [1] "Genero"             "Mayor65"            "Conyuge"
##  [4] "PersCargo"          "Antig.fc.edad"      "Telf_serv"
##  [7] "VariasLineas"       "Int_serv"           "Seguridad"
## [10] "CopiaSeguridad"     "Antivirus_disp"     "Soporte_tecnico"
## [13] "TV_streaming"       "Peliculas"          "Contrato"
## [16] "Fact_sinPapel"      "MetodoPago"         "FacturaMes"
## [19] "FacturaTotal"       "aleatorio1"         "aleatorio2"
## [22] "sqrtxAntig.fc.edad" "raiz4FacturaMes"    "raiz4FacturaTotal"
## [25] "cuartaxaleatorio1"  "xaleatorio2"        "varObj"
```

```
# Se realiza la partición
set.seed(1234567000)

trainIndex <- createDataPartition(todo$varObj, p = 0.8, list = FALSE)

data_train <- todo[trainIndex,]
data_test <- todo[-trainIndex,]
```

El modelado manual de los datos no incluirá las variables transformadas, se comenzará con la creación de un primer modelo de referencia con el cual adoptaremos una estrategia de modelado hacia atrás eliminando progresivamente las variables que menos capacidad predictiva aporten hasta quedarnos finalmente con un modelo con las tres variables más representativas en cuanto a que aportación se refiere. **(CONSULTAR DETALLES EN ANEXO)**

```
# Primer modelo sin las variables transformadas
modeloInicial <- glm(varObj~., data = data_train[ ,c(1:21, 27)], family = binomial)
summary(modeloInicial)
```

Al obtener este primer modelo, se observa que algunos de los valores de la variable **MetodoPago** aparecen en el modelo pero sin aportar un codigo significativo al modelo, por lo que podría ser una acción inteligente replantearnos unificar valores dentro de la variable. Como se observa a continuación, la distribución de la variable revela que el evento de fuga del cliente de la compañía telefónica es relativamente inferior en los valores **Bank transfer (automatic)** y **Credit card (automatic)** que en los restantes valores; por ello, podríamos adoptar la estrategia de modificar la variable unificando los valores de manera que queden dos valores, uno referente a **Automatic Payments** y otra referente a **Not Automatic Payments**.

```
# Distribución de la variable

table(todo$MetodoPago, todo$varObj)
```

```
##
##                               0    1
##   Bank transfer (automatic) 1063  220
##   Credit card (automatic)   1104  186
##   Electronic check          1423 1021
##   Mailed check              1077  259
```

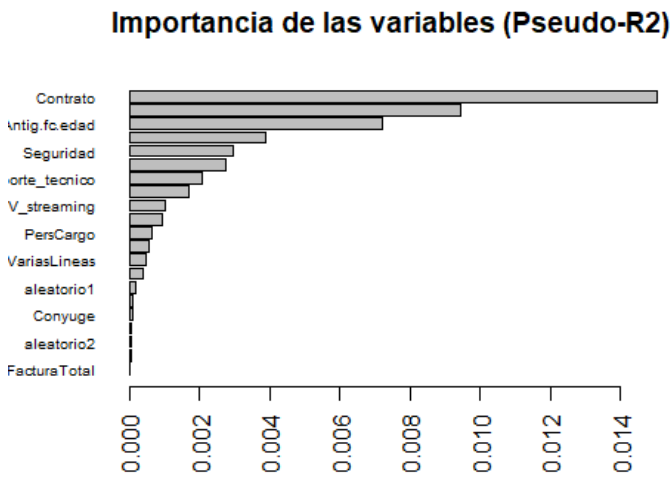
```
# Uno categorías
todo$MetodoPago <- car::recode(todo$MetodoPago, "c('Bank transfer (automatic)', 'Credit card (auto
matic)') = 'Automatic Payments'; c('Electronic check', 'Mailed check') = 'Not Automatic Payments'"
)
todo_obj$MetodoPago <- car::recode(todo_obj$MetodoPago, "c('Bank transfer (automatic)', 'Credit ca
rd (automatic)') = 'Automatic Payments'; c('Electronic check', 'Mailed check') = 'Not Automatic Pa
yments'")

# Actualizar la partición
data_train <- todo[trainIndex,]
data_test <- todo[-trainIndex,]

# Volver a ajustar el modelo inicial
modeloInicial <- glm(varObj~., data = data_train[ ,c(1:21, 27)], family=binomial)
summary(modeloInicial)
```

Los sucesivos modelos quedarán de la siguiente forma atendiendo a un ranking de importancia en cuanto a su aportación al pseudoR2:

```
# Importancia de las variables al pseudo R2
impVariablesLog(modeloInicial, "varObj")
```



- Segundo modelo sin las variables **FacturaTotal**, **aleatorio2**, **Genero**, **FacturaMes**, **Conyuge**, **Antivirus_disp** y **aleatorio1**
- Tercer modelo sin la variable **Mayor65**
- Cuarto modelo sin la variable **CopiaSeguridad**
- Quinto modelo sin la variable **VariasLineas**
- Sexto modelo sin la variable **Peliculas**
- Séptimo modelo sin la variable **PersCargo**
- Octavo modelo sin la variable **Telf_serv**
- Noveno modelo sin la variable **Soporte_tecnico**
- Décimo modelo sin la variable **TV_streaming**
- Undécimo modelo sin la variable **MetodoPago**

- Duodécimo modelo sin la variable **Seguridad**
- Decimotercer modelo sin la variable **Fact_sinPapel** (modelo con las tres variables más importantes)

De los trece modelos manuales que se han desarrollado es conveniente elegir el modelo que más nos parezca razonable en relación a la capacidad predictiva que presente y la complejidad del mismo medido por el número de parámetros. De todos ellos, los que presentan un número razonable de parámetros son los modelos 7, 8 y 9; observando sus valores en pseudoR2 (poca diferencia entre los 3) parece idóneo seleccionar el modelo 8 como el modelo que podríamos considerar como modelo final ganador puesto que es el que mayor capacidad predictiva tiene teniendo una cantidad aceptable de parámetros y será el modelo que posteriormente comparemos con otros modelos de selección clásica de variables y aleatoria.

```
# Resumen modelos manuales backward
modelos_Man<-list(modeloInicial,modelo2,modelo3,modelo4,
                  modelo5,modelo6,modelo7,modelo8,
                  modelo9,modelo10,modelo11, modelo12, modelo13)

# Saco los parametros de los modelos
modelos_Par<-c(modeloInicial$rank,modelo2$rank,modelo3$rank,modelo4$rank,
                  modelo5$rank,modelo6$rank,modelo7$rank,modelo8$rank,
                  modelo9$rank,modelo10$rank,modelo11$rank, modelo12$rank, modelo13$rank)

# Saco los pseudoR
psR_Man<-lapply(modelos_Man,pseudoR2,data_test,"varObj")

# Saco los VIF
vif_Man<-lapply(modelos_Man,car::vif)

# Para calcular los VIF máximos
vifMAx_Man<- c()
for (i in 1:length(vif_Man)){
  if (class(vif_Man[[i]])=='numeric'){
    vifMAx_Man[i]=max(vif_Man[[i]])
  } else{
    vifMAx_Man[i]=max(vif_Man[[i]][,3])
  }
}

tabla_modelos <- tibble(
  Modelo = c('modeloInicial','modelo2','modelo3','modelo4',
                  'modelo5','modelo6','modelo7','modelo8',
                  'modelo9','modelo10','modelo11', 'modelo12',
                  'modelo13'),
  Parametros=modelos_Par,
  pseudoR= unlist(psR_Man),
  VIF_max=vifMAx_Man)

tibble(tabla_modelos)

## # A tibble: 13 x 4
##    Modelo        Parametros pseudoR VIF_max
##    <chr>              <int>   <dbl>   <dbl>
##  1 modeloInicial         24   0.279    3.20
##  2 modelo2               17   0.280    1.34
##  3 modelo3               16   0.277    1.34
##  4 modelo4               15   0.276    1.31
##  5 modelo5               14   0.271    1.26
##  6 modelo6               13   0.269    1.25
##  7 modelo7               12   0.270    1.24
##  8 modelo8               11   0.271    1.24
```

```
##  9 modelo9               10   0.266    1.24
## 10 modelo10               9   0.260    1.21
## 11 modelo11               8   0.260    1.19
## 12 modelo12               7   0.254    1.17
## 13 modelo13               6   0.250    1.16
```

Creo una nueva variable que contendrá al modelo 8 para luego comparar con los modelos de selección clásica de variables y aleatoria en validación cruzada.

```
# Almaceno el modelo manual ganador

modeloManual <- modelo8
```

En este punto se aplicará la técnica de selección de variables clásica con variables originales con la ayuda de la función step() y para ello se creará un modelo mínimo llamado null, un modelo máximo sin las transformaciones de las variables llamado full, elegiremos la dirección del proceso entre forward, backward o both y aplicaremos la métrica AIC o BIC que tienen tendencia a modelos mas complejos y mas simples respectivamente.

```
# Creación de los modelos null y full

null <- glm(varObj~1, data = data_train, family = binomial) #Modelo minimo
full <- glm(varObj~., data = data_train[ ,c(1:21, 27)], family = binomial) #Modelo maximo sin las transformaciones

# Modelo que va desde null hasta full en ámbas direcciones con la métrica AIC

modeloStepAIC <- step(null, scope = list(lower = null, upper = full), direction = "both", trace = 0)

# Evaluación de la capacidad predictiva del modelo

psr_clas1 <- pseudoR2(modeloStepAIC, data_test, "varObj")

# Modelo que va desde full a null hacia atrás con la métrica AIC

modeloBackAIC <- step(full, scope = list(lower = null, upper = full), direction = "backward", trace = 0)


# Evaluación de la capacidad predictiva del modelo

psr_clas2 <- pseudoR2(modeloBackAIC, data_test, "varObj")

# Modelo que va desde null a full en ambas direcciones con la métrica BIC

modeloStepBIC <- step(null, scope = list(lower = null, upper = full), trace = 0,
                      direction = "both", k = log(nrow(data_train)))

# Evaluación de la capacidad predictiva del modelo

psr_clas3 <- pseudoR2(modeloStepBIC, data_test, "varObj")

# Modelo que va desde full a null hacia atrás con la métrica BIC

modeloBackBIC <- step(full, scope = list(lower = null, upper = full), trace = 0,
                      direction = "backward",k = log(nrow(data_train)))

# Evaluación de la capacidad predictiva del modelo

psr_clas4 <- pseudoR2(modeloBackBIC, data_test, "varObj")
```

Al igual que en el modelado manual se puede obtener una tabla a modo de resumen con el número de parámetros y la capacidad predictiva de cada uno de los modelos. A la vista de la información que se

observa en la tabla, parece que lo más sensato sería seleccionar el modeloStepBIC para la posterior comparación en validación cruzada con otros modelos puesto que es el que menos parámetros tiene y más capacidad predictiva tiene para ese numéro de parámetros.

```
# Resumen
param_Clas <- c(modeloStepAIC$rank, modeloBackAIC$rank,
modeloStepBIC$rank, modeloBackBIC$rank)

psR_Clas <- c(psr_clas1,psr_clas2,psr_clas3,psr_clas4)
tibble(Modelo = c('StepAIC','BackAIC', 'StepBIC', 'BackBIC'),
      parametros = param_Clas,
      pseudoR = psR_Clas)

## # A tibble: 4 x 3
##   Modelo  parametros pseudoR
##   <chr>        <int>   <dbl>
## 1 StepAIC         17   0.280
## 2 BackAIC         17   0.280
## 3 StepBIC         13   0.272
## 4 BackBIC         13   0.269
```

Creo una nueva variable que contendrá al modeloStepBIC para luego comparar con otros modelos en validación cruzada.

```
# Almaceno el modelo ganador

modeloEP <- modeloStepBIC
```

Ahora se va a permitir que la función step() pueda seleccionar interacciones entre las variables originales.

```
#Genero interacciones
formInt <- formulaInteracciones(todo_obj[,c(1:21, 27)], 22)
fullInt <- glm(formInt, data = data_train, family = binomial)

# Modelo con interacciones que va de null a full en ambas direcciones con la métrica AIC

modeloStepAIC_int <- step(null, scope = list(lower = null, upper = fullInt), direction = "both",
trace = 0)
```

```
# Evaluación de la capacidad predictiva del modelo

psr_clas1_int <- pseudoR2(modeloStepAIC_int, data_test, "varObj")

# Modelo con interacciones que va de null a full en ambas direcciones con la métrica BIC

modeloStepBIC_int <- step(null, scope=list(lower = null, upper = fullInt),
                  direction = "both",k = log(nrow(data_train)), trace = 0)

# Evaluación de la capacidad predictiva del modelo

psr_clas2_int <- pseudoR2(modeloStepBIC_int, data_test, "varObj")
```

En la tabla de resumen de estos dos modelos se observa que ambos modelos tienen más parámetros que los modelos que hemos venido seleccionando hasta el momento y la capacidad predictiva de ellos, lejos de mejorar, parece empeorar; por lo que no se va a seleccionar ningún modelo con interacciones.

```
# Resumen
param_Clas <- c(modeloStepAIC_int$rank, modeloStepBIC_int$rank)

psR_Clas <- c(psr_clas1_int,psr_clas2_int)
tibble(Modelo = c('StepAIC_int', 'StepBIC_int'),
```

```
      parametros = param_Clas,
      pseudoR = psR_Clas)
```

```
## # A tibble: 2 x 3
##   Modelo      parametros pseudoR
##   <chr>            <int>   <dbl>
## 1 StepAIC_int         36   0.275
## 2 StepBIC_int         15   0.278
```

Ahora se va a probar con las variables originales y sus transformaciones prestando atención a posibles casos de colinealidad entre las variables en caso que algún modelo presente buenas características y finalmente resulte en el modelo ganador.

```
# Se prueba con todas las transformaciones

fullT <- glm(varObj~., data = data_train, family = binomial)

# Modelo con transformadas que va de null a full en ambas direcciones con la métrica AIC

modeloStepAIC_trans <- step(null, scope = list(lower = null, upper = fullT), trace = 0, direction
= "both")

# Evaluación de la capacidad predictiva del modelo

psr_clas1_trans <- pseudoR2(modeloStepAIC_trans, data_test, "varObj")

# Modelo con transformadas que va de null a full en ambas direcciones con la métrica BIC

modeloStepBIC_trans <- step(null, scope = list(lower = null, upper = fullT),
                  trace = 0, direction = "both",k = log(nrow(data_train)))
# Evaluación de la capacidad predictiva del modelo

psr_clas2_trans <- pseudoR2(modeloStepBIC_trans, data_test, "varObj")
```

De los modelos resultantes, parece indicado seleccionar el modeloStepBIC_trans puesto que tiene un número de parámetros similar a los que hemos venido seleccionando anteriormente y la capacidad predictiva mejora notablemente.

```
# Resumen
param_Clas <- c(modeloStepAIC_trans$rank, modeloStepBIC_trans$rank)

psR_Clas <- c(psr_clas1_trans,psr_clas2_trans)
tibble(Modelo = c('StepAIC_trans', 'StepBIC_trans'),
      parametros = param_Clas,
      pseudoR = psR_Clas)
```

```
## # A tibble: 2 x 3
##   Modelo        parametros pseudoR
##   <chr>              <int>   <dbl>
## 1 StepAIC_trans         16   0.294
## 2 StepBIC_trans         12   0.290
```

Creo una nueva variable que contendrá al modeloStepBIC_trans para luego comparar con otros modelos en validación cruzada.

```
# Almaceno el modelo ganador

modeloT <- modeloStepBIC_trans
```

El conjunto más completo de efectos se obtendrá a continuación combinando todas las variables originales con sus transformaciones y las interacciones entre ellas.

```
# Transformaciones e interacciones

formIntT <- formulaInteracciones(todo_obj, 27)
fullIntT <- glm(formIntT, data = data_train, family = binomial)



# Modelo con transformaciones e interacciones que va de null a full en ambas direcciones con la mé
trica AIC

modeloStepAIC_transInt <- step(null, scope = list(lower = null, upper = fullIntT), trace = 0,
direction = "both")
```

```
# Evaluación de la capacidad predictiva del modelo

psr_clas1_intTrans <- pseudoR2(modeloStepAIC_transInt, data_test, "varObj")

# Modelo con transformaciones e interacciones que va de null a full en ambas direcciones con la mé
trica BIC

modeloStepBIC_transInt <- step(null, scope = list(lower = null, upper = fullIntT), trace = 0,
direction = "both",k = log(nrow(data_train)))
```

```
# Evaluación de la capacidad predictiva del modelo

psr_clas2_intTrans <- pseudoR2(modeloStepBIC_transInt, data_test, "varObj")
```

En la tabla de resumen de los modelos, el más indicado para ser seleccionado parece tener el mismo número de parámetros y la misma capacidad predictiva que el modelo seleccionado únicamente con las variables originales y sus transformadas; por lo que no se seleccionará ninguno de los modelos desarrollados en este punto.

```
# Resumen
param_Clas <- c(modeloStepAIC_transInt$rank, modeloStepBIC_transInt$rank)

psR_Clas <- c(psr_clas1_intTrans,psr_clas2_intTrans)
tibble(Modelo = c('StepAIC_transInt', 'StepBIC_transInt'),
      parametros = param_Clas,
      pseudoR = psR_Clas)

## # A tibble: 2 x 3
##   Modelo           parametros pseudoR
##   <chr>                 <int>   <dbl>
## 1 StepAIC_transInt         34   0.297
## 2 StepBIC_transInt         12   0.290
```

Finalizamos el modelado por selección de variables desarrollando unos modelos más de manera aleatoria sometiendo a la función step() a pruebas de robustez haciendo que trabaje con submuestras de datos que contienen diferentes observaciones. Con esto, se repite el proceso clásico con distintas observaciones y se evalúa la estabilidad de los resultados.

```
# Selección aleatoria de variables
rep <- 20
prop <- 0.7
modelosGenerados<-c()
for (i in 1:rep){
  set.seed(12345+i)
  subsample <-data_train[sample(1:nrow(data_train), prop * nrow(data_train), replace = T),]
  full <- glm(formIntT, data = subsample, family = binomial)
  null <- glm(varObj~1, data = subsample, family = binomial)
  modeloAux <- step(null,scope=list(lower=null,upper=full),direction="both",trace=0,k=log(nrow(sub
sample)))
```

```
  modelosGenerados <- c(modelosGenerados,paste(sort(unlist(strsplit(as.character(formula(modeloAux
))[3]," [+] "))),collapse = "+"))
}
(freq(modelosGenerados, sort = "dec") -> fr)
```

Una vez que ya hemos obtenido todos los modelos tanto los manuales como los de selección de variables clásica y selección de variables aleatoria; estamos en disposición de comparar todos los modelos que hemos ido seleccionando anteriormente para someterlo a la evaluación por validación cruzada repetida.

Como modelo ganador final se va a seleccionar el modelo 1 que se corresponde al modelo ganador generado en el modelado manual puesto que si se atiende a la cuestión de sesgo y varianza todos los modelos tienen un valor similar que varian muy poco pero, en cuanto a número de parámetros, existe un modelo que es el que menor número de parámetros tiene y este es el modelo manual. Cabe señalar que todos son buenos modelos ya que el valor del ROC en todos ellos es superior a 0,8 y muy similares entre si lo que indica una capacidad de predicción buena. Así pues, la razón diferencial a la hora de haber elegido el modelo ganador final ha sido el grado de complejidad de los modelos candidatos entendido como el número de parámetros de cada uno, cuanto menos parámetros más preferible es seleccionar el modelo en cuestión.

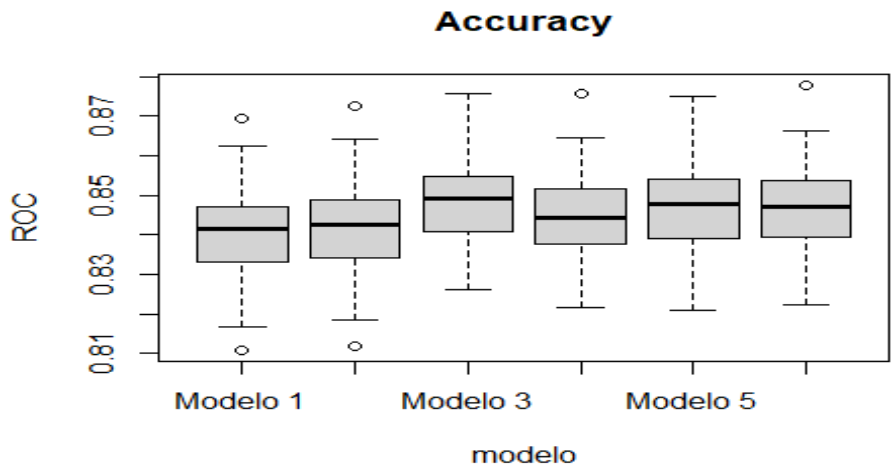
```
# Evaluación por validación cruzada repetida

total2<-c()
modelos2<-c(formula(modeloManual), formula(modeloEP), formula(modeloT),
            as.formula(paste('varObj ~', rownames(fr)[1])),
            as.formula(paste('varObj ~', rownames(fr)[2])),
            as.formula(paste('varObj ~', rownames(fr)[3])))

auxVarObj <- todo$varObj
todo$varObj <- make.names(todo$varObj)

for (i in 1:length(modelos2)){
  set.seed(1712)
  vcr<-train(as.formula(modelos2[[i]]), data = todo,
             method = "glm", family = binomial, metric = "ROC",
             trControl = trainControl(method="repeatedcv", number=5, repeats=20, summaryFunction =
twoClassSummary,
                                      classProbs = TRUE,
                                      returnResamp="all")
  )
  total2<-rbind(total2,cbind(vcr$resample[,1:2],modelo=rep(paste("Modelo",i),
                                                           nrow(vcr$resample))))
}

todo$varObj <- auxVarObj
boxplot(ROC~modelo,data=total2,main="Accuracy")
```




```
# Valores medios respecto a la media

aggregate(ROC~modelo, data = total2, mean)

##     modelo       ROC
## 1 Modelo 1 0.8405377
## 2 Modelo 2 0.8417758
## 3 Modelo 3 0.8484076 # El que mejor valor tiene
## 4 Modelo 4 0.8440269
## 5 Modelo 5 0.8464581
## 6 Modelo 6 0.8463671

# Desviación respecto a la media

aggregate(ROC~modelo, data = total2, sd)

##     modelo        ROC
## 1 Modelo 1 0.01123130 # El que mejor valor tiene
## 2 Modelo 2 0.01120747
## 3 Modelo 3 0.01032024
## 4 Modelo 4 0.01068589
## 5 Modelo 5 0.01033480
## 6 Modelo 6 0.01048827
```

Una vez que ya he decidido mi modelo ganador final, consulto sus parámetros para asegurarme que no exista colinealidad en el modelo (al ser un modelo manual donde se descartaban las transformaciones, ya sé de antemano que no existirá colinealidad en el modelo).

```
# Parámetros del modelo

modeloManual$formula

## varObj ~ Antig.fc.edad + Int_serv + Seguridad + Soporte_tecnico +
##     TV_streaming + Contrato + Fact_sinPapel + MetodoPago
```

Ya podemos consultar los parámetros de nuestro modelo ganador final y realizar las oportunas interpretaciones para conocer qué variables influyen más en el evento de fuga de los clientes en la compañía telefónica. La interpretación del modelo es bueno hacerla sobre los datos completos ya que los estimadores van a resultar ser más robustos al basarse en una gran cantidad de observaciones. Recordemos que al tratarse de una regresion logística, la interpretación no se hará a partir de los betas de las variables, es necesario calcular los OR aplicando las exponenciales; por ello se hará uso de la

librería epiDisplay que adicionalmente nos devolverá el intervalo de confianza y ajustará los OR por los grados de libertad que será finalmente los datos que interpretaremos.

A continuación se muestra los OR ajustados por los grados de libertad pero también puede observarse los OR en crudo. **(CONSULTAR DETALLES EN ANEXO)**

```
# Ajuste del modelo a datos completos para obtener estimadores fiables
```

```
modFinal <- glm(formula(modeloManual),data = todo_obj, family = binomial)
```

```
logistic.display(modFinal)
```

- Un aumento unitario del mes de antigüedad del cliente en la compañía telefónica reduce el riesgo de fuga en un 3%.
- Los clientes que tienen fibra óptica contratada tienen un riesgo 2,54 veces superior de fuga que los clientes que tienen ADSL contratado y la no contratación de servicios de internet tiene un riesgo de fuga 0,35 veces superior que la contratación de ADSL.
- La contratación de servicios de seguridad tiene asociado un riesgo de fuga de 0,63 veces superior a la no contratación de este servicio.
- El soporte técnico contratado conlleva un riesgo de fuga de 0,67 veces superior que la no contratación de soporte técnico.
- Los clientes con TV por streaming contratada tienen un riesgo de fuga 1,52 veces superior que los clientes que no han contratado este servicio.
- Los contratos anuales y bienales se asocian con un riesgo de fuga mayor que los contratos mensuales.
- No recibir la factura en papel por parte del cliente tiene un riesgo de fuga de 1,55 superior que los clientes que sí reciben la factura en papel.
- Los clientes que realizan los pagos mediante métodos no automáticos (electronic check y mailed check) tienen un riesgo de fuga de 1,39 veces superior a los clientes con métodos de pago automáticos (bank transfer y credit card).

De esta manera, parecer ser que el hecho de tener fibra óptica es el indicador de mayor riesgo en el evento de fuga del cliente de la compañía telefónica; por contra, la permanencia de un cliente en la compañía telefónica reduce en un 3% el evento de fuga por cada mes que transcurre en ella.

```
##                                                 adj. OR(95%CI)
## Antig.fc.edad (cont. var.)                      0.97 (0.97,0.97)
##
## Int_serv: ref.=DSL
##    Fiber optic                                  2.54 (2.18,2.98)
##    No                                           0.35 (0.27,0.45)
##
## Seguridad: 1 vs 0                               0.63 (0.53,0.75)
##
## Soporte_tecnico: 1 vs 0                         0.67 (0.56,0.79)
##
## TV_streaming: 1 vs 0                            1.52 (1.31,1.78)
##
## Contrato: ref.=Month-to-month
##    One year                                     0.46 (0.37,0.57)
##    Two year                                     0.21 (0.14,0.3)
##
```

```
## Fact_sinPapel: 1 vs 0                                      1.55 (1.33,1.8)
##
## MetodoPago: Not Automatic Payments vs Automatic Payments 1.39 (1.2,1.62)
##
```

Llega el momento de buscar el punto de corte para la probabilidad estimada y para ello, se utilizará una función que calcule este punto de corte y que involucre a Youden y Kappa.

```
# Modifico la función sensEspCorte para valorar en función de Youden y Kappa
sensEspCorte <- function(modelo,dd,nombreVar,ptoCorte,evento){

  probs <-predict(modelo,newdata=dd,type="response")

  cm<-confusionMatrix(data=factor(ifelse(probs>ptoCorte,1,0)), reference=dd[,nombreVar],positive=e
vento)

  c(cm$overall[1:2],cm$byClass[1:2])
}

# Generamos una rejilla de puntos de corte
posiblesCortes<-seq(0,1,0.01)

# Aplicamos función sensEspCorte a cada punto de la rejilla
rejilla<-data.frame(t(rbind(posiblesCortes,sapply(posiblesCortes,function(x)
  sensEspCorte(modeloManual,data_test,"varObj",x,"1")))))

# Generamos Youden
rejilla$Youden<-rejilla$Sensitivity+rejilla$Specificity-1

# Generamos Index
rejilla$Index <- rejilla$Kappa+rejilla$Youden

# Puntos de corte Youden
rejilla[which.max(rejilla$Youden),]

##    posiblesCortes  Accuracy     Kappa Sensitivity Specificity    Youden
## 37           0.36 0.7771654 0.4846297   0.7507418   0.7867095 0.5374514
##       Index
## 37 1.022081

# Punto máximo Index
rejilla[which.max(rejilla$Index),]

##    posiblesCortes Accuracy     Kappa Sensitivity Specificity   Youden    Index
## 38           0.37 0.780315 0.4875344   0.7418398   0.7942122 0.536052 1.023586
```

Los dos puntos de cortes obtenidos son el 0,36 y el 0,37 de los cuales voy a elegir para la probabilidad estimada el 0,37 porque podemos ver que tanto el accuracy, el valor Kappa y la especificidad mejoran levemente.

A continuación, en la matriz de confusión observamos un p-valor (9.872e-05) con el cual podemos rechazar la hipotesis nula de que el modelo es igual a un no modelo o modelo inocente. El valor Kappa (0.4875) es aceptable en el sentido que se encuentra en un rango de 0.4-0.6 con lo que podemos afirmar que la clasificación es buena (lo deseable son valores Kappa superiores a 0.8 con los cuales podriamos hablar de una clasificación perfecta). Para el contraste del test de Mcnemar temenos un p-valor (4.775e-10) bastante pequeño con el que podemos rechazar la hipotesis nula y afirmar que existe asociación significativa entre la realidad y las predicciones.

En cuanto a las predicciones, vemos que el modelo es capaz de reconocer a 741 de 933 (0) por lo que hay 192 falsos negativos y reconoce 250 de 337 (1) por lo que hay 87 falsos positivos.

Por ultimo, el valor del accuracy es lo bastante alto como para poder afirmar que no es un modelo basado puramente en el azar.

```
# Predicciones en test en forma de probabilidad estimada
predTest<-predict(modeloManual,data_test, type = "response")


clasTest<-factor(ifelse(predTest>0.37,1,0))

# Matriz de confusión
confusionMatrix(clasTest,data_test$varObj, positive = '1')

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 741  87
##          1 192 250
##
##                Accuracy : 0.7803
##                  95% CI : (0.7565, 0.8028)
##     No Information Rate : 0.7346
##     P-Value [Acc > NIR] : 9.872e-05
##
##                   Kappa : 0.4875
##
##  Mcnemar's Test P-Value : 4.775e-10
##
##             Sensitivity : 0.7418
##             Specificity : 0.7942
##          Pos Pred Value : 0.5656
##          Neg Pred Value : 0.8949
##              Prevalence : 0.2654
##          Detection Rate : 0.1969
##    Detection Prevalence : 0.3480
##       Balanced Accuracy : 0.7680
##
##        'Positive' Class : 1
##
```

Finalmente, llega el momento de aplicar el modelo a los datos de test para predecir cuántos clientes se fugarán. De esta manera, se predice que de los clientes de nuestro archivo de test un 68,4% no se fugarán de la compañía telefónica mientras que un 31,6% sí lo harán.

```
# Lectura de los datos de test
datosFuga <- readRDS("C:/Users/96has/Documents/NTIC/6. Documentación minería de Datos y Modelizaci
ón Predictiva-20211212/Tarea/FugaClientes_test.RDS")

# Cambio el nombre de la variable Antiguedad
colnames(datosFuga)[6] <- 'Antig.fc.edad'

# Cambio la variable Mayor65 que aparece como numérica y unifico categorías en MetodoPago
datosFuga$Mayor65  <- as.character(datosFuga$Mayor65)

datosFuga$MetodoPago <- car::recode(datosFuga$MetodoPago, "c('Bank transfer (automatic)', 'Credit
card (automatic)')
                                = 'Automatic Payments'; c('Electronic check', 'Mailed check') = 'No
t Automatic Payments'")

#Recodificación de los valores de las variables dicotómicas
datosFuga$Conyuge <- car::recode(datosFuga$Conyuge, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datosFuga$PersCargo <- car::recode(datosFuga$PersCargo, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datosFuga$Telf_serv <- car::recode(datosFuga$Telf_serv, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datosFuga$VariasLineas <- car::recode(datosFuga$VariasLineas, "'Yes' = 1; 'No' = 0", as.factor = T
```

```
RUE)
datosFuga$Seguridad <- car::recode(datosFuga$Seguridad, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datosFuga$CopiaSeguridad <- car::recode(datosFuga$CopiaSeguridad, "'Yes' = 1; 'No' = 0", as.factor
= TRUE)
datosFuga$Antivirus_disp <- car::recode(datosFuga$Antivirus_disp, "'Yes' = 1; 'No' = 0", as.factor
= TRUE)
datosFuga$Soporte_tecnico <- car::recode(datosFuga$Soporte_tecnico, "'Yes' = 1; 'No' = 0", as.fact
or = TRUE)
datosFuga$TV_streaming <- car::recode(datosFuga$TV_streaming, "'Yes' = 1; 'No' = 0", as.factor = T
RUE)
datosFuga$Peliculas <- car::recode(datosFuga$Peliculas, "'Yes' = 1; 'No' = 0", as.factor = TRUE)
datosFuga$Fact_sinPapel <- car::recode(datosFuga$Fact_sinPapel, "'Yes' = 1; 'No' = 0", as.factor =
TRUE)

# Aplico el modelo
varObjTest <- runif(nrow(datosFuga))
inputTest <- cbind(datosFuga, Transf_Auto(Filter(is.numeric, datosFuga), varObjTest))
todoTest <- data.frame(inputTest, varObjTest)
resultadosTest <- factor(ifelse(predict(modeloManual, todoTest, type = 'response') > 0.37, 1, 0))
datosFuga$Fuga <- resultadosTest
dfTest <- datosFuga[ ,c(1, 21)]
colnames(dfTest)[2] <- 'Fuga_pred'
freq(dfTest$Fuga_pred)

##     n    % val%
## 0 472 68.4 68.4
## 1 218 31.6 31.6

# Guardo los resultados en el archivo FugaPredict_HassanChafiXavier.RDS

saveRDS(dfTest, 'FugaPredict_HassanChafiXavier.RDS')
```

Los datos contenido en el archivo FugaPredict_HassanChafiXavier.RDS contienen el ID de los 690 clientes con su correspondiente predicción de fuga.