Tarea 1: ¡Cliente a la fuga!

Javier Aos Aragonés

1. Lectura e inspección de datos

```
# Seleccionamos el directorio de trabajo
setwd("G:/My Drive/1.3 Master/Modules/08_Data_mining/TareaI")
# Cargamos las funciones
source("Funciones_R.R")
# Cargamos paquetes necesarios
paquetes(c('questionr','psych','car','corrplot','ggplot2','gridExtra',
           'kableExtra','dplyr','DMwR2','caret','pROC','stats','glmnet',
           'epiDisplay'))
```

```
# Seleccionamos el directorio de trabajo
df <- readRDS("FugaClientes_Training.RDS")
str(df)
```

Lo primero que vemos es que la variable 'ID' está configurada como factor cuando todas sus observaciones son distintas entre sí. Ya que sus todo están compuestos tanto de números como de letras vamos a convertirla en una cadena de texto, aunque realmente no la vamos a usar en el modelo ya que no nos aporta información predictiva. Por lo demás, todas las variables parecen estar en su configuración correcta.

```
df$ID <- as.character(df$ID)
sapply(Filter(is.numeric, df),function(x) length(unique(x)))
```

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

```
# Comprobamos que las numéricas realmente lo sean
```

```
# Comprobamos rápidamente la distribución de las variables
summary(df)
```

En principio no vemos nada raro en la distribución de las variables. Quizás lo más llamativo podría ser el máximo de antigüedad de 72 años, aunque no tendría porqué ser un dato atípico o un error ya que perfectamente se podría dar la situación. Lo que también observamos rápidamente es que la mayoría de variables tienes todo perdidos, luego lo veremos con más detalle.

Como no se detectan errores graves vamos a pasar con el análisis de valores atípicos y perdidos. Primero separaremos la variable objetivo y crearemos un nuevo conjunto de todo sin ella.

```
varObjBin<-df$Fuga
input<-as.data.frame(df[,-(21)])
```

2. Depuración de los datos

Valores atípicos

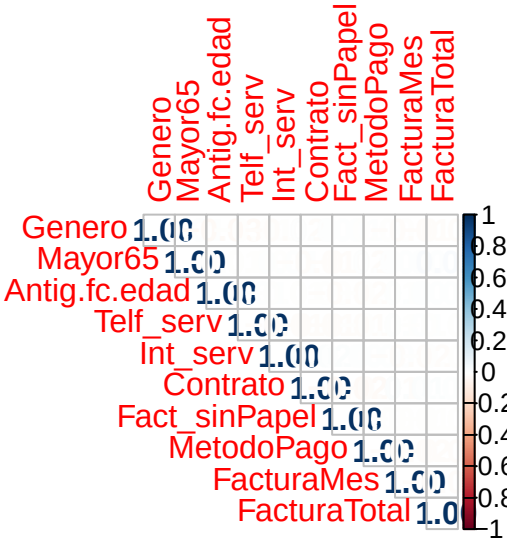
```
t<-data.frame(sort(
  round(sapply(Filter(
    is.numeric, input),function(nOut) atipicosAmissing(
      nOut)[[2]])/nrow(input)*100,3), decreasing = T))
names(t)<-"% Outliers por variable"
head(t, 5)
```

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

Como podemos ver no se detectan valores atípicos en las variables numéricas.

Valores perdidos

```
# Vemos la relación de missings
corrplot(cor(is.na(input[colnames(
  input)[colSums(is.na(input))>0]])),method = "number",type = "upper")
```



Vemos que no existe ningún patrón de correlación entre los valores perdidos.

```r
# Missing por variable
prop_missingsVars <- apply(is.na(input),2,mean)
t <- data.frame(sort(prop_missingsVars*100, decreasing = T))
names(t)<-"% Missing por Variable"
head(t, 10)
```

```
##               % Missing 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
```

Los valores perdidos no son extremadamente altos, por lo que los vamos a imputar por valores aleatorios.

```r
# Missing por observación
input$prop_missings<-apply(is.na(input),1,mean)
summary(input$prop_missings)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.02292 0.05000 0.15000
```

No hay observaciones con más de un 50% de valores perdidos, entonces conservamos todas.

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

```r
# Imputamos las cualitativas
input[,as.vector(which(sapply(input, class)=="factor"))]<-sapply(
  Filter(is.factor, input),function(x) ImputacionCuali(x,"aleatorio"))
```

```r
# Volvemos a pasar a factor los chr
input[,as.vector(which(sapply(input, class)=="character"))] <- lapply(
  input[,as.vector(which(sapply(input, class)=="character"))] , factor)
input$ID <- as.character(input$ID)
```

```r
# Reviso que no queden todo missings
sum(is.na(input))
```

```
## [1] 4
```

```r
# Volvemos a pasar el código para los missing que quedan
if (any(is.na(input))){
input[,as.vector(which(sapply(input, class)=="numeric"))]<-sapply(
```

```
  Filter(is.numeric, input),function(x) ImputacionCuant(x,"aleatorio"))
# Reviso que no queden todo missings
sum(is.na(input))
}
```

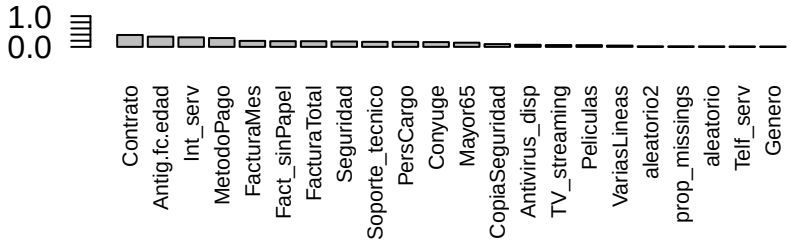
```
## [1] 0
```

```
# Guardamos el archivo depurado
saveRDS(cbind(varObjBin,input),"todoFugaDep.RDS")
```

3. Estudio de variables y relaciones con la variable objetivo

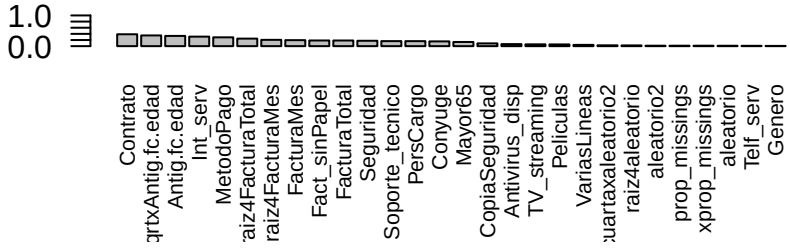
```
# Creamos dos variables aleatorias de control
input$aleatorio<-runif(nrow(input))
input$aleatorio2<-runif(nrow(input))
```

```
graficoVcramer(input[,-1],varObjBin)
```




```
# Buscamos las mejores transformaciones
input_bin<-cbind(input,Transf_Auto(Filter(is.numeric, input),varObjBin))
# Guardamos el dataset con las tranformaciones
todo_bin<-data.frame(input_bin,varObjBin)
saveRDS(todo_bin,"todo_bin_Vino.RDS")
```

```
graficoVcramer(input_bin[,-1],varObjBin) # Importancia de las transformaciones
```



Vemos que las transformaciones no mejoran demasiado la importancia de ninguna variable.

```
todo <- todo_bin
freq(todo$varObjBin)
```

```
##      n    % val%
## 0 4667 73.5 73.5
## 1 1686 26.5 26.5
```

Como podemos ver, en este caso, tenemos un desbalanceo hacia los 0 ya que su frecuencia representa el 73.5%. Esto quiere decir, que el modelo tendrá mayor dificultad en reconocer los 1 ya que posee menos información de ellos. Hay que tener cuidado si vemos que obtenemos una precisión del 73% mirando la sensibilidad y especificidad para comprobar el correcto funcionamiento del modelo.

```
set.seed(123456)
trainIndex <- createDataPartition(todo$varObjBin, p=0.8, list=FALSE)
data_train <- todo[trainIndex,c(2:23,30)]
data_test <- todo[-trainIndex,c(2:23,30)]

freq(data_train$varObjBin)
```

```
##      n    % val%
## 0 3734 73.5 73.5
## 1 1349 26.5 26.5
```

```
freq(data_test$varObjBin)
```

```
##     n    % val%
## 0 933 73.5 73.5
## 1 337 26.5 26.5
```

4. Modelado manual

```
# Creamos un modelo inicial completo
modeloInicial<-glm(varObjBin~.,data=data_train,family=binomial)
pseudoR2(modeloInicial,data_train,"varObjBin")
```

```
## [1] 0.2759208
```

```
pseudoR2(modeloInicial,data_test,"varObjBin")
```

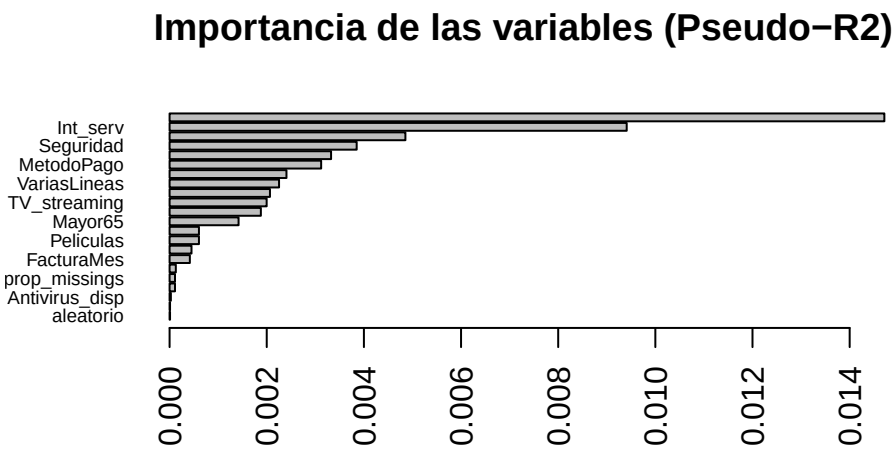
```
## [1] 0.2481161
```

```
modeloInicial$rank #número de parámetros
```

```
## [1] 27
```

Vemos que el modelo baja mucho su R2 en test, generaliza mal, probablemente sobrestima en train.

```
impVariablesLog(modeloInicial,"varObjBin")
```




```
# Modelo con las variables más importantes
modelo2<-glm(varObjBin~Contrato+Int_serv+Antig.fc.edad,
             data=data_train,family=binomial)
```

```
modelo3<-glm(varObjBin~Contrato*Antig.fc.edad+Int_serv,
             data=data_train,family=binomial)
pseudoR2(modelo3,data_train,"varObjBin")
```

```
## [1] 0.2415853
```

```
pseudoR2(modelo3,data_test,"varObjBin")
```

```
## [1] 0.2361613
```

```
modelo3$rank
```

```
## [1] 8
```

El modelo mejora mínimamente, pero vemos que existe un efecto potenciador interesante para la variable Contrato.

```
modelo4<-glm(varObjBin~Contrato+Antig.fc.edad+Int_serv+Seguridad+Fact_sinPapel,
             data=data_train,family=binomial)
```

```
modelo5<-glm(varObjBin~Contrato+Antig.fc.edad+Int_serv+Seguridad+Fact_sinPapel+Telf_serv+MetodoPago+Sop
             data=data_train,family=binomial)
```

```
modelo6<-glm(varObjBin~Contrato+Antig.fc.edad+Int_serv+Seguridad+Fact_sinPapel+Telf_serv+MetodoPago+Sop
```

```
modelo7<-glm(varObjBin~Contrato+Antig.fc.edad+Int_serv+Seguridad+Fact_sinPapel+Telf_serv+MetodoPago+Sop
pseudoR2(modelo7,data_train,"varObjBin")
```

```
## [1] 0.2740158
```

```
pseudoR2(modelo7,data_test,"varObjBin")
```

```
## [1] 0.2455866
```

```
modelo7$rank
```

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

Aquí vemos que conseguimos prácticamente los mismos R2 que en el modelo completo pero con 19 variables en vez de 27.

```
modelo8<-glm(varObjBin~Contrato+Antig.fc.edad+Int_serv+Seguridad+Fact_sinPapel+Telf_serv+MetodoPago+Sop
```

```
modelo9<-glm(varObjBin~Contrato*Antig.fc.edad+Int_serv+Seguridad+Fact_sinPapel+Telf_serv+MetodoPago+Sop
pseudoR2(modelo9,data_train,"varObjBin")
```

```
## [1] 0.2791125
```

```
pseudoR2(modelo9,data_test,"varObjBin")
```

```
## [1] 0.2524179
```

```
modelo9$rank
```

```
## [1] 23
```

Aquí vemos que incluso superamos al modelo completo pero con menos variables.

5. Modelado por selección de variables

Selección de variables clásica

Variables

En primer lugar, vamos a cargar los todo y crear las particiones.

```
# Hago la partición con las transformaciones
set.seed(123456)
trainIndex <- createDataPartition(todo$varObjBin, p=0.8, list=FALSE)
data_train <- todo[trainIndex,]
data_test <- todo[-trainIndex,]
```

Ahora vamos a crear un modelo vacío y otro completo

```
null<-glm(varObjBin~1, data=data_train, family = binomial) # Modelo vacío
full<-glm(varObjBin~., data=data_train[,c(2:23,30)], family = binomial) # Modelo completo
```

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

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

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

Variables + interacciones

```
formInt <- formulaInteracciones(todo[,c(2:23,30)],23)
fullInt <- glm(formInt, data=data_train, family = binomial) # Modelo completo
```

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

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

Variables + transformaciones

```
fullT <- glm(varObjBin~. -ID, data = data_train, family = binomial)

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

```
modeloStepBIC_trans<-step(null, scope=list(lower=null, upper=fullT), direction="both",k=log(nrow(data_t
```

Variables + transformaciones + interacciones

```
formIntT<-formulaInteracciones(todo[,c(-1)],29)
fullIntT<-glm(formIntT, data=data_train, family = binomial)
```

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

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

Selección de variables aleatorias

```
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(varObjBin~1,data=subsample,family=binomial)
  modeloAux<-step(null,scope=list(lower=null,upper=full),direction="both",trace=0,k=log(nrow(subsample))
  modelosGenerados<-c(modelosGenerados,paste(sort(unlist(strsplit(as.character(formula(modeloAux))[3],"
}
(freq(modelosGenerados,sort="dec")->fr)
```

```
modeloAleatorio1<-glm(varObjBin~Contrato+Fact_sinPapel+Int_serv+Mayor65+Peliculas+raiz4FacturaTotal+Segu
```

```
modeloAleatorio2<-glm(varObjBin~Contrato+Fact_sinPapel+Int_serv+Peliculas+raiz4FacturaTotal+Seguridad+S
```

```
modeloAleatorio3<-glm(varObjBin~Contrato+CopiaSeguridad+Fact_sinPapel+Int_serv+Mayor65+Peliculas+raiz4Fa
```

Selección de variables por Lasso

```
y <- as.double(as.matrix(data_train[, 30]))
x<-model.matrix(varObjBin~.-ID, data=data_train)[,-1] #no cambiar el -1
set.seed(1712)
cv.lasso <- cv.glmnet(x,y,nfolds=5)
```

```
(betas<-coef(cv.lasso, s=cv.lasso$lambda.1se))
```

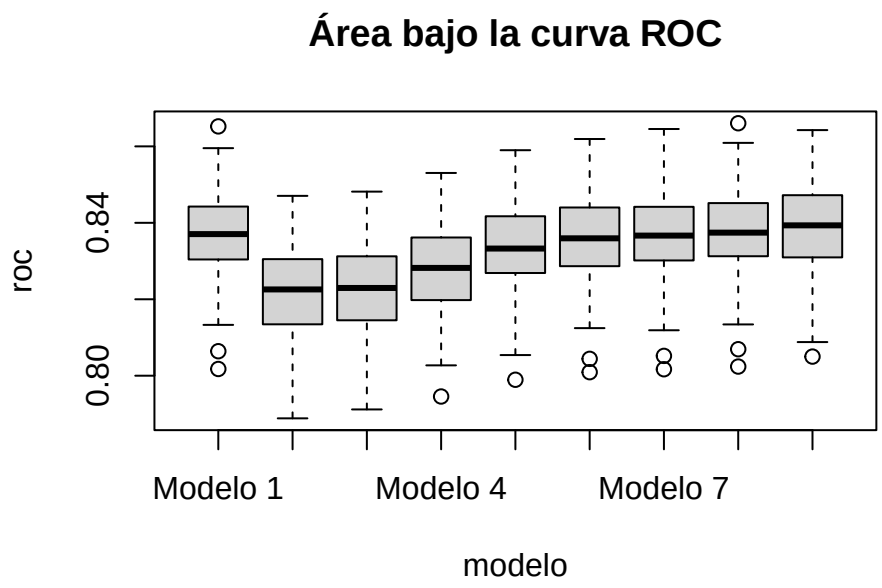
6. Comparación de modelos por validación cruzada repetida

Mejor modelo manual

```
# Copia de la variable original
auxVarObj <- todo$varObjBin

# Formateo la variable objetivo para que funcione el código
todo$varObjBin <-  make.names(todo$varObjBin)

total<-c()
modelos<-sapply(list(modeloInicial,modelo2,modelo3,modelo4,modelo5,modelo6,modelo7,modelo8,modelo9),for
for (i in 1:length(modelos)){
  set.seed(1712)
  vcr<-train(as.formula(modelos[[i]]), data = todo,
             method = "glm", family="binomial",metric = "ROC",
             trControl = trainControl(method="repeatedcv", number=5, repeats=20,
                                      summaryFunction=twoClassSummary,
                                      classProbs=TRUE,returnResamp="all")
  )
  total<-rbind(total,data.frame(roc=vcr$resample[,1],modelo=rep(paste("Modelo",i),
                                                                nrow(vcr$resample))))
}
boxplot(roc~modelo,data=total,main="Área bajo la curva ROC")
```




```
aggregate(roc~modelo, data = total, mean)
```

```
##     modelo       roc
## 1 Modelo 1 0.8368612
## 2 Modelo 2 0.8225731
## 3 Modelo 3 0.8229563
## 4 Modelo 4 0.8284432
## 5 Modelo 5 0.8338489
## 6 Modelo 6 0.8361252
## 7 Modelo 7 0.8371333
## 8 Modelo 8 0.8376023
## 9 Modelo 9 0.8390619
```

```
aggregate(roc~modelo, data = total, sd)
```

```
##     modelo        roc
## 1 Modelo 1 0.01153985
## 2 Modelo 2 0.01165079
## 3 Modelo 3 0.01151219
## 4 Modelo 4 0.01148835
## 5 Modelo 5 0.01185892
## 6 Modelo 6 0.01187137
## 7 Modelo 7 0.01170079
## 8 Modelo 8 0.01169318
## 9 Modelo 9 0.01148915
```

```
car::vif(modelo5)
```

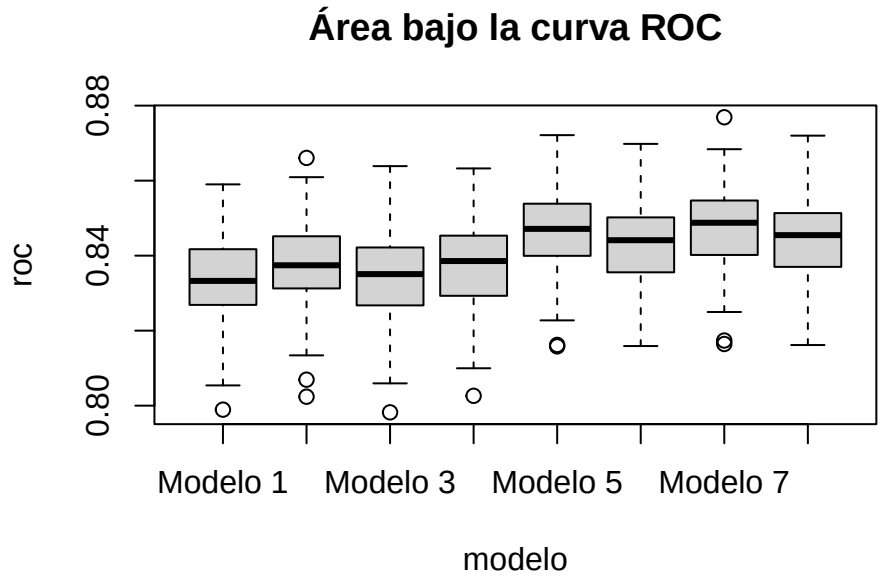
```
##                     GVIF Df GVIF^(1/(2*Df))
## Contrato        1.305163  2        1.068849
```

```
## Antig.fc.edad   1.405095  1        1.185367
## Int_serv        1.832080  2        1.163419
## Seguridad       1.099323  1        1.048486
## Fact_sinPapel   1.092284  1        1.045124
## Telf_serv       1.318523  1        1.148269
## MetodoPago      1.230603  3        1.035189
## Soporte_tecnico 1.127972  1        1.062060
```

Nos quedamos con el modelo 5, ya que a partir de ese modelo, los siguientes presentan un aumento mínimo del ROC, aumentando el número de variables. Por el principio de parsimonia, nos quedamos con el modelo que mayor aumento nos brinda con el mínimo número de parámetros, en este caso el modelo 5 con 13 variables.

Mejor modelo por selección de variables clásica

```
total<-c()
modelos<-sapply(list(modelo5,modeloStepAIC,modeloStepBIC,modeloStepBIC_int,
                     modeloStepAIC_trans,modeloStepBIC_trans,modeloStepAIC_transInt,modeloStepBIC_transI
for (i in 1:length(modelos)){
  set.seed(1712)
  vcr<-train(as.formula(modelos[[i]]), data = todo,
             method = "glm", family="binomial",metric = "ROC",
             trControl = trainControl(method="repeatedcv", number=5, repeats=20,
                                      summaryFunction=twoClassSummary,
                                      classProbs=TRUE,returnResamp="all")
  )
  total<-rbind(total,data.frame(roc=vcr$resample[,1],modelo=rep(paste("Modelo",i),
                                                                nrow(vcr$resample))))
}
boxplot(roc~modelo,data=total,main="Área bajo la curva ROC")
```



```
aggregate(roc~modelo, data = total, mean)
```

```
##     modelo       roc
## 1 Modelo 1 0.8338489
## 2 Modelo 2 0.8376023
## 3 Modelo 3 0.8348344
## 4 Modelo 4 0.8376206
## 5 Modelo 5 0.8465828
## 6 Modelo 6 0.8432927
## 7 Modelo 7 0.8475951
## 8 Modelo 8 0.8445595
```

```
aggregate(roc~modelo, data = total, sd)
```

```
##     modelo        roc
## 1 Modelo 1 0.01185892
## 2 Modelo 2 0.01169318
## 3 Modelo 3 0.01171669
## 4 Modelo 4 0.01149762
## 5 Modelo 5 0.01097127
## 6 Modelo 6 0.01059446
## 7 Modelo 7 0.01085398
## 8 Modelo 8 0.01047815
```

Los 2 mejores modelos son el 5 y 7, según el ROC.

```
length(coef(modeloStepAIC_trans))
```

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

```
length(coef(modeloStepAIC_transInt))
```

```
## [1] 48
```

```
length(coef(modeloStepBIC_transInt))
```

```
## [1] 14
```

```
length(coef(modeloStepBIC_trans))
```

```
## [1] 12
```

Sin embargo vemos que tanto el modelo 6 como el 8 (que son el mismo) tan solo con 12 parámetros se acercan mucho al ROC de los otros dos. Aplicando de nuevo el principio de parsimonia, seleccionamos el modelo 8 "modeloStepBIC_transInt".

Vamos a crear el mismo modelo pero sin la transformación raiz4FacturaTotal para ver si aporta mucho poco al ROC y así simplificar algo la interpretación.

```
modeloStepBIC_transInt2 <- glm(varObjBin ~ Contrato + Int_serv + FacturaTotal + TV_streaming +
    VariasLineas + Fact_sinPapel + Peliculas + Seguridad + Mayor65,data=data_train,family=binomial)
pseudoR2(modeloStepBIC_transInt2,data_test,"varObjBin")
```

```
## [1] 0.2243036
```

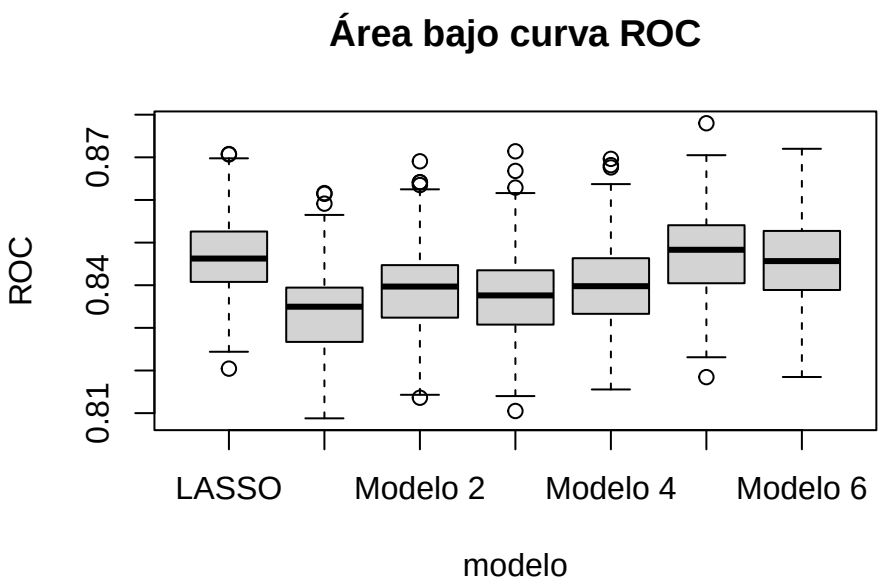
```
pseudoR2(modeloStepBIC_transInt,data_test,"varObjBin")
```

```
## [1] 0.2581719
```

Mejor modelo por selección de variables aleatoria y Lasso

```
auxVarObj_train <- data_train$varObjBin
data_train$varObjBin <- make.names(data_train$varObjBin)
total2<-c()
modelos2<-sapply(list(modelo5,modeloStepBIC_transInt,modeloStepBIC_transInt2,modeloAleatorio1,modeloAlea
for (i in 1:length(modelos2)){
  set.seed(1712)
  vcr<-train(as.formula(modelos[[i]]), data = data_train,
             method = "glm", family="binomial",metric = "ROC",
             trControl = trainControl(method="repeatedcv", number=5, repeats=20,
                                      summaryFunction=twoClassSummary,
                                      classProbs=TRUE,returnResamp="all")
  )
  total2<-rbind(total2,cbind(vcr$resample[,1:2],modelo=rep(paste("Modelo",i),
                                                           nrow(vcr$resample))))
}
set.seed(1712)
lassovcr <- train(varObjBin ~ . -ID, data = data_train,
                  method = "glmnet",family="binomial",metric="ROC",
                  tuneGrid=expand.grid(.alpha=1,.lambda=cv.lasso$lambda.1se),
                  trControl = trainControl(method="repeatedcv", number=5, repeats=20,
                                           returnResamp="all",summaryFunction=twoClassSummary,classProbs
)
total2<-rbind(total2,cbind(lassovcr$resample[,1:2],modelo=rep("LASSO",
                                                           nrow(vcr$resample))))

boxplot(formula=ROC~modelo,data=total2,main="Área bajo curva ROC")
```




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

```
##     modelo       ROC
## 1    LASSO 0.8468068
## 2 Modelo 1 0.8346643
## 3 Modelo 2 0.8393689
## 4 Modelo 3 0.8378920
## 5 Modelo 4 0.8399628
## 6 Modelo 5 0.8483622
## 7 Modelo 6 0.8458812
```

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

```
##     modelo        ROC
## 1    LASSO 0.01081810
## 2 Modelo 1 0.01144427
## 3 Modelo 2 0.01157412
## 4 Modelo 3 0.01181226
## 5 Modelo 4 0.01166164
## 6 Modelo 5 0.01100970
## 7 Modelo 6 0.01068338
```

Vemos que la mayoría tienen un ROC parecido siendo el mejor modelo el 5.

```
formula(modeloStepBIC_transInt2)
```

```
## varObjBin ~ Contrato + Int_serv + FacturaTotal + TV_streaming +
##     VariasLineas + Fact_sinPapel + Peliculas + Seguridad + Mayor65
```

7. Elección del modelo final

Vemos que el modelo aleatorio 2 consigue un ROC de 0.85 con el mismo número de variables que el modeloStepBIC_transInt2, el problema es que en el modelo aleatorio 2 se incluye la transformación de la raiz4FacturaTotal, mientras en en el modeloStepBIC_transInt2 se incluye la variable sin transformar, lo que hace mucho más facil su posterior interpretación. Si a esto le sumamos que el ROC tan solo gana un 0.01 con la variable transformada, vamos a seleccionar como modelo final el modeloStepBIC_transInt2.

8. Evaluación e interpretación de parámetros

Ajustamos el modelo final a todos los todo disponibles para su interpretación.

```
todo$varObjBin <- auxVarObj
modelofinal <- glm(formula(modeloStepBIC_transInt2), data=todo, family=binomial)
pseudoR2(modelofinal,todo,"varObjBin")
```

```
## [1] 0.2549706
```

```
car::vif(modelofinal)
```

```
##                    GVIF Df GVIF^(1/(2*Df))
## Contrato       1.320909  2        1.072058
## Int_serv       1.593980  2        1.123623
## FacturaTotal   2.099880  1        1.449096
## TV_streaming   1.449206  1        1.203830
## VariasLineas   1.355382  1        1.164209
## Fact_sinPapel  1.092899  1        1.045418
## Peliculas      1.432098  1        1.196703
## Seguridad      1.117560  1        1.057147
## Mayor65        1.058619  1        1.028892
```

```
epiDisplay::logistic.display(modelofinal)
```

```
##
## Logistic regression predicting varObjBin : 1 vs 0
##
##                                crude OR(95%CI)          adj. OR(95%CI)
## Contrato: ref.=Month-to-month
##    One year                    0.19 (0.16,0.23)         0.44 (0.36,0.54)
##    Two year                    0.06 (0.05,0.08)         0.22 (0.17,0.3)
##
## Int_serv: ref.=DSL
##    Fiber optic                 2.99 (2.62,3.41)         2.89 (2.46,3.4)
##    No                          0.38 (0.31,0.48)         0.39 (0.31,0.5)
##
## FacturaTotal (cont. var.)      0.9998 (0.9997,0.9998)   0.9996 (0.9995,0.9996)
##
## TV_streaming: Yes vs No        1.31 (1.17,1.47)         1.5 (1.27,1.76)
##
## VariasLineas: Yes vs No        1.21 (1.08,1.35)         1.35 (1.15,1.57)
```

```
##
## Fact_sinPapel: Yes vs No      2.49 (2.2,2.81)         1.48 (1.28,1.72)
##
## Peliculas: Yes vs No          1.29 (1.16,1.45)        1.44 (1.22,1.69)
##
## Seguridad: Yes vs No          0.36 (0.32,0.42)        0.64 (0.54,0.76)
##
## Mayor65: 1 vs 0               2.14 (1.86,2.46)        1.36 (1.15,1.61)
##
##                               P(Wald's test) P(LR-test)
## Contrato: ref.=Month-to-month                < 0.001
##    One year                   < 0.001
##    Two year                   < 0.001
##
## Int_serv: ref.=DSL                          < 0.001
##    Fiber optic                < 0.001
##    No                         < 0.001
##
## FacturaTotal (cont. var.)     < 0.001        < 0.001
##
## TV_streaming: Yes vs No       < 0.001        < 0.001
##
## VariasLineas: Yes vs No       < 0.001        < 0.001
##
## Fact_sinPapel: Yes vs No      < 0.001        < 0.001
##
## Peliculas: Yes vs No          < 0.001        < 0.001
##
## Seguridad: Yes vs No          < 0.001        < 0.001
##
## Mayor65: 1 vs 0               < 0.001        < 0.001
##
## Log-likelihood = -2738.6883
## No. of observations = 6353
## AIC value = 5501.3767
```

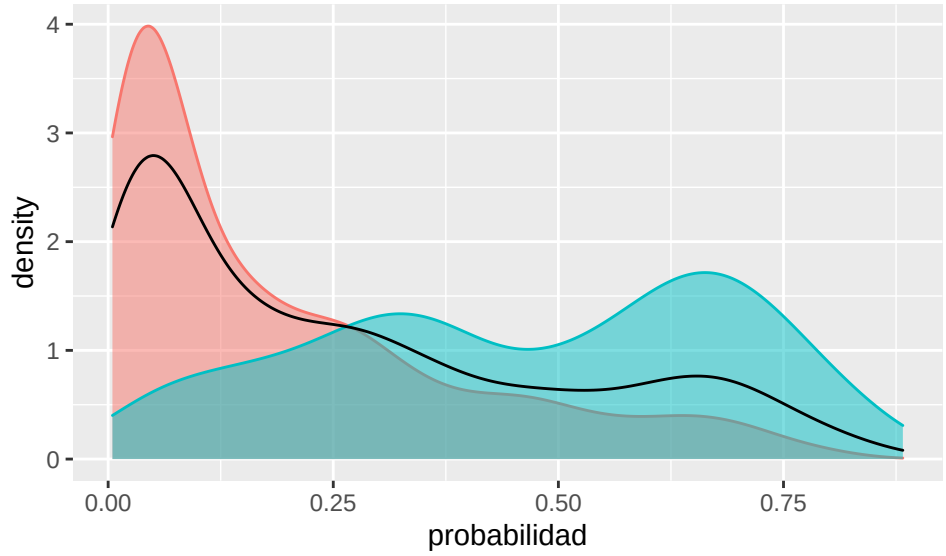
Viendo el modelo final podemos sacar las siguientes conclusiones:

- La probabilidad de fuga respecto a no fuga de un cliente con contrato de un año 0.44 veces la correspondiente a clientes con contrato mes a mes. Es decir, La probabilidad de fuga respecto a no fuga se reduce un 56% en clientes con contrato de un año respecto a clientes con contrato mensual.
- En el mismo caso para clientes con contrato de dos años, la probabilidad de fuga se reduce en un 80% frente a los clientes con contrato mensual. Aquí vemos que cuanto mayor el tiempo del contrato menor es la probabilidad de fuga.
- La probabilidad de fuga respecto a no fuga de un cliente con contrato de fibra óptica es 2.95 veces la correspondiente a clientes con contrato de DSL. Es decir, la probabilidad de fuga aumenta un 295% si el cliente tiene contratada fibra óptica con respecto a DSL.
- La probabilidad de fuga respecto a no fuga de un cliente sin contrato de Internet es 0.38 veces la correspondiente a clientes con contrato DSL. Es decir, un cliente sin contrato de Internet tiene un 62% menos probabilidad de fugarse que uno con contrato de Internet DSL.
- El aumento unitario de la Factura Total disminuye el odds del evento en un 0,0004%, pudiendo variar entre 0,0005% y 0,0004% con el 95% de confianza.
- La probabilidad de fuga es 1.49 veces mayor en clientes con TV en streaming contratado que en clientes sin el servicio.

- La probabilidad de fuga es 1.34 veces mayor en clientes con varias líneas de teléfono frente a los clientes con solo una línea.
- La probabilidad de fuga es 1.5 veces mayor en clientes con factura sin papel que en los clientes con factura en papel.
- La probabilidad de fuga es 1.42 veces mayor en clientes con el servicio de películas contratado que en clientes sin el servicio.
- La probabilidad de fuga es 0.64 veces mayor en clientes con servicio de seguridad contratado que en clientes sin el servicio.
- La probabilidad de fuga es 1.33 veces mayor en clientes con más de 65 años que en clientes con menos edad.

9. Búsqueda del punto de corte óptimo para la probabilidad estimada

```
# Gráfico de las probabilidades obtenidas
hist_targetbinaria(predict(modeloStepBIC_transInt2, newdata=data_test,type="response"),data_test$varObj
```



Parece que al modelo le irá bien reconociendo a los ceros (área roja) que como ya vimos que eran mayor porcentaje. En cambio, para los unos, que teníamos una menor representación, vemos que al modelo le cuesta más reconocerlos.

```
# Probamos dos puntos de corte
sensEspCorte(modeloStepBIC_transInt,data_test,"varObjBin",0.5,"1")
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.7913386      0.4866469      0.9013934      0.6406250      0.8293886
```

```
sensEspCorte(modeloStepBIC_transInt,data_test,"varObjBin",0.27,"1")
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.7362205      0.7774481      0.7213290      0.5019157      0.8997326
```

```r
# Generamos una rejilla de puntos de corte
posiblesCortes <- seq(0,1,0.01)
rejilla <- data.frame(t(rbind(posiblesCortes,sapply(posiblesCortes,function(x) sensEspCorte(modeloStepBI
rejilla$Youden <- rejilla$Sensitivity+rejilla$Specificity-1
```

```r
rejilla$posiblesCortes[which.max(rejilla$Youden)]
```

```
## [1] 0.27
```

```r
rejilla$posiblesCortes[which.max(rejilla$Accuracy)]
```

```
## [1] 0.52
```

```r
sensEspCorte(modeloStepBIC_transInt2,data_test,"varObjBin",0.26,"1")
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.7307087      0.7833828      0.7116827      0.4953096      0.9009498
```

```r
sensEspCorte(modeloStepBIC_transInt2,data_test,"varObjBin",0.53,"1")
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.7921260      0.4658754      0.9099678      0.6514523      0.8250729
```

```r
# Evaluamos la estabilidad del modelo a partir de las diferencias en train y test:
todo$varObjBin <- auxVarObj
data_train$varObjBin <- auxVarObj_train
pseudoR2(modeloStepBIC_transInt2,data_train,"varObjBin")
```

```
## [1] 0.261806
```

```r
pseudoR2(modeloStepBIC_transInt2,data_test,"varObjBin")
```

```
## [1] 0.2243036
```

Vemos que el modelo no parece muy estable ya que pierde bastante pseudoR2 para los datos de test frente a los de train.

```r
roc(data_train$varObjBin, predict(modeloStepBIC_transInt2,data_train,type = "response"), direction="<")
```

```
## Setting levels: control = 0, case = 1
```

```
##
## Call:
## roc.default(response = data_train$varObjBin, predictor = predict(modeloStepBIC_transInt2,     data_tr
##
## Data: predict(modeloStepBIC_transInt2, data_train, type = "response") in 3734 controls (data_train$va
## Area under the curve: 0.8357
```

```
roc(data_test$varObjBin, predict(modeloStepBIC_transInt2,data_test,type = "response"), direction="<")
```

```
## Setting levels: control = 0, case = 1
```

```
##
## Call:
## roc.default(response = data_test$varObjBin, predictor = predict(modeloStepBIC_transInt2,     data_tes
##
## Data: predict(modeloStepBIC_transInt2, data_test, type = "response") in 933 controls (data_test$varOb
## Area under the curve: 0.8147
```

```
sensEspCorte(modeloStepBIC_transInt2,data_train,"varObjBin",0.26,"1")
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.7424749      0.7842847      0.7273701      0.5096339      0.9032258
```

```
sensEspCorte(modeloStepBIC_transInt2,data_test,"varObjBin",0.26,"1")
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.7307087      0.7833828      0.7116827      0.4953096      0.9009498
```

10. Predicción para los datos de test

```
# Generar el factor con las clases estimadas en test
pred_test <- factor(ifelse(predict(modeloStepBIC_transInt2,data_test,type = "response")>0.26,1,0))

# Tablas marginales
table(pred_test)
```

```
## pred_test
##   0   1
## 737 533
```

```
# Matriz de confusión
confusionMatrix(pred_test,data_test$varObjBin, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 664  73
##          1 269 264
##
##                Accuracy : 0.7307
##                  95% CI : (0.7054, 0.7549)
##     No Information Rate : 0.7346
##     P-Value [Acc > NIR] : 0.6383
```

```
##
##                  Kappa : 0.4175
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7834
##             Specificity : 0.7117
##          Pos Pred Value : 0.4953
##          Neg Pred Value : 0.9009
##              Prevalence : 0.2654
##          Detection Rate : 0.2079
##    Detection Prevalence : 0.4197
##       Balanced Accuracy : 0.7475
##
##        'Positive' Class : 1
##
```

11. Construcción del dataset de entrega con el ID y Fuga_pred

```
data_train <- todo
data_test <- readRDS("FugaClientes_test.RDS")
data_test$Mayor65 <- as.factor(data_test$Mayor65)
```

```
probs <- predict(modelofinal, data_test, type='response')
data_test$Fuga_pred <- factor(ifelse(probs>0.26, 1, 0))
FugaPredict_JavierAos <- data_test %>% dplyr::select(ID, Fuga_pred)
```