Random forest. Detección de churn. Caso Telecomunicaciones

Adolfo Sánchez Burón

Introducción al algoritmo de Random Forest

Random forest es uno de los denóminados en Machine Learning como métodos de ensamble (ensemble method), esto es, son técnicas que convinan múltiples algoritmos, permitiendo alcanzar una mayor precisión y estabilidad del modelo resultante. Los tres métodos más comunes son: Bagging, Boosting and Stacking. RF es del primero de ellos.

Por tanto, RF es un método de ensamble que permite construir una multitud de árboles decisión en la fase de entrenamiento, posibilitando corregir aspectos como el sobre ajuste (sesgo-varianza), y mejorando el resultado final. Permite tareas de clasificación, regresión y otras (como determinar la importancia relativa de las variables en un mdelo de aprendizaje).

RF trabaja de la siguiente manera: construye árboles de decisión individuales,los cuales crecen hasta su máxima extensión posible sin ue medie un proceso de poda. Para ello, se emplean con los datos de entrenamiento, pero incluyendo datos ligeramente distintos en cada árbol (bootstrapping). La predicción final, mediante un algoritmo de RF, es la media de las predicciones de todos los árboles que lo forman.

En RF es importante considerar toda una serie de hipeparámetros:

- ntree: número de árboles a incluir en el modelo de RF.
- mtry: número de variables predictoras como candidatas en cada ramifiación. Por defecto, apra problemas de casificación será la raiz cuadrada del número de variables, y para regresión número de variables dividido entre 3.
- sampsize: el número de muestras sobre las cuales entrenar.
- nodesize: mínimo número de muestras dentro de los nodos terminales. Para casos de clasificación, por defecto será 2, y para regresión será 5.
- maxnodes: máximo número de nodos terminales. Por defecto, no se planifica un proceso de poda, dejando crecer los árboles hasta su límite máximo.

Se recomienda la lectura de:

Amat,J.(2017). Árboles de decisión, random forest, gradient boosting y C5.0 (https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_c50#Random_Forest)

Orellana,J.(2018). Arboles de decision y Random Forest (https://bookdown.org/content/2031/)

Características del caso

El caso empleado en este análisis es el 'Telco Customer Churn', que puede descargarse el dataset original de Kaggle (https://www.kaggle.com/blastchar/telco-customer-churn). Este dataset ha sido previamente trabajado en cuanto a:

- análisis descriptivo
- limpieza de anomalías, missing y outliers
- peso predictivo de las variables mediante random forest
- discretización de las variables continuas para facilitar la interpretación posterior

Por lo que finalmente se emplea en este caso un dataset prepardo para iniciar el análisis de RL, que puede descargarse de Github.

El objetivo del caso es predecir la probabilidad de que un determinado cliente puede abandonar (churn) la empresa. La explicación de esta conducta estará basada en toda una serie de variables predictoras que se pueden clasificar en cuatro grupos:

- Churn: la variable TARGET, con puntuaciones de 0 (no abandonó la empresa) y 1 (sí abandonó la empresa)
- Servicios contratados: phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies.
- Información sobre cuentas dle cliente: how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges.
- Variables demográficas: gender, age range, and if they have partners and dependents

Tras estudiar el peso predictivo de estas variables sobre la TARGET, finalmente se redujo el número de predictores a 6: Internet Service, Contract, Payment Method, tenure, Monthly Charges y Total Charges.

Proceso

1. Entorno

El primer punto tratará sobre la preparación del entorno, donde se mostrará la descarga de las librerías empleadas y la importación de datos.

2. Análisis descriptivo

Se mostrarán y explicarán las funciones empleadas en este paso, dividiéndolas en tres grupos: Análisis inicial, Tipología de datos y Análisis descriptivo (gráficos).

3. Modelización

Se preparará lo necesario para modelizar, mediante dos pasos:

a. Preparar funciones:

- Matriz de confusión
- Métricas
- Umbrales
- Curva ROC y AUC

b. Particiones del dataset en dos grupos: training (70%) y test (30%)

4. Modelización con Random Forest

Por motivos didácticos, se dividirá en seis pasos:

- Paso 1. Primer modelo
- Paso 2. Segundo modelo
- Paso 3. Predict
- Paso 4. Umbrales
- Paso 5. Matriz de confusión

- Paso 6. Métricas definitivas

1. Entorno

1.1. Instalar librerías

```
library(data.table) #para leer y escribir datos de forma rapida
library(dplyr) #para manipulación de datos
library(tidyr) #para manipulación de datos
library(ggplot2) #para gráficos
library(ROCR) #para evaluar modelos
library(DataExplorer) #para realizar el análisis descriptivo con gráficos
library(randomForest) #para modelizar con RF
```

1.2. Importar datos

Como el dataset ha sido peviamente trabajado para poder modelizar directamente, si deseas seguir este tutorial, lo puedes descargar de GitHub (https://github.com/AdSan-R).

```
# Importamos los datos y los incluimos en un data frame llamado df1
df1 <- fread("TelcoChurn.csv")
```

```
options(scipen=999) #Desactivar la notación científica
```

2. Análisis descriptivo

2.1. Análisis inicial

```
head(df1) #con esta función podemos ver la estructura de los primeros 6 casos
```

```
##    InternetService       Contract             PaymentMethod tenure_DISC
## 1:             DSL Month-to-month          Electronic check     Grupo 1
## 2:             DSL       One year              Mailed check     Grupo 2
## 3:             DSL Month-to-month              Mailed check     Grupo 1
## 4:             DSL       One year Bank transfer (automatic)     Grupo 3
## 5:     Fiber optic Month-to-month          Electronic check     Grupo 1
## 6:     Fiber optic Month-to-month          Electronic check     Grupo 1
##    MonthlyCharges_DISC TotalCharges_DISC TARGET
## 1:             Grupo 1           Grupo 1     No
## 2:             Grupo 2           Grupo 3     No
## 3:             Grupo 2           Grupo 1     Si
## 4:             Grupo 2           Grupo 3     No
## 5:             Grupo 3           Grupo 1     Si
## 6:             Grupo 4           Grupo 2     Si
```

```
str(df1) #mostrar la estructura del dataset y los tipos de variables
```

```
## Classes 'data.table' and 'data.frame':   7032 obs. of  7 variables:
##  $ InternetService    : chr  "DSL" "DSL" "DSL" "DSL" ...
##  $ Contract           : chr  "Month-to-month" "One year" "Month-to-month" "One year" ...
##  $ PaymentMethod      : chr  "Electronic check" "Mailed check" "Mailed check" "Bank transfer (automati
c)" ...
##  $ tenure_DISC        : chr  "Grupo 1" "Grupo 2" "Grupo 1" "Grupo 3" ...
##  $ MonthlyCharges_DISC: chr  "Grupo 1" "Grupo 2" "Grupo 2" "Grupo 2" ...
##  $ TotalCharges_DISC  : chr  "Grupo 1" "Grupo 3" "Grupo 1" "Grupo 3" ...
##  $ TARGET             : chr  "No" "No" "Si" "No" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Puede observarse que todas son “chr”, esto es, “character”, por tanto, vamos a pasarlas a Factor.

2.2. Tipología de datos

```
df1 <- mutate_if(df1, is.character, as.factor) #identifica todas las variables character y transformarlas
 en factores

str(df1) #estructura de la base de datos después de la transformación
```

```
## Classes 'data.table' and 'data.frame':   7032 obs. of  7 variables:
##  $ InternetService    : Factor w/ 3 levels "DSL","Fiber optic",..: 1 1 1 1 2 2 2 1 2 1 ...
##  $ Contract           : Factor w/ 3 levels "Month-to-month",..: 1 2 1 2 1 1 1 1 1 2 ...
##  $ PaymentMethod      : Factor w/ 4 levels "Bank transfer (automatic)",..: 3 4 4 1 3 3 2 4 3 1 ...
##  $ tenure_DISC        : Factor w/ 4 levels "Grupo 1","Grupo 2",..: 1 2 1 3 1 1 2 1 2 4 ...
##  $ MonthlyCharges_DISC: Factor w/ 4 levels "Grupo 1","Grupo 2",..: 1 2 2 2 3 4 3 1 4 2 ...
##  $ TotalCharges_DISC  : Factor w/ 4 levels "Grupo 1","Grupo 2",..: 1 3 1 3 1 2 3 1 3 3 ...
##  $ TARGET             : Factor w/ 2 levels "No","Si": 1 1 2 1 2 2 1 1 2 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

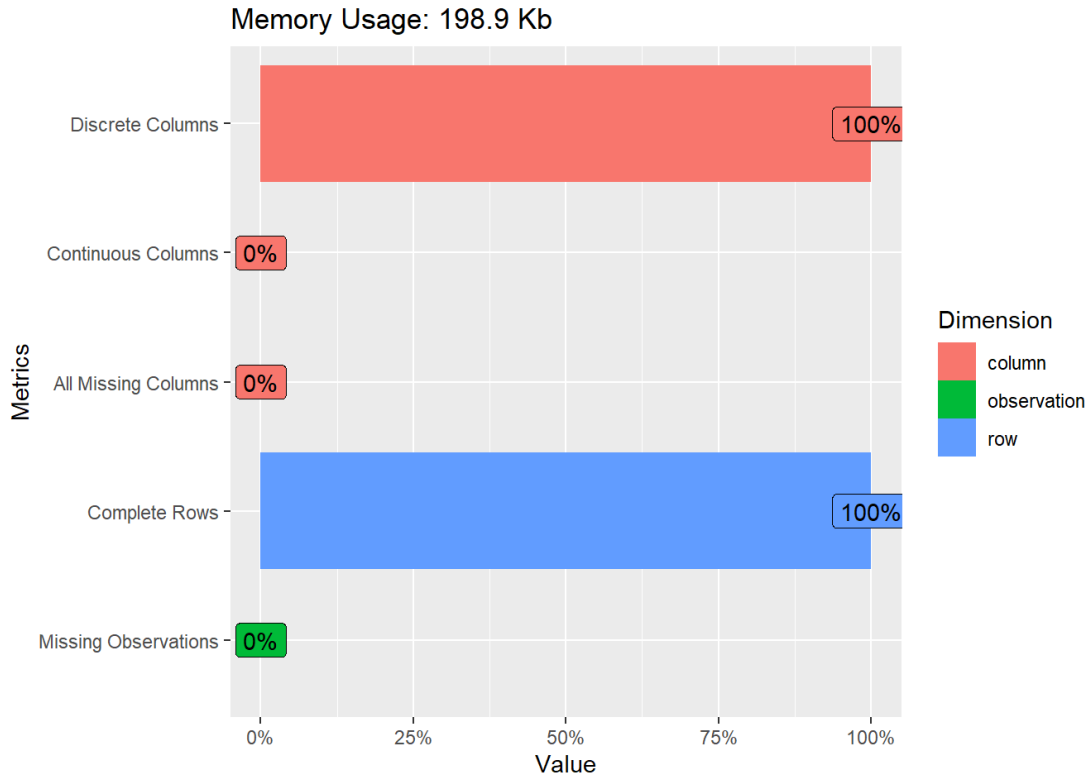
Ahora se puede observar que todas las variables son de tipo “Factor”

```
lapply(df1,summary) #mostrar la distribución de frecuencias en cada categoría de todas las variables
```

```
## $InternetService
##         DSL Fiber optic          No
##        2416        3096        1520
##
## $Contract
## Month-to-month       One year       Two year
##           3875           1472           1685
##
## $PaymentMethod
## Bank transfer (automatic)   Credit card (automatic)          Electronic check
##                       1542                       1521                       2365
##              Mailed check
##                       1604
##
## $tenure_DISC
## Grupo 1 Grupo 2 Grupo 3 Grupo 4
##    2723    1308    1182    1819
##
## $MonthlyCharges_DISC
## Grupo 1 Grupo 2 Grupo 3 Grupo 4
##    1758    1761    1755    1758
##
## $TotalCharges_DISC
## Grupo 1 Grupo 2 Grupo 3 Grupo 4
##    1758    1758    1758    1758
##
## $TARGET
##   No   Si
## 5163 1869
```

2.3. Análisis descriptivo (gráficos)

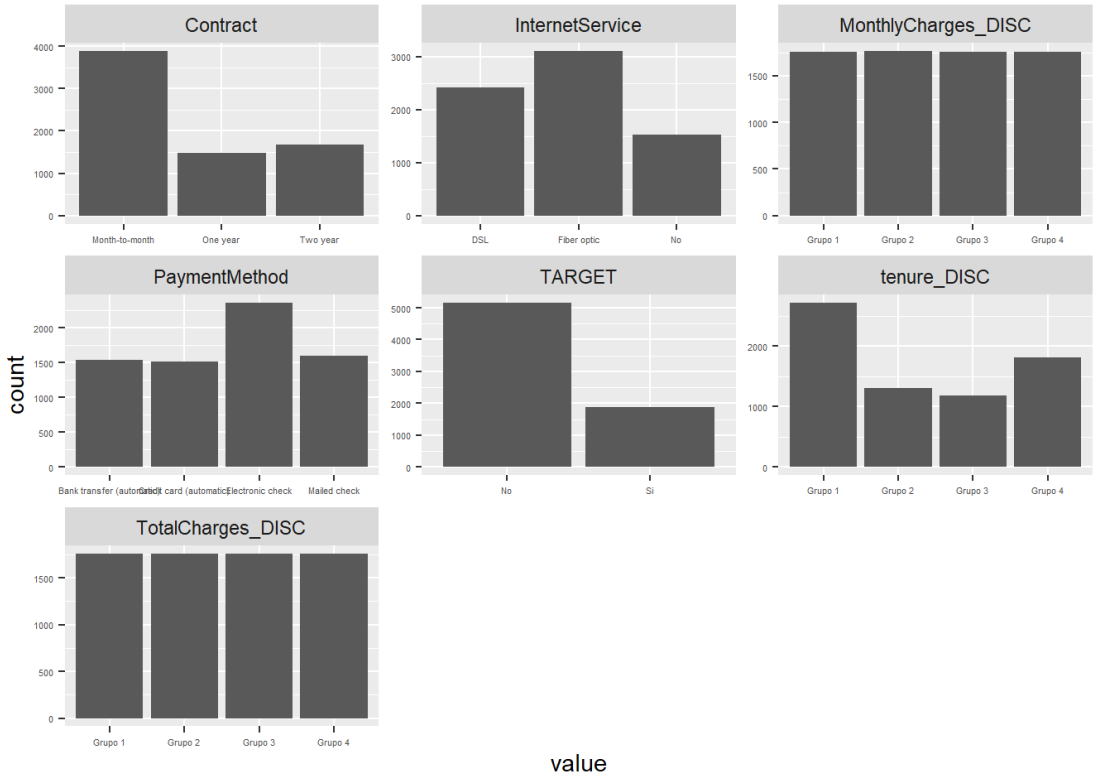
```
plot_intro(df1) #gráfico para observar la distribución de variables y los casos missing por columnas, obse
rvaciones y filas
```



Como se ha trabajado previamente, no existen casos missing, por lo que podemos seguir el análisis descriptivo

```
#Análisis visual de frecuencias de cada categoría por variable
df1 %>%
  gather() %>%
  ggplot(aes(value)) +
  geom_bar()+
  facet_wrap(~ key, scales = "free")+
  theme(axis.text=element_text(size=4))
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```



En los gráficos anteriores pueden observarse las categorías de cada variable, algunas de ellas dicotomizadas previamente, por lo que haremos un repaso de cada una:

- Internet Service: tiene tres niveles: DSL, Fiber optic, No.
- Contract (tipo de contrato): tiene tres niveles, Month-to-month, One year, Two years.
- Payment Method: con tres niveles, Bank transfer, Credit card, Electronic check.
- Tenure: variable originalmente cuantitativa, que se discretizó en cuatro categorías por cuartiles.
- Monthly Charges: se discretizó en cuatro categorías. Grupo 1 (<= 35.59), Grupo 2 (> 35.59 & <= 70.35), Grupo 3 (> 70.35 & <= 89.86), Grupo 4 (> 89.86).
- Total Charges: se discretizó en cuatro categorías. Grupo 1 (<= 401.4), Grupo 2 (> 401.4 & <= 1397.5), Grupo 3 (> 1397.5 & <= 3794.7), Grupo 4 (> 3794.7).
- TARGET: con dos niveles Sí han abandonado (churn), No han abandonado (churn).

Parece que la distribución de frecuencias en todas las variables es aceptable, incluso en la variable TARGET, que suele dar más problemas.

---

3. Modelización

3.1. Preparar funciones

Tomadas del curso de Machine Learning Predictivo (https://www.datascience4business.com/08_mlc-salespage-b) de DS4B) :

- Matriz de confusión
- Métricas
- Umbrales
- Curva ROC y AUC

Función para la matriz de confusión

En esta función se prepara la matriz de confusión (ver en otro post), donde se observa qué casos coinciden entre la puntuación real (obtenida por cada sujeto) y la puntuación predicha ("scoring") por el modelo, estableciendo previmente un límite ("umbral") para ello.

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

Funcion para métricas de los modelos

Los indicadores a observar serán:

- Acierto (accuracy) = (TRUE POSITIVE + TRUE NEGATIVE) / TODA LA POBLACIÓN
- Precisión = TRUE POSITIVE / (TRUE POSITIVE + FALSE POSITIVE)
- Cobertura (recall, sensitivity) = TRUE POSITIVE / (TRUE POSITIVE + FALSE NEGATIVE)
- F1 = 2* (precisión * cobertura) (precisión + cobertura)

```
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

Con esta función se analiza el efecto que tienen distintos umbrales sobre los indicadores de la matriz de confusión (precisión y cobertura). Lo que buscaremnos será aquél que maximice la relación entre cobertura y precisión (F1).

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

Funciones para calcular la curva ROC y el AUC

Por último, se preapra una función para calcular la curva ROC y el AUC.

- Curva ROC (Relative Operating Characteristic): resentación gráfica de la relación entre la cobertura (proporción de verdaderos positivos) y la especificidad (razón de falsos positivos). Muestra el rendimiento del modelo en todos los umbrales de clasificación.
- AUC (Area Under The Curve): mide el área que queda debajo de la curva. Indica en qué medida el modelo será capaz de clasificar adecuadamente. La AUC tiene un rango entre 0 y 1. Si es igual o cercano a 0.5, no tiene capacidad discriminativa.

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

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

3.2. Particiones de training (70%) y test (30%)

Se divide la muestra en dos partes:

1. Training o entrenamiento (70% de la muestra): servirá para entrenar al modelo de clasificación.
2. Test (30%): servirá para validar el modelo. La característica fundamental es que esta muestra no debe haber tenido contacto previamente con el funcionamiento del modelo.

```
# Lanzamos una semilla para que salgan siempre los mismos datos
set.seed(12345)

# Creamos los dataframes

# Generamos una variable aleatoria con una distribución 70-30
df1$random<-sample(0:1,size = nrow(df1),replace = T,prob = c(0.3,0.7))

train<-filter(df1,random==1)
test<-filter(df1,random==0)

#Eliminamos ya la random
df1$random <- NULL
```

---

4. Modelización con Random Forest

Paso 1. Primer modelo

```
rf<-randomForest(TARGET ~ InternetService + Contract + PaymentMethod + tenure_DISC + MonthlyCharges_DISC +
TotalCharges_DISC,train,importance=T)
rf
```

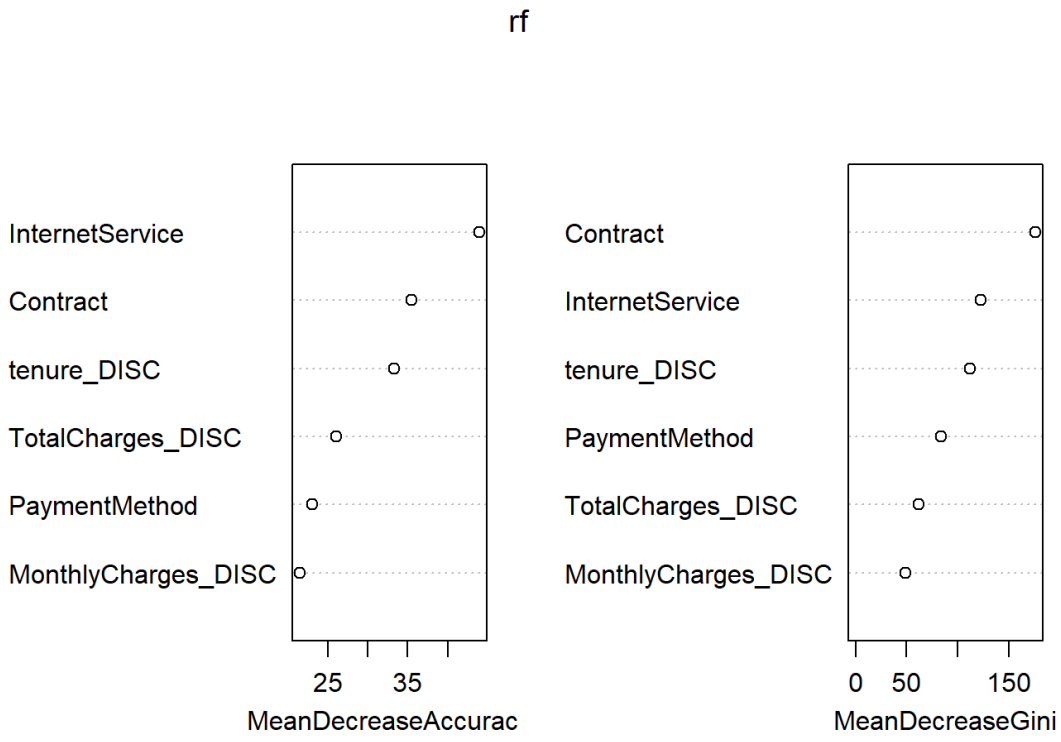
```
##
## Call:
##  randomForest(formula = TARGET ~ InternetService + Contract +      PaymentMethod + tenure_DISC + Monthl
yCharges_DISC + TotalCharges_DISC,      data = train, importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 21.52%
## Confusion matrix:
##      No  Si class.error
## No 3099 482   0.1345993
## Si  580 773   0.4286770
```

Empleamos la función randomForest, que nos da la siguiente información:

- Type of random forest: classification (ya que trabajamos con una TARGET binaria)
- Number of trees: 500 árboles analizados
- No. of variables tried at each split: número de variables seleccionadas para cada árbol = 2
- OOB estimate of error rate: 21.52%
- Además aporta una matriz de confusión.

Visualización de importancia relativa de cada variable en el modelo

```
varImpPlot(rf)
```



Aporta dos indicadores TASA DE ACIERTO (MeanDecreaseAccuracy) y el ÍNDICE DE GINI. Aunque normalmente coinciden, pueden no hacerlo.

Interpretación:

Se puede observar que:

a. En los dos indicadores, la variable con puntuación casi nula es MonthlyCharges_DISC.

b. Las siguientes variables con menos peso, en el primer indicador, es PaymentMethod, y en el segundo es TotalCharges_DISC.

Se opta por eliminar del modelo a la variable MonthlyCharges_DISC y quedarnos con las otras dos.

Paso 2. Segundo modelo

Creamos de nuevo el modelo con las nuevas variables

```
rf2<-randomForest(TARGET ~ InternetService + Contract + PaymentMethod + tenure_DISC + TotalCharges_DISC,tr
ain,importance=T)
rf2
```

```
##
## Call:
##  randomForest(formula = TARGET ~ InternetService + Contract +      PaymentMethod + tenure_DISC + TotalC
harges_DISC, 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: 22.42%
## Confusion matrix:
##      No  Si class.error
## No 3138 443   0.1237085
## Si  663 690   0.4900222
```

Aporta la siguiente información:

- Number of trees: 500 (500 árboles analizados)
- No. of variables tried at each split: 2 (número de variables seleccionadas para cada árbol = 2)
- OOB estimate of error rate: 22.42%

Paso 3. Predict

Aplicamos el modelo entrenado al conjunto de test (30%), generando un vector con las probabilidades en cada caso de ser 0 o 1.

Notar que por el método predict de randomForest hay que poner el “type = prob” para tener el scoring, lo cual nos reporta una matriz con dos columnas (hacer churn o no hacer churn). Nos quedamos con la segunda columna “[,2]”

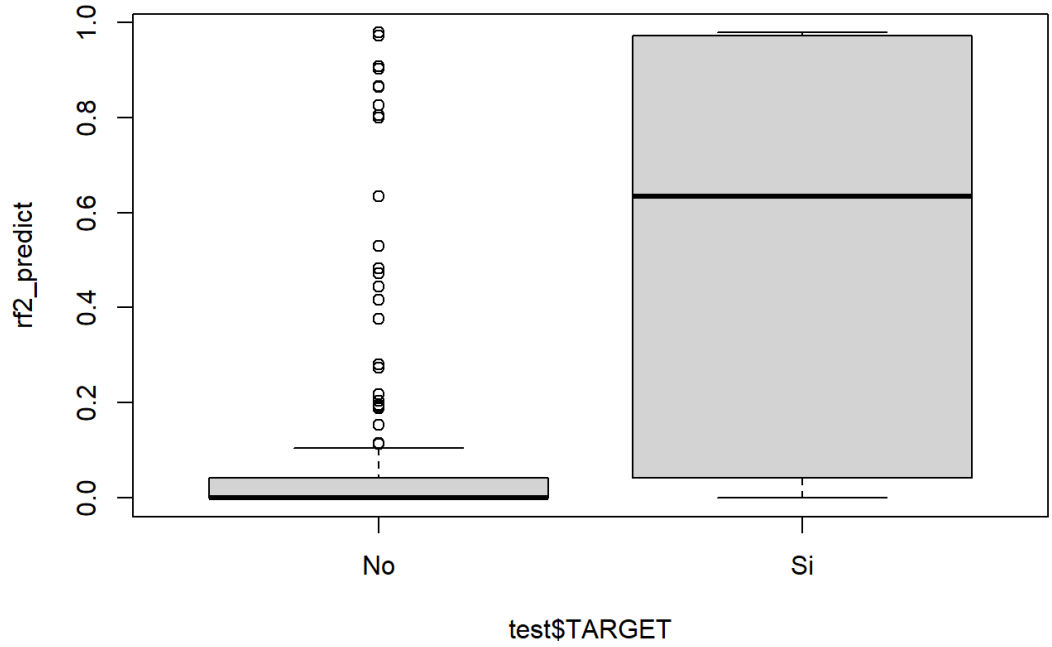
```
rf2_predict<-predict(rf2,test,type = 'prob')[,2]
head(rf2_predict)
```

```
##     1     2     3     4     5     6
## 0.482 0.000 0.078 0.000 0.634 0.000
```

Lanzamos un “head” para ver los 6 primeros. Lo que quiere decir que: el sujeto 1 tendrá una probabilidad de clasificarse como 1 (Sí Churn) del 48,2%. El segundo de 0,0%, etc.

A continuación lanzamos un plot de caja y bigotes, para ver si discrimina bien entre las dos categorías, esto es, si la media de rf2_predict de los clientes que sí hacen churn con la media de los que no hacen churn es diferente.

```
plot(rf2_predict~test$TARGET)
```



Se observa en la gráfica que la media de los que Sí y los que No es muy diferente, incluso discrimina bien entre los cuartiles.

Paso 4. Umbrales

Ahora vamos a transformar la probabilidad obtenida en otra dicotómica, es decir, cuando el cliente SÍ hara churn o NO.

Con la función umbrales probamos diferentes cortes

```
umb_rf2<-umbrales(test$TARGET,rf2_predict)
umb_rf2
```

```
##    umbral  acierto precision cobertura       F1
## 1    0.05 75.64347  50.33289  73.25581 59.66851
## 2    0.10 77.12107  52.78638  66.08527 58.69191
## 3    0.15 77.88370  54.20712  64.92248 59.08289
## 4    0.20 79.69495  58.15217  62.20930 60.11236
## 5    0.25 80.21926  59.40410  61.82171 60.58879
## 6    0.30 80.40991  59.84991  61.82171 60.81983
## 7    0.35 80.40991  59.84991  61.82171 60.81983
## 8    0.40 80.64824  60.41667  61.82171 61.11111
## 9    0.45 80.55291  60.38462  60.85271 60.61776
## 10   0.50 80.02860  60.94808  52.32558 56.30865
## 11   0.55 80.60057  63.26034  50.38760 56.09493
## 12   0.60 80.60057  63.26034  50.38760 56.09493
## 13   0.65 80.40991  65.39589  43.21705 52.04201
## 14   0.70 80.40991  65.39589  43.21705 52.04201
## 15   0.75 80.40991  65.39589  43.21705 52.04201
## 16   0.80 80.40991  65.39589  43.21705 52.04201
## 17   0.85 79.93327  66.10169  37.79070 48.08878
## 18   0.90 79.40896  67.07317  31.97674 43.30709
## 19   0.95 79.07531  67.74194  28.48837 40.10914
```

Seleccionamos automáticamente el mejor umbral

```
umbral_final_rf2<-umb_rf2[which.max(umb_rf2$F1),1]
umbral_final_rf2
```

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

Como puede observarse en la tabla anterior, el indicador F1 crece a medida que los umbrales aumentan (esto es, se maximiza progresivamente la F1), pero llega a un punto que empieza a decrecer: umbral de 0.4.

Paso 5. Matriz de confusión

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

```
confusion(test$TARGET,rf2_predict,umbral_final_rf2)
```

```
##
## real FALSE TRUE
##   No  1373  209
##   Si   197  319
```

```
rf2_metricas<-filter(umb_rf2,umbral==umbral_final_rf2)
rf2_metricas
```

```
##   umbral  acierto precision cobertura       F1
## 1    0.4 80.64824  60.41667  61.82171 61.11111
```

Observamos que para el umbral 0.4, tenemos un modelo con las métricas:

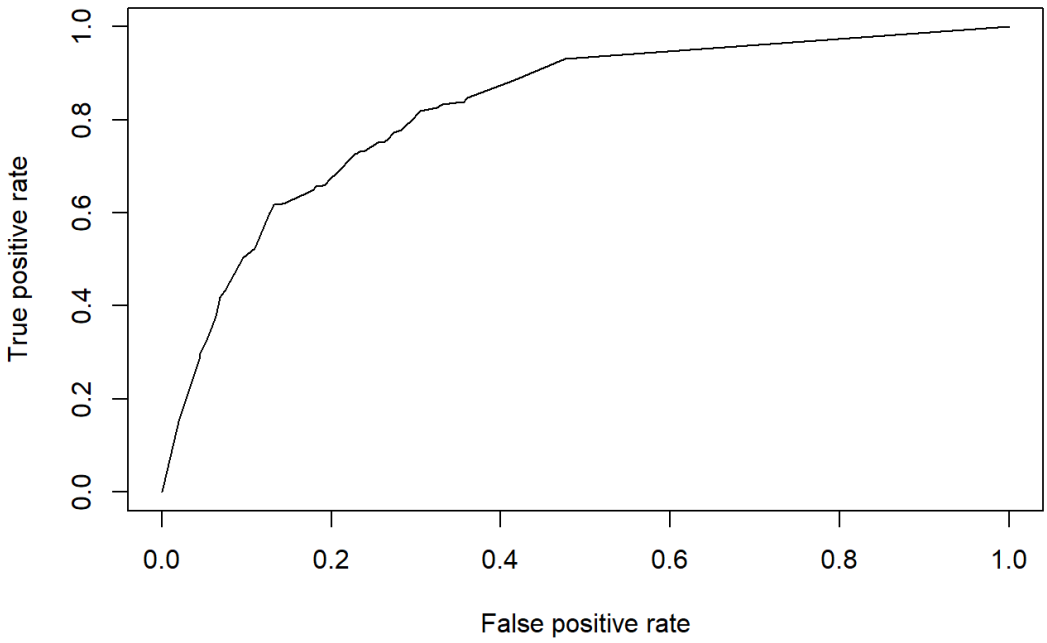
- acierto = 80.64824
- precision = 60.41667
- cobertura = 61.82171
- F1 = 61.11111

Paso 6. Métricas definitivas

Evaluamos la ROC

Evaluamos la ROC

```
#creamos el objeto prediction
rf2_prediction<-prediction(rf2_predict,test$TARGET)
#visualizamos la ROC
roc(rf2_prediction)
```



En la curva ROC, la línea diagonal que divide el gráfico en dos partes iguales indica que el modelo no tiene ninguna capacidad predictiva. Todo el área que está por encima de esa diagonal hasta la curva, indica la capacidad predictiva del modelo.

Métricas definitivas

Sacamos las métricas definitivas incluyendo el AUC

```
rf2_metricas<-cbind(rf2_metricas,AUC=round(auc(rf2_prediction),2)*100)
print(t(rf2_metricas))
```

```
##               [,1]
## umbral     0.40000
## acierto   80.64824
## precision 60.41667
## cobertura 61.82171
## F1        61.11111
## AUC       82.00000
```

Obtenemos las métricas definitivas añadiendo la métrica AUC, que indica el porcentaje de predicción del modelo, un 82%, lo que indica que es un buen modelo.