# Árboles de decisión y Random Forest. German Credit Data

**Adolfo Sánchez Burón**



# Algoritmos empleados

En este caso se vana a emplear dos algoritmos para comprobar cuál de ellos tiene unas métricas superiores para predecir la probilidad de impago en un crédito bancario.

Para un breve resumen de ambas técnicas se puede ver en:

- Árboles de decisión (https://www.ml2projects.com/post/churn-ardec) y
- Random forest (https://www.ml2projects.com/post/churn-randomf)

# Características del caso

El caso empleado en este análisis es el 'German Credit Data', que puede descargarse el dataset original desde [UCI](https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data) (https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)). 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, que puede descargarse de GitHub (https://github.com/AdSan-R/MachineLearning_R/tree/main/dataset).

El objetivo del caso es predecir la probabilidad de que un determinado cliente puede incluir un crédito bancario. La explicación de esta conducta estará basada en toda una serie de variables predictoras que se explicarán posterormente.

# 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. Preparación de la modelización

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

4. Modelización con Random Forest

Por motivos didácticos, se dividirá la modelización de los dos algoritmos en una sucesión de pasos.

---

# 1. Entorno

## 1.1. Instalar librerías

```r
library(dplyr)       # Manipulación de datos
library(knitr)       # Para formato de tablas
library(ggplot2)     # Gráficos
library(Information) # Exploración de datos con teoría del información
library(ROCR)        # Rendimiento del modelo y curva ROC
library(lattice)     # Tratamiento de gráficos
library(caret)       # División de muestra, Clasificación y regresión
library(rpart)       # Crear arboles de decisión
library(rpart.plot)  # Gráfico del árbol
library(randomForest)# Modelización mediante random forest
library(DataExplorer)# Análisis descriptivo con gráficos
```

```r
options(scipen=999)
#Desactiva la notación científica
```

## 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/MachineLearning_R/tree/main/dataset).

```r
df <- read.csv("CreditBank")
```

---

# 2. Análisis descriptivo

## 2.1. Análisis inicial

```r
head(df) #ver los primeros 6 casos
```

```
##   X chk_ac_status_1 credit_history_3 duration_month_2 savings_ac_bond_6
## 1 1             A11           04.A34            00-06               A65
## 2 2             A12       03.A32.A33              42+               A61
## 3 3             A14           04.A34            06-12               A61
## 4 4             A11       03.A32.A33            36-42               A61
## 5 5             A11       03.A32.A33            12-24               A61
## 6 6             A14       03.A32.A33            30-36               A65
##   purpose_4 property_type_12 age_in_yrs_13 credit_amount_5 p_employment_since_7
## 1       A43             A121           60+          0-1400                  A75
## 2       A43             A121          0-25           5500+                  A73
## 3       A46             A121         45-50       1400-2500                  A74
## 4       A42             A122         40-45           5500+                  A74
## 5       A40             A124         50-60       4500-5500                  A73
## 6       A46             A124         30-35           5500+                  A73
##   housing_type_15 other_instalment_type_14 personal_status_9 foreign_worker_20
## 1            A152                    A143               A93              A201
## 2            A152                    A143               A92              A201
## 3            A152                    A143               A93              A201
## 4            A153                    A143               A93              A201
## 5            A153                    A143               A93              A201
## 6            A153                    A143               A93              A201
##   other_debtors_or_grantors_10 instalment_pct_8 good_bad_21
## 1                         A101               4        Good
## 2                         A101               2         Bad
## 3                         A101               2        Good
## 4                         A103               2        Good
## 5                         A101               3         Bad
## 6                         A101               2        Good
```

# 2.2. Tipología de datos

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

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ X                           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ chk_ac_status_1             : chr  "A11" "A12" "A14" "A11" ...
##  $ credit_history_3            : chr  "04.A34" "03.A32.A33" "04.A34" "03.A32.A33"
...
##  $ duration_month_2            : chr  "00-06" "42+" "06-12" "36-42" ...
##  $ savings_ac_bond_6           : chr  "A65" "A61" "A61" "A61" ...
##  $ purpose_4                   : chr  "A43" "A43" "A46" "A42" ...
##  $ property_type_12            : chr  "A121" "A121" "A121" "A122" ...
##  $ age_in_yrs_13               : chr  "60+" "0-25" "45-50" "40-45" ...
##  $ credit_amount_5             : chr  "0-1400" "5500+" "1400-2500" "5500+" ...
##  $ p_employment_since_7        : chr  "A75" "A73" "A74" "A74" ...
##  $ housing_type_15             : chr  "A152" "A152" "A152" "A153" ...
##  $ other_instalment_type_14    : chr  "A143" "A143" "A143" "A143" ...
##  $ personal_status_9           : chr  "A93" "A92" "A93" "A93" ...
##  $ foreign_worker_20           : chr  "A201" "A201" "A201" "A201" ...
##  $ other_debtors_or_grantors_10: chr  "A101" "A101" "A101" "A103" ...
##  $ instalment_pct_8            : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ good_bad_21                 : chr  "Good" "Bad" "Good" "Good" ...
```

Puede observarse que todas son “chr”, esto es, “character”, por tanto, vamos a pasarlas a Factor. Además, instalment_pct_8 aparece como “entero” cuando es factor. También la transformamos.

```
df <- mutate_if(df, is.character, as.factor) #identifica todas las character y las pas
a a factores
#Sacamos la esructura

df$instalment_pct_8 <- as.factor(df$instalment_pct_8 )

str(df)
```

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ X                          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ chk_ac_status_1            : Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1
4 4 2 4 2 ...
##  $ credit_history_3           : Factor w/ 4 levels "01.A30","02.A31",..: 4 3 4 3 3
3 3 3 3 4 ...
##  $ duration_month_2           : Factor w/ 7 levels "00-06","06-12",..: 1 7 2 6 3 5
3 5 2 4 ...
##  $ savings_ac_bond_6          : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1
5 3 1 4 1 ...
##  $ purpose_4                  : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4
1 8 4 2 5 1 ...
##  $ property_type_12           : Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2
3 1 3 ...
##  $ age_in_yrs_13              : Factor w/ 8 levels "0-25","25-30",..: 8 1 6 5 7 3
7 3 8 2 ...
##  $ credit_amount_5            : Factor w/ 6 levels "0-1400","1400-2500",..: 1 6 2
6 5 6 3 6 3 5 ...
##  $ p_employment_since_7       : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3
3 5 3 4 1 ...
##  $ housing_type_15            : Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2
1 2 2 ...
##  $ other_instalment_type_14   : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3
3 3 3 ...
##  $ personal_status_9          : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3
3 3 3 1 4 ...
##  $ foreign_worker_20          : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1
1 1 ...
##  $ other_debtors_or_grantors_10: Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1
1 1 1 ...
##  $ instalment_pct_8           : Factor w/ 4 levels "1","2","3","4": 4 2 2 2 3 2 3
2 2 4 ...
##  $ good_bad_21                : Factor w/ 2 levels "Bad","Good": 2 1 2 2 1 2 2 2 2
1 ...
```

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

Para los siguientes análisis:

1. Eliminamos a la variable X (número de cliente) del df.
2. Renombramos la variable good_bad_21 como “target”.

```
#Creamos la variable "target"
df$target <- as.factor(df$good_bad_21)

#Eliminamos la variable "good_bad_21" y eliminamos x
df <- select(df,-good_bad_21, -X)

str(df)
```

```
## 'data.frame':    1000 obs. of  16 variables:
##  $ chk_ac_status_1            : Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1
4 4 2 4 2 ...
##  $ credit_history_3           : Factor w/ 4 levels "01.A30","02.A31",..: 4 3 4 3 3
3 3 3 3 4 ...
##  $ duration_month_2           : Factor w/ 7 levels "00-06","06-12",..: 1 7 2 6 3 5
3 5 2 4 ...
##  $ savings_ac_bond_6          : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1
5 3 1 4 1 ...
##  $ purpose_4                  : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4
1 8 4 2 5 1 ...
##  $ property_type_12           : Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2
3 1 3 ...
##  $ age_in_yrs_13              : Factor w/ 8 levels "0-25","25-30",..: 8 1 6 5 7 3
7 3 8 2 ...
##  $ credit_amount_5            : Factor w/ 6 levels "0-1400","1400-2500",..: 1 6 2
6 5 6 3 6 3 5 ...
##  $ p_employment_since_7       : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3
3 5 3 4 1 ...
##  $ housing_type_15            : Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2
1 2 2 ...
##  $ other_instalment_type_14   : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3
3 3 3 ...
##  $ personal_status_9          : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3
3 3 3 1 4 ...
##  $ foreign_worker_20          : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1
1 1 ...
##  $ other_debtors_or_grantors_10: Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1
1 1 1 ...
##  $ instalment_pct_8           : Factor w/ 4 levels "1","2","3","4": 4 2 2 2 3 2 3
2 2 4 ...
##  $ target                     : Factor w/ 2 levels "Bad","Good": 2 1 2 2 1 2 2 2 2
1 ...
```

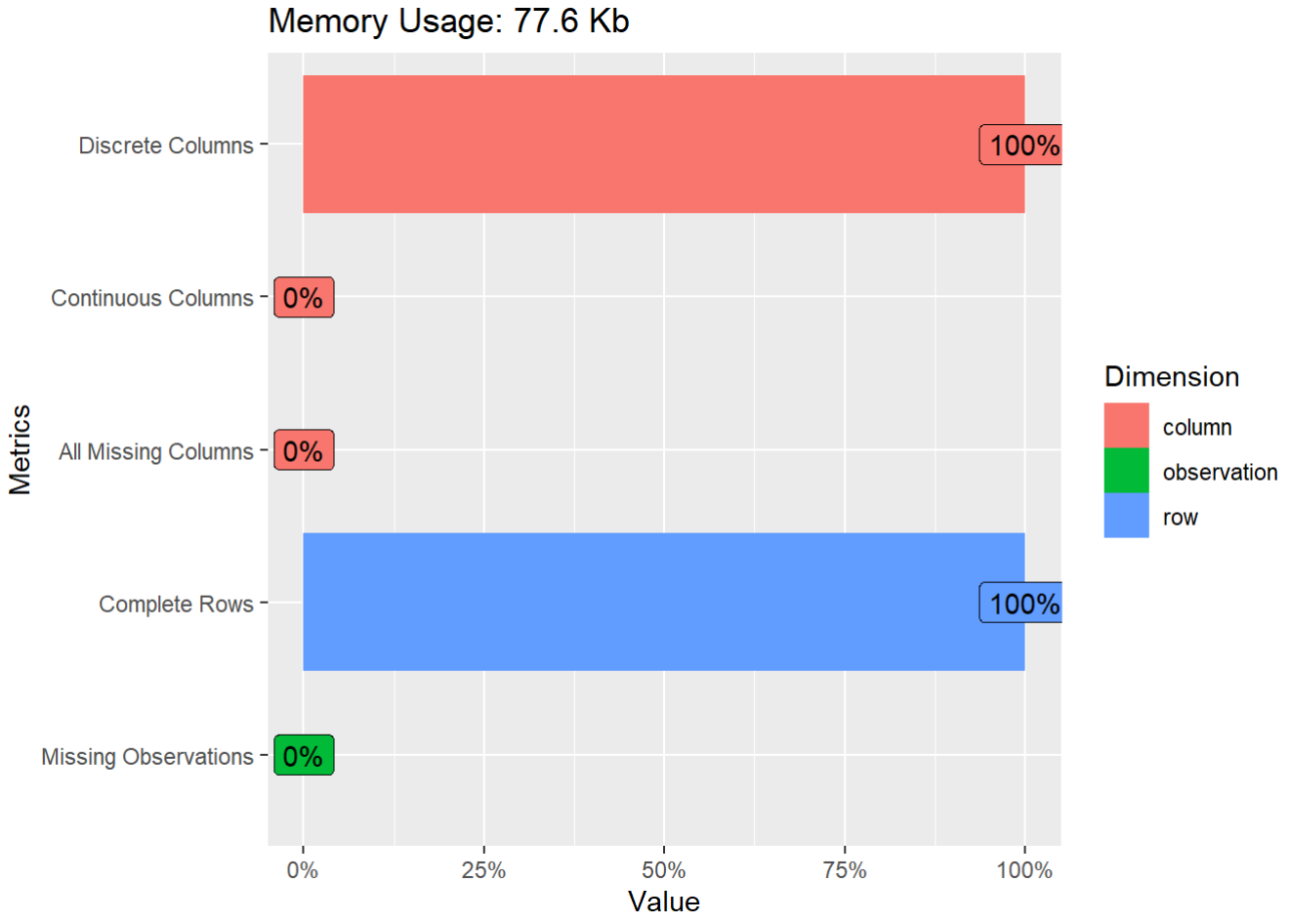
```
lapply(df,summary) #mostrar la distribución de frecuencias en cada categoría de todas
 las variables
```

```
## $chk_ac_status_1
## A11 A12 A13 A14
## 274 269  63 394
##
## $credit_history_3
##     01.A30     02.A31 03.A32.A33     04.A34
##         40         49        618        293
##
## $duration_month_2
## 00-06 06-12 12-24 24-30 30-36 36-42   42+
##    82   277   411    57    86    17    70
##
## $savings_ac_bond_6
## A61 A62 A63 A64 A65
## 603 103  63  48 183
##
## $purpose_4
##  A40  A41 A410  A42  A43  A44  A45  A46  A48  A49
##  234  103   12  181  280   12   22   50    9   97
##
## $property_type_12
## A121 A122 A123 A124
##  282  232  332  154
##
## $age_in_yrs_13
##  0-25 25-30 30-35 35-40 40-45 45-50 50-60   60+
##   190   221   177   138    88    73    68    45
##
## $credit_amount_5
##    0-1400 1400-2500 2500-3500 3500-4500 4500-5500     5500+
##       267        270        149         98         48       168
##
## $p_employment_since_7
## A71 A72 A73 A74 A75
##  62 172 339 174 253
##
## $housing_type_15
## A151 A152 A153
##  179  713  108
##
## $other_instalment_type_14
## A141 A142 A143
##  139   47  814
##
## $personal_status_9
## A91 A92 A93 A94
##  50 310 548  92
##
## $foreign_worker_20
## A201 A202
##  963   37
```

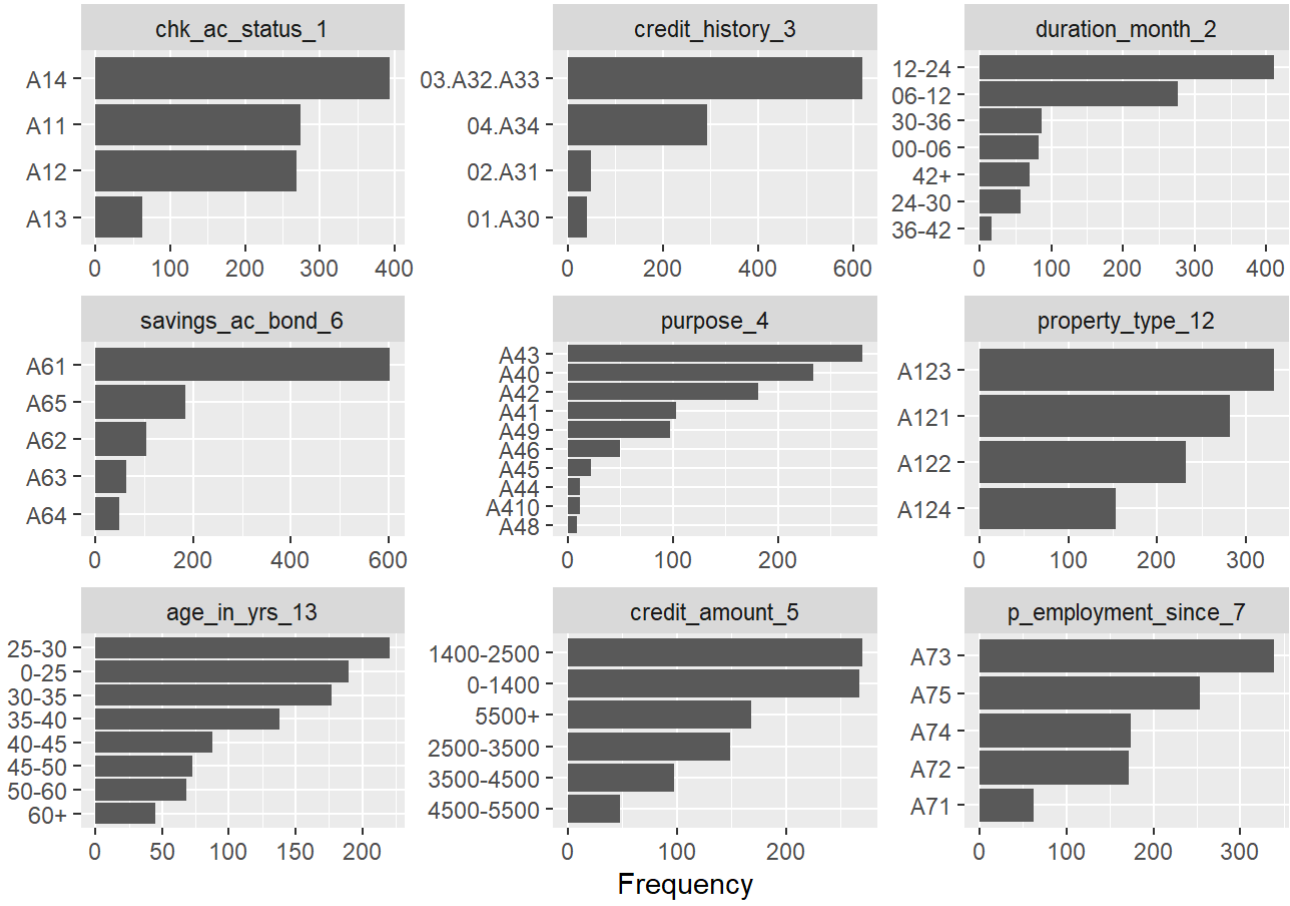
```
##
## $other_debtors_or_grantors_10
## A101 A102 A103
##  907   41   52
##
## $instalment_pct_8
##   1   2   3   4
## 136 231 157 476
##
## $target
##  Bad Good
##  300  700
```

# 2.3. Análisis descriptivo (gráficos)

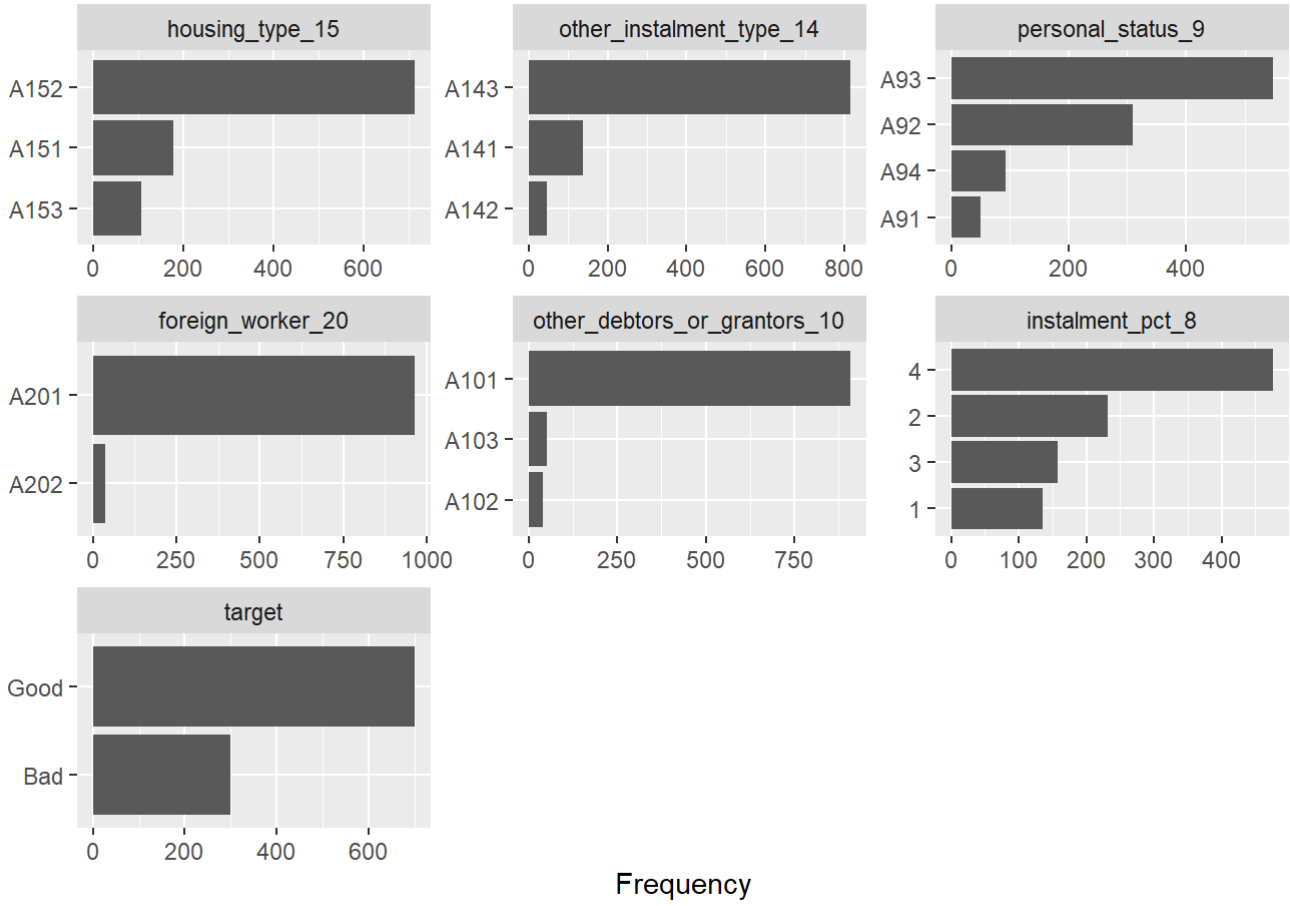
```
plot_intro(df) #gráfico para observar la distribución de variables y los casos missing por columnas, observaciones y filas
```




```
plot_bar(df) #gráfico para observar la distribución de frecuencias en variables categóricas
```








De las gráficas anteriores se puede observar:

1. La distribución de la target es adecuada y no necesita trabajo posterior.

2. Se puede observar que varias variables tienen algunas categorías con poca frecuencia. Sería oportuno analizar la conveniencia de recodificar en categorías con mayor representación.

# 3. Preparación para modelización

## Particiones de training (70%) y test (30%)

Se segmenta la muestra en dos partes (train y test) empleando el programa Caret.

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.

```
set.seed(100)  # Para reproducir los mismos resultados
partition <- createDataPartition(y = df$target, p = 0.7, list = FALSE)
train <- df[partition,]
test <- df[-partition,]
```

# 4. Modelización con Árboles de decisión

## Paso 1. Primer modelo

```
mod_dt<-rpart(target~., train,
          method = 'class',
          parms = list(  split = "information"),
          control = rpart.control(cp = 0.00001))
#Lanzamos la runción "rpart2" para hallar el AD, que incluye>:
#El modelo a entrenar: TARGET ~ InternetService + Contract + PaymentMethod + tenure_DI
SC + MonthlyCharges_DISC + TotalCharges_DISC
#El df: "train"
#method = 'class'. Método de clasificación
#split: criterio de corte. Se tienen dos opciones: 'information' o 'gini'
#cp: criterio de complejidad. Se comienza emplenado un cp muy pequeño (esto es, dejamo
s que el árbol sea muy profundo), y posteriormente, se irán añadiendo condiciones más
 restrictivas (siendo el árbol menos profundo y más interpretable).

# Imprimimos los resultados
printcp(mod_dt)
```

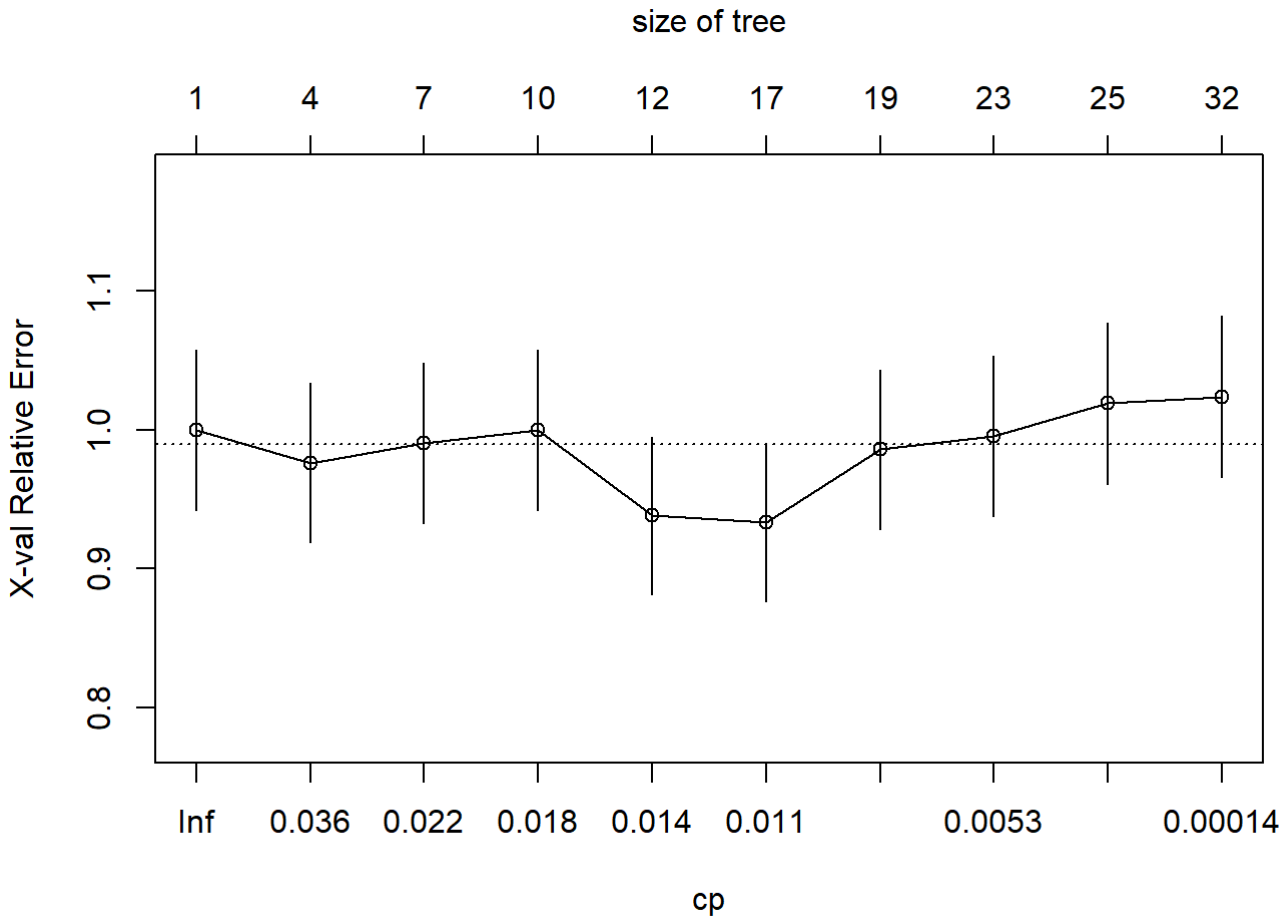
```
##
## Classification tree:
## rpart(formula = target ~ ., data = train, method = "class", parms = list(split = "i
nformation"),
##     control = rpart.control(cp = 0.00001))
##
## Variables actually used in tree construction:
##  [1] age_in_yrs_13               chk_ac_status_1
##  [3] credit_amount_5             credit_history_3
##  [5] duration_month_2            housing_type_15
##  [7] instalment_pct_8            other_debtors_or_grantors_10
##  [9] other_instalment_type_14    p_employment_since_7
## [11] property_type_12            purpose_4
## [13] savings_ac_bond_6
##
## Root node error: 210/700 = 0.3
##
## n= 700
##
##           CP nsplit rel error  xerror     xstd
## 1  0.0523810      0   1.00000 1.00000 0.057735
## 2  0.0253968      3   0.80476 0.97619 0.057334
## 3  0.0190476      6   0.72857 0.99048 0.057577
## 4  0.0166667      9   0.66667 1.00000 0.057735
## 5  0.0119048     11   0.63333 0.93810 0.056656
## 6  0.0095238     16   0.57143 0.93333 0.056569
## 7  0.0059524     18   0.55238 0.98571 0.057496
## 8  0.0047619     22   0.52857 0.99524 0.057656
## 9  0.0019048     24   0.51905 1.01905 0.058044
## 10 0.0000100     31   0.50476 1.02381 0.058119
```

Interpretación de la tabla:

a. Observamos la columna xerror (error de validación cruzada), la cual se va reduciendo hasta que empieza a crecer.

b. Tomamos el datos de CP (criterior de complejidad) de ese nivel para incluirlo posteriormente al modelo.

```
plotcp(mod_dt)
```




Se observa que el error de validación cruzada (xerror = 0.90476) minimiza en un cp = 0.0095238 de complejidad.Por tanto, los dos siguientes pasos son:

a. Generamos un nuevo árbol con ese parámetro, esto es más restrictivo.

b. Además, le añadimos otro parámetro de restricción: el número de niveles, que no tenga mas de 7 niveles de profundidad (maxdepth = 7), aunque no sabemos dónde va a parar antes, por el cp o por el número de niveles.

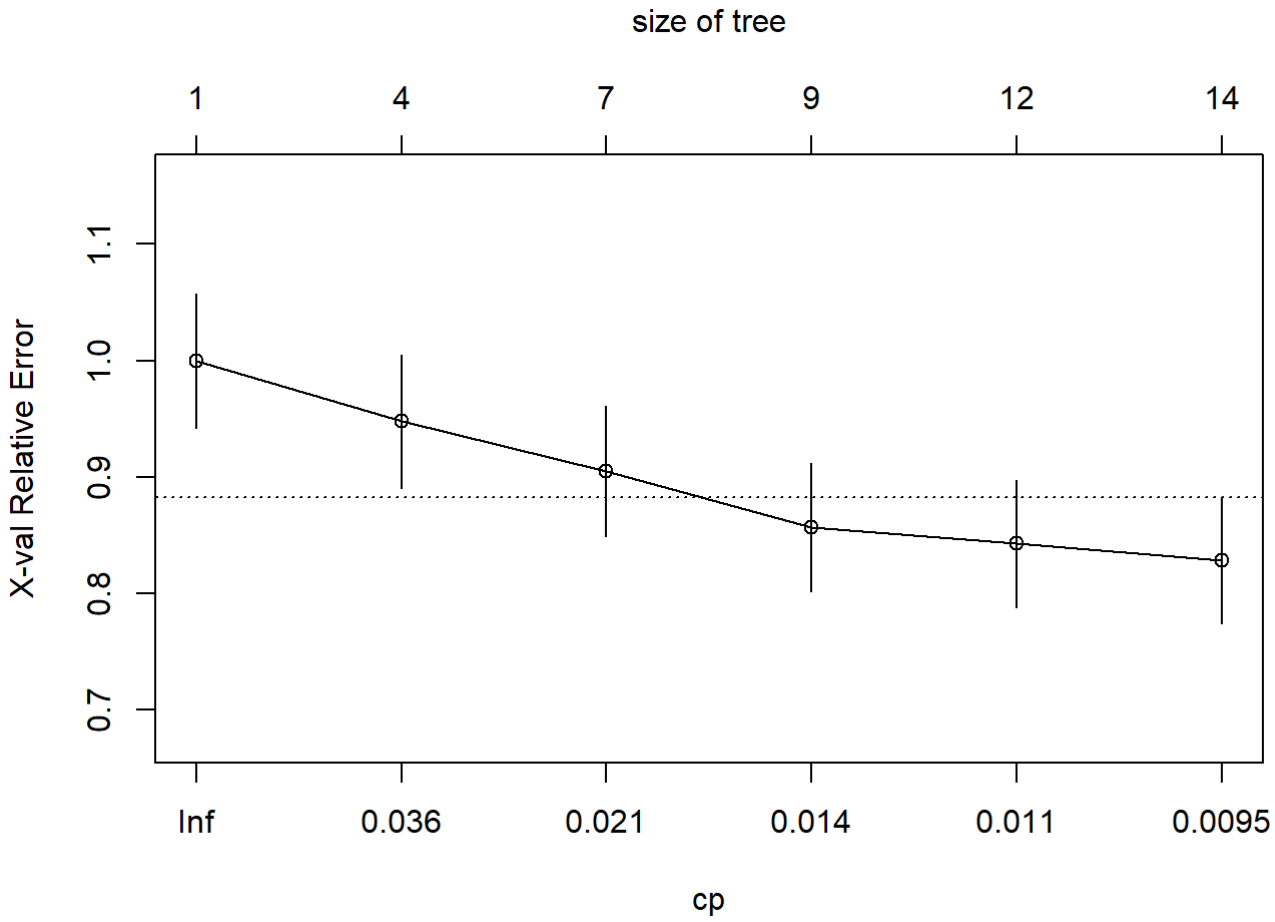
# Paso 2. Segundo modelo

```
mod_dt2<-rpart(target~., train,
          method = 'class',
          parms = list(  split = "information"),
          control = rpart.control(cp = 0.0095,
                                  maxdepth = 7))

printcp(mod_dt2)
```

```
##
## Classification tree:
## rpart(formula = target ~ ., data = train, method = "class", parms = list(split = "i
nformation"),
##     control = rpart.control(cp = 0.0095, maxdepth = 7))
##
## Variables actually used in tree construction:
## [1] age_in_yrs_13        chk_ac_status_1      credit_amount_5
## [4] credit_history_3     duration_month_2     instalment_pct_8
## [7] p_employment_since_7 purpose_4
##
## Root node error: 210/700 = 0.3
##
## n= 700
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0523810      0   1.00000 1.00000 0.057735
## 2 0.0253968      3   0.80476 0.94762 0.056830
## 3 0.0166667      6   0.72857 0.90476 0.056027
## 4 0.0119048      8   0.69524 0.85714 0.055064
## 5 0.0095238     11   0.65714 0.84286 0.054761
## 6 0.0095000     13   0.63810 0.82857 0.054450
```

Se observa que xerror no asciende y que se corta en el nivel de profundidad de 6.

```
#Observamos gráficamente el resultado anterior
plotcp(mod_dt2)
```



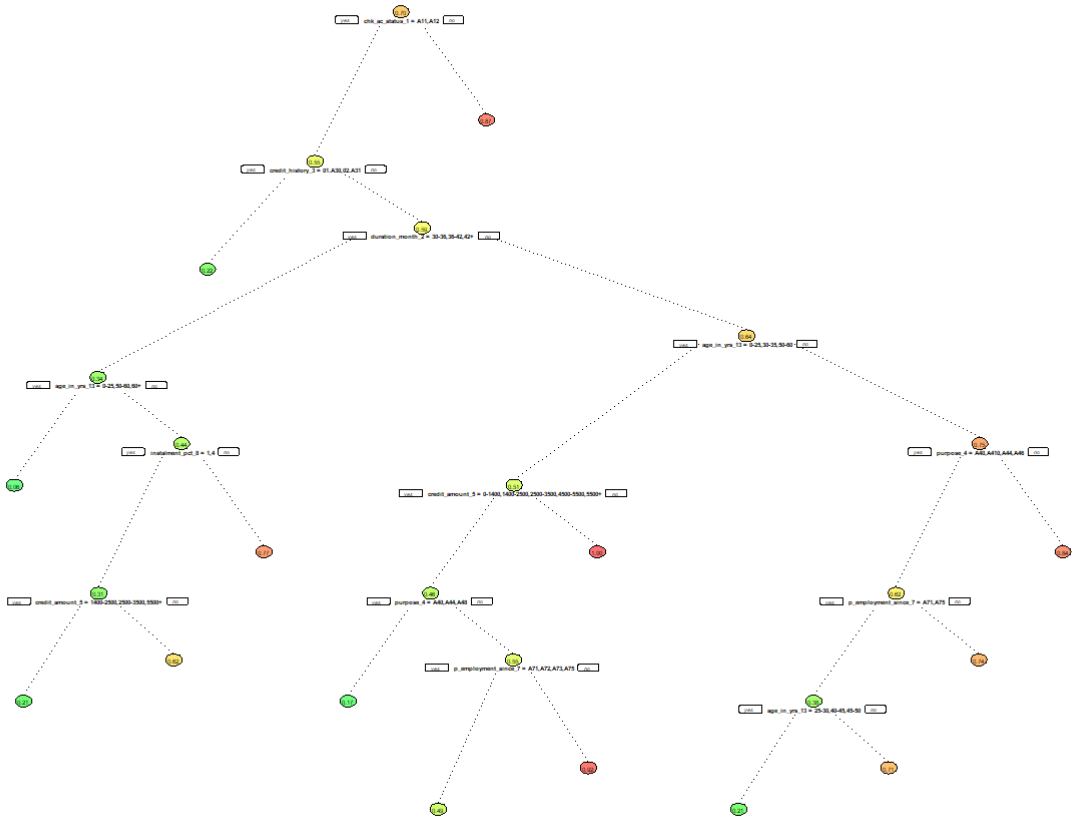

Se observa que xerror no asciende. Parece que el árbol es bastante estable, por lo que pasamos a interpretarlo.

# Paso 3. Interpretación del árbol

Seguimos tres pasos:

### A. Creación del gráfico del árbol

```
rpart.plot(mod_dt2,type=2,extra = 7, under = TRUE,under.cex = 0.7,fallen.leaves=F,gap
 = 0,cex=0.2, yesno = 2,box.palette = "GnYlRd",branch.lty = 3)
```



En muchas ocasiones el árbol no es fácilmente visible si no se amplía considerablemente, lo cual podemos hacerlo exportándolo a pdf o en un power point. Si observamos la parte superior del árbol podremos ver estos nodos y divisiones:

- La variable con mayor capacidad discriminativa es chk_ac_status_1 = A11,A12 (el 70% de clientes está en este nodo).
- Los que No tienen categoría A11 y A12 tienen una probabilidad de no devolver el crédito del 87%. Este es un nodo terminal ya que no tiene más divisiones. Desde un punto de vista de negocio viene a decir que son clientes bastante poco proclives al incumplimiento.
- Los que Sí tienen esta categoría, si además tienen credit_history_3 = 01.A30,02.A31, tendrán una probabilidad de cumplir con el pago del 22%.

**B. Reglas del árbol**

Sacamos las reglas de división del árbol, necesarias para hacer una implantación de negocio posterior

```
rpart.rules(mod_dt2,style = 'tall',cover = T)
```

```
## target is 0.06 with cover 2% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 30-36 or 36-42 or 42+
##     age_in_yrs_13 is 0-25 or 50-60 or 60+
##
## target is 0.17 with cover 3% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
##     age_in_yrs_13 is 0-25 or 30-35 or 50-60
##     purpose_4 is A40 or A44 or A48
##     credit_amount_5 is 0-1400 or 1400-2500 or 2500-3500 or 4500-5500 or 5500+
##
## target is 0.21 with cover 3% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 30-36 or 36-42 or 42+
##     age_in_yrs_13 is 25-30 or 30-35 or 35-40 or 40-45 or 45-50
##     credit_amount_5 is 1400-2500 or 2500-3500 or 5500+
##     instalment_pct_8 is 1 or 4
##
## target is 0.21 with cover 2% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
##     age_in_yrs_13 is 25-30 or 40-45 or 45-50
##     purpose_4 is A40 or A410 or A44 or A46
##     p_employment_since_7 is A71 or A75
##
## target is 0.22 with cover 6% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 01.A30 or 02.A31
##
## target is 0.49 with cover 11% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
##     age_in_yrs_13 is 0-25 or 30-35 or 50-60
##     purpose_4 is A41 or A410 or A42 or A43 or A45 or A46 or A49
##     credit_amount_5 is 0-1400 or 1400-2500 or 2500-3500 or 4500-5500 or 5500+
##     p_employment_since_7 is A71 or A72 or A73 or A75
##
## target is 0.62 with cover 1% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 30-36 or 36-42 or 42+
##     age_in_yrs_13 is 25-30 or 30-35 or 35-40 or 40-45 or 45-50
##     credit_amount_5 is 3500-4500 or 4500-5500
##     instalment_pct_8 is 1 or 4
##
```

```
## target is 0.71 with cover 1% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
##     age_in_yrs_13 is 35-40 or 60+
##     purpose_4 is A40 or A410 or A44 or A46
##     p_employment_since_7 is A71 or A75
##
## target is 0.74 with cover 6% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
##     age_in_yrs_13 is 25-30 or 35-40 or 40-45 or 45-50 or 60+
##     purpose_4 is A40 or A410 or A44 or A46
##     p_employment_since_7 is A72 or A73 or A74
##
## target is 0.77 with cover 2% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 30-36 or 36-42 or 42+
##     age_in_yrs_13 is 25-30 or 30-35 or 35-40 or 40-45 or 45-50
##     instalment_pct_8 is 2 or 3
##
## target is 0.84 with cover 13% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
##     age_in_yrs_13 is 25-30 or 35-40 or 40-45 or 45-50 or 60+
##     purpose_4 is A41 or A42 or A43 or A45 or A48 or A49
##
## target is 0.87 with cover 46% when
##     chk_ac_status_1 is A13 or A14
##
## target is 0.92 with cover 2% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
##     age_in_yrs_13 is 0-25 or 30-35 or 50-60
##     purpose_4 is A41 or A410 or A42 or A43 or A45 or A46 or A49
##     credit_amount_5 is 0-1400 or 1400-2500 or 2500-3500 or 4500-5500 or 5500+
##     p_employment_since_7 is A74
##
## target is 1.00 with cover 1% when
##     chk_ac_status_1 is A11 or A12
##     credit_history_3 is 03.A32.A33 or 04.A34
##     duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
##     age_in_yrs_13 is 0-25 or 30-35 or 50-60
##     credit_amount_5 is 3500-4500
```

```
#style sirve para que la salida sea mas legible y cover añade el % de casos e los que
 aplica la regla
```

Interpretación:

Vemos el primero:

```
target is 0.06 with cover 2% when
    chk_ac_status_1 is A11 or A12
    credit_history_3 is 03.A32.A33 or 04.A34
    duration_month_2 is 30-36 or 36-42 or 42+
    age_in_yrs_13 is 0-25 or 50-60 or 60+
```

- Grupo con scoring (probabilidad de “Bad credit”) es del 6%.
- Compuesto por el 2% de los clientes
- Regla:

a. tienen chk_ac_status_1 en las categorías A11 o A12

b. tienen credit_history_3 03.A32.A33 o 04.A34

c. tienen en duration_month_2 en las categorías 30-36, 36-42 y 42+

d. tienen age_in_yrs_13 en las categorías 0-25, 50-60 y 60+

- Interpretación de negocio: en principio, a estos clientes se les puede conceder el crédito ya que su probilidad de incumplimiento es muy reducido.

Vemos el último:

```
target is 1.00 with cover 1% when
    chk_ac_status_1 is A11 or A12
    credit_history_3 is 03.A32.A33 or 04.A34
    duration_month_2 is 00-06 or 06-12 or 12-24 or 24-30
    age_in_yrs_13 is 0-25 or 30-35 or 50-60
    credit_amount_5 is 3500-4500
```

- Grupo con scoring (probabilidad de “Bad credit”) es del 100%.
- Compuesto por el 1% de los clientes
- Regla: tienen

    a. En chk_ac_status_1 están en las categorías A11 or A12

    b. En credit_history_3 están en las categorías 03.A32.A33 or 04.A34

    c. En duration_month_2 están en las categorías 00-06 or 06-12 or 12-24 or 24-30

    d. En age_in_yrs_13 is 0-25 or 30-35 or 50-60

    e. En credit_amount_5 is 3500-4500

- Interpretación de negocio: en principio, estos clientes no son nad fiables. No se les debe conceder el crédito.

**C. Introducir datos en un df**

Llevamos el nodo final de cada cliente a un data.frame para poder hacer una explotacion posterior (por ejemplo para saber las características de cada nodo, como edad, etc.)

```
#Se usa el predict específico de rpart y con el parámetro nn
ar2_numnodos<-rpart.predict(mod_dt2, test, nn = T)

head(ar2_numnodos)
```

```
##          Bad      Good  nn
## 4  0.2307692 0.7692308  43
## 7  0.1261538 0.8738462   3
## 9  0.1261538 0.8738462   3
## 13 0.5131579 0.4868421 178
## 14 0.8333333 0.1666667  88
## 23 0.2564103 0.7435897  93
```

INTERPRETACIÓN

El cliente 4 tiene una probabilidad de no devolver el crédito del 23,08% y cae en el nodo 43

El cliente 7 tiene una probabilidad de churn del 12,62% y cae en el nodo 3.

# Paso 4. Predict

Vamos a calcular los scorings y evaluar el modelo

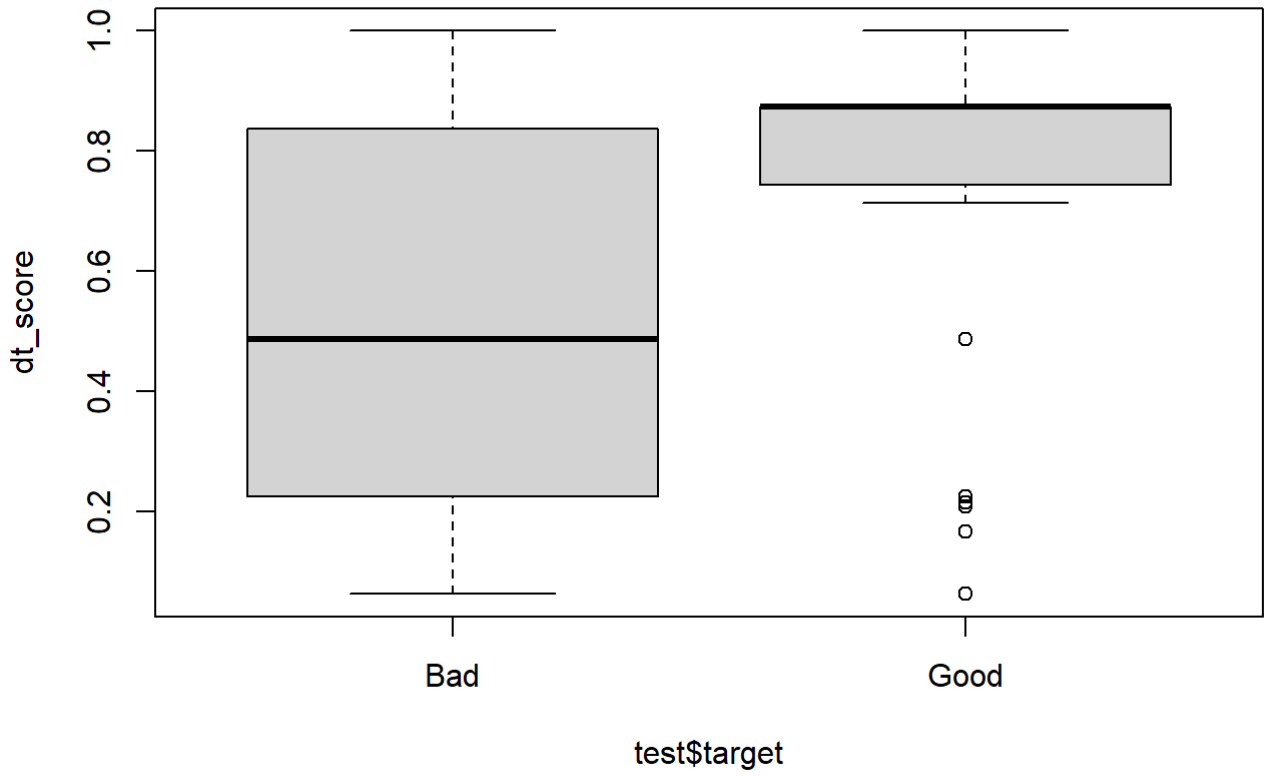
```
dt_score<-predict(mod_dt2,test,type = 'prob')[,2]
#Samos el predict para el modelo "ar2":
#Con el data frame "test"
#Se utiliza "type=prob" (que reporta la probabilidad para cada caso)
#Se le incluye [,2], es decir, la segunda columna, ya que interesa la probabilidad de
 que sea 1 (que haga churn)

#Sacamos los 6 primeros casos
head(dt_score)
```

```
##         4         7         9        13        14        23
## 0.7692308 0.8738462 0.8738462 0.4868421 0.1666667 0.7435897
```

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 53,61%. El segundo de 7,31%, etc.

Vemos la capacidad de discriminación entre los dos niveles de la TARGET

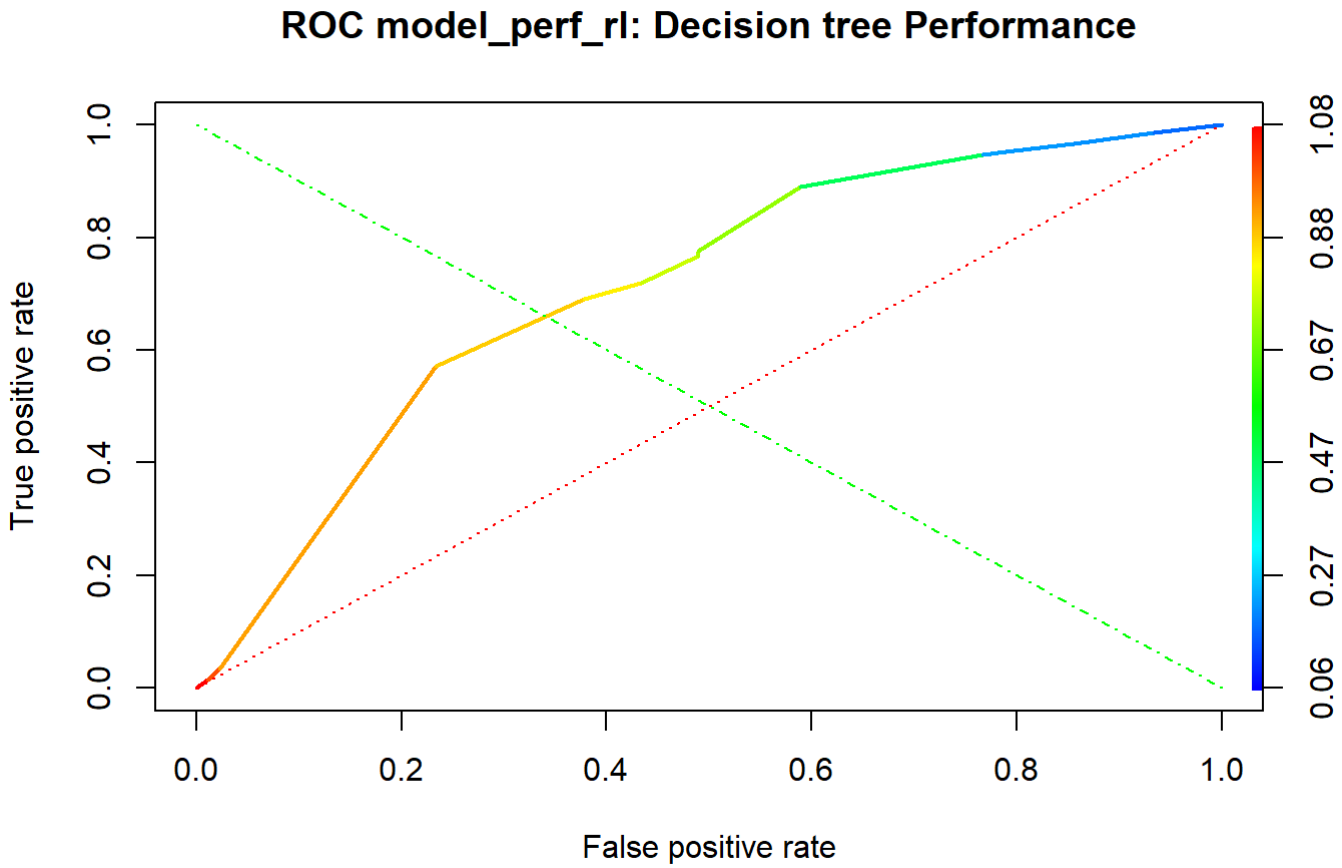
```
plot(dt_score ~ test$target)
```




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 5. Curva ROC

```
pred_dt <- prediction(dt_score, test$target)
perf_dt <- performance(pred_dt,"tpr","fpr")
#library(ROCR)
plot(perf_dt, lwd=2, colorize=TRUE, main="ROC model_perf_rl: Decision tree Performanc
e")
lines(x=c(0, 1), y=c(0, 1), col="red", lwd=1, lty=3);
lines(x=c(1, 0), y=c(0, 1), col="green", lwd=1, lty=4)
```




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.

# Paso 6. Umbrales y matriz de confusión

A continuación se probarán distintos umbrales para maximizar el F1 al transformar la probabilidad obtenida en otra dicotómica (Good y Bad credit).

En otros proyectos hemos empleado funciones. En este caso lo haremos una por una para entender mejor le proceso. Lo que vamos cambiando es el umbral ("treshold"), observando en cada caso cómo varían las mátricas de la matriz de confusión (accuracy, sensitividad, precision y F1).

```
dt_score2<-predict(mod_dt2,test,type = 'prob')[,2]

score2 <- ifelse(dt_score2 > 0.20, "Good", "Bad")
MC <- table(test$target, score2)
Acc2 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen2 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr2 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F12 <- 2*Pr2*Sen2/(Pr2+Sen2)

score3 <- ifelse(dt_score2 > 0.30, "Good", "Bad")
MC <- table(test$target, score3)
Acc3 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen3 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr3 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F13 <- 2*Pr3*Sen3/(Pr3+Sen3)

score4 <- ifelse(dt_score2 > 0.40, "Good", "Bad")
MC <- table(test$target, score4)
Acc4 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen4 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr4 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F14 <- 2*Pr4*Sen4/(Pr4+Sen4)

score5 <- ifelse(dt_score2 > 0.50, "Good", "Bad")
MC <- table(test$target, score5)
Acc5 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen5 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr5 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F15 <- 2*Pr5*Sen5/(Pr5+Sen5)

score6 <- ifelse(dt_score2 > 0.60, "Good", "Bad")
MC <- table(test$target, score6)
Acc6 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen6 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr6 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F16 <- 2*Pr6*Sen6/(Pr6+Sen6)

score7 <- ifelse(dt_score2 > 0.70, "Good", "Bad")
MC <- table(test$target, score7)
Acc7 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen7 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr7 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F17 <- 2*Pr7*Sen7/(Pr7+Sen7)

score8 <- ifelse(dt_score2 > 0.80, "Good", "Bad")
MC <- table(test$target, score8)
Acc8 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen8 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr8 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F18 <- 2*Pr8*Sen8/(Pr8+Sen8)
```

```
#salida<-c(Acc2,Acc3,Acc4,Acc5,Acc6,Acc7,Acc8)
#salida
#salida<-c(Sen2,Sen3,Sen4,Sen5,Sen6,Sen7,Sen8)
#salida
#salida<-c(Pr2,Pr3,Pr4,Pr5,Pr6,Pr7,Pr8)
#salida
salida<-c(F12,F13,F14,F15,F16,F17,F18)
salida
```

```
## [1] 82.85714 83.11111 83.11111 78.17746 78.17746 78.17746 74.55013
```

Se puede observar que el límite donde se mazimiza la F1 es en 0,2, con un F1 = 81.25000

# Paso 7. Métricas definitivas

Sacamos las métricas definitivas incluyendo el AUC

```
score3 <- ifelse(dt_score2 > 0.3, "Good", "Bad")
MC <- table(test$target, score3)
dt_Acc <- round((MC[1,1] + MC[2,2]) / sum(MC) *100, 2)
dt_Sen <- round(MC[2,2] / (MC[2,2] + MC[1,2]) *100, 2)
dt_Pr <- round(MC[2,2] / (MC[2,2] + MC[2,1]) *100, 2)
dt_F1 <- round(2*dt_Pr*dt_Sen/(dt_Pr+dt_Sen), 2)

#KS & AUC
dt_KS <- round(max(attr(perf_dt,'y.values')[[1]]-attr(perf_dt,'x.values')[[1]])*100, 2
)
dt_AUROC <- round(performance(pred_dt, measure = "auc")@y.values[[1]]*100, 2)

cat("Acierto_ad: ",dt_Acc,"\tSensibilidad_ad: ", dt_Sen, "\tPrecision_ad:", dt_Pr, "\t
F1_ad:", dt_F1, "\tAUROC_ad: ",dt_AUROC,"\tKS_ad: ", dt_KS, "\n")
```

```
## Acierto_ad:  74.67   Sensibilidad_ad:  77.92     Precision_ad: 89.05     F1_ad: 83.
11    AUROC_ad:  70.97    KS_ad:  33.81
```

Obtenemos las métricas definitivas añadiendo la métrica AUC, que indica el porcentaje de predicción del modelo, un 70,97%, lo que indica que es un modelo relativamente bueno.

---

# 5. Modelización con Random Forest

# Paso 1. Primer modelo

```
mod_rf<-randomForest(target~., train,importance=T)
mod_rf
```

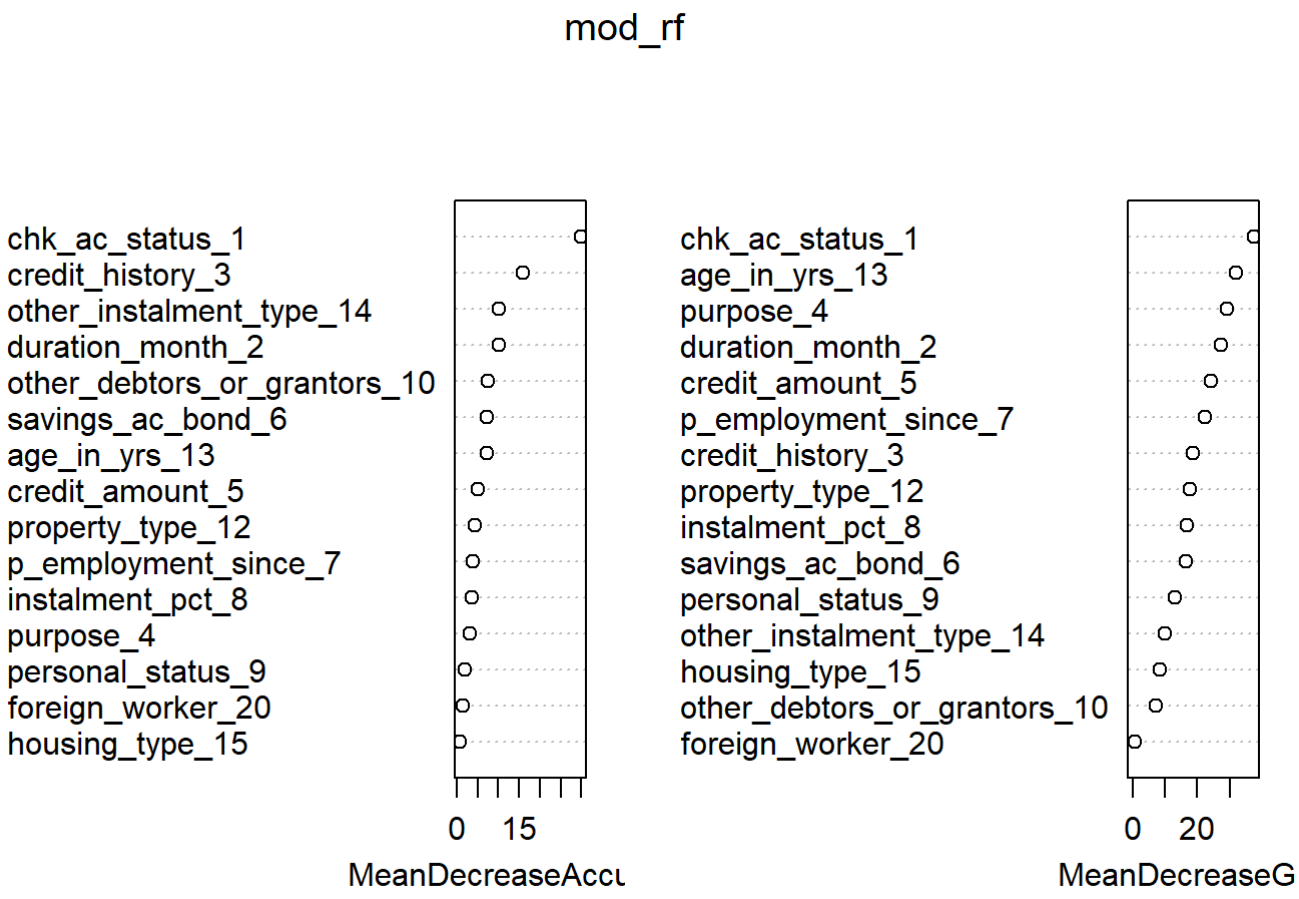
```
##
## Call:
##  randomForest(formula = target ~ ., data = train, importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 22.86%
## Confusion matrix:
##      Bad Good class.error
## Bad   96  114  0.54285714
## Good  46  444  0.09387755
```

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 = 3
- OOB estimate of error rate: 23%
- Además aporta una matriz de confusión.

## Visualización de importancia relativa de cada variable en el modelo

```
varImpPlot(mod_rf)
```



Para unir los resultados aportado por el Mean Decrease Accuracy y el Mean Decrease Gini vamos a emplear un método para unir las dos métricas, propuesto en el DS4B curso de Machine Learning Predictivo (https://www.datascience4business.com/08_mlc-salespage-b):

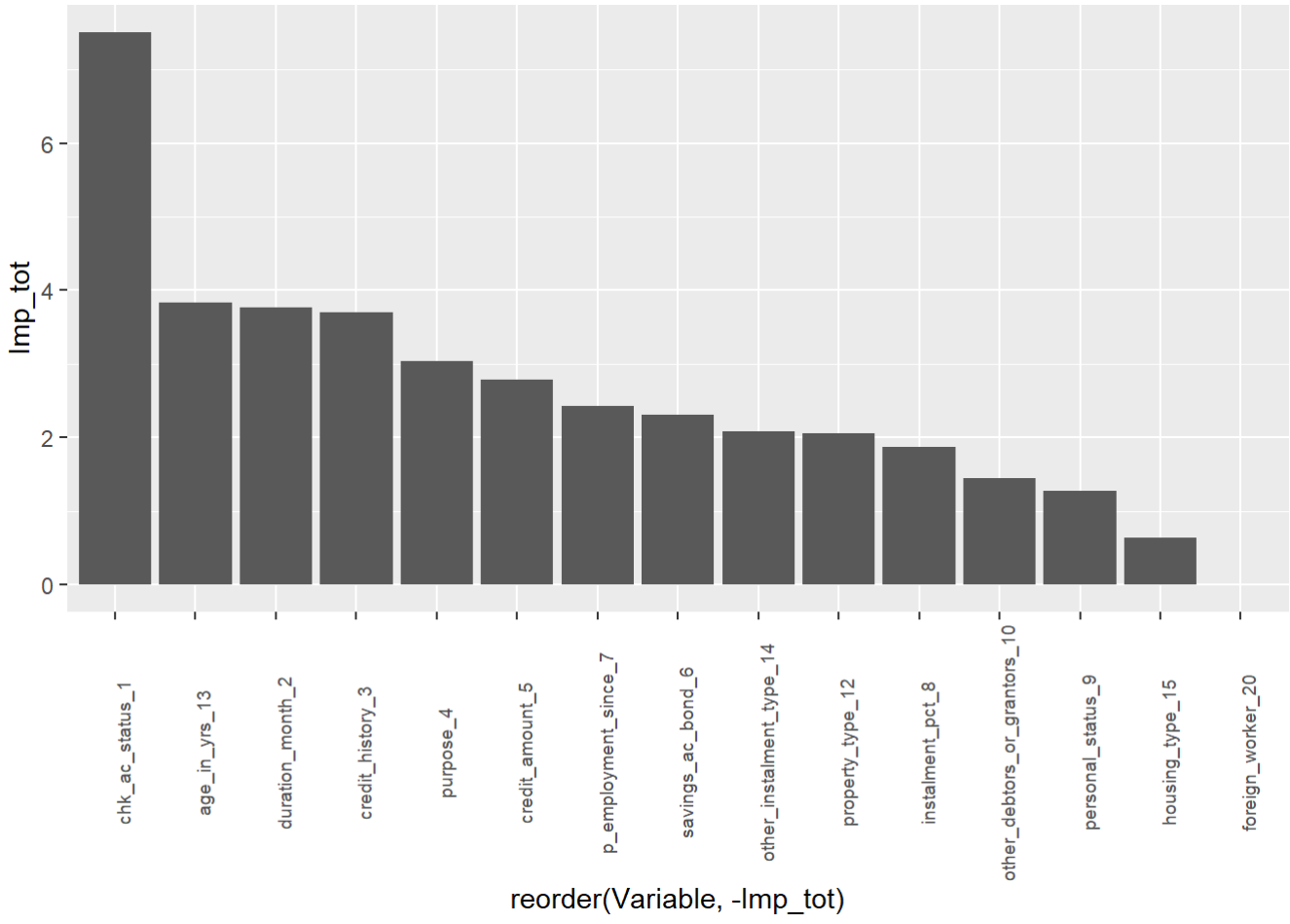
```
importancia <- importance(mod_rf)[,3:4]
importancia
```

```
##                              MeanDecreaseAccuracy MeanDecreaseGini
## chk_ac_status_1                         29.9593461       37.6466874
## credit_history_3                        15.8235507       18.7031636
## duration_month_2                        10.0417359       27.1909807
## savings_ac_bond_6                        7.2821187       16.3752492
## purpose_4                                3.2946432       29.1606798
## property_type_12                         4.4928695       17.6472886
## age_in_yrs_13                            7.1538872       31.8873120
## credit_amount_5                          5.0567832       24.1279401
## p_employment_since_7                     3.8286815       22.2407459
## housing_type_15                          0.7101342        8.4744696
## other_instalment_type_14                10.1328208       10.1134226
## personal_status_9                        1.9927075       13.1122406
## foreign_worker_20                        1.5833562        0.8129158
## other_debtors_or_grantors_10             7.5723423        7.2618802
## instalment_pct_8                         3.5926860       16.9248212
```

```
# Normalizamos para poner las dos variables en la misma escala y poderlas comparar. Lo
s valores negativos son las que menos predicen y los positivos los que mas
importancia_norm <- as.data.frame(scale(importancia))
#creamos una única variable como suma de las otras
#scale lo que hace es calcular una puntuación típica (z =(x-Med)/Sx)
importancia_norm
```

```
##                              MeanDecreaseAccuracy MeanDecreaseGini
## chk_ac_status_1                          3.04781484      1.869348103
## credit_history_3                         1.12943548     -0.007479116
## duration_month_2                         0.34478107      0.833450282
## savings_ac_bond_6                       -0.02972864     -0.238116943
## purpose_4                               -0.57087192      1.028597975
## property_type_12                        -0.40825973     -0.112089796
## age_in_yrs_13                           -0.04713103      1.298738715
## credit_amount_5                         -0.33173059      0.529979922
## p_employment_since_7                    -0.49839718      0.343006397
## housing_type_15                         -0.92161756     -1.020885683
## other_instalment_type_14                 0.35714228     -0.858506617
## personal_status_9                       -0.74755859     -0.561399106
## foreign_worker_20                       -0.80311195     -1.779953145
## other_debtors_or_grantors_10             0.00965783     -1.141022816
## instalment_pct_8                        -0.53042431     -0.183668170
```

```
#A continuación, se pasan las ounbtuaciones típicas a valores positivos y se suman par
a ver qué variable predice más.
importancia_norm <- importancia_norm %>% mutate(
  Variable = rownames(importancia_norm),
  Imp_tot = MeanDecreaseAccuracy + MeanDecreaseGini) %>%
  mutate(Imp_tot = Imp_tot + abs(min(Imp_tot))) %>%
  arrange(desc(Imp_tot)) %>%
  select(Variable,Imp_tot,MeanDecreaseAccuracy,MeanDecreaseGini)
#Con select se reordena de mayor a menor.

# Para interpretar la tabla final es recomendable no olo mirar la suma de los dos valo
res (Imp_tot), también conviene mirar las columnas de ACCURACY y GINI.

# Hacemos un gráfico para ver la curva de caída de importancia
ggplot(importancia_norm, aes(reorder(Variable,-Imp_tot),Imp_tot)) + geom_bar(stat = "i
dentity") + theme(axis.text.x = element_text(angle = 90,size = 7))
```




```
importancia_norm
```

```
##                         Variable   Imp_tot MeanDecreaseAccuracy MeanDecreaseGini
## 1                chk_ac_status_1 7.5002280           3.04781484      1.869348103
## 2                  age_in_yrs_13 3.8346728          -0.04713103      1.298738715
## 3               duration_month_2 3.7612964           0.34478107      0.833450282
## 4               credit_history_3 3.7050215           1.12943548     -0.007479116
## 5                     purpose_4 3.0407911          -0.57087192      1.028597975
## 6                credit_amount_5 2.7813144          -0.33173059      0.529979922
## 7           p_employment_since_7 2.4276743          -0.49839718      0.343006397
## 8              savings_ac_bond_6 2.3152195          -0.02972864     -0.238116943
## 9       other_instalment_type_14 2.0817008           0.35714228     -0.858506617
## 10              property_type_12 2.0627156          -0.40825973     -0.112089796
## 11              instalment_pct_8 1.8689726          -0.53042431     -0.183668170
## 12 other_debtors_or_grantors_10 1.4517001           0.00965783     -1.141022816
## 13             personal_status_9 1.2741074          -0.74755859     -0.561399106
## 14               housing_type_15 0.6405618          -0.92161756     -1.020885683
## 15             foreign_worker_20 0.0000000          -0.80311195     -1.779953145
```

Como resultados de la tabla anterior, se opta por eliminar housing_type_15 y foreign_worker_20.

# Paso 2. Segundo modelo

```
#Eliminamos del df de trabajo las dos variables.
df <- select(df,-housing_type_15, -foreign_worker_20)

#Lanzamos un segundo modelo
mod_rf2<-randomForest(target~., train,importance=T)
mod_rf2
```

```
##
## Call:
##  randomForest(formula = target ~ ., data = train, importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 23.14%
## Confusion matrix:
##      Bad Good class.error
## Bad   92  118  0.56190476
## Good  44  446  0.08979592
```

Aporta la siguiente información:

- Number of trees: 500 (500 árboles analizados)
- No. of variables tried at each split: 3 (número de variables seleccionadas para cada árbol = 3)
- OOB estimate of error rate: 23.14%

# 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]”

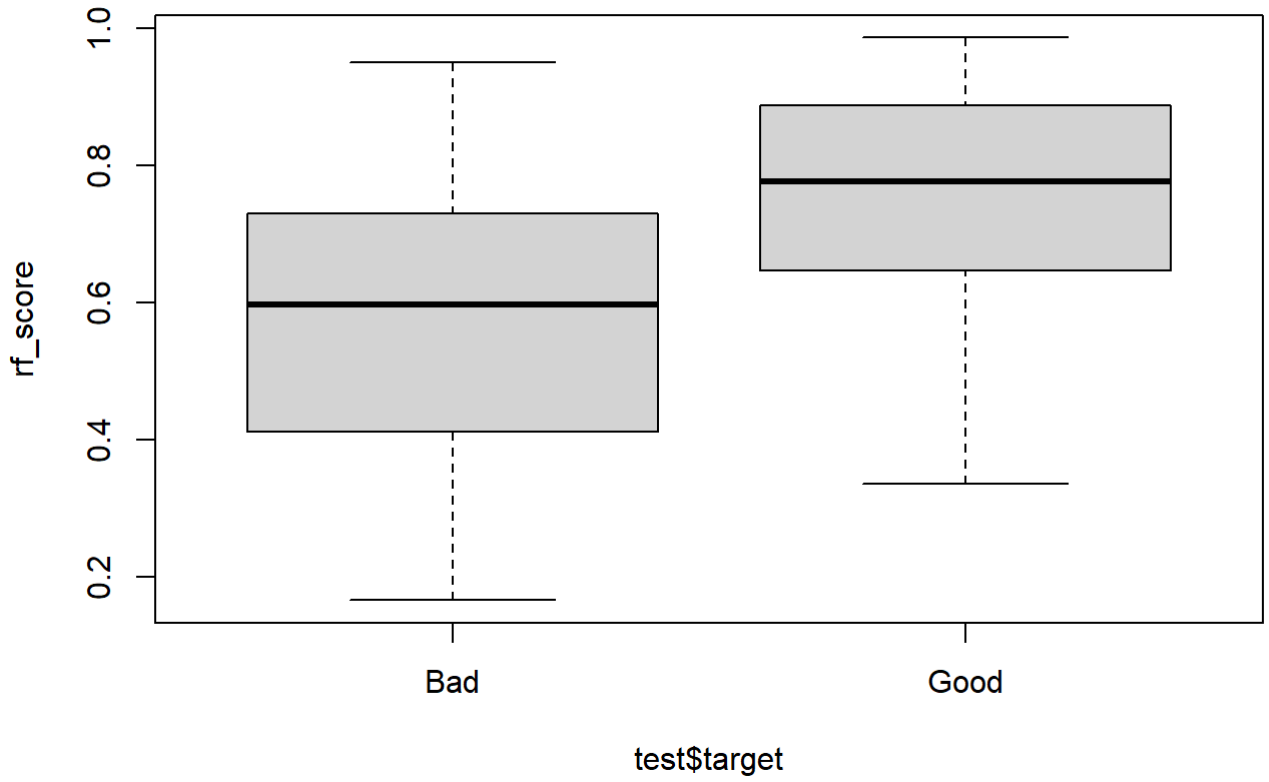
```
rf_score<-predict(mod_rf2,test,type = 'prob')[,2]
head(rf_score)
```

```
##     4     7     9    13    14    23
## 0.534 0.908 0.930 0.548 0.532 0.804
```

Lanzamos un “head” para ver los 6 primeros. Lo que quiere decir que: el sujeto 1 tendrá una probabilidad de clasificarse como 1 (Goog credit) del 54,2%. El segundo de 91,8%, 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 rf_score de los clientes “que sí hacen churn”Good" con la media de clientes “Bad” es diferente.

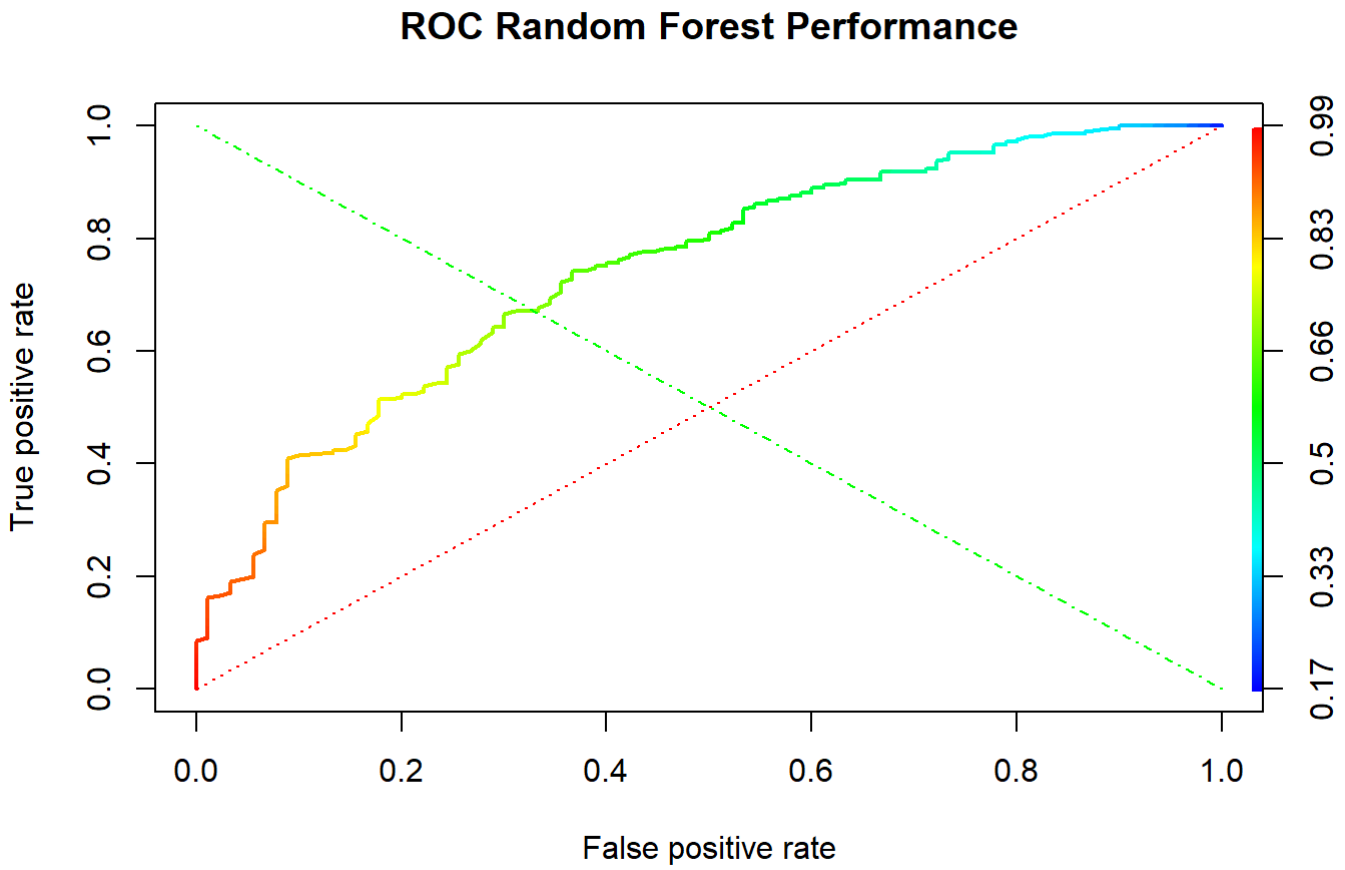
```
plot(rf_score~test$target)
```




Se observa en la gráfica que la media de los Bad y los Good es relativamente diferente.

# Paso 4. Curva ROC

```
pred_rf <- prediction(rf_score, test$target)
perf_rf <- performance(pred_rf,"tpr","fpr")
#Library(ROCR)
plot(perf_rf, lwd=2, colorize=TRUE, main="ROC Random Forest Performance")
lines(x=c(0, 1), y=c(0, 1), col="red", lwd=1, lty=3);
lines(x=c(1, 0), y=c(0, 1), col="green", lwd=1, lty=4)
```




# Paso 5. Umbrales y matriz de confusión

A continuación se probarán distintos umbrales para maximizar el F1 al transformar la probabilidad obtenida en otra dicotómica (Good y Bad credit).

En otros proyectos hemos empleado funciones. En este caso lo haremos una por una para entender mejor le proceso. Lo que vamos cambiando es el umbral ("treshold"), observando en cada caso cómo varían las mátricas de la matriz de confusión (exactitud, sensibilidad, precisión y F1).

```
rf_score2<-predict(mod_rf2,test,type = 'prob')[,2]

score2 <- ifelse(rf_score2 > 0.20, "Good", "Bad")
MC <- table(test$target, score2)
Acc2 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen2 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr2 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F12 <- 2*Pr2*Sen2/(Pr2+Sen2)

score3 <- ifelse(rf_score2 > 0.30, "Good", "Bad")
MC <- table(test$target, score3)
Acc3 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen3 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr3 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F13 <- 2*Pr3*Sen3/(Pr3+Sen3)

score4 <- ifelse(rf_score2 > 0.40, "Good", "Bad")
MC <- table(test$target, score4)
Acc4 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen4 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr4 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F14 <- 2*Pr4*Sen4/(Pr4+Sen4)

score5 <- ifelse(rf_score2 > 0.50, "Good", "Bad")
MC <- table(test$target, score5)
Acc5 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen5 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr5 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F15 <- 2*Pr5*Sen5/(Pr5+Sen5)

score6 <- ifelse(rf_score2 > 0.60, "Good", "Bad")
MC <- table(test$target, score6)
Acc6 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen6 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr6 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F16 <- 2*Pr6*Sen6/(Pr6+Sen6)

score7 <- ifelse(dt_score2 > 0.70, "Good", "Bad")
MC <- table(test$target, score7)
Acc7 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen7 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr7 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F17 <- 2*Pr7*Sen7/(Pr7+Sen7)

score8 <- ifelse(rf_score2 > 0.80, "Good", "Bad")
MC <- table(test$target, score8)
Acc8 <- (MC[1,1] + MC[2,2]) / sum(MC) *100
Sen8 <- MC[2,2] / (MC[2,2] + MC[1,2]) *100
Pr8 <- MC[2,2] / (MC[2,2] + MC[2,1]) *100
F18 <- 2*Pr8*Sen8/(Pr8+Sen8)
```

```
#salida<-c(Acc2,Acc3,Acc4,Acc5,Acc6,Acc7,Acc8)
#salida
#salida<-c(Sen2,Sen3,Sen4,Sen5,Sen6,Sen7,Sen8)
#salida
#salida<-c(Pr2,Pr3,Pr4,Pr5,Pr6,Pr7,Pr8)
#salida
salida<-c(F12,F13,F14,F15,F16,F17,F18)
salida
```

```
## [1] 82.67717 83.49901 83.57588 83.15098 79.43262 78.17746 59.56113
```

Se puede observar que el límite donde se mazimiza la F1 es en 0,4, con un F1 = 83.99168

# Paso 6. Métricas definitivas

```
score4 <- ifelse(rf_score2 > 0.40, "Good", "Bad")
MC <- table(test$target, score4)
rf_Acc <- round((MC[1,1] + MC[2,2]) / sum(MC) *100, 2)
rf_Sen <- round(MC[2,2] / (MC[2,2] + MC[1,2]) *100, 2)
rf_Pr <- round(MC[2,2] / (MC[2,2] + MC[2,1]) *100, 2)
rf_F1 <- round(2*rf_Pr*rf_Sen/(rf_Pr+rf_Sen), 2)

#KS & AUC
rf_KS <- round(max(attr(perf_rf,'y.values')[[1]]-attr(perf_rf,'x.values')[[1]])*100, 2
)
rf_AUROC <- round(performance(pred_rf, measure = "auc")@y.values[[1]]*100, 2)

cat("Acierto_rf: ",rf_Acc,"\tSensibilidad_rf: ", rf_Sen, "\tPrecision_rf:", rf_Pr, "\t
F1-rf:", rf_F1, "\tAUROC_rf: ",rf_AUROC,"\tKS_rf: ", rf_KS, "\n")
```

```
## Acierto_rf:  73.67   Sensibilidad_rf:  74.17     Precision_rf: 95.71     F1-rf: 83.
57    AUROC_rf:  74.41    KS_rf:  37.62
```

# 6. Comparación de los dos modelos

```
# Performance Table
models <- c('Árboles de decisión', 'Random forest')

#Accuracy
models_Acc <- c(dt_Acc, rf_Acc)

#Sensibilidad
models_Sen <- c(dt_Sen, rf_Sen)

#Precisión
models_Pr <- c(dt_Pr, rf_Pr)

#F1
models_F1 <- c(dt_F1, rf_F1)

# AUCs
models_AUC <- c(dt_AUROC, rf_AUROC)

# KS
models_KS <- c(dt_KS, rf_KS)
```

```
# Combine AUC and KS
metricas <- as.data.frame(cbind(models, models_Acc, models_Sen, models_Pr, models_F1,
 models_AUC, models_KS))
```

```
# Colnames
colnames(metricas) <- c("Model", "Acc", "Sen", "Pr", "F1", "AUC", "KS")
```

```
# Display Performance Reports
kable(metricas, caption ="Comparision of Model Performances")
```

Comparision of Model Performances

| Model | Acc | Sen | Pr | F1 | AUC | KS |
|---|---|---|---|---|---|---|
| Árboles de decisión | 74.67 | 77.92 | 89.05 | 83.11 | 70.97 | 33.81 |
| Random forest | 73.67 | 74.17 | 95.71 | 83.57 | 74.41 | 37.62 |

Conclusión:

Se observa que los dos modelos son muy semejantes en cuanto a acierto y sensibilidad. Pero es superior el modelo de Random forest en Precisión, ligeramente en F1, y en AUC. Por tato, parece que el modelo de RF es superior al de árboles de decisión.