Entrega Clasificación SVM

Arturo Gonzalez Moya

06 junio, 2021

Contents

1 Enunciado

Se considera la base de datos *"bank-full.csv"*. Los datos están relacionados con una campaña de marketing directo de una institución bancaria portuguesa. Las campañas de marketing se basaban en llamadas telefónicas. A menudo, se requería más de un contacto con el mismo cliente, para saber si el producto (depósito bancario a plazo) sería ("sí") o no ("no") suscrito. La descripción de la base de datos se puede encontrar en el fichero *"bank-names.txt"*. Observar bien la descripción de los datos ya que pueden existir variables que no son útiles para nuestro estudio. Se trata de resolver un problema de clasificación usando SVMs sobre el fichero *"bank_full"* y siguiendo los pasos descritos en las clases. Debéis desarrollar un estudio que incluya, al menos, los siguientes puntos:

- Exploración de datos - valores desaparecidos, valores atípicos
- Visualización de datos
- Matriz de correlaciones
- Partición de los datos 70:30 (librería "Caret" si se usa R)
- Paquete "e1071" o "Kernlab" (si se usa R)
- Ajuste del modelo y Plots
- Matriz de confusión

2 Introducción

La base de datos "bank_full" es la base de datos de una campaña de marketing directo de una institución bancaria portuguesa. Las campañas de marketing se basaban en llamadas telefónicas. A menudo, se requería más de un contacto con el mismo cliente, para saber si el producto (depósito bancario a plazo) sería ('sí') o no ('no') suscrito. Esta base de datos se puede encontrar en el Repositorio UCI de aprendizaje automático. El objetivo es saber si podemos predecir si un individuo se suscribirá a un depósito bancario a plazo o no. Abordaremos el problema utilizando maquinas de soporte vectorial.

3 Exploración y limpieza de los datos

Comenzamos la exploración de los datos. Lo primero que haremos será cargar el fichero de datos. *bank_full.csv*.

```
datos_banco <- read.csv("bank-full.csv", sep = ";")

dim(datos_banco)
```

```
## [1] 45211    17
```

```
summary(datos_banco)
```

```
##       age                 job           marital          education
##  Min.   :18.00   blue-collar:9732   divorced: 5207   primary  : 6851
##  1st Qu.:33.00   management :9458   married :27214   secondary:23202
##  Median :39.00   technician :7597   single  :12790   tertiary :13301
##  Mean   :40.94   admin.     :5171                    unknown  : 1857
##  3rd Qu.:48.00   services   :4154
##  Max.   :95.00   retired    :2264
##                  (Other)    :6835
##  default        balance       housing      loan            contact
##  no :44396   Min.   : -8019   no :20081   no :37967   cellular :29285
##  yes:  815   1st Qu.:    72   yes:25130   yes: 7244   telephone: 2906
##              Median :   448                           unknown  :13020
##              Mean   :  1362
##              3rd Qu.:  1428
##              Max.   :102127
##
##       day            month          duration         campaign
##  Min.   : 1.00   may    :13766   Min.   :   0.0   Min.   : 1.000
##  1st Qu.: 8.00   jul    : 6895   1st Qu.: 103.0   1st Qu.: 1.000
##  Median :16.00   aug    : 6247   Median : 180.0   Median : 2.000
##  Mean   :15.81   jun    : 5341   Mean   : 258.2   Mean   : 2.764
##  3rd Qu.:21.00   nov    : 3970   3rd Qu.: 319.0   3rd Qu.: 3.000
##  Max.   :31.00   apr    : 2932   Max.   :4918.0   Max.   :63.000
##                  (Other): 6060
##      pdays          previous           poutcome       y
##  Min.   : -1.0   Min.   :  0.0000   failure: 4901   no :39922
##  1st Qu.: -1.0   1st Qu.:  0.0000   other  : 1840   yes: 5289
##  Median : -1.0   Median :  0.0000   success: 1511
##  Mean   : 40.2   Mean   :  0.5803   unknown:36959
```

```
##  3rd Qu.: -1.0   3rd Qu.:  0.0000
##  Max.   :871.0   Max.   :275.0000
##
```

Este conjunto de datos tiene 45211 observaciones y 17 variables que son las siguientes:

- **age**: Edad del cliente (variable numérica).
- **job**: Tipo de trabajo (variable categórica). Las categorias son: *admin.*, *unknown*, *services*, *unemployed*, *management*, *housemaid*, *entrepreneur*, *student*, *blue-collar*, *self-employed*, *retired* y *technician*.
- **marital**: Estado civil (variable categórica). Las categorias son: *married*, *divorced* y *single*.
- **education**: Nivel de eduación (variable categórica). Las categorias son: *unknown*, *secondary*, *primary* y *tertiary*.
- **default**: Variable binaria que indica si el cliente tiene crédito en mora.
- **balance**: Balance medio anual, en euros (variable numérica).
- **housing**: Variable binaria que indica si el cliente tiene un prestamo para la vivienda.
- **loan**: Variable binaria que indica si el cliente tiene un préstamo personal.
- **contact**: Tipo de contacto (variable categórica). Las categorias son: *unknown*, *telephone* y *cellular*.
- **day**: Día del último contacto del mes (variable numérica).
- **month**: Mes del último contacto del año (variable categórica). Las categorias son: *jan*, *feb*, *mar*, *apr*, *may*, *jun*, *jul*, *aug*, *sep*, *oct*, *nov* y *dec*.
- **duration**: Duración del último contacto, en segundos (variable numérica).
- **campaign**: Número de contactos realizados durante esta campaña y para este cliente (variable numérica).
- **pdays**: Número de días que pasaron después de que el cliente fue contactado por última vez desde una campaña anterior (variable numérica). Si aparece un -1 es que el cliente no fue contactado previamente.
- **previous**: Número de contactos realizados antes de esta campaña y para este cliente (variable numérica).
- **poutcome**: Resultado de la campaña de marketing anterior (variable categórica). Las categorias son: *unknown*, *other*, *failure* y *success*.
- **y**: Variable categórica objetivo que nos indica si un cliente se ha subscrito a un deposito a plazo.

Para este estudio, no tendremos en cuenta las variables **pdays** y **poutcome** ya que no corresponden a la misma campaña y la variable **default** ya que si tiene un crédito que pagar, no se suscribirá a al depósito.

```
datos_banco$pdays<-NULL
datos_banco$poutcome<-NULL
datos_banco$default<-NULL
```

3.1 Análisis exploratorio de los datos

Veamos primero si exiten valores perdido en el conjunto de datos.

```r
anyNA(datos_banco)
```

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

Podemos observar que no. Además, antes hemos visto que ningúna variable categórica tiene tampoco valores perdidos.

Un problema que tenemos es que la variable **month** tiene muchos niveles. Lo que haremos será agruparlos en trimestres.

```r
summary(datos_banco$month)
```

```
##   apr   aug   dec   feb   jan   jul   jun   mar   may   nov   oct   sep
##  2932  6247   214  2649  1403  6895  5341   477 13766  3970   738   579
```

```r
datos_banco$month <- gsub('^jan', 'First_Trim', datos_banco$month)
datos_banco$month <- gsub('^feb', 'First_Trim', datos_banco$month)
datos_banco$month <- gsub('^mar', 'First_Trim', datos_banco$month)
datos_banco$month <- gsub('^apr', 'Second_Trim', datos_banco$month)
datos_banco$month <- gsub('^may', 'Second_Trim', datos_banco$month)
datos_banco$month <- gsub('^jun', 'Second_Trim', datos_banco$month)
datos_banco$month <- gsub('^jul', 'Third_Trim', datos_banco$month)
datos_banco$month <- gsub('^aug', 'Third_Trim', datos_banco$month)
datos_banco$month <- gsub('^sep', 'Third_Trim', datos_banco$month)
datos_banco$month <- gsub('^oct', 'Fourth_Trim', datos_banco$month)
datos_banco$month <- gsub('^nov', 'Fourth_Trim', datos_banco$month)
datos_banco$month <- gsub('^dec', 'Fourth_Trim', datos_banco$month)

datos_banco$month <- as.factor(datos_banco$month)
```

```r
summary(datos_banco$month)
```

```
##  First_Trim Fourth_Trim Second_Trim  Third_Trim
##        4529        4922       22039       13721
```

En el caso de la variable **job** podemos observar que, al igual que la variable **month**, tiene muchos niveles.

```r
summary(datos_banco$job)
```

```
##        admin.   blue-collar  entrepreneur     housemaid    management
##          5171          9732          1487          1240          9458
##       retired self-employed      services       student    technician
##          2264          1579          4154           938          7597
##    unemployed       unknown
##          1303           288
```

Lo que haremos será agrupar en 5 niveles. que seran los siguientes:

- white-collar: admin., entrepreneur, management, technician

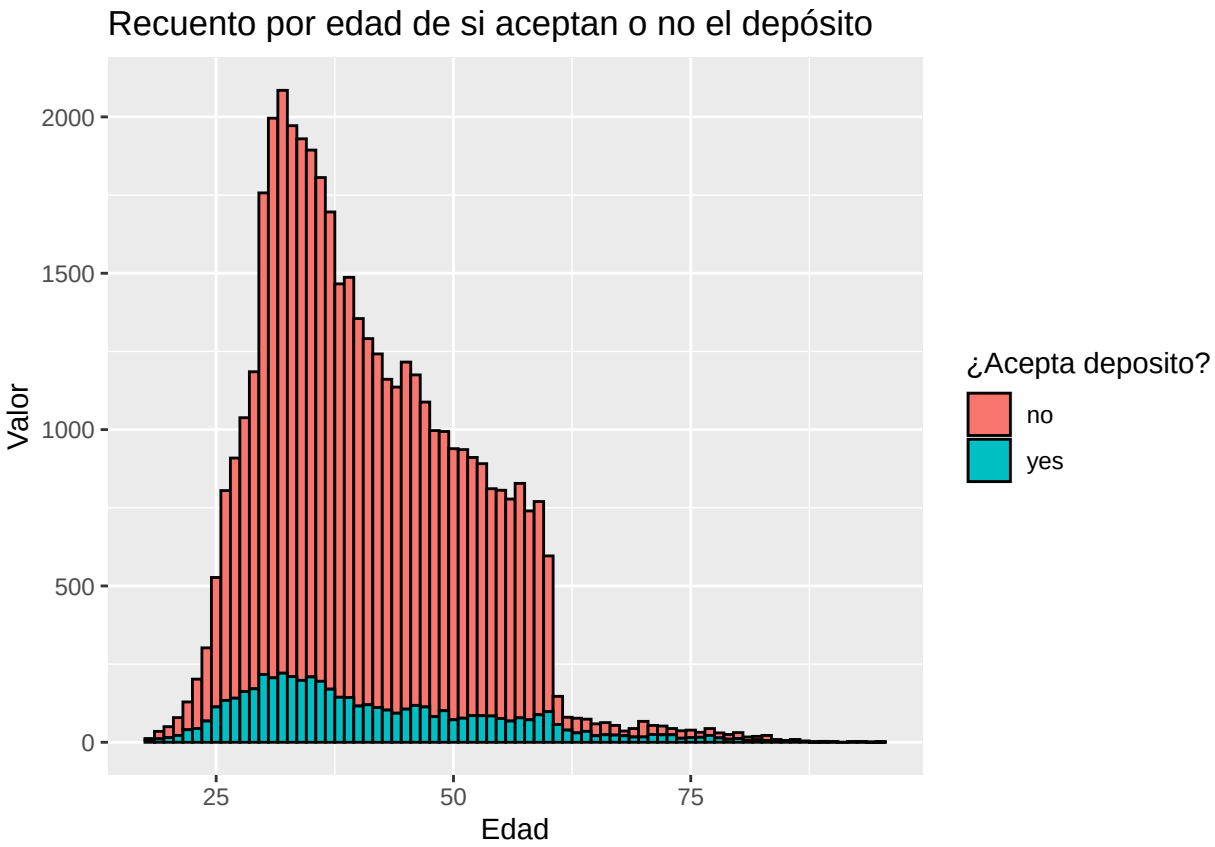
- blue-collar
- services
- not_work: student, retired, unemployed
- other/unknown: housemaid, self-employed, unknown

```
datos_banco$job <- gsub('^admin.', 'white-collar', datos_banco$job)
datos_banco$job <- gsub('^entrepreneur', 'white-collar', datos_banco$job)
datos_banco$job <- gsub('^management', 'white-collar', datos_banco$job)
datos_banco$job <- gsub('^technician', 'white-collar', datos_banco$job)
datos_banco$job <- gsub('^student', 'not_work', datos_banco$job)
datos_banco$job <- gsub('^retired', 'not_work', datos_banco$job)
datos_banco$job <- gsub('^unemployed', 'not_work', datos_banco$job)
datos_banco$job <- gsub('^housemaid', 'other_unknown', datos_banco$job)
datos_banco$job <- gsub('^self-employed', 'other_unknown', datos_banco$job)
datos_banco$job <- gsub('^unknown', 'other_unknown', datos_banco$job)

datos_banco$job <- as.factor(datos_banco$job)
```

3.1.1 Visualización de los datos

Comenzamos haciendo un recuento de la variable **age** (que es una variable numérica) con respecto a la variable **y** que es la que queremos predecir. Por lo tanto, haremos un histograma.

```
ggplot(datos_banco) +
  aes(x=as.numeric(age), group=y, fill=y) +
  geom_histogram(binwidth=1, color='black')+
  xlab("Edad") +
  ylab("Valor") +
  ggtitle("Recuento por edad de si aceptan o no el depósito")+
  labs(fill="¿Acepta deposito?")
```

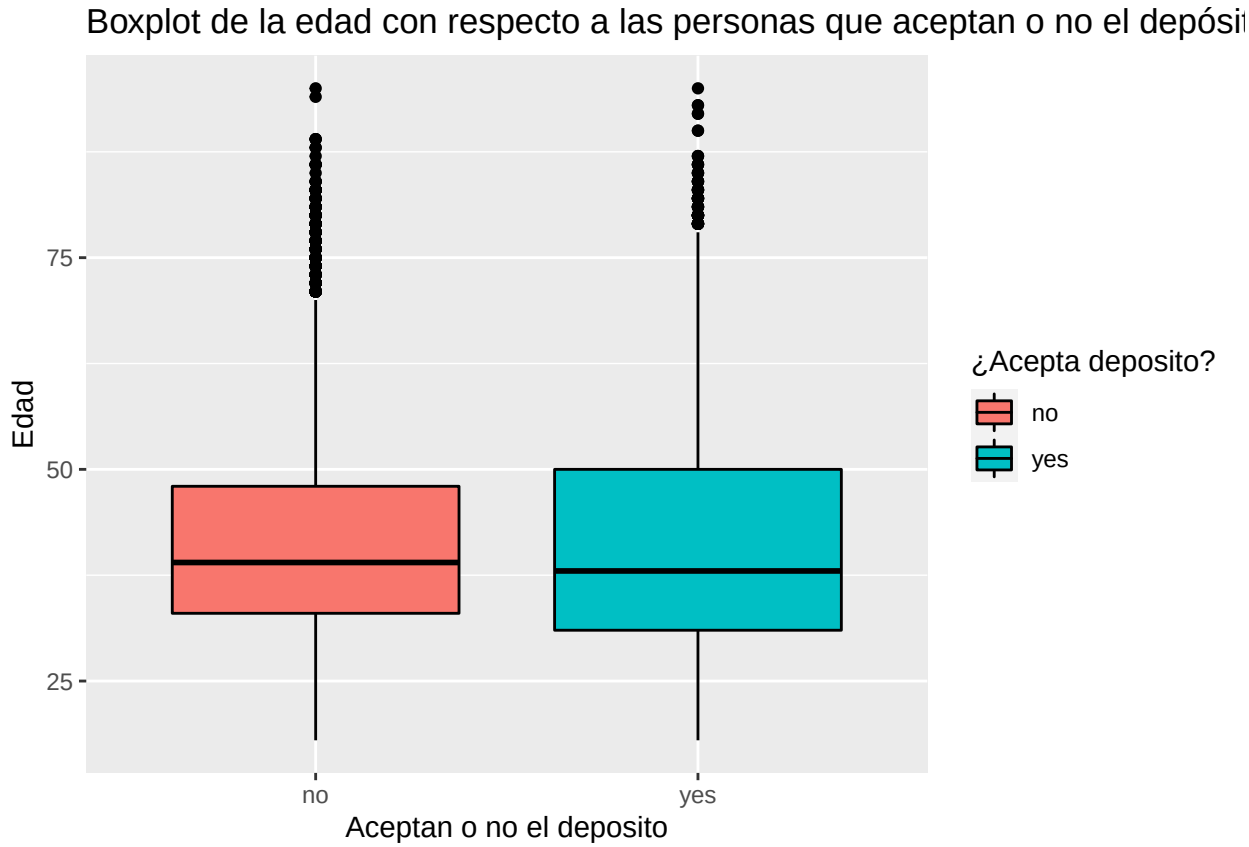


Podemos observar que la mayoría de los individuos no se suscriben al deposito bancario a plazo. Además, los que más se suscriben se encuentran entre las edades de 25 a 60 años. Vemos que las personas con edad entre 30 y 40 años son a las que más se les ha ofrecido el deposito a plazo.

Ya que el banco esta ofreciendo un deposito a plazo, se puede intuir que las personas jovenes se suscribirán más a este deposito que las personas mayores. Veámoslo con un boxplot.

```
ggplot(datos_banco) +
  aes(x= y, y= age, group=y, fill=y) +
  geom_boxplot(binwidth=1, color='black')+
  xlab("Aceptan o no el deposito") +
  ylab("Edad") +
  ggtitle("Boxplot de la edad con respecto a las personas que aceptan o no el depósito")+
  labs(fill="¿Acepta deposito?")
```

```
## Warning: Ignoring unknown parameters: binwidth
```

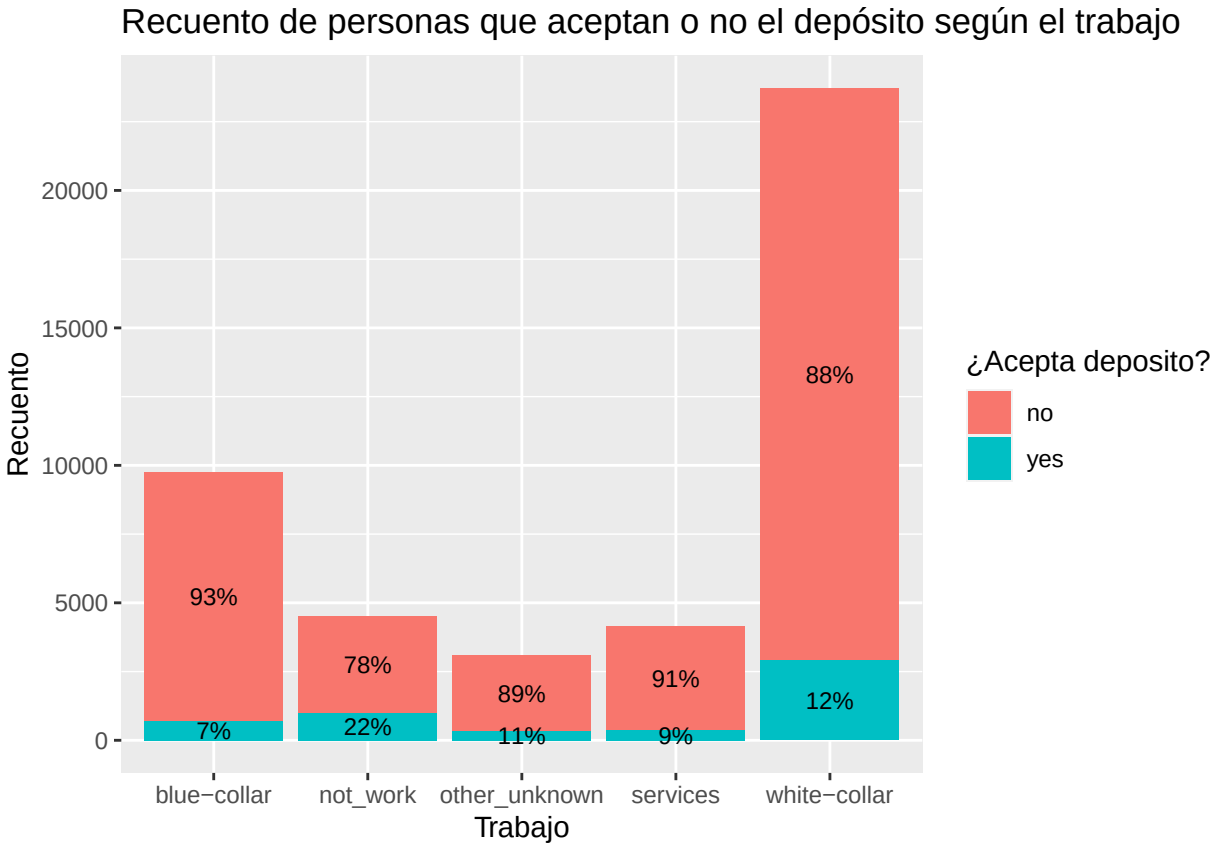


Para explorar la relación entre la clase de trabajo y si se suscriben al deposito a plazo, se calculan los recuentos en las dos categorías de suscripción en las cinco categorías de clase de trabajo, así como las proporciones dentro del grupo. Los resultados se representan como un diagrama de barras.

```
recuento <- mutate(dplyr::summarise(group_by(datos_banco,job, y), count = n()))
```

```
## `summarise()` regrouping output by 'job' (override with `.groups` argument)
```

```
recuento <- ddply(recuento, .(job), transform, percent = count/sum(count) * 100)

recuento <- ddply(recuento, .(job), transform, pos = (cumsum(count) - 0.5 * count))
recuento$label <- paste0(sprintf("%.0f", recuento$percent), "%")

ggplot(recuento, aes(x = job, y = count, fill = y)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5),size=3)+
  xlab("Trabajo") +
  ylab("Recuento") +
  ggtitle("Recuento de personas que aceptan o no el depósito según el trabajo")+
  labs(fill="¿Acepta deposito?")
```

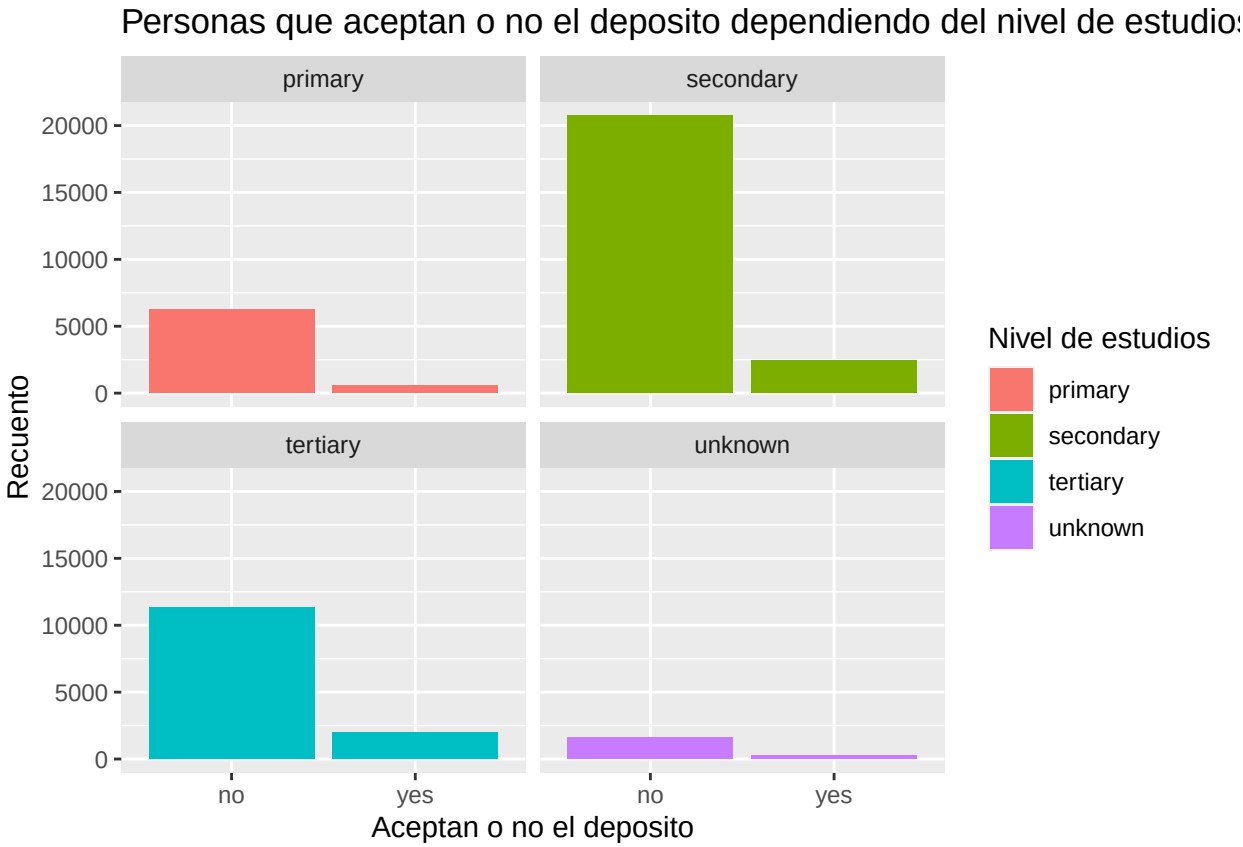


Podemos observar que el porcentaje de los individuos que se suscriben al depósito es mayor en el grupo de los que no trabajan. El porcentaje de los individuos que menos se suscriben corresponde a los del grupo *blue-collar*.

Pregunta: ¿La educación superior influye en la suscripción del depósito?

Respuesta: Exploramos esta pregunta trazando la educación en función de la suscripción al depósito, como se muestra en la siguiente figura.

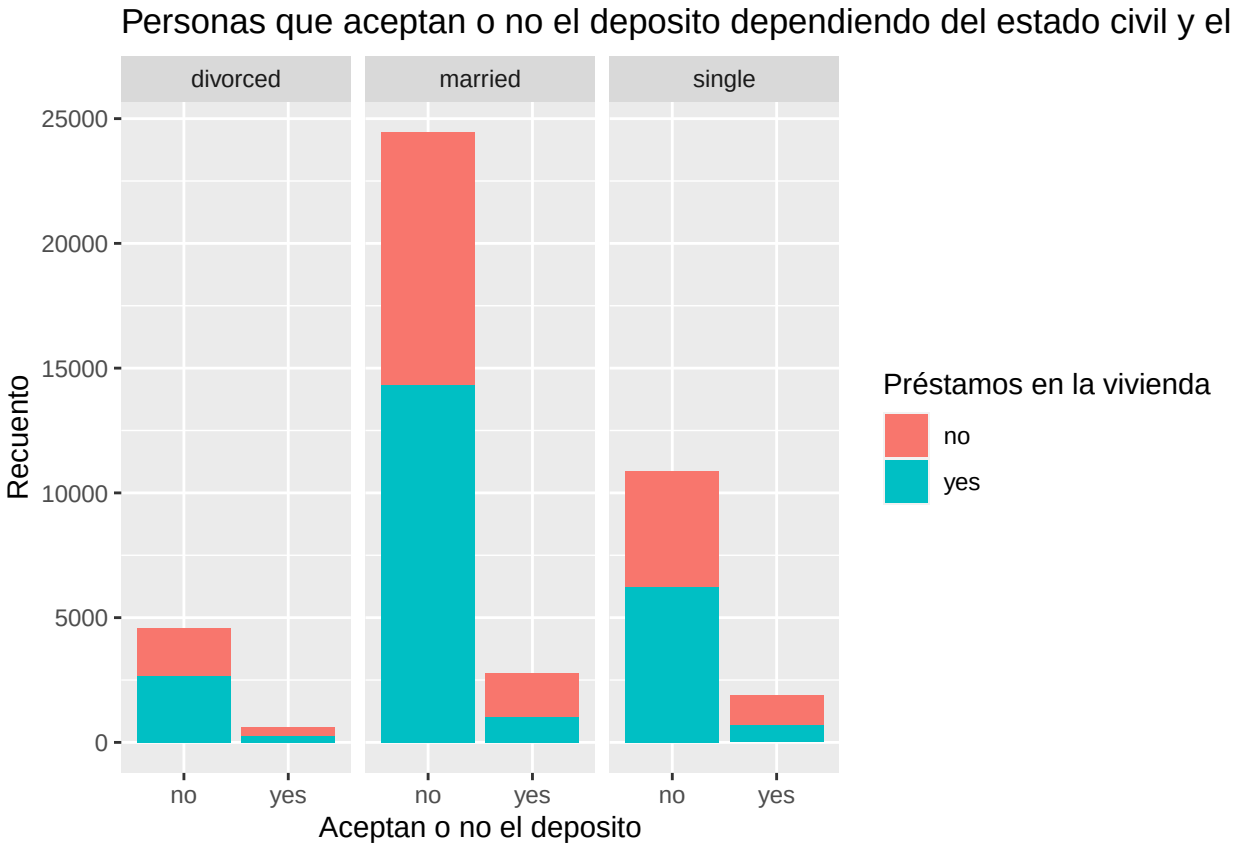
```
ggplot(datos_banco, mapping = aes(x= y, fill = education))+
  geom_bar( stat = "count")+
  facet_wrap(~education)+
  xlab("Aceptan o no el deposito") +
  ylab("Recuento") +
  ggtitle("Personas que aceptan o no el deposito dependiendo del nivel de estudios")+
  labs(fill="Nivel de estudios")
```



Podemos observar que, en proporción, las que tienen más nivel de estudios aceptan más el depósito a plazo.

Pasamos a obervar si existen diferencias entre los individuos dependiendo del estado civil y de tiene un préstamo para la vivienda.

```
ggplot(datos_banco, mapping = aes(x= y, fill = housing))+
  geom_bar( stat = "count")+
  facet_wrap(~marital)+
  xlab("Aceptan o no el deposito") +
  ylab("Recuento") +
  ggtitle("Personas que aceptan o no el deposito dependiendo del estado civil y el préstamo en la vivie
  labs(fill="Préstamos en la vivienda")
```

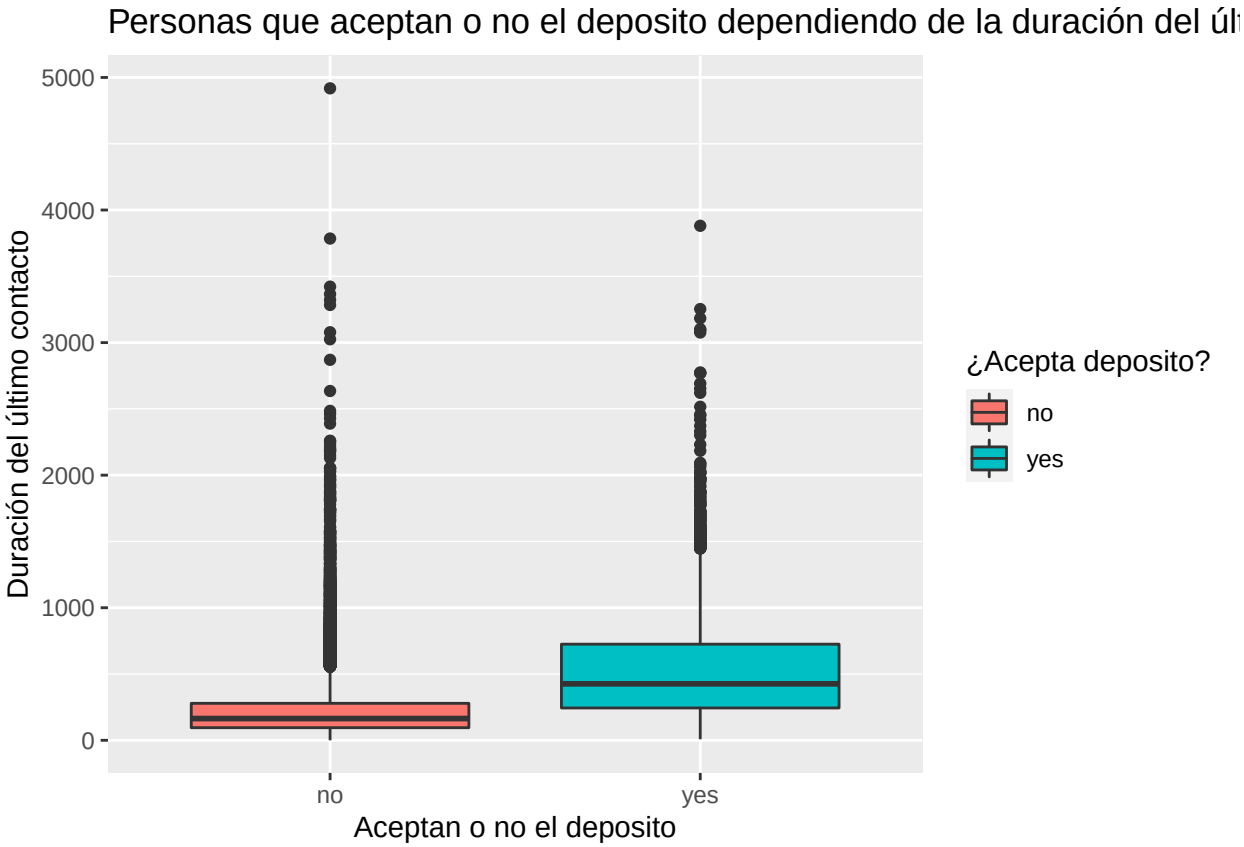


Podemos ver que no existen difeencias muy significativas en los invididuos que aceptan o no el depósito dependiendo del estado civil. Recalcar que los individuos con un préstamo para la vivienda que rechazan el déposito son más que los que tienen un préstamo para la vivienda y aceptan el depósito.

Pregunta: ¿La duración del último contacto influye en la suscripción del depósito?

Respuesta: Exploramos esta pregunta dibujando la duración del último contacto en función de la suscripción al depósito, como se muestra en la siguiente figura.

```
ggplot(datos_banco, mapping = aes(x= y, y = duration, fill = y))+
  geom_boxplot()+
  xlab("Aceptan o no el deposito") +
  ylab("Duración del último contacto") +
  ggtitle("Personas que aceptan o no el deposito dependiendo de la duración del último contacto")+
  labs(fill="¿Acepta deposito?")
```



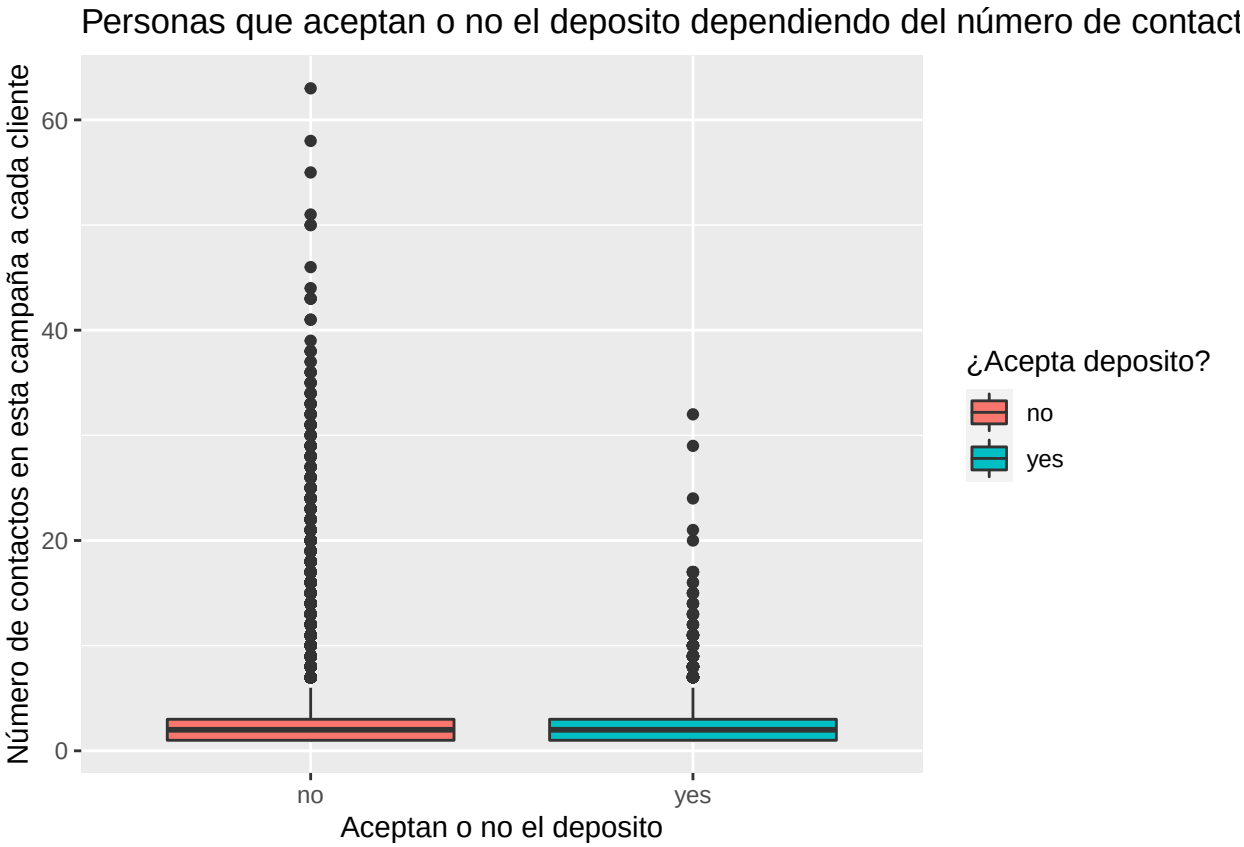
Podemos observar que para las personas que aceptan el depósito, la duración del último contacto es mayor que los que no aceptan. Ya que en esta variable encontramos una diferencía clara, lo que haremos será eliminar los outliers que se corresponden con un coeficiente de 3 en el boxplot.

```
## Para eliminar los outliers mas a adelante.
p <- filter(datos_banco, y == "yes")
a<-which(p$duration %in% boxplot.stats(p$duration, coef = 3)$out)
q <- filter(datos_banco, y == "no")
b<-which(q$duration %in% boxplot.stats(q$duration, coef = 3)$out)

c<- union(a,b)
```

Pregunta: ¿El número de contactos realizados durante esta campaña y para este cliente influye en la suscripción del depósito?

Respuesta: Exploramos esta pregunta graficando el número de contactos realizados para cada cliente en esta campaña en función de la suscripción al depósito, como se muestra en la siguiente figura.

```
ggplot(datos_banco, mapping = aes(x= y, y = campaign, fill = y))+
  geom_boxplot()+
  xlab("Aceptan o no el deposito") +
  ylab("Número de contactos en esta campaña a cada cliente") +
  ggtitle("Personas que aceptan o no el deposito dependiendo del número de contactos esta campaña")+
  labs(fill="¿Acepta deposito?")
```



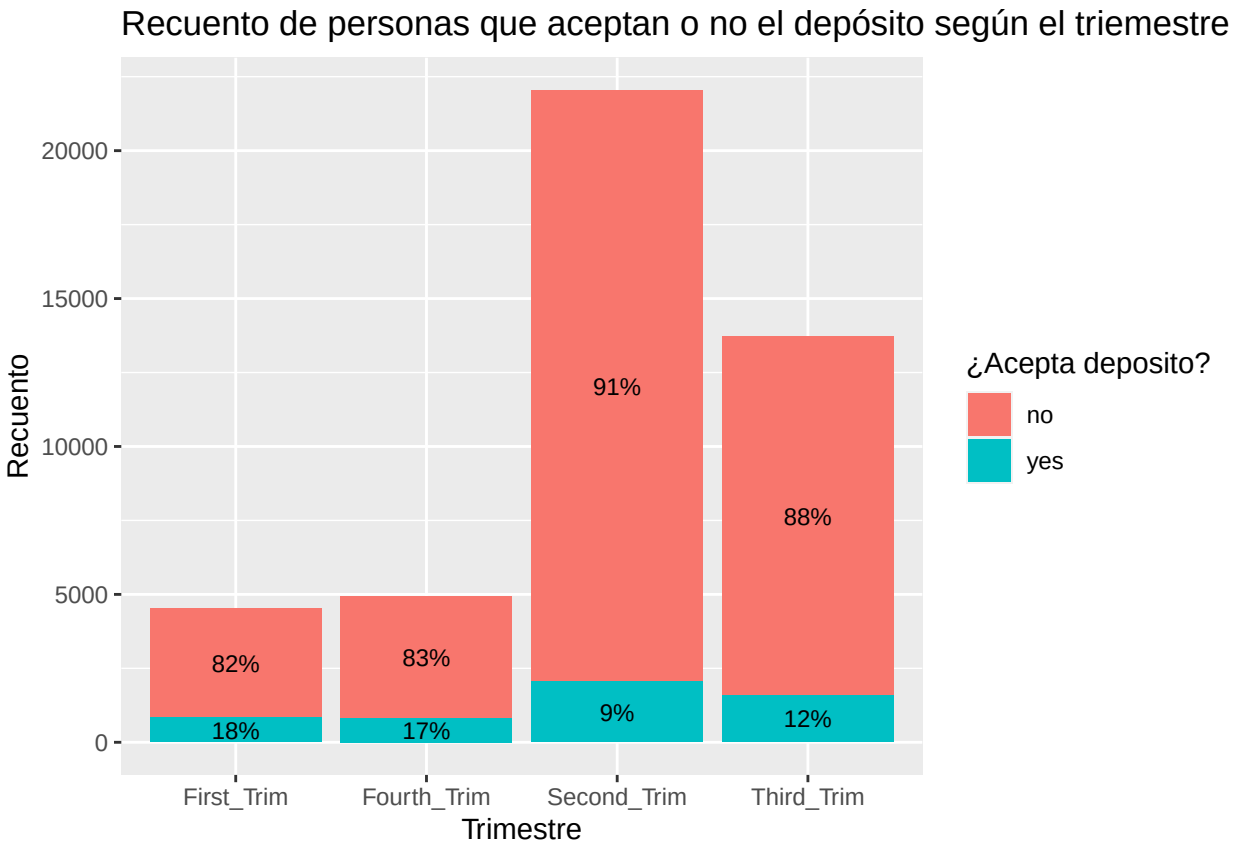
Vemos que no existen diferencias claras entre los individuos que se suscriben o no al depósito.

Estudiemos ahora si el trimestre en el que se tuvo el último contacto con el cliente tiene relación con su suscripción al depósito.

```
recuento <- mutate(dplyr::summarise(group_by(datos_banco,month, y), count = n()))
```

```
## `summarise()` regrouping output by 'month' (override with `.groups` argument)
```

```
recuento <- ddply(recuento, .(month), transform, percent = count/sum(count) * 100)

recuento <- ddply(recuento, .(month), transform, pos = (cumsum(count) - 0.5 * count))
recuento$label <- paste0(sprintf("%.0f", recuento$percent), "%")

ggplot(recuento, aes(x = month, y = count, fill = y)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5),size=3)+
  xlab("Trimestre") +
  ylab("Recuento") +
  ggtitle("Recuento de personas que aceptan o no el depósito según el triemestre en el que las contacta
  labs(fill="¿Acepta deposito?")
```



Podemos observar que en el segundo trimestre es cuando más contactos a clientes se realizaron. En el primer y cuarto trimestre se dan los porcentajes más altos de gente que se suscribe al depósito.

3.1.2 Detectando "outliers" y variables "skewed" (asimétricas)

Una forma de estudiar la existencia de variables asimétricas es utilizando algún índice de asimetría como el determinado por la función skewness(). Una variable se considera "muy asimétrica" si su valor absoluto es mayor que 1. Se considera una variable, "moderately skewed" si su valor absoluto es mayor que 0.5.

```
skewedVars<- NA
for(i in names(datos_banco)){
   if(is.numeric(datos_banco[,i])){
     if(i != "y"){
       # Enters this block if variable is non-categorical
       skewVal <- skewness(datos_banco[,i])
       print(paste(i, skewVal, sep = ": "))
       if(abs(skewVal) > 1){
         skewedVars <- c(skewedVars, i)
       }
     }
   }
}
```

```
## [1] "age: 0.684795204786645"
## [1] "balance: 8.36003094725269"
```

```
## [1] "day: 0.0930759258389723"
## [1] "duration: 3.14421377701039"
## [1] "campaign: 4.89848763841056"
## [1] "previous: 41.8450660879732"
```

```
skewedVars
```

```
## [1] NA         "balance"  "duration" "campaign" "previous"
```

En nuestro caso, podemos observar que las variables *balance*, *duration*, *campaign* y *previous* son muy asimétricas. Lo que haremos por lo tanto será eliminar todas las mencionadas menos *duration*, a la que le eliminaremos los outliers ya que vimos que podía ser relevante en el estudio.

```
datos_banco$balance <- NULL
datos_banco$previous <- NULL
datos_banco$campaign <- NULL

datos_banco <- datos_banco[-c,] #Eliminamos los outliers de duration
```

Hemos eliminado el siguiente número de outliers.

```
length(c)
```

```
## [1] 760
```

También eliminaremos la variable *contact* ya que se considera que no se considera de gran importancia en el estudio si se ha contactado con el individuo mediante teléfono movil o mediante teléfono fijo.

```
datos_banco$contact <- NULL
```

Hagamos un resumen de los datos tras la eliminación de estas variables y de los outliers.

```
summary(datos_banco)
```

```
##       age                    job            marital          education
##  Min.   :18.00   blue-collar  : 9562   divorced: 5115   primary  : 6731
##  1st Qu.:33.00   not_work     : 4449   married :26767   secondary:22811
##  Median :39.00   other_unknown: 3069   single  :12569   tertiary :13081
##  Mean   :40.95   services     : 4048                    unknown  : 1828
##  3rd Qu.:48.00   white-collar :23323
##  Max.   :95.00
##  housing      loan            day                month          duration
##  no :19794   no :37342   Min.   : 1.00   First_Trim : 4445   Min.   :   0
##  yes:24657   yes: 7109   1st Qu.: 8.00   Fourth_Trim: 4836   1st Qu.: 103
##                          Median :16.00   Second_Trim:21642   Median : 180
##                          Mean   :15.81   Third_Trim :13528   Mean   : 258
##                          3rd Qu.:21.00                       3rd Qu.: 318
##                          Max.   :31.00                       Max.   :4918
##    y
##  no :39215
##  yes: 5236
```

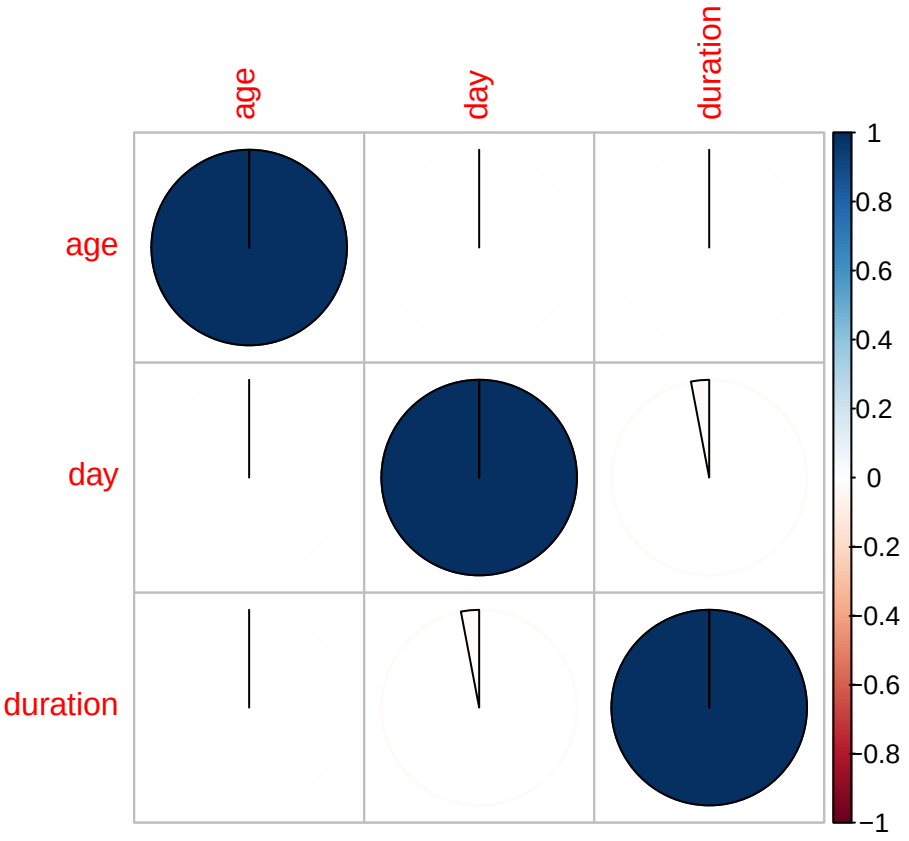
```
##
##
##
##
```

Vemos que nos hemos quedado con 9 variables más la variable que vamos a predecir.

3.1.3 Detección de correlaciones

Ahora buscamos variables con altas correlaciones entre sí. La correlación mide la relación entre dos variables. Cuando dos variables están tan altamente correlacionadas que se explican entre sí (al punto de que uno puede predecir la variable con el otro), entonces tenemos un problema de colinealidad (o multicolinealidad). Por lo tanto, es importante tratar el problema de colinealidad. Veamos ahora, si nuestros datos tienen este problema o no. Es importante tener en cuenta que la correlación solo funciona para variables continuas. Podemos calcular las correlaciones usando la función "cor()".

```
correlat<- cor(datos_banco[c(1,7,9)])
corrplot(correlat, method = "pie")
```




```
highlyCor <- colnames(datos_banco)[findCorrelation(correlat, cutoff = 0.7, verbose = TRUE)]
```

```
## All correlations <= 0.7
```

De la figura, es evidente que ninguna de las variables están altamente correlacionadas entre sí.

Ya que tenemos variables categóricas, lo que haremos será pasarlas a one-hot encoding para poder trabajar con ellas en el modelo. Eliminaremos una de las columnas generadas de cada categoría para eliminiar la dependencia lineal entre ellas. Además, escalaremos las variables numéricas ya que los rangos en los que se encuentran son muy diferentes.

```
datos_banco[,c(1,7,9)] <- scale(datos_banco[,c(1,7,9)])
datos_banco$y <- as.factor(ifelse(datos_banco$y == "no",0,1))
datos_banco_oh <- dummy_cols(datos_banco, select_columns = c("job",
                                                             "marital",
                                                             "education",
                                                             "housing",
                                                             "loan",
                                                             "month"),
    remove_first_dummy = TRUE, remove_selected_columns = TRUE)
str(datos_banco_oh)
```

```
## 'data.frame':    44451 obs. of  18 variables:
##  $ age                : num  1.603 0.287 -0.748 0.569 -0.748 ...
##  $ day                : num  -1.3 -1.3 -1.3 -1.3 -1.3 ...
##  $ duration           : num  0.0118 -0.4159 -0.7076 -0.6454 -0.2332 ...
##  $ y                  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 ...
##  $ job_not_work       : int  0 0 0 0 0 0 0 1 0 ...
##  $ job_other_unknown  : int  0 0 0 0 1 0 0 0 0 ...
##  $ job_services       : int  0 0 0 0 0 0 0 0 0 ...
##  $ job_white-collar   : int  1 1 1 0 0 1 1 1 0 1 ...
##  $ marital_married    : int  1 0 1 1 0 1 0 0 1 0 ...
##  $ marital_single     : int  0 1 0 0 1 0 1 0 0 1 ...
##  $ education_secondary: int  0 1 1 0 0 0 0 0 0 1 ...
##  $ education_tertiary : int  1 0 0 0 0 1 1 1 0 0 ...
##  $ education_unknown  : int  0 0 0 1 1 0 0 0 0 0 ...
##  $ housing_yes        : int  1 1 1 1 0 1 1 1 1 1 ...
##  $ loan_yes           : int  0 0 1 0 0 0 1 0 0 0 ...
##  $ month_Fourth_Trim  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month_Second_Trim  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ month_Third_Trim   : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
colnames(datos_banco_oh)[8] <- "job_white_collar"
```

3.2 Creación del conjunto de entrenamiento y del conjunto de test

El 70% de los datos originales se utilizan como conjunto de entrenamiento, mientras que el resto (el 30%) se utilizan como conjunto de prueba.

```
set.seed(3456)
trainIndex <- createDataPartition(datos_banco_oh$y, p = .7,
                                  list = FALSE,
                                  times = 1)
banco_train <- datos_banco_oh[trainIndex, ]
banco_test <- datos_banco_oh[-trainIndex, ]
```

3.2.1 Balanceo del conjunto de datos

Antes de proseguir, observemos que el conjunto de datos que tenemos se encuentra desbalanceado.

```
table(datos_banco_oh$y)/nrow(datos_banco_oh)
```

```
##
##         0         1
## 0.8822074 0.1177926
```

Tenemos que un 88.22% de individuos no se suscriben al depósito mientras que solo un 11.77% si que lo hacen.

Realizaremos técnicas de undersampling y oversampling del paquete unbalanced.

Se han decidido usar SMOTE, ENN, SMOTE + ENN, CNN, NCL, OSS, Tomek_links, random oversampling y random undersampling con diferentes parametros y se ha tomado la F-medida para seleccionar la mejor forma de balanceo para nuestro conjunto de datos.

```
#Para ver que metodo de balanceo da mejor F-medida.
medidas = matrix(rep(0,40), nrow = 10, ncol = 4)
banco_train = banco_train[, c(1:3,18,5:17,4)]
for (k in 1:10) {
  for (i in 1:2) {
    X= banco_train[,-18]
    Y= banco_train[,18]
    res_SMOTE=ubSMOTE(X, Y, perc.over = i*100, k = k, perc.under = 0, verbose = TRUE)
    data = data.frame(res_SMOTE$X,res_SMOTE$Y)
    names(data) <- names(banco_train)
    banco_train_balanced=rbind(banco_train,data)

    svm4 <- ksvm(y ~ ., data = banco_train_balanced)
    svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
    medidas[k,i] = F_meas(svm4.pred,banco_test$y, relevant = "1")

    X= banco_train_balanced[,-18]
    Y= banco_train_balanced[,18]
    ENN = ubENN(X, Y, k = 3, verbose = TRUE)
    banco_train_balanced <- cbind(ENN$X, y = ENN$Y)

    svm4 <- ksvm(y ~ ., data = banco_train_balanced)
    svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
    medidas[k,(i+2)] = F_meas(svm4.pred,banco_test$y, relevant = "1")
  }
}

medidas


X= banco_train[,-18]
Y= banco_train[,18]
for (j in 1:2) {
  CNN <- ubCNN(X,Y,k=j)
  banco_train_balanced <- cbind(CNN$X, y=CNN$Y)
```

```
  svm4 <- ksvm(y ~ ., data = banco_train_balanced)
  svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
  medidas_under = F_meas(svm4.pred,banco_test$y, relevant = "1")
  print(medidas_under)
}

for (k in 1:5) {
  ENN <- ubENN(X,Y,k=k)
  banco_train_balanced <- cbind(ENN$X, y=ENN$Y)

  svm4 <- ksvm(y ~ ., data = banco_train_balanced)
  svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
  medidas_under = F_meas(svm4.pred,banco_test$y, relevant = "1")
  print(medidas_under)

  if (k %% 2 == 1){
    NCL = ubNCL(X, Y, k = k, verbose = TRUE)
    banco_train_balanced <- cbind(NCL$X, y=NCL$Y)

    svm4 <- ksvm(y ~ ., data = banco_train_balanced)
    svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
    medidas_under = F_meas(svm4.pred,banco_test$y, relevant = "1")
    print(medidas_under)
  }
}

OSS <- ubOSS(X, Y, verbose = TRUE)
banco_train_balanced <- cbind(OSS$X, y=OSS$Y)

svm4 <- ksvm(y ~ ., data = banco_train_balanced)
svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
medida_OSS = F_meas(svm4.pred,banco_test$y,relevant = "1")
print(medida_OSS)

tomek <- ubTomek(X, Y, verbose = TRUE)
banco_train_balanced <- cbind(tomek$X, y=tomek$Y)

svm4 <- ksvm(y ~ ., data = banco_train_balanced)
svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
medida_tomek = F_meas(svm4.pred,banco_test$y, relevant = "1")
print(medida_tomek)

set.seed(3456)
random <- ubUnder(X, Y, perc = 30, method = "percPos", w = NULL)
banco_train_balanced <- cbind(random$X, y=random$Y)

svm4 <- ksvm(y ~ ., data = banco_train_balanced)
svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
medida_under = F_meas(svm4.pred,banco_test$y, relevant = "1")
print(medida_under)

set.seed(3456)
random_over <- ubOver(X, Y, k = 3.6799, verbose=TRUE)
```

```
banco_train_balanced <- cbind(random_over$X, y=random_over$Y)

svm4 <- ksvm(y ~ ., data = banco_train_balanced)
svm4.pred <- predict(svm4, newdata = banco_test, type = 'response')
medida_over = F_meas(svm4.pred,banco_test$y, relevant = "1")
print(medida_over)
```

El método de balanceo que mejor F-medida ha proporcionado y que se ha escogido es el método *random undersampling*.

```
banco_train = banco_train[, c(1:3,18,5:17,4)] #ponemos la variable dependiente la última
X= banco_train[,-18]
Y= banco_train[,18]

set.seed(3456)
random <- ubUnder(X, Y, perc = 30, method = "percPos", w = NULL)
banco_train_balanced <- cbind(random$X, y=random$Y)
table(banco_train_balanced$y)/nrow(banco_train_balanced)
```

```
##
##   0   1
## 0.7 0.3
```

Podemos obsevar que el conjunto de datos de entremaniento ahora tiene un 70% de los datos de la clase mayoritaría y un 30% de la clase minoritaria.

4 Modelos

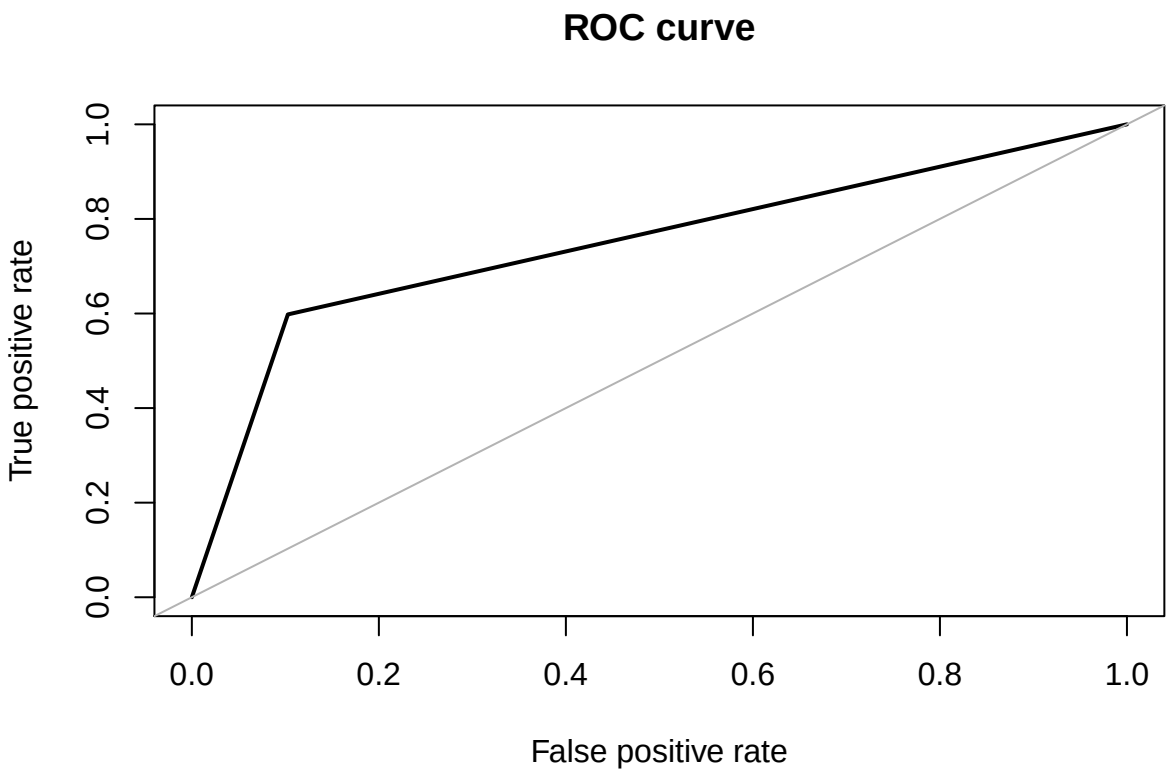
Una vez tenemos los datos limpios, podemos proceder a construir los modelos utilizando diferentes técnicas. Las medidas que utilizaremos para seleccionar el mejor modelo serán la F-medida y la medida AUC ya que tratamos con un conjunto de datos no balanceado.

Emplearemos varias técnicas de aprendizaje automático (además de la regresión logística que también utilizaremos) para predecir si una observación se suscribe al depósito o no. Los modelos son entrenados en el conjunto de entrenamiento y validados en el conjunto de prueba.

4.1 Modelo SVM

Los primeros modelos que ajustaremos estarán basados en las máquinas de soporte vectorial. Comenzaremos con un modelo de svm con kernel lineal y con los parámetros base de la función.

```
svm <- ksvm(y~., data=banco_train_balanced)

pre_svm <- predict(svm,banco_test)
auc_1 <- roc.curve(banco_test$y, pre_svm)$auc
```




```
f_medida_1<-  F_meas(pre_svm,banco_test$y,relevant = "1")
cm_imp_1 <- confusionMatrix(pre_svm,banco_test$y, positive = "1")
cm_imp_1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10556   631
##          1  1208   939
##
##                Accuracy : 0.8621
##                  95% CI : (0.8561, 0.8679)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4274
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.59809
##             Specificity : 0.89731
##          Pos Pred Value : 0.43735
##          Neg Pred Value : 0.94360
##              Prevalence : 0.11774
```

```
##          Detection Rate : 0.07042
##    Detection Prevalence : 0.16102
##       Balanced Accuracy : 0.74770
##
##        'Positive' Class : 1
##
```

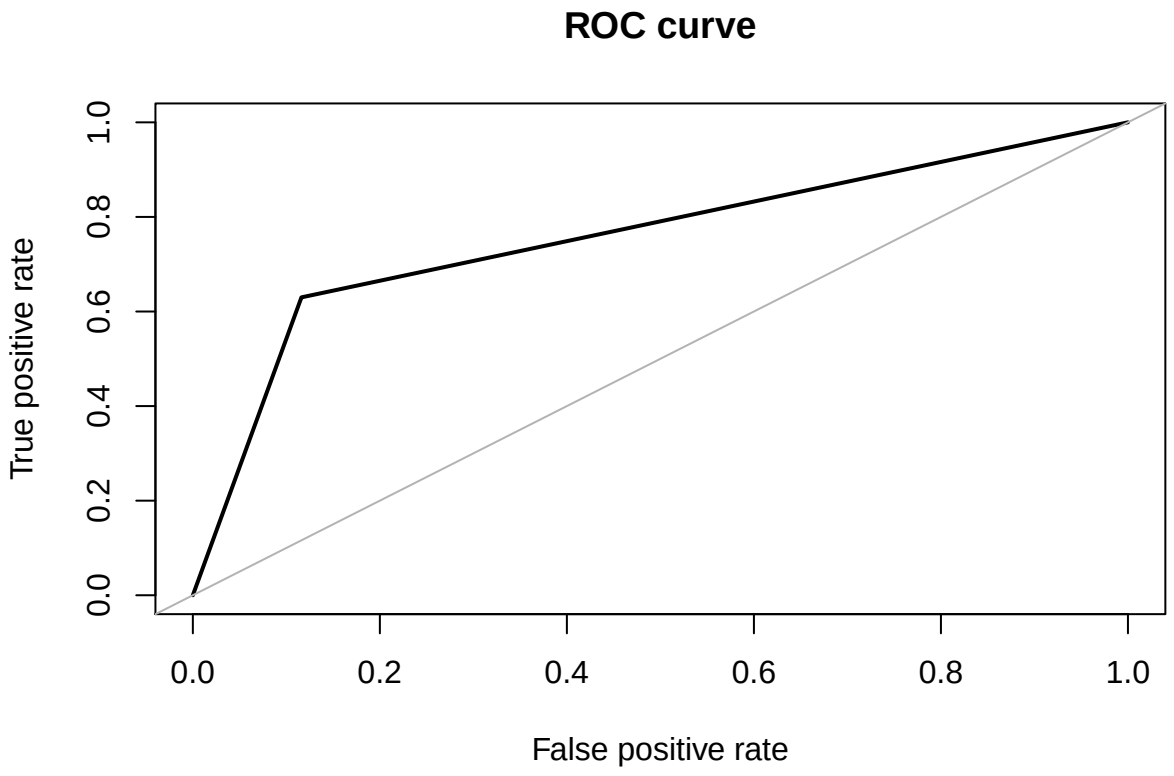
Aquí podemos ver el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.5052462 y 0.7477015 respectivamente.

Ahora intentaremos mejorar estas medidas buscando una parrilla con mejores parámetros utilizando validación cruzda.

```
gamma <- c(10^seq(-6,-1, by=1))
cost <-c(0.01,0.1,1,5,10,50,100)

tuned = tune.svm(y~., data = banco_train_balanced, gamma = gamma, cost = cost, tunecontrol=tune.control
```

Se ha obtenido que los parámetros optimos son $\gamma = 0.01$ y $C = 100$. Creemos el modelo con estos parámetros utilizando un kernel lineal.

```
learn_imp_svm <- svm(y~., data=banco_train_balanced,
                     cost=100, gamma=0.01)

pre_imp_svm <- predict(learn_imp_svm,banco_test)
auc_svm_cv <- roc.curve(banco_test$y, pre_imp_svm)$auc
```



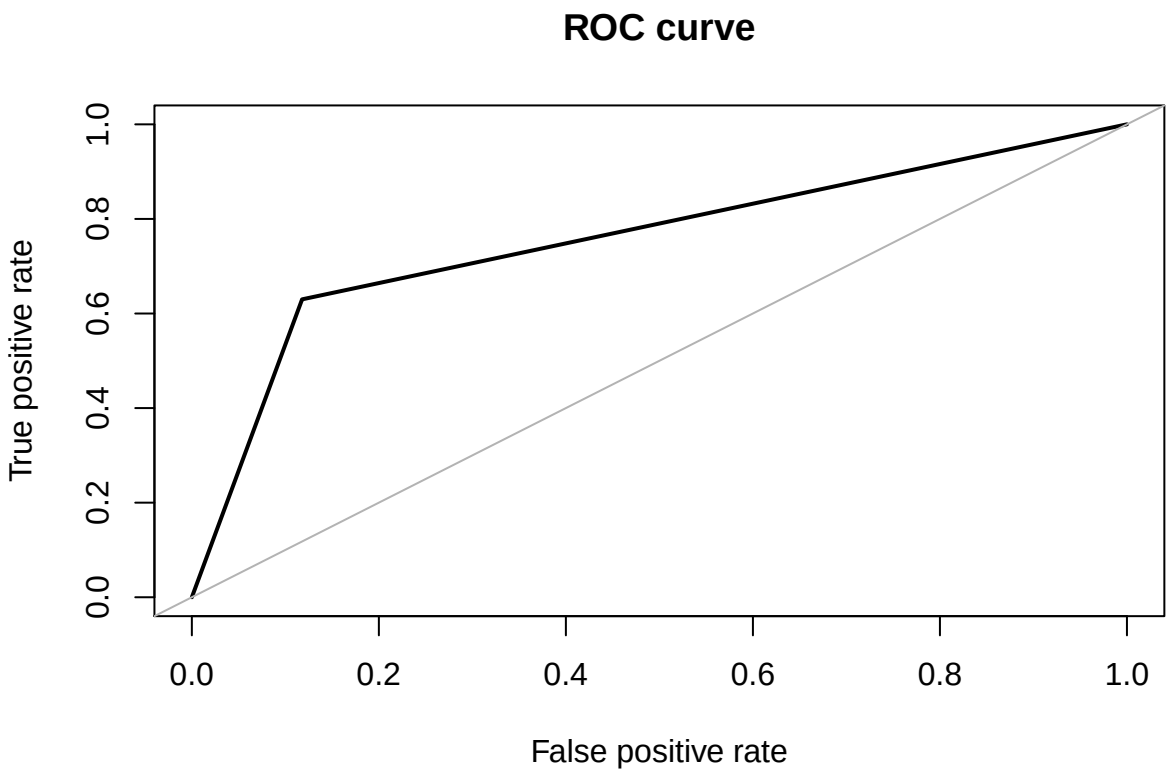
```
f_medida_svm_cv <-  F_meas(pre_imp_svm, banco_test$y, relevant = "1")
cm_imp_svm_cv <- confusionMatrix(pre_imp_svm, banco_test$y, positive = "1")
cm_imp_svm_cv
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10398   581
##          1  1366   989
##
##                Accuracy : 0.854
##                  95% CI : (0.8479, 0.8599)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4223
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62994
##             Specificity : 0.88388
##          Pos Pred Value : 0.41996
##          Neg Pred Value : 0.94708
##              Prevalence : 0.11774
##          Detection Rate : 0.07417
##    Detection Prevalence : 0.17662
##       Balanced Accuracy : 0.75691
##
##        'Positive' Class : 1
##
```

Aquí podemos ver el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.503949 y 0.7569097 respectivamente.

Veamos si variando ligeramente el parámetro γ obtenemos un mejor modelo.

```
learn_imp_svm <- svm(y~., data=banco_train_balanced,
                     cost=100, gamma=0.015)

pre_imp_svm <- predict(learn_imp_svm,banco_test)
auc_svm_cv_g1 <- roc.curve(banco_test$y, pre_imp_svm)$auc
```



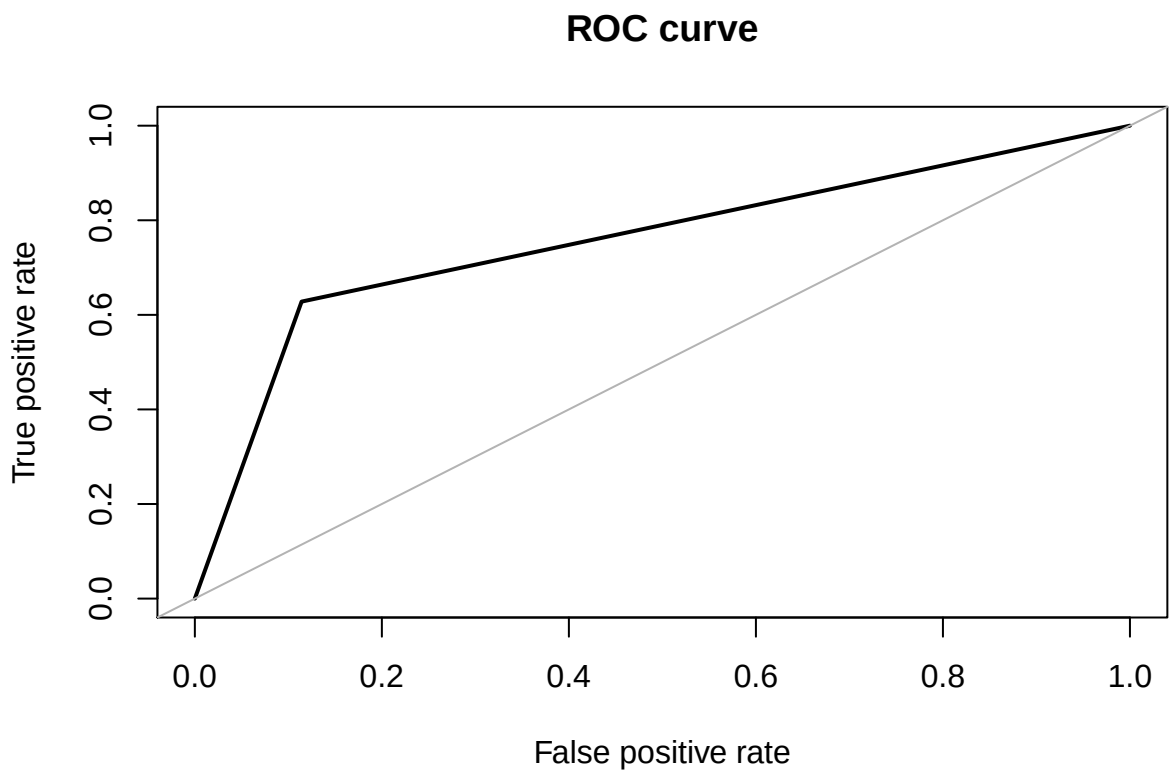

```
f_medida_svm_cv_g1 <-  F_meas(pre_imp_svm, banco_test$y, relevant = "1")
cm_imp_svm_cv_g1 <- confusionMatrix(pre_imp_svm, banco_test$y, positive = "1")
cm_imp_svm_cv_g1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10377   581
##          1  1387   989
##
##                Accuracy : 0.8524
##                  95% CI : (0.8463, 0.8584)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4189
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62994
##             Specificity : 0.88210
##          Pos Pred Value : 0.41625
##          Neg Pred Value : 0.94698
##              Prevalence : 0.11774
```

```
##          Detection Rate : 0.07417
##    Detection Prevalence : 0.17819
##       Balanced Accuracy : 0.75602
##
##        'Positive' Class : 1
##
```

Podemos ver el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.5012671 y 0.7560171 respectivamente.

```
learn_imp_svm <- svm(y~., data=banco_train_balanced,
                     cost=100, gamma=0.02)

pre_imp_svm <- predict(learn_imp_svm,banco_test)
auc_svm_cv_g2 <- roc.curve(banco_test$y, pre_imp_svm)$auc
```



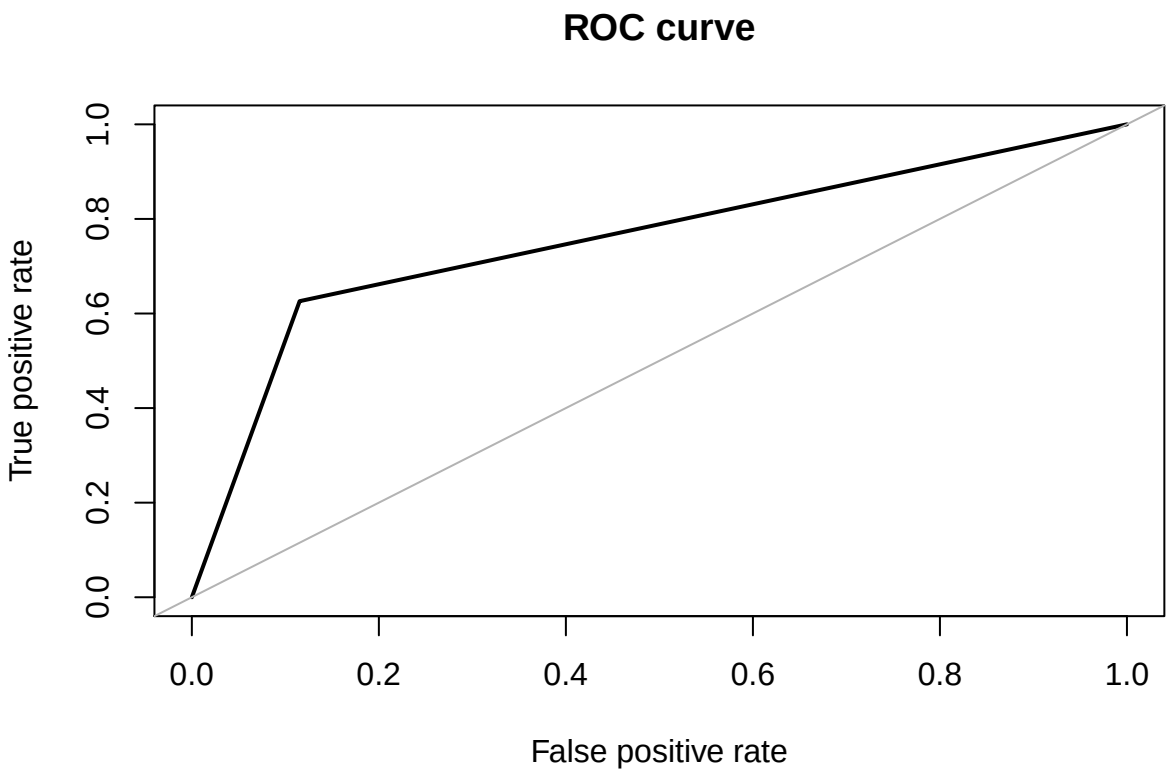

```
f_medida_svm_cv_g2 <-  F_meas(pre_imp_svm, banco_test$y, relevant = "1")
cm_imp_svm_cv_g2 <- confusionMatrix(pre_imp_svm, banco_test$y, positive = "1")
cm_imp_svm_cv_g2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10421   584
```

```
##          1  1343   986
##
##                Accuracy : 0.8555
##                  95% CI : (0.8494, 0.8614)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4249
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62803
##             Specificity : 0.88584
##          Pos Pred Value : 0.42336
##          Neg Pred Value : 0.94693
##              Prevalence : 0.11774
##          Detection Rate : 0.07395
##    Detection Prevalence : 0.17467
##       Balanced Accuracy : 0.75693
##
##        'Positive' Class : 1
##
```

Podemos ver el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.5057707 y 0.7569318 respectivamente.

```
learn_imp_svm <- svm(y~., data=banco_train_balanced,
                     cost=100, gamma=0.0095)

pre_imp_svm <- predict(learn_imp_svm,banco_test)
auc_svm_cv_g3 <- roc.curve(banco_test$y, pre_imp_svm)$auc
```




```
f_medida_svm_cv_g3 <-  F_meas(pre_imp_svm, banco_test$y, relevant = "1")
cm_imp_svm_cv_g3 <- confusionMatrix(pre_imp_svm, banco_test$y, positive = "1")
cm_imp_svm_cv_g3
```

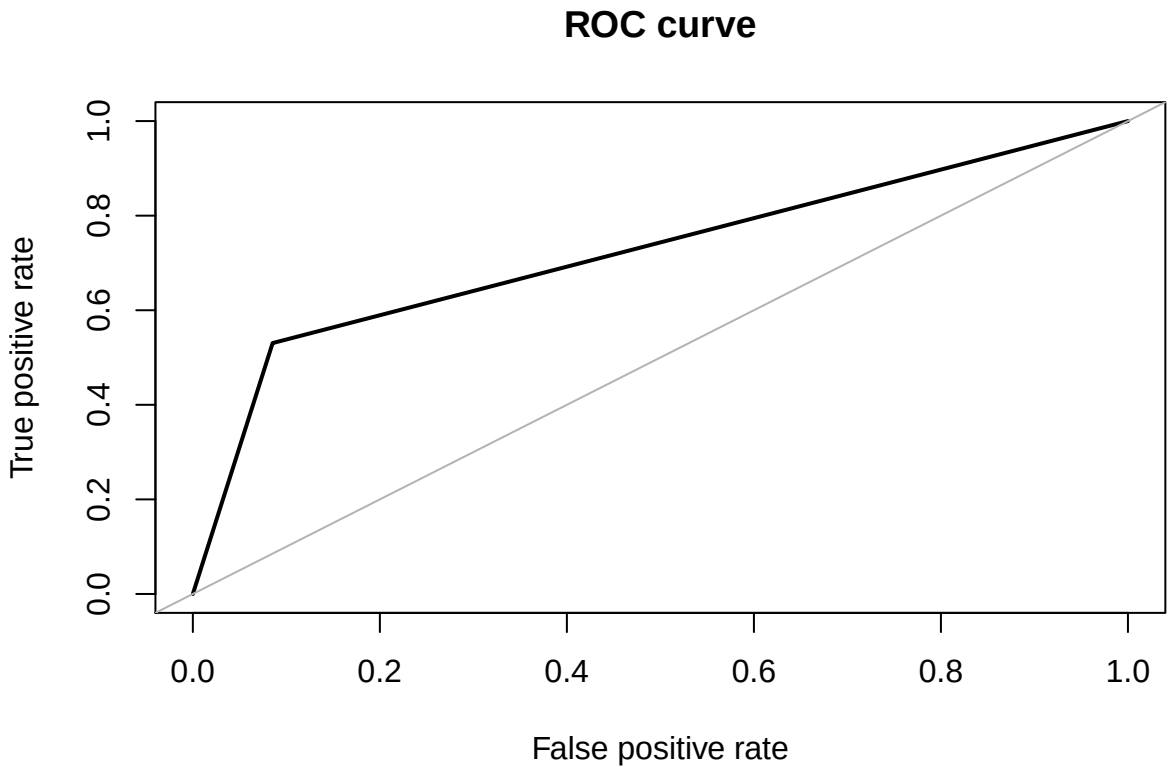
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10408   587
##          1  1356   983
##
##                Accuracy : 0.8543
##                  95% CI : (0.8482, 0.8602)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4214
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62611
##             Specificity : 0.88473
##          Pos Pred Value : 0.42027
##          Neg Pred Value : 0.94661
##              Prevalence : 0.11774
```

```
##          Detection Rate : 0.07372
##    Detection Prevalence : 0.17542
##       Balanced Accuracy : 0.75542
##
##        'Positive' Class : 1
##
```

Podemos observar el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.5057707 y 0.7569318 respectivamente.

Probemos ahora si cambiando a diferentes kernels podemos obtener un mejor modelo.

```
learn_imp_svm <- svm(y~., data=banco_train_balanced,
                     cost=100, gamma=0.01, kernel = "polynomial", degree = 3)

pre_imp_svm <- predict(learn_imp_svm,banco_test)
auc_svm_cv_k1 <- roc.curve(banco_test$y, pre_imp_svm)$auc
```



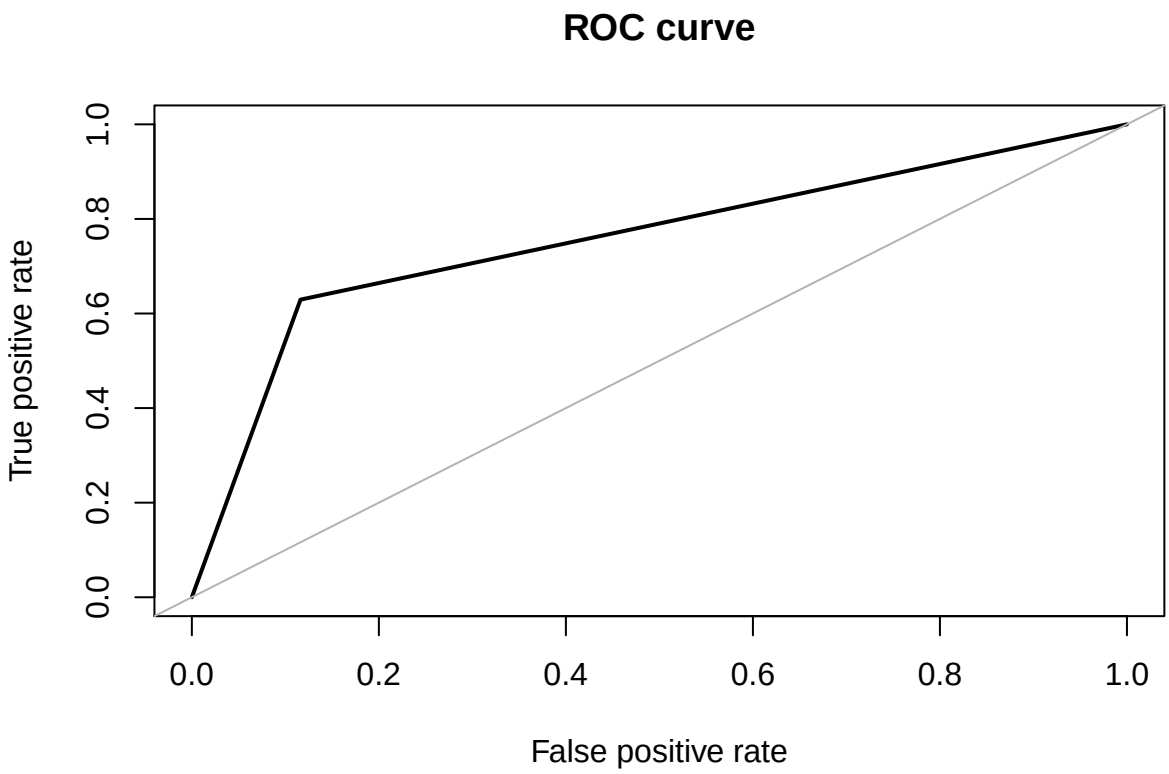

```
f_medida_svm_cv_k1 <-  F_meas(pre_imp_svm, banco_test$y, relevant = "1")
cm_imp_svm_cv_k1 <- confusionMatrix(pre_imp_svm, banco_test$y, positive = "1")
cm_imp_svm_cv_k1
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction     0     1
##          0 10760   737
##          1  1004   833
##
##                Accuracy : 0.8694
##                  95% CI : (0.8636, 0.8751)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4147
##
##  Mcnemar's Test P-Value : 1.829e-10
##
##             Sensitivity : 0.53057
##             Specificity : 0.91465
##          Pos Pred Value : 0.45346
##          Neg Pred Value : 0.93590
##              Prevalence : 0.11774
##          Detection Rate : 0.06247
##    Detection Prevalence : 0.13777
##       Balanced Accuracy : 0.72261
##
##        'Positive' Class : 1
##
```

Podemos ver el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.4889932 y 0.7226141 respectivamente.

```
learn_imp_svm <- ksvm(y~., data=banco_train_balanced,
                      C=100, kernel = "rbfdot")

pre_imp_svm <- predict(learn_imp_svm,banco_test)
auc_svm_cv_k2 <- roc.curve(banco_test$y, pre_imp_svm)$auc
```




```
f_medida_svm_cv_k2 <-  F_meas(pre_imp_svm, banco_test$y, relevant = "1")
cm_imp_svm_cv_k2 <- confusionMatrix(pre_imp_svm, banco_test$y, positive = "1")
cm_imp_svm_cv_k2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10397   582
##          1  1367   988
##
##                Accuracy : 0.8538
##                  95% CI : (0.8477, 0.8598)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4217
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6293
##             Specificity : 0.8838
##          Pos Pred Value : 0.4195
##          Neg Pred Value : 0.9470
##              Prevalence : 0.1177
```

```
##          Detection Rate : 0.0741
##    Detection Prevalence : 0.1766
##       Balanced Accuracy : 0.7565
##
##        'Positive' Class : 1
##
```

Podemos ver el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.5034395 y 0.7565487 respectivamente.

Por último en los modelos de svm, vamos a obtener los parámetros óptimos sin utilizar validación cruzada a ver si conseguimos un modelo con mejores medidas.

```
parms<-expand.grid(cost=cost,gamma=gamma)
gamma <- c(10^seq(-6,-1, by=1))
cost <-c(0.01,0.1,1,5,10,50,100)

total_accuracy_svm <- function(trainset, testset){
  f_medida <- NULL; auc <- NULL
  for(i in 1:NROW(parms)){
    learn_svm <- svm(y~., data = trainset,gamma=parms$gamma[i], cost=parms$cost[i])
    pre_svm <- predict(learn_svm, testset)
    f_medida[i] <- F_meas(pre_svm,testset$y, relevant = "1")
    auc[i] <- roc.curve(testset$y, pre_svm, plotit = F)$auc
  }
  f_medida
}

c <- total_accuracy_svm(banco_train_balanced,banco_test)
opt_parms <- which(c==max(c))[1]

learn_imp_svm <- svm(y~., data=banco_train_balanced,
                     cost=parms$cost[opt_parms], gamma=parms$gamma[opt_parms])
summary(learn_imp_svm)
pre_imp_svm <- predict(learn_imp_svm,banco_test)
cm_imp_svm <- confusionMatrix(pre_imp_svm, banco_test$y, positive = "1")
cm_imp_svm
```

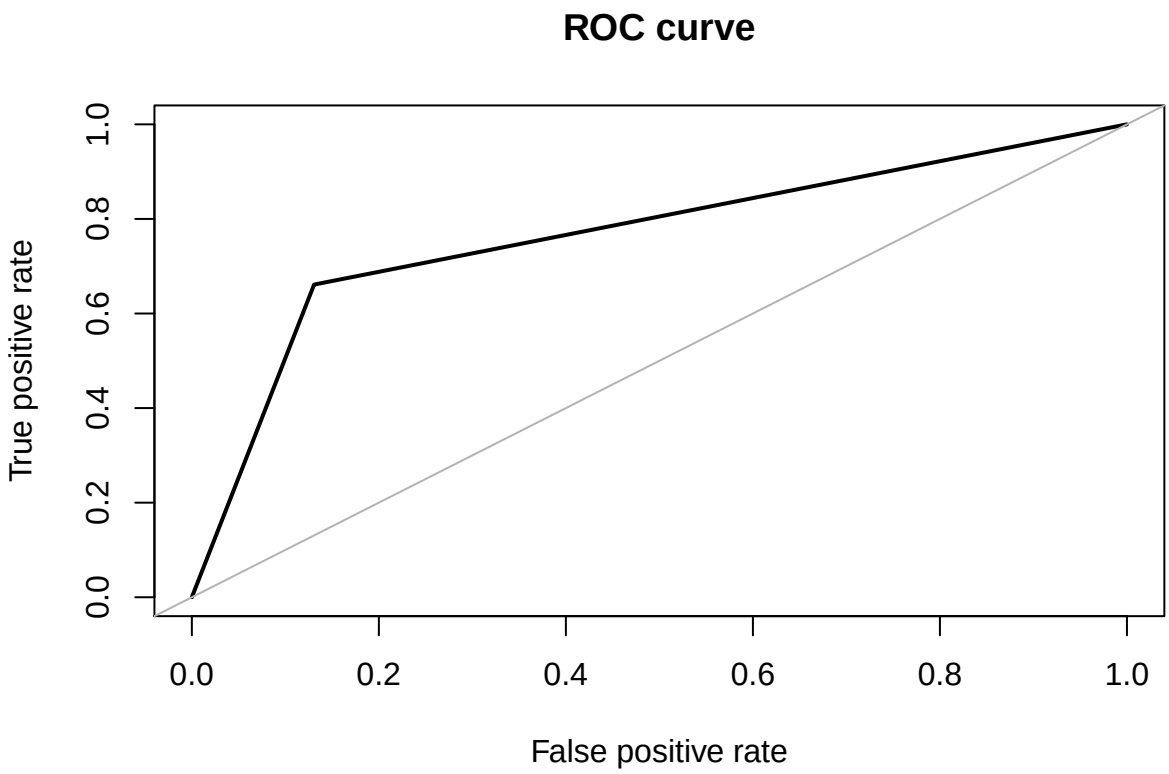
Los parámetros óptimos obtenidos son $C = 100$, $\gamma = 0.01$ igual que los obtenidos con validación cruzada.

4.2 Otros modelos

4.2.1 Arbol de clasificación

El primer modelo diferente a los que utilizan máquinas de soporte vectorial será el arbol de clasificación.

```
tree <- rpart(y ~ ., data = banco_train_balanced, method = 'class', cp = 1e-3)
tree.pred <- predict(tree, newdata = banco_test, type = 'class')

auc_tree<- roc.curve(banco_test$y, tree.pred)$auc
```




```
f_medida_tree <-  F_meas(tree.pred, banco_test$y, relevant = "1")

cm_tree <- confusionMatrix(tree.pred, banco_test$y, positive = "1")
cm_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10226   532
##          1  1538  1038
##
##                Accuracy : 0.8448
##                  95% CI : (0.8385, 0.8509)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4152
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.66115
##             Specificity : 0.86926
##          Pos Pred Value : 0.40295
##          Neg Pred Value : 0.95055
```

```
##              Prevalence : 0.11774
##          Detection Rate : 0.07785
##    Detection Prevalence : 0.19319
##       Balanced Accuracy : 0.76520
##
##        'Positive' Class : 1
##
```

Podemos observar el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.5007236 y 0.7652043 respectivamente.

4.2.2 Regresion lineal

Pasamos a realizar una regresión lineal con todas las variables que hemos seleccionado.

```
m1 <- glm(y ~ ., data = banco_train_balanced, family = binomial('logit'))
summ <- summary(m1)
summ
```

```
##
## Call:
## glm(formula = y ~ ., family = binomial("logit"), data = banco_train_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.4672  -0.6614  -0.4245   0.5929   2.5713
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.86571    0.13035  -6.641 3.11e-11 ***
## age                  0.02251    0.02638   0.853 0.393463
## day                 -0.09590    0.02380  -4.030 5.57e-05 ***
## duration             1.17399    0.02700  43.475  < 2e-16 ***
## month_Third_Trim    -0.89698    0.07831 -11.454  < 2e-16 ***
## job_not_work         0.71896    0.09319   7.715 1.21e-14 ***
## job_other_unknown   -0.02305    0.11854  -0.194 0.845795
## job_services         0.01648    0.11051   0.149 0.881445
## job_white_collar     0.29578    0.07854   3.766 0.000166 ***
## marital_married      0.08939    0.08064   1.108 0.267649
## marital_single       0.38118    0.09116   4.182 2.90e-05 ***
## education_secondary  0.32787    0.08448   3.881 0.000104 ***
## education_tertiary   0.57639    0.09400   6.132 8.70e-10 ***
## education_unknown    0.49858    0.13258   3.761 0.000169 ***
## housing_yes         -1.00114    0.05450 -18.368  < 2e-16 ***
## loan_yes            -0.66738    0.07778  -8.580  < 2e-16 ***
## month_Fourth_Trim   -0.09848    0.09056  -1.087 0.276848
## month_Second_Trim   -0.70877    0.07590  -9.339  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 14930  on 12219  degrees of freedom
```

```
## Residual deviance: 10961  on 12202  degrees of freedom
## AIC: 10997
##
## Number of Fisher Scoring iterations: 5
```

Podemos ver que la medida AIC de la regresión es 1.099667×10^4. Veamos si aplicando *"backward selection"* o *"forward selection"* podemos obtener un valor de AIC más bajo. Primero aplicaremos *"backward selection"*.

```
m_full <- m1
m_null <- glm(y ~ 1, data = banco_train_balanced, family = binomial('logit'))

back<-step(m_full, trace = F, scope = list(lower=formula(m_null), upper=formula(m_full)),
     direction = 'backward')

back
```

```
##
## Call:  glm(formula = y ~ day + duration + month_Third_Trim + job_not_work +
##     job_white_collar + marital_single + education_secondary +
##     education_tertiary + education_unknown + housing_yes + loan_yes +
##     month_Second_Trim, family = binomial("logit"), data = banco_train_balanced)
##
## Coefficients:
##         (Intercept)                  day             duration
##             -0.8233              -0.1000               1.1729
##    month_Third_Trim         job_not_work     job_white_collar
##             -0.8472               0.7358               0.2964
##      marital_single  education_secondary   education_tertiary
##              0.2851               0.3190               0.5648
##   education_unknown          housing_yes             loan_yes
##              0.4986              -1.0075              -0.6703
##   month_Second_Trim
##             -0.6622
##
## Degrees of Freedom: 12219 Total (i.e. Null);  12207 Residual
## Null Deviance:       14930
## Residual Deviance: 10960     AIC: 10990
```

Podemos ver que la medida AIC de la regresión utilizando *"backward selection"* es 1.0989674×10^4. Veamos que ocurre ahora utilizando *"forward selection"*

```
forward <- step(m_null, trace = F, scope = list(lower=formula(m_null), upper=formula(m_full)),
     direction = 'forward')

forward
```

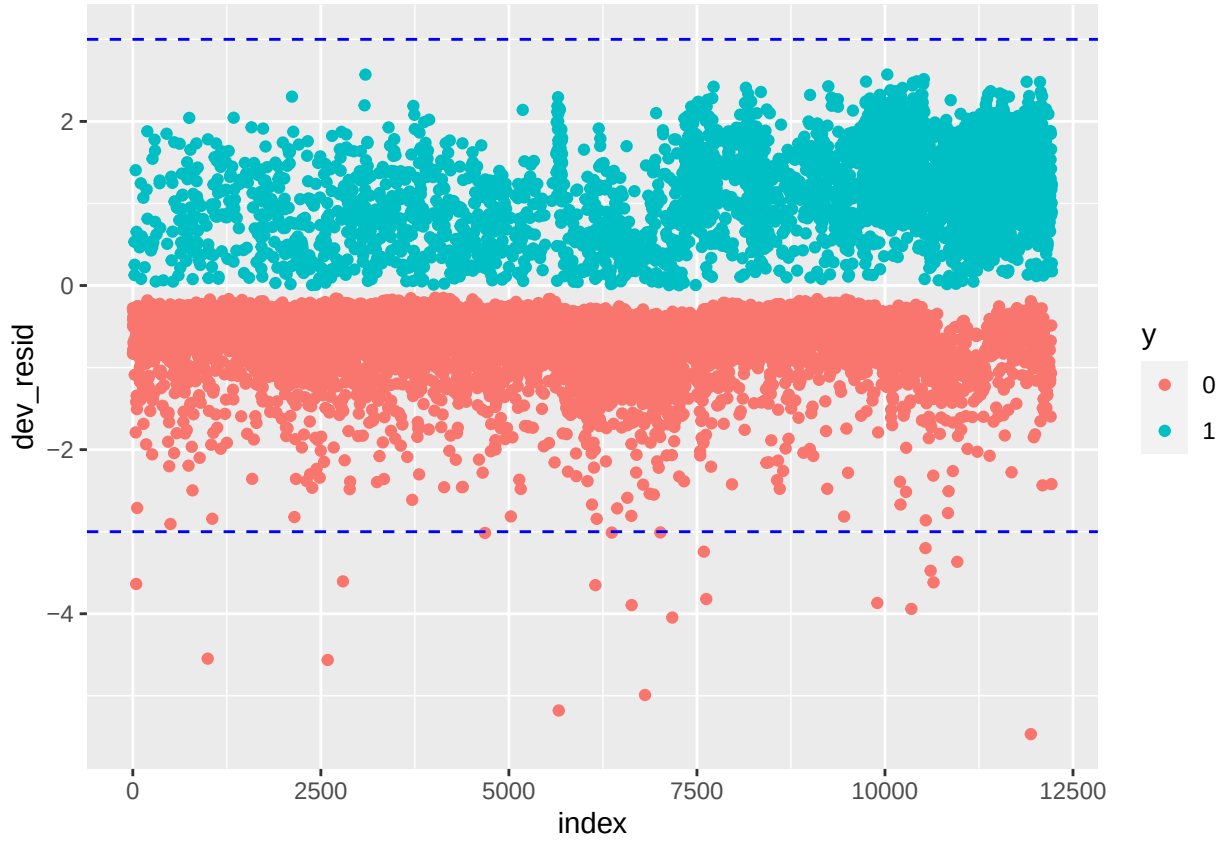
```
##
## Call:  glm(formula = y ~ duration + housing_yes + loan_yes + month_Third_Trim +
##     month_Second_Trim + marital_single + job_not_work + job_white_collar +
##     education_tertiary + day + education_secondary + education_unknown,
##     family = binomial("logit"), data = banco_train_balanced)
##
## Coefficients:
```

```
##         (Intercept)             duration          housing_yes
##             -0.8233               1.1729              -1.0075
##            loan_yes     month_Third_Trim    month_Second_Trim
##             -0.6703              -0.8472              -0.6622
##      marital_single         job_not_work     job_white_collar
##              0.2851               0.7358               0.2964
##  education_tertiary                  day  education_secondary
##              0.5648              -0.1000               0.3190
##   education_unknown
##              0.4986
##
## Degrees of Freedom: 12219 Total (i.e. Null);  12207 Residual
## Null Deviance:       14930
## Residual Deviance: 10960     AIC: 10990
```

Vemos que la medida AIC de la regresión utilizando *"forward selection"* es 1.0989674×10^4. Vemos que en los tres casos, la medida AIC es muy similar, siendo mejor utilziando el método de *"backward selection"*.

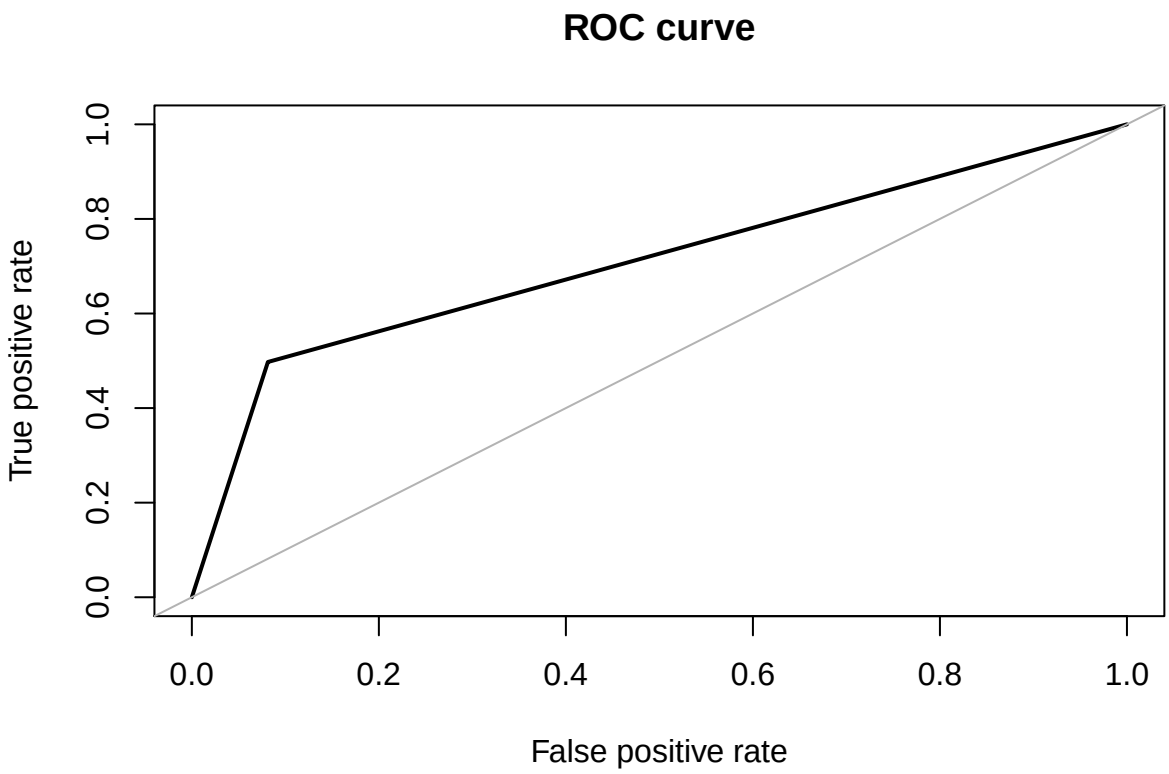
Veamos gráficamente la desviación de los residuos de la regresión.

```
index <- 1:dim(banco_train_balanced)[1]
dev_resid <- residuals(m1)
y <- banco_train_balanced$y
dff <- data.frame(index, dev_resid, y)

ggplot(dff, aes(x = index, y = dev_resid, color = y)) +
  geom_point() +
  geom_hline(yintercept = 3, linetype = 'dashed', color = 'blue') +
  geom_hline(yintercept = -3, linetype = 'dashed', color = 'blue')
```



Por último, veamos la matriz de confusión y las medidas *F_medida* y *AUC* de la regresión.

```
prob <- predict(m1, banco_test, type = 'response')
pred <- rep('0', length(prob))
pred[prob>=.5] <- '1'

auc_glm <- roc.curve(banco_test$y, as.factor(pred))$auc
```




```
f_medida_glm <-  F_meas(as.factor(pred), banco_test$y, relevant = "1")

cm_glm <- confusionMatrix(as.factor(pred),banco_test$y, positive = "1")
cm_glm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10807   789
##          1   957   781
##
##                Accuracy : 0.8691
##                  95% CI : (0.8632, 0.8747)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3977
##
##  Mcnemar's Test P-Value : 6.425e-05
##
##             Sensitivity : 0.49745
##             Specificity : 0.91865
##          Pos Pred Value : 0.44937
##          Neg Pred Value : 0.93196
```

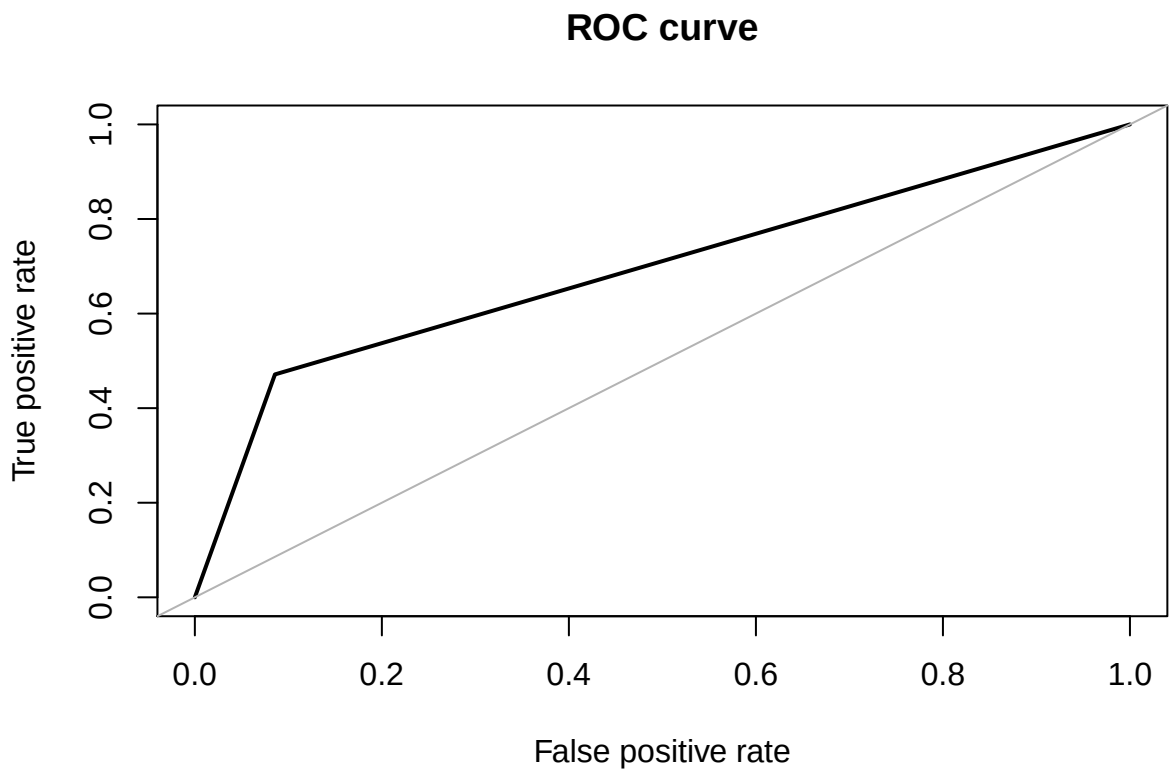
```
##              Prevalence : 0.11774
##          Detection Rate : 0.05857
##    Detection Prevalence : 0.13034
##       Balanced Accuracy : 0.70805
##
##        'Positive' Class : 1
##
```

Podemos observar el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.4721886 y 0.7080512 respectivamente.

4.2.3 Arbol de decisión

Ahora vamos a construir un modelo utilizando un arbol de decisión.

```
tree.model<- rpart(y~., data=banco_train_balanced, method="class", minbucket=20)
tree.predict<- predict(tree.model, banco_test, type = "class")

auc_tree2 <- roc.curve(banco_test$y, tree.predict)$auc
```




```
f_medida_tree2 <-  F_meas(tree.predict, banco_test$y, relevant = "1")

confusionMatrix(tree.predict,banco_test$y, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10756   830
##          1  1008   740
##
##                Accuracy : 0.8622
##                  95% CI : (0.8562, 0.868)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3676
##
##  Mcnemar's Test P-Value : 3.65e-05
##
##             Sensitivity : 0.4713
##             Specificity : 0.9143
##          Pos Pred Value : 0.4233
##          Neg Pred Value : 0.9284
##              Prevalence : 0.1177
##          Detection Rate : 0.0555
##    Detection Prevalence : 0.1311
##       Balanced Accuracy : 0.6928
##
##        'Positive' Class : 1
##
```

Podemos observar el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.4460518 y 0.6928262 respectivamente.

4.2.4 Redes neuronales

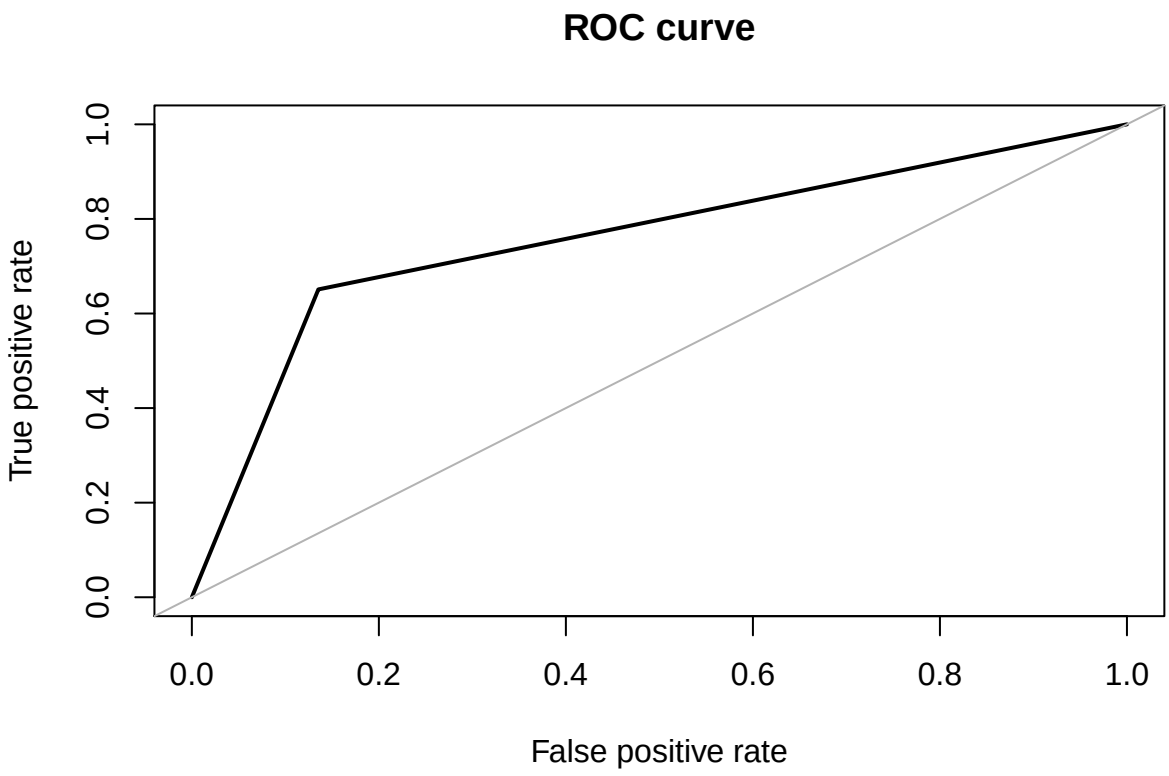
El siguiente modelo será una red neuronal. Tendrá una capa con 20 neuronas y un máximo de iteraciones para encontrar los pesos óptimos de 500.

```
set.seed(3456)
nn1 <- nnet(y ~ ., data = banco_train_balanced, size = 20, maxit = 500)
```

```
## # weights:  381
## initial  value 7188.610634
## iter  10 value 5193.745490
## iter  20 value 4942.532143
## iter  30 value 4810.799293
## iter  40 value 4705.925730
## iter  50 value 4610.336409
## iter  60 value 4521.873043
## iter  70 value 4439.685433
## iter  80 value 4367.958807
## iter  90 value 4320.448825
## iter 100 value 4284.301547
## iter 110 value 4255.577857
## iter 120 value 4226.252697
```

```
## iter 130 value 4199.810335
## iter 140 value 4177.522894
## iter 150 value 4152.876343
## iter 160 value 4129.215176
## iter 170 value 4097.688977
## iter 180 value 4062.618320
## iter 190 value 4031.887620
## iter 200 value 3998.474415
## iter 210 value 3971.868625
## iter 220 value 3958.499585
## iter 230 value 3950.033869
## iter 240 value 3945.995100
## iter 250 value 3942.970571
## iter 260 value 3941.906283
## iter 270 value 3941.164791
## iter 280 value 3940.639854
## iter 290 value 3940.178448
## iter 300 value 3939.750174
## iter 310 value 3939.599402
## iter 320 value 3939.257889
## iter 330 value 3939.178898
## iter 340 value 3939.102148
## iter 350 value 3939.057689
## iter 360 value 3939.032416
## iter 370 value 3939.009944
## iter 380 value 3938.979281
## iter 380 value 3938.979276
## final  value 3938.979171
## converged
```

```r
nn1.pred <- predict(nn1, banco_test, type = "class")

auc_nn <- roc.curve(banco_test$y, nn1.pred)$auc
```




```
f_medida_nn <-  F_meas(as.factor(nn1.pred), banco_test$y, relevant = "1")

confusionMatrix(as.factor(nn1.pred),banco_test$y , positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10172   548
##          1  1592  1022
##
##                Accuracy : 0.8395
##                  95% CI : (0.8332, 0.8457)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4003
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.65096
##             Specificity : 0.86467
##          Pos Pred Value : 0.39097
##          Neg Pred Value : 0.94888
##              Prevalence : 0.11774
```

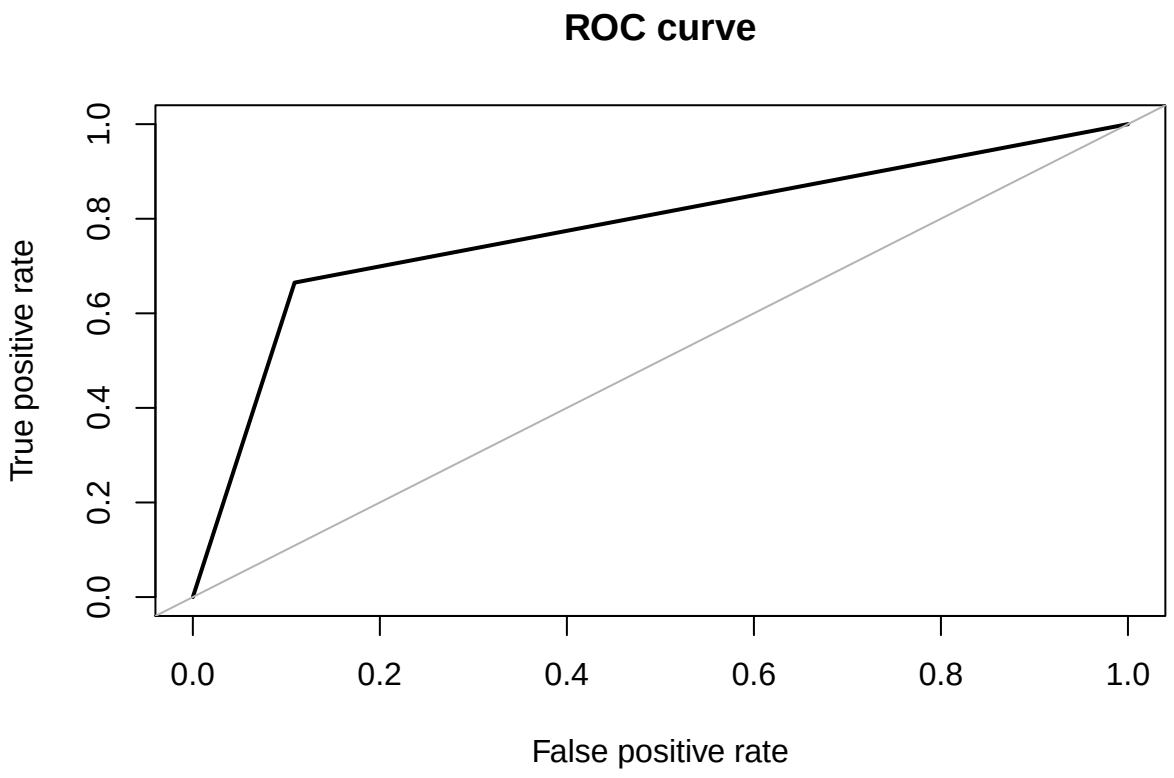
```
##          Detection Rate : 0.07665
##    Detection Prevalence : 0.19604
##       Balanced Accuracy : 0.75781
##
##        'Positive' Class : 1
##
```

Podemos observar el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.4885277 y 0.7578136 respectivamente.

4.2.5 Bosques aleatorios

Continuamos con un modelo basado en arboles aleatorios. El número de arboles será 2000

```
rf3 <- randomForest(y ~ ., data = banco_train_balanced, ntree = 2000)
rf3.pred <- predict(rf3, newdata = banco_test, type = 'class')

auc_rf <- roc.curve(banco_test$y, rf3.pred)$auc
```




```
f_medida_rf <-  F_meas(as.factor(rf3.pred), banco_test$y, relevant = "1")

confusionMatrix(rf3.pred,banco_test$y, positive = "1")
```

```
## Confusion Matrix and Statistics
```

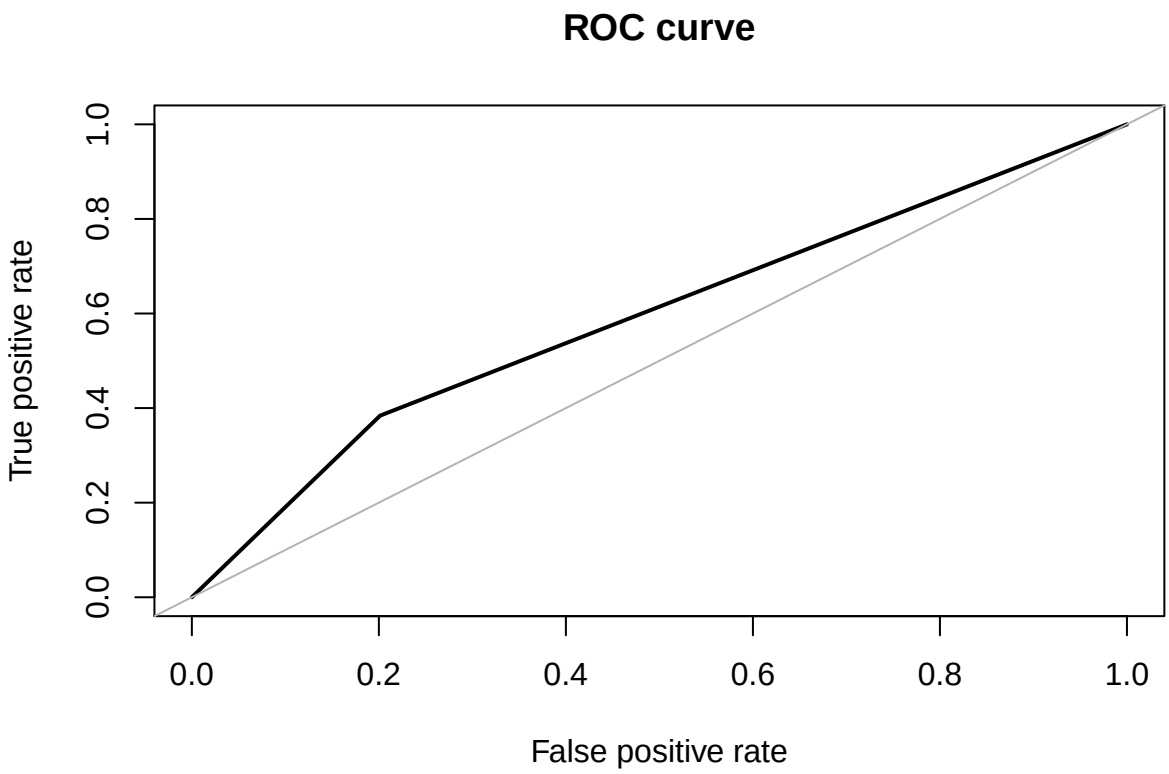
```
##
##           Reference
## Prediction     0     1
##          0 10484   526
##          1  1280  1044
##
##                Accuracy : 0.8646
##                  95% CI : (0.8586, 0.8703)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4604
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6650
##             Specificity : 0.8912
##          Pos Pred Value : 0.4492
##          Neg Pred Value : 0.9522
##              Prevalence : 0.1177
##          Detection Rate : 0.0783
##    Detection Prevalence : 0.1743
##       Balanced Accuracy : 0.7781
##
##        'Positive' Class : 1
##
```

Podemos observar el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.5362096 y 0.7780808 respectivamente.

4.2.6 kNN

Por último, crearemos un modelo k-NN, utilizando $k = 100$.

```
prc_test_pred <- knn(train = banco_train_balanced, test = banco_test,cl = banco_train_balanced$y, k=100)

auc_knn <- roc.curve(banco_test$y, prc_test_pred)$auc
```




```
f_medida_knn <-  F_meas(as.factor(prc_test_pred), banco_test$y, relevant = "1")

confusionMatrix(prc_test_pred, banco_test$y, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9395  967
##          1 2369  603
##
##                Accuracy : 0.7498
##                  95% CI : (0.7424, 0.7571)
##     No Information Rate : 0.8823
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1317
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.38408
##             Specificity : 0.79862
##          Pos Pred Value : 0.20289
##          Neg Pred Value : 0.90668
##              Prevalence : 0.11774
```

```
##          Detection Rate : 0.04522
##    Detection Prevalence : 0.22289
##       Balanced Accuracy : 0.59135
##
##        'Positive' Class : 1
##
```

Podemos observar el gráfico de la curva roc y la matriz de confusión. Para este modelo, los valores de la f-medida y la medida AUC son 0.2655218 y 0.5913497 respectivamente.

5 Conclusiones

Para ver que modelo seleccionamos de todos los que hemos creado, haremos una tabla con las medidas seleccionadas en un principio.

```
tabla_resumen <- data.frame("svm_inicial" = c(f_medida_1, auc_1),
                            "svm_100_0.01" = c(f_medida_svm_cv, auc_svm_cv),
                            "svm_100_0.015" = c(f_medida_svm_cv_g1, auc_svm_cv_g1),
                            "svm_100_0.02" = c(f_medida_svm_cv_g2, auc_svm_cv_g2),
                            "svm_100_0.0095" = c(f_medida_svm_cv_g2, auc_svm_cv_g3),
                            "svm_100_poly" = c(f_medida_svm_cv_k1, auc_svm_cv_k1),
                            "svm_100_rbfdot" = c(f_medida_svm_cv_k2, auc_svm_cv_k2),
                            "class_tree" = c(f_medida_tree, auc_tree),
                            "reg_lineal" = c(f_medida_glm, auc_glm),
                            "dec_tree" = c(f_medida_tree2, auc_tree2),
                            "red_neuronal" = c(f_medida_nn, auc_nn),
                            "random_forest" = c(f_medida_rf, auc_rf),
                            "knn" = c(f_medida_knn, auc_knn),row.names = c("F-medida", "AUC"))

tabla_resumen <- t(tabla_resumen)

kable( tabla_resumen , caption = "Tabla resumen modelos"
       , row.names = TRUE
      )
```

Table 1: Tabla resumen modelos

	F-medida	AUC
svm_inicial	0.5052462	0.7477015
svm_100_0.01	0.5039490	0.7569097
svm_100_0.015	0.5012671	0.7560171
svm_100_0.02	0.5057707	0.7569318
svm_100_0.0095	0.5057707	0.7554239
svm_100_poly	0.4889932	0.7226141
svm_100_rbfdot	0.5034395	0.7565487
class_tree	0.5007236	0.7652043
reg_lineal	0.4721886	0.7080512
dec_tree	0.4460518	0.6928262
red_neuronal	0.4885277	0.7578136
random_forest	0.5362096	0.7780808
knn	0.2655218	0.5913497

Viendo la tabla resumen de los diferentes modelos, podemos decir que el que mejor resultados con respecto a la F-medida y a la medida AUC es el de bosques aleatorios por lo que es el que seleccionaríamos. Añadir que los modelos que utilizan máquinas de soporte vectorial no se quedan muy lejos del de bosques aleatorios si miramos la F-medida y la medida AUC. Podemos observar también que cambiar el kernel de las máquinas de soporte vectorial no ha mejorado el modelo.