SVM: Práctica de clasificación

Jun De Wu

12/06/2021

Contenidos


```
require(tidyverse)
require(ggcorrplot)
require(fastDummies)
require(unbalanced)
require(caTools)
require(neuralnet)
require(caret)
require(moments)
require(e1071)
require(kernlab)
require(class)
require(rpart)
require(MASS)
require(randomForest)
```

1 Introducción

El data set que tratamos es un conjunto de datos relacionado con campañas directas de marketing de una institución bancaria portuguesa. Las campañas de marketing se basaban en llamadas telefónicas. Usualmente, se requiere más de una llamada con el mismo cliente para saber si el producto (depósito bancario a plazo) sería suscrito o no. El objetivo es predecir, con todas las variables y observaciones que disponemos, los clientes potenciales que se suscriben al depósito a plazo.

2 Exploración y limpieza de los datos

```
bank <- read_csv2("bank-full.csv", col_names = TRUE)
glimpse(bank)
```

```
## Rows: 45,211
## Columns: 17
## $ age       <dbl> 58, 44, 33, 47, 33, 35, 28, 42, 58, 43, 41, 29, 53, 58, 5...
## $ job       <chr> "management", "technician", "entrepreneur", "blue-collar"...
## $ marital   <chr> "married", "single", "married", "married", "single", "mar...
## $ education <chr> "tertiary", "secondary", "secondary", "unknown", "unknown...
## $ default   <chr> "no", "no", "no", "no", "no", "no", "no", "yes", "no", "n...
## $ balance   <dbl> 2143, 29, 2, 1506, 1, 231, 447, 2, 121, 593, 270, 390, 6,...
## $ housing   <chr> "yes", "yes", "yes", "yes", "no", "yes", "yes", "yes", "y...
## $ loan      <chr> "no", "no", "yes", "no", "no", "no", "yes", "no", "no", "...
## $ contact   <chr> "unknown", "unknown", "unknown", "unknown", "unknown", "u...
## $ day       <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ month     <chr> "may", "may", "may", "may", "may", "may", "may", "may", "...
## $ duration  <dbl> 261, 151, 76, 92, 198, 139, 217, 380, 50, 55, 222, 137, 5...
## $ campaign  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ pdays     <dbl> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -...
## $ previous  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ poutcome  <chr> "unknown", "unknown", "unknown", "unknown", "unknown", "u...
## $ y         <chr> "no", "no", "no", "no", "no", "no", "no", "no", "no", "no...
```

Las variables que tenemos sobre los clientes son:

- `age`: Edad
- `job`: Trabajo
- `marital`: Estado civil
- `education`: Nivel de educación
- `default`: Impago
- `balance`: Saldo medio anual en la cuenta
- `housing`: Préstamo de vivienda
- `loan`: Préstamos personales

Las variables relacionas con la última llamada de la campaña actual:

- `contact`: Vía de comunicación
- `day`: Día del mes de la última llamada
- `month`: Mes del año de la última llamada

- `duration`: Duración de la última llamada en segundos

Las variables relacionadas con otros atributos:

- `campaign`: Número de llamadas realizadas durante esta campaña para este cliente
- `pdays`: Número de días que han pasado desde que el cliente recibiera la última llamada de la anterior campaña (-1 significa que el cliente no ha sido contactado anteriormente)
- `previous`: Número de llamadas realizadas antes de esta campaña para este cliente
- `poutcome`: Resultado de la anterior campaña de marketing

La variable objetivo:

- `y`: Indica si el cliente se ha suscrito a un depósito a plazo

Las variables que no son numéricas son factor y están en formato character, así que tenemos que transformarlas.

```
bank[sapply(bank, is.character)] <- lapply(bank[sapply(bank, is.character)],
                                   as.factor)
summary(bank)
```

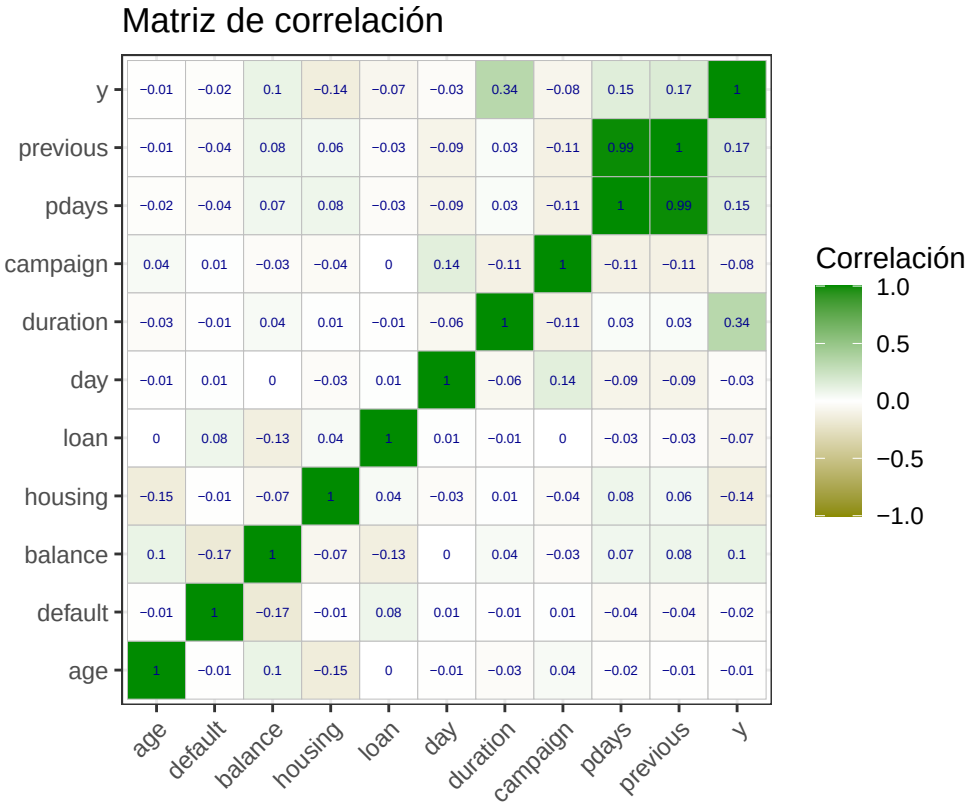
```
##       age                 job           marital          education
##  Min.   :18.00   blue-collar:9732   divorced: 5207   primary  : 6851
##  1st Qu.:33.00   management :9458   married :27214   secondary:23202
##  Median :39.00   technician :7597   single  :12790   tertiary :13301
##  Mean   :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
##
```

```
bank_correlacion <- bank %>%
  mutate(default = as.integer(recode(default, "no" = 0, "yes" = 1)), housing =
           as.integer(recode(housing, "no" = 0, "yes" = 1)), loan =
```

```
          as.integer(recode(loan, "no" = 0, "yes" = 1)),
         y = as.integer(recode(y, "no" = 0, "yes" = 1)))

correlacion <- cor(bank_correlacion[, sapply(bank_correlacion, is.numeric)],
                   method = "spearman")

ggcorrplot(correlacion, lab = TRUE, lab_size = 1.7, legend.title = "Correlación",
           lab_col = "blue4", colors = c("yellow4", "white", "green4")) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1)) +
  labs(x = "", y = "", title = "Matriz de correlación")
```



Hay una correlación de 0.99 entre las variables `previous` y `pdays`, tienen una relación prácticamente lineal. Más tarde, junto con otros métodos que aplicaremos para estudiar las variables, decidiremos cuál de las dos quitar o quitar las dos.

2.1 Variables cualitativas

Vamos a ir recorriendo cada variable cualitativa y realizar un diagrama de barras para reflejar la distribución de cada variable filtrando por la variable objetivo.

2.1.1 Variable `job`

```
levels(bank$job)
```

```
##  [1] "admin."        "blue-collar"   "entrepreneur"  "housemaid"
```

```
##  [5] "management"    "retired"       "self-employed" "services"
##  [9] "student"       "technician"    "unemployed"    "unknown"
```

Tenemos 12 niveles de la variable `job`, demasiadas para poder realizar la visualización como para posteriormente aplicar el algoritmo de ANN. Lo que haremos será agrupar los trabajos por similitud. Empezamos por agrupar en una nueva categoría "Self-employed" los trabajos "admin.", "entrepreneur", "management" y "self-employed".

```
bank$job <- gsub('^admin.', 'Self_employed', bank$job)
bank$job <- gsub('^entrepreneur', 'Self_employed', bank$job)
bank$job <- gsub('^management', 'Self_employed', bank$job)
bank$job <- gsub('^self-employed', 'Self_employed', bank$job)
```

Continuamos por la segunda categoría "Services" que incluiría los trabajos "blue-collar", "housemaid", "services" y "technician".

```
bank$job <- gsub('^blue-collar', 'Services', bank$job)
bank$job <- gsub('^housemaid', 'Services', bank$job)
bank$job <- gsub('^services', 'Services', bank$job)
bank$job <- gsub('^technician', 'Services', bank$job)
```

La última categoría es "No-workers/Unknown" que incluiría los trabajos "retired", "student" y "unemployed".

```
bank$job <- gsub('^retired', 'No_workers', bank$job)
bank$job <- gsub('^student', 'No_workers', bank$job)
bank$job <- gsub('^unemployed', 'No_workers', bank$job)
```

```
bank$job <- as.factor(bank$job)
summary(bank$job)
```

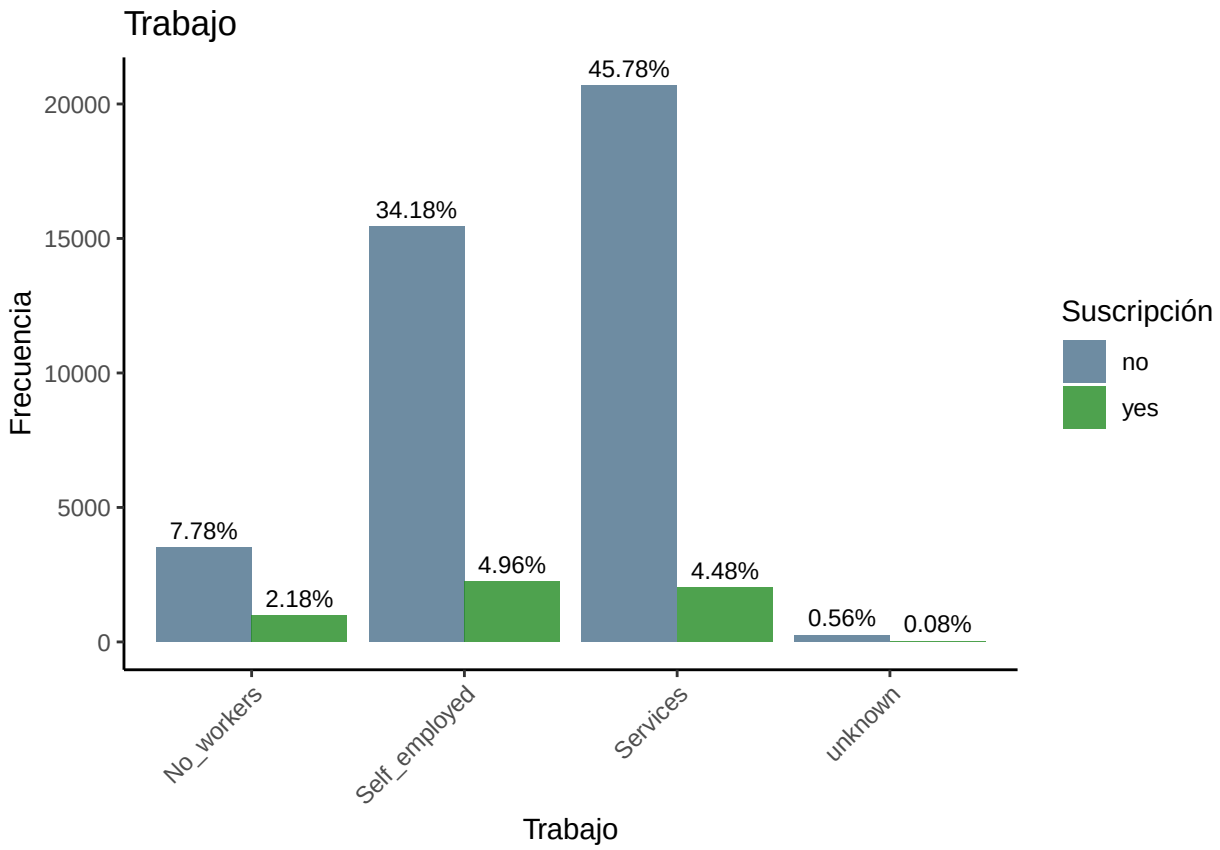
```
##    No_workers Self_employed      Services       unknown
##          4505         17695         22723           288
```

Dejamos la categoría "unknown" porque contiene pocas observaciones y, más adelante, cuando realicemos los modelos de redes neuronales artificiales las quitaremos.

```
bank_1 <- bank
```

Ahora que hemos agrupado los 12 niveles que tenía originalmente la variable, vamos a visualizar con un histograma los datos filtrando por si están suscritos o no al depósito a corto plazo.

```
bank %>%
  ggplot(aes(x = job, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Trabajo", y = "Frecuencia", title = "Trabajo") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



Observamos que, por lo desbalanceado que está el data set, hay un gran número de los entrevistados que no se suscriben. Parece que los perfiles de trabajadores que más se suscriben son los del sector servicio y los que son sus propios jefes.

A continuación, quitamos los entrevistados que no sabemos que ocupación tienen.

```
bank_1 <- bank_1 %>%
  filter(bank_1$job != "unknown") %>%
  droplevels()
```

2.1.2 Variable `marital`

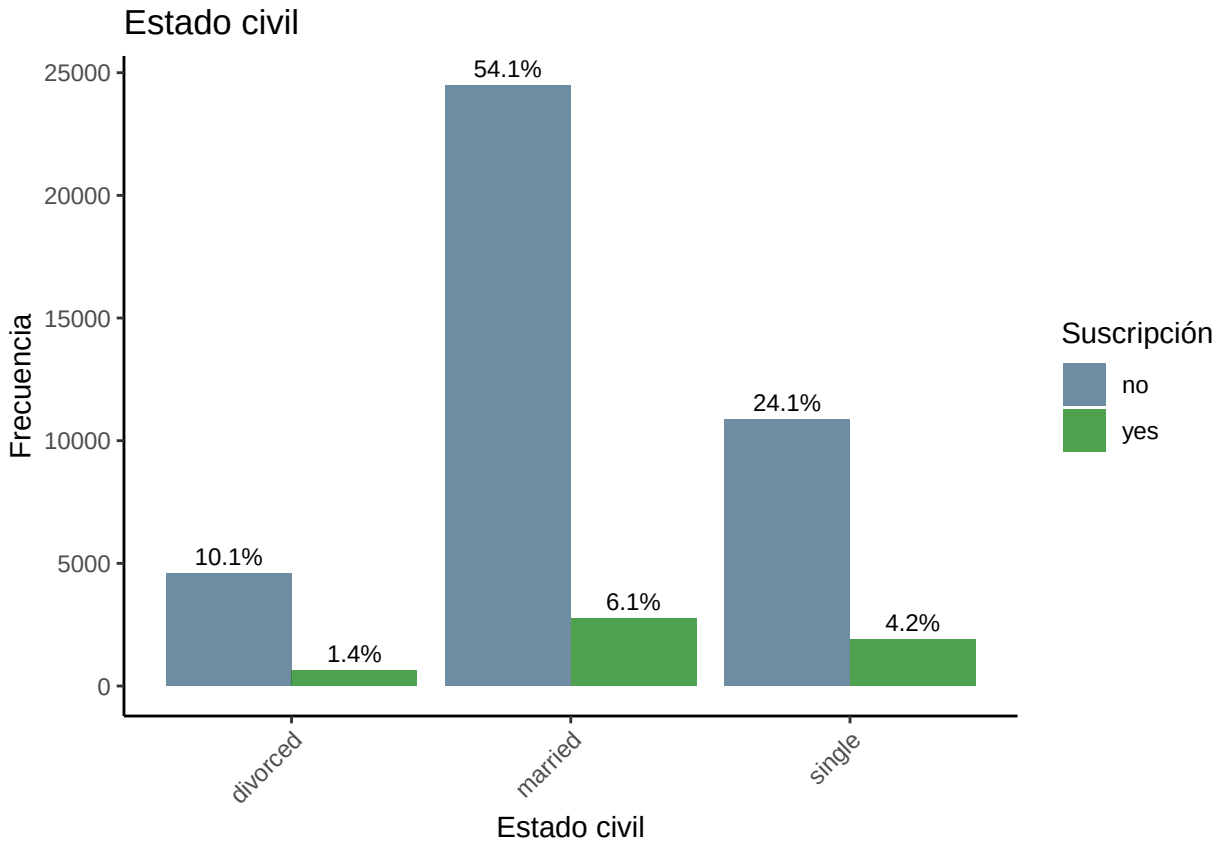
```
summary(bank$marital)
```

```
## divorced  married   single
##     5207    27214    12790
```

Vemos que hay 5,207 personas divorciadas, 27,214 casadas y 12,790 solteras. La categoría que predomina en la variable `marital` es la de casados.

```
bank %>%
  ggplot(aes(x = marital, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
```

```
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Estado civil", y = "Frecuencia", title = "Estado civil") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



Como se podía prever, en números globales los casados son los que más se suscriben, pero los solteros que se suscriben son los que tienen mayor ratio respecto al total de los solteros.

2.1.3 Variable `education`

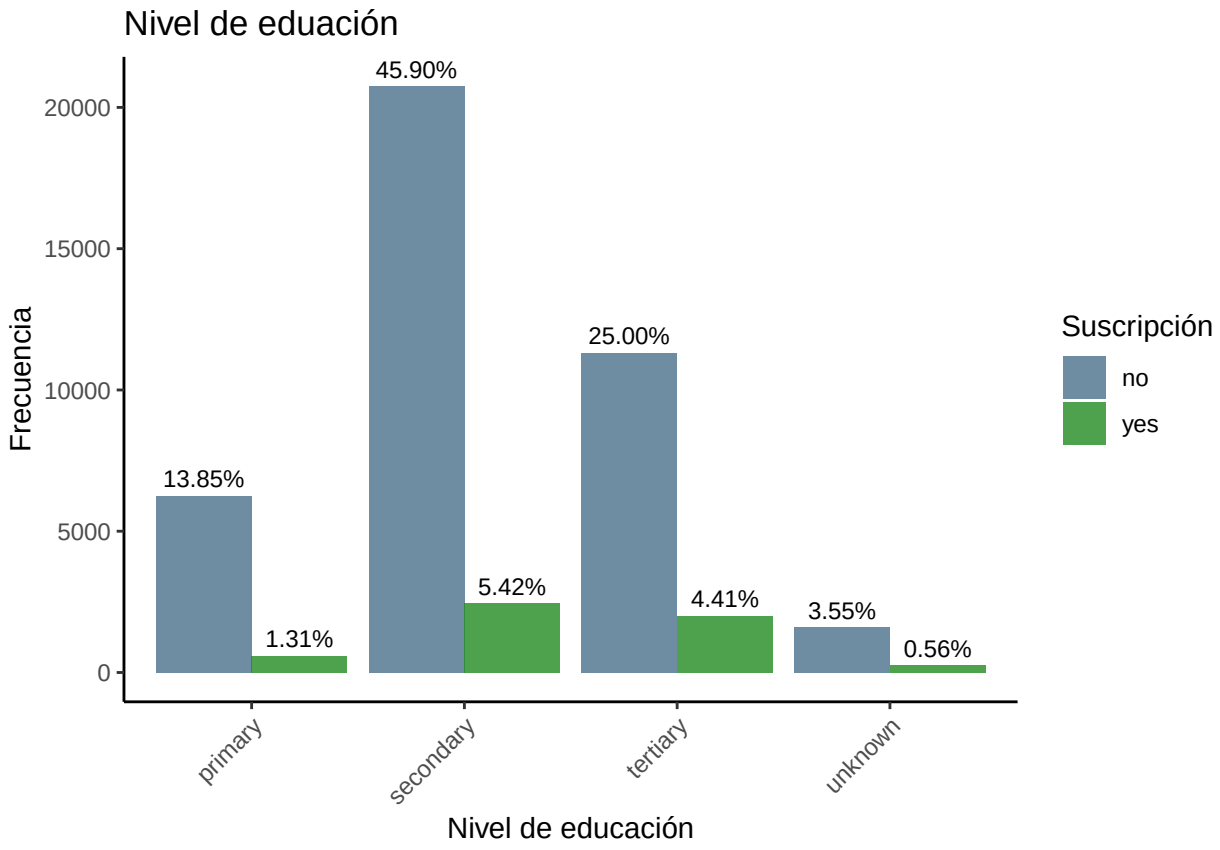
```
summary(bank$education)
```

```
##   primary secondary  tertiary   unknown
##      6851     23202     13301      1857
```

Hay 6,851 que tienen la educación primaria, 23,202 que tienen la educación secundaria, 13,301 que tienen la educación terciaria y 1,857 que desconocemos su nivel educativo. Estos 1,857 los quitaremos también, como pasó con la variable `job`.

```
bank %>%
  ggplot(aes(x = education, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
```

```
            size = 3) +
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Nivel de educación", y = "Frecuencia", title = "Nivel de eduación") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



Los que más se suscriben en términos absolutos son los que tienen educación secundaria, pero los que se suscriben teniendo la educación terciaria son los que tienen mejor ratio de suscripción respecto al total de la categoría.

Quitamos los que desconocemos su nivel de educación de nuestro data set, para más tarde realizar los modelos de redes neuronales artificiales.

```
bank_1 <- bank_1 %>%
  filter(bank_1$education != "unknown") %>%
  droplevels()
```

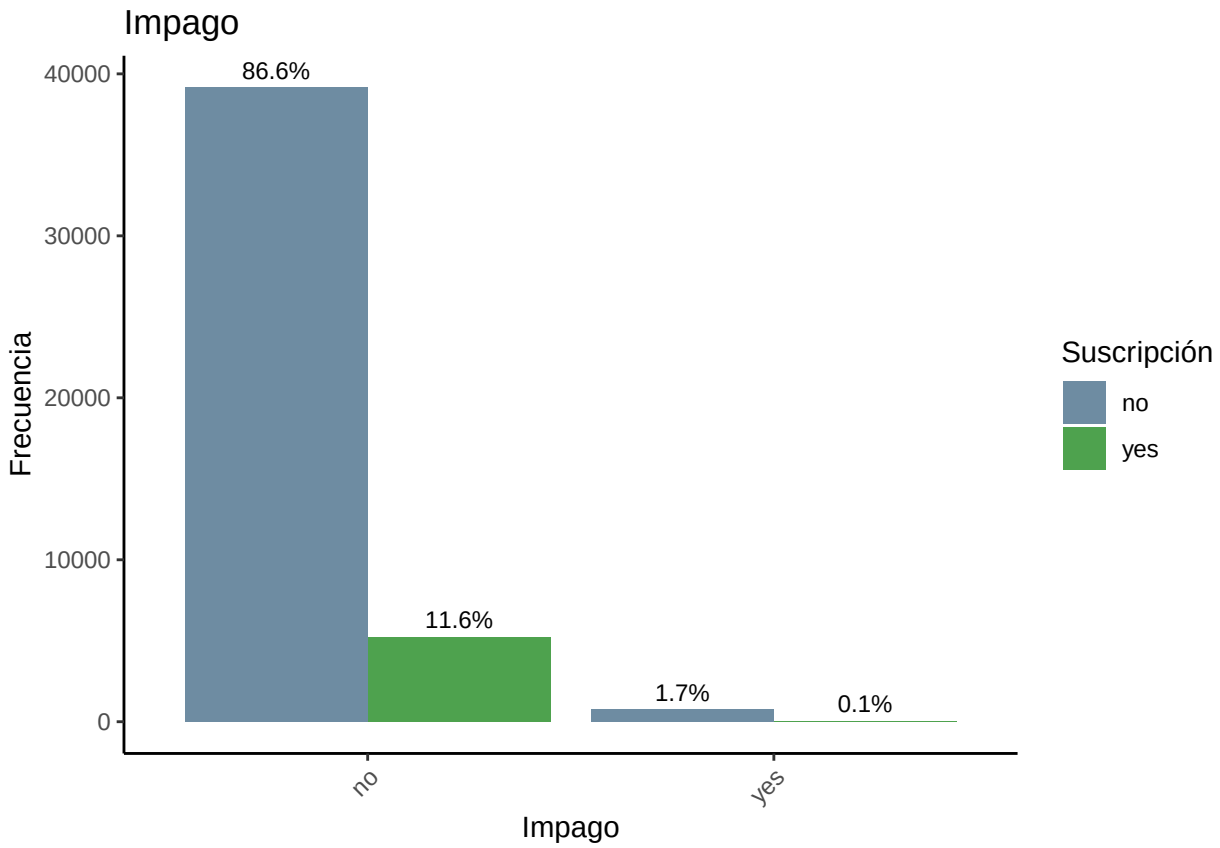
2.1.4 Variable `default`

```
summary(bank$default)
```

```
##    no   yes
## 44396   815
```

La mayoría de los entrevistados no tienen impagos, una parte ínfima tiene algún impago en su cuenta.

```
bank %>%
  ggplot(aes(x = default, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Impago", y = "Frecuencia", title = "Impago") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



Aquí se ve que los que sí tienen impago no se suscriben (un 0.1% del total se suscriben, un número despreciable) y el 11.6% del total se suscriben sin tener impagos.

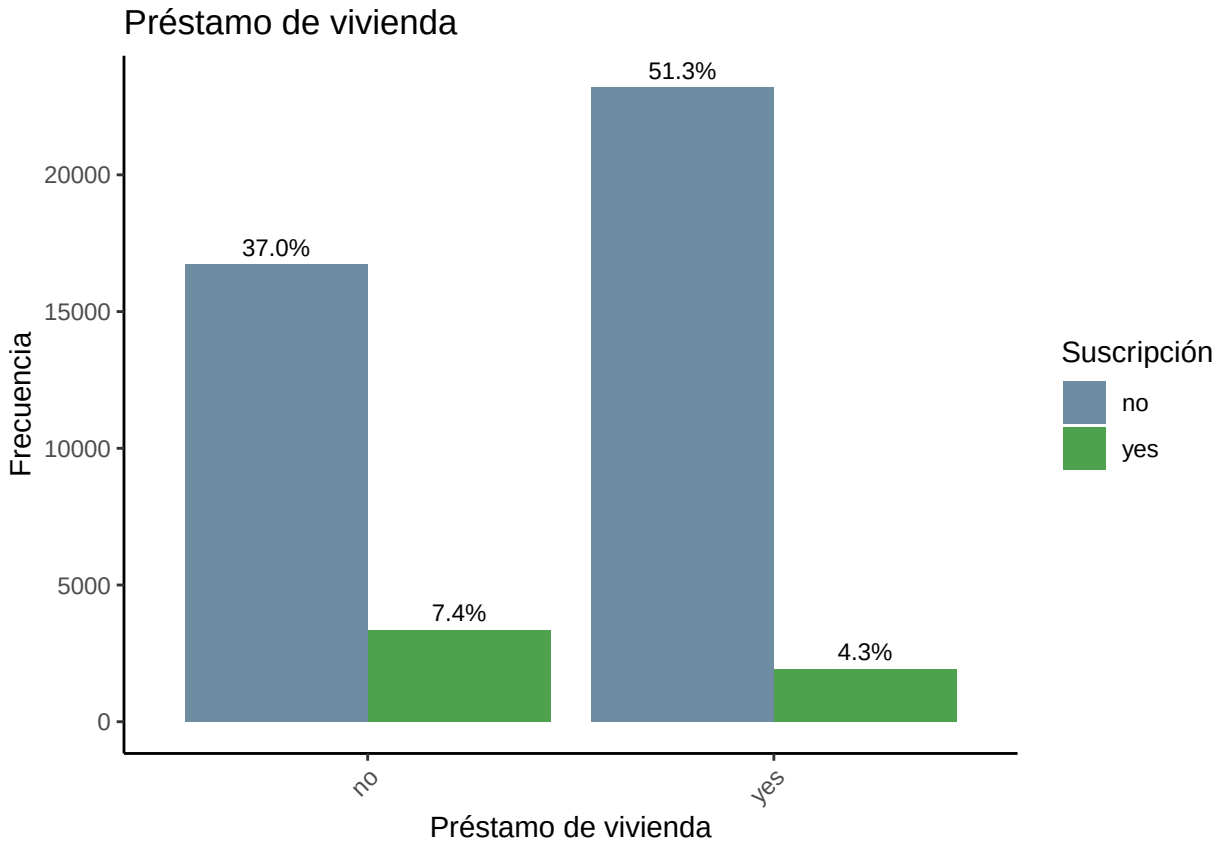
2.1.5 Variable `housing`

```
summary(bank$housing)
```

```
##    no   yes
## 20081 25130
```

Más de la mitad tienen préstamos de vivienda, algo lógico teniendo en cuenta el rango de edad, que luego veremos, en el cual se mueven los entrevistados.

```
bank %>%
  ggplot(aes(x = housing, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Préstamo de vivienda", y = "Frecuencia", title = "Préstamo de vivienda") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



Seguramente por tener una menor carga económica al carecer de préstamos de vivienda, los que más se suscriben son los que no tienen estos préstamos.

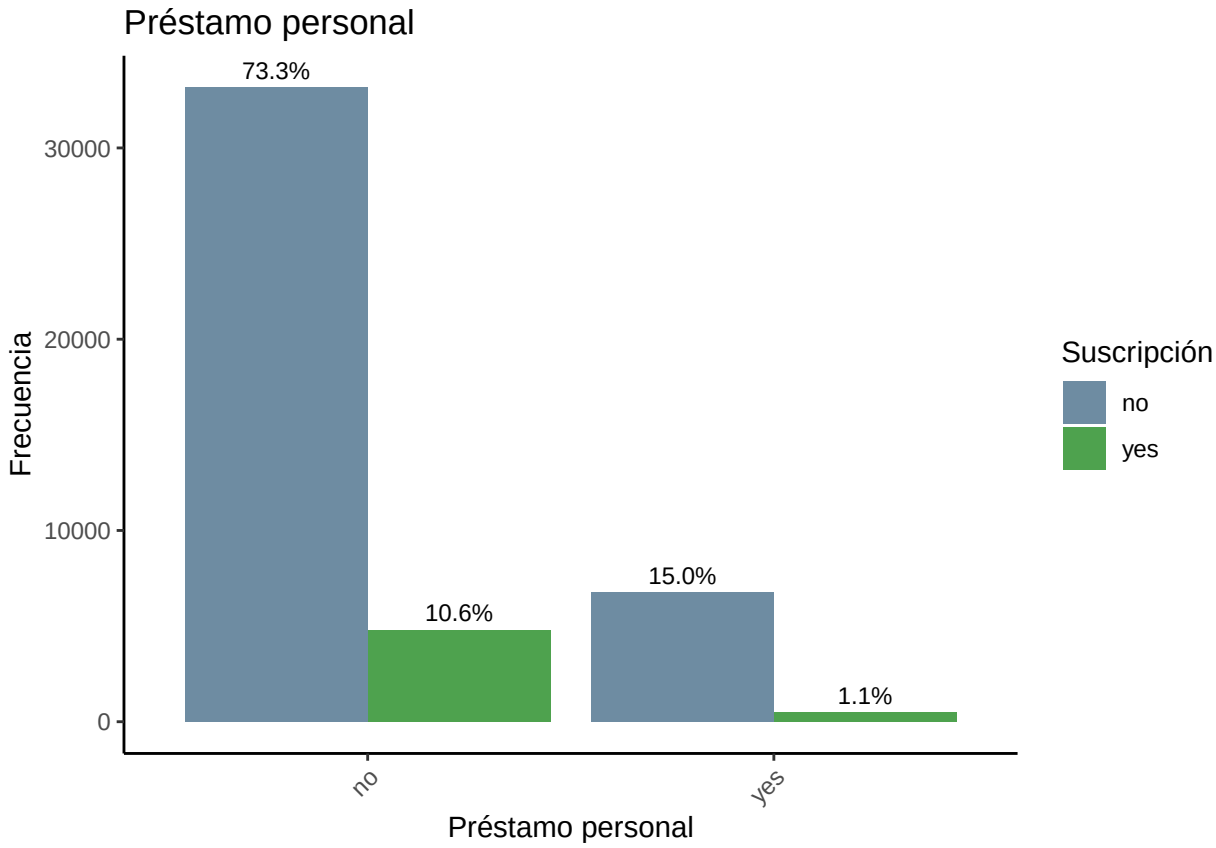
2.1.6 Variable `loan`

```
summary(bank$loan)
```

```
##    no   yes
## 37967  7244
```

Aquí cambia el asunto, la mayoría no tienen préstamos personales.

```
bank %>%
  ggplot(aes(x = loan, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Préstamo personal", y = "Frecuencia", title = "Préstamo personal") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



Como es obvio, quienes no tienen préstamos personales (deudas) tienen más flexibilidad para participar en este tipo de inversiones.

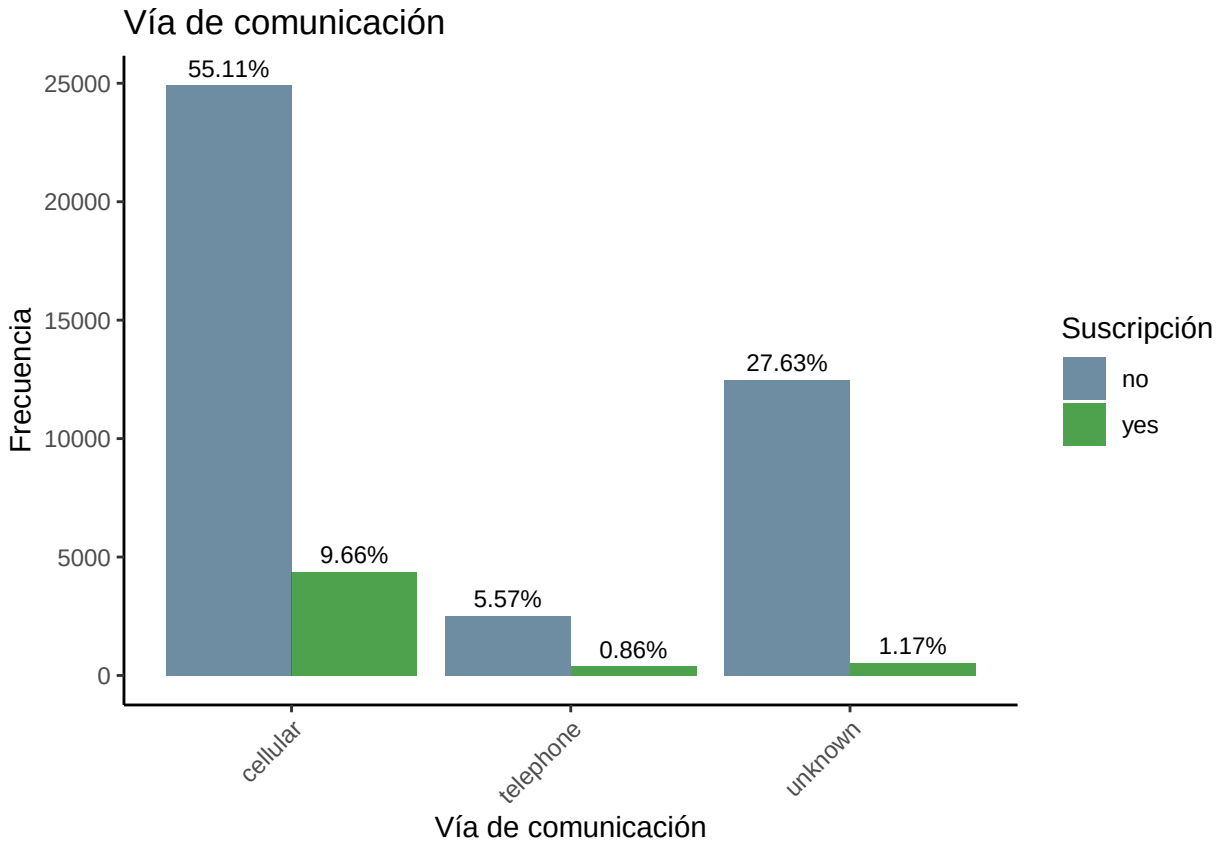
2.1.7 Variable `contact`

```
summary(bank$contact)
```

```
##  cellular telephone   unknown
##      29285      2906     13020
```

Tenemos que un 28.8% de los datos son desconocidos, con lo cual podemos creer que esta variable no nos puede aportar suficiente información para nuestros modelos.

```
bank %>%
  ggplot(aes(x = contact, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Vía de comunicación", y = "Frecuencia", title = "Vía de comunicación") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



La gran mayoría de los contactos se producen vía teléfono móvil, y también tienen el mayor porcentaje de suscripciones. Al tener tantos valores desconocidos, descartaremos esta variable.

```
bank_1 <- bank_1 %>%
  dplyr::select(-c("contact"))
```

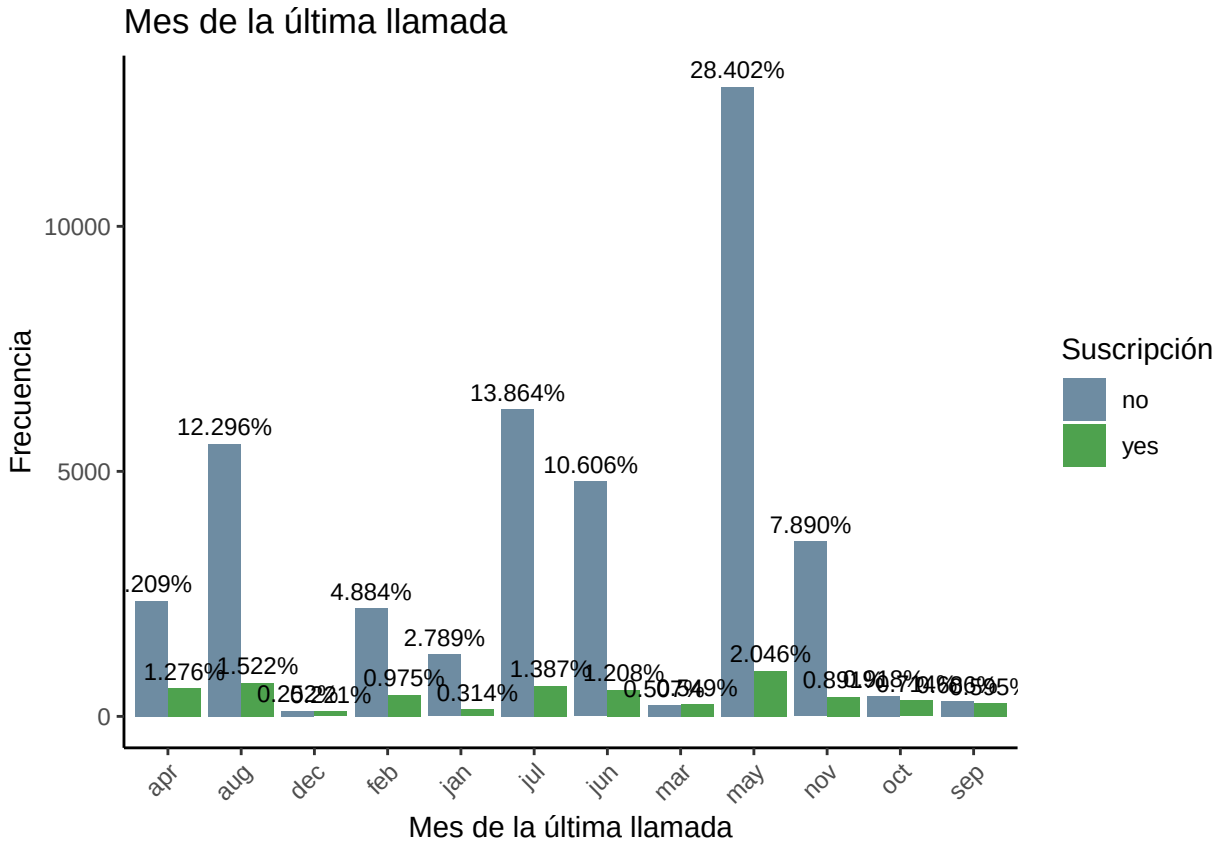
2.1.8 Variable month

```
summary(bank$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
```

Esta variable hace referencia al mes del año de la última llamada.

```
bank %>%
  ggplot(aes(x = month, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Mes de la última llamada", y = "Frecuencia", title = "Mes de la última llamada") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



Vemos que los meses donde más se han realizado las últimas llamadas son mayo, junio, julio y agosto, correspondiendo los meses de verano (exceptuando mayo).

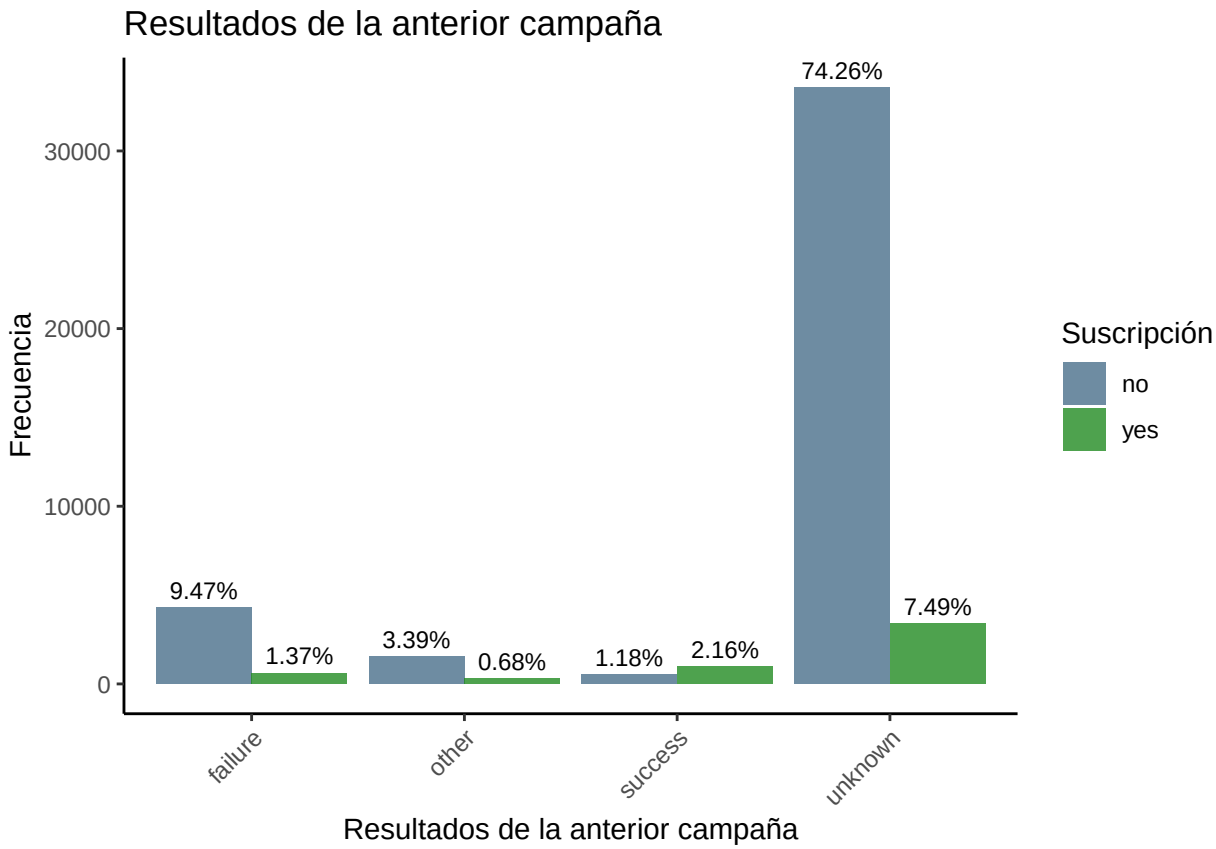
2.1.9 Variable `poutcome`

```
summary(bank$poutcome)
```

```
## failure   other success unknown
##    4901    1840    1511   36959
```

La mayoría de los datos son desconocidos.

```
bank %>%
  ggplot(aes(x = poutcome, fill = y, label =
               scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Suscripción", values = c("no" = "skyblue4",
                                              "yes" = "forestgreen")) +
  labs(x = "Resultados de la anterior campaña", y = "Frecuencia",
       title = "Resultados de la anterior campaña") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()
```



Vemos que un 81.75% de los datos son desconocidos, con lo cual descartaremos esta variable.

```
bank_1 <- bank_1 %>%
  dplyr::select(-c("poutcome"))
```

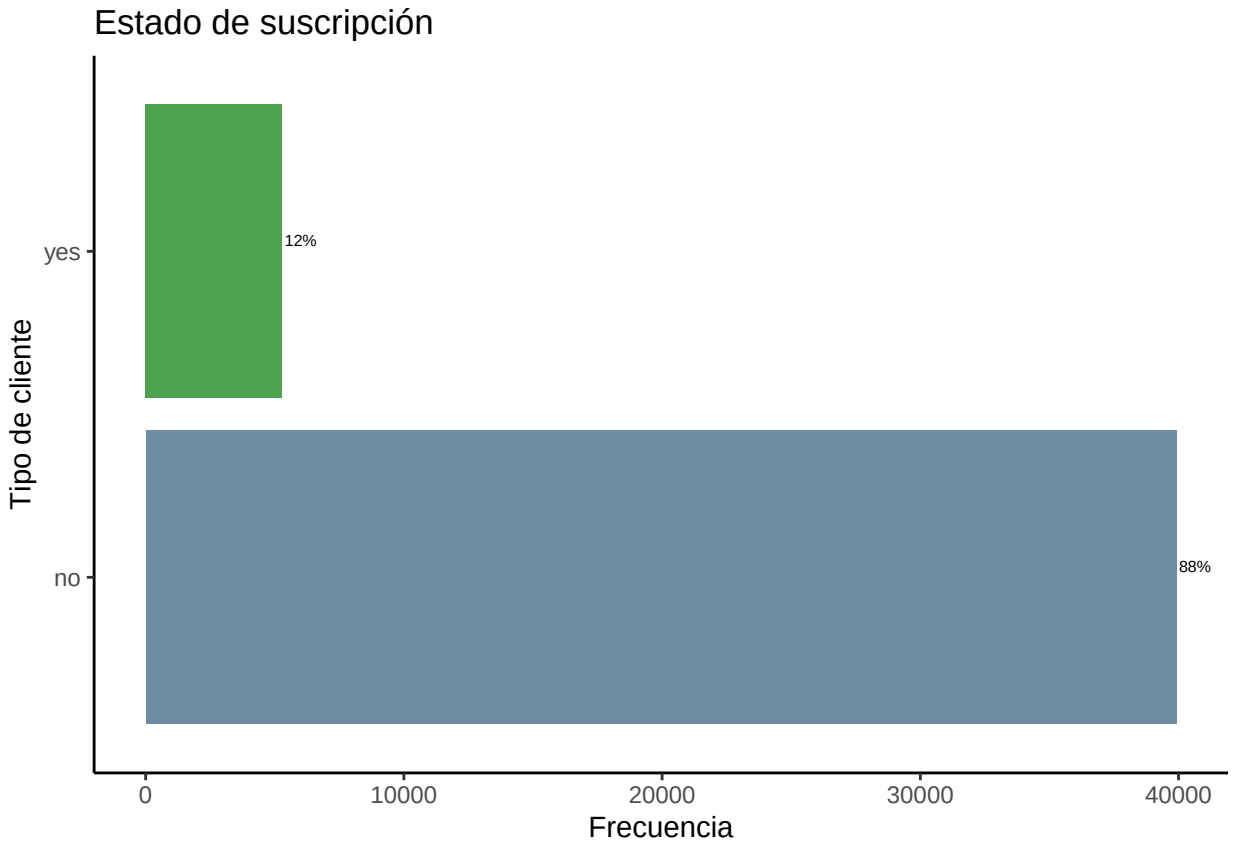
2.1.10 Variable y

```
summary(bank$y)
```

```
##    no   yes
## 39922  5289
```

Hay 39,922 personas que no están suscritas y solamente 5,289 que sí, teniendo un data set totalmente desbalanceado respecto a esta variable (es nuestra variable objetivo).

```
bank %>%
  ggplot(aes(x = y, label = scales::percent(prop.table(stat(count))))) +
  geom_bar(fill = c("skyblue4", "forestgreen"), alpha = 0.8) +
  geom_text(stat = 'count',
              position = position_dodge(.9),
              vjust = -0.5,
              hjust = -0.08,
              size = 2) +
  labs(x = "Tipo de cliente", y = "Frecuencia", title = "Estado de suscripción") +
  coord_flip() +
  theme_classic()
```



Al estar tan desbalanceado, antes de realizar los modelos de redes neuronales artificiales aplicaremos alguna técnica para ajustar conjuntos de datos desbalanceados.

2.2 Variables cuantitativas

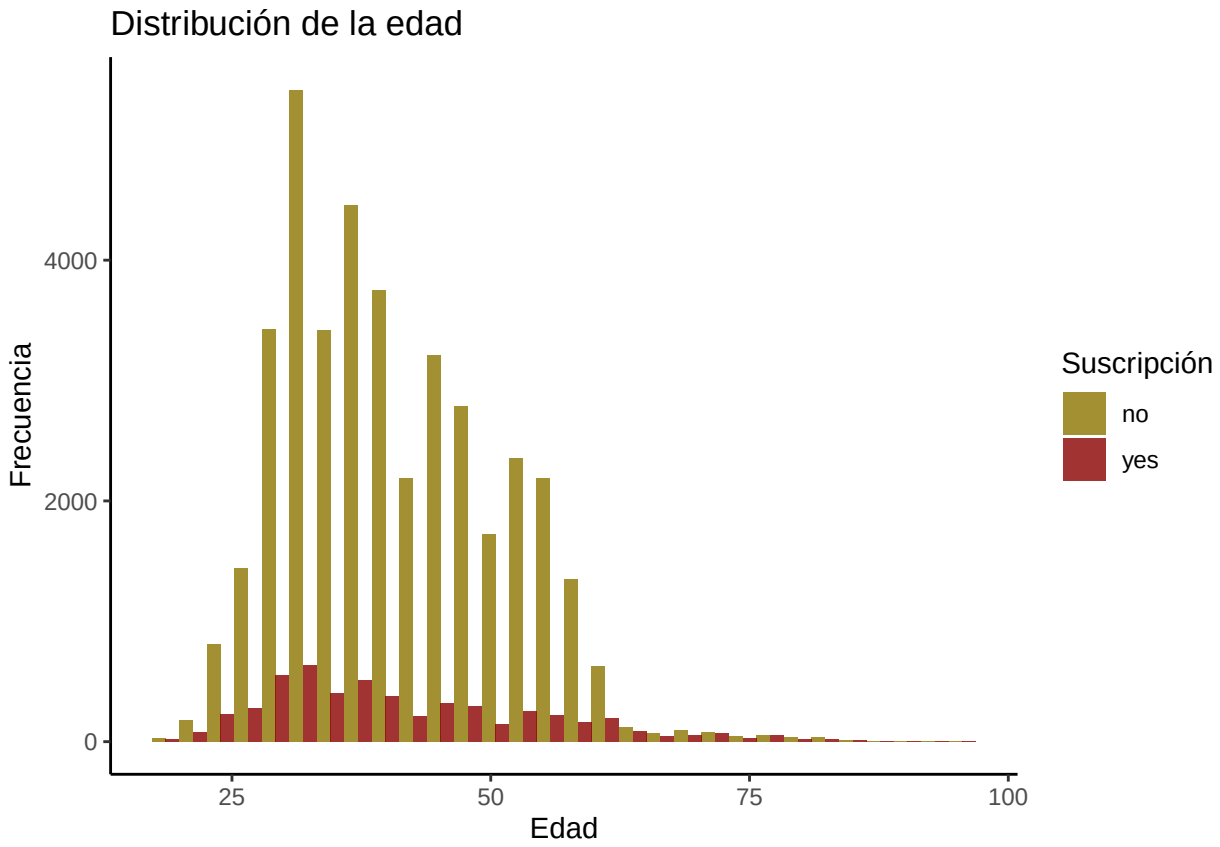
2.2.1 Variable age

```
summary(bank$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   33.00   39.00   40.94   48.00   95.00
```

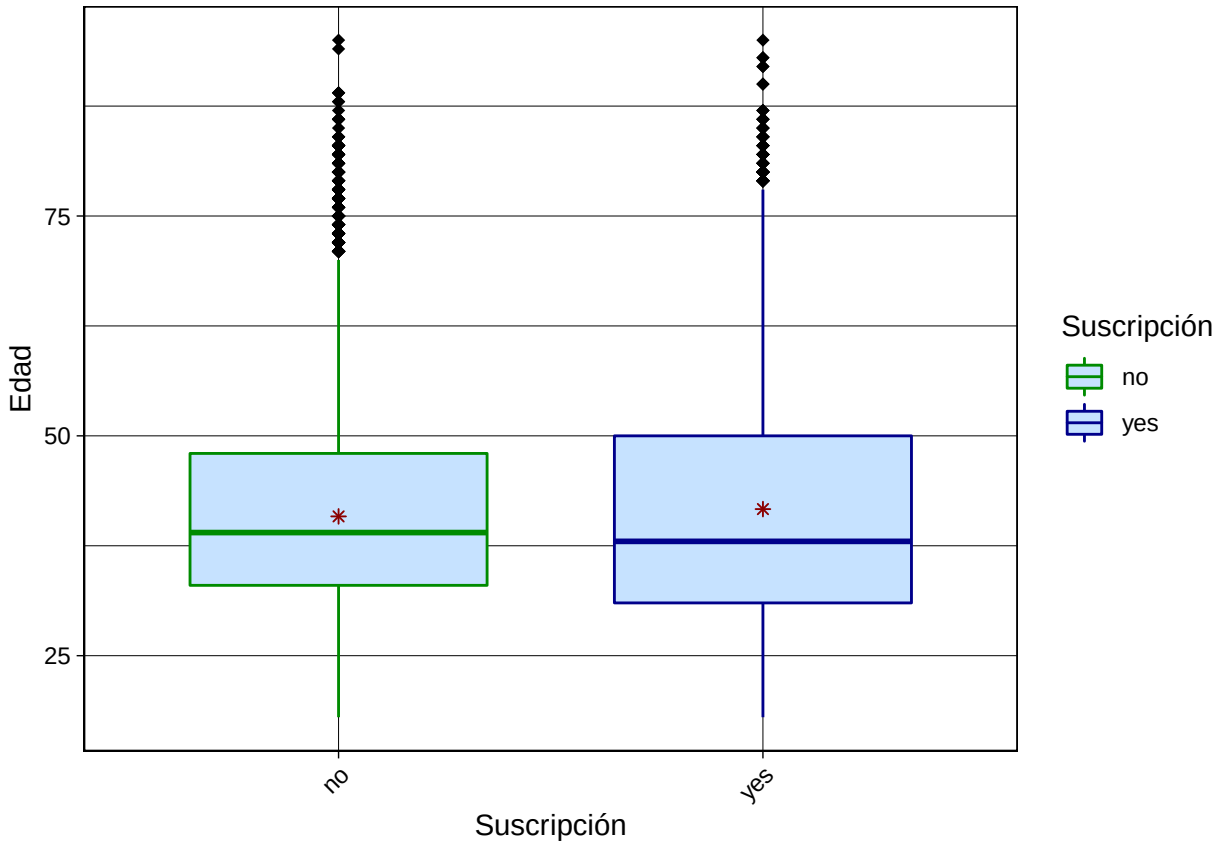
La media de edad de los encuestados es de 41 años, los encuestados tienen un perfil de persona estable. La persona más mayor que se ha entrevistado tiene 95 años, mientras que la persona más joven tiene 18 años.

```
bank %>%
  ggplot(aes(x = age, fill = y)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Suscripción", values = c("no" = "gold4",
                                              "yes" = "red4")) +
  labs(x = "Edad", y = "Frecuencia",
       title = "Distribución de la edad") +
  theme_classic()
```



Este histograma nos muestra la distribución de los encuestados por edad, filtrado por la suscripción.

```
bank %>%
  ggplot(aes(x = y, y = age, color = y)) +
  geom_boxplot(width = 0.7, fill = "slategray1", outlier.colour = "black",
               outlier.size = 2, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Suscripción", values =
                       c("green4", "blue4")) +
  theme_linedraw() +
  labs(x = "Suscripción", y = "Edad") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



Con este boxplot vemos que la media y la mediana de edad de los dos grupos son parecidas. Decidimos quitar los outliers que están por encima de los 75 años ya que suelen ser personas mayores y pueden comportarse diferente que el resto de los encuestados.

```
bank_1 <- bank_1 %>%
  filter(bank_1$age < 75)
```

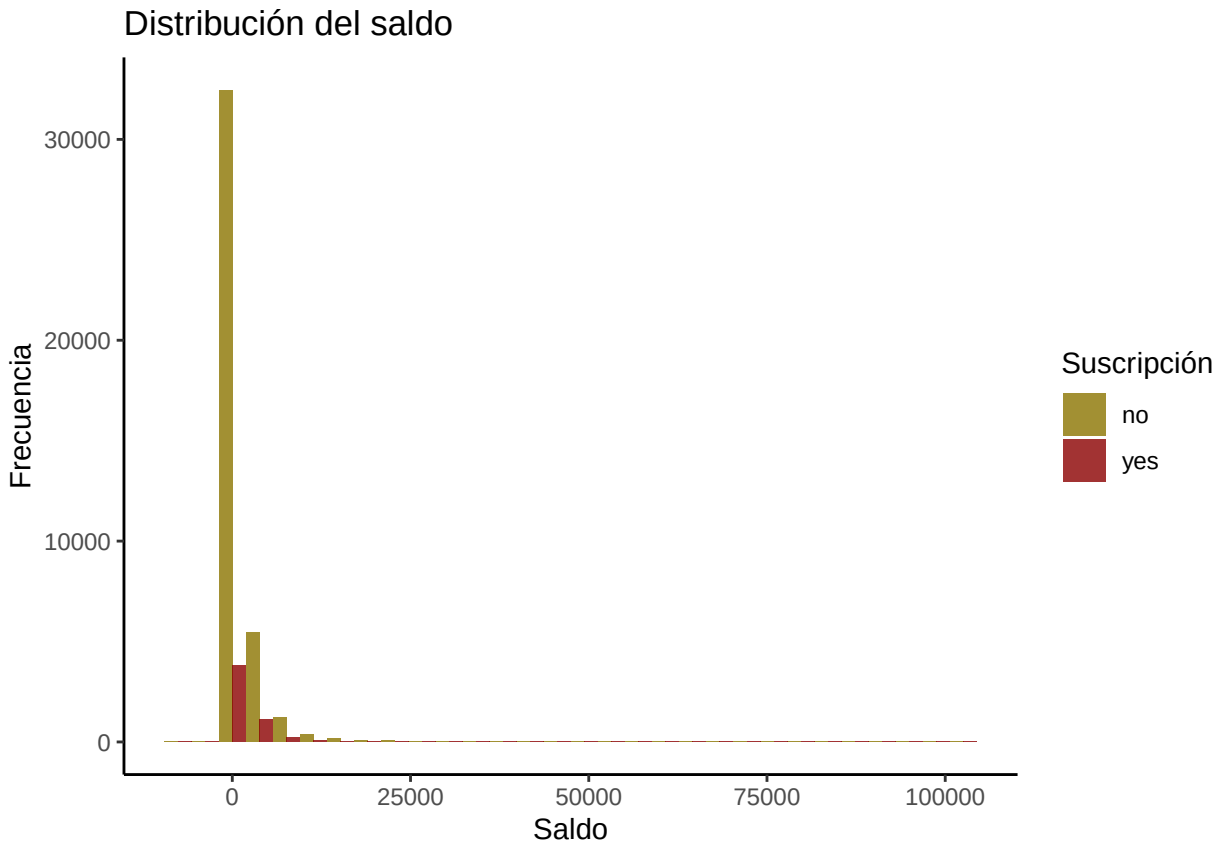
2.2.2 Variable `balance`

```
summary(bank$balance)
```

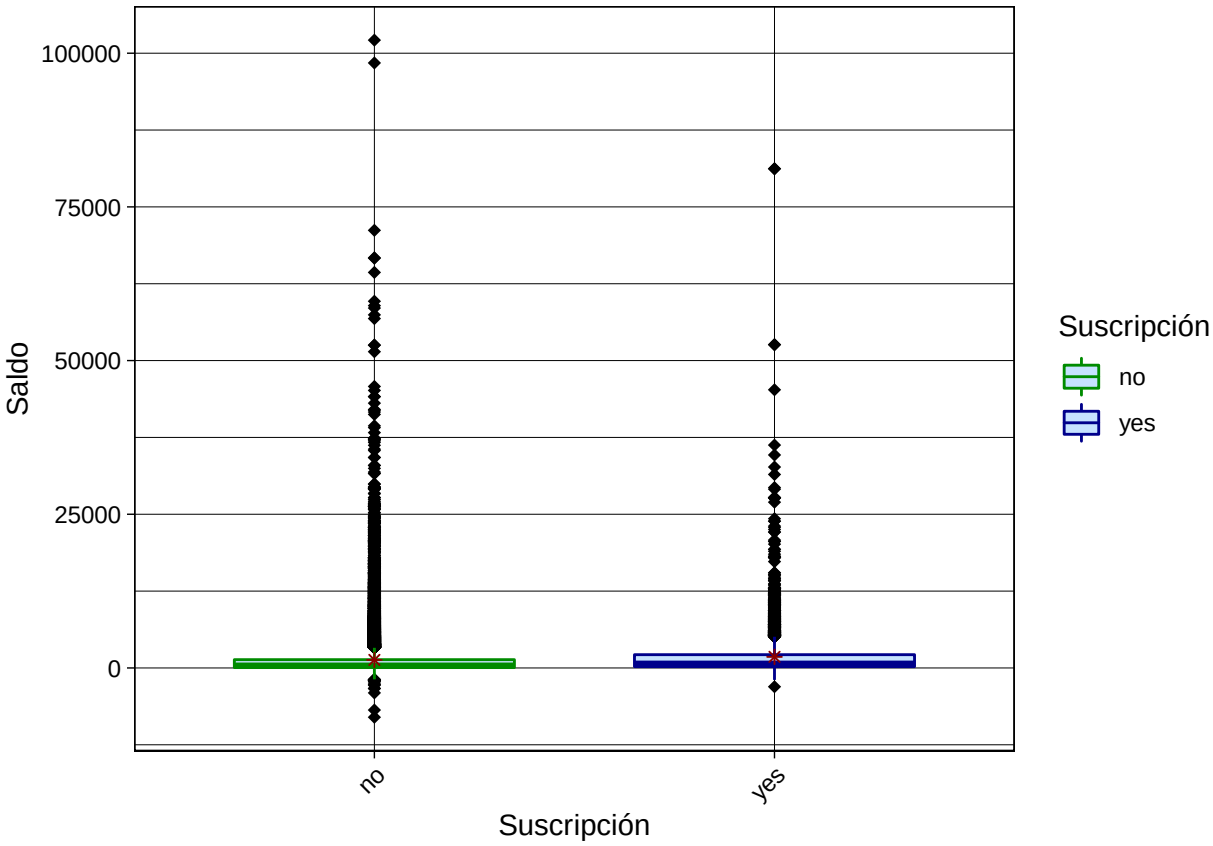
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -8019      72     448    1362    1428  102127
```

Se puede ver que la media del saldo anual medio en la cuenta es de 1362, un número no muy alto y explica el por qué del rechazo de mucha gente a suscribirse. Los casos extremos son -8,019 por la parte baja, mientras que alguien tiene 102,127 en su cuenta bancaria.

```
bank %>%
  ggplot(aes(x = balance, fill = y)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Suscripción", values = c("no" = "gold4",
                                              "yes" = "red4")) +
  labs(x = "Saldo", y = "Frecuencia",
       title = "Distribución del saldo") +
  theme_classic()
```




```
bank %>%
  ggplot(aes(x = y, y = balance, color = y)) +
  geom_boxplot(width = 0.7, fill = "slategray1", outlier.colour = "black",
               outlier.size = 2, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Suscripción", values =
                       c("green4", "blue4")) +
  theme_linedraw() +
  labs(x = "Suscripción", y = "Saldo") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



Aquí tenemos un gran número de outliers para ambos casos. Decidimos quitar los que tienen saldo mayor o igual que 25,000 porque no son representativos.

```
bank_1 <- bank_1 %>%
  filter(bank_1$balance < 25000)
```

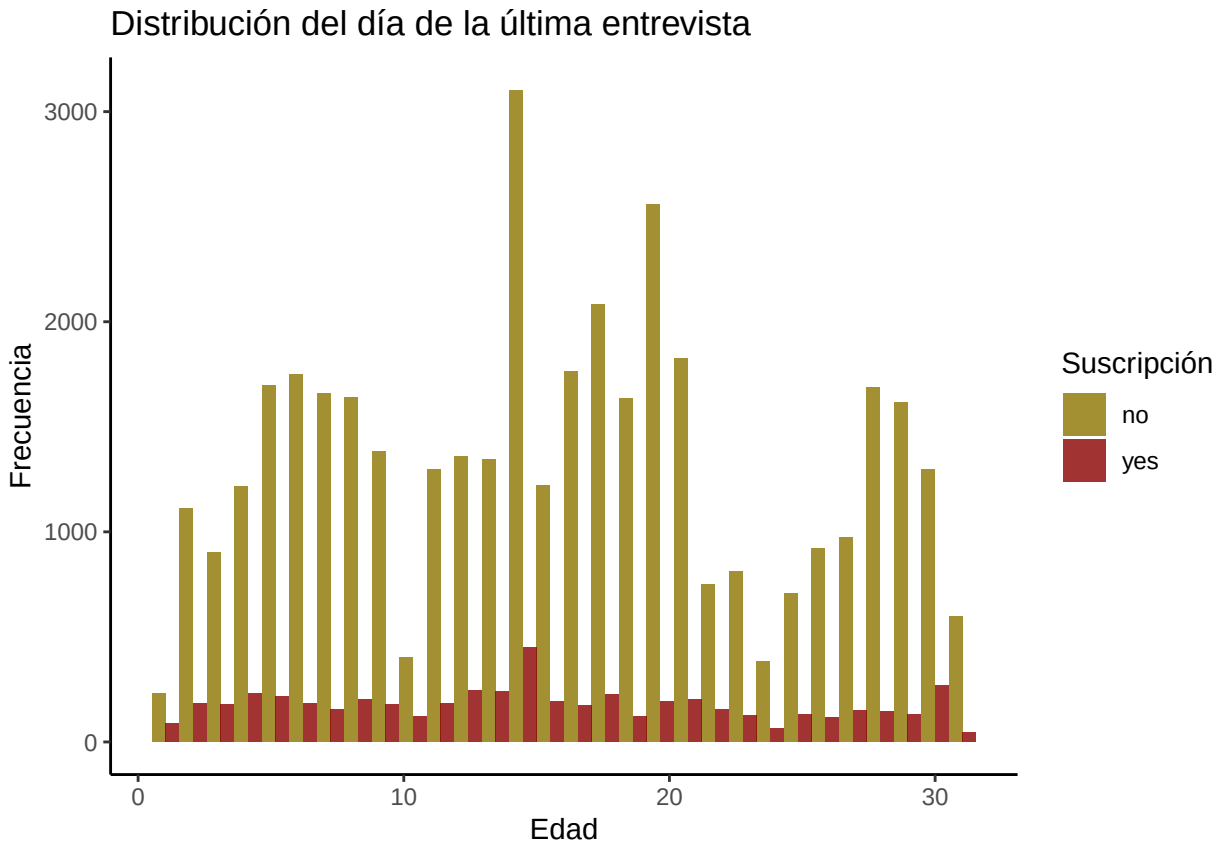
2.2.3 Variable day

```
summary(bank$day)
```

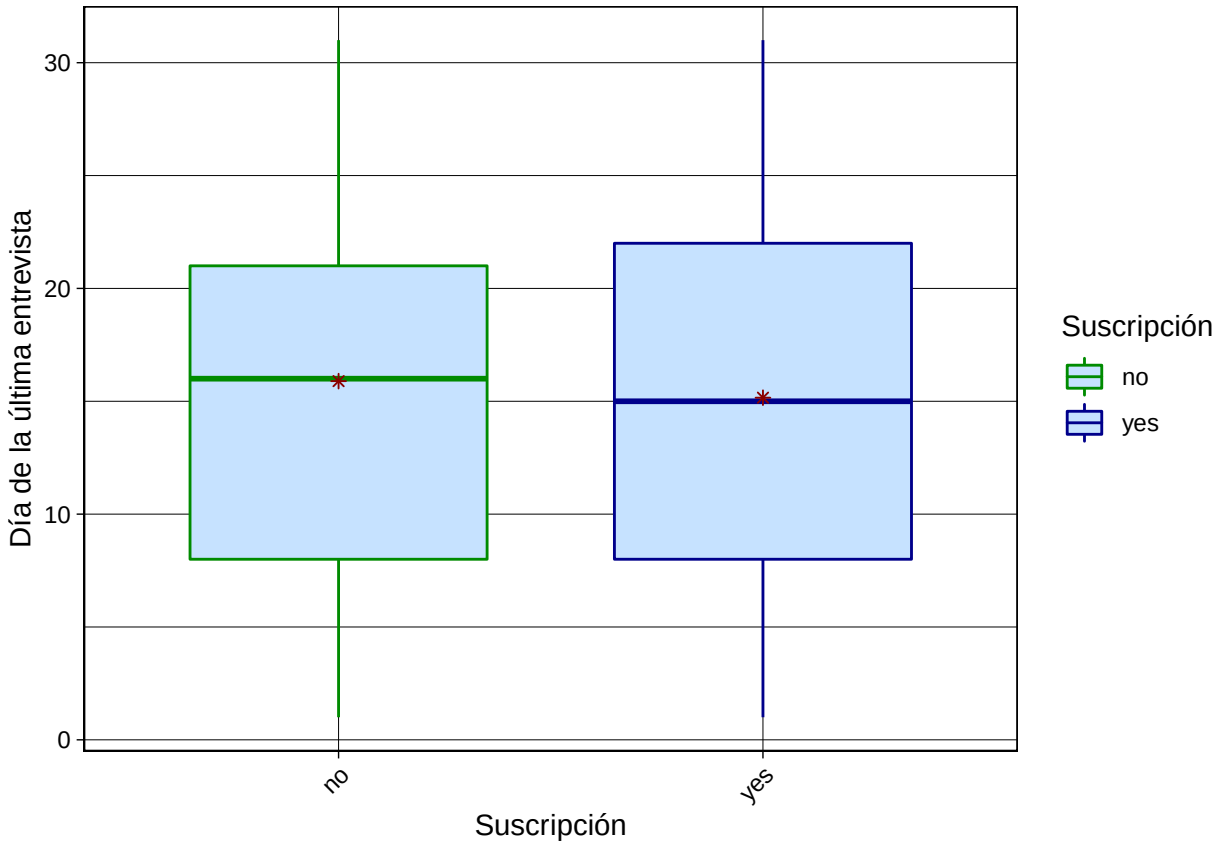
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    8.00   16.00   15.81   21.00   31.00
```

La mayoría de las llamadas se realizan sobre el ecuador del mes.

```
bank %>%
  ggplot(aes(x = day, fill = y)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Suscripción", values = c("no" = "gold4",
                                              "yes" = "red4")) +
  labs(x = "Edad", y = "Frecuencia",
       title = "Distribución del día de la última entrevista") +
  theme_classic()
```




```
bank %>%
  ggplot(aes(x = y, y = day, color = y)) +
  geom_boxplot(width = 0.7, fill = "slategray1", outlier.colour = "black",
               outlier.size = 2, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Suscripción", values =
                       c("green4", "blue4")) +
  theme_linedraw() +
  labs(x = "Suscripción", y = "Día de la última entrevista") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



La media y la mediana de las dos clases se parecen muchísimo, sin ninguna presencia de outliers en ambos casos.

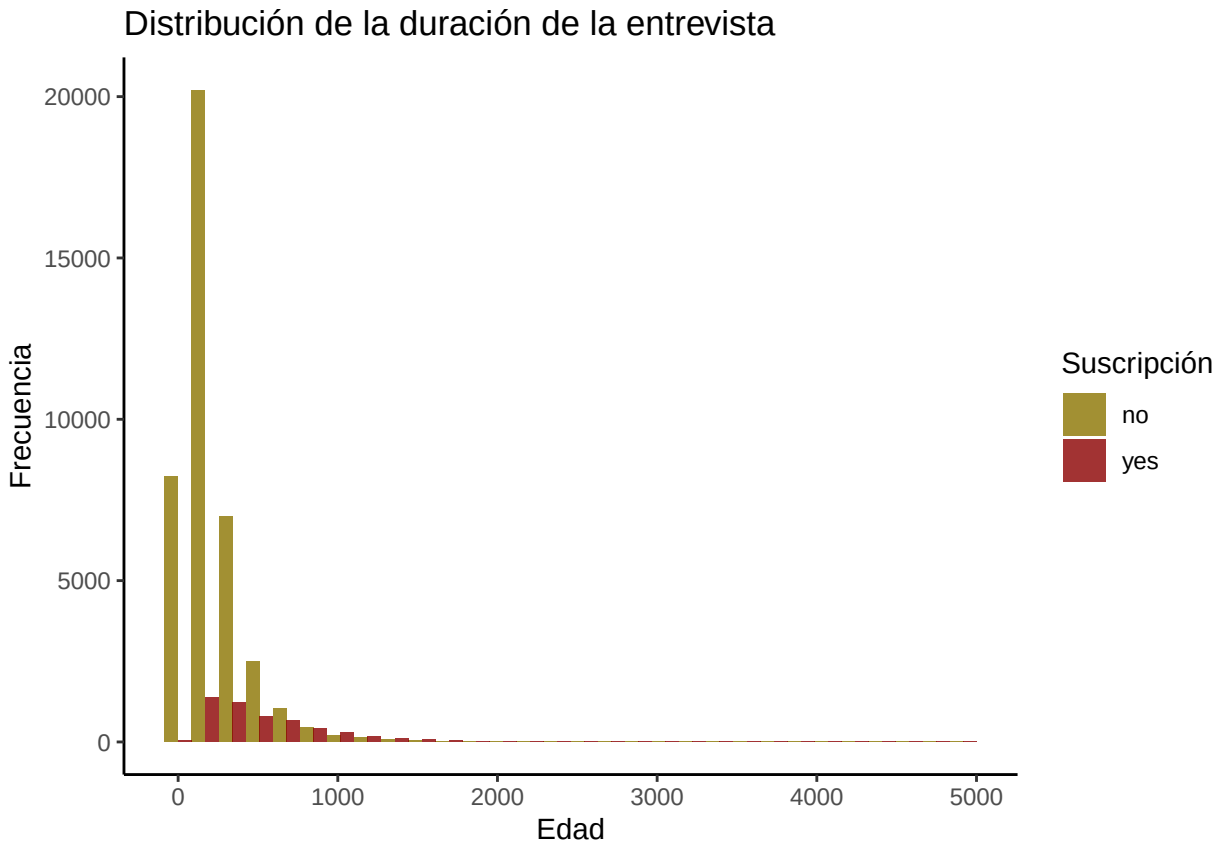
2.2.4 Variable `duration`

```
summary(bank$duration)
```

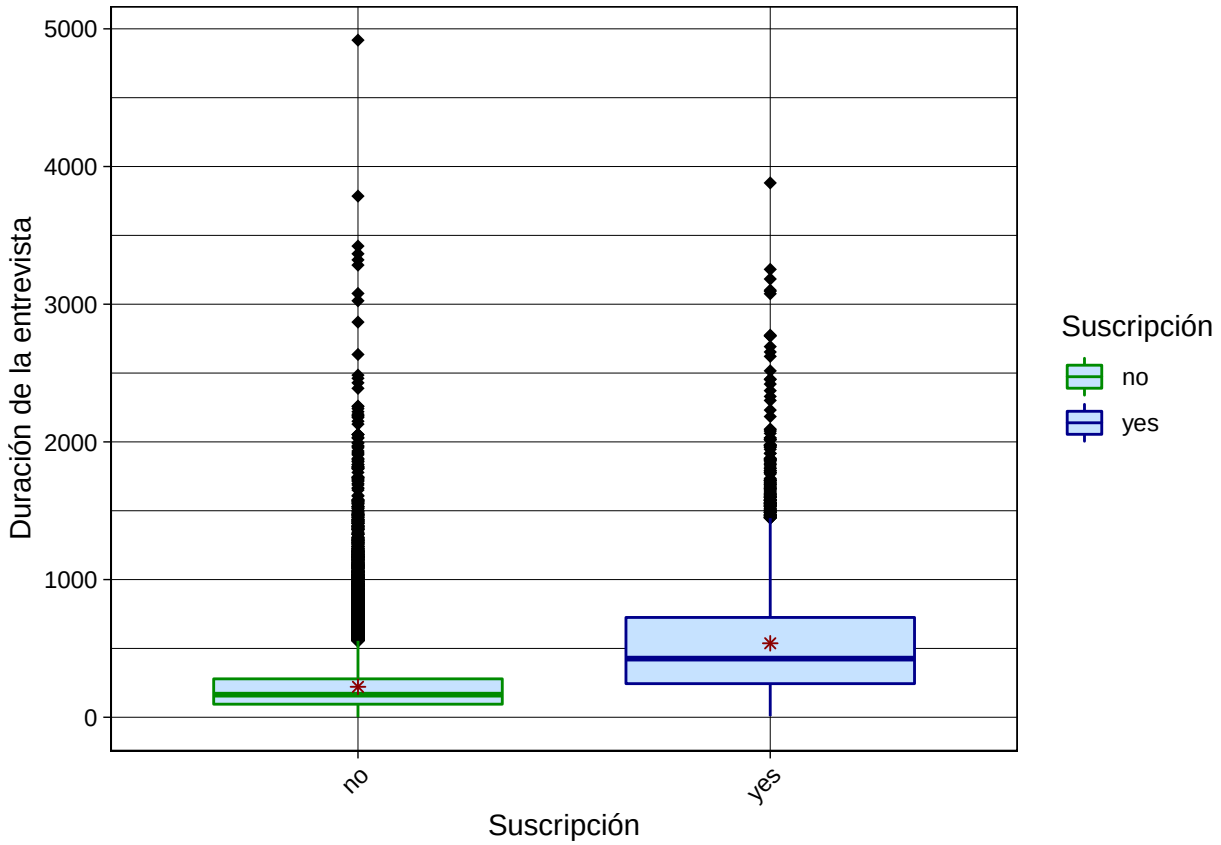
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   103.0   180.0   258.2   319.0  4918.0
```

La duración de la llamada puede ser determinante en la decisión del cliente. La media en segundos de las llamadas es de 258.2 segundos, mientras que la llamada más longeva ha durado 4918 segundos y la más corta 0 segundos (sin contestar).

```
bank %>%
  ggplot(aes(x = duration, fill = y)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Suscripción", values = c("no" = "gold4",
                                              "yes" = "red4")) +
  labs(x = "Edad", y = "Frecuencia",
       title = "Distribución de la duración de la entrevista") +
  theme_classic()
```




```
bank %>%
  ggplot(aes(x = y, y = duration, color = y)) +
  geom_boxplot(width = 0.7, fill = "slategray1", outlier.colour = "black",
               outlier.size = 2, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Suscripción", values =
                       c("green4", "blue4")) +
  theme_linedraw() +
  labs(x = "Suscripción", y = "Duración de la entrevista") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



Vemos que la media y la mediana de los segundos de la llamda de los que sí se suscriben son más altas que los que no se suscriben. Hay también muchos outliers, pero quitaremos el único que está por encima de los 4,000 segundos de llamada.

```
bank_1 <- bank_1 %>%
  filter(bank_1$duration < 4000)
```

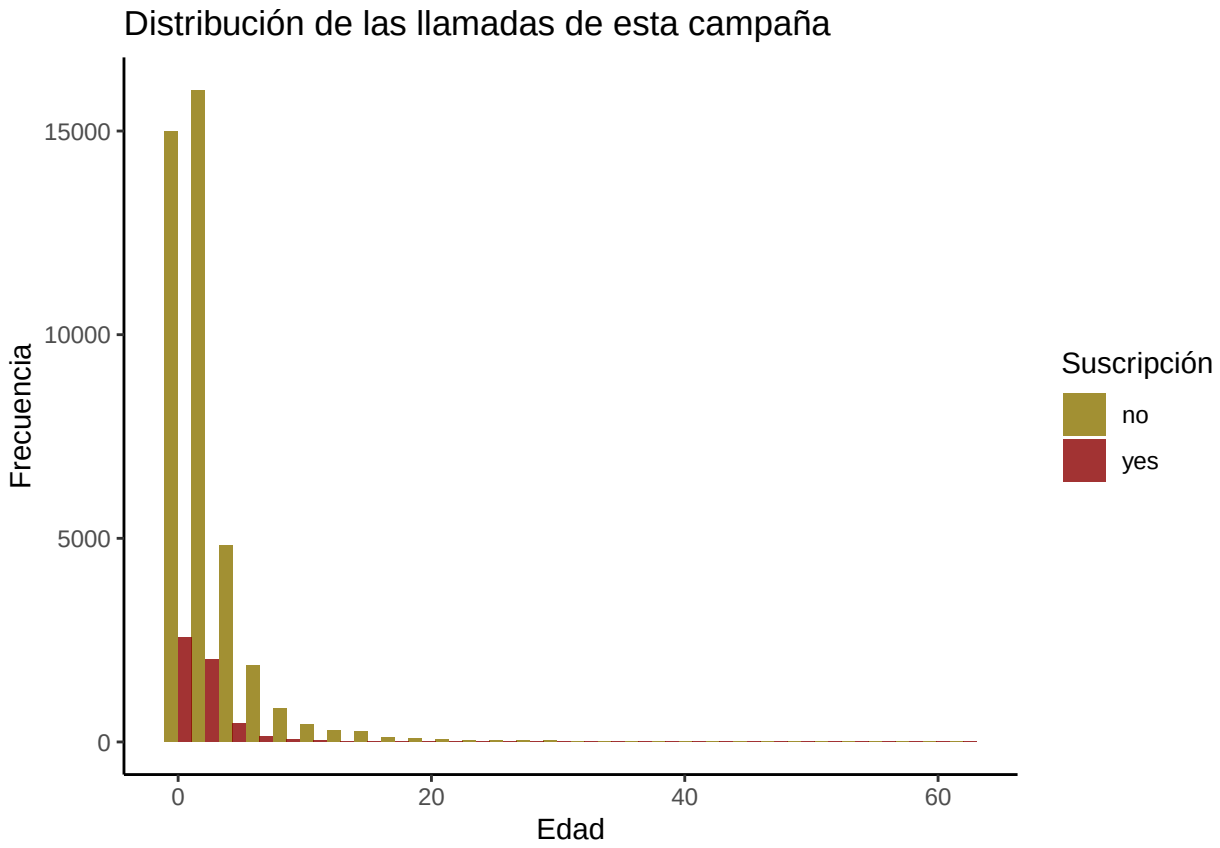
2.2.5 Variable campaign

```
summary(bank$campaign)
```

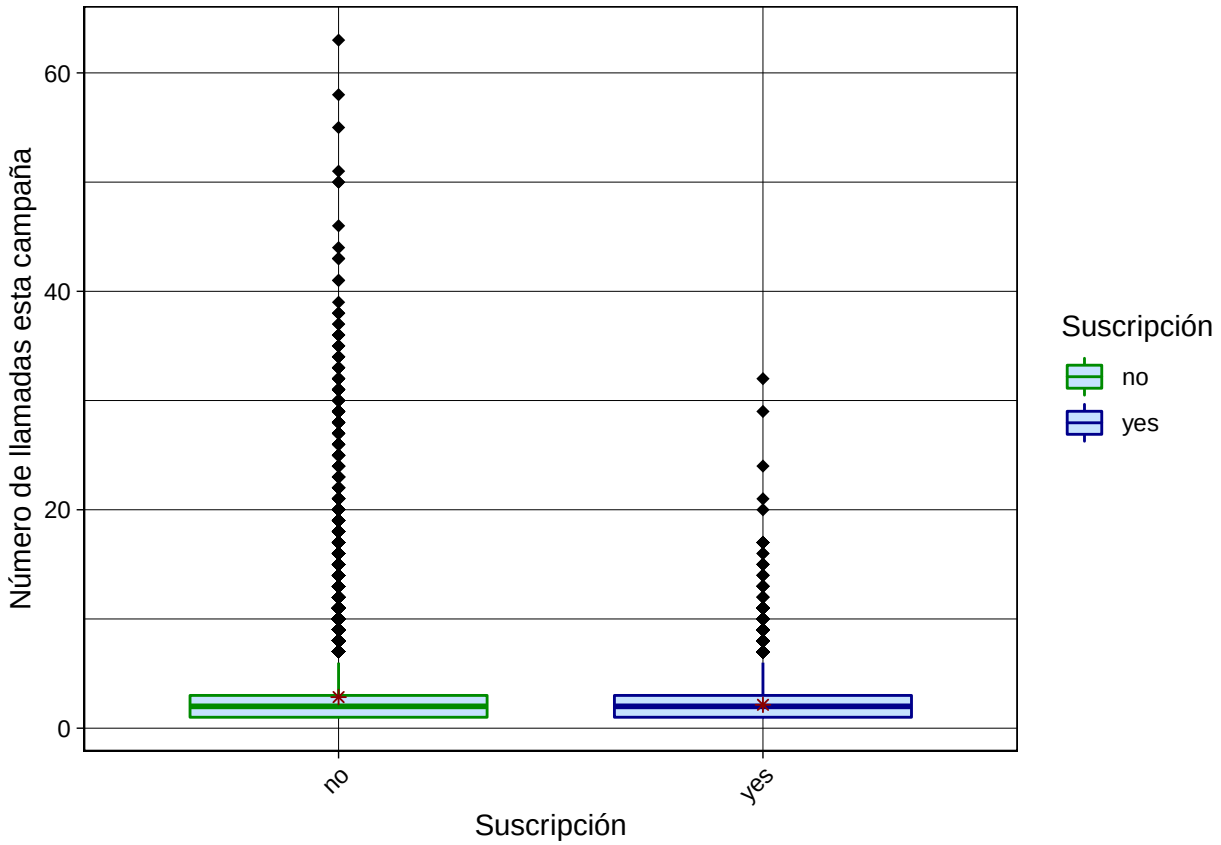
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   2.764   3.000  63.000
```

De media se ha realizado 2.764 llamadas a los clientes en esta campaña. El máximo es 63 y el mínimo es 1.

```
bank %>%
  ggplot(aes(x = campaign, fill = y)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Suscripción", values = c("no" = "gold4",
                                              "yes" = "red4")) +
  labs(x = "Edad", y = "Frecuencia",
       title = "Distribución de las llamadas de esta campaña") +
  theme_classic()
```




```
bank %>%
  ggplot(aes(x = y, y = campaign, color = y)) +
  geom_boxplot(width = 0.7, fill = "slategray1", outlier.colour = "black",
               outlier.size = 2, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Suscripción", values =
                       c("green4", "blue4")) +
  theme_linedraw() +
  labs(x = "Suscripción", y = "Número de llamadas esta campaña") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



Aquí hay un montón de outliers. Prácticamente todas las personas que reciben más 5 llamadas puede considerarse outlier ya que no es usual que se llame tantas veces para ofrecer un producto. Quitaremos todas las observaciones que se han llamado más de 5 veces.

```
bank_1 <- bank_1 %>%
  filter(bank_1$campaign <= 5)
```

2.2.6 Variable pday

```
bank_pday <- bank %>%
  filter(pdays != "-1")
summary(bank_pday$pdays)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   133.0   194.0   224.6   327.0   871.0
```

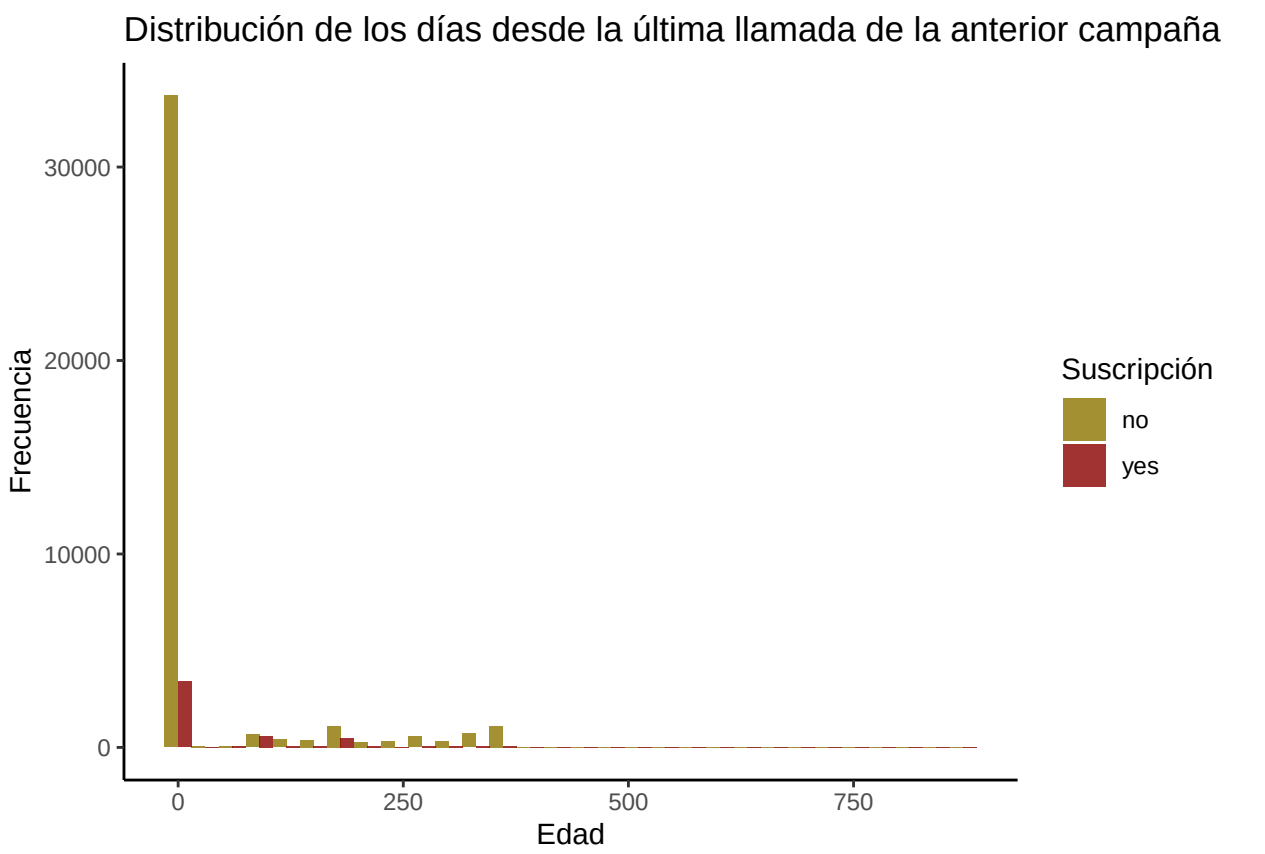
```
table(bank$pdays == "-1")
```

```
##
## FALSE  TRUE
##  8257 36954
```

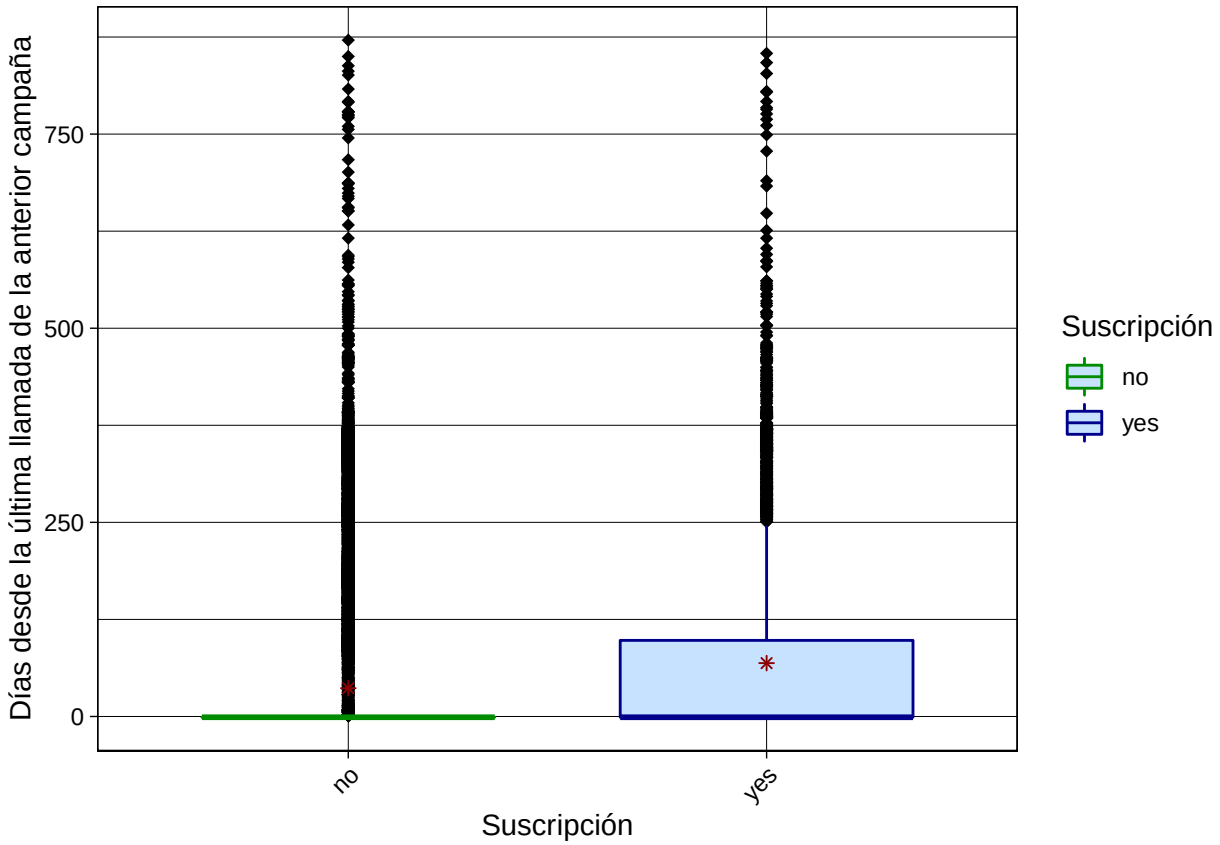
Vemos que la gran mayoría de los clientes no han sido contactados anteriormente. Lo que hemos hecho es ver, de los clientes que sí habían sido contactados anteriormente, los cuartiles y el mínimo y el máximo. La media es de 224 días, siendo el mínimo 1 dia (contactar al día siguiente de la última llamada) y el máximo 871 días.

```
bank %>%
  ggplot(aes(x = pdays, fill = y)) +
```

```
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Suscripción", values = c("no" = "gold4",
                                              "yes" = "red4")) +
  labs(x = "Edad", y = "Frecuencia",
       title = "Distribución de los días desde la última llamada de la anterior campaña") +
  theme_classic()
```




```
bank %>%
  ggplot(aes(x = y, y = pdays, color = y)) +
  geom_boxplot(width = 0.7, fill = "slategray1", outlier.colour = "black",
               outlier.size = 2, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Suscripción", values =
                       c("green4", "blue4")) +
  theme_linedraw() +
  labs(x = "Suscripción", y = "Días desde la última llamada de la anterior campaña") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



Como dijimos antes, la alta presencia de outliers son clientes que sí habían sido contactados anteriormente, mientras que la gran mayoría se sitúan en -1 ya que no han sido contactados.

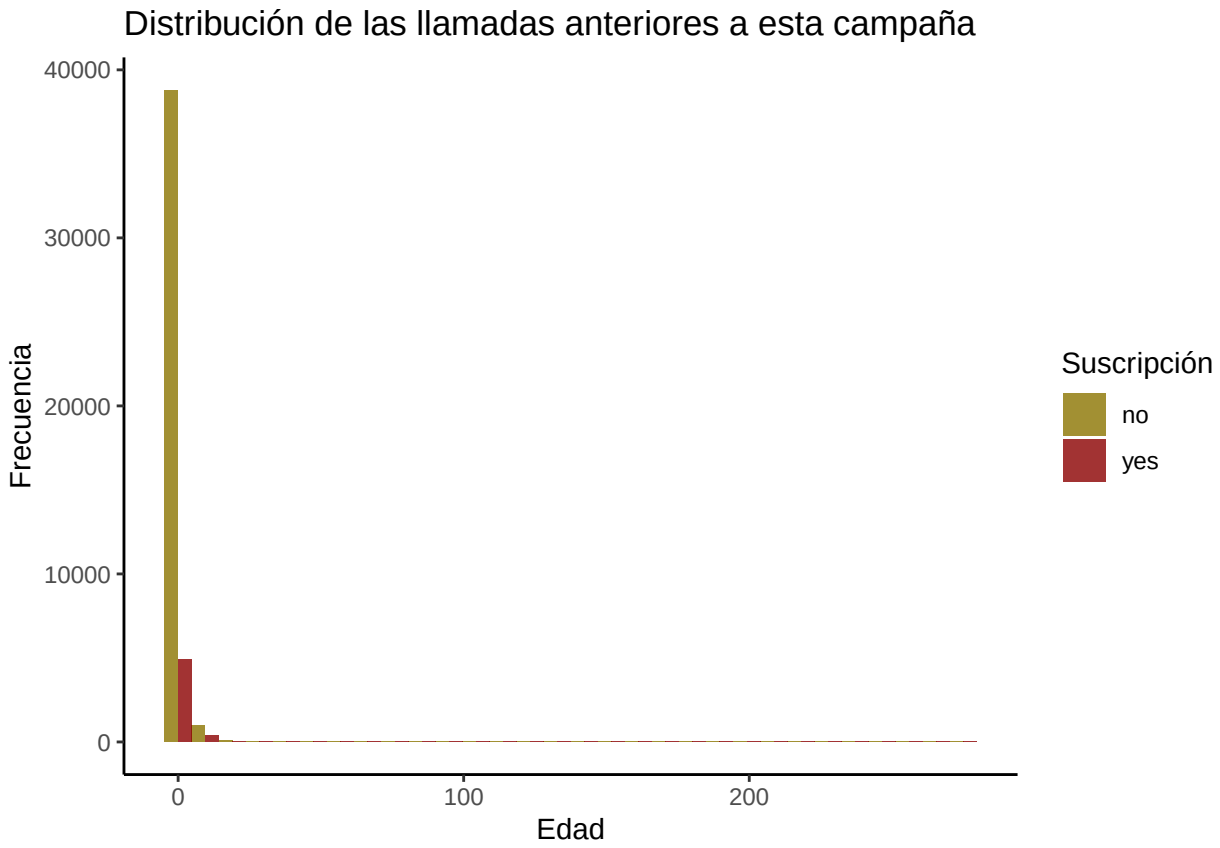
2.2.7 Variable `previous`

```
summary(bank$previous)
```

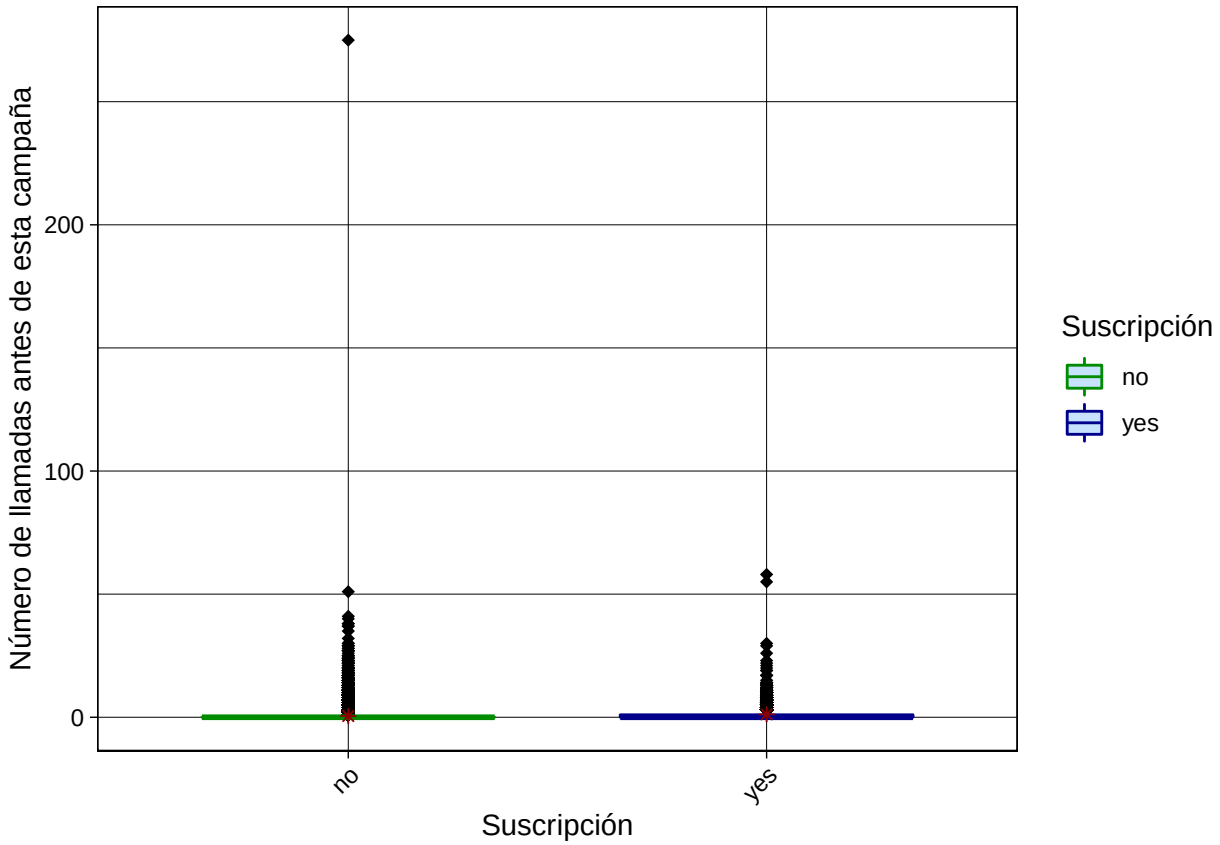
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    0.0000    0.0000    0.0000    0.5803    0.0000 275.0000
```

De media solo se ha llamado menos de una vez a los clientes en campañas anteriores. Quiere decir que la mayoría de los clientes que se llama en esta campaña no han recibido llamadas en la anterior campaña.

```
bank %>%
  ggplot(aes(x = previous, fill = y)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Suscripción", values = c("no" = "gold4",
                                              "yes" = "red4")) +
  labs(x = "Edad", y = "Frecuencia",
       title = "Distribución de las llamadas anteriores a esta campaña") +
  theme_classic()
```




```
bank %>%
  ggplot(aes(x = y, y = previous, color = y)) +
  geom_boxplot(width = 0.7, fill = "slategray1", outlier.colour = "black",
               outlier.size = 2, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Suscripción", values =
                       c("green4", "blue4")) +
  theme_linedraw() +
  labs(x = "Suscripción", y = "Número de llamadas antes de esta campaña") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



Aún sabiendo que hay tantísimos outliers, no podemos quitarlos ya que los que sí han recibido llamadas en la anterior campaña nos puede proporcionar información importante. Quitaremos el único outlier que ha recibido más de 100 llamadas en la campaña anterior.

```
bank_1 <- bank_1 %>%
  filter(bank_1$previous < 100)
```

2.3 Asimetría en las variables

Vamos a utilizar la función `skewness` del paquete `moments` para estudiar el coeficiente de asimetría de cada una de las variables que tenemos.

```
bank_skew <- as.data.frame(bank)
```

```
skewedVars<- NA

for(i in names(bank_skew)){
   if(is.numeric(bank_skew[,i])){
     if(i != "y"){
       # Enters this block if variable is non-categorical
       skewVal <- skewness(bank_skew[,i])
       print(paste(i, skewVal, sep = ": "))
       if(abs(skewVal) > 0.5){
         skewedVars <- c(skewedVars, i)
       }
     }
   }
```

```
}
```

```
## [1] "age: 0.684772484936892"
## [1] "balance: 8.35975358163196"
## [1] "day: 0.0930728378047164"
## [1] "duration: 3.1441094595968"
## [1] "campaign: 4.89832511843031"
## [1] "pdays: 2.61554190902143"
## [1] "previous: 41.8436777698195"
```

```
skewedVars
```

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

Vemos que las variable `balance`, `duration`, `campaign`, `pdays` y `previous` son muy asimétricas ya que su valor absoluto es mayor que 1. Quitaremos la variable `previous` ya que su valor es excesivamente grande y los datos son totalmente asimétricos. Podríamos pensar en quitar también `balance` pero esta variable representa el saldo medio anual de nuestro clientes y es una variable altamente importante en nuestro estudio. Observamos también que hay variables que podemos excluir para nuestro estudio. La variable que quitaremos es `pdays` ya que la mayoría de los clientes no han sido contactados anteriormente y supone unos datos totalmente desbalanceados.

```
bank_1 <- bank_1 %>%
  dplyr::select(-c("pdays", "previous"))
summary(bank_1)
```

```
##       age                     job            marital          education
##  Min.   :18.00   No_workers   : 3685   divorced: 4518   primary  : 5983
##  1st Qu.:32.00   Self_employed:15401   married :23065   secondary:20902
##  Median :39.00   Services     :19643   single  :11146   tertiary :11844
##  Mean   :40.46
##  3rd Qu.:48.00
##  Max.   :74.00
##
##  default        balance      housing      loan            day
##  no :38041   Min.   :-8019   no :16581   no :32335   Min.   : 1.00
##  yes:  688   1st Qu.:   73   yes:22148   yes: 6394   1st Qu.: 8.00
##              Median :  444                           Median :15.00
##              Mean   : 1277                           Mean   :15.45
##              3rd Qu.: 1406                           3rd Qu.:21.00
##              Max.   :24870                           Max.   :31.00
##
##      month          duration         campaign       y
##  may    :12245   Min.   :   0.0   Min.   :1.000   no :34073
##  jul    : 5608   1st Qu.: 109.0   1st Qu.:1.000   yes: 4656
##  aug    : 4801   Median : 186.0   Median :2.000
##  jun    : 4363   Mean   : 264.4   Mean   :2.013
##  nov    : 3679   3rd Qu.: 324.0   3rd Qu.:3.000
##  apr    : 2719   Max.   :3881.0   Max.   :5.000
##  (Other): 5314
```

3 SVM

3.1 SVM con kernel lineal

Primero de todos, vamos a pasar las variables categóricas a variables dummy.

```
bank_2 <- bank_1 %>%
  mutate(y = ifelse(y == "no", 0, 1)) %>%
  dummy_cols(select_columns = c("job", "marital", "education", "default", "housing", "loan", "month")) %
  dplyr::select(-c("job", "marital", "education", "default", "housing", "loan", "month"))

bank_2 <- bank_2[, c(1:5, 7:33, 6)]
glimpse(bank_2)
```

```
## Rows: 38,729
## Columns: 33
## $ age                 <dbl> 58, 44, 33, 35, 28, 42, 58, 43, 41, 29, 53, 57,...
## $ balance             <dbl> 2143, 29, 2, 231, 447, 2, 121, 593, 270, 390, 6...
## $ day                 <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ duration            <dbl> 261, 151, 76, 139, 217, 380, 50, 55, 222, 137, ...
## $ campaign            <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ job_No_workers      <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,...
## $ job_Self_employed   <int> 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,...
## $ job_Services        <int> 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1,...
## $ marital_divorced    <int> 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,...
## $ marital_married     <int> 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1,...
## $ marital_single      <int> 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,...
## $ education_primary   <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,...
## $ education_secondary <int> 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,...
## $ education_tertiary  <int> 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ default_no          <int> 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ default_yes         <int> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ housing_no          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ housing_yes         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ loan_no             <int> 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ loan_yes            <int> 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_apr           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_aug           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_dec           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_feb           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_jan           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_jul           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_jun           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_mar           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_may           <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ month_nov           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_oct           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ month_sep           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ y                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

A continuación, definiremos la función de normalización max-min para más tarde aplicarla a nuestro conjunto de datos que utilizaremos para modelar.

```
normalize <- function(x)
  {
```

```
    return((x- min(x)) /(max(x)-min(x)))
  }

bank.norm <- as.data.frame(sapply(bank_2[,-33], normalize))
bank.norm <- bank.norm %>%
  mutate(y = bank_2$y)
```

Una vez que tenemos nuestro conjunto normalizado, vamos a separarlo en conjunto de entrenamiento y conjunto de validación con un ratio de 70:30.

```
set.seed(284)
split <- sample.split(bank.norm$y, SplitRatio = 0.7)
bank.train <- subset(bank.norm, split == TRUE)
bank.test <- subset(bank.norm, split == FALSE)
table(bank.train$y)
```

```
##
##     0     1
## 23851  3259
```

Recordemos que nuestro data set está desbalanceado respecto de la variable objetivo `y`. En nuestro conjunto de entrenamiento tenemos 23,851 encuestados que no se han suscrito y solamente 3,259 que sí se han suscrito. Para resolver este problema utilizaremos una técnica de oversampling que consiste en crear observaciones artificiales de la clase minoritaria, en este caso los que se suscriben.

```
X <- bank.train[,-33]
Y <- as.factor(bank.train$y)
res_SMOTE <- ubSMOTE(X, Y, perc.over = 500, k = 10, perc.under = 0, verbose = TRUE)
bank.train.2 <- rbind(bank.train,data.frame(res_SMOTE$X,y=res_SMOTE$Y))
bank.train.2$y <- as.factor(bank.train.2$y)
```

A continuación realizaremos "tuning" para encontrar el valor óptimo del parámetro coste para el SVM con kernel lineal y empleando 10-Fold Cross Validation. Los valores del parámetro coste que pasaremos son $\{0.001, 0.01, 0.1, 1, 5, 10, 100\}$.

```
set.seed(284)
svm_lineal <- tune(svm, y~.,data=bank.train.2 ,kernel ="linear",
              ranges =list(cost=c(0.001,0.01,0.1, 1,5,10,100)))
saveRDS(svm_lineal, "./svm_lineal.rds")
```

La ejecución del algoritmo ha durado más de 30 horas, con lo cual es inviable ejecutarlo de nuevo tanto para pasarlo a formato html y/o pdf como para que otros lo ejecuten. La solución ha sido guardar los resultados en un archivo .rds llamado "svm_lineal.rds" y que llamaremos para trabajar con ellos.

```
svm_lineal <- readRDS("svm_lineal.rds")
summary(svm_lineal)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.1796032
```

```
##
## - Detailed performance results:
##    cost     error  dispersion
## 1 1e-03 0.1826034 0.004722595
## 2 1e-02 0.1810604 0.004845860
## 3 1e-01 0.1808460 0.004898873
## 4 1e+00 0.1808889 0.004800992
## 5 5e+00 0.1809746 0.004830191
## 6 1e+01 0.1808889 0.004852118
## 7 1e+02 0.1796032 0.004127473
```

Vemos que el valor óptimo del parámetro coste es 100, con una mejor performance de 0.1796032. Vamos a realizar la matriz de confusión correspondiente.

```
svm_lineal_pred <- predict(svm_lineal$best.model, bank.test)
tab_svm_lineal <- confusionMatrix(as.factor(svm_lineal_pred), as.factor(bank.test$y), positive = "1")
tab_svm_lineal
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8414  358
##          1 1808 1039
##
##                Accuracy : 0.8136
##                  95% CI : (0.8064, 0.8206)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3915
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.74374
##             Specificity : 0.82313
##          Pos Pred Value : 0.36495
##          Neg Pred Value : 0.95919
##              Prevalence : 0.12023
##          Detection Rate : 0.08942
##    Detection Prevalence : 0.24503
##       Balanced Accuracy : 0.78343
##
##        'Positive' Class : 1
##
```

```
tab_svm_lineal$byClass["F1"]
```

```
##        F1
## 0.4896324
```

Obtenemos una accuracy del 0.8135812, una sensibilidad del 0.7437366 y una F-medida del 0.4896324. Veamos para SVM con kernel radiales y polinomiales.

3.2 SVM con kernel radial

Para realizar el SVM con kernel radial elegimos un coste 100 y una $\gamma = 0.1$. De nuevo la ejecución del algoritmo es muy largo, con lo cual guardaremos el modelo en un archivo .rds llamado "svm_rbf.rds".

```
svm_rbf <- svm(y~.,data=bank.train.2 ,kernel ="radial", cost = 100, gamma = 0.1)
saveRDS(svm_rbf, "./svm_rbf.rds")
```

```
svm_rbf <- readRDS("svm_rbf.rds")
svm_rbf_pred <- predict(svm_rbf, bank.test)
tab_svm_rbf <- confusionMatrix(as.factor(svm_rbf_pred), as.factor(bank.test$y), positive = "1")
tab_svm_rbf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9203  758
##          1 1019  639
##
##                Accuracy : 0.8471
##                  95% CI : (0.8404, 0.8536)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.331
##
##  Mcnemar's Test P-Value : 6.925e-10
##
##             Sensitivity : 0.4574
##             Specificity : 0.9003
##          Pos Pred Value : 0.3854
##          Neg Pred Value : 0.9239
##              Prevalence : 0.1202
##          Detection Rate : 0.0550
##    Detection Prevalence : 0.1427
##       Balanced Accuracy : 0.6789
##
##        'Positive' Class : 1
##
```

```
tab_svm_rbf$byClass["F1"]
```

```
##        F1
## 0.4183306
```

Vemos algo diferente aquí. El accuracy ha subido hasta un valor de 0.8470608 pero la sensibilidad (0.4574087) y la F-medida (0.4183306) tienen unos valores muy bajos. Observamos que aunque el accuracy haya subido, debido a que clasifica más observaciones en la clase "0", los resultados son peores en relación con las otras medidas.

3.3 SVM con kernel polinomial

```
svm_poly_1 <- svm(y~.,data=bank.train.2 ,kernel ="polynomial", cost = 100, gamma = 0.1, degree = 2)
saveRDS(svm_poly_1, "./svm_poly_1.rds")
```

```
svm_poly_1 <- readRDS("svm_rbf.rds")
svm_poly_pred_1 <- predict(svm_poly_1, bank.test)
tab_svm_poly_1 <- confusionMatrix(as.factor(svm_poly_pred_1), as.factor(bank.test$y), positive = "1")
tab_svm_poly_1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9203  758
##          1 1019  639
##
##                Accuracy : 0.8471
##                  95% CI : (0.8404, 0.8536)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.331
##
##  Mcnemar's Test P-Value : 6.925e-10
##
##             Sensitivity : 0.4574
##             Specificity : 0.9003
##          Pos Pred Value : 0.3854
##          Neg Pred Value : 0.9239
##              Prevalence : 0.1202
##          Detection Rate : 0.0550
##    Detection Prevalence : 0.1427
##       Balanced Accuracy : 0.6789
##
##        'Positive' Class : 1
##
```

```
tab_svm_poly_1$byClass["F1"]
```

```
##        F1
## 0.4183306
```

```
svm_poly_2 <- svm(y~.,data=bank.train.2 ,kernel ="polynomial", cost = 100, gamma = 0.1, degree = 3)
saveRDS(svm_poly_2, "./svm_poly_2.rds")
```

```
svm_poly_2 <- readRDS("svm_rbf.rds")
svm_poly_pred_2 <- predict(svm_poly_2, bank.test)
tab_svm_poly_2 <- confusionMatrix(as.factor(svm_poly_pred_2), as.factor(bank.test$y), positive = "1")
tab_svm_poly_2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9203  758
##          1 1019  639
##
##                Accuracy : 0.8471
##                  95% CI : (0.8404, 0.8536)
```

```
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.331
##
##  Mcnemar's Test P-Value : 6.925e-10
##
##             Sensitivity : 0.4574
##             Specificity : 0.9003
##          Pos Pred Value : 0.3854
##          Neg Pred Value : 0.9239
##              Prevalence : 0.1202
##          Detection Rate : 0.0550
##    Detection Prevalence : 0.1427
##       Balanced Accuracy : 0.6789
##
##        'Positive' Class : 1
##
```

```
tab_svm_poly_2$byClass["F1"]
```

```
##        F1
## 0.4183306
```

Nos da exactamente el mismo resultado que el SVM con kernel radial.

4 Otros modelos de Machine Learning

4.1 Regresión logística

Primero de todo vamos a entrenar el modelo con el conjunto de entrenamiento, realizando la regresión logística múltiple teniendo a la variable y como variable de respuesta y todas las otras variables como explicativas (quitaremos un grupo para cada variable dummy y será nuestro grupo de control). Utilizaremos el método de eliminación hacia atrás para encontrar el modelo más óptimo.

```
reg_log <- glm(y~. -job_Services-marital_single-education_tertiary-default_yes-housing_yes-loan_yes-
                month_sep, data = bank.train.2, family = binomial)
summary(reg_log)
```

```
##
## Call:
## glm(formula = y ~ . - job_Services - marital_single - education_tertiary -
##     default_yes - housing_yes - loan_yes - month_sep, family = binomial,
##     data = bank.train.2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.5695  -0.6360  -0.2096   0.6803   2.6676
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.47907    0.16781  -8.814  < 2e-16 ***
## age                -0.12008    0.08006  -1.500    0.134
## balance             0.77513    0.17602   4.404 1.06e-05 ***
```

```
## day                 -0.43857    0.05150  -8.517  < 2e-16 ***
## duration            23.88972    0.25206  94.777  < 2e-16 ***
## campaign            -1.07271    0.05106 -21.009  < 2e-16 ***
## job_No_workers       0.58916    0.04204  14.013  < 2e-16 ***
## job_Self_employed    0.14566    0.03046   4.781 1.74e-06 ***
## marital_divorced    -0.28443    0.04748  -5.991 2.09e-09 ***
## marital_married     -0.29073    0.03176  -9.154  < 2e-16 ***
## education_primary   -0.84847    0.04770 -17.789  < 2e-16 ***
## education_secondary -0.31688    0.03057 -10.366  < 2e-16 ***
## default_no           0.73326    0.12110   6.055 1.40e-09 ***
## housing_no           0.94472    0.02885  32.751  < 2e-16 ***
## loan_no              0.81473    0.04104  19.853  < 2e-16 ***
## month_apr           -1.36262    0.10073 -13.527  < 2e-16 ***
## month_aug           -2.18963    0.09809 -22.323  < 2e-16 ***
## month_dec           -0.27338    0.18101  -1.510    0.131
## month_feb           -1.58231    0.10200 -15.514  < 2e-16 ***
## month_jan           -2.71192    0.11995 -22.609  < 2e-16 ***
## month_jul           -2.43601    0.09893 -24.623  < 2e-16 ***
## month_jun           -2.12445    0.09857 -21.552  < 2e-16 ***
## month_mar            0.56541    0.13746   4.113 3.90e-05 ***
## month_may           -2.65928    0.09670 -27.499  < 2e-16 ***
## month_nov           -2.25410    0.10070 -22.385  < 2e-16 ***
## month_oct           -0.13771    0.11975  -1.150    0.250
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 64667  on 46663  degrees of freedom
## Residual deviance: 40429  on 46638  degrees of freedom
## AIC: 40481
##
## Number of Fisher Scoring iterations: 5
```

La interpretación de los coeficientes, tomando `balance` como ejemplo, sería: un aumento de una unidad en la variable `balance` (una unidad más de saldo medio anual en la cuenta) hace aumentar el logaritmo de la probabilidad de suscribirse al depósito a plazo en 0.775.

Vemos que la mayoría de las variables son significativas al menos a un 5% de nivel de significación (debido a los p-valores), las únicas variables que no son significativas son: `age`, `month_dec` y `month_oct`.

Ahora vamos a realizar la predicción del modelo con el conjunto de testing. La predicción es en probabilidades, consideraremos que si es mayor que 0.5 será que se suscribe y, si es menor que 0.5, es que no se sucriibe.

```
reg_log_probs <- predict(reg_log, newdata = bank.test, type = "response")
reg_log_pred <- ifelse(reg_log_probs > 0.5, 1, 0)
tab_log <- confusionMatrix(as.factor(reg_log_pred), as.factor(bank.test$y), positive = "1")
tab_log
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8494  367
##          1 1728 1030
##
```

```
##                Accuracy : 0.8197
##                  95% CI : (0.8126, 0.8266)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.73729
##             Specificity : 0.83095
##          Pos Pred Value : 0.37346
##          Neg Pred Value : 0.95858
##              Prevalence : 0.12023
##          Detection Rate : 0.08865
##    Detection Prevalence : 0.23737
##       Balanced Accuracy : 0.78412
##
##        'Positive' Class : 1
##
```

```
tab_log$byClass["F1"]
```

```
##        F1
## 0.4957882
```

El accuracy que nos sale es 0.8196919, la sensibilidad es 0.7372942 y, finalmente, el valor de la F-medida es 0.4957882.

Vamos a eliminar las variables `age`, `month_dec` y `month_oct` del modelo de regresión logística por no ser significativas.

```
reg_log_1<- glm(y~. -job_Services-marital_single-education_tertiary-default_yes-housing_yes-loan_yes-
                  month_sep-age-month_dec-month_oct, data = bank.train.2, family = binomial)
summary(reg_log_1)
```

```
##
## Call:
## glm(formula = y ~ . - job_Services - marital_single - education_tertiary -
##     default_yes - housing_yes - loan_yes - month_sep - age -
##     month_dec - month_oct, family = binomial, data = bank.train.2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.5734  -0.6363  -0.2095   0.6804   2.6596
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.60308    0.15118 -10.604  < 2e-16 ***
## balance              0.75336    0.17552   4.292 1.77e-05 ***
## day                 -0.44310    0.05115  -8.663  < 2e-16 ***
## duration            23.89255    0.25205  94.794  < 2e-16 ***
## campaign            -1.07376    0.05102 -21.046  < 2e-16 ***
## job_No_workers       0.57495    0.04087  14.067  < 2e-16 ***
## job_Self_employed    0.14505    0.03045   4.764 1.90e-06 ***
## marital_divorced    -0.31432    0.04344  -7.236 4.63e-13 ***
```

```
## marital_married     -0.31429    0.02767 -11.359  < 2e-16 ***
## education_primary   -0.85751    0.04727 -18.140  < 2e-16 ***
## education_secondary -0.31789    0.03056 -10.403  < 2e-16 ***
## default_no           0.73123    0.12107   6.040 1.54e-09 ***
## housing_no           0.93995    0.02866  32.794  < 2e-16 ***
## loan_no              0.81607    0.04104  19.887  < 2e-16 ***
## month_apr           -1.25437    0.06816 -18.404  < 2e-16 ***
## month_aug           -2.08485    0.06526 -31.948  < 2e-16 ***
## month_feb           -1.47562    0.07169 -20.585  < 2e-16 ***
## month_jan           -2.60379    0.09341 -27.875  < 2e-16 ***
## month_jul           -2.32671    0.06583 -35.346  < 2e-16 ***
## month_jun           -2.01755    0.06647 -30.351  < 2e-16 ***
## month_mar            0.67006    0.11611   5.771 7.89e-09 ***
## month_may           -2.54917    0.06258 -40.736  < 2e-16 ***
## month_nov           -2.14615    0.06836 -31.397  < 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: 64667  on 46663  degrees of freedom
## Residual deviance: 40433  on 46641  degrees of freedom
## AIC: 40479
##
## Number of Fisher Scoring iterations: 5
```

Ahora todas las variabls son significativas al 5%.

```
reg_log_probs_1 <- predict(reg_log_1, newdata = bank.test, type = "response")
reg_log_pred_1 <- ifelse(reg_log_probs_1 > 0.5, 1, 0)
tab_log_1 <- confusionMatrix(as.factor(reg_log_pred_1), as.factor(bank.test$y), positive = "1")
tab_log_1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8497  367
##          1 1725 1030
##
##                Accuracy : 0.82
##                  95% CI : (0.8128, 0.8269)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4005
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.73729
##             Specificity : 0.83125
##          Pos Pred Value : 0.37387
##          Neg Pred Value : 0.95860
##              Prevalence : 0.12023
##          Detection Rate : 0.08865
```

```
##    Detection Prevalence : 0.23711
##       Balanced Accuracy : 0.78427
##
##        'Positive' Class : 1
##
```

```
tab_log_1$byClass["F1"]
```

```
##        F1
## 0.4961464
```

Ahora los valores que tenemos son:

- Acurracy: 0.8199501
- Sensibilidad: 0.7372942
- F-medida: 0.4961464

Los resultados mejoran ligeralmente que el modelo sin quitar las variables no significativas.

4.2 KNN (K-nearest neighbors)

El método de K-Nearest Neighbors es otro método de clasificación. El algoritmo reconoce patrones en los datos sin un aprendizaje específico, consiguiendo un criterio de agrupamiento de los datos a partir de un conjunto de entrenamiento.

A continuación vamos a clasificar nuestras observaciones, para más tarde construir la matriz de confusión.

```
set.seed(248)
bank_knn <- knn(train = bank.train.2[,-33], test = bank.test[,-33], cl = bank.train.2$y, k = 10)
tab_knn <- confusionMatrix(as.factor(bank_knn), as.factor(bank.test$y), positive = "1")
```

```
tab_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8167  542
##          1 2055  855
##
##                Accuracy : 0.7765
##                  95% CI : (0.7688, 0.784)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2801
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.61203
##             Specificity : 0.79896
##          Pos Pred Value : 0.29381
##          Neg Pred Value : 0.93777
##              Prevalence : 0.12023
##          Detection Rate : 0.07359
##    Detection Prevalence : 0.25045
```

```
##       Balanced Accuracy : 0.70549
##
##        'Positive' Class : 1
##
```

```
tab_knn$byClass["F1"]
```

```
##        F1
## 0.3970281
```

Los valores que tenemos son:

- Acurracy: 0.7764868
- Sensibilidad: 0.6120258
- F-medida: 0.3970281

4.3 Decision tree

Como nuestra variable dependiente es binaria, vamos a construir un árbol de clasificación en vez de un árbol de regresión. Este algoritmo, de forma general, separa los datos en grupos (las clases de la variable dependiente) utilizando la mejor variable explicativa en cada nodo. Vamos a construir el árbol de clasificación con el conjunto de entrenamiento cogiendo el parámetro de complejidad (cp) igual a 0. Luego haremos la predicción con el conjunto de test, para más adelante dibujar la matriz de confusión.

```
bank_tree <- rpart(y ~., data = bank.train.2, method = "class",
                   control = rpart.control(cp = 0))
tree_prediction <- predict(bank_tree, newdata = bank.test, type = "class")
tab_tree <- confusionMatrix(as.factor(tree_prediction), as.factor(bank.test$y), positive = "1")
tab_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8771  463
##          1 1451  934
##
##                Accuracy : 0.8353
##                  95% CI : (0.8284, 0.842)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4035
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.66858
##             Specificity : 0.85805
##          Pos Pred Value : 0.39161
##          Neg Pred Value : 0.94986
##              Prevalence : 0.12023
##          Detection Rate : 0.08039
##    Detection Prevalence : 0.20527
##       Balanced Accuracy : 0.76331
##
```

```
##        'Positive' Class : 1
##
```

```
tab_tree$byClass["F1"]
```

```
##        F1
## 0.4939186
```

Los valores de las medidas son:

- Acurracy: 0.8352698
- Sensibilidad: 0.6685755
- F-medida: 0.4939186

A continuación vamos a podar el árbol. Para ello, vamos a hacer un análisis del parámetro de complejidad. Cogeremos el valor del parámetro que minimice el error de la validación cruzada (xerror).

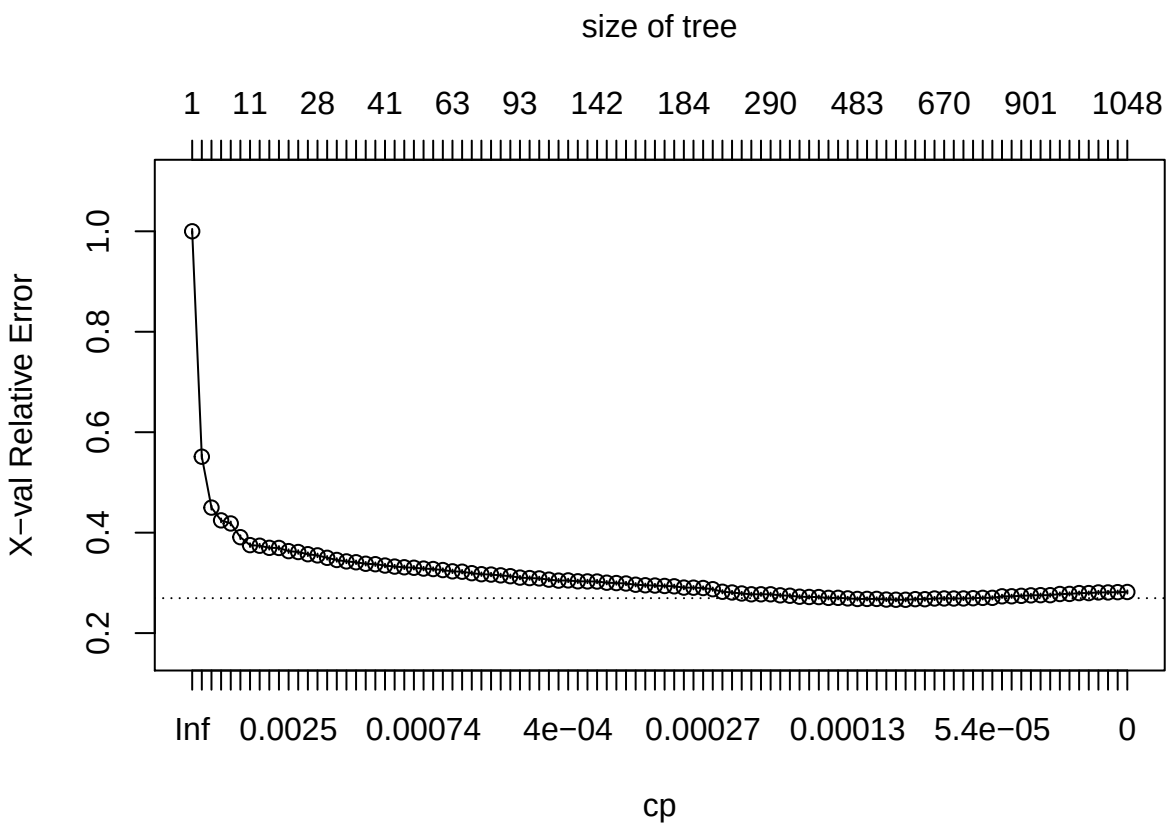
```
printcp(bank_tree)
```

```
##
## Classification tree:
## rpart(formula = y ~ ., data = bank.train.2, method = "class",
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
##  [1] age                 balance             campaign
##  [4] day                 duration            education_primary
##  [7] education_secondary education_tertiary  housing_no
## [10] housing_yes         job_No_workers      job_Self_employed
## [13] job_Services        loan_no             loan_yes
## [16] marital_divorced    marital_married     marital_single
## [19] month_apr           month_aug           month_dec
## [22] month_feb           month_jan           month_jul
## [25] month_jun           month_mar           month_may
## [28] month_nov           month_oct           month_sep
##
## Root node error: 22813/46664 = 0.48888
##
## n= 46664
##
##            CP nsplit rel error  xerror      xstd
## 1  4.5088e-01      0   1.00000 1.00000 0.0047334
## 2  5.0103e-02      1   0.54912 0.55118 0.0042012
## 3  1.8148e-02      3   0.44891 0.44988 0.0039221
## 4  5.5232e-03      4   0.43076 0.42445 0.0038399
## 5  4.7634e-03      5   0.42524 0.41831 0.0038193
## 6  3.1999e-03      9   0.39938 0.39101 0.0037233
## 7  3.1123e-03     10   0.39618 0.37518 0.0036646
## 8  2.8273e-03     11   0.39307 0.37395 0.0036600
## 9  2.8054e-03     20   0.35826 0.36979 0.0036441
## 10 2.7177e-03     21   0.35546 0.36957 0.0036432
## 11 2.3452e-03     22   0.35274 0.36321 0.0036186
## 12 1.7534e-03     24   0.34805 0.36142 0.0036116
## 13 1.5780e-03     25   0.34629 0.35708 0.0035944
## 14 1.4027e-03     27   0.34314 0.35475 0.0035852
## 15 1.2274e-03     29   0.34033 0.34989 0.0035656
```

```
## 16 1.0959e-03     30   0.33910 0.34564 0.0035483
## 17 1.0301e-03     31   0.33801 0.34274 0.0035365
## 18 9.6436e-04     33   0.33595 0.34103 0.0035294
## 19 9.3514e-04     34   0.33498 0.33810 0.0035172
## 20 9.2053e-04     39   0.32933 0.33718 0.0035134
## 21 8.3286e-04     40   0.32841 0.33450 0.0035021
## 22 8.0364e-04     45   0.32424 0.33244 0.0034934
## 23 7.8902e-04     48   0.32183 0.33104 0.0034875
## 24 7.4519e-04     54   0.31710 0.33012 0.0034836
## 25 7.3058e-04     55   0.31635 0.32845 0.0034765
## 26 7.0135e-04     58   0.31416 0.32753 0.0034725
## 27 6.7944e-04     59   0.31346 0.32556 0.0034640
## 28 6.5752e-04     62   0.31136 0.32297 0.0034528
## 29 6.1369e-04     63   0.31070 0.32227 0.0034498
## 30 5.9177e-04     64   0.31009 0.31934 0.0034370
## 31 5.6985e-04     66   0.30890 0.31728 0.0034279
## 32 5.5524e-04     69   0.30697 0.31657 0.0034248
## 33 5.2602e-04     78   0.29891 0.31500 0.0034178
## 34 4.8218e-04     81   0.29733 0.31329 0.0034102
## 35 4.6465e-04     92   0.29133 0.31052 0.0033978
## 36 4.6026e-04    100   0.28677 0.30956 0.0033935
## 37 4.3835e-04    102   0.28585 0.30886 0.0033903
## 38 4.1643e-04    107   0.28365 0.30640 0.0033792
## 39 4.0547e-04    109   0.28282 0.30452 0.0033706
## 40 3.9451e-04    113   0.28120 0.30491 0.0033724
## 41 3.7990e-04    120   0.27844 0.30312 0.0033642
## 42 3.7259e-04    127   0.27576 0.30290 0.0033632
## 43 3.6712e-04    141   0.27002 0.30272 0.0033624
## 44 3.6164e-04    149   0.26708 0.30040 0.0033517
## 45 3.5068e-04    153   0.26564 0.29965 0.0033483
## 46 3.3972e-04    160   0.26318 0.29878 0.0033442
## 47 3.2876e-04    164   0.26182 0.29632 0.0033328
## 48 3.2145e-04    166   0.26117 0.29523 0.0033277
## 49 3.1780e-04    171   0.25954 0.29470 0.0033252
## 50 3.0684e-04    175   0.25827 0.29387 0.0033213
## 51 2.9223e-04    180   0.25674 0.29317 0.0033180
## 52 2.8493e-04    183   0.25586 0.29058 0.0033058
## 53 2.8054e-04    188   0.25437 0.29045 0.0033051
## 54 2.6301e-04    193   0.25297 0.28983 0.0033022
## 55 2.4109e-04    209   0.24867 0.28742 0.0032907
## 56 2.2794e-04    233   0.24254 0.28234 0.0032662
## 57 2.1917e-04    240   0.24078 0.28089 0.0032591
## 58 2.0821e-04    270   0.23381 0.27892 0.0032495
## 59 2.0456e-04    278   0.23215 0.27730 0.0032415
## 60 2.0164e-04    284   0.23092 0.27730 0.0032415
## 61 1.9726e-04    289   0.22991 0.27730 0.0032415
## 62 1.8995e-04    299   0.22785 0.27519 0.0032311
## 63 1.7534e-04    305   0.22671 0.27440 0.0032272
## 64 1.6073e-04    343   0.21961 0.27243 0.0032174
## 65 1.5342e-04    348   0.21878 0.27191 0.0032147
## 66 1.4612e-04    366   0.21602 0.27173 0.0032139
## 67 1.3881e-04    377   0.21435 0.27015 0.0032059
## 68 1.3150e-04    384   0.21330 0.27024 0.0032064
## 69 1.2274e-04    462   0.20252 0.26910 0.0032006
```

```
## 70 1.2055e-04    482   0.19954 0.26796 0.0031949
## 71 1.1689e-04    491   0.19818 0.26809 0.0031955
## 72 1.1272e-04    494   0.19783 0.26809 0.0031955
## 73 1.0959e-04    504   0.19660 0.26678 0.0031889
## 74 1.0228e-04    534   0.19327 0.26643 0.0031871
## 75 9.4975e-05    549   0.19165 0.26638 0.0031869
## 76 8.7669e-05    576   0.18862 0.26761 0.0031931
## 77 7.8902e-05    636   0.18257 0.26752 0.0031927
## 78 7.6711e-05    651   0.18134 0.26888 0.0031995
## 79 7.5145e-05    669   0.17990 0.26897 0.0032000
## 80 7.3058e-05    676   0.17937 0.26888 0.0031995
## 81 6.5752e-05    691   0.17828 0.26897 0.0032000
## 82 5.8446e-05    729   0.17578 0.26950 0.0032026
## 83 5.4793e-05    776   0.17201 0.27015 0.0032059
## 84 5.2602e-05    787   0.17139 0.27024 0.0032064
## 85 4.3835e-05    792   0.17113 0.27327 0.0032215
## 86 3.6529e-05    878   0.16710 0.27331 0.0032217
## 87 3.2876e-05    884   0.16688 0.27419 0.0032261
## 88 2.9223e-05    900   0.16635 0.27515 0.0032309
## 89 2.6301e-05    909   0.16609 0.27554 0.0032329
## 90 2.5048e-05    919   0.16583 0.27554 0.0032329
## 91 2.1917e-05    926   0.16565 0.27787 0.0032443
## 92 1.7534e-05    962   0.16486 0.27804 0.0032452
## 93 1.4612e-05    967   0.16477 0.27967 0.0032531
## 94 1.2524e-05    997   0.16434 0.27967 0.0032531
## 95 1.0959e-05   1011   0.16416 0.28098 0.0032596
## 96 8.7669e-06   1019   0.16407 0.28098 0.0032596
## 97 7.3058e-06   1029   0.16399 0.28155 0.0032623
## 98 0.0000e+00   1047   0.16385 0.28186 0.0032638
```

```
plotcp(bank_tree)
```




```
min_best_cp <- bank_tree$cptable[which.min(bank_tree$cptable[,"xerror"]),"CP"]
```

Con el valor del cp óptimo, podamos el árbol y construimos la predicción junto con la matriz de confusión.

```
bank_tree_fit <- prune(bank_tree, min_best_cp)
tree_fit_prediction <- predict(bank_tree_fit, newdata = bank.test, type = "class")
tab_fit_tree <- confusionMatrix(as.factor(tree_fit_prediction), as.factor(bank.test$y), positive = "1")
tab_fit_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8682  355
##          1 1540 1042
##
##                Accuracy : 0.8369
##                  95% CI : (0.8301, 0.8436)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4357
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.74588
##             Specificity : 0.84934
##          Pos Pred Value : 0.40356
##          Neg Pred Value : 0.96072
```

```
##              Prevalence : 0.12023
##          Detection Rate : 0.08968
##    Detection Prevalence : 0.22222
##       Balanced Accuracy : 0.79761
##
##        'Positive' Class : 1
##
```

```
tab_fit_tree$byClass["F1"]
```

```
##        F1
## 0.5237497
```

Conseguimos unos resultados mejores que el árbol sin podar:

- Acurracy: 0.8369051
- Sensibilidad: 0.745884
- F-medida: 0.5237497

4.4 LDA (Análisis Discriminante Lineal)

El análisis discriminante lineal es un método que se utiliza para encontrar una combinación lineal de cualidades que caracterizan o separan dos o más clases de objetos. Se suele utilizar este método para reducir la dimensionalidad del conjunto de datos, pero también se usa como un clasificador lineal.

Utilizamos la función `lda` para realizar el modelo, para más tarde predecir los valores con el conjunto de test y conseguir la matriz de confusión.

```
bank_lda <- lda(y ~., data = bank.train.2)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda_pred <- predict(bank_lda, bank.test, type = "response")
tab_lda <- confusionMatrix(as.factor(lda_pred$class), as.factor(bank.test$y), positive = "1")
tab_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8672  411
##          1 1550  986
##
##                Accuracy : 0.8312
##                  95% CI : (0.8243, 0.838)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4099
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.70580
##             Specificity : 0.84837
##          Pos Pred Value : 0.38880
##          Neg Pred Value : 0.95475
```

```
##              Prevalence : 0.12023
##          Detection Rate : 0.08486
##    Detection Prevalence : 0.21826
##       Balanced Accuracy : 0.77708
##
##        'Positive' Class : 1
##
```

```
tab_lda$byClass["F1"]
```

```
##        F1
## 0.5013984
```

Los valores que nos salen son decentes y son:

- Acurracy: 0.8312247
- Sensibilidad: 0.7057981
- F-medida: 0.5013984

4.5 Radom Forest

Un solo árbol de decisión no es suficiente, ya que sufren de problemas de sesgo y varianza en las predicciones. Para poder mejorar tanto los problemas comentados y una mejor precisión, vamos a introducir el concepto de Random Forest. Random Forest es un método tipo ensemble que está formado por un grupo de modelos predictivos (clasificatorios en nuestro caso) alcanzando una precisión y una estabilidad mejores. Los árboles sufren problemas de sesgo y varianza en las predicciones; Random Forest forma parte de los métodos de Bagging y éstos funcionan de la siguiente manera:

- Crear muchos subconjuntos de datos.
- Construir múltiples modelos.
- Combinar los modelos construidos.

Random Forest crea un grupo de modelos aparentemente débiles (múltiples árboles de decisión), para combinarlos y transformarlos en un modelo más potente.

Usamos la función `randomForest` del paquete `randomForest` asignando en 1,000 el número de árboles.

```
bank_rf <- randomForest(y ~ ., data = bank.train.2, ntree = 1000)
rf_pred <- predict(bank_rf, newdata = bank.test, type = 'class')
tab_rf <- confusionMatrix(as.factor(rf_pred), as.factor(bank.test$y), positive = "1")
tab_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9135  430
##          1 1087  967
##
##                Accuracy : 0.8694
##                  95% CI : (0.8632, 0.8755)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 0.9997
##
##                   Kappa : 0.487
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.69220
##             Specificity : 0.89366
##          Pos Pred Value : 0.47079
##          Neg Pred Value : 0.95504
##              Prevalence : 0.12023
##          Detection Rate : 0.08323
##    Detection Prevalence : 0.17678
##       Balanced Accuracy : 0.79293
##
##        'Positive' Class : 1
##
```

```
tab_rf$byClass["F1"]
```

```
##        F1
## 0.5604173
```

Definitivamente estos son los mejores resultados alcanzados hasta este momento:

- Acurracy: 0.869438
- Sensibilidad: 0.6921976
- F-medida: 0.5604173

4.6 ANN (Redes neuronales artificiales)

Se han realizado pruebas con diferentes valores de los parámetros y las conclusiones más importantes son:

- El algoritmo "backpropagation" no se utiliza ya que no converge.
- Se ha probado con ratio de aprendizaje más pequeños, como 0.01 0 0.001, y tampoco converge.
- Nos hemos visto obligado a poner un "threshold" de 1 para la convergencia del algoritmo.

Por todo esto, los valores de los parámetros que son fijos son:

- threshold = 1
- err.fct = "sse"
- linear.output = FALSE, ya que se trata de clasificación
- learningrate = 0.1
- act.fct = "logistic"

Ejecutamos el algoritmo con una capa de 3 neuronas.

```
set.seed(284)
ANN <- neuralnet(y ~. , data = bank.train.2,
                 hidden = c(3),
                 threshold = 1,
                 err.fct="sse",
                 linear.output=FALSE,
                 learningrate = 0.1,
                 act.fct = "logistic")

ANN_pred <- round(compute(ANN, bank.test[, -33])$net.result)
```

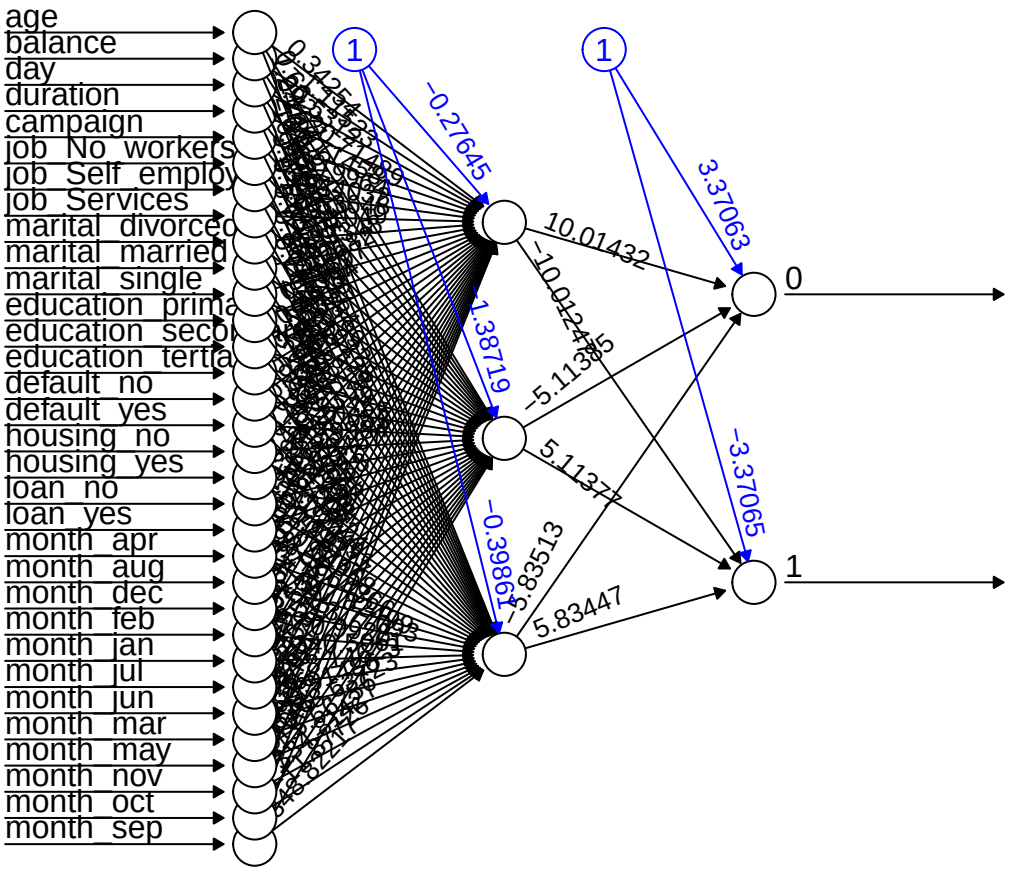
```r
cm_2 <- confusionMatrix(as.factor(ANN_pred[,2]), as.factor(bank.test$y), positive = "1")
cm_2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8289  245
##          1 1933 1152
##
##                Accuracy : 0.8125
##                  95% CI : (0.8053, 0.8196)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4177
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.82462
##             Specificity : 0.81090
##          Pos Pred Value : 0.37342
##          Neg Pred Value : 0.97129
##              Prevalence : 0.12023
##          Detection Rate : 0.09915
##    Detection Prevalence : 0.26551
##       Balanced Accuracy : 0.81776
##
##        'Positive' Class : 1
##
```

```r
cm_2$byClass["F1"]
```

```
##        F1
## 0.5140562
```

```r
plot(ANN, rep = "best")
```



Las medidas que conseguimos son:

- Acurracy: 0.8125484
- Sensibilidad: 0.8246242
- F-medida: 0.5140562

5 Conclusión

Vamos a poner en una tabla los resultados de los diferentes métodos que hemos empleado.

```
resultados <- data.frame("Accuracy" = c(unname(tab_svm_lineal$overall["Accuracy"]),
                                        unname(tab_log_1$overall["Accuracy"]),
                                        unname(tab_knn$overall["Accuracy"]),
                                        unname(tab_fit_tree$overall["Accuracy"]),
                                        unname(tab_lda$overall["Accuracy"]),
                                        unname(tab_rf$overall["Accuracy"]),
                                        unname(cm_2$overall["Accuracy"])),
                         "Sensitivity" = c(unname(tab_svm_lineal$byClass[1]),
                                           unname(tab_log_1$byClass[1]),
                                           unname(tab_knn$byClass[1]),
                                           unname(tab_fit_tree$byClass[1]),
                                           unname(tab_lda$byClass[1]),
                                           unname(tab_rf$byClass[1]),
                                           unname(cm_2$byClass[1])),
                         "F-medida" = c(unname(tab_svm_lineal$byClass[7]),
```

```
                                    unname(tab_log_1$byClass[7]),
                                    unname(tab_knn$byClass[7]),
                                    unname(tab_fit_tree$byClass[7]),
                                    unname(tab_lda$byClass[7]),
                                    unname(tab_rf$byClass[7]),
                                    unname(cm_2$byClass[7])))

rownames(resultados) <- c("SVM kernel lineal", "Regresión logística", "K-NN", "Decision tree", "LDA", "
resultados
```

```
##                       Accuracy Sensitivity  F.medida
## SVM kernel lineal   0.8135812   0.7437366 0.4896324
## Regresión logística 0.8199501   0.7372942 0.4961464
## K-NN                0.7764868   0.6120258 0.3970281
## Decision tree       0.8369051   0.7458840 0.5237497
## LDA                 0.8312247   0.7057981 0.5013984
## Random-Forest       0.8694380   0.6921976 0.5604173
## ANN                 0.8125484   0.8246242 0.5140562
```

Vemos que para el accuracy el mejor modelo es Random Forest con 0.869438, si miramos la sensibilidad el mejor modelo es ANN (Redes neuronales artificiales) con 0.8246242 y por último, si miramos la F-medida el mejor modelo también es Random Forest con 0.5604173. Concluimos que si hay que elegir uno de estos modelos, elegiríamos el Random Forest porque las medidas que nos han salido indican un mejor rendimiento de este modelo.