DATACARD - BANK MARKETING DATASET

Context
Find the best strategies to improve for the next marketing campaign. How can the financial institution have a greater effectiveness for future marketing campaigns? In order to answer this, we have to analyze the last marketing campaign the bank performed and identify the patterns that will help us find conclusions in order to develop future strategies.

Source
[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

Usability
8.24

License
CC0: Public Domain

Expected update frequency
Not specified

```
None
install.packages(c(
  "dplyr",     # Manipulación de datos
  "ggplot2",   # Visualización
  "readr",     # Lectura de CSV
  "caret",
  "MASS",
  "rpart",
  "class",
  "kernlab",
  "randomForest",
  "e1071",
  "pROC"
))
```

dplyr: Herramientas para manipular y transformar datos (filtrar, agrupar, resumir, etc.).
ggplot2: Paquete muy popular para crear gráficos de alta calidad basados en la gramática de los gráficos.
readr: Permite leer archivos de texto y CSV de forma rápida y eficiente.
caret: Framework para crear, entrenar y evaluar modelos de aprendizaje automático de forma unificada.
MASS: Contiene conjuntos de datos y funciones estadísticas, incluyendo modelos lineales y discriminantes.
rpart: Implementa árboles de decisión para clasificación y regresión.
class: Implementa el algoritmo k-Nearest Neighbors (kNN) para clasificación.
kernlab: Paquete para trabajar con métodos de aprendizaje basados en kernels, como máquinas de soporte vectorial (SVM).
randomForest: Permite entrenar modelos de bosques aleatorios para clasificación y regresión.

e1071: Incluye implementaciones de SVM, Naive Bayes y otras técnicas de machine learning.
pROC: Se utiliza para analizar curvas ROC (Receiver Operating Characteristic) y calcular métricas como AUC.

None

```
library(dplyr)
library(ggplot2)
library(readr)
library(caret)
library(MASS)
library(rpart)
library(class)
library(kernlab)
library(randomForest)
library(e1071)
library(pROC)
```

Cargamos todas las librerías

None

```
bank_data <- read.csv("bank.csv", sep = ",")
head(bank_data,10)
```

Cargamos el dataset en memoria e imprimimos los primeros 5 registros

Description: df [5 × 17]

	age <int>	job <chr>	marital <chr>	education <chr>	default <chr>	balance <int>	housing <chr>	loan <chr>
1	59	admin.	married	secondary	no	2343	yes	no
2	56	admin.	married	secondary	no	45	no	no
3	41	technician	married	secondary	no	1270	yes	no
4	55	services	married	secondary	no	2476	yes	no
5	54	admin.	married	tertiary	no	184	no	no

5 rows | 1-9 of 17 columns

Description: df [5 × 17]

contact <chr>	day <int>	month <chr>	duration <int>	campaign <int>	pdays <int>	previous <int>	poutcome <chr>
unknown	5	may	1042	1	-1	0	unknown
unknown	5	may	1467	1	-1	0	unknown
unknown	5	may	1389	1	-1	0	unknown
unknown	5	may	579	1	-1	0	unknown
unknown	5	may	673	2	-1	0	unknown

5 rows | 10-17 of 17 columns

Tenemos los siguientes campos:

1. **age**: edad del cliente
2. **job**: tipo de trabajo
3. **marital**: estado civil
4. **education**: nivel educativo
5. **default**: si tiene crédito en incumplimiento
6. **balance**: saldo promedio anual
7. **housing**: si tiene préstamo de vivienda
8. **loan**: si tiene préstamo personal
9. **contact**: tipo de comunicación usada
10. **day**, **month**: día y mes del último contacto
11. **duration**: duración del último contacto (segundos)
12. **campaign**: número de contactos durante la campaña actual
13. **pdays**: días desde el último contacto anterior (-1 si ninguno)
14. **previous**: número de contactos antes de la campaña actual
15. **poutcome**: resultado de la campaña anterior
16. **deposit:** variable objetivo (si el cliente hizo un depósito a plazo)

```
None
colSums(is.na(bank_data))
```

```
      age       job   marital education   default   balance   housing      loan   contact       day     month  duration 
        0         0         0         0         0         0         0         0         0         0         0         0 
 campaign     pdays  previous  poutcome   deposit 
        0         0         0         0         0 
```

Verificar valores nulos(no hay)

```
None
summary(bank_data, is.numeric)
```

Utilizamos el sumario estadístico de los datos numericos para tener una visión general:

```
      age          balance            day           duration       campaign          pdays            previous      
 Min.   :18.00   Min.   :-6847   Min.   : 1.00   Min.   :   2   Min.   : 1.000   Min.   : -1.00   Min.   : 0.0000  
 1st Qu.:32.00   1st Qu.:  122   1st Qu.: 8.00   1st Qu.: 138   1st Qu.: 1.000   1st Qu.: -1.00   1st Qu.: 0.0000  
 Median :39.00   Median :  550   Median :15.00   Median : 255   Median : 2.000   Median : -1.00   Median : 0.0000  
 Mean   :41.23   Mean   : 1529   Mean   :15.66   Mean   : 372   Mean   : 2.508   Mean   : 51.33   Mean   : 0.8326  
 3rd Qu.:49.00   3rd Qu.: 1708   3rd Qu.:22.00   3rd Qu.: 496   3rd Qu.: 3.000   3rd Qu.: 20.75   3rd Qu.: 1.0000  
 Max.   :95.00   Max.   :81204   Max.   :31.00   Max.   :3881   Max.   :63.000   Max.   :854.00   Max.   :58.0000  
```

```
None
cat_vars <- select_if(bank_data, Negate(is.numeric))
lapply(cat_vars, table)
```

Para los datos categoricos contaremos los valores unicos

```
$job

       admin.   blue-collar  entrepreneur     housemaid    management       retired self-employed      services       student    technician
         1334          1944           328           274          2566           778           405           923           360          1823
   unemployed       unknown
          357            70

$marital

divorced  married   single
    1293     6351     3518

$education

  primary secondary  tertiary   unknown
     1500      5476      3689       497

$default

   no   yes
10994   168

$housing

  no  yes
5881 5281

$loan

  no  yes
9702 1460

$contact

 cellular telephone   unknown
     8042       774      2346

$month

 apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
 923 1519  110  776  344 1514 1222  276 2824  943  392  319

$poutcome

failure   other success unknown
   1228     537    1071    8326

$deposit

  no  yes
5873 5289
```

Y rescatamos lo siguiente:
El dataset tiene 11 162 registros y múltiples variables categóricas y numéricas.
Este es un problema de clasificación, y la variable objetivo es deposit (yes/no).

Variable	Observación	Inferencia
age	18–95 años (media ≈ 41)	La mayoría de clientes está en edad laboral, rango amplio.
balance	media: 1529 €, rango: –6847 a 81 204 €	Hay valores negativos (descubiertos bancarios). Distribución muy asimétrica (algunos balances enormes).
day	1–31	Día del mes de la última campaña. No parece tener correlación fuerte por sí sola.
duration	media: 372 s, máx: 3881 s	Duración de la llamada, debemos tomar atencion especial a esta variable, puede ser la más predictiva del dataset, (una llamada mas larga puede significar mayor interes del cliente, y un deposito)
campaign	1–63	Número de contactos realizados durante la

		campaña. Muchos tienen 1 o 2, algunos clientes fueron contactados muchas veces.
pdays	media: 51.3, pero 75% de los casos es –1	Valor –1 indica que el cliente no fue contactado anteriormente.
previous	media: 0.83, máx: 58	Muy pocos contactos previos con el cliente. Distribución sesgada.

Variables categoricas

Variable	Ejemplos	Observación
job	admin., technician, management, etc.	Se podría agrupar o codificar como factor.
marital	married, single, divorced	Normalizada.
education	primary, secondary, tertiary, unknown	Lista clara, pero puede tener valores “unknown”.
default	yes/no	Casi todos “no”.
housing, loan	yes/no	Pueden afectar la disposición al depósito.
contact	celular, teléfono, desconocido	“unknown” podría indicar contacto nulo.
month	may, jun, jul...	Puede tener relación temporal con campañas.
poutcome	success, failure, unknown	Resultado de campañas anteriores.
deposit	yes/no	Variable objetivo (target).

Aspectos clave a considerar antes del modelado

Outliers: en balance y duration. Se debe revisar log-transformación o winsorización.

Categorías “unknown”: conviene codificarlas explícitamente o reemplazarlas con “missing”.

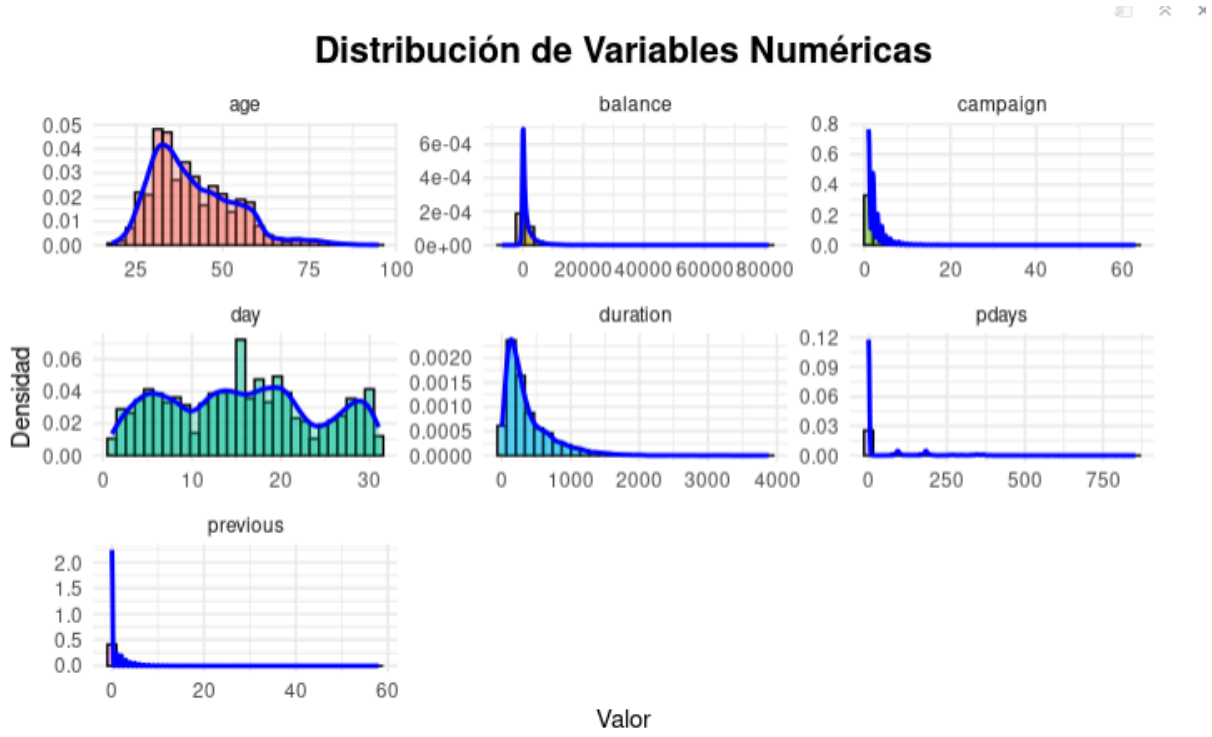
Desbalance de clases: conviene verificar la proporción de deposit == yes vs no.

Duration es muy influyente, pero no debe usarse para predicción futura (porque solo se conoce después de la llamada).

```
None
num_vars <- bank_data[, sapply(bank_data, is.numeric)]

# Convertir a formato largo sin pivot_longer
num_data_long <- data.frame(
  Variable = rep(names(num_vars), each = nrow(num_vars)),
  Valor = unlist(num_vars)
)

# Graficar histogramas suavizados (versión moderna)
ggplot(num_data_long, aes(x = Valor, fill = Variable)) +
  geom_histogram(aes(y = after_stat(density)), bins = 30, alpha = 0.5, color
= "black") +
  geom_density(alpha = 0.3, color = "blue", linewidth = 1) +
```

```
  facet_wrap(~Variable, scales = "free", ncol = 3) +
  theme_minimal() +
  labs(
    title = "Distribución de Variables Numéricas",
    x = "Valor",
    y = "Densidad"
  ) +
  theme(
    legend.position = "none",
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold")
```

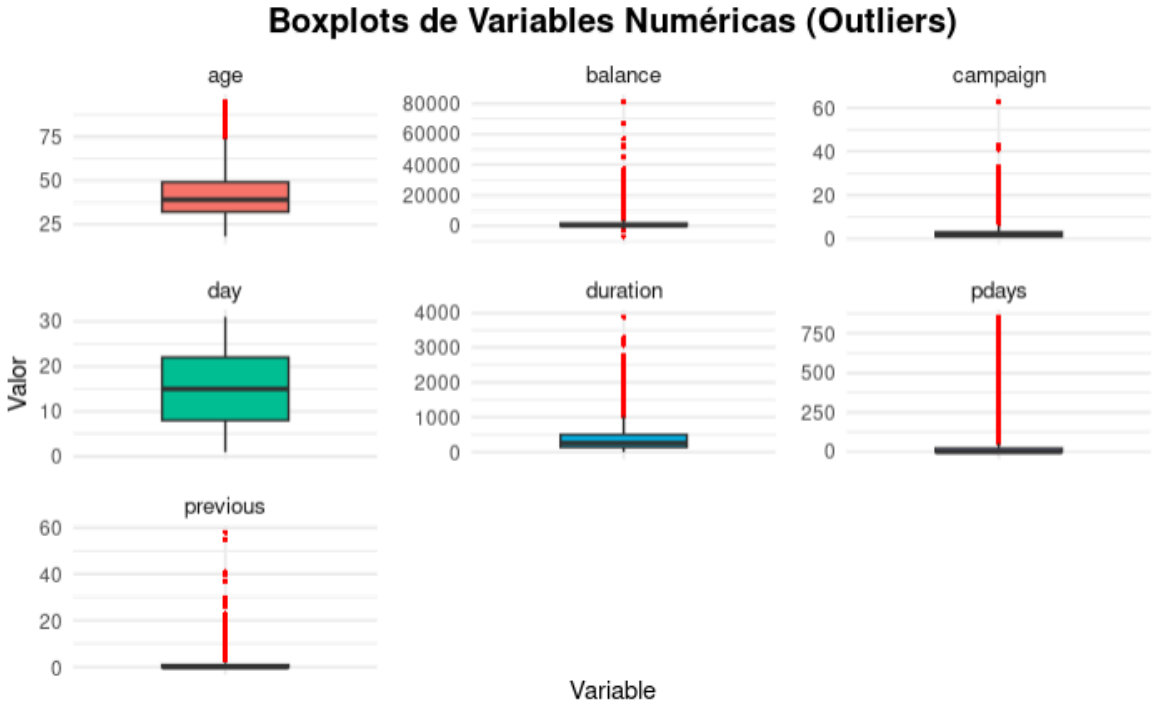
En este fragmento de codigo graficamos la distribución de las variables numéricas:
Filtra solo las variables numéricas del dataset.
Convierte los datos a formato largo (una columna para el nombre de la variable y otra para los valores).
Usa ggplot2 para crear histogramas con curvas de densidad suavizadas.
facet_wrap genera un gráfico separado por cada variable numérica.



Creamos boxplots para poder visualizar los outliers, observamos que:
pdays sigue apareciendo con outliers, y no es un error, es consecuencia de la estructura de los datos, la mayoría son -1 : la distribución está muy sesgada.

```
None
ggplot(num_data_long, aes(x = Variable, y = Valor, fill = Variable)) +
  geom_boxplot(
    outlier.colour = "red",
    outlier.shape = 16,
    outlier.size = 0.8,
```

```
  width = 0.5
) +
facet_wrap(~Variable, scales = "free", ncol = 3) +
theme_minimal(base_size = 11) +
labs(
  title = "Boxplots de Variables Numéricas (Outliers Reducidos)",
  x = "Variable",
  y = "Valor"
) +
theme(
  legend.position = "none",
  strip.text = element_text(size = 10),
  axis.text.x = element_blank(),
  plot.title = element_text(hjust = 0.5, size = 15, face = "bold"),
  panel.spacing = unit(0.8, "lines")
)
```



Podemos observar lo siguiente:
Ninguna de las distribuciones tiene forma de campana.
Las distribuciones de age(edad) y day(días) presentan mayor dispersión que las demás. La primera es (casi) unimodal y asimétrica positiva, mientras que la segunda es multimodal (con 3 picos principales).
Las distribuciones numéricas restantes presentan un pico alto cerca del origen, lo que significa que los valores más altos son menos frecuentes que los más bajos.
Esto sugiere que, antes de entrenar modelos de clasificación, podría ser útil escalar o normalizar los datos, o bien capar los outliers si afectan el rendimiento.

Revisamos el porcentaje de outliers en nuestros datos, para eso utilizaremos el metodo IQR

None

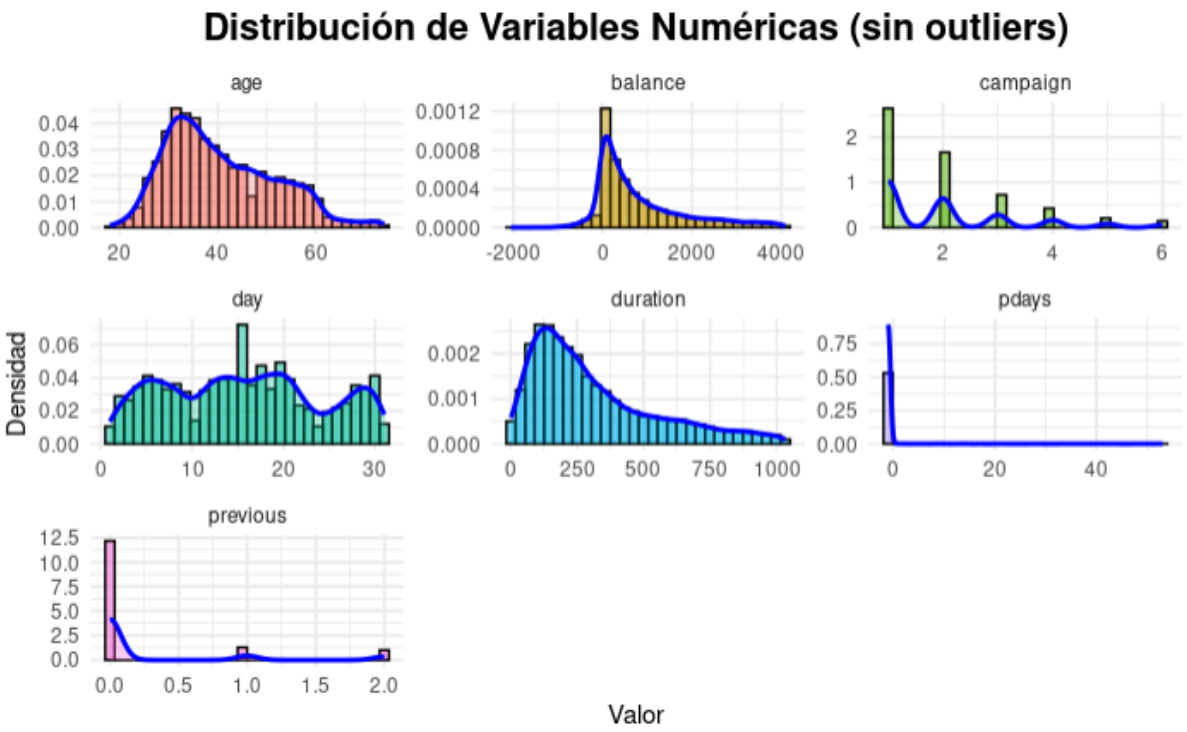
```
outlier_percentage <- function(x) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR_val <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR_val
  upper_bound <- Q3 + 1.5 * IQR_val
  outliers <- sum(x < lower_bound | x > upper_bound, na.rm = TRUE)
  perc <- (outliers / length(x)) * 100
  return(round(perc, 2))
}

numeric_vars <- bank_data[, sapply(bank_data, is.numeric)]
outlier_summary <- sapply(numeric_vars, outlier_percentage)

sort(outlier_summary, decreasing = TRUE)
```

```
pdays previous  balance duration campaign      age      day
24.64    11.27     9.45     5.70     5.38     1.53     0.00
```

None

```
numeric_vars <- bank_data[, sapply(bank_data, is.numeric)]

# Rremplazar outliers por NA
replace_outliers_with_na <- function(x) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR_val <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR_val
  upper_bound <- Q3 + 1.5 * IQR_val
  x[ x < lower_bound | x > upper_bound ] <- NA
  return(x)
}

bank_data_no_outliers <- as.data.frame(lapply(numeric_vars,
replace_outliers_with_na))

num_data_long_no_outliers <- do.call(rbind,
  lapply(names(bank_data_no_outliers), function(var) {
    data.frame(
      Variable = var,
      Valor = bank_data_no_outliers[[var]]
    )
  })
)

# Eliminar outliers
num_data_long_no_outliers <-
num_data_long_no_outliers[!is.na(num_data_long_no_outliers$Valor), ]

ggplot(num_data_long_no_outliers, aes(x = Valor, fill = Variable)) +
```

```
  geom_histogram(aes(y = after_stat(density)), bins = 30, alpha = 0.5, color
= "black") +
  geom_density(alpha = 0.3, color = "blue", linewidth = 1) +
  facet_wrap(~Variable, scales = "free", ncol = 3) +
  theme_minimal() +
  labs(
    title = "Distribución de Variables Numéricas (sin outliers)",
    x = "Valor",
    y = "Densidad"
  ) +
  theme(
    legend.position = "none",
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold")
  )
```

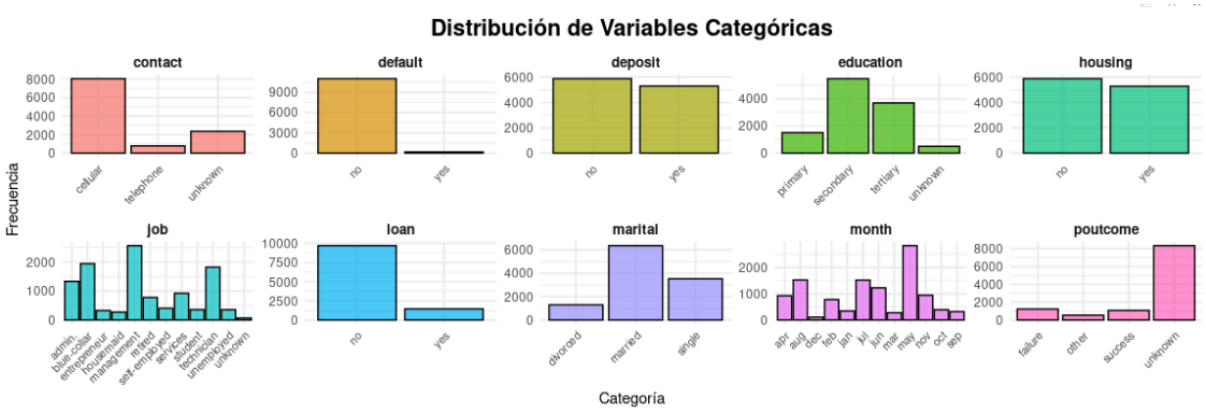


A pesar de quitar outliers, pdays(dias desde el ultimo contacto) tiene un valor de -1 para muchos casos, igual con previous(cantidad de veces que se le contacto al cliente), que tiene 0

Ahora graficamos las columnas de datos categoricos

```
None
cat_vars <- bank_data[, sapply(bank_data, Negate(is.numeric))]

cat_data_long <- data.frame(
  Variable = rep(names(cat_vars), each = nrow(cat_vars)),
  Valor = unlist(cat_vars)
)
```

```
ggplot(cat_data_long, aes(x = Valor, fill = Variable)) +
  geom_bar(color = "black", alpha = 0.7) +
  facet_wrap(~Variable, scales = "free", ncol = 5) +
  theme_minimal() +
  labs(
    title = "Distribución de Variables Categóricas",
    x = "Categoría",
    y = "Frecuencia"
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    strip.text = element_text(size = 10, face = "bold")
  )
```



Podemos notar que:
La mayoría de los clientes trabajan como administrativos, en el sector servicios o en la construcción.
La mayoría de los clientes están casados.
Se comunican principalmente con el personal de marketing por teléfono celular.
La mayoría tiene estudios secundarios o universitarios.
La gran mayoría de los clientes no tiene deudas pendientes ni préstamos.

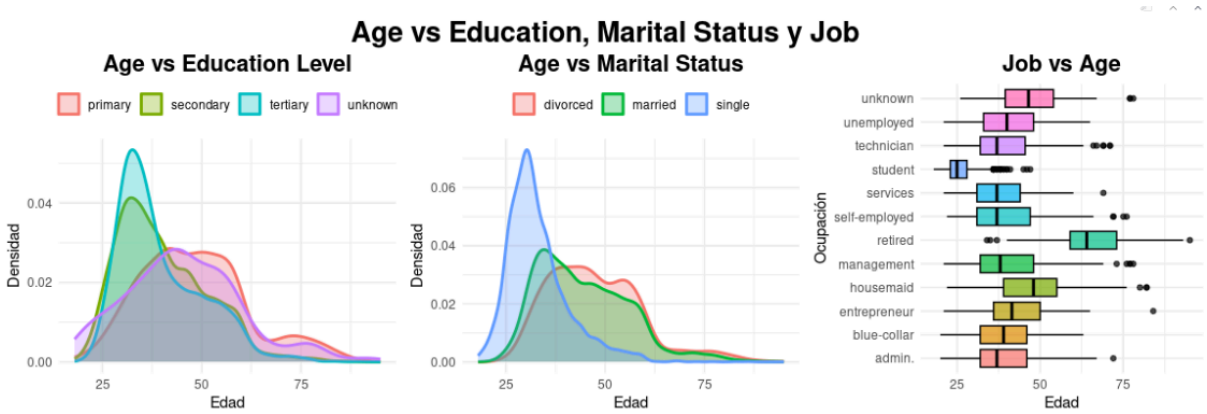
Generamos un grafico que nos permita observar cómo varía la edad según la educación, el estado civil y la ocupación dentro del dataset, el cual tendra:

- Dos gráficos de densidad muestran cómo se distribuye la edad según el nivel educativo y el estado civil.
- Un diagrama de cajas (boxplot) resume la edad por tipo de ocupación.

None

```
p1 <- ggplot(bank_data, aes(x = age, color = education, fill = education)) +
  geom_density(alpha = 0.3, linewidth = 1) +
  theme_minimal() +
  labs(
    title = "Age vs Education Level",
    x = "Edad",
    y = "Densidad"
```

```
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    legend.title = element_blank(),
    legend.position = "top",
    axis.text = element_text(size = 9),
    strip.text = element_text(size = 10, face = "bold")
  )

# --- Age vs Marital Status ---
p2 <- ggplot(bank_data, aes(x = age, color = marital, fill = marital)) +
  geom_density(alpha = 0.3, linewidth = 1) +
  theme_minimal() +
  labs(
    title = "Age vs Marital Status",
    x = "Edad",
    y = "Densidad"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    legend.title = element_blank(),
    legend.position = "top",
    axis.text = element_text(size = 9),
    strip.text = element_text(size = 10, face = "bold")
  )

# --- Job vs Age ---
p3 <- ggplot(bank_data, aes(y = job, x = age, fill = job)) +
  geom_boxplot(alpha = 0.7, color = "black") +
  theme_minimal() +
  labs(
    title = "Job vs Age",
    x = "Edad",
    y = "Ocupación"
  ) +
  theme(
    legend.position = "none",
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    axis.text = element_text(size = 9),
    strip.text = element_text(size = 10, face = "bold")
  )

# --- Combinar los gráficos ---
grid.arrange(
  p1, p2, p3,
  ncol = 3,
  top = textGrob(
    "Age vs Education, Marital Status y Job",
    gp = gpar(fontsize = 20, fontface = "bold")
  )
)
```



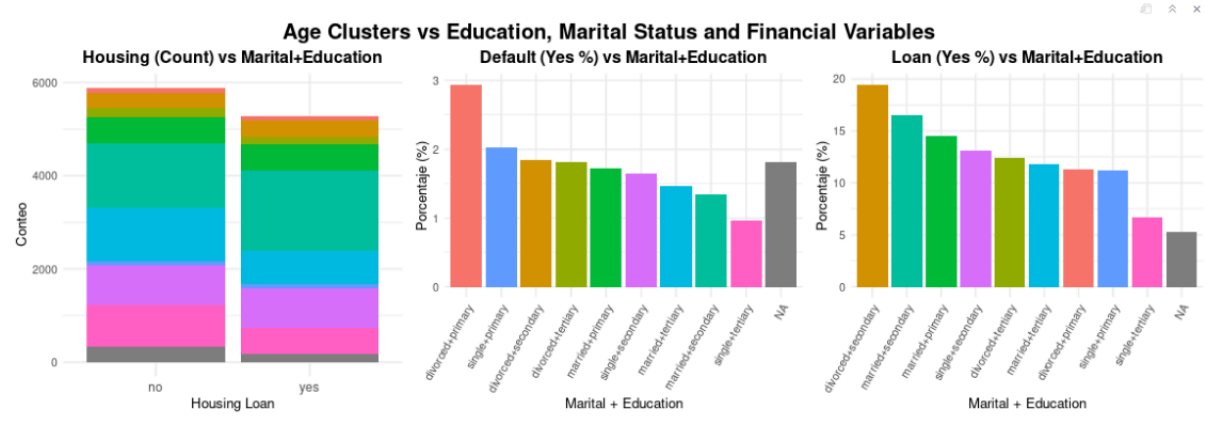
De aca podemos notar que:
Los clientes más jóvenes suelen tener estudios secundarios o superiores, mientras que los mayores generalmente tienen estudios primarios. Esto significa que, en promedio, los clientes más jóvenes tienen un nivel educativo superior al de los mayores.
Como era de esperar, los clientes solteros suelen ser los más jóvenes. La distribución de clientes casados y divorciados no difiere mucho, aunque los divorciados tienden a ser un poco mayores.
Por último, los estudiantes suelen ser los clientes más jóvenes, y los jubilados, los mayores, esto es natural.

La gran mayoría de los clientes no tiene deudas pendientes ni préstamos.

None

```
# --- Age vs Housing ---
p4 <- ggplot(bank_data, aes(x = age, color = housing, fill = housing)) +
  geom_density(alpha = 0.3, linewidth = 1) +
  theme_minimal() +
  labs(
    title = "Age vs Housing Loan",
    x = "Edad",
    y = "Densidad"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    legend.title = element_blank(),
    legend.position = "top",
    axis.text = element_text(size = 9),
    strip.text = element_text(size = 10, face = "bold")
  )

# --- Age vs Default ---
p5 <- ggplot(bank_data, aes(x = age, color = default, fill = default)) +
  geom_density(alpha = 0.3, linewidth = 1) +
  theme_minimal() +
  labs(
    title = "Age vs Credit Default",
    x = "Edad",
    y = "Densidad"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
```

```
    legend.title = element_blank(),
    legend.position = "top",
    axis.text = element_text(size = 9),
    strip.text = element_text(size = 10, face = "bold")
  )

# --- Age vs Personal Loan ---
p6 <- ggplot(bank_data, aes(x = age, color = loan, fill = loan)) +
  geom_density(alpha = 0.3, linewidth = 1) +
  theme_minimal() +
  labs(
    title = "Age vs Personal Loan",
    x = "Edad",
    y = "Densidad"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    legend.title = element_blank(),
    legend.position = "top",
    axis.text = element_text(size = 9),
    strip.text = element_text(size = 10, face = "bold")
  )

# --- Combinar los gráficos ---
grid.arrange(
  p4, p5, p6,
  ncol = 3,
  top = textGrob(
    "Age vs Housing, Default y Loan",
    gp = gpar(fontsize = 20, fontface = "bold")
  )
)
```



La proporción de clientes con y sin préstamo hipotecario varía según el grupo de edad: las personas mayores de 65 años generalmente no tienen préstamo, mientras que los clientes de otras categorías (por ejemplo, los menores de 25 años) sí lo tienen.
El porcentaje de clientes con créditos en mora es relativamente bajo. Este porcentaje de morosidad presenta una gran variabilidad entre los grupos de edad: los clientes entre 35 y 45 años tienen el porcentaje de morosidad más alto, y los mayores de 65 años, el más bajo.

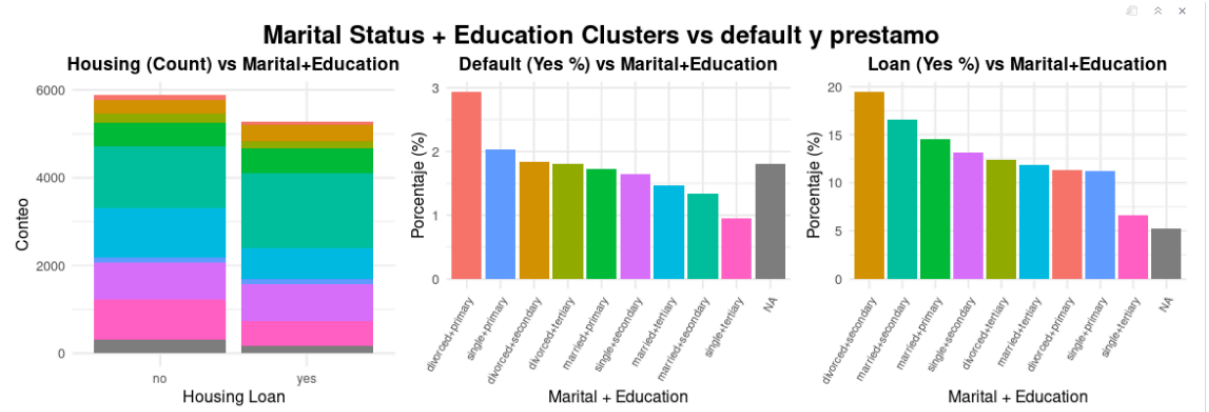
El porcentaje de clientes con préstamo hipotecario varía considerablemente según el grupo de edad. Casi el 16% de los clientes de entre 45 y 55 años tienen préstamo hipotecario, mientras que menos del 1% de los mayores de 65 años lo tienen.

None

```
bank_data <- bank_data %>%
  mutate(
    marital_edu = case_when(
      marital == "single" & education == "primary" ~ "single+primary",
      marital == "married" & education == "primary" ~ "married+primary",
      marital == "divorced" & education == "primary" ~ "divorced+primary",
      marital == "single" & education == "secondary" ~ "single+secondary",
      marital == "married" & education == "secondary" ~ "married+secondary",
      marital == "divorced" & education == "secondary" ~
"divorced+secondary",
      marital == "single" & education == "tertiary" ~ "single+tertiary",
      marital == "married" & education == "tertiary" ~ "married+tertiary",
      marital == "divorced" & education == "tertiary" ~ "divorced+tertiary",
      TRUE ~ NA_character_
    )
  )

# --- Agrupar por marital_edu y calcular mediana de balance ---
marital_edu_groups <- bank_data %>%
  group_by(marital_edu) %>%
  summarise(balance_median = median(balance, na.rm = TRUE)) %>%
  arrange(desc(balance_median))

# --- Calcular porcentaje de default == "yes" ---
default_yes <- bank_data %>%
  filter(default == "yes") %>%
  count(marital_edu, name = "count_yes")

default_no <- bank_data %>%
  filter(default == "no") %>%
  count(marital_edu, name = "count_no")

default_yes_perc <- default_yes %>%
  full_join(default_no, by = "marital_edu") %>%
  mutate(
    count_yes = ifelse(is.na(count_yes), 0, count_yes),
    count_no = ifelse(is.na(count_no), 0, count_no),
    `default percentage` = (count_yes / (count_yes + count_no)) * 100
  ) %>%
  arrange(desc(`default percentage`))

# --- Calcular porcentaje de loan == "yes" ---
loan_yes <- bank_data %>%
  filter(loan == "yes") %>%
  count(marital_edu, name = "count_yes")

loan_no <- bank_data %>%
  filter(loan == "no") %>%
  count(marital_edu, name = "count_no")
```

```r
loan_yes_perc <- loan_yes %>%
  full_join(loan_no, by = "marital_edu") %>%
  mutate(
    count_yes = ifelse(is.na(count_yes), 0, count_yes),
    count_no = ifelse(is.na(count_no), 0, count_no),
    `loan percentage` = (count_yes / (count_yes + count_no)) * 100
  ) %>%
  arrange(desc(`loan percentage`))

# --- Housing (Count) vs Marital+Education ---
p1 <- ggplot(bank_data, aes(x = housing, fill = marital_edu)) +
  geom_bar(position = "stack", color = "black", alpha = 0.8) +
  theme_minimal(base_size = 12) +
  labs(
    title = "Housing (Count) vs Marital+Education",
    x = "Housing Loan",
    y = "Conteo"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 13),
    legend.position = "none",
    axis.text.x = element_text(size = 9)
  )

# --- Default (Yes %) vs Marital+Education ---
p2 <- ggplot(default_yes_perc, aes(x = reorder(marital_edu, -`default
percentage`),
                                   y = `default percentage`, fill =
marital_edu)) +
  geom_col(color = "black", alpha = 0.8) +
  theme_minimal(base_size = 12) +
  labs(
    title = "Default (Yes %) vs Marital+Education",
    x = "Marital + Education",
    y = "Porcentaje (%)"
  ) +
  theme(
    axis.text.x = element_text(angle = 60, hjust = 1, size = 8),
    plot.title = element_text(hjust = 0.5, face = "bold", size = 13),
    legend.position = "none"
  )

# --- Loan (Yes %) vs Marital+Education ---
p3 <- ggplot(loan_yes_perc, aes(x = reorder(marital_edu, -`loan
percentage`),
                                y = `loan percentage`, fill = marital_edu))
+
  geom_col(color = "black", alpha = 0.8) +
  theme_minimal(base_size = 12) +
  labs(
    title = "Loan (Yes %) vs Marital+Education",
    x = "Marital + Education",
    y = "Porcentaje (%)"
  ) +
  theme(
```

```
    axis.text.x = element_text(angle = 60, hjust = 1, size = 8),
    plot.title = element_text(hjust = 0.5, face = "bold", size = 13),
    legend.position = "none"
  )

# --- Combinar los gráficos ---
grid.arrange(
  p1, p2, p3,
  ncol = 3,
  top = textGrob(
    "Marital Status + Education Clusters vs default y prestamo",
    gp = gpar(fontsize = 18, fontface = "bold")
  )
)
```



Algunas categorías de estado civil y nivel educativo tienen menos probabilidades de tener un préstamo hipotecario: casados con estudios superiores y solteros con estudios superiores. Los clientes casados con estudios secundarios tienen más probabilidades de tener un préstamo hipotecario.
El porcentaje de clientes con créditos en mora es relativamente bajo (menos del 3%). Este porcentaje de morosidad presenta una gran variabilidad dentro de los grupos de estado civil y nivel educativo: los divorciados con estudios primarios tienen el porcentaje de morosidad más alto, mientras que los solteros con estudios superiores tienen el más bajo.
El porcentaje de clientes con un préstamo varía considerablemente según su grupo de estado civil y nivel educativo. Casi el 20% de los divorciados con estudios secundarios tienen un préstamo, mientras que menos del 7% de los solteros con estudios superiores lo tienen.

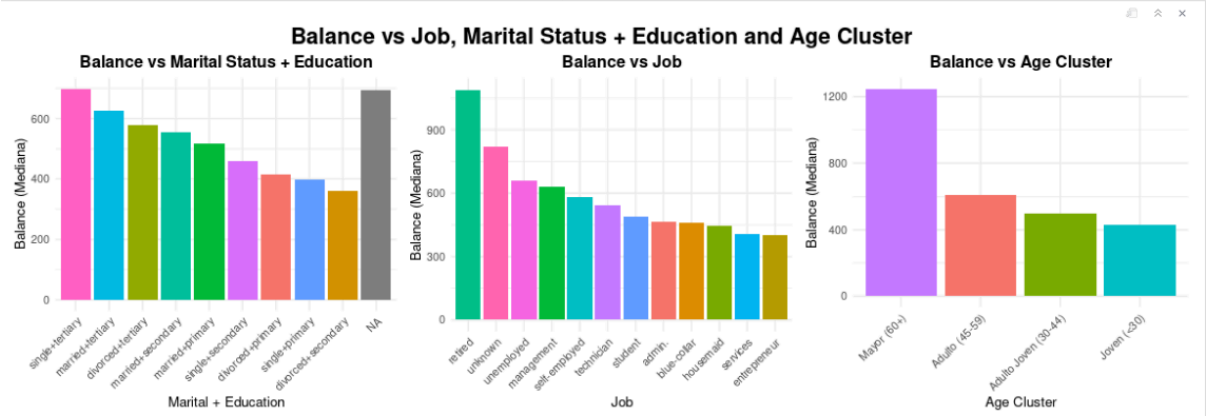
El saldo de los clientes depende de su edad, estado civil, nivel educativo y también de su trabajo.

None

```
marital_edu_groups <- bank_data %>%
  group_by(marital_edu) %>%
  summarise(balance = median(balance, na.rm = TRUE)) %>%
  arrange(desc(balance))
```

```r
# --- Agrupar por job ---
job_groups <- bank_data %>%
  group_by(job) %>%
  summarise(balance = median(balance, na.rm = TRUE)) %>%
  arrange(desc(balance))

# --- Agrupar por age_cluster ---
age_balance_groups <- bank_data %>%
  group_by(age_cluster) %>%
  summarise(balance = median(balance, na.rm = TRUE)) %>%
  arrange(desc(balance))

# --- 1. Balance vs Marital Status + Education ---
p1 <- ggplot(marital_edu_groups, aes(x = reorder(marital_edu, -balance), y =
balance, fill = marital_edu)) +
  geom_col() +
  theme_minimal(base_size = 10) +
  labs(
    title = "Balance vs Marital Status + Education",
    x = "Marital + Education",
    y = "Balance (Mediana)"
  ) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "none"
  )

# --- 2. Balance vs Job ---
p2 <- ggplot(job_groups, aes(x = reorder(job, -balance), y = balance, fill =
job)) +
  geom_col() +
  theme_minimal(base_size = 10) +
  labs(
    title = "Balance vs Job",
    x = "Job",
    y = "Balance (Mediana)"
  ) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "none"
  )

# --- 3. Balance vs Age Cluster ---
p3 <- ggplot(age_balance_groups, aes(x = reorder(age_cluster, -balance), y =
balance, fill = age_cluster)) +
  geom_col() +
  theme_minimal(base_size = 10) +
  labs(
    title = "Balance vs Age Cluster",
    x = "Age Cluster",
    y = "Balance (Mediana)"
  ) +
```

```
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "none"
  )

# --- Mostrar los tres gráficos ---
grid.arrange(
  p1, p2, p3,
  ncol = 3,
  top = textGrob(
    "Balance vs Job, Marital Status + Education and Age Cluster",
    gp = gpar(fontsize = 16, fontface = "bold")
  )
)
```



Las personas solteras con estudios superiores tienen el saldo medio más alto; las personas divorciadas con estudios secundarios, el más bajo. Esto se refleja en los gráficos de impago y préstamos, donde las personas solteras con estudios superiores presentaron los porcentajes más bajos de impago y préstamos. Las personas divorciadas con menor nivel educativo presentaron los porcentajes más altos de impago y préstamos.

Las personas jubiladas tienen el saldo medio más alto. Resulta curioso que las personas desempleadas tengan el segundo saldo medio más alto.

En cuanto a grupos de edad, las personas mayores de 65 años (los jubilados) tienen el saldo medio más alto, seguidas por las personas entre 55 y 65 años, y así sucesivamente.

Conclusiones de la segmentación de clientes

Edad y educación:

- Edad promedio: 41 años.
- Jóvenes : educación secundaria o superior.
- Mayores : educación primaria.
- Solteros : grupo de menor edad.

Préstamo hipotecario:

- La mayoría no tiene préstamo hipotecario.
- Personas mayores de 65 años : rara vez tienen préstamo.
- Menores de 25 años : más propensos a tener préstamo.
- Casados o solteros con educación superior : menor probabilidad de tener préstamo.
- Casados con educación secundaria : mayor probabilidad de tenerlo.

Crédito en mora:

- Menos del 3 % de los clientes presenta mora.
- Mayor incidencia: 35–45 años.
- Menor incidencia: mayores de 65 años.
- Divorciados con educación primaria : mayor tasa de impago.
- Solteros con educación superior : menor tasa de impago.

Préstamos personales:

- Alta variabilidad por edad, estado civil y educación.
- 45–55 años: ~16 % con préstamo.
- >65 años: <1 % con préstamo.
- Divorciados con secundaria: ~20 % con préstamo.
- Solteros con superior: <7 % con préstamo.

Saldo:

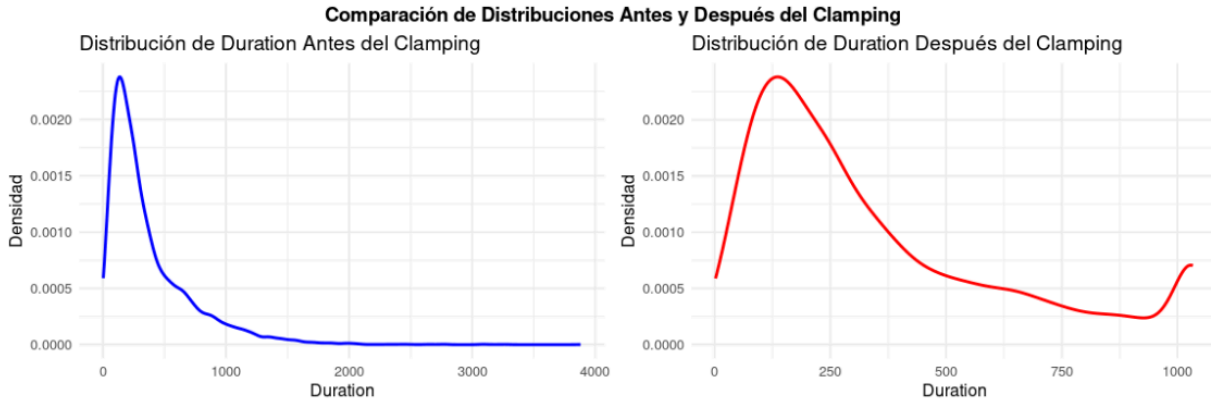
- Saldo promedio: 1529 $.
- Solteros con educación superior : saldo mediano más alto.
- Divorciados con educación secundaria : saldo mediano más bajo.
- Por edad: jubilados tienen el saldo más alto; 55–65 años el segundo.
- La gran mayoría de los clientes no tiene deudas pendientes ni préstamos.

CLAMPING DEL DATASET PARA ELIMINAR OUTLIERS

None

```
outlier_imputer <- function(data, features) {
  data_out <- data

  for (col in features) {
    Q1 <- quantile(data_out[[col]], 0.25, na.rm = TRUE)
    Q3 <- quantile(data_out[[col]], 0.75, na.rm = TRUE)
    IQR <- Q3 - Q1
    lowerL <- Q1 - 1.5 * IQR
    higherL <- Q3 + 1.5 * IQR

    # Reemplazar outliers por límites
    data_out[[col]] <- pmin(pmax(data_out[[col]], lowerL), higherL)
  }

  return(data_out)
}

# --- Columnas a procesar ---
features <- c("age", "balance", "day", "duration", "campaign", "pdays",
"previous")
```

```
# --- Aplicar función ---
capped_data <- outlier_imputer(bank_data, features)

# --- Graficar antes y después para "duration" ---
p1 <- ggplot(bank_data, aes(x = duration)) +
  geom_density(color = "blue", size = 1) +
  labs(
    title = "Distribución de Duration Antes del Clamping",
    x = "Duration",
    y = "Densidad"
  ) +
  theme_minimal(base_size = 12)

p2 <- ggplot(capped_data, aes(x = duration)) +
  geom_density(color = "red", size = 1) +
  labs(
    title = "Distribución de Duration Después del Clamping",
    x = "Duration",
    y = "Densidad"
  ) +
  theme_minimal(base_size = 12)

grid.arrange(p1, p2, ncol = 2,
             top = textGrob(
               "Comparación de Distribuciones Antes y Después del Clamping",
               gp = gpar(fontsize = 14, fontface = "bold")
             ))
```



La gran mayoría de los clientes no tiene deudas pendientes ni préstamos.

```
None
library(randomForest)
library(dplyr)

# --- Copiar los datos ---
data_feature <- capped_data

# --- Label encoding para variables categóricas ---
```

```r
for (col in names(data_feature)) {
  if (is.character(data_feature[[col]]) || is.factor(data_feature[[col]])) {
    data_feature[[col]] <- as.numeric(as.factor(data_feature[[col]]))
  }
}

# --- Eliminar filas con valores NA ---
data_feature <- na.omit(data_feature)

# --- Dividir variables predictoras y target ---
X <- data_feature[, setdiff(names(data_feature), "deposit")]
y <- as.factor(data_feature$deposit)

# --- Modelo Random Forest ---
set.seed(1)
random_forest <- randomForest(
  x = X,
  y = y,
  ntree = 500,
  maxnodes = 100,
  importance = TRUE
)

# --- Importancia de las variables ---
importances_df <- data.frame(
  feature = rownames(random_forest$importance),
  importance = round(random_forest$importance[, 1], 3)
) %>%
  arrange(desc(importance))

# --- Mostrar resultados ---
print(importances_df)
```

Description: df [15 × 2]

	feature <chr>	importance <dbl>
duration	duration	0.132
poutcome	poutcome	0.041
month	month	0.039
pdays	pdays	0.034
previous	previous	0.029
contact	contact	0.023
housing	housing	0.014
age	age	0.011
day	day	0.007
balance	balance	0.004

1-10 of 15 rows

Description: df [15 × 2]

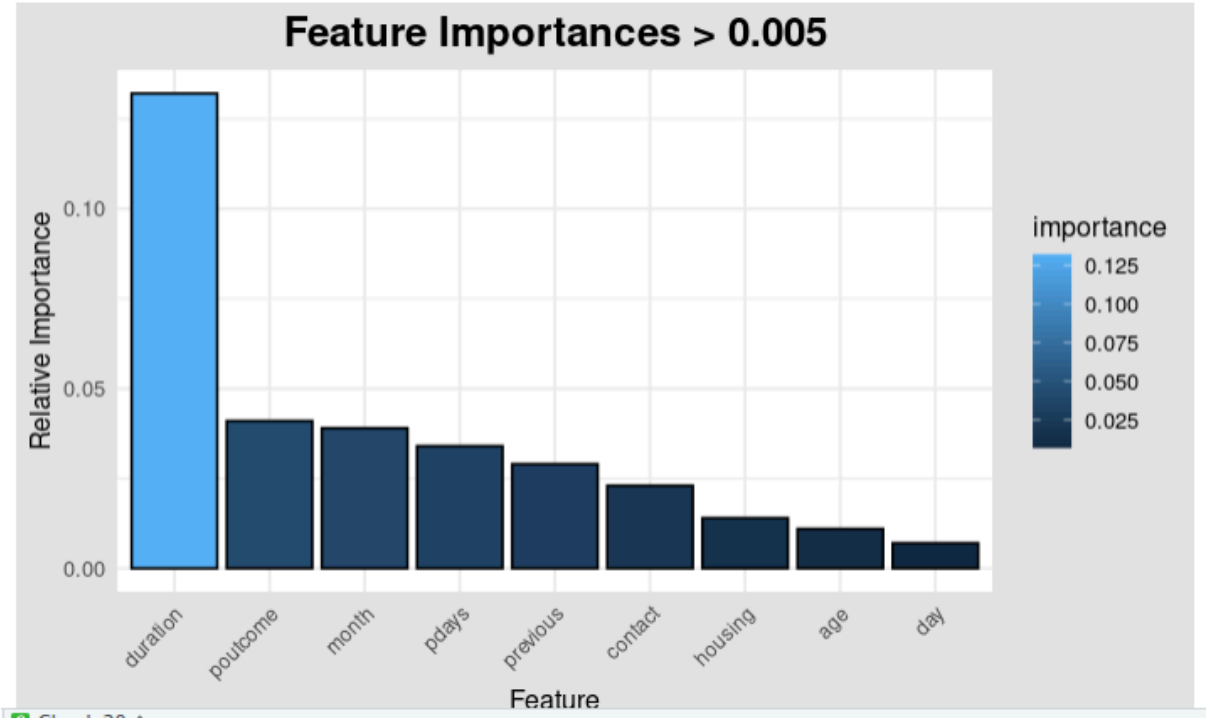
	feature <chr>	importance <dbl>
campaign	campaign	0.002
loan	loan	0.001
marital_edu	marital_edu	0.001
job	job	0.000
default	default	0.000

11-15 of 15 rows

El código convierte las variables categóricas en numéricas, elimina valores faltantes y entrena un modelo Random Forest con maxnodes = 100 para estimar la importancia de cada variable en predecir si un cliente hará un depósito. Luego ordena y muestra las variables más relevantes.

None

```
library(ggplot2)

# Filtrar solo las variables con importancia > 0.025
important_features <- importances %>%
  filter(importance > 0.005)

# Crear el gráfico
ggplot(important_features, aes(x = reorder(feature, -importance), y =
importance, fill = importance)) +
  geom_col(color = "black") +
  theme_minimal(base_size = 12) +
  labs(
    title = "Feature Importances > 0.005",
    x = "Feature",
    y = "Relative Importance"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 18),
    axis.text.x = element_text(angle = 45, hjust = 1, size = 9),
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "gray90", color = NA)
  )
```

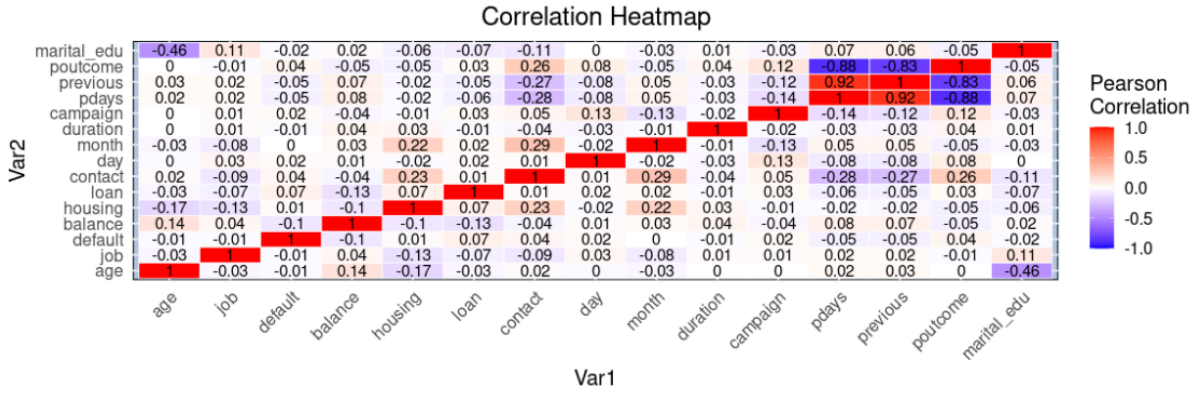


La «duración» se considera, con diferencia, el predictor más importante.
Luego tenemos características de segundo nivel (como «saldo», «mes», «día» y «antigüedad») y de tercer nivel (como «contacto» y «resultado»).
Las características que no se muestran en este gráfico (como «impago» o «préstamo») no se consideran tan importantes para predecir el resultado de la campaña de marketing.

CORRELACIÓN ENTRE VARIABLES

None

```
# Calcular matriz de correlación
corr_matrix <- cor(X, use = "complete.obs", method = "pearson")

# Convertir a formato largo para ggplot
corr_long <- melt(corr_matrix)

# Heatmap
ggplot(corr_long, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  geom_text(aes(label = round(value, 2)), color = "black", size = 4) +
  theme_minimal(base_size = 15) +
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   hjust = 1)) +
  ggtitle("Correlation Heatmap") +
  theme(plot.title = element_text(hjust = 0.5),
        panel.background = element_rect(fill = "lightsteelblue"))
```



Existen algunas variables fuertemente correlacionadas entre sí: 'pdays' vs 'previous', 'pdays' vs 'poutcome' y 'previous' vs 'poutcome'. Al menos dos de ellas ('pdays' vs 'previous'), con un coeficiente de Pearson > 0.9, deben ser tratadas.

Por este motivo, descarto 'pdays' y 'previous'. 'poutcome' tiene una mayor importancia como variable predictora.

```r
capped_data$pdays <- NULL
capped_data$previous <- NULL

data_feature$pdays <- NULL
data_feature$previous <- NULL

colnames(capped_data)
colnames(data_feature)
```

```
 [1] "age"       "job"       "default"   "balance"   "housing"   "loan"      "contact"   "day"       "month"     "duration"  "campaign"
[12] "poutcome"  "deposit"   "marital_edu"
 [1] "age"       "job"       "default"   "balance"   "housing"   "loan"      "contact"   "day"       "month"     "duration"  "campaign"
[12] "poutcome"  "deposit"   "marital_edu"
```

Creacion de train y test:

- X y y se separan.
- División entrenamiento/prueba con 70%-30% (train_test_split).
- Codificación de la variable objetivo (deposit) a 0/1.
- Identificación de columnas categóricas y separación en binarias/multi-clase.
- Label encoding para columnas binarias (yes → 1, no → 0).

None

```r
# --- Separar X e y ---
X <- capped_data[, names(capped_data) != "deposit"]
y <- capped_data$deposit

# --- Dividir en entrenamiento y prueba (70%-30%) ---
set.seed(42)
library(caTools)
split <- sample.split(y, SplitRatio = 0.7)

X_train <- X[split == TRUE, ]
X_test  <- X[split == FALSE, ]

y_train <- y[split == TRUE]
y_test  <- y[split == FALSE]
```

```
dim(X_train); dim(X_test)
length(y_train); length(y_test)

# --- Label encoding para la variable objetivo ---
y_train <- ifelse(y_train == "yes", 1, 0)
y_test  <- ifelse(y_test == "yes", 1, 0)

# --- Identificar variables categóricas ---
cat_cols <- names(X_train)[sapply(X_train, is.character)]

# Separar binarias y multiclase
bin_cols   <- c()
multi_cols <- c()

for(col in cat_cols){
  n_unique <- length(unique(X_train[[col]]))
  cat("feature =", col, "; cardinality =", n_unique, "\n")
  if(n_unique <= 2){
    bin_cols <- c(bin_cols, col)
  } else {
    multi_cols <- c(multi_cols, col)
  }
}

cat("\nBinary columns:", bin_cols, "\n")
cat("Multi-class columns:", multi_cols, "\n")

# --- Label encoding para las columnas binarias ---
for(col in bin_cols){
  X_train[[col]] <- ifelse(X_train[[col]] == "yes", 1, 0)
  X_test[[col]]  <- ifelse(X_test[[col]] == "yes", 1, 0)
}
```

Ahora agrupamos los datos irrelevantes en categoria otros
Convierte las clases raras (<5%) en 'other'.
Muestra las proporciones de cada categoría.
Detecta columnas con NA y reemplaza los NA de marital_edu por 'other'.

```
None
# --- Función para reemplazar clases raras (< 0.05) por 'other' ---
remove_005 <- function(train, test, column) {
  prop_table <- prop.table(table(train[[column]]))

  # Detectar clases con frecuencia < 0.05
  rare_classes <- names(prop_table[prop_table < 0.05])

  # Reemplazar en train y test
  train[[column]][train[[column]] %in% rare_classes] <- "other"
  test[[column]][test[[column]] %in% rare_classes]   <- "other"
```

```
  return(list(train = train, test = test))
}

# --- Aplicar la función a todas las columnas multiclase ---
for(col in multi_cols){
  res <- remove_005(X_train, X_test, col)
  X_train <- res$train
  X_test  <- res$test
}

# --- Revisar proporciones de cada categoría ---
for(col in multi_cols){
  prop <- prop.table(table(X_train[[col]]))
  print(prop)
  cat("\n")
}

# --- Revisar columnas con NA ---
list_nulls <- names(X_train)[sapply(X_train, function(x) any(is.na(x)))]
list_nulls

# --- Imputar NA con 'other' en 'marital_edu' ---
if("marital_edu" %in% list_nulls){
  X_train$marital_edu[is.na(X_train$marital_edu)] <- "other"
  X_test$marital_edu[is.na(X_test$marital_edu)]   <- "other"
}

# --- Confirmar que no hay NA ---
list_nulls <- names(X_train)[sapply(X_train, function(x) any(is.na(x)))]
list_nulls
```

model.matrix(~ . -1, ...) genera las columnas dummy para cada nivel de la variable categórica.
Se eliminan las columnas originales multiclase y se reemplazan por las columnas one-hot.
Se combinan con las columnas numéricas/binarizadas, listas para usar en Random Forest, XGBoost o cualquier otro modelo.

None

```
# --- Separar las columnas numéricas/binarizadas ---
num_train <- X_train[, !(names(X_train) %in% multi_cols)]
num_test  <- X_test[,  !(names(X_test)  %in% multi_cols)]

# --- One-hot encoding para las columnas multiclase ---
OHE_train <- as.data.frame(model.matrix(~ . -1, data = X_train[,
multi_cols]))
OHE_test  <- as.data.frame(model.matrix(~ . -1, data = X_test[,
multi_cols]))

# --- Combinar columnas numéricas/binarizadas con las one-hot encoded ---
OHE_X_train <- cbind(OHE_train, num_train)
```

```
OHE_X_test  <- cbind(OHE_test, num_test)

# --- Revisar dimensiones ---
dim(OHE_X_train)
dim(OHE_X_test)
```

La gran mayoría de los clientes no tiene deudas pendientes ni préstamos.

None

```
# --- Identificar columnas numéricas ---
num_features <- names(OHE_X_train)[sapply(OHE_X_train, is.numeric)]

# --- Escalado Min-Max usando caret ---
library(caret)

scaler <- preProcess(OHE_X_train[, num_features], method = c("range"))

OHE_X_train[, num_features] <- predict(scaler, OHE_X_train[, num_features])
OHE_X_test[, num_features]  <- predict(scaler, OHE_X_test[, num_features])

# --- Revisar primeras filas ---
head(OHE_X_train)
```

Description: df [6 × 33]

	jobadmin. <dbl>	jobblue-collar <dbl>	jobmanagement <dbl>	jobother <dbl>	jobretired <dbl>	jobservices <dbl>	jobtechnician <dbl>	contacttelephone <dbl>
3	0	0	0	0	0	0	1	0
5	1	0	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0
8	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	1	0
10	0	0	0	0	0	1	0	0

6 rows | 1-9 of 33 columns

Detecta las columnas numéricas.
Ajusta un Min-Max scaler sobre OHE_X_train.
Transforma tanto OHE_X_train como OHE_X_test con el mismo rango.
Ahora los valores de las columnas numéricas estarán en el rango [0, 1].

Estamos listos para el entrenamiento

None

```
library(MASS)         # LDA, QDA
library(class)        # KNN
library(rpart)        # CART
library(e1071)        # SVM
library(randomForest) # Random Forest
```

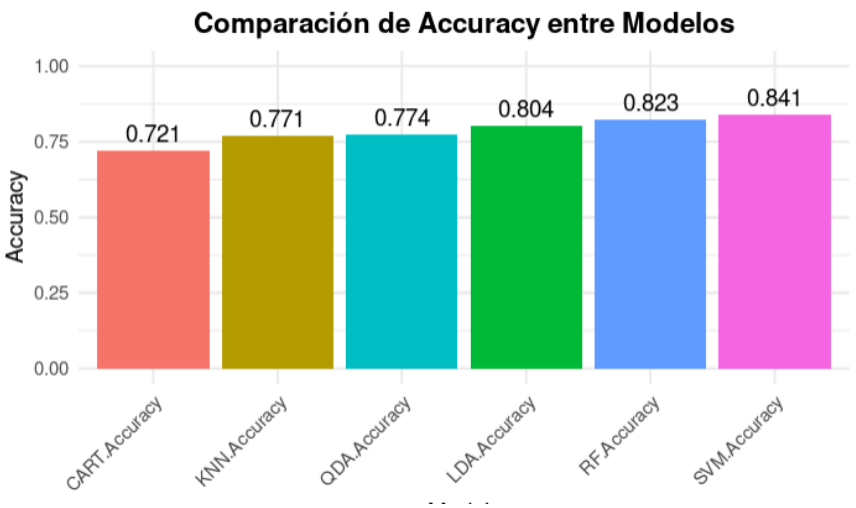
```
library(caret)        # Confusion Matrix

# --- 1. Preparar datos ---
# Asegurarse de que y sea factor
y_train_factor <- factor(y_train)
y_test_factor  <- factor(y_test)

# Reducir colinealidad usando PCA para QDA
pca <- prcomp(OHE_X_train, center = TRUE, scale. = TRUE)
# Elegir primeros componentes que expliquen ~95% de la varianza
cumvar <- cumsum(pca$sdev^2) / sum(pca$sdev^2)
num_pc <- min(which(cumvar >= 0.95))
X_train_pca <- pca$x[, 1:num_pc]
X_test_pca  <- predict(pca, newdata = OHE_X_test)[, 1:num_pc]

# --- 2. LDA ---
lda_model <- lda(X_train_pca, grouping = y_train_factor)
lda_pred  <- predict(lda_model, X_test_pca)$class
lda_cm    <- confusionMatrix(lda_pred, y_test_factor)
print("LDA Confusion Matrix:")
print(lda_cm)

# --- 3. QDA ---
qda_model <- qda(X_train_pca, grouping = y_train_factor)
qda_pred  <- predict(qda_model, X_test_pca)$class
qda_cm    <- confusionMatrix(qda_pred, y_test_factor)
print("QDA Confusion Matrix:")
print(qda_cm)

# --- 4. CART ---
cart_model <- rpart(y_train_factor ~ ., data = data.frame(X_train_pca,
y_train_factor))
cart_pred  <- predict(cart_model, data.frame(X_test_pca), type = "class")
cart_cm    <- confusionMatrix(cart_pred, y_test_factor)
print("CART Confusion Matrix:")
print(cart_cm)

# --- 5. KNN ---
k <- 5
knn_pred <- knn(train = X_train_pca, test = X_test_pca, cl = y_train_factor,
k = k)
knn_cm   <- confusionMatrix(knn_pred, y_test_factor)
print("KNN Confusion Matrix:")
print(knn_cm)

# --- 6. SVM ---
svm_model <- svm(X_train_pca, y_train_factor, kernel = "radial", probability
= TRUE)
svm_pred  <- predict(svm_model, X_test_pca)
svm_cm    <- confusionMatrix(svm_pred, y_test_factor)
print("SVM Confusion Matrix:")
print(svm_cm)

# --- 7. Random Forest ---
```

```
rf_model <- randomForest(X_train_pca, y_train_factor, ntree = 500,
importance = TRUE)
rf_pred  <- predict(rf_model, X_test_pca)
rf_cm    <- confusionMatrix(rf_pred, y_test_factor)
print("Random Forest Confusion Matrix:")
print(rf_cm)

# --- 8. Comparar Accuracy ---
accuracy <- c(
  LDA = lda_cm$overall['Accuracy'],
  QDA = qda_cm$overall['Accuracy'],
  CART = cart_cm$overall['Accuracy'],
  KNN = knn_cm$overall['Accuracy'],
  SVM = svm_cm$overall['Accuracy'],
  RF = rf_cm$overall['Accuracy']
)
print("Model Accuracies:")
print(accuracy)
```

Graficamos las precisiones:

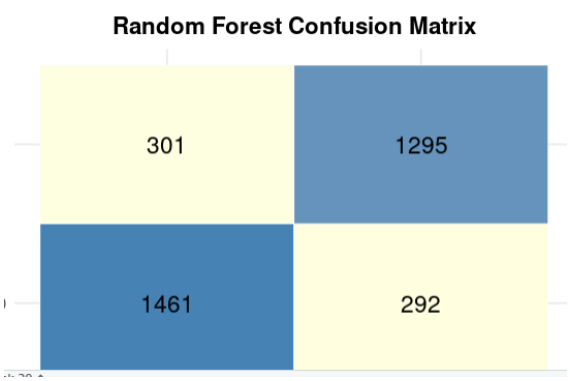
None

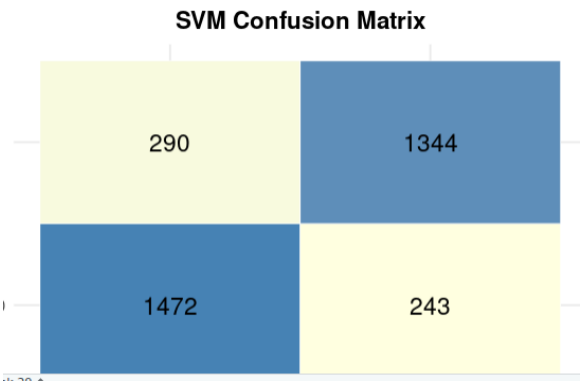
```
# Convertir a data frame
accuracy_df <- data.frame(
  Model = names(accuracy),
  Accuracy = as.numeric(accuracy)
)

# Gráfico de barras
ggplot(accuracy_df, aes(x = reorder(Model, Accuracy), y = Accuracy, fill =
Model)) +
  geom_col(show.legend = FALSE) +
  geom_text(aes(label = round(Accuracy, 3)), vjust = -0.5) +
  ylim(0,1) +
  labs(
    title = "Comparación de Accuracy entre Modelos",
    x = "Modelo",
    y = "Accuracy"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```



None

```
plot_confusion <- function(pred, true, title = "Confusion Matrix") {
  cm <- confusionMatrix(as.factor(pred), as.factor(true))
  cm_table <- as.data.frame(cm$table)
  colnames(cm_table) <- c("Prediction", "Reference", "Freq")

  ggplot(cm_table, aes(x = Reference, y = Prediction, fill = Freq)) +
    geom_tile(color = "white") +
    geom_text(aes(label = Freq), size = 6) +
    scale_fill_gradient(low = "lightyellow", high = "steelblue") +
    labs(title = title, x = "Actual", y = "Predicted") +
    theme_minimal(base_size = 14) +
    theme(plot.title = element_text(face = "bold", hjust = 0.5))
}

# --- Graficar para Random Forest ---
plot_confusion(rf_pred, y_test, "Random Forest Confusion Matrix")

# --- Graficar para SVM ---
plot_confusion(svm_pred, y_test, "SVM Confusion Matrix")
```





Predicciones:

El dataset esta separado de esta manera:
split <- sample.split(y, SplitRatio = 0.7)
X_train <- subset(X, split == TRUE) # conjunto de entrenamiento
X_test <- subset(X, split == FALSE) # conjunto de validación
y_train <- subset(y, split == TRUE)
y_test <- subset(y, split == FALSE)

El 30% se usa para validacion, realizaremos predicciones con nuestros modelos mas exitosos

None

```
# --- Random Forest ---
rf_pred <- predict(rf_model, OHE_X_test)
rf_prob <- predict(rf_model, OHE_X_test, type = "prob")

# --- SVM ---
svm_pred <- predict(svm_model, OHE_X_test)
svm_prob <- predict(svm_model, OHE_X_test, probability = TRUE)

# --- Comparar con valores reales ---
head(data.frame(
  Actual = y_test,
  RF_Pred = rf_pred,
  SVM_Pred = svm_pred
))
```

Description: df [6 × 3]

	Actual <fctr>	RF_Pred <fctr>	SVM_Pred <fctr>
1	1	1	1
2	1	1	1
4	1	0	0
7	1	1	1
12	1	1	1
13	1	1	1

6 rows