

Reto Bankaya

Walls Salcedo Carlos

2025-06-24

Primera parte

Objetivo

El objetivo principal de este proyecto es desarrollar un modelo de riesgo crediticio para Bankaya que permita diferenciar eficazmente entre clientes de “buen” y “mal” comportamiento crediticio. Al integrar información transaccional interna de los solicitantes con su historial crediticio externo, buscaremos generar un riesgo_score predictivo y, con base en él, proponer un sistema de tasas de interés dinámicas individualizadas. Esto facilitará la toma de decisiones de aprobación y pricing de préstamos para la compra de smartphones, optimizando la gestión de riesgos y promoviendo un crecimiento sostenible de la cartera de clientes.

Descripción de variables

Dataset Principal (main_dataset.parquet) Contiene información detallada de las solicitudes de préstamos de Bankaya, incluyendo datos del cliente y su interacción con la plataforma.

-customer_id: Identificador único del cliente.

-loan_id: Identificador único del préstamo.

-ACC_CREATION_DATETIME Fecha de creación de la cuenta del cliente.

-APPLICATION_DATETIME Fecha en la que se solicitó el préstamo.

LOAN_ORIGINATION_DATETIME: Fecha en que el préstamo fue aprobado o iniciado.

max_days_late: Máximo número de días que el cliente se atrasó en un pago.

target: Variable objetivo original (0: buen comportamiento, 1: mal comportamiento).

account_to_application_days: Días entre la creación de cuenta y la solicitud del préstamo.

n_sf_apps: Número de solicitudes previas en la plataforma “SF” (no siempre presente).

first_app_date: Fecha de la primera solicitud de crédito registrada.

last_app_date: Fecha de la última solicitud de crédito registrada.

n_bnpl_apps: Número de aplicaciones tipo “Buy Now Pay Later” hechas por el cliente.

n_bnpl_approved_apps: Número de esas aplicaciones que fueron aprobadas.

first_bnpl_app_date: Fecha de la primera solicitud BNPL.

last_bnpl_app_date: Fecha de la última solicitud BNPL.

n_inquiries_13m: Número de consultas de crédito en los últimos 3 meses.

n_inquiries_16m: Número de consultas de crédito en los últimos 6 meses.

Dataset de Reportes de Crédito (credit_reports.parquet) Este dataset contiene el historial crediticio externo de los clientes, donde cada fila representa un registro de crédito específico del cliente con diversas entidades financieras (ej. préstamos, tarjetas de crédito). Un mismo customer_id puede tener múltiples entradas en este dataset.

-customer_id: Identificador único del cliente (clave de unión con main_dataset).

-REPORT_DATE: Fecha de generación o actualización del reporte de crédito.

-LOAN_OPENING_DATE: Fecha de apertura del crédito externo.

-LOAN_CLOSING_DATE: Fecha de cierre o terminación del crédito externo.

-CREDIT_TYPE: Tipo de crédito (ej., tarjeta de crédito, préstamo personal, hipoteca).

-PAYMENT_FREQUENCY: Frecuencia de los pagos de este crédito (ej., mensual, semanal).

-MAX_CREDIT: Monto máximo de crédito aprobado para esta línea de crédito.

-CREDIT_LIMIT: Límite de crédito asignado para esta línea de crédito.

-PAYMENT_AMOUNT: Monto del pago más reciente registrado.

-CURRENT_BALANCE: Saldo actual pendiente de pago en esta línea de crédito.

-BALANCE_DUE: Monto total vencido o adeudado.

-BALANCE_DUE_WORST_DELAY: Monto máximo que estuvo vencido o adeudado en el peor momento de atraso.

-DELAYED_PAYMENTS: Número de pagos que el cliente ha atrasado en esta cuenta.

-WORST_DELAY: El peor número de días de atraso registrado para este crédito.

-WORST_DELAY_DATE: Fecha en que se registró el peor atraso.

-TOTAL_PAYMENTS: Número total de pagos realizados para esta cuenta.

-TOTAL_REPORTED_PAYMENTS: Número total de pagos reportados a las agencias de crédito.

-UPDATE_DATE: Fecha de la última actualización de este registro de crédito.

-LAST_PURCHASE_DATE: Fecha de la última compra o disposición de crédito.

-LAST_PAYMENT_DATE: Fecha del último pago registrado.

```
# Librerías necesarias
library(arrow)
library(rpart)      # Para Árboles de Decisión
library(rpart.plot) # Para visualizar Árboles de Decisión
library(dplyr)
library(tidyr)
library(ggplot2)
library(corrplot)
library(DT)
library(caret)
library(ROSE) # Para balancear clases
library(randomForest)
library(xgboost)
library(GGally)
library(e1071) # Para SVM
library(nnet) # Para Red Neuronal
library(MLmetrics)
library(knitr)
library(kableExtra)
library(scales)
library(lubridate) # Para manejo de fechas en df2
library(janitor)  # Para limpieza de nombres de columnas en df2
library(tidyverse) # Colección de paquetes, ya tienes algunos individuales pero lo incluyo por si acaso
```

Importamos la base de datos con su respectivo summary y visualización de las primeras filas

```
df <- read_parquet("C:/Users/DELL/Downloads/main_dataset.parquet")
print(head(df))
```

```
## # A tibble: 6 × 17
##   customer_id loan_id ACC_CREATION_DATETIME APPLICATION_DATETIME
##   <int>    <int> <dtm>                <dtm>
## 1      1223      1 2021-08-23 08:57:56 2022-04-26 02:00:00
## 2      5190      2 2022-04-26 04:57:25 2022-04-26 02:00:00
## 3      5194      3 2022-04-26 07:22:35 2022-04-26 02:00:00
## 4      3978      4 2022-03-09 05:26:55 2022-04-26 02:00:00
## 5      4535      5 2022-04-01 08:28:42 2022-04-26 02:00:00
## 6      3604      6 2022-02-21 05:55:32 2022-05-05 02:00:00
## # i 13 more variables: LOAN_ORIGINATION_DATETIME <dtm>, max_days_late <int>,
## #   target <int>, account_to_application_days <int>, n_sf_apps <dbl>,
## #   first_app_date <dtm>, last_app_date <dtm>, n_bnpl_apps <dbl>,
## #   n_bnpl_approved_apps <dbl>, first_bnpl_app_date <dtm>,
## #   last_bnpl_app_date <dtm>, n_inquiries_l3m <dbl>, n_inquiries_l6m <dbl>
```

```
print(summary(df))
```

```

## customer_id loan_id ACC_CREATION_DATETIME
## Min. : 1 Min. : 1 Min. :2020-10-14 13:22:10.00
## 1st Qu.: 3614 1st Qu.: 3614 1st Qu.:2022-02-21 12:46:22.25
## Median : 7228 Median : 7228 Median :2022-07-19 15:29:43.50
## Mean : 7228 Mean : 7228 Mean :2022-06-17 02:24:49.44
## 3rd Qu.:10841 3rd Qu.:10841 3rd Qu.:2022-11-13 01:37:39.25
## Max. :14454 Max. :14454 Max. :2023-05-19 13:55:04.00
##
## APPLICATION_DATETIME LOAN_ORIGINATION_DATETIME
## Min. :2022-04-26 02:00:00.0 Min. :2022-07-01 04:03:20.00
## 1st Qu.:2022-09-15 08:00:00.0 1st Qu.:2022-10-27 16:15:58.25
## Median :2022-12-20 02:00:00.0 Median :2023-01-11 04:05:49.50
## Mean :2022-11-27 21:42:40.9 Mean :2022-12-28 00:04:09.50
## 3rd Qu.:2023-02-04 02:00:00.0 3rd Qu.:2023-03-06 12:07:46.25
## Max. :2023-05-26 01:00:00.0 Max. :2023-05-29 06:18:28.00
##
## max_days_late target account_to_application_days n_sf_apps
## Min. :-7.00 Min. :0.0000 Min. : 0.0 Min. : 1.000
## 1st Qu.: 0.00 1st Qu.:0.0000 1st Qu.: 0.0 1st Qu.: 1.000
## Median : 2.00 Median :0.0000 Median :103.0 Median : 1.000
## Mean :14.23 Mean :0.1868 Mean :163.5 Mean : 1.654
## 3rd Qu.:20.00 3rd Qu.:0.0000 3rd Qu.:271.8 3rd Qu.: 2.000
## Max. :70.00 Max. :1.0000 Max. :901.0 Max. :42.000
## NA's :7648 NA's :7648
## first_app_date last_app_date
## Min. :2021-04-26 19:00:00.00 Min. :2021-04-24 19:00:00.00
## 1st Qu.:2022-02-26 18:00:00.00 1st Qu.:2022-02-24 18:00:00.00
## Median :2022-07-14 19:00:00.00 Median :2022-07-15 19:00:00.00
## Mean :2022-06-15 23:31:39.97 Mean :2022-06-15 20:42:11.52
## 3rd Qu.:2022-10-20 19:00:00.00 3rd Qu.:2022-10-21 19:00:00.00
## Max. :2023-05-11 18:00:00.00 Max. :2023-05-11 18:00:00.00
## NA's :7648 NA's :7648
## n_bnpl_apps n_bnpl_approved_apps first_bnpl_app_date
## Min. : 1.000 Min. : 0.000 Min. :2022-01-06 15:17:08.19
## 1st Qu.: 1.000 1st Qu.: 0.000 1st Qu.:2022-05-01 16:03:56.96
## Median : 1.000 Median : 0.000 Median :2022-08-18 08:36:14.26
## Mean : 1.222 Mean : 0.265 Mean :2022-08-13 04:30:46.84
## 3rd Qu.: 1.000 3rd Qu.: 0.000 3rd Qu.:2022-11-06 13:24:55.19
## Max. :18.000 Max. :15.000 Max. :2023-05-20 11:15:47.00
## NA's :5715 NA's :5715 NA's :5715
## last_bnpl_app_date n_inquiries_l3m n_inquiries_l6m
## Min. :2022-01-06 15:17:08.19 Min. : 0.00 Min. : 0.00
## 1st Qu.:2022-04-20 00:33:33.58 1st Qu.: 0.00 1st Qu.: 0.00
## Median :2022-07-28 12:37:41.68 Median : 0.00 Median : 8.00
## Mean :2022-08-03 04:11:07.58 Mean :10.35 Mean :17.11
## 3rd Qu.:2022-11-05 19:50:47.64 3rd Qu.:14.00 3rd Qu.:26.00

```

```
## Max.      :2023-05-17 09:20:48.00 Max.      :170.00 Max.      :213.00
## NA's      :5715 NA's      :5371 NA's      :5371
```

En el resumen podemos observar que la variable `max_days_late` tiene valores negativos, dado que tener un valor negativo representa que se pagó antes de llegar al retraso se modificarán por 0 (No hay existencia de retraso).

Revisamos la cantidad de valores faltantes por columna

```
colSums(is.na(df))
```

```
##          customer_id          loan_id
##              0              0
##  ACC_CREATION_DATETIME  APPLICATION_DATETIME
##              0              0
##  LOAN_ORIGINATION_DATETIME  max_days_late
##              0              0
##          target account_to_application_days
##              0              0
##          n_sf_apps          first_app_date
##          7648          7648
##          last_app_date          n_bnpl_apps
##          7648          5715
##          n_bnpl_approved_apps  first_bnpl_app_date
##          5715          5715
##          last_bnpl_app_date          n_inquiries_l3m
##          5715          5371
##          n_inquiries_l6m
##          5371
```

Tenemos que para:

- `n_sf_apps`, `first_app_date`, `last_app_date`

Valores faltantes: 7,648 representa más del 50% del total.

Interpretación: Estos campos se relacionan con las solicitudes previas en la plataforma "SF". Los valores faltantes indican que esas personas nunca han realizado una solicitud de crédito previa en esa plataforma. Por tanto:

`n_sf_apps = NA` significa cero solicitudes previas.

`first_app_date` y `last_app_date` son NA porque no hay fechas que registrar.

Se reemplazarán los NA por 0 en `n_sf_apps`

- `n_bnpl_apps`, `n_bnpl_approved_apps`, `first_bnpl_app_date`, `last_bnpl_app_date`

Valores faltantes: 5,715 casos.

Interpretación: Las personas con valores faltantes no han solicitado ni han sido aprobadas en ningún esquema BNPL anteriormente. Se reemplazarán los NA de `n_bnpl_apps` y `n_bnpl_approved_apps` con 0.

- `n_inquiries_l3m`, `n_inquiries_l6m` Valores faltantes: 5,371 casos.

Interpretación: Estas variables registran cuántas veces se ha consultado el historial crediticio del cliente en los últimos 3 y 6 meses. El valor faltante probablemente indica que no existen consultas registradas para ese cliente en ese período, lo que puede ser porque:

Es un cliente nuevo (sin historial). Nunca ha solicitado ningún crédito anteriormente. También se reemplazarán con 0 los valores faltantes

```
# Tratamiento de valores negativos y NA en df

# Valores negativos
df <- df %>%
  mutate(max_days_late = ifelse(max_days_late < 0, 0, max_days_late))

# Columnas donde NA significa 'cero actividad/información'
cols_na_0 <- c("n_sf_apps", "n_bnpl_apps", "n_bnpl_approved_apps",
               "n_inquiries_l3m", "n_inquiries_l6m")

df <- df %>%
  mutate(across(all_of(cols_na_0), ~replace_na(., 0)))

# Codificación
# Convertir enteros a numéricos y target a factor
df <- df %>%
  mutate(across(where(is.integer), as.numeric), # Convertir todas las columnas enteras a numéricas
         target = as.factor(target)) # Variable objetivo a factor

print(colSums(is.na(df))) # Verificamos que no queden NAs en las columnas relevantes
```

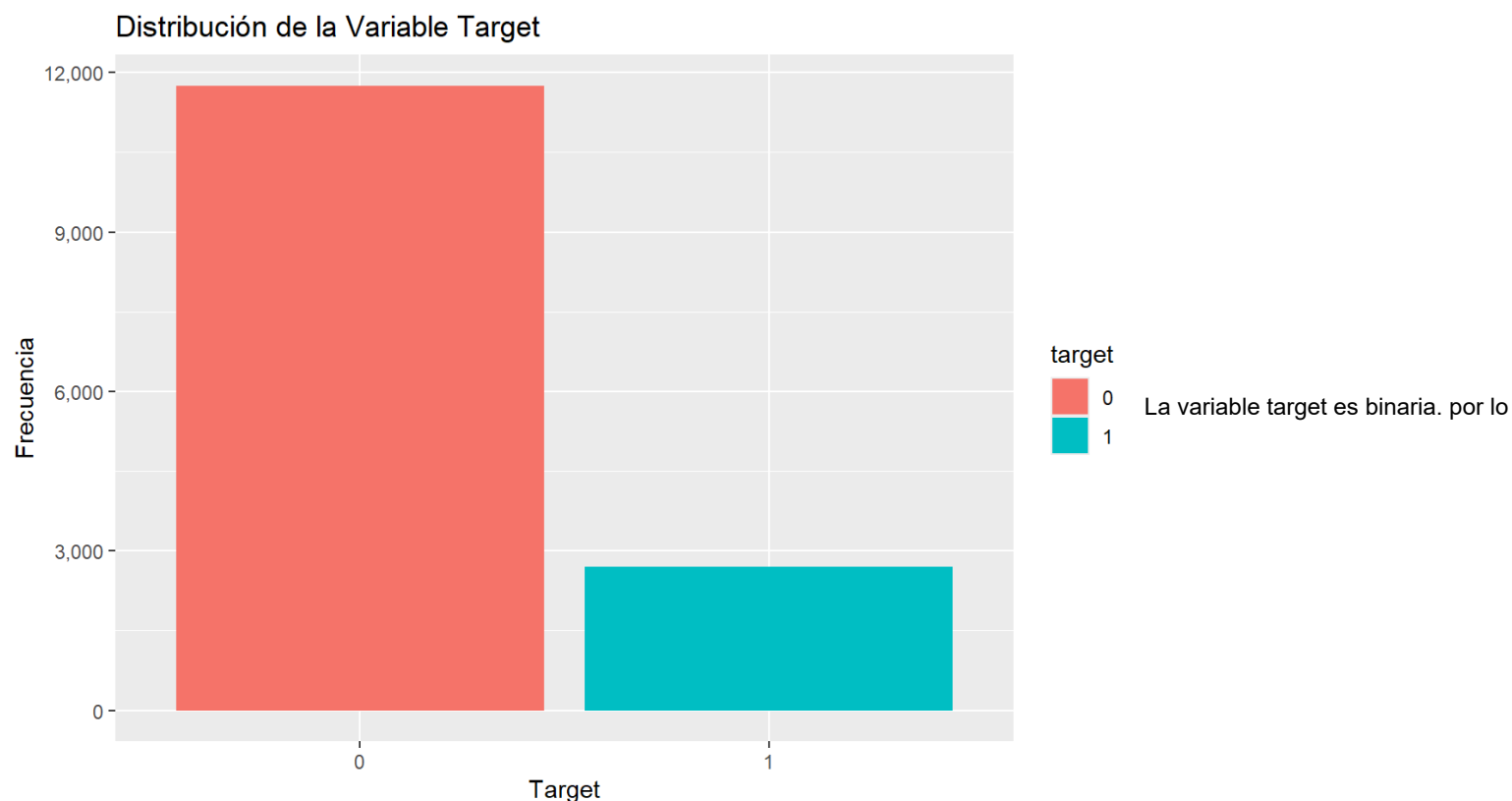
```
##           customer_id           loan_id
##                0                0
##  ACC_CREATION_DATETIME  APPLICATION_DATETIME
##                0                0
##  LOAN_ORIGINATION_DATETIME      max_days_late
##                0                0
##                target account_to_application_days
##                0                0
##           n_sf_apps      first_app_date
##                0                7648
##       last_app_date      n_bnpl_apps
##           7648                0
##  n_bnpl_approved_apps  first_bnpl_app_date
##                0                5715
##   last_bnpl_app_date      n_inquiries_l3m
##           5715                0
##       n_inquiries_l6m
##                0
```

Se ha verificado que no exista ningún valor faltante en las columnas necesarias para nuestra modelación

Análisis Exploratorio

Aquí realizaremos gráficos con su interpretación para las distribuciones, diagrama de correlación, etc.

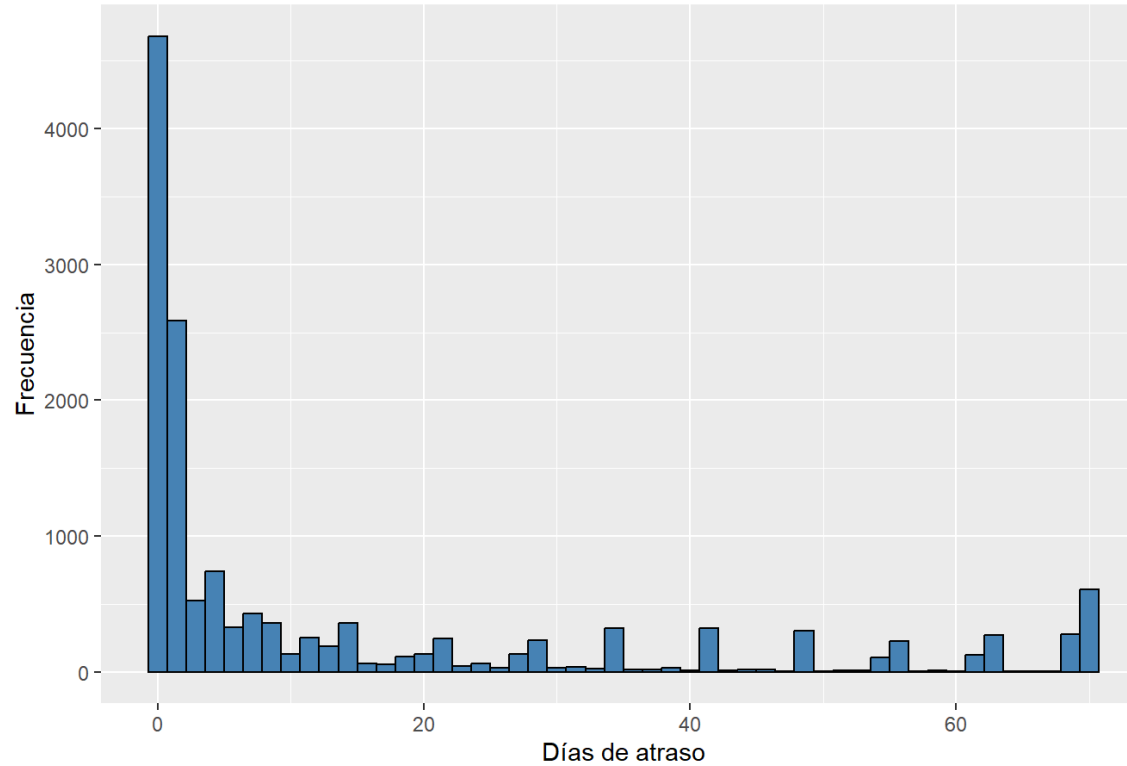
```
# Distribución de la variable target
ggplot(df, aes(x = target, fill = target)) +
  geom_bar() +
  scale_y_continuous(labels = comma) +
  labs(title = "Distribución de la Variable Target", x = "Target", y = "Frecuencia")
```



tanto sus valores solo son 0 y 1, tenemos casi 4 veces mas personas clasificadas con 0 que con 1, esto puede hablarnos de un desbalanceo.

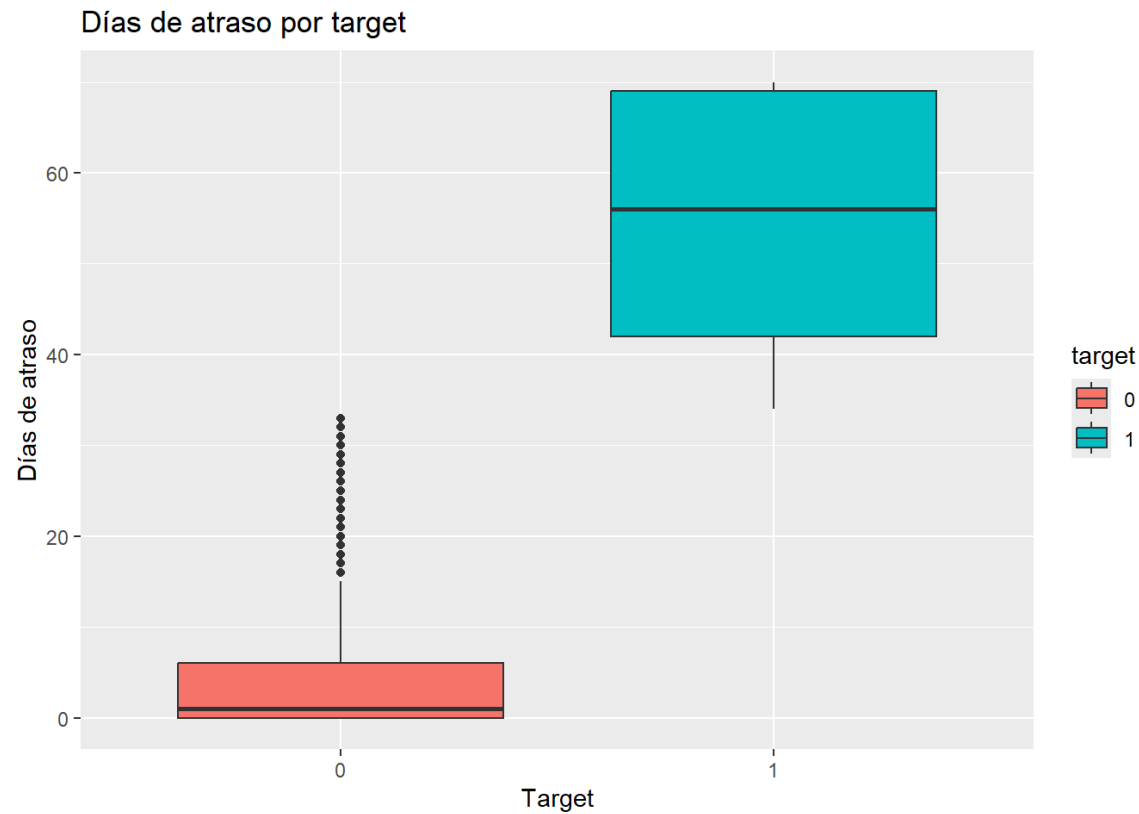
```
# Distribución de días de atraso
ggplot(df, aes(x = max_days_late)) +
  geom_histogram(bins = 50, fill = "steelblue", color = "black") +
  labs(title = "Distribución de días de atraso", x = "Días de atraso", y = "Frecuencia")
```

Distribución de días de atraso



se atrasan, tenemos mas datos en primeros días de retraso que en los posteriores, aunque no sigue una distribución convencional y posiblemente existan valores considerados atípicos por la cantidad de días que tardaron, sin embargo es información relevante para nuestros modelos.

```
# Boxplot de max_days_late por target
ggplot(df, aes(x = target, y = max_days_late, fill = target)) +
  geom_boxplot() +
  labs(title = "Días de atraso por target", x = "Target", y = "Días de atraso")
```

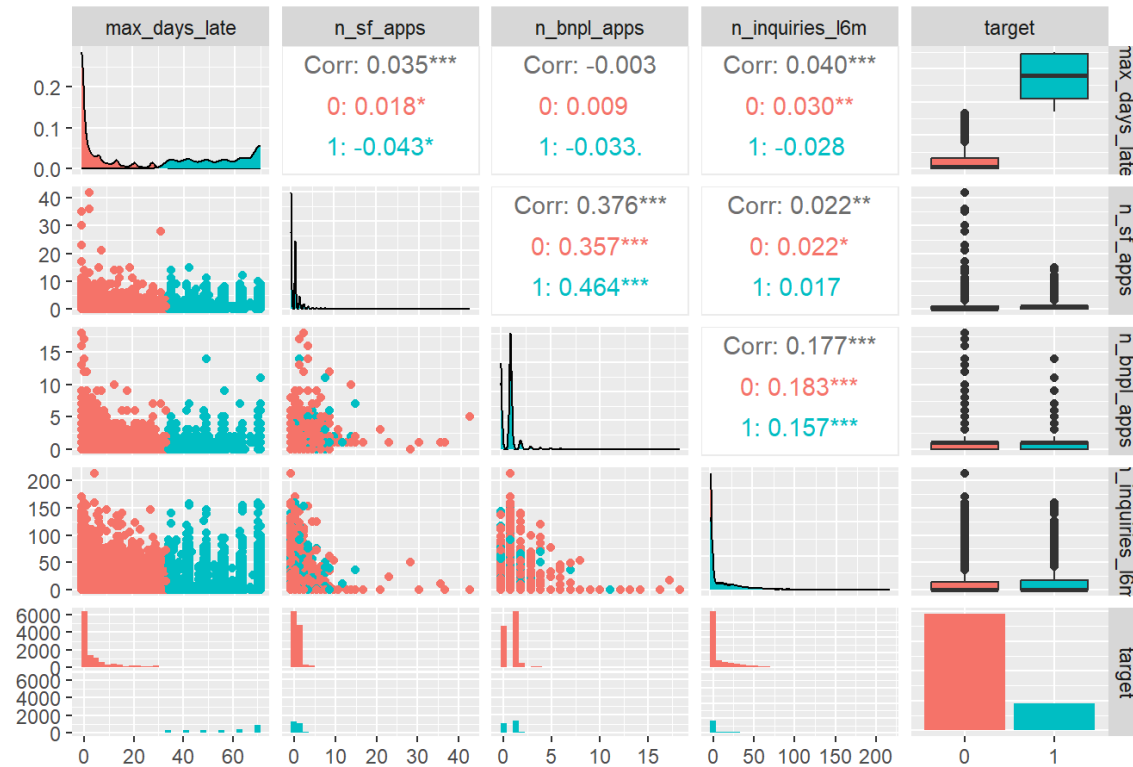



Para la clasificación de cero encontramos valores atípicos, estos se encuentran entre las dos categorías, fácilmente son los que podrían generar errores de clasificación a futuro, también vemos que la mediana es mucho más pequeña que la media por lo que existe asimetría fuerte en la categoría 0 (muchos valores en primeros días), a diferencia de la categoría 1 que es casi simétrica.

```
# Matriz de dispersión y correlación
ggpairs(df %>% dplyr::select(max_days_late, n_sf_apps, n_bnpl_apps, n_inquiries_l6m, target),
        mapping = aes(color = target), title = "Matriz de dispersión y correlación")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Matriz de dispersión y correlación



En el pairplot observamos que la variable que mejor separa las categorías es la variable `max_days_late`, por lo que será la variable mas importante cuando creemos nuestros modelos

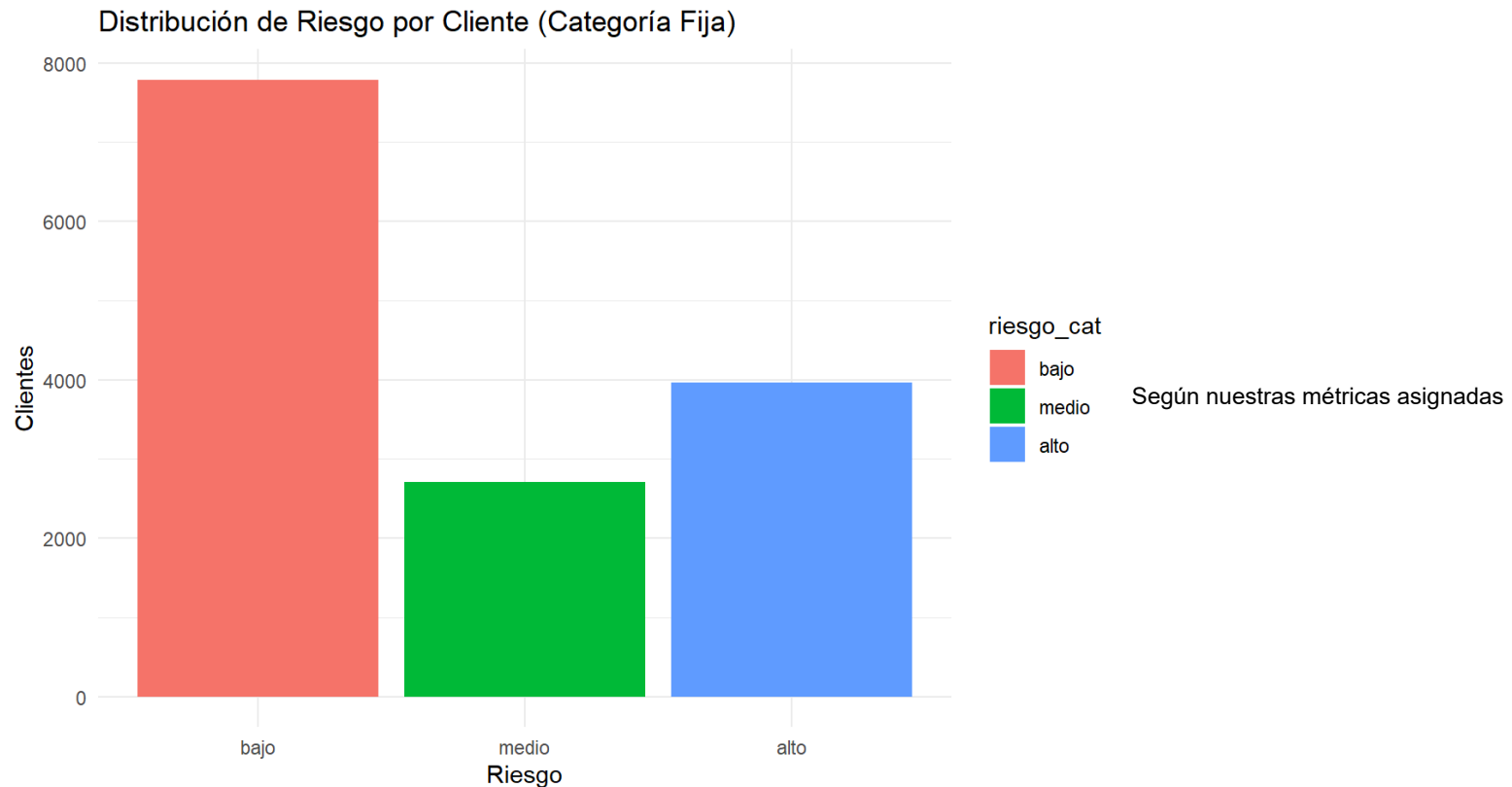
Dado que la variable mas importate es la mencionada anteriormente, utilizaremos dicha variable para clasificar a las personas como: - bajo riesgo: si han tenido un retraso máximo de 3 días - medio: si han retrasado entre 4 y 14 días - alto: si se han retrasado 15 días o mas

Creamos la categoría de riesgo sólo para análisis interpretativo

```
# Creación de la categoría de riesgo
# Esta variable es útil para la parte de tasas fijas, pero no se usará directamente en el modelo predictivo.
df <- df %>%
  mutate(
    riesgo_cat = case_when( # Renombrado a riesgo_cat para evitar confusión con el score numérico
      between(max_days_late, 0, 3) ~ "bajo",
      between(max_days_late, 4, 14) ~ "medio",
      max_days_late >= 15 ~ "alto"
    ),
    riesgo_cat = factor(riesgo_cat, levels = c("bajo", "medio", "alto"))
  )
```

Visualización de categorías creadas

```
df %>%
  count(riesgo_cat) %>%
  ggplot(aes(x = riesgo_cat, y = n, fill = riesgo_cat)) +
  geom_col() +
  labs(title = "Distribución de Riesgo por Cliente (Categoría Fija)", y = "Clientes", x = "Riesgo") +
  theme_minimal()
```



arbitrariamente esta sería la distribución de las categorías de riesgo

Parte dos

En esta sección importaremos los datos de crédito externos, los limpiaremos, codificaremos, crearemos nuevas variables útiles con base en esta base de datos y las agruparemos para agregarlas a nuestro main_dataset posteriormente, teniendo en él toda la información por cliente

Importamos los datos de crédito

```
df2 <- read_parquet("C:/Users/DELL/Downloads/credit_reports.parquet")
print(head(df2))
```

```
## # A tibble: 6 × 29
##   customer_id INQUIRY_TIME      CDC_INQUIRY_ID      INQUIRY_DATE
##         <int> <dtm>          <chr>          <dtm>
## 1         4223 2022-04-01 00:32:36 710278-27993a6e-2885-48d4... 2022-03-31 18:00:00
## 2         4223 2022-04-01 00:32:36 710278-27993a6e-2885-48d4... 2022-03-31 18:00:00
## 3         4223 2022-04-01 00:32:36 710278-27993a6e-2885-48d4... 2022-03-31 18:00:00
## 4         3490 2022-02-15 02:30:22 622857-6b4e9d95-7491-40c3... 2022-02-14 18:00:00
## 5         6486 2022-06-25 01:57:14 875073-46a5f149-19db-4193... 2022-06-24 19:00:00
## 6         6486 2022-06-25 01:57:14 875073-46a5f149-19db-4193... 2022-06-24 19:00:00
## # i 25 more variables: PREVENTION_KEY <chr>, CURRENCY <chr>, MAX_CREDIT <dbl>,
## # CREDIT_LIMIT <dbl>, PAYMENT_AMOUNT <dbl>, UPDATE_DATE <dtm>,
## # LOAN_OPENING_DATE <dtm>, LOAN_CLOSING_DATE <dtm>,
## # WORST_DELAY_DATE <dtm>, REPORT_DATE <dtm>, LAST_PURCHASE_DATE <dtm>,
## # LAST_PAYMENT_DATE <dtm>, PAYMENT_FREQUENCY <chr>, BUSINESS_TYPE <chr>,
## # CREDIT_TYPE <chr>, ACCOUNT_TYPE <chr>, RESPONSABILITY_TYPE <chr>,
## # TOTAL_PAYMENTS <dbl>, DELAYED_PAYMENTS <dbl>, CURRENT_PAYMENT <chr>, ...
```

```
print(summary(df2))
```

```

## customer_id      INQUIRY_TIME      CDC_INQUIRY_ID
## Min.      :      1  Min.      :2021-04-29 22:50:03.74  Length:287356
## 1st Qu.: 2776  1st Qu.:2022-03-22 21:03:45.14  Class :character
## Median : 6187  Median :2022-07-18 18:09:13.48  Mode  :character
## Mean      : 6334  Mean      :2022-07-02 13:56:59.07
## 3rd Qu.: 9760  3rd Qu.:2022-10-27 00:27:07.93
## Max.      :14416  Max.      :2023-05-17 15:20:36.89
##
## INQUIRY_DATE      PREVENTION_KEY      CURRENCY
## Min.      :2021-04-28 19:00:00.00  Length:287356  Length:287356
## 1st Qu.:2022-03-21 18:00:00.00  Class :character  Class :character
## Median :2022-07-17 19:00:00.00  Mode  :character  Mode  :character
## Mean      :2022-07-01 19:44:25.38
## 3rd Qu.:2022-10-26 19:00:00.00
## Max.      :2023-05-16 18:00:00.00
## NA's      :89
## MAX_CREDIT      CREDIT_LIMIT      PAYMENT_AMOUNT
## Min.      :      0  Min.      :      0  Min.      :      0
## 1st Qu.:      548  1st Qu.:      0  1st Qu.:      0
## Median :      2300  Median :      400  Median :      0
## Mean      :      15280  Mean      :      5526  Mean      :      1671
## 3rd Qu.:      7483  3rd Qu.:      3000  3rd Qu.:      149
## Max.      :404040416  Max.      :1900000  Max.      :1800000
## NA's      :182  NA's      :8357  NA's      :89
## UPDATE_DATE      LOAN_OPENING_DATE
## Min.      :2001-04-24 18:00:00  Min.      :1949-12-31 18:00:00.00
## 1st Qu.:2019-03-12 18:00:00  1st Qu.:2017-03-20 18:00:00.00
## Median :2021-11-21 18:00:00  Median :2020-04-01 18:00:00.00
## Mean      :2020-03-25 22:35:03  Mean      :2018-11-22 09:24:26.86
## 3rd Qu.:2022-06-14 19:00:00  3rd Qu.:2021-10-13 19:00:00.00
## Max.      :2023-05-06 18:00:00  Max.      :2023-03-30 18:00:00.00
## NA's      :89  NA's      :89
## LOAN_CLOSING_DATE      WORST_DELAY_DATE
## Min.      :2000-08-27 19:00:00.00  Min.      :1998-10-09 19:00:00.00
## 1st Qu.:2017-01-29 18:00:00.00  1st Qu.:2018-04-29 19:00:00.00
## Median :2020-01-26 18:00:00.00  Median :2021-05-30 19:00:00.00
## Mean      :2018-10-24 16:45:09.91  Mean      :2019-08-10 21:14:42.97
## 3rd Qu.:2021-09-19 19:00:00.00  3rd Qu.:2022-04-19 19:00:00.00
## Max.      :2023-03-30 18:00:00.00  Max.      :2023-04-11 18:00:00.00
## NA's      :94747  NA's      :202698
## REPORT_DATE      LAST_PURCHASE_DATE
## Min.      :2001-04-24 18:00:00.00  Min.      :1949-12-31 18:00:00.00
## 1st Qu.:2019-02-27 18:00:00.00  1st Qu.:2017-07-28 19:00:00.00
## Median :2021-11-09 18:00:00.00  Median :2020-08-02 19:00:00.00
## Mean      :2020-03-21 23:43:32.96  Mean      :2019-02-17 02:26:14.56
## 3rd Qu.:2022-06-11 19:00:00.00  3rd Qu.:2021-11-15 18:00:00.00
## Max.      :2023-05-06 18:00:00.00  Max.      :2023-04-06 18:00:00.00
## NA's      :89  NA's      :4142

```

```

## LAST_PAYMENT_DATE      PAYMENT_FREQUENCY BUSINESS_TYPE
## Min.   :1949-12-31 18:00:00.00 Length:287356      Length:287356
## 1st Qu.:2017-09-01 19:00:00.00 Class :character  Class :character
## Median :2020-09-05 19:00:00.00 Mode  :character  Mode  :character
## Mean   :2019-04-08 19:51:00.57
## 3rd Qu.:2021-12-26 18:00:00.00
## Max.   :2023-05-06 18:00:00.00
## NA's   :31677
## CREDIT_TYPE      ACCOUNT_TYPE      RESPONSABILITY_TYPE TOTAL_PAYMENTS
## Length:287356      Length:287356      Length:287356      Min.   : 0.00
## Class :character    Class :character    Class :character    1st Qu.: 1.00
## Mode  :character    Mode  :character    Mode  :character    Median : 3.00
##                                     Mean   : 22.77
##                                     3rd Qu.: 16.00
##                                     Max.   :1800.00
##                                     NA's   :18645
## DELAYED_PAYMENTS CURRENT_PAYMENT      WORST_DELAY      TOTAL_REPORTED_PAYMENTS
## Min.   : 0.000      Length:287356      Min.   : 0.000      Min.   :0
## 1st Qu.: 0.000      Class :character    1st Qu.: 0.000      1st Qu.:0
## Median : 0.000      Mode  :character    Median : 0.000      Median :0
## Mean   : 3.818
##                                     Mean   : 4.279      Mean   :0
## 3rd Qu.: 1.000
##                                     3rd Qu.: 2.000      3rd Qu.:0
## Max.   :96.000
##                                     Max.   :84.000      Max.   :0
## NA's   :89
##                                     NA's   :3210      NA's   :41941
## CURRENT_BALANCE      BALANCE_DUE      BALANCE_DUE_WORST_DELAY
## Min.   : 0      Min.   : 0      Min.   : 0
## 1st Qu.: 0      1st Qu.: 0      1st Qu.: 0
## Median : 0      Median : 0      Median : 0
## Mean   : 4578      Mean   : 2090      Mean   : 1672
## 3rd Qu.: 273      3rd Qu.: 0      3rd Qu.: 159
## Max.   :3469743      Max.   :1800000      Max.   :1800000
## NA's   :89      NA's   :89      NA's   :89

```

```
print(colSums(is.na(df2)))
```

##	customer_id	INQUIRY_TIME	CDC_INQUIRY_ID
##	0	0	89
##	INQUIRY_DATE	PREVENTION_KEY	CURRENCY
##	89	89	89
##	MAX_CREDIT	CREDIT_LIMIT	PAYMENT_AMOUNT
##	182	8357	89
##	UPDATE_DATE	LOAN_OPENING_DATE	LOAN_CLOSING_DATE
##	89	89	94747
##	WORST_DELAY_DATE	REPORT_DATE	LAST_PURCHASE_DATE
##	202698	89	4142
##	LAST_PAYMENT_DATE	PAYMENT_FREQUENCY	BUSINESS_TYPE
##	31677	89	89
##	CREDIT_TYPE	ACCOUNT_TYPE	RESPONSABILITY_TYPE
##	89	89	89
##	TOTAL_PAYMENTS	DELAYED_PAYMENTS	CURRENT_PAYMENT
##	18645	89	89
##	WORST_DELAY	TOTAL_REPORTED_PAYMENTS	CURRENT_BALANCE
##	3210	41941	89
##	BALANCE_DUE	BALANCE_DUE_WORST_DELAY	
##	89	89	

Limpieza y Tratamiento de NA datos de crédito externos

Observamos que muchas variables tienen 89 valores faltantes, lo que nos dice que son personas sin crédito registrado, imputaremos los valores faltantes para las columnas numéricas con cero. Nos aseguraremos que las columnas de fechas estén guardadas de esa manera.

```
# Eliminar observaciones sin crédito registrado (PAYMENT_FREQUENCY es un buen indicador de registro válido)
df2 <- df2 %>%
  filter(!is.na(PAYMENT_FREQUENCY))

# Imputar valores faltantes en columnas numéricas con 0
cols_to_impute_0_df2 <- c("MAX_CREDIT", "CREDIT_LIMIT", "PAYMENT_AMOUNT", "CURRENT_BALANCE",
  "BALANCE_DUE", "BALANCE_DUE_WORST_DELAY", "DELAYED_PAYMENTS",
  "WORST_DELAY", "TOTAL_PAYMENTS", "TOTAL_REPORTED_PAYMENTS")

df2 <- df2 %>%
  mutate(across(all_of(cols_to_impute_0_df2), ~replace_na(., 0)))

# Convertir columnas de fecha
date_cols_df2 <- c("LOAN_OPENING_DATE", "LOAN_CLOSING_DATE", "UPDATE_DATE",
  "WORST_DELAY_DATE", "REPORT_DATE", "LAST_PURCHASE_DATE",
  "LAST_PAYMENT_DATE")

df2 <- df2 %>%
  mutate(across(all_of(date_cols_df2), ~as_date(.)))

print(colSums(is.na(df2)))
```

##	customer_id	INQUIRY_TIME	CDC_INQUIRY_ID
##	0	0	0
##	INQUIRY_DATE	PREVENTION_KEY	CURRENCY
##	0	0	0
##	MAX_CREDIT	CREDIT_LIMIT	PAYMENT_AMOUNT
##	0	0	0
##	UPDATE_DATE	LOAN_OPENING_DATE	LOAN_CLOSING_DATE
##	0	0	94658
##	WORST_DELAY_DATE	REPORT_DATE	LAST_PURCHASE_DATE
##	202609	0	4053
##	LAST_PAYMENT_DATE	PAYMENT_FREQUENCY	BUSINESS_TYPE
##	31588	0	0
##	CREDIT_TYPE	ACCOUNT_TYPE	RESPONSABILITY_TYPE
##	0	0	0
##	TOTAL_PAYMENTS	DELAYED_PAYMENTS	CURRENT_PAYMENT
##	0	0	0
##	WORST_DELAY	TOTAL_REPORTED_PAYMENTS	CURRENT_BALANCE
##	0	0	0
##	BALANCE_DUE	BALANCE_DUE_WORST_DELAY	
##	0	0	

Únicamente nos quedan valores faltantes en las columnas de fecha, cada una con diferente cantidad de valores pero no podemos borrar las filas con esas ausencias, ya que perderíamos mucha información relevante

Ingeniería de datos

En esta sección crearemos nuevas variables útiles para las características de riesgo externo, en caso de crear algún NA reemplazará por cero

```
# Ingeniería de Variables en df2
df2 <- df2 %>%
  mutate(
    # Tiempo de vida del crédito en meses
    tiempo_credito_meses = as.numeric(interval(LOAN_OPENING_DATE, UPDATE_DATE), "months"),
    # Uso de crédito: Saldo actual / Límite de crédito. Manejar división por cero.
    uso_credito_pct = if_else(CREDIT_LIMIT > 0, CURRENT_BALANCE / CREDIT_LIMIT, 0),
    # Porcentaje de pagos atrasados: Pagos atrasados / Total de pagos.
    pagos_tarde_pct = if_else(TOTAL_PAYMENTS > 0, DELAYED_PAYMENTS / TOTAL_PAYMENTS, 0)
  ) %>%
  # Limitar uso_credito_pct a 1 (no puede ser más del 100%)
  mutate(uso_credito_pct = pmin(uso_credito_pct, 1))

# Reemplazamos posibles NAs
df2 <- df2 %>%
  mutate(across(c(tiempo_credito_meses, uso_credito_pct, pagos_tarde_pct), ~replace_na(., 0)))
```


NOTA CLAVE DEL PROBLEMA: La relación es de uno a muchos (un customer_id en main_dataset puede tener múltiples registros en credit_reports). Para usar esta información en el modelo del main_dataset (que es por loan_id, y cada loan_id tiene un customer_id), necesitamos resumir el historial de credit_reports por customer_id.

Agregación de credit_reports a nivel customer_id

```
df2_aggregated <- df2 %>%
  group_by(customer_id) %>%
  summarise(
    # Métricas de uso de crédito
    mean_uso_credito_ext = mean(uso_credito_pct, na.rm = TRUE),
    max_uso_credito_ext = max(uso_credito_pct, na.rm = TRUE),

    # Métricas de atraso
    max_worst_delay_ext = max(WORST_DELAY, na.rm = TRUE), # Peor atraso en cualquier crédito externo
    mean_delayed_payments_pct_ext = mean(pagos_tarde_pct, na.rm = TRUE),

    # Cantidad y tipo de créditos
    n_external_credits = n(), # Número total de créditos externos reportados
    n_distinct_credit_types = n_distinct(CREDIT_TYPE),

    # Historial de crédito
    mean_tiempo_credito_meses_ext = mean(tiempo_credito_meses, na.rm = TRUE),
    max_total_payments_ext = max(TOTAL_PAYMENTS, na.rm = TRUE),

  ) %>%
  ungroup()

# Verificamos que no haya ningún NA en la agrupación que creamos
colSums(is.na(df2_aggregated))
```

```
##           customer_id      mean_uso_credito_ext
##                0                0
##      max_uso_credito_ext      max_worst_delay_ext
##                0                0
## mean_delayed_payments_pct_ext      n_external_credits
##                0                0
##      n_distinct_credit_types mean_tiempo_credito_meses_ext
##                0                0
##      max_total_payments_ext
##                0
```

```
# Resumen
summary(df2_aggregated)
```

```
## customer_id mean_uso_credito_ext max_uso_credito_ext max_worst_delay_ext
## Min. : 1 Min. :0.00000 Min. :0.0000 Min. : 0.00
## 1st Qu.: 2698 1st Qu.:0.09056 1st Qu.:1.0000 1st Qu.:13.00
## Median : 5694 Median :0.17677 Median :1.0000 Median :34.00
## Mean : 6114 Mean :0.21368 Mean :0.8974 Mean :42.02
## 3rd Qu.: 9429 3rd Qu.:0.29917 3rd Qu.:1.0000 3rd Qu.:84.00
## Max. :14416 Max. :1.00000 Max. :1.0000 Max. :84.00
## mean_delayed_payments_pct_ext n_external_credits n_distinct_credit_types
## Min. : 0.00000 Min. : 1.00 Min. : 1.000
## 1st Qu.: 0.09635 1st Qu.: 11.00 1st Qu.: 4.000
## Median : 0.43310 Median : 22.00 Median : 5.000
## Mean : 0.89468 Mean : 31.32 Mean : 5.184
## 3rd Qu.: 1.18198 3rd Qu.: 41.00 3rd Qu.: 7.000
## Max. :25.00000 Max. :269.00 Max. :14.000
## mean_tiempo_credito_meses_ext max_total_payments_ext
## Min. : 0.4415 Min. : 0.0
## 1st Qu.: 10.8016 1st Qu.: 33.0
## Median : 16.8158 Median : 96.0
## Mean : 20.5255 Mean : 234.1
## 3rd Qu.: 25.6553 3rd Qu.: 360.0
## Max. :183.0418 Max. :1800.0
```

En el summary podemos ver que tenemos 14416 clientes, el promedio de uso de crédito externo es de 21%, el promedio de máximo uso de crédito externo es de casi 90%, la mayor parte de los clientes se atrasan en los primeros días, etc.

PARTE 3: Integración de Datos y Modelado Final

En esta sección se unirá la información de las bases de datos y las variables creadas por cliente según su id, crearemos unos modelos de aprendizaje automático, finalmente compararemos las métricas por modelo.

```
# Unir por customer_id
df_final_model <- left_join(df, df2_aggregated, by = "customer_id")
colSums(is.na(df_final_model))
```

```
##          customer_id          loan_id
##          0              0
##    ACC_CREATION_DATETIME    APPLICATION_DATETIME
##          0              0
##    LOAN_ORIGINATION_DATETIME    max_days_late
##          0              0
##          target    account_to_application_days
##          0              0
##          n_sf_apps    first_app_date
##          0              7648
##          last_app_date    n_bnpl_apps
##          7648              0
##    n_bnpl_approved_apps    first_bnpl_app_date
##          0              5715
##          last_bnpl_app_date    n_inquiries_l3m
##          5715              0
##          n_inquiries_l6m    riesgo_cat
##          0              0
##    mean_uso_credito_ext    max_uso_credito_ext
##          5282              5282
##    max_worst_delay_ext    mean_delayed_payments_pct_ext
##          5282              5282
##    n_external_credits    n_distinct_credit_types
##          5282              5282
##    mean_tiempo_credito_meses_ext    max_total_payments_ext
##          5282              5282
```

Los NA significa que el cliente con ese ID no tiene datos externos, posiblemente no ha solicitado ningún crédito fuera de este negocio. Los NA serán reemplazados por cero en las variables que provienen de la base de datos de crédito

```
# Reemplazamos NAs
df_final_model <- df_final_model %>%
  mutate(across(c(mean_uso_credito_ext, max_uso_credito_ext,
                  max_worst_delay_ext, mean_delayed_payments_pct_ext,
                  n_external_credits, n_distinct_credit_types,
                  mean_tiempo_credito_meses_ext, max_total_payments_ext),
    ~replace_na(., 0)))
```

Selección de variables para nuestro modelo de aprendizaje automático, omitiremos las variables de fecha de nuestro main_dataset

```
# Incluimos las variables numéricas y la variable objetivo de nuestro nuevo dataset
```

```
vars_usar_final <- c("target", "max_days_late", "n_sf_apps", "n_bnpl_apps",  
                    "n_bnpl_approved_apps", "n_inquiries_l3m", "n_inquiries_l6m",  
                    "account_to_application_days",  
                    # Nuevas variables de credit reports agregadas  
                    "mean_uso_credito_ext", "max_uso_credito_ext",  
                    "max_worst_delay_ext", "mean_delayed_payments_pct_ext",  
                    "n_external_credits", "n_distinct_credit_types",  
                    "mean_tiempo_credito_meses_ext", "max_total_payments_ext"  
)
```

```
df_model_final <- df_final_model %>% dplyr::select(all_of(vars_usar_final))
```

```
df_model_final <- df_model_final %>%
```

```
  mutate(across(where(is.integer), as.numeric))
```

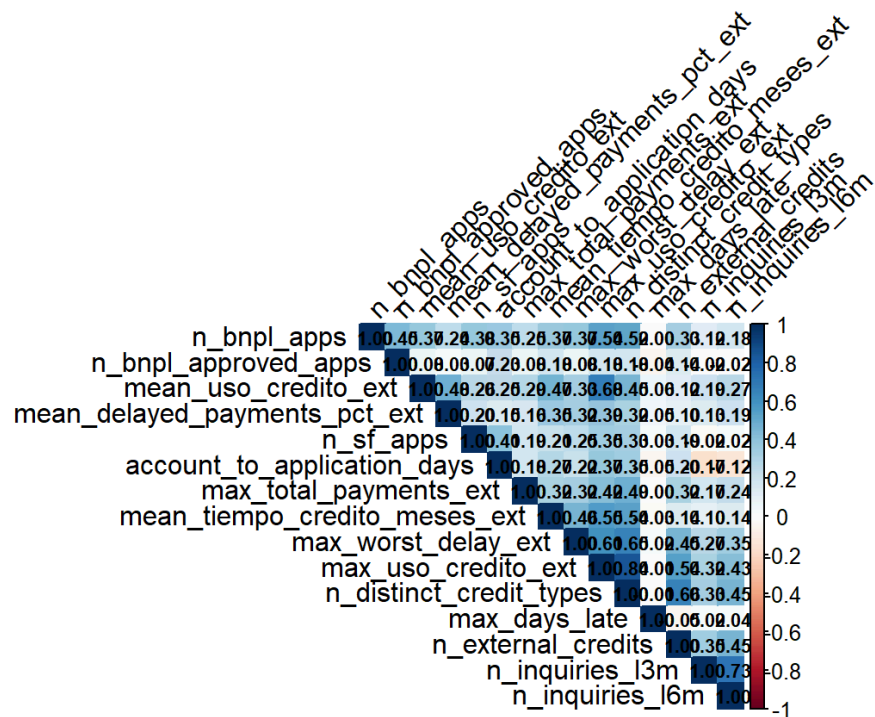
```
summary(df_model_final)
```

```
## target      max_days_late      n_sf_apps      n_bnpl_apps
## 0:11754  Min.    : 0.00  Min.    : 0.0000  Min.    : 0.0000
## 1: 2700  1st Qu.: 0.00  1st Qu.: 0.0000  1st Qu.: 0.0000
##          Median : 2.00  Median : 0.0000  Median : 1.0000
##          Mean   :14.28  Mean   : 0.7788  Mean   : 0.7387
##          3rd Qu.:20.00  3rd Qu.: 1.0000  3rd Qu.: 1.0000
##          Max.   :70.00  Max.   :42.0000  Max.   :18.0000
## n_bnpl_approved_apps n_inquiries_l3m  n_inquiries_l6m
## Min.    : 0.0000      Min.    : 0.000  Min.    : 0.00
## 1st Qu.: 0.0000      1st Qu.: 0.000  1st Qu.: 0.00
## Median : 0.0000      Median : 0.000  Median : 0.00
## Mean   : 0.1602      Mean   : 6.504  Mean   : 10.76
## 3rd Qu.: 0.0000      3rd Qu.: 0.000  3rd Qu.: 15.00
## Max.   :15.0000      Max.   :170.000  Max.   :213.00
## account_to_application_days mean_uso_credito_ext max_uso_credito_ext
## Min.    : 0.0          Min.    :0.0000  Min.    :0.0000
## 1st Qu.: 0.0          1st Qu.:0.0000  1st Qu.:0.0000
## Median :103.0         Median :0.0771  Median :1.0000
## Mean   :163.5         Mean   :0.1356  Mean   :0.5694
## 3rd Qu.:271.8         3rd Qu.:0.2207  3rd Qu.:1.0000
## Max.   :901.0         Max.   :1.0000  Max.   :1.0000
## max_worst_delay_ext mean_delayed_payments_pct_ext n_external_credits
## Min.    : 0.00      Min.    : 0.00000      Min.    : 0.00
## 1st Qu.: 0.00      1st Qu.: 0.00000      1st Qu.: 0.00
## Median :13.00      Median : 0.06919      Median : 9.00
## Mean   :26.66      Mean   : 0.56773      Mean   : 19.87
## 3rd Qu.:53.00      3rd Qu.: 0.68072      3rd Qu.: 28.00
## Max.   :84.00      Max.   :25.00000      Max.   :269.00
## n_distinct_credit_types mean_tiempo_credito_meses_ext max_total_payments_ext
## Min.    : 0.00      Min.    : 0.000      Min.    : 0.0
## 1st Qu.: 0.00      1st Qu.: 0.000      1st Qu.: 0.0
## Median : 4.00      Median : 9.963      Median : 26.0
## Mean   : 3.29      Mean   : 13.025      Mean   : 148.5
## 3rd Qu.: 6.00      3rd Qu.: 20.011      3rd Qu.: 102.0
## Max.   :14.00      Max.   :183.042      Max.   :1800.0
```

Del summary anterior llama la atención la proporción de datos en target, cero tiene una proporción mucho mayor, de nuevo hablamos de un desbalance para la construcción del modelo de aprendizaje automático

Realizamos un diagrama de correlación para evitar tener multicolinealidad perfecta, que las variables aporten lo mismo y/o afecten a nuestro modelo

```
# Correlación de las variables del modelo final
cor_matrix_final <- cor(dplyr::select(df_model_final, where(is.numeric)), use = "pairwise.complete.obs")
corrplot(cor_matrix_final, method = "color", type = "upper", order = "hclust",
          addCoef.col = "black", tl.col = "black", tl.srt = 45, number.cex = 0.7)
```



Las únicas variables correlacionadas

fuertemente son: max_uso_crédito_ext y n_distinct_credit_types Sin embargo no es tan alto como para preocuparnos, se mantendrán todas las variables.

Realizamos la partición en datos de entrenamiento y prueba, utilizando el 80% de los datos para entrenar el modelo

```
# Partición en datos de entrenamiento y prueba
set.seed(42)
part <- createDataPartition(df_model_final$target, p = 0.8, list = FALSE)
train <- df_model_final[part, ]
test <- df_model_final[-part, ]
table(train$target)
```

```
##
##      0      1
## 9404 2160
```

Como tenemos muchos mas datos en la clase 0 realizaremos un balanceo en nuestros datos de entrenamiento

```
# Balanceo con ROSE
train_bal <- ROSE(target ~ ., data = train, seed = 42)$data
# Verificamos el balanceo
table(train_bal$target)
```

```
##
##      0      1
## 5761 5803
```

Creamos una función para los resultados de las métricas de nuestros modelos y poderlos comparar posteriormente con una tabla

```
# Función para las métricas
eval_metrics <- function(pred, obs, modelo) {
  pred <- factor(pred, levels = c("0", "1"))
  obs <- factor(obs, levels = c("0", "1"))
  cm <- confusionMatrix(pred, obs, positive = "1")
  f1 <- F1_Score(pred, obs, positive = "1")
  prec <- cm$byClass["Precision"]
  rec <- cm$byClass["Sensitivity"]
  acc <- cm$overall["Accuracy"]
  kappa <- cm$overall["Kappa"]
  data.frame(Modelo = modelo, Accuracy = acc, Kappa = kappa,
             F1 = f1, Precision = prec, Recall = rec)
}
resultados <- list()
```

Regresión logística

```
modelo_glm <- glm(target ~ ., data = train_bal, family = "binomial")
pred_prob_glm <- predict(modelo_glm, newdata = test, type = "response")
pred_glm_class <- as.factor(ifelse(pred_prob_glm > 0.7, 1, 0)) # Clasificación binaria con umbral 0.7
res_glm <- eval_metrics(pred_glm_class, test$target, "Regresión Logística")
print(confusionMatrix(pred_glm_class, test$target, positive = "1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 2329    0
##           1   21  540
##
##           Accuracy : 0.9927
##           95% CI : (0.9889, 0.9955)
##           No Information Rate : 0.8131
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.9764
##
##           Mcnemar's Test P-Value : 1.275e-05
##
##           Sensitivity : 1.0000
##           Specificity : 0.9911
##           Pos Pred Value : 0.9626
##           Neg Pred Value : 1.0000
##           Prevalence : 0.1869
##           Detection Rate : 0.1869
##           Detection Prevalence : 0.1941
##           Balanced Accuracy : 0.9955
##
##           'Positive' Class : 1
##
```

```
resultados[["Regresión Logística"]] <- res_glm
```

Árbol de decisión

```
# Usamos rpart para árboles de clasificación
modelo_dt <- rpart(target ~ ., data = train_bal, method = "class",
                   control = rpart.control(minsplit = 20, cp = 0.01)) # minsplit y cp para controlar complejidad
pred_dt <- predict(modelo_dt, newdata = test, type = "class")
res_dt <- eval_metrics(pred_dt, test$target, "Árbol de Decisión")
print(confusionMatrix(pred_dt, test$target, positive = "1"))
```

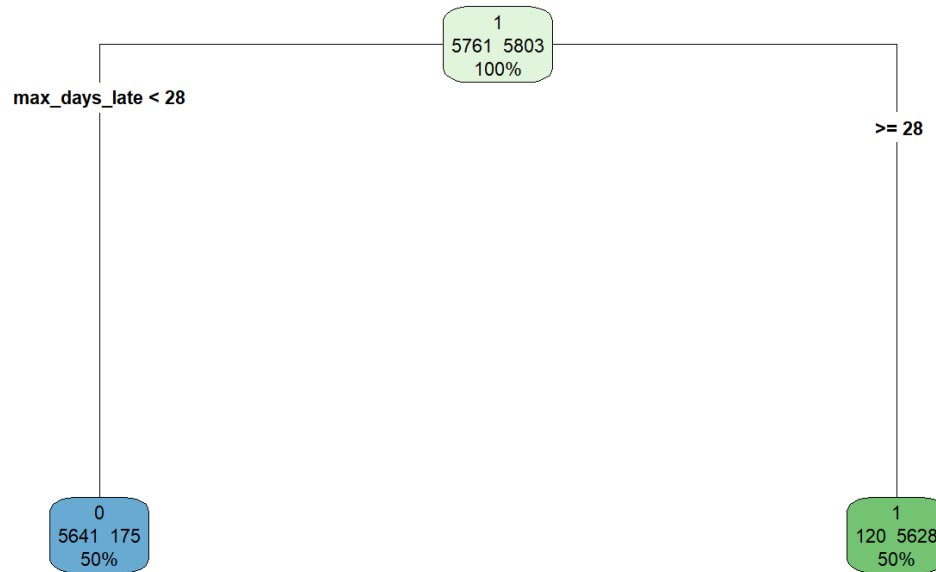


```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 2325    0
##           1   25  540
##
##           Accuracy : 0.9913
##           95% CI : (0.9873, 0.9944)
##       No Information Rate : 0.8131
##       P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.972
##
##  Mcnemar's Test P-Value : 1.587e-06
##
##           Sensitivity : 1.0000
##           Specificity : 0.9894
##           Pos Pred Value : 0.9558
##           Neg Pred Value : 1.0000
##           Prevalence : 0.1869
##           Detection Rate : 0.1869
##       Detection Prevalence : 0.1955
##           Balanced Accuracy : 0.9947
##
##           'Positive' Class : 1
##
```

```
resultados[["Árbol de Decisión"]] <- res_dt
```

```
# Visualización del Árbol de Decisión (opcional, puede ser grande)
rpart.plot(modelo_dt, type = 4, extra = 101, fallen.leaves = TRUE, cex = 0.7,
  main = "Árbol de Decisión para Predicción de Riesgo")
```

Árbol de Decisión para Predicción de Riesgo



Con el árbol de decisión verificamos

que la variable más importante es mas_day_late

XGBoost

```
train_x <- as.matrix(dplyr::select(train_bal, -target))
train_y <- as.numeric(train_bal$target) - 1 # 0 y 1 para XGBoost
dtrain <- xgb.DMatrix(data = train_x, label = train_y)

# Parámetros optimizados para XGBoost (puedes ajustar más si es necesario)
params <- list(objective = "binary:logistic",
               eval_metric = "auc",
               eta = 0.1,      # Tasa de aprendizaje
               max_depth = 4,  # Profundidad máxima del árbol
               subsample = 0.8, # Submuestreo de filas
               colsample_bytree = 0.8 # Submuestreo de columnas
               )

xgb_model <- xgb.train(params = params, dtrain, nrounds = 150, verbose = 0) # Aumentar nrounds para mejor rendimiento

# Predicciones de probabilidad en el conjunto de prueba
test_x <- as.matrix(dplyr::select(test, -target))
pred_prob_xgb <- predict(xgb_model, test_x)
pred_xgb_class <- as.factor(ifelse(pred_prob_xgb > 0.5, 1, 0)) # Clasificación binaria con umbral 0.5

res_xgb <- eval_metrics(pred_xgb_class, test$target, "XGBoost")
print(confusionMatrix(pred_xgb_class, test$target, positive = "1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 2318    0
##           1   32  540
##
##           Accuracy : 0.9889
##           95% CI : (0.9844, 0.9924)
##           No Information Rate : 0.8131
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.9644
##
## Mcnemar's Test P-Value : 4.251e-08
##
##           Sensitivity : 1.0000
##           Specificity : 0.9864
##           Pos Pred Value : 0.9441
##           Neg Pred Value : 1.0000
##           Prevalence : 0.1869
##           Detection Rate : 0.1869
##           Detection Prevalence : 0.1979
##           Balanced Accuracy : 0.9932
##
##           'Positive' Class : 1
##
```

```
resultados[["XGBoost"]] <- res_xgb
```

Bosque aleatorio

```
set.seed(42)
modelo_rf <- randomForest(target ~ ., data = train_bal, ntree = 100, maxnodes = 80)
pred_rf <- predict(modelo_rf, newdata = test)
res_rf <- eval_metrics(pred_rf, test$target, "Random Forest")
print(confusionMatrix(pred_rf, test$target, positive = "1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 2310    0
##           1   40  540
##
##           Accuracy : 0.9862
##           95% CI : (0.9812, 0.9901)
##       No Information Rate : 0.8131
##       P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.9557
##
##  Mcnemar's Test P-Value : 6.984e-10
##
##           Sensitivity : 1.0000
##           Specificity : 0.9830
##           Pos Pred Value : 0.9310
##           Neg Pred Value : 1.0000
##           Prevalence : 0.1869
##           Detection Rate : 0.1869
##       Detection Prevalence : 0.2007
##           Balanced Accuracy : 0.9915
##
##           'Positive' Class : 1
##
```

```
resultados[["Random Forest"]] <- res_rf
```

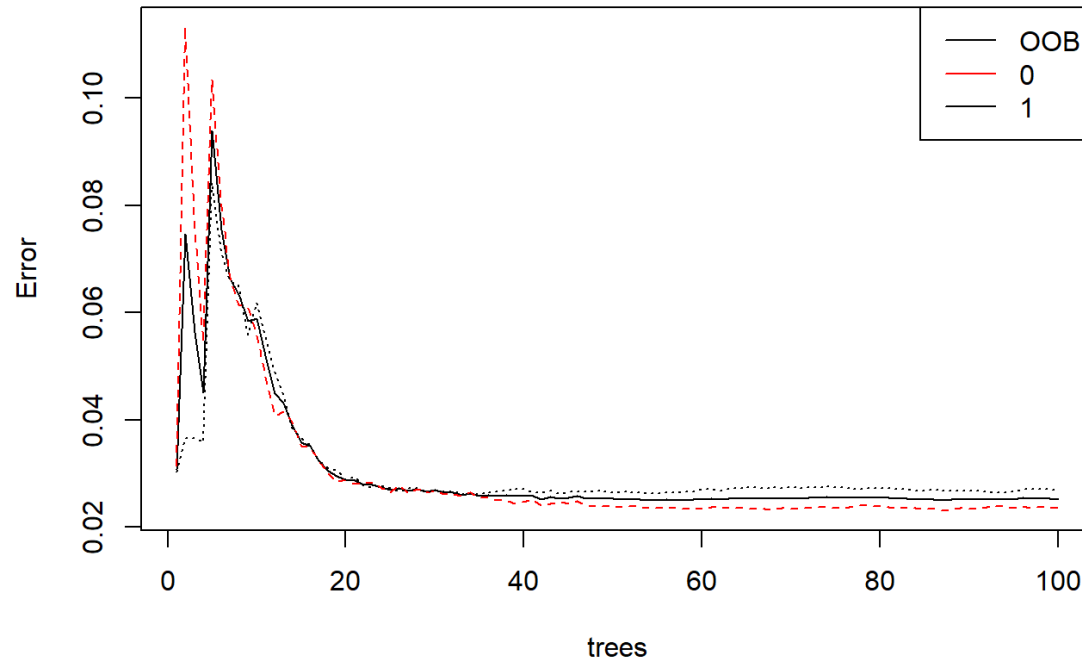
Error OOB para bosque aleatorio

```
# Tabla error OOB
oob_error <- modelo_rf$err.rate[modelo_rf$ntree, "OOB"]
cat("Error OOB final:", round(oob_error * 100, 2), "%\n")
```

```
## Error OOB final: 2.53 %
```

```
plot(modelo_rf,
     main = "Error OOB vs Número de Árboles",
     col = c("black", "red"))
legend("topright",
     legend = colnames(modelo_rf$err.rate),
     col = c("black", "red"),
     lty = 1)
```

Error OOB vs Número de Árboles

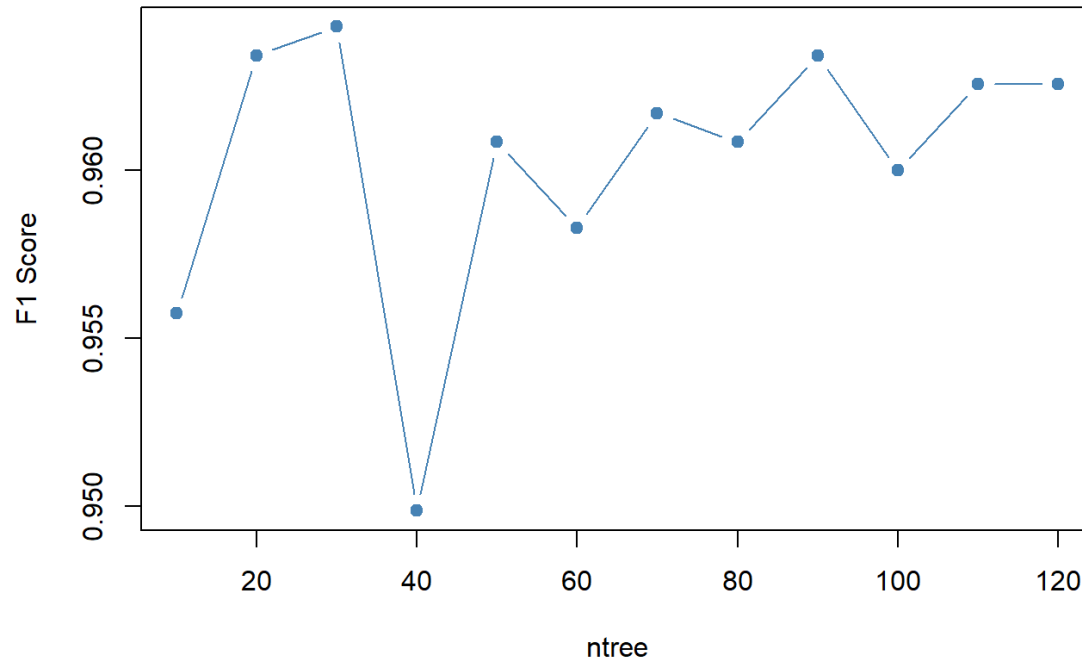


```
oob_error <- modelo_rf$err.rate[modelo_rf$ntree, "OOB"]  
cat("Error OOB final:", round(oob_error * 100, 2), "%\n")
```

```
## Error OOB final: 2.53 %
```

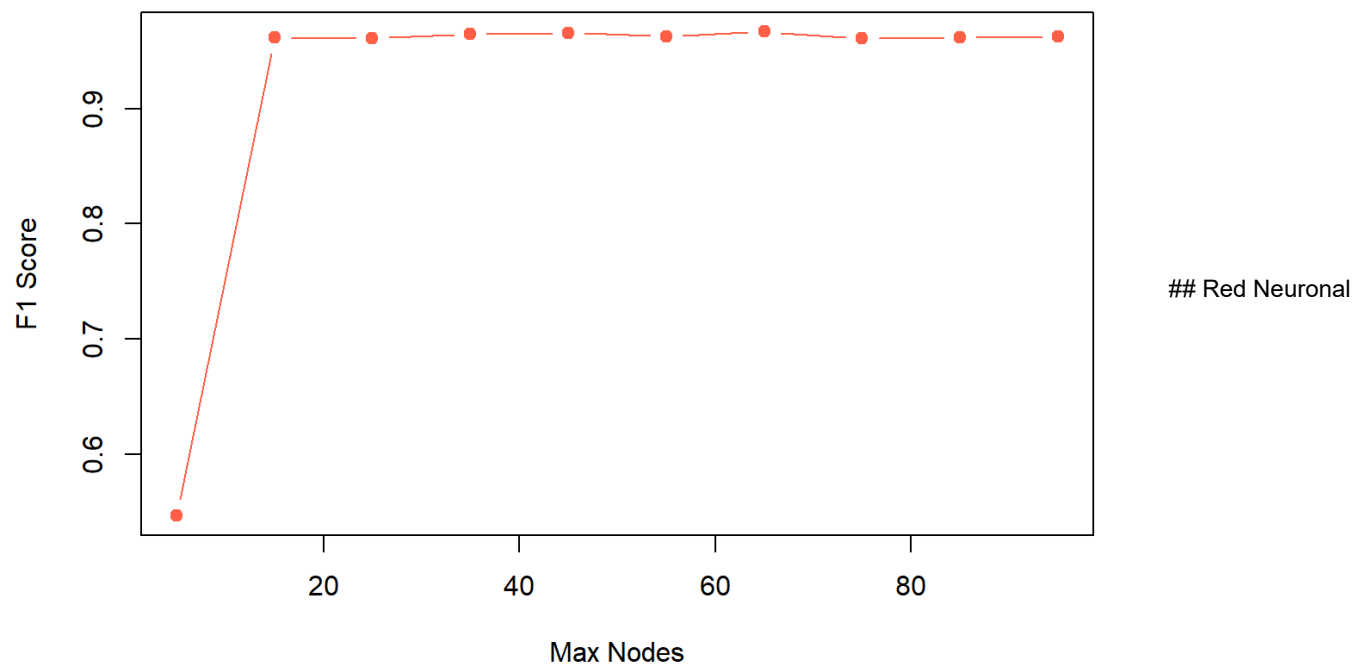
```
# F1 vs Número de árboles  
n_arboles <- seq(10, 120, by = 10)  
f1_scores <- sapply(n_arboles, function(n) {  
  m <- randomForest(target ~ ., data = train_bal, ntree = n)  
  pred <- predict(m, newdata = test)  
  F1_Score(pred, test$target, positive = "1")  
})  
plot(n_arboles, f1_scores, type = "b", pch = 19, col = "steelblue",  
     main = "F1 Score vs Número de Árboles", xlab = "ntree", ylab = "F1 Score")
```

F1 Score vs Número de Árboles



```
# F1 vs Número de nodos
max_nodes <- seq(5, 100, by = 10)
f1_nodes <- sapply(max_nodes, function(depth) {
  m <- randomForest(target ~ ., data = train_bal, ntree = 90, maxnodes = depth)
  pred <- predict(m, newdata = test)
  F1_Score(pred, test$target, positive = "1")
})
plot(max_nodes, f1_nodes, type = "b", pch = 19, col = "tomato",
     main = "F1 Score vs Nodos Máximos", xlab = "Max Nodes", ylab = "F1 Score")
```

F1 Score vs Nodos Máximos



```
set.seed(42)
nn_model <- nnet(target ~ ., data = train_bal, size = 5, decay = 0.01, maxit = 100, trace = FALSE)
pred_nn <- predict(nn_model, test, type = "class")
pred_nn <- factor(pred_nn, levels = c("0", "1"))
res_nn <- eval_metrics(pred_nn, test$target, "Neural Net")
print(confusionMatrix(pred_nn, test$target, positive = "1"))
```









```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 2300    0
##           1   50  540
##
##           Accuracy : 0.9827
##           95% CI : (0.9773, 0.9871)
##           No Information Rate : 0.8131
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.945
##
## Mcnemar's Test P-Value : 4.219e-12
##
##           Sensitivity : 1.0000
##           Specificity : 0.9787
##           Pos Pred Value : 0.9153
##           Neg Pred Value : 1.0000
##           Prevalence : 0.1869
##           Detection Rate : 0.1869
##           Detection Prevalence : 0.2042
##           Balanced Accuracy : 0.9894
##
##           'Positive' Class : 1
##
```

```
resultados[["NeuralNet"]] <- res_nn
```

Tabla comparativa de modelos por métricas

```
resumen <- do.call(rbind, resultados)
DT::datatable(resumen,
  caption = "Tabla Comparativa de Modelos (con características de external credit)",
  options = list(pageLength = 10, # Mostrar 10 filas por página
    dom = 'tip'))
```

Tabla Comparativa de Modelos (con características de external credit)

	Modelo 	Accuracy 	Kappa 	F1 	Precision 	Recall 
Regresión	Regresión	0.9927335640138408	0.9764403088497328	0.9809264305177112	0.9625668449197861	1
Logística	Logística					

	Modelo	Accuracy	Kappa	F1	Precision	Recall
	Árbol de Decisión	0.9913494809688581	0.9720313558501886	0.9773755656108597	0.9557522123893806	1
	XGBoost	0.9889273356401385	0.9643748651730409	0.9712230215827338	0.9440559440559441	1
	Random Forest	0.986159169550173	0.955715599141894	0.9642857142857143	0.9310344827586207	1
	NeuralNet	0.9826989619377162	0.9450256800456532	0.9557522123893806	0.9152542372881356	1

Showing 1 to 5 of 5 entries

Previous

1

Next

Todos los modelos tienen excelentes métricas, acertando el 100% de las predicciones para cero y fallando en un porcentaje mínimo para 1

Parte 4: Generación del Riesgo Score Final y Tasas de Interés

Calcularemos el score de riesgo y una tasa de interés dinámica (por persona), donde las personas de mayor riesgo tendrán una tasa de interés superior. Utilizaremos el modelo XGBoost

Comenzamos calculando el score de riesgo bajo score = bajo riesgo alto score = alto riesgo

```
# El riesgo_score será la probabilidad de incumplimiento (target=1)
features_for_prediction <- xgb_model$feature_names
# Filtrar df_final_model
data_for_prediction <- as.matrix(df_final_model %>% dplyr::select(all_of(features_for_prediction)))

probabilidades_final_score <- predict(xgb_model, newdata = xgb.DMatrix(data = data_for_prediction))
df_final_model$riesgo_score_final <- round(probabilidades_final_score, 4)
head(df_final_model)
```

```
## # A tibble: 6 × 27
##   customer_id loan_id ACC_CREATION_DATETIME APPLICATION_DATETIME
##         <dbl>   <dbl> <dtm>                <dtm>
## 1         1223     1 2021-08-23 08:57:56    2022-04-26 02:00:00
## 2          5190     2 2022-04-26 04:57:25    2022-04-26 02:00:00
## 3          5194     3 2022-04-26 07:22:35    2022-04-26 02:00:00
## 4          3978     4 2022-03-09 05:26:55    2022-04-26 02:00:00
## 5          4535     5 2022-04-01 08:28:42    2022-04-26 02:00:00
## 6          3604     6 2022-02-21 05:55:32    2022-05-05 02:00:00
## # i 23 more variables: LOAN_ORIGINATION_DATETIME <dtm>, max_days_late <dbl>,
## #   target <fct>, account_to_application_days <dbl>, n_sf_apps <dbl>,
## #   first_app_date <dtm>, last_app_date <dtm>, n_bnpl_apps <dbl>,
## #   n_bnpl_approved_apps <dbl>, first_bnpl_app_date <dtm>,
## #   last_bnpl_app_date <dtm>, n_inquiries_l3m <dbl>, n_inquiries_l6m <dbl>,
## #   riesgo_cat <fct>, mean_uso_credito_ext <dbl>, max_uso_credito_ext <dbl>,
## #   max_worst_delay_ext <dbl>, mean_delayed_payments_pct_ext <dbl>, ...
```

Seleccionamos los límites de las tasas de interés

Mínimo 10% = Para bajo riesgo Máximo 50% = Para alto riesgo

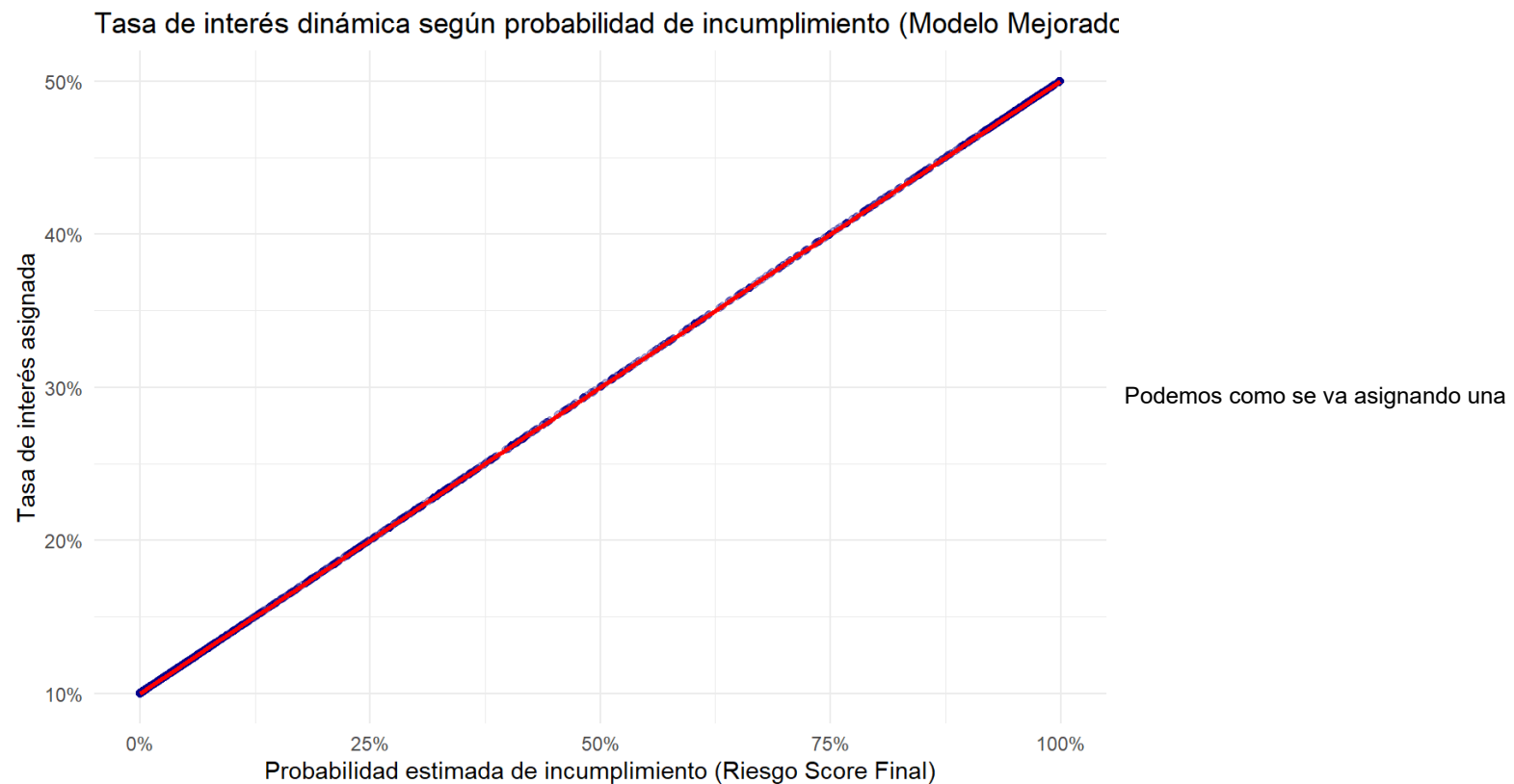
```
# Tasa de interés dinámica
tasa_base <- 0.10 # Mínimo 10%
tasa_max <- 0.50 # Máximo 50%

# Cálculo dinámico de la tasa individual basado en el riesgo_score_final (probabilidad de incumplimiento)
df_final_model <- df_final_model %>%
  mutate(tasa_interes_dinamica = tasa_base + riesgo_score_final * (tasa_max - tasa_base))
```

Graficamos

```
# Visualización: tasa vs probabilidad de riesgo
ggplot(df_final_model, aes(x = riesgo_score_final, y = tasa_interes_dinamica)) +
  geom_point(alpha = 0.3, color = "darkblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Tasa de interés dinámica según probabilidad de incumplimiento (Modelo Mejorado)",
       x = "Probabilidad estimada de incumplimiento (Riesgo Score Final)", y = "Tasa de interés asignada") +
  scale_y_continuous(labels = percent) +
  scale_x_continuous(labels = percent) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



tasa de interés de forma lineal dependiendo del score de riesgo

Creamos una tabla final con los ID correspondientes, el riesgo calculado con XGBoost y la tasa de interés dinámica asignada

```
# Tabla final
tabla <- df_final_model %>%
  dplyr::select(customer_id, loan_id, riesgo_score_final, tasa_interes_dinamica) %>%
  distinct() # Asegura que no haya duplicados de customer_id-loan_id

# Renombrar riesgo_score_final a riesgo_score como lo pide el entregable
tabla <- tabla %>%
  rename(riesgo_score = riesgo_score_final)

# Mostramos un ejemplo de tabla con las primeras 100 observaciones que otorga esa semilla
set.seed(42)
DT::datatable(tabla %>%
  head(100) %>%
  mutate(across(c(riesgo_score, tasa_interes_dinamica), ~ round(.x, 3))),
  caption = "Ejemplos de Riesgo Score y Tasa de Interés Dinámica por Préstamo (Árbol de Decisión)",
  options = list(pageLength = 10,
    dom = 'tip'))
```

Ejemplos de Riesgo Score y Tasa de Interés Dinámica por Préstamo (Árbol de Decisión)

	customer_id	loan_id	riesgo_score	tasa_interes_dinamica
1	1223	1	0.001	0.1
2	5190	2	0	0.1
3	5194	3	0	0.1
4	3978	4	0	0.1
5	4535	5	0	0.1
6	3604	6	0	0.1
7	271	7	0	0.1
8	5430	8	0	0.1
9	5128	9	0	0.1
10	4402	10	0	0.1

Showing 1 to 10 of 100 entries

Previous

1

2

3

4

5

...

10

Next

Conclusión:

En los modelos de aprendizaje automático se ve un error en la clasificación de las variables de tipo 1, mientras que para 0 prácticamente todos tienen el 100% de efectividad. Seleccionamos el modelo XGBoost para continuar con los cálculos y asignación de tasas de interés

Este modelo proporciona a Bankaya una herramienta analítica poderosa para automatizar y mejorar las decisiones crediticias. La capacidad de discernir el riesgo con mayor precisión se traducirá en:

-Reducción de Morosidad: Al identificar mejor a los clientes riesgosos, se pueden rechazar solicitudes o asignar tasas que compensen el riesgo. - Expansión Cautelosa del Mercado: Permite aprobar clientes con un riesgo moderado a una tasa adecuada, ampliando la base de clientes de forma controlada. -Mejora de la Experiencia del Cliente: Los clientes de bajo riesgo se benefician de tasas más bajas.