

Trabajo final: Credit Card Customer

Jun De Wu

15/02/2021

Contenidos

1	Contexto del problema y modelo de datos.	1
2	Análisis Exploratorio (EDA)	4
2.1	VARIABLES CUALITATIVAS	4
2.2	VARIABLES CUANTITATIVAS	11
2.3	VARIABLES CUALITATIVAS VS VARIABLES CUANTITATIVAS	14
2.4	TABLAS DE ESTADÍSTICOS	19
3	Machine Learning	24
3.1	MÉTODOS DE REGRESIÓN	25
3.2	MÉTODOS DE CLASIFICACIÓN	28
3.3	ACP, CLUSTERING Y OTROS ESTUDIOS CON ACP	37
3.4	MÉTODOS MÁS POTENTES	40

1 Contexto del problema y modelo de datos.

En esta primera parte nos centraremos en introducir nuestro data set y explicar las variables, limpiar nuestros datos y hacer un análisis breve del tipo de dato que mejor se ajusta a cada variable. En este caso, se ha escogido un data set sobre clientes de tarjetas de crédito (consulta en <https://www.kaggle.com/sakshigoyal7/credit-card-customers>) y el fichero que cargaremos se llama “BankChurners.csv”.

El conjunto de datos nos proporciona información sobre los clientes de un banco que más abajo procederemos a explicar cada variable. El objetivo más interesante en este conjunto de datos es predecir qué clientes van a dar de baja los servicios del banco, de esta forma el banco puede ponerse en contacto con ellos para hacer cambiar su decisión. Tenemos información cualitativa como la edad, el nivel de educación, ingresos anuales, etc, y también tenemos información cuantitativa como son el límite de crédito en la tarjeta, cantidad total de transacción, etc.

Para empezar, vamos a proceder con la carga de datos.

```
require(tidyverse)
require(patchwork)
require(ggpubr)
require(ggcrrplot)
require(modeest)
require(fastDummies)
require(caTools)
require(MLmetrics)
require(e1071)
require(caret)
```

```
require(cluster)
require(devtools)
require(ElemStatLearn)
require(ROSE)
require(class)
require(rpart)
require(rpart.plot)

bank <- read_csv("BankChurners.csv")
```

Vamos a quitar la primera variable **CLIENTNUM** y las dos últimas ya que no son de utilidad en nuestro estudio.

```
bank <- bank %>%
  dplyr::select(-c("CLIENTNUM", "Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_m")
glimpse(bank)
```

```

## Rows: 10,127
## Columns: 20
## $ Attrition_Flag <chr> "Existing Customer", "Existing Customer", ...
## $ Customer_Age <dbl> 45, 49, 51, 40, 40, 44, 51, 32, 37, 48, 42...
## $ Gender <chr> "M", "F", "M", "F", "M", "M", "M", "M", ...
## $ Dependent_count <dbl> 3, 5, 3, 4, 3, 2, 4, 0, 3, 2, 5, 1, 1, 3, ...
## $ Education_Level <chr> "High School", "Graduate", "Graduate", "Hi...
## $ Marital_Status <chr> "Married", "Single", "Married", "Unknown", ...
## $ Income_Category <chr> "$60K - $80K", "Less than $40K", "$80K - $...
## $ Card_Category <chr> "Blue", "Blue", "Blue", "Blue", "B...
## $ Months_on_book <dbl> 39, 44, 36, 34, 21, 36, 46, 27, 36, 36, 31...
## $ Total_Relationship_Count <dbl> 5, 6, 4, 3, 5, 3, 6, 2, 5, 6, 5, 6, 3, 5, ...
## $ Months_Inactive_12_mon <dbl> 1, 1, 1, 4, 1, 1, 1, 2, 2, 3, 3, 2, 6, 1, ...
## $ Contacts_Count_12_mon <dbl> 3, 2, 0, 1, 0, 2, 3, 2, 0, 3, 2, 3, 0, 3, ...
## $ Credit_Limit <dbl> 12691.0, 8256.0, 3418.0, 3313.0, 4716.0, 4...
## $ Total_Revolving_Bal <dbl> 777, 864, 0, 2517, 0, 1247, 2264, 1396, 25...
## $ Avg_Open_To_Buy <dbl> 11914.0, 7392.0, 3418.0, 796.0, 4716.0, 27...
## $ Total_Amt_Chng_Q4_Q1 <dbl> 1.335, 1.541, 2.594, 1.405, 2.175, 1.376, ...
## $ Total_Trans_Amt <dbl> 1144, 1291, 1887, 1171, 816, 1088, 1330, 1...
## $ Total_Trans_Ct <dbl> 42, 33, 20, 20, 28, 24, 31, 36, 24, 32, 42...
## $ Total_Ct_Chng_Q4_Q1 <dbl> 1.625, 3.714, 2.333, 2.333, 2.500, 0.846, ...
## $ Avg_Utilization_Ratio <dbl> 0.061, 0.105, 0.000, 0.760, 0.000, 0.311, ...

```

Antes de detallar las variables, vamos a comprobar si nuestro conjunto de datos contiene elementos NA.

`anyNA(bank)`

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

Por suerte, no tenemos NA's en nuestro data set. A continuación vamos a ver las variables que tenemos:

- **Attrition_Flag:** Actividad del cliente.
 - **Customer_Age:** Edad de los clientes en años.
 - **Gender:** Género del cliente.
 - **Dependent_count:** Número de dependientes.
 - **Education_Level:** Nivel de educación.
 - **Marital_Status:** Estado civil.
 - **Income_Category:** Ingreso anual.
 - **Card_Category:** Tipo de producto.
 - **Months_on_book:** Periodo de relación con el banco.
 - **Total_Relationship_Count:** Número total de productos manejados por el cliente.

- Months_Inactive_12_mon: Número de meses inactivos en los últimos 12 meses.
- Contacts_Count_12_mon: Número de contactos en los últimos 12 meses.
- Credit_Limit: Límite de crédito en la tarjeta.
- Total_Revolving_Bal: Saldo renovable total en la tarjeta de crédito.
- Avg_Open_To_Buy: Disposición a comprar línea de crédito (Promedio de los últimos 12 meses).
- Total_Amt_Chng_Q4_Q1: Cambio en la cantidad de transacción.
- Total_Trans_Amt: Cantidad total de transacción.
- Total_Trans_Ct: Recuento total de transacciones.
- Total_Ct_Chng_Q4_Q1: Cambio en el recuento de transacciones.
- Avg_Utilization_Ratio: Índice de utilización promedio de la tarjeta.

En total hay 10,127 observaciones y 20 variables. Como nos indica en la tabla de arriba, tenemos 6 variables cualitativas (en forma de character) y son Attrition_Flag, Gender, Education_Level, Marital_Status, Income_Category y Card_Category; por otra parte tenemos 14 variables cuantitativas (numéricas) compuestas por el resto de variables.

- Las variables cualitativas son todas categóricas, con lo cual nos interesa cambiar su tipo de character a factor.
- Las variables cuantitativas que tenemos son del tipo numérico, siendo correcta esta asignación, pero hay algunas variables que las pasamos a integer ya que son valores enteros. Estas variables son: Customer_Age, Dependent_count, Months_on_book, Total_Relationship_Count, Months_Inactive_12_mon, Contacts_Count_12_mon, Total_Trans_Amt y Total_Trans_Ct.

Vamos a cambiar el tipo de dato de las variables cualitativas, pasaremos de tener variables del tipo character a tipo factor; las variables cuantitativas que mencionamos anteriormente también serán cambiadas a tipo integer.

```
bank <- bank %>%
  mutate(Attrition_Flag = factor(Attrition_Flag), Gender = factor(Gender), Education_Level =
    factor(Education_Level, levels = c("Unknown", "Uneducated", "High School",
                                         "College", "Graduate", "Post-Graduate",
                                         "Doctorate")), Marital_Status =
    factor(Marital_Status), Income_Category = factor(Income_Category,
                                                   c("Unknown", "Less than $40K",
                                                 "$40K - $60K", "$60K - $80K",
                                                 "$80K - $120K", "$120K +")),
  Card_Category = factor(Card_Category), Customer_Age = as.integer(Customer_Age),
  Dependent_count = as.integer(Dependent_count), Months_on_book =
    as.integer(Months_on_book), Total_Relationship_Count =
    as.integer(Total_Relationship_Count), Months_Inactive_12_mon =
    as.integer(Months_Inactive_12_mon), Contacts_Count_12_mon =
    as.integer(Contacts_Count_12_mon), Total_Trans_Amt =
    as.integer(Total_Trans_Amt), Total_Trans_Ct = as.integer(Total_Trans_Ct))
```

A continuación vamos a validar de que nuestras variables para ver que todos los valores son correctos y están dentro del rango correspondiente.

```
tabla_rangos = data.frame(table(bank$Attrition_Flag == "Attrited Customer" |
  bank$Attrition_Flag == "Existing Customer"), table(bank$Customer_Age > 0),
  table(bank$Gender == "F" | bank$Gender == "M"), table(bank$Dependent_count >= 0),
  table(bank$Education_Level == "Unknown" | bank$Education_Level == "Uneducated" |
  bank$Education_Level == "High School" | bank$Education_Level == "College" |
  bank$Education_Level == "Graduate" | bank$Education_Level == "Post-Graduate" |
  bank$Education_Level == "Doctorate"),
  table(bank$Marital_Status == "Unknown" | bank$Marital_Status == "Married" |
  bank$Marital_Status == "Single" | bank$Marital_Status == "Divorced"),
  table(bank$Income_Category == "Unknown" | bank$Income_Category == "Less than $40K")
```

```

| bank$Income_Category == "$40K - $60K" | bank$Income_Category == "$60K - $80K" |
| bank$Income_Category == "$80K - $120K" | bank$Income_Category == "$120K +"),
table(bank$Card_Category == "Blue" | bank$Card_Category == "Gold",
| bank$Card_Category == "Silver" | bank$Card_Category == "Platinum"),
table(bank$Months_on_book > 0), table(bank$Total_Relationship_Count > 0),
table(bank$Months_Inactive_12_mon >= 0 & bank$Months_Inactive_12_mon <= 12),
table(bank$Contacts_Count_12_mon >= 0), table(bank$Credit_Limit >= 0),
table(bank$Total_Revolving_Bal >= 0), table(bank$Avg_Open_To_Buy >= 0),
table(bank$Total_Amt_Chng_Q4_Q1 >= 0), table(bank$Total_Trans_Amt >= 0),
table(bank$Total_Trans_Ct >= 0), table(bank$Total_Ct_Chng_Q4_Q1 >= 0),
table(bank$Avg_Utilization_Ratio >= 0)
)

glimpse(tabla_rangos[, sapply(tabla_rangos, is.numeric)])

```

```

## Rows: 1
## Columns: 20
## $ Freq      <int> 10127
## $ Freq.1    <int> 10127
## $ Freq.2    <int> 10127
## $ Freq.3    <int> 10127
## $ Freq.4    <int> 10127
## $ Freq.5    <int> 10127
## $ Freq.6    <int> 10127
## $ Freq.7    <int> 10127
## $ Freq.8    <int> 10127
## $ Freq.9    <int> 10127
## $ Freq.10   <int> 10127
## $ Freq.11   <int> 10127
## $ Freq.12   <int> 10127
## $ Freq.13   <int> 10127
## $ Freq.14   <int> 10127
## $ Freq.15   <int> 10127
## $ Freq.16   <int> 10127
## $ Freq.17   <int> 10127
## $ Freq.18   <int> 10127
## $ Freq.19   <int> 10127

```

Los valores numéricos nos indican las observaciones que cumplen con las condiciones que hemos aplicado. Vemos que todos los valores son igual al número total de observaciones que tenemos en el data set, con lo cual todas las variables están dentro de los rangos correspondientes, por lo que no tenemos observaciones atípicas.

2 Análisis Exploratorio (EDA)

2.1 Variables cualitativas

Vamos a empezar el análisis exploratorio con las variables cualitativas. Recordamos que tenemos `Attrition_Flag`, `Gender`, `Education_Level`, `Marital_Status`, `Income_Category` y `Card_Category`. Para empezar veamos la cantidad de clientes que siguen en el banco y la que se ha dado de baja.

```

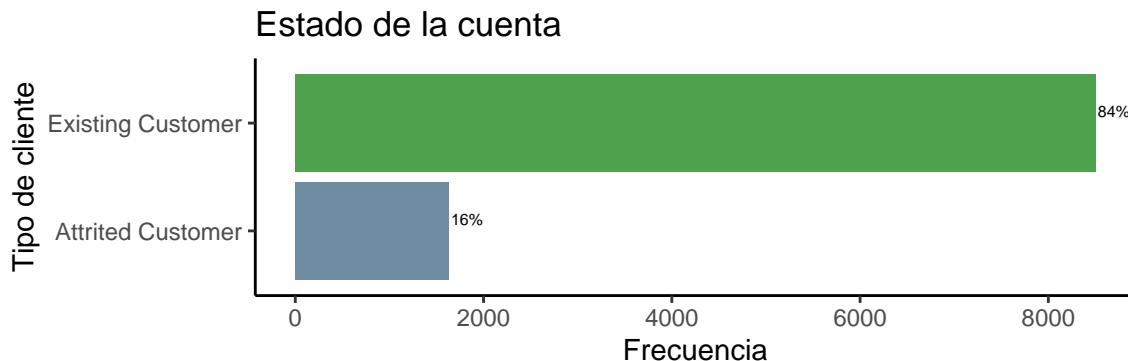
bank %>%
  ggplot(aes(x = Attrition_Flag, label = scales::percent(prop.table(stat(count)))) +
  geom_bar(fill = c("skyblue4", "forestgreen"), alpha = 0.8) +

```

```

geom_text(stat = 'count',
          position = position_dodge(.9),
          vjust = -0.5,
          hjust = -0.08,
          size = 2) +
  labs(x = "Tipo de cliente", y = "Frecuencia", title = "Estado de la cuenta") +
  coord_flip() +
  theme_classic()

```



El 16% de los clientes que tenemos que como observación se han dado de baja, mientras que el 84% siguen manteniendo la cuenta en el banco. En términos generales, hay pocos clientes que han prescindido de los servicios de la empresa, pero en términos relativos sigue siendo un porcentaje significativo que el banco puede estar interesado en intentar analizar las razones de su baja. A continuación analizamos las otras variables cualitativas para ver su influencia sobre la variable `Attrition_Flag`. Realizaremos contrastes de independencia entre `Attrition_Flag` y las otras variables cualitativas utilizando el test χ^2 de independencia. El contraste de independencia sirve para ver si dos criterios, X y Y para clasificar una población n son independientes entre ellos o no, obteniendo el siguiente contraste:

$$\begin{cases} H_0 : \text{Los criterios de clasificación } X \text{ e } Y \text{ son independientes} \\ H_1 : \text{Los criterios de clasificación } X \text{ e } Y \text{ no son independientes} \end{cases}$$

A modo de ejemplo, vemos cómo se realiza el contraste para `Attrition_Flag` y `Gender`, los otros se hacen de forma parecida. De esta forma, nuestro contraste a realizar es el siguiente:

$$\begin{cases} H_0 : \text{Los criterios de clasificación Attrition Flag y Gender son independientes} \\ H_1 : \text{Los criterios de clasificación Attrition Flag y Gender no son independientes} \end{cases}$$

Empezamos creando la tabla de contingencia que nos da la relación entre estas dos variables.

```

tabla_contingencia_gender <- table(bank$Attrition_Flag, bank$Gender)
tabla_contingencia_gender

```

```

##
##           F     M
##   Attrited Customer 930  697
##   Existing Customer 4428 4072

```

A partir de esta, creamos las frecuencias marginales:

```

tabla_marginal_gender <- addmargins(tabla_contingencia_gender)
tabla_marginal_gender

```

```

##          F      M   Sum
## Attrited Customer  930   697 1627
## Existing Customer 4428  4072 8500
## Sum                  5358  4769 10127

```

También tenemos que crear la tabla de las frecuencias esperadas ya que nos tenemos que asegurar que ningún valor de la tabla sea menor que 5 para poder realizar correctamente el test χ^2 .

```

tabla_frec Esperadas_gender <-
  rowSums(tabla_contingencia_gender) %*% t(colSums(tabla_contingencia_gender)) / sum(tabla_contingencia_gen
tabla_frec Esperadas_gender

##          F      M
## [1,] 860.8143 766.1857
## [2,] 4497.1857 4002.8143

```

Ningún valor es menor que 5, con lo cual podemos proceder con el test.

```
chisq.test(tabla_contingencia_gender)
```

```

## 
## Pearson's Chi-squared test with Yates' continuity correction
##
## data: tabla_contingencia_gender
## X-squared = 13.866, df = 1, p-value = 0.0001964

```

El p-valor es suficientemente pequeño para rechazar la hipótesis nula y concluimos que el tipo de cliente y su edad no son independientes.

Para las otras variables cualitativas hacemos el mismo contraste con respecto a `Attrition_Flag`.

`Education_Level`:

```

tabla_contingencia_education <- table(bank$Attrition_Flag, bank$Education_Level)
tabla_marginal_education <- addmargins(tabla_contingencia_education)
tabla_frec Esperadas_education <-
  rowSums(tabla_contingencia_education) %*% t(colSums(tabla_contingencia_education)) / sum(tabla_contingencia_
tabla_frec Esperadas_education

##          Unknown Uneducated High School College Graduate Post-Graduate Doctorate
## [1,]    244.042     238.9009    323.4078 162.7482   502.5433     82.90037   72.45749
## [2,]   1274.958    1248.0991   1689.5922 850.2518  2625.4567    433.09963  378.54251
chisq.test(tabla_contingencia_education)

```

```

## 
## Pearson's Chi-squared test
##
## data: tabla_contingencia_education
## X-squared = 12.511, df = 6, p-value = 0.05149

```

Con un nivel de significación $\alpha = 5\%$ aceptamos la hipótesis nula, `Education_Level` y `Attrition_Flag` son independientes.

`Marital_Status`:

```

tabla_contingencia_marital <- table(bank$Attrition_Flag, bank$Marital_Status)
tabla_marginal_marital <- addmargins(tabla_contingencia_marital)
tabla_frec Esperadas_marital <-

```

```

rowSums(tabla_contingencia_marital) %*% t(colSums(tabla_contingencia_marital)) / sum(tabla_contingencia_marital)

##      Divorced   Married   Single Unknown
## [1,] 120.1734 753.0117 633.4809 120.3341
## [2,] 627.8266 3933.9883 3309.5191 628.6659

chisq.test(tabla_contingencia_marital)

##
## Pearson's Chi-squared test
##
## data: tabla_contingencia_marital
## X-squared = 6.0561, df = 3, p-value = 0.1089

```

Con un nivel de significación $\alpha = 5\%$ aceptamos la hipótesis nula, Marital_Status y Attrition_Flag son independientes.

Income_Category:

```

tabla_contingencia_income <- table(bank$Attrition_Flag, bank$Income_Category)
tabla_marginal_income <- addmargins(tabla_contingencia_income)
tabla_frec Esperadas_income <-
  rowSums(tabla_contingencia_income) %*% t(colSums(tabla_contingencia_income)) / sum(tabla_contingencia_income)

##      Unknown Less than $40K $40K - $60K $60K - $80K $80K - $120K $120K +
## [1,] 178.6535      572.1089    287.5807    225.2448    246.6125 116.7995
## [2,] 933.3465     2988.8911   1502.4193   1176.7552   1288.3875 610.2005

chisq.test(tabla_contingencia_income)

```

```

##
## Pearson's Chi-squared test
##
## data: tabla_contingencia_income
## X-squared = 12.832, df = 5, p-value = 0.025

```

Con un nivel de significación $\alpha = 5\%$ rechazamos la hipótesis nula, Income_Category y Attrition_Flag no son independientes.

Card_Category

```

tabla_contingencia_card <- table(bank$Attrition_Flag, bank$Card_Category)
tabla_marginal_card <- addmargins(tabla_contingencia_card)
tabla_frec Esperadas_card <-
  rowSums(tabla_contingencia_card) %*% t(colSums(tabla_contingencia_card)) / sum(tabla_contingencia_card)

##      Blue Gold Platinum Silver
## [1,] 1515.984 18.63652 3.213192 89.16609
## [2,] 7920.016 97.36348 16.786808 465.83391

```

```

chisq.test(tabla_contingencia_card)

## Warning in chisq.test(tabla_contingencia_card): Chi-squared approximation may be
## incorrect

##
## Pearson's Chi-squared test

```

```

## 
## data: tabla_contingencia_card
## X-squared = 2.2342, df = 3, p-value = 0.5252

```

Tenemos una frecuencia esperada de la tabla menor que 5, el test χ^2 aproxima de forma incorrecta. Para arreglar esto, simularemos el valor del p-valor con 5000 replicaciones.

```

set.seed(NULL)
chisq.test(tabla_contingencia_card,simulate.p.value = TRUE, B=5000)

```

```

## 
## Pearson's Chi-squared test with simulated p-value (based on 5000
## replicates)
## 
## data: tabla_contingencia_card
## X-squared = 2.2342, df = NA, p-value = 0.5207

```

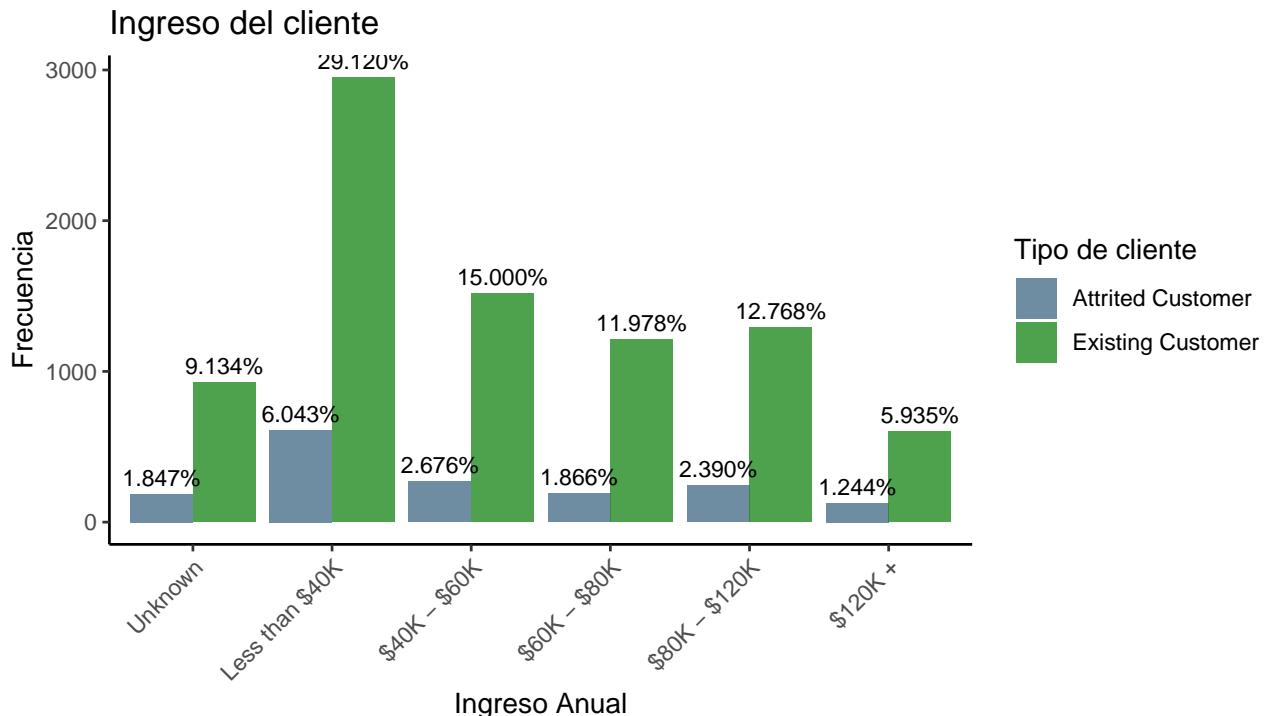
El p-valor es suficientemente grande como para aceptar la hipótesis nula, Attrition_Flag y Card_Category son independientes.

Después de realizar todos estos contrastes hemos extraído que las variables que no son independientes de Attrition_Flag son Gender e Income_Category. Cogeremos estas dos variables para realizar un análisis sobre su relación con la variable del tipo de cliente.

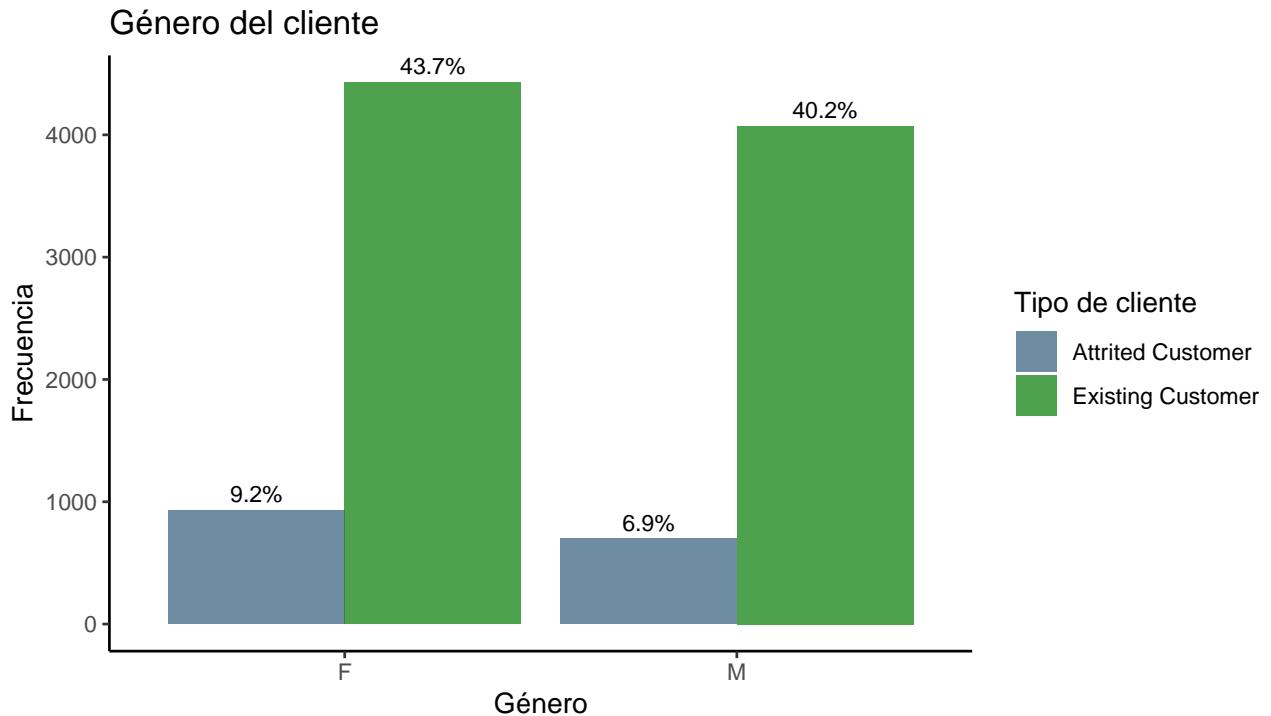
```

bank %>%
  ggplot(aes(x = Income_Category, fill = Attrition_Flag, label =
    scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Tipo de cliente", values = c("Attrited Customer" = "skyblue4",
                                                 "Existing Customer" = "forestgreen")) +
  labs(x = "Ingreso Anual", y = "Frecuencia", title = "Ingreso del cliente") +
  scale_x_discrete(guide = guide_axis(angle = 45)) +
  theme_classic()

```



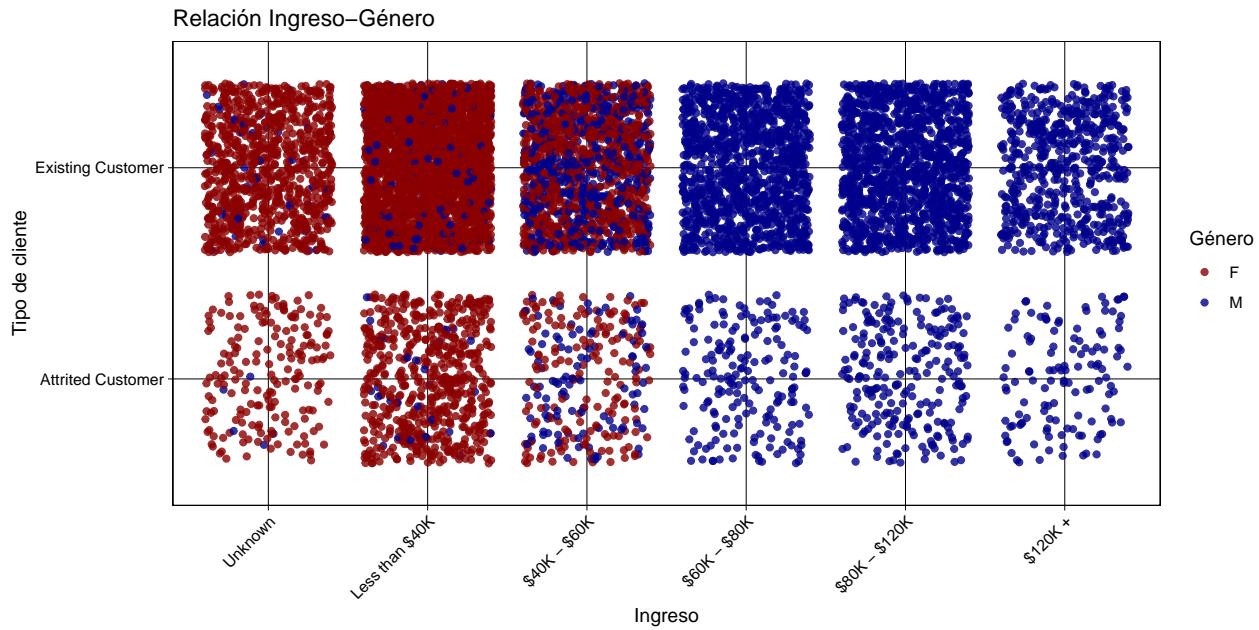
```
bank %>%
  ggplot(aes(x = Gender, fill = Attrition_Flag, label =
    scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge", alpha = 0.8) +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_fill_manual("Tipo de cliente", values = c("Attrited Customer" = "skyblue4",
                                                 "Existing Customer" = "forestgreen")) +
  labs(x = "Género", y = "Frecuencia", title = "Género del cliente") +
  theme_classic()
```



Los clientes que más abundan son los de ingresos menores que 40,000\$, siendo también la categoría con más bajas con un 6.043%. A nivel global, hay pocos que clientes que se dan de baja con otro tipo de ingreso. Respecto al género del cliente, vemos que un 52.9% de los clientes son mujeres mientras que el otro 47.1% son hombres. Hay un mayor número de clientes mujeres que abandonan los servicios del banco (9.2% frente a un 6.9%), aunque las diferencias entre los dos géneros son insignificantes a simple vista.

Realizamos un análisis visual de la relación entre el ingreso y el género del cliente con respecto a la anulación del servicio de tarjetas con un geom_jitter.

```
bank %>%
  ggplot(aes(x = Income_Category, y = Attrition_Flag)) +
  geom_jitter(aes(color = Gender), alpha = 0.8) +
  scale_color_manual("Género", values = c("F" = "red4", "M" = "blue4")) +
  labs(x = "Ingreso", y = "Tipo de cliente", title = "Relación Ingreso-Género") +
  theme_linedraw() +
  scale_x_discrete(guide = guide_axis(angle = 45))
```



Se puede observar que hay una clara diferenciación en los ingresos entre las mujeres y los hombres. Parece ser que el 100% de las mujeres se sitúan en ingresos menores de 60,000\$ anuales mientras que todos los clientes que tienen ingresos “mayores” son hombres. En el grupo de ingresos con mayor abandono de los servicios se sitúa en los ingresos 40,000\$, siendo en su mayoría mujeres.

2.2 Variables cuantitativas

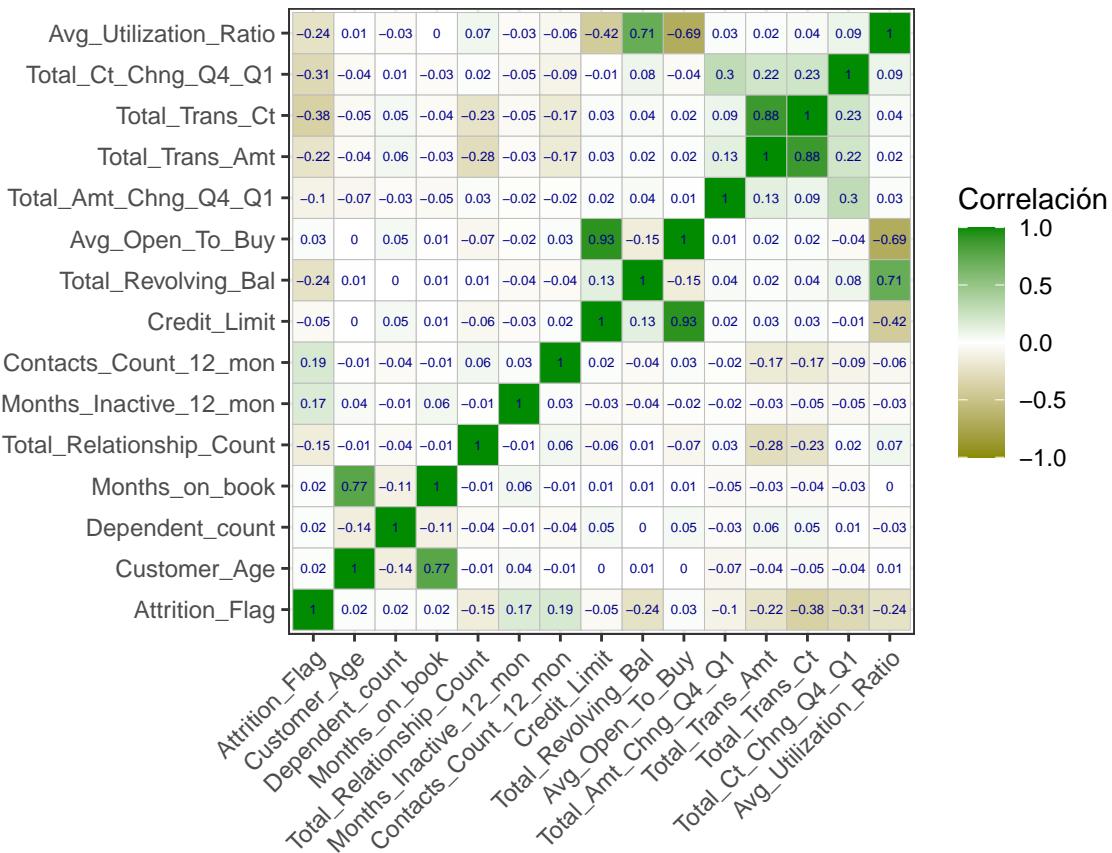
En total hay 14 variables cuantitativas y no vamos a realizar un análisis exhaustivo de cada una, con lo cual vamos a elegir las que consideremos que son las más importantes en relación a la fuga de clientes. Para ello, vamos a graficar la matriz de correlación de las variables cuantitativas junto con `Attrition_Flag`, pero antes tenemos que pasar esta variable a numérica.

```
bank_correlacion <- bank %>%
  mutate(Attrition_Flag = as.integer(recode(Attrition_Flag, "Existing Customer" = 0,
                                             "Attrited Customer" = 1)))

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

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

Matriz de correlación



Primero observamos que las variables `Avg_Open_To_Buy` y `Credit_Limit` tienen correlación 0.93, `Total_Trans_Ct` y `Total_Trans_Amt` tienen correlación 0.88, `Total_Revolving_Bal` y `Avg_Utilization_Ratio` tienen correlación 0.71 y `Customer_Age` y `Months_on_book` tienen correlación 0.77, con lo cual estas variables tienen correlación perfecta y lo tendremos en cuenta a la hora de realizar los modelos ya que tendremos que eliminar una de ellas en cada caso. En lo que respecta a `Attrition_Flag`, no hay grandes correlaciones pero las 4 más significativas son con `Total_Revolving_Bal` (Saldo renovable total en la tarjeta de crédito), `Total_Trans_Ct` (Recuento total de transacciones), `Total_Ct_Chng_Q4_Q1` (Cambio en el recuento de transacciones) y `Avg_Utilization_Ratio` (Índice de utilización promedio de la tarjeta).

Estas variables serán las que analizaremos mediante gráficos. Dibujaremos un histograma de cada variable para ilustrar mejor la distribución de éstas respecto al tipo de cliente.

```
p1 <- bank %>%
  ggplot(aes(x = Total_Revolving_Bal, fill = Attrition_Flag)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Tipo de cliente", values = c("Attrited Customer" = "gold4",
                                                 "Existing Customer" = "red4")) +
  labs(x = "Saldo renovable total", y = "Frecuencia",
       title = "Distribución del saldo renovable total") +
  theme_classic()

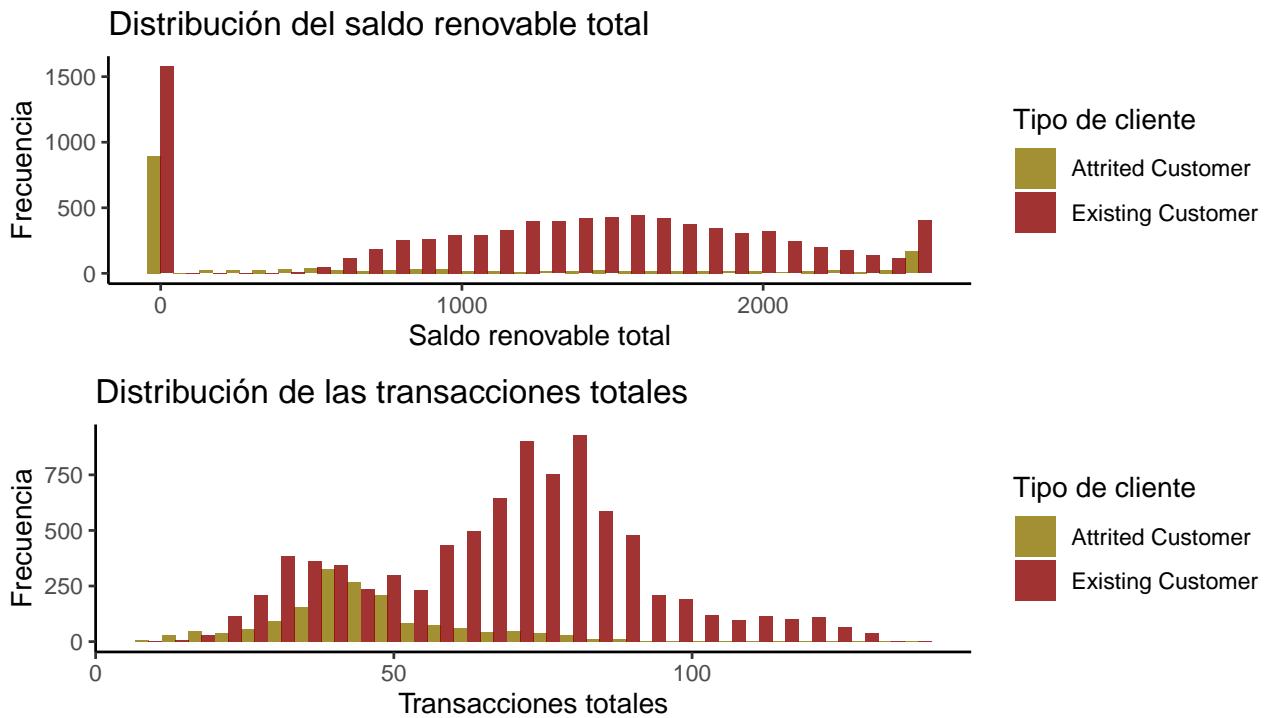
p2 <- bank %>%
  ggplot(aes(x = Total_Trans_Ct, fill = Attrition_Flag)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Tipo de cliente", values = c("Attrited Customer" = "gold4",
```

```

    "Existing Customer" = "red4")) +
  labs(x = "Transacciones totales", y = "Frecuencia",
       title = "Distribución de las transacciones totales") +
  theme_classic()

ggarrange(p1, p2, nrow = 2, ncol = 1)

```



En el histograma de Total_Revolving_Bal hay una gran cantidad de clientes que tienen un saldo renovable bajo, apilándose muchos de ellos en 0 saldo renovable total. Además, se puede apreciar cómo los clientes que se han dado de baja tienen un saldo renovable total muy bajo a lo largo de todo el histograma, como una posible condición de aquellos que abandonan el banco. En lo que se refiere a las transacciones totales, se aprecia claramente que los clientes que se dan de baja realizan muchísimas menos transacciones que los clientes que siguen en el banco.

```

p3 <- bank %>%
  ggplot(aes(x = Total_Ct_Chng_Q4_Q1, fill = Attrition_Flag)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Tipo de cliente", values = c("Attrited Customer" = "gold4",
                                                 "Existing Customer" = "red4")) +
  labs(x = "Cambio en las transacciones", y = "Frecuencia",
       title = "Distribución del cambio en las transacciones") +
  theme_classic()

p4 <- bank %>%
  ggplot(aes(x = Avg_Utilization_Ratio, fill = Attrition_Flag)) +
  geom_histogram(position = "dodge", alpha = 0.8) +
  scale_fill_manual("Tipo de cliente", values = c("Attrited Customer" = "gold4",
                                                 "Existing Customer" = "red4")) +
  labs(x = "Uso promedio de la tarjeta", y = "Frecuencia",
       title = "Distribución del uso promedio de la tarjeta") +

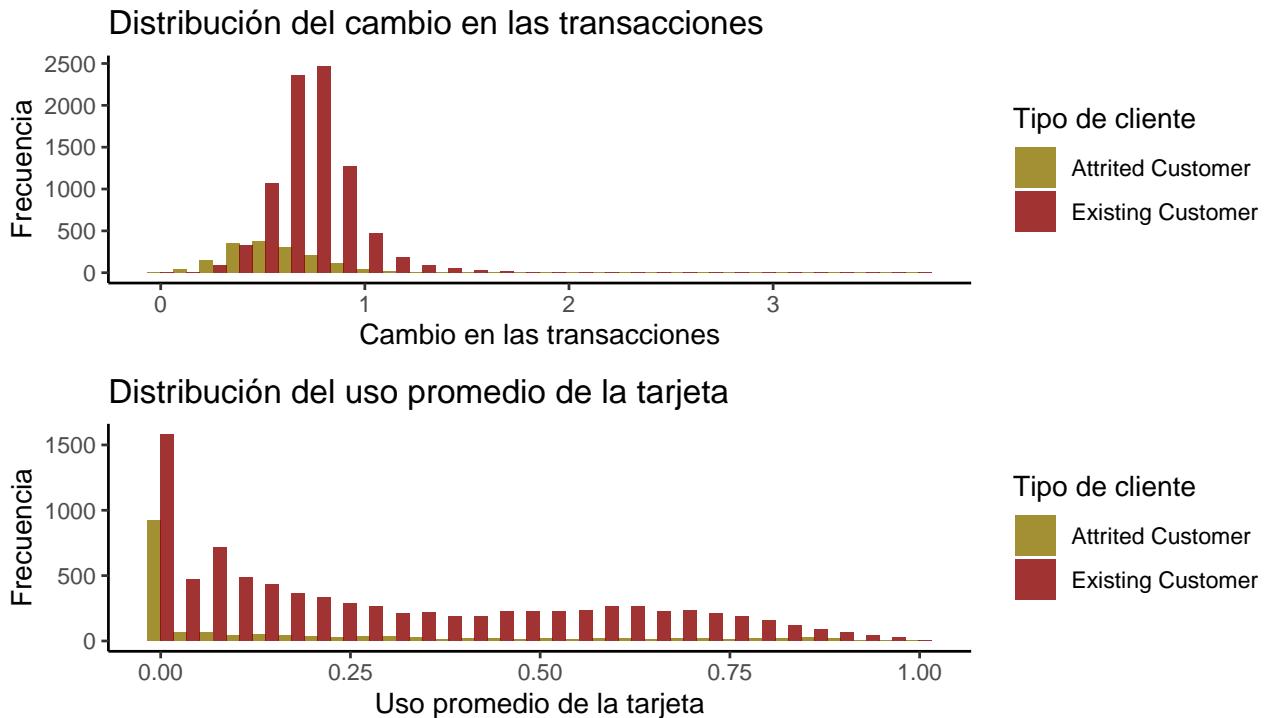
```

```

theme_classic()

ggarrange(p3, p4, nrow = 2, ncol = 1)

```



El cambio en las transacciones sigue predominando los clientes que siguen, con un bajo aporte del resto. En el uso promedio de la tarjeta sigue un poco como el saldo renovable total, clara dominancia de los clientes fieles y en consecuencia, más uso de las tarjetas de éstos respecto a los que se van.

2.3 Variables cualitativas vs Variables cuantitativas

Una vez que hayamos escogido las variables cualitativas y cuantitativas más influyentes sobre nuestra variable `Attrition_Flag`, es hora de ver el comportamiento de éstos de forma conjunta. Para ello, vamos a comparar las variables cualitativas que hemos escogido anteriormente con las variables cuantitativas que más correlación tienen con `Attrition_Flag`. Utilizamos diagramas de violín con su correspondiente diagrama de caja dentro, siempre separándolos entre las dos categorías de `Attrition_Flag`, añadiendo además la media de cada nivel de las variables cualitativas respecto a la variable cuantitativa que estamos estudiando. Después de graficar estos diagramas, realizaremos un ANOVA de un factor para determinar si las medias de las variables cuantitativas para cada categoría de las variables cualitativas son iguales.

Empezaremos con la variable de saldo renovable total respecto al ingreso:

```

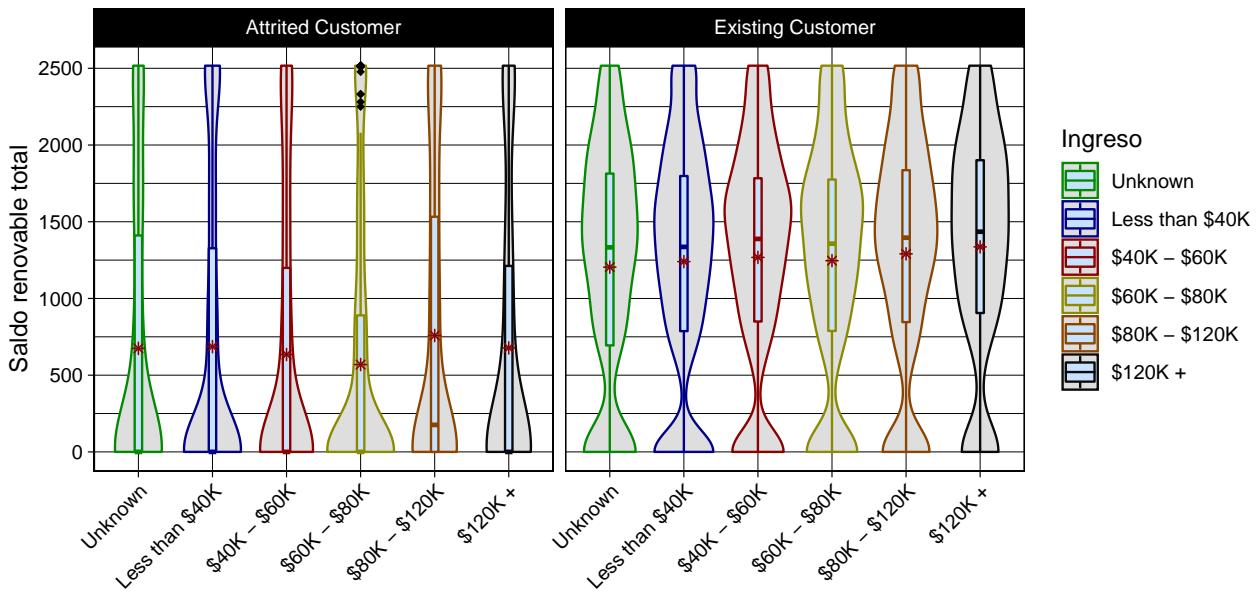
bank %>%
  ggplot(aes(x = Income_Category, y = Total_Revolving_Bal, color = Income_Category)) +
  geom_violin(fill = "gray87") +
  geom_boxplot(width = 0.1, fill = "slategray1", outlier.colour = "black",
               outlier.size = 1.5, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Ingreso", values =
    c("green4", "blue4", "red4", "yellow4", "darkorange4", "grey4")) +
  theme_linedraw() +

```

```

  labs(x = "", y = "Saldo renovable total") +
  facet_wrap(~Attrition_Flag) +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

```



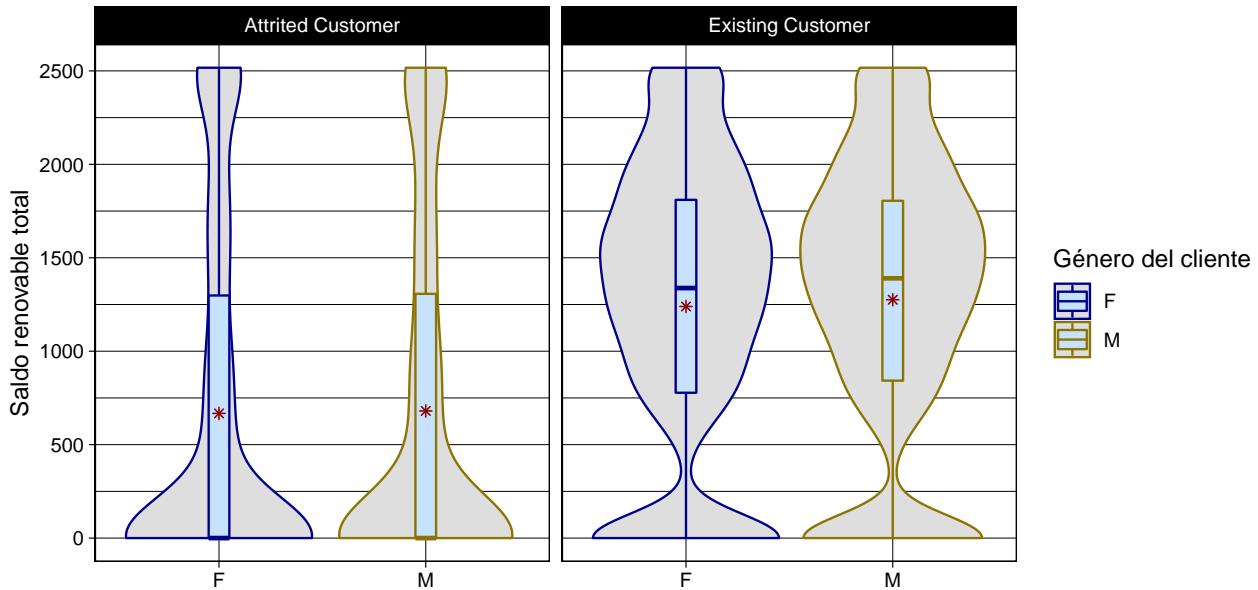
Los clientes que se van tienen claramente un saldo renovable menor que los que se quedan. A excepción de los clientes que se van con ingreso entre 80,000 y 120,000 dólares, el resto tienen las medianas y las medias parecidas. De forma más homogénea se comportan las medias y las medianas para los clientes que se quedan.

Seguimos con los diagramas del saldo renovable respecto al sexo del cliente:

```

bank %>%
  ggplot(aes(x = Gender, y = Total_Revolving_Bal, color = Gender)) +
  geom_violin(fill = "gray87") +
  geom_boxplot(width = 0.1, fill = "slategray1", outlier.colour = "black",
               outlier.size = 1.5, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Género del cliente", values = c("blue4", "gold4")) +
  theme_linedraw() +
  labs(x = "", y = "Saldo renovable total") +
  facet_wrap(~Attrition_Flag)

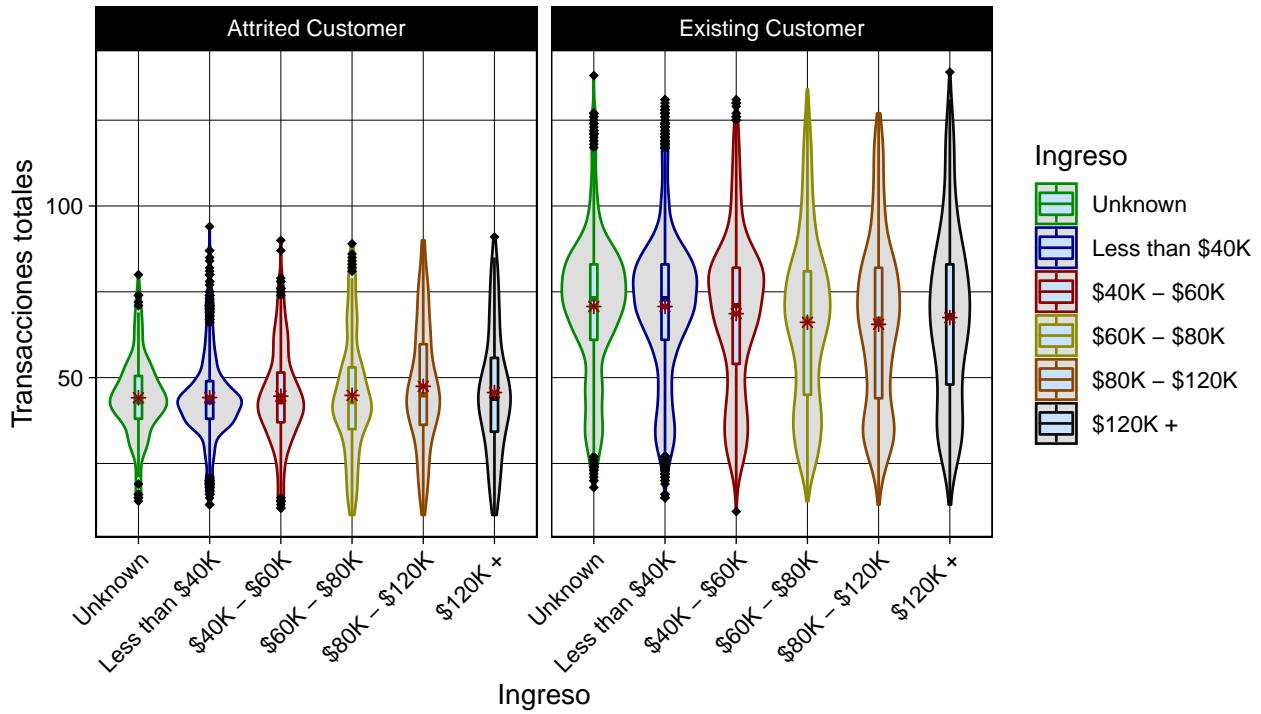
```



El comportamiento del saldo renovable entre las mujeres y los hombres son prácticamente idénticas, sin ninguna diferencia clara.

Continuamos con las transacciones totales y el ingreso:

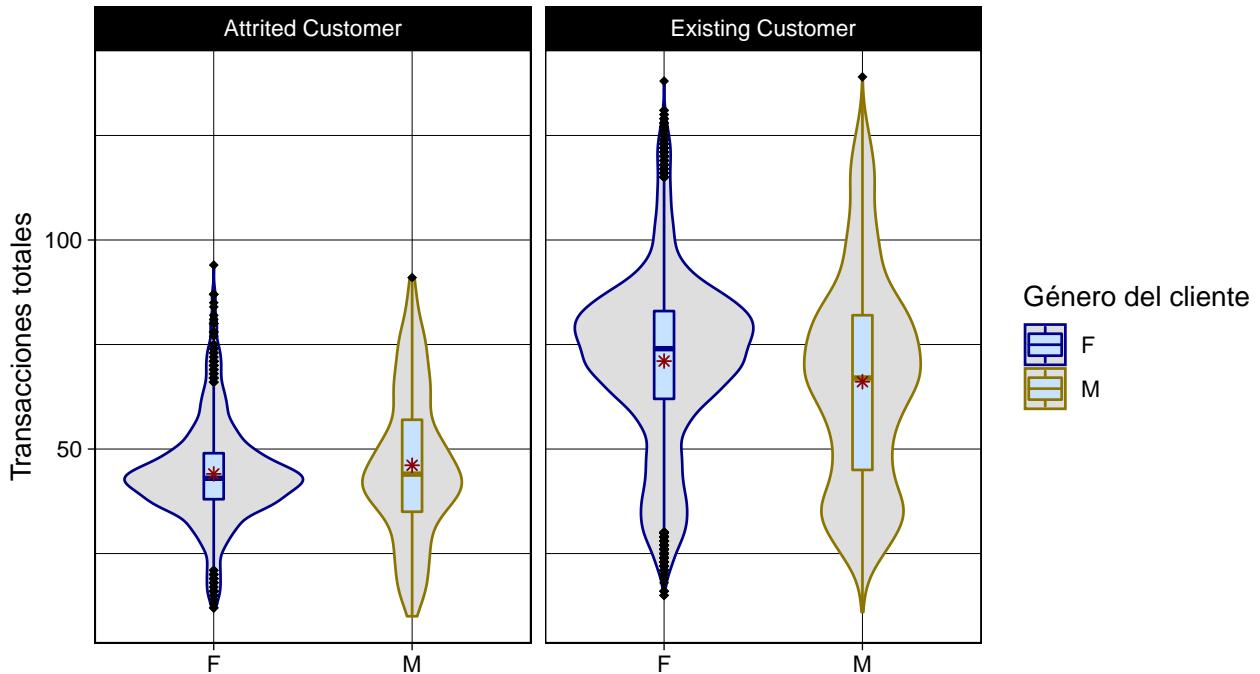
```
bank %>%
  ggplot(aes(x = Income_Category, y = Total_Trans_Ct, color = Income_Category)) +
  geom_violin(fill = "gray87") +
  geom_boxplot(width = 0.1, fill = "slategray1", outlier.colour = "black",
               outlier.size = 1.5, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5, color ="red4") +
  scale_color_manual("Ingreso", values =
    c("green4", "blue4", "red4","yellow4","darkorange4","grey4")) +
  theme_linedraw() +
  facet_wrap(~Attrition_Flag) +
  labs(x = "Ingreso", y = "Transacciones totales") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



De manera similar que con los casos anteriores, exceptuando la diferencia entre los clientes que se van y los que no, no se aprecia apenas diferencia entre los niveles de ingreso respecto a las transacciones totales. Sí que podemos observar que hay una gran cantidad de outliers en este caso. Esta cantidad de valores atípicos puede ser debido a que el número de transacciones que realizar una persona puede diferir mucho entre usuarios, además de que los clientes con ingresos por debajo de 60,000 dólares tienden a tener más comportamientos atípicos que los otros.

Por último, tenemos los diagramas entre las transacciones totales y el género:

```
bank %>%
  ggplot(aes(x = Gender, y = Total_Trans_Ct, color = Gender)) +
  geom_violin(fill = "gray87") +
  geom_boxplot(width = 0.1, fill = "slategray1", outlier.colour = "black",
              outlier.size = 1.5, outlier.shape = 18) +
  stat_summary(fun.y = mean, geom = "point", shape = 8, size = 1.5,
              color = "red4", show_guide = FALSE) +
  scale_color_manual("Género del cliente", values = c("blue4", "gold4")) +
  facet_wrap(~Attrition_Flag) +
  theme_linedraw() +
  facet_wrap(~Attrition_Flag) +
  labs(x = "", y = "Transacciones totales")
```



Parece que puede haber una pequeña diferencia entre las medias de las transacciones en cada sexo. Como comentábamos anteriormente, la cantidad de outliers en este caso puede ser debido a la gran diferencia que hay en las transacciones realizadas entre diferentes clientes. En este caso en particular, vemos que los outliers se concentran exclusivamente (exceptuando, como parece indicar los diagramas, uno o dos outliers en los hombres) en las mujeres. No es fácil saber la causa sin entrar en cuestiones sexistas, pero sí que puede haber una mayor variabilidad entre las mujeres en su transacciones totales debido a un mayor uso de las tarjetas.

Como hemos visto, parece ser que puede haber una diferencia en las medias de `Total_Trans_Ct` en los diferentes niveles de `Income_Category` y `Gender`, vamos a realizar un ANOVA de un factor para realizar el contraste pertinente.

```
bank_attrited <- bank %>%
  filter(Attrition_Flag == "Attrited Customer")

bank_existing <- bank %>%
  filter(Attrition_Flag == "Existing Customer")
```

Empezamos a ver si las medias de las transacciones totales para cada nivel de ingreso son iguales o no:

```
summary(aov(bank_attrited$Total_Trans_Ct ~ bank_attrited$Income_Category))
```

```
##                                     Df Sum Sq Mean Sq F value Pr(>F)
## bank_attrited$Income_Category     5   2102   420.4   1.987 0.0777 .
## Residuals                         1621 342999   211.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Obtenemos un p-valor de 0.0777, con lo cual aceptamos con un nivel de significación de $\alpha = 5\%$ la hipótesis nula de que las medias son iguales.

De la misma forma realizamos los otros tests.

```
summary(aov(bank_existing$Total_Trans_Ct ~ bank_existing$Income_Category))
```

```
##                                     Df Sum Sq Mean Sq F value    Pr(>F)
## bank_existing$Income_Category     5   37550   7510   14.41 4.41e-14 ***
## Residuals                         1621 342999   211.6
```

```

## Residuals           8494 4426814      521
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
summary(aov(bank_attrited$Total_Trans_Ct~bank_attrited$Gender))

##                               Df Sum Sq Mean Sq F value Pr(>F)
## bank_attrited$Gender     1   1689   1688.8   7.991 0.00476 **
## Residuals                 1625 343412    211.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
summary(aov(bank_existing$Total_Trans_Ct~bank_existing$Gender))

##                               Df Sum Sq Mean Sq F value Pr(>F)
## bank_existing$Gender     1   51635   51635   99.44 <2e-16 ***
## Residuals                 8498 4412729      519
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Obtenemos que las medias de las transacciones totales entre las diferentes categorías de `Gender` para los dos tipos de clientes no son iguales; de la misma forma tenemos que las medias de las transacciones totales de los diferentes niveles de ingreso de los clientes que se quedan tampoco son iguales.

2.4 Tablas de estadísticos

Otra de las cosas que podemos hacer es calcular los estadísticos más importantes para las dos variables cuantitativas que hemos utilizado en el apartado del análisis exploratorio respecto a las variables cualitativas `Gender` e `Income_Category`, siempre separando por si el cliente sigue utilizando los servicios del banco o no. Los estadísticos empleados en cada caso son la media, la desviación típica, la mediana y la moda.

Creamos primero las dos tablas de los estadísticos del saldo renovable total respecto al ingreso y al sexo.

```

tabla_1 <- bank %>%
  group_by(Attrition_Flag, Income_Category) %>%
  summarise(Media = mean(Total_Revolving_Bal),
            Desv_tip = sd(Total_Revolving_Bal),
            Mediana = median(Total_Revolving_Bal),
            Moda = mlv(Total_Revolving_Bal, method = "mfv")[1])

table_2 <- bank %>%
  group_by(Attrition_Flag, Gender) %>%
  summarise(Media = mean(Total_Revolving_Bal),
            Desv_tip = sd(Total_Revolving_Bal),
            Mediana = median(Total_Revolving_Bal),
            Moda = mlv(Total_Revolving_Bal, method = "mfv")[1])

```

Viasualizamos la tabla respecto al ingreso, graficando los estadísticos.

```
tabla_1
```

```

## # A tibble: 12 x 6
## # Groups:   Attrition_Flag [2]
##   Attrition_Flag Income_Category Media Desv_tip Mediana Moda
##   <fct>          <fct>        <dbl>    <dbl>    <dbl> <dbl>
## 1 Attrited Customer Unknown       675.     917.      0     0

```

```

## 2 Attrited Customer Less than $40K    686.    943.      0      0
## 3 Attrited Customer $40K - $60K       635.    900.      0      0
## 4 Attrited Customer $60K - $80K       569.    849.      0      0
## 5 Attrited Customer $80K - $120K      759.    943.    176      0
## 6 Attrited Customer $120K +           678.    932.      0      0
## 7 Existing Customer Unknown          1204.    781.   1333      0
## 8 Existing Customer Less than $40K   1240.    762.   1336      0
## 9 Existing Customer $40K - $60K     1268.    738.   1388      0
## 10 Existing Customer $60K - $80K    1246.    759.   1357      0
## 11 Existing Customer $80K - $120K   1291.    754.   1396      0
## 12 Existing Customer $120K +        1336.    745.   1435      0

```

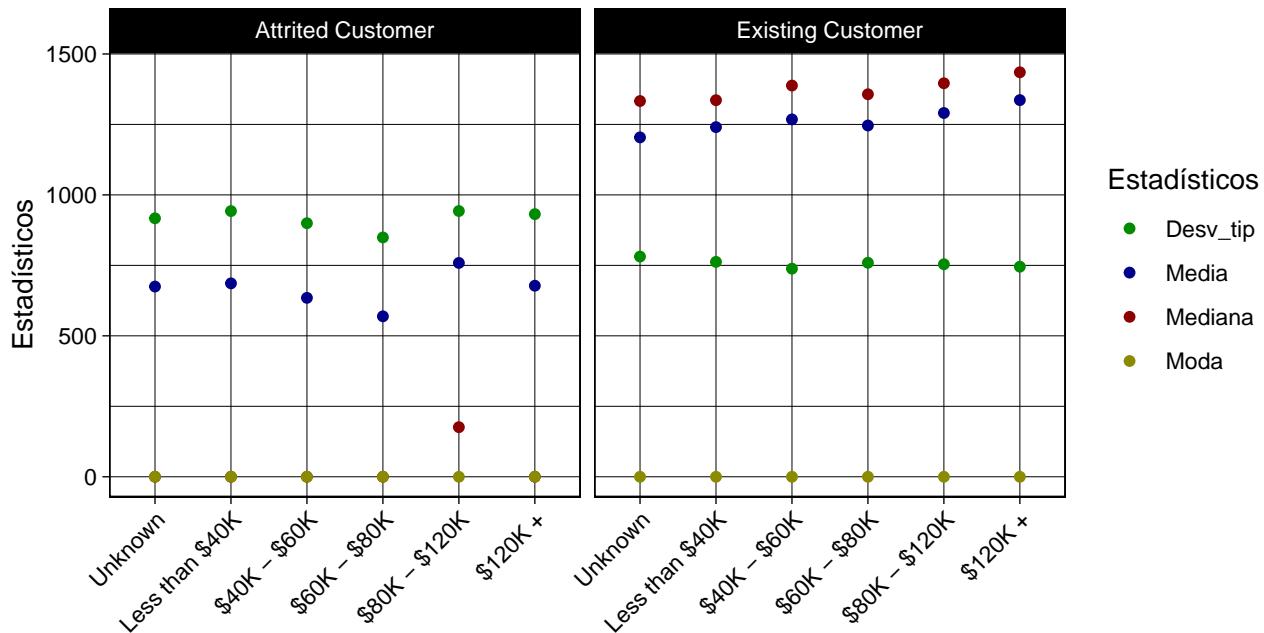
tabla_1 %>%

```

gather(Estadisticos, Valor, -1, -2) %>%
ggplot(aes(x = Income_Category, y = Valor)) +
geom_point(aes(color = Estdadisticos)) +
facet_wrap(~Attrition_Flag) +
theme_linedraw() +
scale_color_manual("Estdadisticos", values = c("green4", "blue4",
                                              "red4", "yellow4")) +
labs(x = "", y = "Estdadisticos", title =
      "Estdadisticos del saldo renovable por ingreso") +
theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

```

Estadísticos del saldo renovable por ingreso



Tanto para los clientes que se van como para los que se quedan, la moda es 0 en los diferentes niveles de ingreso. Podemos observar los diferentes estadísticos en ambos casos, pero el hecho más curioso puede ser que las medianas para cada nivel de ingreso en los clientes que se van son 0 exceptuando para los que tienen ingreso entre 80,000 y 120,000 dólares, indicando poco rango de oscilación del saldo renovable total en este grupo. La media es más alta en todas las niveles de ingreso en los que se quedan, de la misma forma que las medianas indicando un mayor rango en esta categoría.

```

table_2

## # A tibble: 4 x 6
## # Groups: Attrition_Flag [2]
##   Attrition_Flag   Gender Media Desv_tip Mediana Moda
##   <fct>          <fct>  <dbl>    <dbl>    <dbl>  <dbl>
## 1 Attrited Customer F      667.     925.      0       0
## 2 Attrited Customer M      680.     918.      0       0
## 3 Existing Customer F     1239.    766.    1338      0
## 4 Existing Customer M     1275.    749.    1390      0

table_2 %>%
  gather(Estadisticos, Valor, -1, -2) %>%
  ggplot(aes(x = Gender, y = Valor)) +
  geom_point(aes(color = Estdadisticos)) +
  facet_wrap(~Attrition_Flag) +
  theme_linedraw() +
  scale_color_manual("Estdadisticos", values = c("green4", "blue4",
                                                "red4", "yellow4")) +
  labs(x = "", y = "Estdadisticos", title =
    "Estdadisticos del saldo renovable por género")

```

Estadísticos del saldo renovable por género



El comportamiento de los estadísticos diferenciando por sexo se comporta igual que el caso anterior.

Realizamos las tablas de estadísticos de las transacciones totales respecto al ingreso y al sexo.

```

tabla_3 <- bank %>%
  group_by(Attrition_Flag, Income_Category) %>%
  summarise(Media = mean(Total_Trans_Ct),
            Desv_tip = sd(Total_Trans_Ct),
            Mediana = median(Total_Trans_Ct),
            Moda = mlv(Total_Trans_Ct, method = "mfv")[1])

```

```

tabla_4 <- bank %>%
  group_by(Attrition_Flag, Gender) %>%
  summarise(Media = mean(Total_Trans_Ct),
            Desv_tip = sd(Total_Trans_Ct),
            Mediana = median(Total_Trans_Ct),
            Moda = mlv(Total_Trans_Ct, method = "mfv")[1])

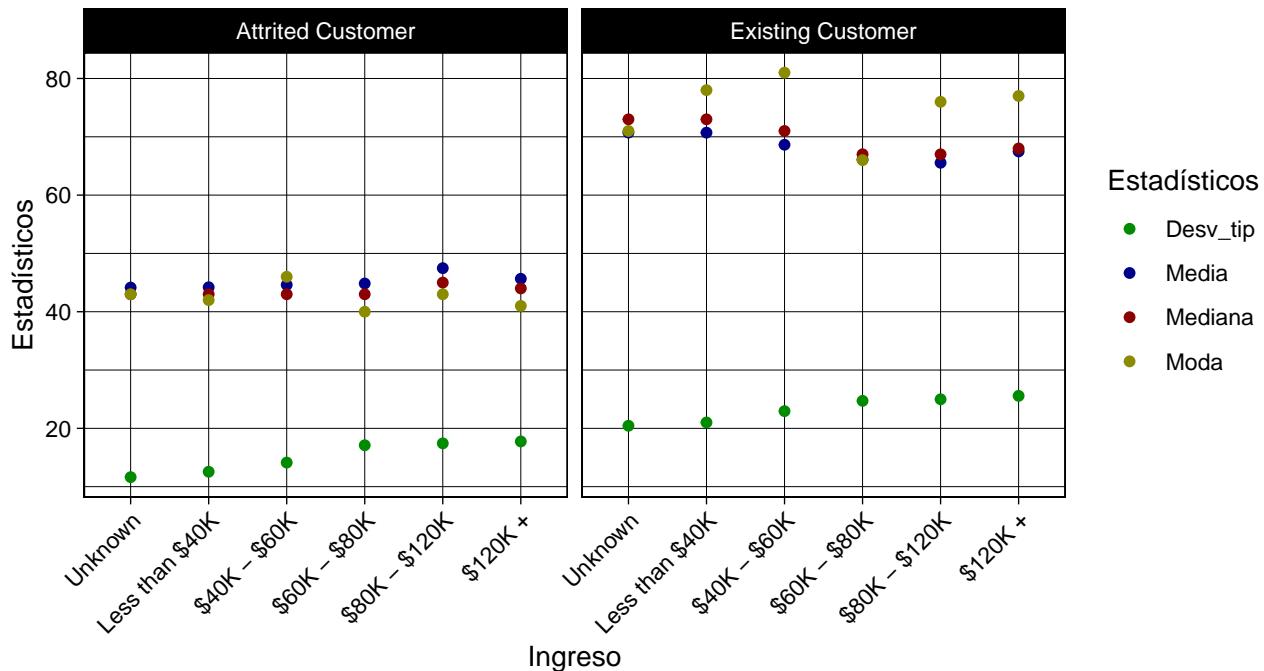
tabla_3

## # A tibble: 12 x 6
## # Groups:   Attrition_Flag [2]
##   Attrition_Flag Income_Category Media Desv_tip Mediana Moda
##   <fct>          <fct>        <dbl>    <dbl>    <dbl>   <int>
## 1 Attrited Customer Unknown      44.1     11.6     43     43
## 2 Attrited Customer Less than $40K 44.2     12.6     43     42
## 3 Attrited Customer $40K - $60K  44.6     14.1     43     46
## 4 Attrited Customer $60K - $80K  44.8     17.1     43     40
## 5 Attrited Customer $80K - $120K 47.5     17.4     45     43
## 6 Attrited Customer $120K +     45.7     17.8     44     41
## 7 Existing Customer Unknown     70.7     20.4     73     71
## 8 Existing Customer Less than $40K 70.7     21.0     73     78
## 9 Existing Customer $40K - $60K  68.6     22.9     71     81
## 10 Existing Customer $60K - $80K 66.1     24.7     67     66
## 11 Existing Customer $80K - $120K 65.5     25.0     67     76
## 12 Existing Customer $120K +    67.5     25.6     68     77

tabla_3 %>%
  gather(Estadisticos, Valor, -1, -2) %>%
  ggplot(aes(x = Income_Category, y = Valor)) +
  geom_point(aes(color = Estdadisticos)) +
  facet_wrap(~Attrition_Flag) +
  theme_linedraw() +
  scale_color_manual("Estdadisticos", values = c("green4", "blue4",
                                                "red4", "yellow4")) +
  labs(x = "Ingreso", y = "Estdadisticos", title =
       "Estdadisticos de las transacciones totales por ingreso") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

```

Estadísticos de las transacciones totales por ingreso



Nos damos cuenta que, otra vez, los estadísticos son más alto en los clientes que se quedan. Indica que éste grupo tiene una media de transacciones totales mayor, igual que el rango de oscilación de las transacciones con la mediana. De igual forma, la moda es más alta, con lo cual el número de transacciones con mayor frecuencia para cada nivel de ingreso es casi el doble en el caso de los clientes que se quedan.

tabla_4

```
## # A tibble: 4 x 6
## # Groups:   Attrition_Flag [2]
##   Attrition_Flag   Gender Media Desv_tip Mediana Moda
##   <fct>          <fct>  <dbl>    <dbl>    <dbl>  <int>
## 1 Attrited Customer F      44.1     11.9     43     42
## 2 Attrited Customer M      46.1     17.4     44     43
## 3 Existing Customer F     71.0     20.5     74     81
## 4 Existing Customer M     66.1     25.1     67     66

tabla_4 %>%
  gather(Estadisticos, Valor, -1, -2) %>%
  ggplot(aes(x = Gender, y = Valor)) +
  geom_point(aes(color = Estandisticos)) +
  facet_wrap(~Attrition_Flag) +
  theme_linedraw() +
  scale_color_manual("Estandisticos", values = c("green4", "blue4",
                                                "red4", "yellow4")) +
  labs(x = "", y = "Estandisticos", title =
    "Estandisticos de las transacciones totales por género")
```

Estadísticos de las transacciones totales por género



El comportamiento es parecido al caso anterior, con una ligera diferencia entre los hombres y las mujeres de los clientes que se quedan.

3 Machine Learning

Como explicamos al principio del trabajo, el objetivo todo el estudio que hemos realizado es intentar explicar el comportamiento de los clientes que se quedan y se van (la variable `Attrition_Flag`), con lo cual los algoritmos (aprendizaje supervisado y no supervisado) que implementamos son con ese fin: clasificar los dos grupos de clientes. Al tratarse de una variable cualitativa no podemos realizar métodos de regresión, con lo cual escogeremos la variable `Total_Trans_Ct` (número total de transacciones) para aplicar estos métodos de regresión y predecir la variable en cuestión.

Para las regresiones utilizaremos dos métodos: la regresión múltiple y el support vector regression (SVR). Para los métodos de clasificación aplicaremos la regresión logística, K-nearest neighbors (KNN), support vector machine (SVM), decision tree, random forest y gradient boosting.

Además, realizaremos también un análisis de componentes principales (ACP) para reducir la dimensión de nuestro data set a 2 componentes principales, para después aplicar el método de k-means de clustering (aprendizaje no supervisado) y graficar el resultado. Aprovecharemos que habremos reducido las dimensiones de nuestro conjunto de datos para volver a realizar la regresión logística y el support vector machine y de esta forma poder pintar los resultados en el plano.

Empezaremos por pasar nuestras variables cualitativas a numéricas para poder aplicar los modelos, guardando la variable original `Attrition_Flag` como factor ya que la necesitaremos para support vector machine, decision tree, random forest y gradient boosting.

```
bank <- bank%>%
  mutate(Attrition_Flag_1 = ifelse(Attrition_Flag == "Existing Customer", 0, 1),
        Education_Level = recode(Education_Level, "Unknown" = 0, "Uneducated" = 1,
                                   "High School" = 2, "College" = 3, "Graduate" = 4, "Post-Graduate" = 5,
                                   "Doctorate" = 6),
        Income_Category = recode(Income_Category, "Unknown" = 0, "Less than $40K" = 1,
```

```

"$40K - $60K" = 2, "$60K - $80K" = 3, "$80K - $120K" = 4, "$120K +" = 5),
Gender = recode(Gender, "M" = 1, "F" = 0)) %>%
dummy_cols(select_columns = c("Marital_Status", "Card_Category")) %>%
dplyr::select(-c("Marital_Status", "Card_Category"))

```

Separamos nuestro data set en dos conjuntos: el conjunto de entrenamiento y el conjunto de validación. También crearemos estos dos conjuntos escalando las variables cuantitativas, sin incluir las variables numéricas que provienen de una variable factor, para los métodos que usan distancias: el KNN y el support vector machine (SVM).

```

set.seed(284)
split <- sample.split(bank$Attrition_Flag, SplitRatio = 0.8)
bank.train <- subset(bank, split == TRUE)
bank.train.scale <- bank.train
bank.train.scale[,c(2,4,7:18)] <- scale(bank.train[,c(2,4,7:18)])
bank.test <- subset(bank, split == FALSE)
bank.test.scale <- bank.test
bank.test.scale[,c(2,4,7:18)] <- scale(bank.test[,c(2,4,7:18)])

```

3.1 Métodos de regresión

3.1.1 Regresión lineal múltiple

Recordamos que la regresión lineal múltiple es un método en el cual predecimos una variable dependiente o respuesta (en nuestro caso será Total_Trans_Ct) a partir de una combinación lineal de las variables independientes o explicativas. Nuestro data set modificado contiene la variable del tipo de cliente en dos tipos: numérico y factor, con lo que tendremos que excluir la variable de tipo factor como variable explicativa. Además, hemos transformado las variables del estado civil y el tipo de tarjeta en variables ficticias, con lo que tenemos que quitar una de cada tipo para poder realizar la regresión. Excluiremos también la variable Avg_Open_To_Buy ya que nos produce valores NA's.

Entrenamos el modelo de regresión lineal múltiple con nuestro conjunto de entrenamiento.

```

reg_lineal <- lm(Total_Trans_Ct ~ . - Attrition_Flag -
                  Marital_Status_Married - Card_Category_Blue - Avg_Open_To_Buy,
                  data = bank.train)

```

A continuación construiremos un modelo óptimo con el método de la eliminación hacia atrás.

```
best_reg <- step(reg_lineal, direction = "backward")
```

```
summary(best_reg)
```

```

##
## Call:
## lm(formula = Total_Trans_Ct ~ Customer_Age + Gender + Dependent_count +
##     Income_Category + Total_Relationship_Count + Months_Inactive_12_mon +
##     Contacts_Count_12_mon + Total_Revolving_Bal + Total_Amt_Chng_Q4_Q1 +
##     Total_Trans_Amt + Total_Ct_Chng_Q4_Q1 + Avg_Utilization_Ratio +
##     Attrition_Flag_1 + Marital_Status_Divorced + Marital_Status_Single +
##     Marital_Status_Unknown, data = bank.train)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -47.240   -7.999    0.935   8.639   35.202

```

```

## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)           5.184e+01  1.284e+00 40.374 < 2e-16 ***
## Customer_Age        -6.158e-02  1.711e-02 -3.600 0.000320 ***  
## Gender              -3.911e+00  4.447e-01 -8.794 < 2e-16 ***  
## Dependent_count      6.200e-01  1.052e-01  5.892 3.97e-09 ***  
## Income_Category     2.311e-01  1.497e-01  1.543 0.122849    
## Total_Relationship_Count -2.389e-01  9.476e-02 -2.521 0.011733 *  
## Months_Inactive_12_mon  4.920e-01  1.352e-01  3.640 0.000274 ***  
## Contacts_Count_12_mon -2.236e-01  1.254e-01 -1.783 0.074697 .  
## Total_Revolving_Bal   -2.867e-03  2.258e-04 -12.697 < 2e-16 ***  
## Total_Amt_Chng_Q4_Q1  -4.568e+00  6.690e-01 -6.828 9.26e-12 ***  
## Total_Trans_Amt       5.269e-03  4.442e-05 118.625 < 2e-16 ***  
## Total_Ct_Chng_Q4_Q1  -1.444e+00  6.311e-01 -2.288 0.022153 *  
## Avg_Utilization_Ratio 5.788e+00  6.855e-01  8.444 < 2e-16 ***  
## Attrition_Flag_1      -1.760e+01  4.212e-01 -41.794 < 2e-16 ***  
## Marital_Status_Divorced 3.466e+00  5.316e-01  6.520 7.44e-11 ***  
## Marital_Status_Single  3.818e+00  2.947e-01 12.957 < 2e-16 ***  
## Marital_Status_Unknown 3.077e+00  5.302e-01  5.803 6.75e-09 ***  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 12.11 on 8085 degrees of freedom
## Multiple R-squared:  0.7316, Adjusted R-squared:  0.731 
## F-statistic:  1377 on 16 and 8085 DF,  p-value: < 2.2e-16

```

A modo de ejemplo comentamos qué significa el coeficiente de la variable `Total_Revolving_Bal`. El valor del coeficiente que nos da el modelo es -0.002867 y lo interpretamos como: un aumento de una unidad en `Total_Revolving_Bal` hace que la variable `Total_Trans_Ct` baje un 0.002867 unidades.

Con el conjunto de validación crearemos las predicciones con el modelo creado anteriormente y después calcularemos la raíz del error cuadrático medio (RMSE) y la utilizaremos como medida de precisión de nuestro modelo.

```

reg_lineal_pred <- predict(best_reg, newdata = bank.test)
rmse_reg_lin <- sqrt(MSE(y_pred = reg_lineal_pred, y_true = bank.test$Total_Trans_Ct))
rmse_reg_lin

```

```
## [1] 12.05613
```

El RMSE que obtenemos está en la escala de la variable respuesta y se podría interpretar como un valor aceptable. Además de todo esto, pintaremos el normal QQ-plot y el gráfico de Fitted values vs Residuals.

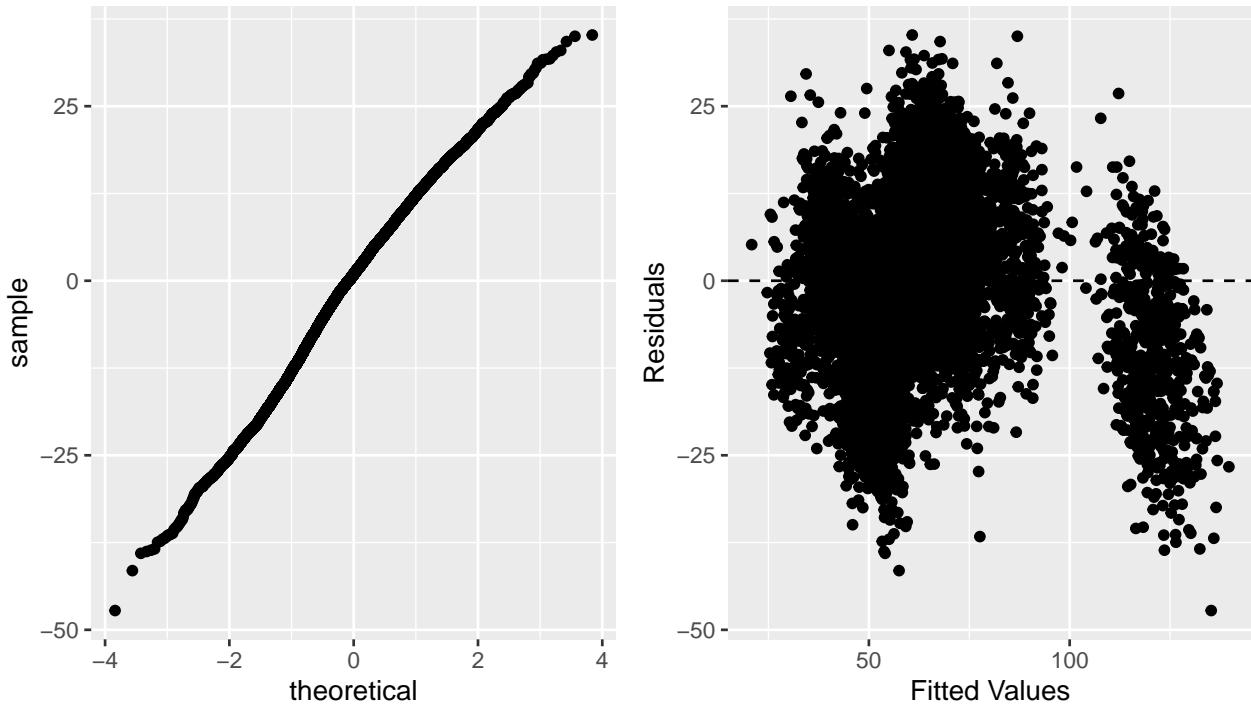
```

qq_plot_1 <- ggplot(as.data.frame(residuals(best_reg)), aes(sample = residuals(best_reg))) +
  geom_qq()

residual_plot_1 <- ggplot(data = best_reg, aes(x = .fitted, y = .resid))+
  geom_jitter()+
  geom_hline(yintercept = 0, linetype = "dashed")+
  xlab("Fitted Values")+
  ylab("Residuals")

ggarrange(qq_plot_1, residual_plot_1)

```



Se puede observar cómo el normal QQ-plot sí parece estar bien ya que la mayoría de los puntos se mueven sobre la recta diagonal que pasa por el 0, mientras que el gráfico de los residuos contra los valores ajustados parece algo más caótico.

3.1.2 Support vector regression (SVR)

Dejando atrás la regresión lineal múltiple, continuaremos con el support vector regression. Realizamos el mismo proceso, entrenamos el modelo con el conjunto de entrenamiento y luego calculamos la predicción, para más tarde calcular el RMSE.

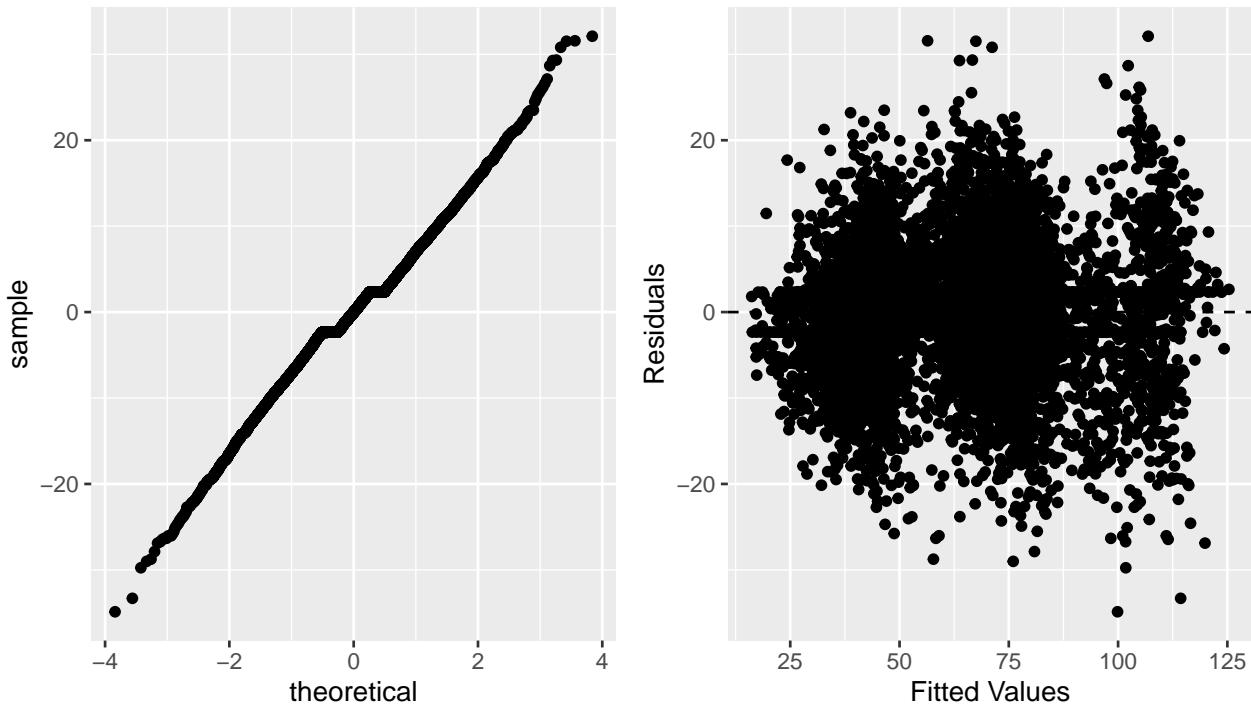
```
svr <- svm(Total_Trans_Ct ~ . -Attrition_Flag - Marital_Status_Married - Card_Category_Blue,
            data = bank.train,
            type = "eps-regression",
            kernel = "radial")
svr_pred <- predict(svr, newdata = bank.test)
rmse_svr <- sqrt(MSE(y_pred = svr_pred, y_true = bank.test$Total_Trans_Ct))
rmse_svr

## [1] 9.330735
```

Nos sale un RMSE más pequeño que la regresión lineal múltiple, con lo cual es mejor este modelo a priori.

```
qq_plot_2 <- ggplot(as.data.frame(residuals(svr)), aes(sample = residuals(svr))) +
  geom_qq()

residual_plot_2 <- ggplot(data = data.frame("fitted" = svr$fitted, "resid" = svr$residuals),
                           aes(x = fitted, y = resid)) +
  geom_jitter()+
  geom_hline(yintercept = 0, linetype = "dashed")+
  xlab("Fitted Values")+
  ylab("Residuals")
ggarrange(qq_plot_2, residual_plot_2)
```



El normal QQ-plot y el gráfico de los valores ajustados vs residuos son muy parecidos que la regresión lineal.

3.2 Métodos de clasificación

En este apartado aplicaremos los algoritmos de regresión logística, KNN, SVM y decision tree para clasificar la variable `Attrition_Flag`.

3.2.1 Regresión logística

Primero de todo vamos a entrenar el modelo con el conjunto de entrenamiento, realizando la regresión logística múltiple teniendo a la variable `Attrition_Flag_1` como variable de respuesta y todas las otras variables como explicativas (exceptuando `Attrition_Flag`, `Marital_Status_Married` y `Card_Category_Blue`). Utilizaremos el método de eliminación hacia atrás para encontrar el modelo más óptimo.

```
reg_log <- glm(Attrition_Flag_1 ~ . - Attrition_Flag - Marital_Status_Married - Card_Category_Blue,
                 data = bank.train, family = binomial)
reg_backward <- step(reg_log, direction = "backward")

summary(reg_backward)

##
## Call:
## glm(formula = Attrition_Flag_1 ~ Gender + Dependent_count + Income_Category +
##     Months_on_book + Total_Relationship_Count + Months_Inactive_12_mon +
##     Contacts_Count_12_mon + Credit_Limit + Total_Revolving_Bal +
##     Total_Amt_Chng_Q4_Q1 + Total_Trans_Amt + Total_Trans_Ct +
##     Total_Ct_Chng_Q4_Q1 + Marital_Status_Divorced + Marital_Status_Single +
##     Marital_Status_Unknown + Card_Category_Gold + Card_Category_Platinum +
##     Card_Category_Silver, family = binomial, data = bank.train)
##
```

```

## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.7296 -0.3571 -0.1703 -0.0682  3.5253
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           5.549e+00  3.737e-01 14.848 < 2e-16 ***
## Gender                -9.456e-01  1.413e-01 -6.694 2.17e-11 ***
## Dependent_count        1.259e-01  3.333e-02  3.778 0.000158 ***
## Income_Category        1.728e-01  4.957e-02  3.486 0.000490 ***
## Months_on_book         -1.211e-02  5.273e-03 -2.297 0.021603 *
## Total_Relationship_Count -4.556e-01  3.072e-02 -14.833 < 2e-16 ***
## Months_Inactive_12_mon  4.973e-01  4.263e-02 11.667 < 2e-16 ***
## Contacts_Count_12_mon  4.899e-01  4.101e-02 11.946 < 2e-16 ***
## Credit_Limit            -1.449e-05  6.412e-06 -2.261 0.023774 *
## Total_Revolving_Bal    -1.014e-03  5.269e-05 -19.242 < 2e-16 ***
## Total_Amt_Chng_Q4_Q1   -5.432e-01  2.118e-01 -2.564 0.010335 *
## Total_Trans_Amt          4.761e-04  2.530e-05 18.814 < 2e-16 ***
## Total_Trans_Ct            -1.181e-01  4.144e-03 -28.488 < 2e-16 ***
## Total_Ct_Chng_Q4_Q1    -2.847e+00  2.118e-01 -13.438 < 2e-16 ***
## Marital_Status_Divorced 4.078e-01  1.725e-01  2.364 0.018086 *
## Marital_Status_Single    5.625e-01  9.429e-02  5.966 2.43e-09 ***
## Marital_Status_Unknown   4.500e-01  1.670e-01  2.694 0.007055 **
## Card_Category_Gold       9.331e-01  3.971e-01  2.350 0.018795 *
## Card_Category_Platinum   1.054e+00  6.959e-01  1.514 0.130030
## Card_Category_Silver     4.522e-01  2.168e-01  2.086 0.036999 *
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 7143.2 on 8101 degrees of freedom
## Residual deviance: 3748.7 on 8082 degrees of freedom
## AIC: 3788.7
##
## Number of Fisher Scoring iterations: 6

```

La interpretación de los coeficientes, tomando `Total_Trans_Amt` como ejemplo, sería: un aumento de una unidad en la variable `Total_Trans_Amt` (una unidad más de la cantidad total de transacciones) hace aumentar el logaritmo de la probabilidad de darse da baja del banco (en comparación con quedarse en el banco) en 0.0004761.

Realizamos la predicción del modelo con el conjunto de test. Como la predicción nos la dará en probabilidades, consideramos que una probabilidad mayor que 0.5 es que el cliente se da de baja. Después, dibujamos la matriz de confusión y mostramos la precisión del modelo utilizando `accuracy`.

```

reg_backward_probs <- predict(reg_backward, newdata = bank.test, type = "response")
reg_backward_pred <- ifelse(reg_backward_probs > 0.5, 1, 0)
reg_back_cm <- table(reg_backward_pred, bank.test$Attrition_Flag_1,
                      dnn = c("Predicción","Valor Real"))
reg_back_cm

##             Valor Real
## Predicción      0      1
##                 0 1637 137
##                 1   63 188

```

```

reg_back_accu <- mean(reg_backward_pred == bank.test$Attrition_Flag_1)
reg_back_accu

## [1] 0.9012346

```

Vemos que la precisión del modelo es del 90.1234568% y nos fijamos en el error tipo II. El número de casos en los cuales nosotros predecimos que el cliente se queda y al final se da de baja es 137, correspondiendo con los falsos negativos y su ratio es 0.4215385 ($\frac{FN}{TP+FN}$). Es un ratio un tanto alto, y no nos interesa que sea tanto ya que quiere decir que predecimos mal los casos en los que creemos que se quedan pero al final se van. Una solución sería considerar que se van a partir de una probabilidad más baja que 0.5, pero en detrimento bajamos la precisión.

3.2.2 KNN

El método de K-Nearest Neighbors es otro método de clasificación. El algoritmo reconoce patrones en los datos sin un aprendizaje específico, consiguiendo un criterio de agrupamiento de los datos a partir de un conjunto de entrenamiento. Necesitamos normalizar los datos que tenemos para poder proceder con el método, y luego creamos de nuevos los conjuntos de entrenamiento y de validación normalizados.

```

normalizar <- function(x){
  return ((x - min(x))/(max(x) - min(x)))
}
bank.normalized <- as.data.frame(lapply(bank[,2:27], normalizar))
set.seed(284)
split.norm <- sample.split(bank.normalized$Attrition_Flag_1, SplitRatio = 0.8)
bank.train.norm <- subset(bank.normalized, split.norm == TRUE)
bank.test.norm <- subset(bank.normalized, split.norm == FALSE)

```

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

```

bank_knn <- knn(train = bank.train.norm[,-18], test = bank.test.norm[,-18],
                  cl = bank.train.norm$Attrition_Flag_1, k = 10)
knn_cm <- table(bank_knn, bank.test.norm$Attrition_Flag_1, dnn =
                  c("Predicción","Valor Real"))
knn_cm

##          Valor Real
## Predicción      0     1
##               0 1661  207
##               1   39  118
knn_accu <- mean(bank_knn == bank.test.norm$Attrition_Flag_1)
knn_accu

## [1] 0.8785185

```

Obtenemos una precisión del 87.8518519%, mejorando la regresión logística y el análisis discriminante lineal. Vemos también que el número de falsos positivos que hay son 207, y el ratio de falsos negativos es de 0.6369231, bastante más bajo que los otros dos modelos.

3.2.3 Decision tree

Como nuestra variable dependiente es binaria, vamos a construir un árbol de clasificación en vez de un árbol de regresión. Este algoritmo, de forma general, separa los datos en grupos (las clases de la variable dependiente) utilizando la mejor variable explicativa en cada nodo. Vamos a construir el árbol de clasificación

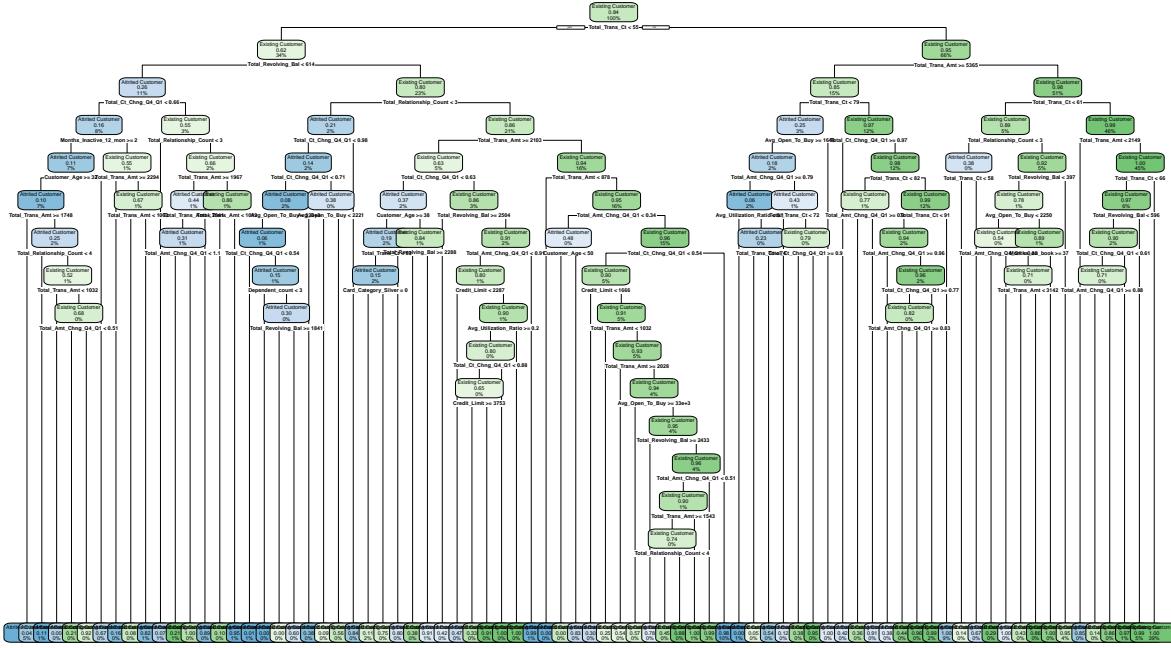
con el conjunto de entrenamiento cogiendo el parámetro de complejidad (cp) igual a 0. Luego haremos la predicción con el conjunto de test, para más adelante dibujar la matriz de confusión.

```
bank_tree <- rpart(Attrition_Flag ~.-Attrition_Flag_1, data = bank.train,
                     method = "class", control = rpart.control(cp = 0))
tree_prediction <- predict(bank_tree, newdata = bank.test, type = "class")
tree_cm <- confusionMatrix(tree_prediction, bank.test$Attrition_Flag)
tree_cm

## Confusion Matrix and Statistics
##
##                               Reference
## Prediction          Attrited Customer Existing Customer
## Attrited Customer           260              81
## Existing Customer            65             1619
##
##                               Accuracy : 0.9279
##                               95% CI : (0.9158, 0.9388)
## No Information Rate : 0.8395
## P-Value [Acc > NIR] : <2e-16
##
##                               Kappa : 0.7377
##
## Mcnemar's Test P-Value : 0.2145
##
##                               Sensitivity : 0.8000
##                               Specificity : 0.9524
## Pos Pred Value : 0.7625
## Neg Pred Value : 0.9614
## Prevalence : 0.1605
## Detection Rate : 0.1284
## Detection Prevalence : 0.1684
## Balanced Accuracy : 0.8762
##
## 'Positive' Class : Attrited Customer
##
```

Obtenemos una precisión del modelo del 92.7901235% y 65 falsos negativos, con lo cual el ratio de falsos negativos es 0.2. Ahora vamos a dibujar el árbol que hemos obtenido.

```
rpart.plot(bank_tree)
```



Nos sale un árbol bastante grande y difícil de interpretar. Nos da a pensar que podemos podar el árbol. Para ello, vamos a hacer un análisis del parámetro de complejidad. Cogeremos el valor del parámetro que minimice el error de la validación cruzada (xerror).

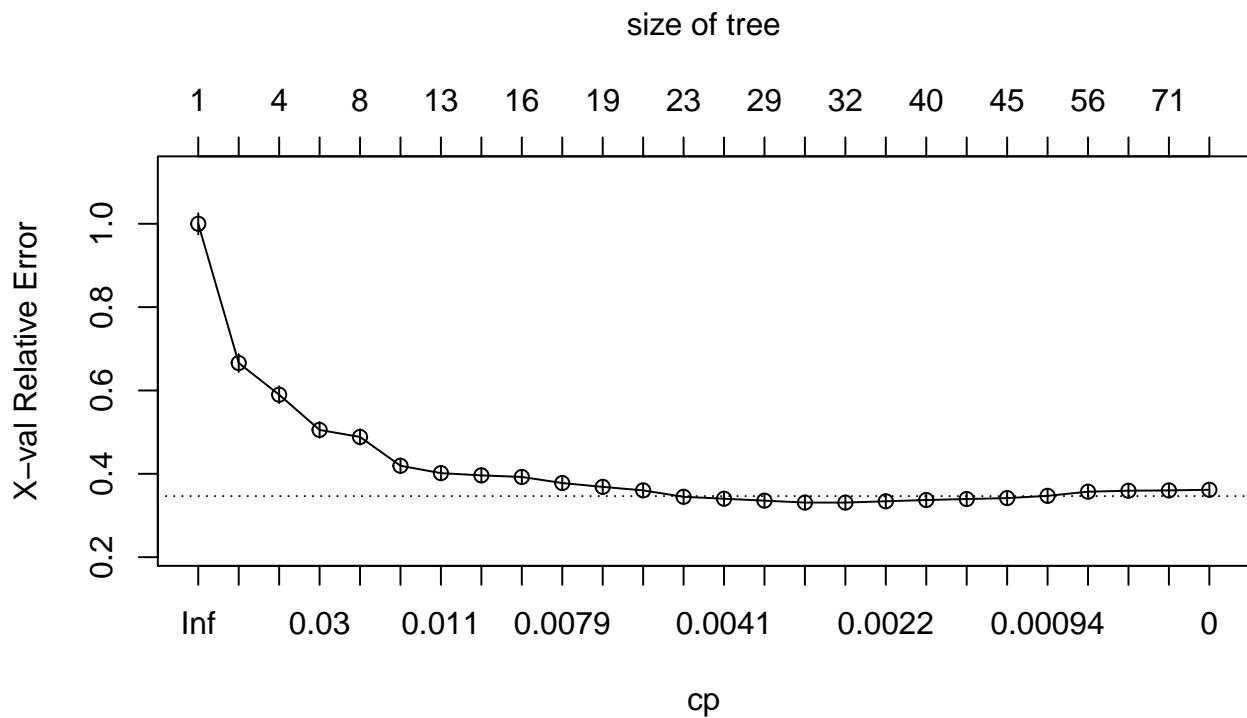
```
printcp(bank_tree)
```

```
##
## Classification tree:
## rpart(formula = Attrition_Flag ~ . - Attrition_Flag_1, data = bank.train,
##       method = "class", control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] Avg_Open_To_Buy          Avg_Utilization_Ratio      Card_Category_Silver
## [4] Credit_Limit              Customer_Age             Dependent_count
## [7] Months_Inactive_12_mon   Months_on_book         Total_Amt_Chng_Q4_Q1
## [10] Total_Ct_Chng_Q4_Q1     Total_Relationship_Count Total_Revolving_Bal
## [13] Total_Trans_Amt          Total_Trans_Ct
##
## Root node error: 1302/8102 = 0.1607
##
## n= 8102
##
##          CP nsplit rel error xerror      xstd
## 1  0.16935484      0 1.000000 1.000000 0.025389
## 2  0.07603687      2 0.661290 0.665900 0.021371
## 3  0.03955453      3 0.585250 0.589860 0.020251
## 4  0.02265745      5 0.506140 0.505380 0.018885
## 5  0.01881720      7 0.460830 0.488480 0.018594
## 6  0.01228879     10 0.394780 0.419350 0.017331
## 7  0.00998464     12 0.370200 0.401690 0.016988
## 8  0.00960061     13 0.360220 0.396310 0.016882
## 9  0.00806452     15 0.341010 0.392470 0.016806
## 10 0.00768049     17 0.324880 0.377880 0.016511
## 11 0.00691244     18 0.317200 0.368660 0.016321
```

```

## 12 0.00614439      19  0.31029 0.36022 0.016145
## 13 0.00460829      22  0.29186 0.34485 0.015817
## 14 0.00358423      24  0.28264 0.34025 0.015717
## 15 0.00345622      28  0.26575 0.33564 0.015617
## 16 0.00307220      30  0.25883 0.33103 0.015515
## 17 0.00230415      31  0.25576 0.33103 0.015515
## 18 0.00204813      36  0.24347 0.33410 0.015583
## 19 0.00179211      39  0.23733 0.33717 0.015650
## 20 0.00153610      42  0.23195 0.33948 0.015701
## 21 0.00115207      44  0.22888 0.34178 0.015751
## 22 0.00076805      46  0.22657 0.34716 0.015867
## 23 0.00061444      55  0.21966 0.35714 0.016080
## 24 0.00038402      60  0.21659 0.35945 0.016128
## 25 0.00012801      70  0.21275 0.36022 0.016145
## 26 0.00000000      76  0.21198 0.36175 0.016177
plotcp(bank_tree)

```



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

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

```

bank_tree_fit <- prune(bank_tree, min_best_cp)
tree_fit_prediction <- predict(bank_tree_fit, newdata = bank.test, type = "class")
tree_fit_cm <- confusionMatrix(tree_fit_prediction, bank.test$Attrition_Flag)
tree_fit_cm

```

```

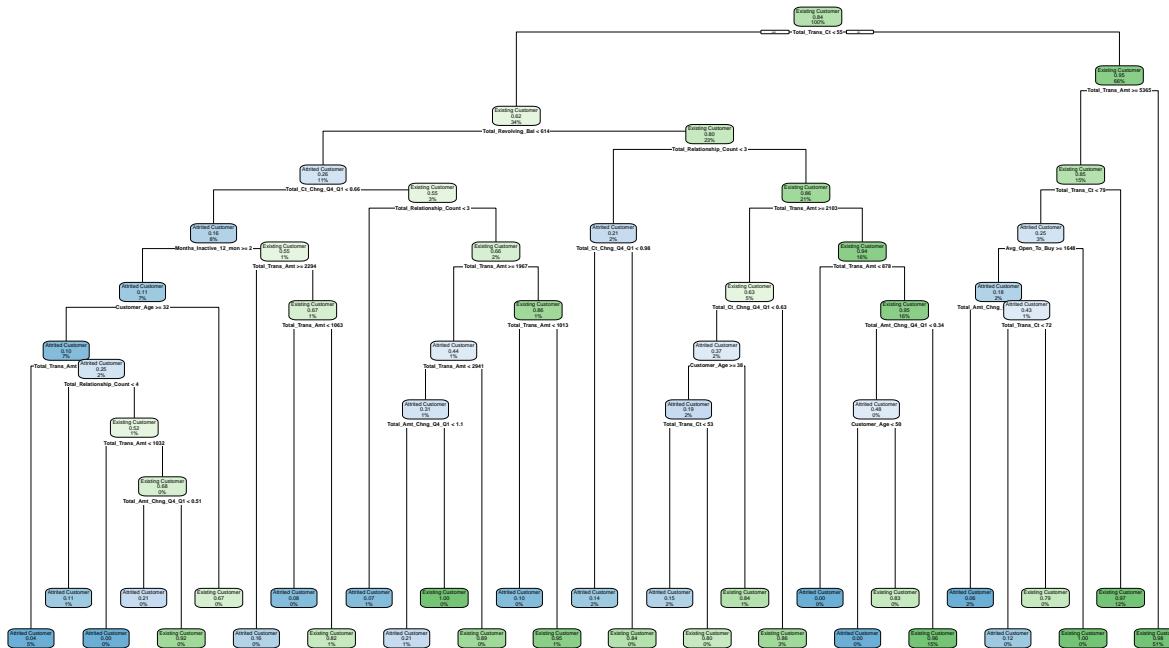
## Confusion Matrix and Statistics
##
##                               Reference
##                               Attrited Customer Existing Customer
## Prediction

```

```

## Attrited Customer          238      52
## Existing Customer          87       1648
##
##                               Accuracy : 0.9314
##                               95% CI : (0.9195, 0.942)
## No Information Rate : 0.8395
## P-Value [Acc > NIR] : < 2.2e-16
##
##                               Kappa : 0.7337
##
## McNemar's Test P-Value : 0.003929
##
##                               Sensitivity : 0.7323
##                               Specificity : 0.9694
## Pos Pred Value : 0.8207
## Neg Pred Value : 0.9499
## Prevalence : 0.1605
## Detection Rate : 0.1175
## Detection Prevalence : 0.1432
## Balanced Accuracy : 0.8509
##
## 'Positive' Class : Attrited Customer
##
rpart.plot(bank_tree_fit)

```



Nos sale una precisión de 93.1358025%, ligeramente mejor que el árbol sin podar. Además, tenemos 87 falsos negativos, un poco más que el árbol sin podar. Aunque puede parecer imperceptible, el árbol realmente se poda y se reducen las ramas y los nodos, pero sigue siendo un árbol difícil de interpretar.

3.2.4 SVM

De la misma forma que en el caso de la regresión, realizaremos esta vez el mismo proceso pero para clasificar la variable `Attrition_Flag`. Al tratarse de clasificación,

```
svm <- svm(Attrition_Flag ~ . -Attrition_Flag_1 - Marital_Status_Married - Card_Category_Blue,
            data = bank.train.scale,
            type = "C-classification",
            kernel = "linear")
svm_pred <- predict(svm, newdata = bank.test.scale)
svm_cm <- confusionMatrix(svm_pred, bank.test.scale$Attrition_Flag)
svm_cm$table

##                                     Reference
## Prediction              Attrited Customer Existing Customer
##   Attrited Customer          179             57
##   Existing Customer          146            1643
svm_cm$overall[1]

## Accuracy
## 0.8997531
```

Obtenemos una precisión del 89.9753086% y 146 falsos negativos, con lo cual el ratio de falsos negativos es 0.4492308.

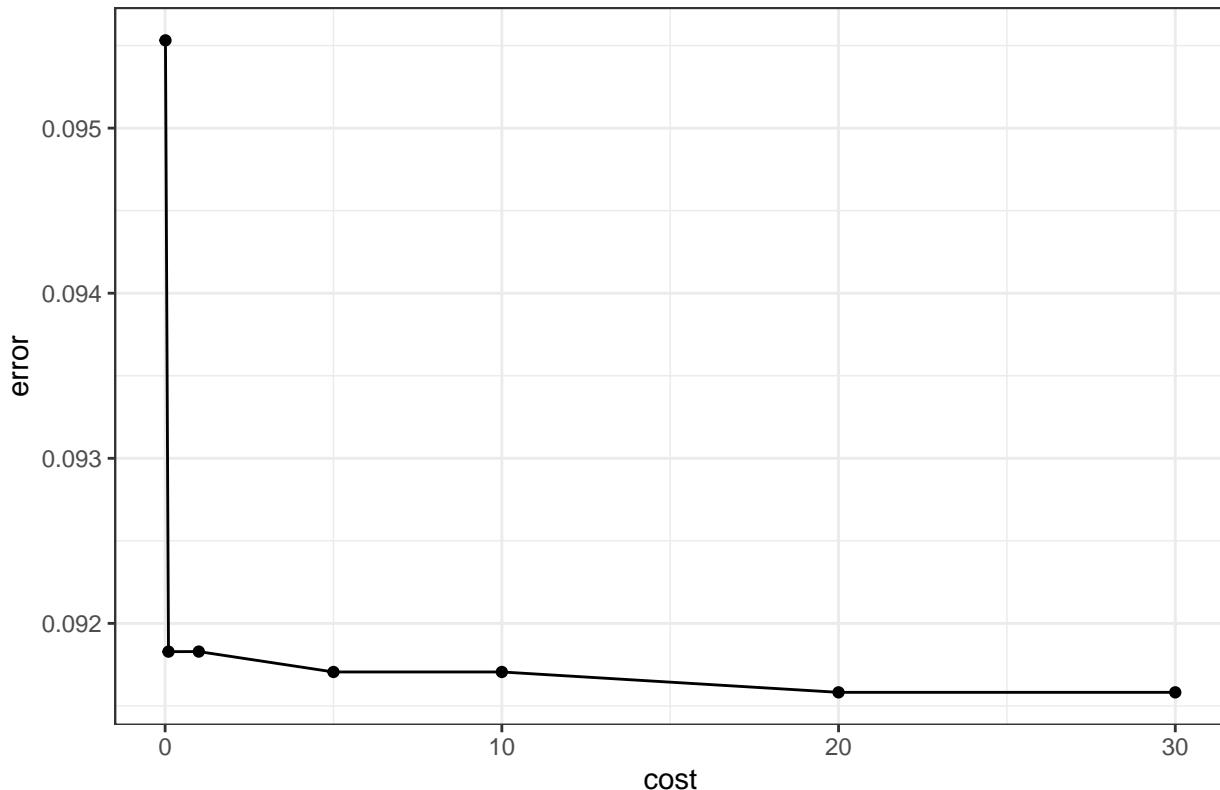
Es importante introducir el concepto de hiperparámetro: son aquellos parámetros en los que el usuario debe introducir sus valores y no es el propio algoritmo que encuentra su valor óptimo. Necesitamos hacer tuning (encontrar el mejor hiperparámetro para el algoritmo) para encontrar el mejor hiperparámetro cost (aplicaremos tuning por grid search).

```
set.seed(284)
svm_cv <- tune("svm", Attrition_Flag ~ . -Attrition_Flag_1 - Marital_Status_Married - Card_Category_Blue
                kernel = 'linear',
                ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 20, 30)))
summary(svm_cv)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##     20
##
## - best performance: 0.09158226
##
## - Detailed performance results:
##   cost      error dispersion
## 1  0.01  0.09553150  0.01104071
## 2  0.10  0.09182887  0.01080976
## 3  1.00  0.09182917  0.01143689
## 4  5.00  0.09170571  0.01169795
## 5 10.00  0.09170571  0.01169795
## 6 20.00  0.09158226  0.01167958
## 7 30.00  0.09158226  0.01167958
```

```
ggplot(data = svm_cv$performances, aes(x = cost, y = error)) +
  geom_line() +
  geom_point() +
  labs(title = "Error de clasificación vs hiperparámetro C") +
  theme_bw()
```

Error de clasificación vs hiperparámetro C



```
best_smv <- svm_cv$best.model
best_svm_pred <- predict(best_smv, newdata = bank.test.scale)
best_svm_cm <- confusionMatrix(best_svm_pred, bank.test.scale$Attrition_Flag)
best_svm_cm$table
```

```
##                               Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer              179                  57
##   Existing Customer              146                1643
best_svm_cm$overall[1]
```

```
## Accuracy
## 0.8997531
```

No hay diferencia ya que el error rate son muy parecidos para todos los valores del hiperparámetro cost.

La precisión que conseguimos después de realizar el tuning es 89.9753086%, con 146 falsos negativos y el ratio de falsos negativos es 0.4492308.

3.3 ACP, Clustering y otros estudios con ACP

3.3.1 ACP

Nuestro data set contiene muchas observaciones y más de 20 variables. Una de las cosas interesantes es poder pintar en un gráfico resultados visuales, pero con tantas variables no podemos proyectar sobre el plano o el espacio. Para conseguir este fin, vamos a realizar un método de reducción de dimensiones (Análisis de componentes principales) hasta conseguir quedarnos con 2 componentes principales.

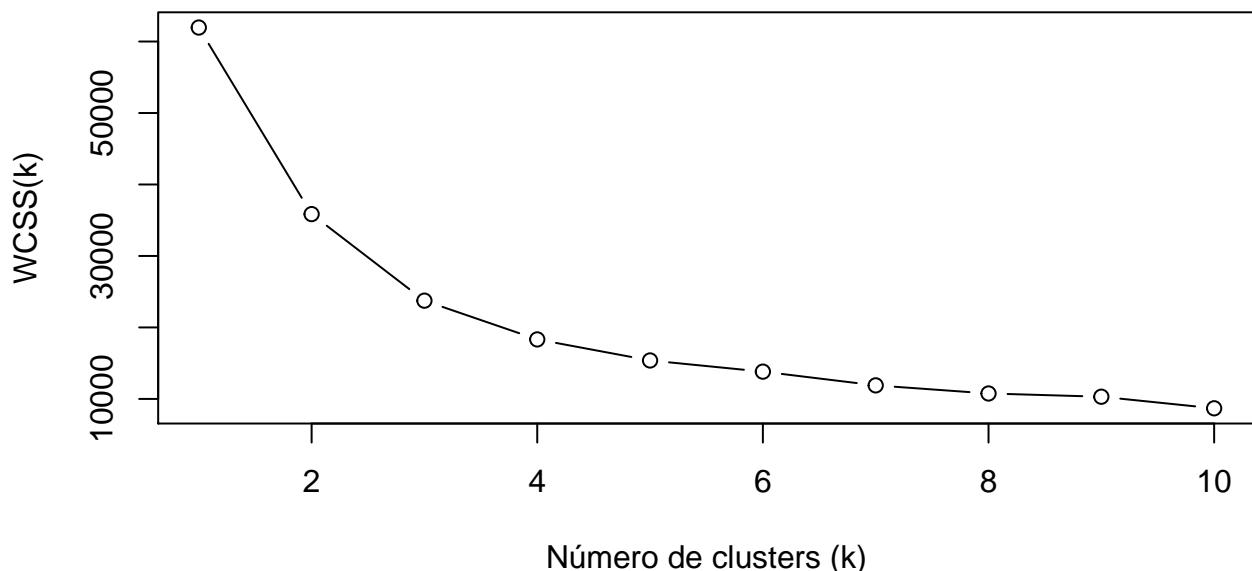
```
acp = preProcess(x = select(bank, -c("Attrition_Flag", "Attrition_Flag_1")),
                  method = "pca", pcaComp = 2)
acp = predict(acp, bank)
```

3.3.2 K-means Clustering

A continuación realizaremos clustering usando k-means. Primero utilizaremos el método del codo para buscar el número de componentes principales que optimiza la elección de k.

```
set.seed(284)
wcss = vector()
for (i in 1:10){
  wcss[i] <- sum(kmeans(acp[,-1], i)$withinss)
}
plot(1:10, wcss, type = 'b', main = "Método del codo",
     xlab = "Número de clusters (k)", ylab = "WCSS(k)")
```

Método del codo



Observamos que a partir de 4 clusters la curva empieza a tener poca bajada pronunciada, con lo cual coger 4 cluster podría ser la mejor opción.

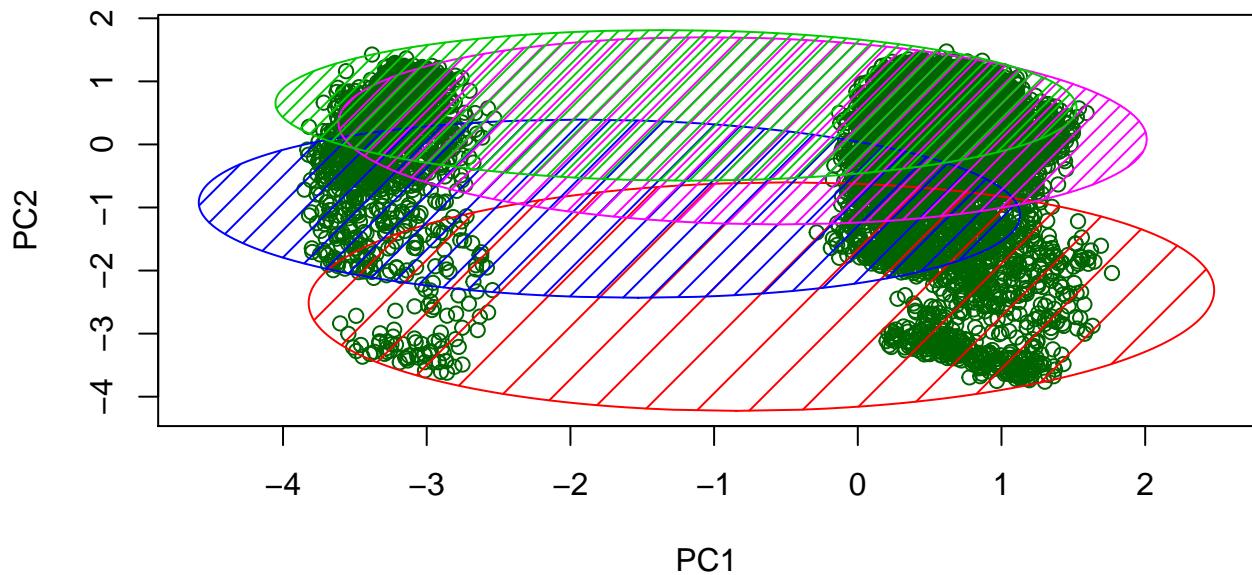
```
k_means = kmeans(acp[,-c(1,2)], 4, iter.max = 300, nstart = 10)
clusplot(acp,
          k_means$cluster,
```

```

lines = 0,
shade = TRUE,
color = TRUE,
plotchar = FALSE,
span = TRUE,
main = "Clustering de clientes",
xlab = "PC1",
ylab = "PC2"
)

```

Clustering de clientes



These two components explain 77.6 % of the point variability.

Las dos componentes principales nos explica un 77.6% de la variabilidad de los puntos. Se puede observar cómo el cluster verde está prácticamente dentro del císter lila.

3.3.3 Regresión logística después de ACP

De la misma forma que hemos hecho con el data set `bank`, para los conjuntos de entrenamiento y de validación también aplicaremos ACP para reducir la dimensión en 2. De esta forma podremos proyectar la precisión de los modelos en el plano.

```

acp_1 <- preProcess(x = select(bank.train, -c("Attrition_Flag", "Attrition_Flag_1")),
                      method = "pca", pcaComp = 2)
acp_train <- predict(acp_1, bank.train)
acp_test <- predict(acp_1, bank.test)

acp_train = select(acp_train, -c("Attrition_Flag"))
acp_test = select(acp_test, -c("Attrition_Flag"))

```

Como nuestro conjunto de datos está desbalanceado respecto a la variable `Attrition_Flag` ya que hay un 84% aproximadamente de clientes que se quedan y solamente hay un 16% de clientes que abandonan. De

alguna forma no hemos prestado atención en este desbalanceo ya que con los métodos sin aplicar ACP al final los resultados son bastante decentes, pero al aplicar ACP y reducir todo a 2 variables se nota muchísimo más este efecto de desbalanceo. Utilizaremos el paquete ROSE para balancear nuestros datos, y en vez de fijarnos en el accuracy nos centraremos más en la medida de AUC (área bajo la curva) que nos da la función `roc.curve`.

```
data.rose.train <- ROSE(Attrition_Flag_1 ~., data = acp_train, seed = 1)$data
reg_log_acp <- glm(Attrition_Flag_1 ~., data = data.rose.train, family = binomial)
reg_log_probs <- predict(reg_log_acp, newdata = acp_test, type = "response")
reg_log_pred <- ifelse(reg_log_probs > 0.5, 1, 0)
reg_cm <- table(reg_log_pred, acp_test$Attrition_Flag_1, dnn = c("Predicción", "Valor Real"))
reg_accu <- mean(reg_log_pred == acp_test$Attrition_Flag_1)
reg_cm

##          Valor Real
## Predicción      0     1
##                 0 1071 118
##                 1  629 207
reg_accu

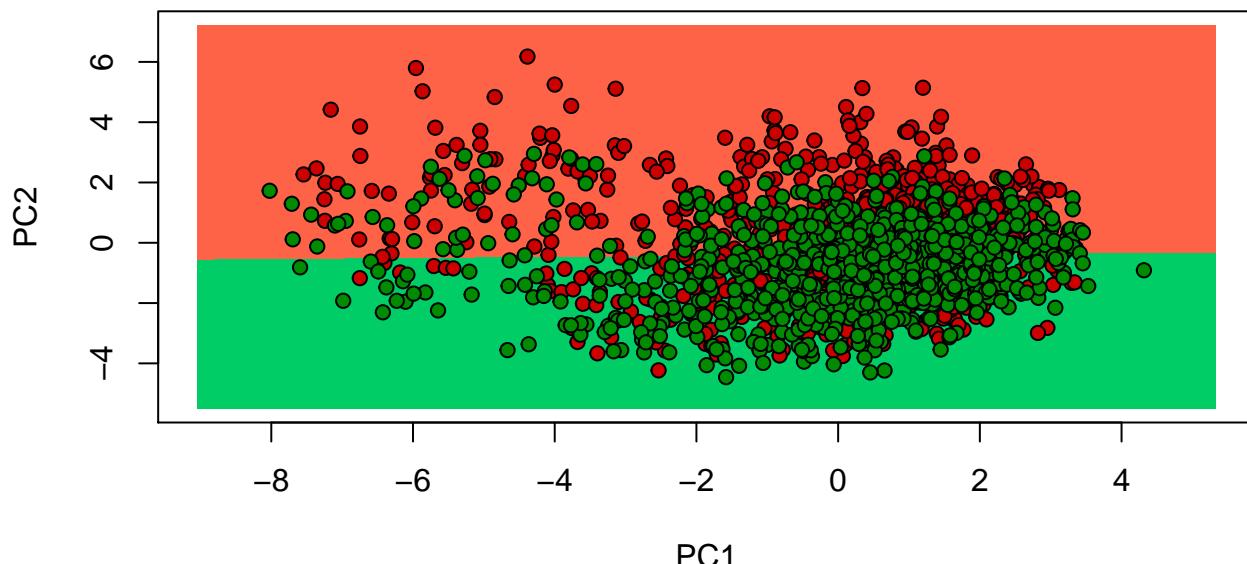
## [1] 0.6311111
```

Obtenemos una precisión del 63.111111%, un valor bastante peor que la regresión logística anterior. Vemos también que el número de falsos positivos que hay son 118, y el ratio de falsos negativos es de 0.3630769.

A continuación, pintaremos en un gráfico la región de la predicción y donde realmente se sitúan los puntos en el conjunto de validación.

```
data.rose.test <- ROSE(Attrition_Flag_1 ~., data = acp_test, seed = 1)$data
set = data.rose.test
X1 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
X2 = seq(min(set[, 3]) - 1, max(set[, 3]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('PC1', "PC2")
y_grid = predict(reg_log_acp, newdata = grid_set, type = "response")
y_grid = ifelse(y_grid > 0.5, 1, 0)
plot(set[,c(2,3)],
     main = 'Clasificación (Conjunto de Validación)',
     xlab = 'PC1', ylab = 'PC2',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set[, c(2,3)], pch = 21, bg = ifelse(set[, 1] == 1, 'green4', 'red3'))
```

Clasificación (Conjunto de Validación)



Realmente se puede observar que el modelo se equivoca mucho, dándonos cuenta de que el modelo de regresión logística tal vez no sea un buen modelo para clasificar.

```
roc.curve(acp_test$Attrition_Flag_1, reg_lineal_pred, plotit = F)
```

```
## Area under the curve (AUC): 0.889
```

Pero la medida AUC nos da un valor alto indicando que es un buen modelo.

3.4 Métodos más potentes

Como ampliación de los métodos usados anteriormente vamos a implementar los métodos de Random Forest y Gradient Boosting. Son dos algoritmos bastante pesados computacionalmente, y es por esta razón por la que los hemos dejado para lo último.

3.4.1 Random Forest

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

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

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

R tiene una función específica llamada `randomForest` del paquete `randomForest`, pero esta función no nos encuentra el mejor valor de `mtry` (nuestro hiperparámetro). Usaremos el paquete `caret` que contiene la función `train`, la cual nos encuentra el valor de `mtry` que maximiza la precisión de los modelos con cross

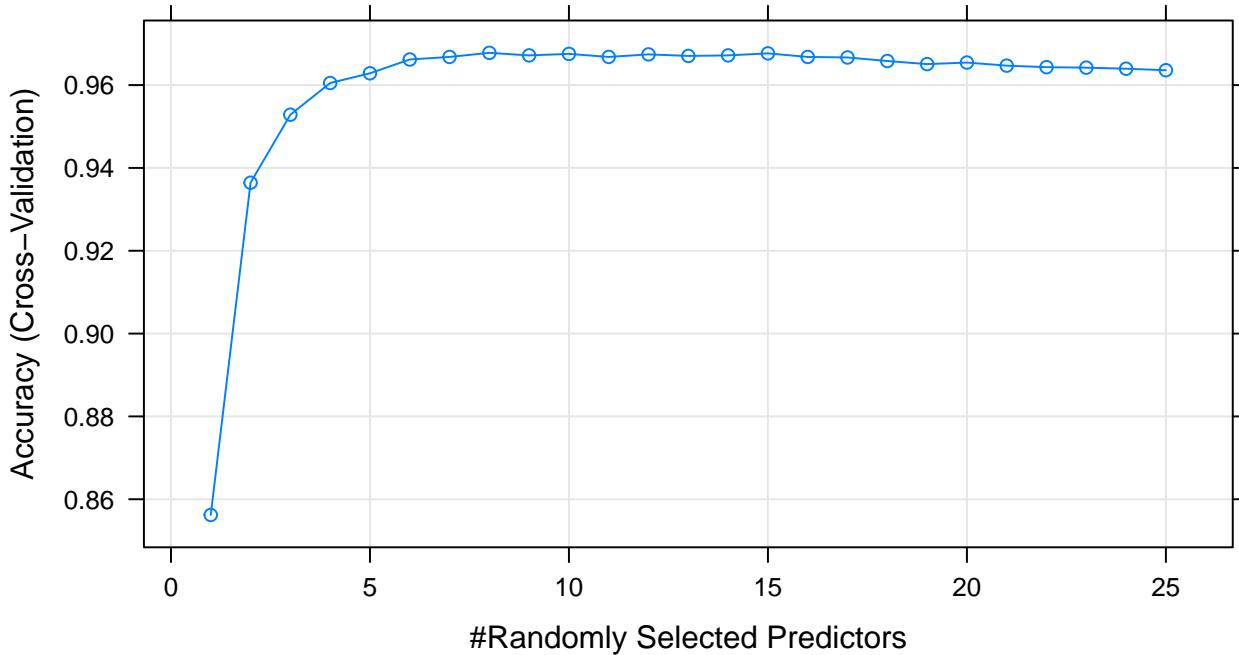
validation K-folds. A continuación construiremos el modelo con el conjunto de entrenamiento y también usaremos cross-validation 5-folds.

```
fitcontrol <- trainControl(method = "cv", number = 5, summaryFunction = defaultSummary)
rf_grid <- expand.grid(mtry = c(1:25))
bank_rf <- train(Attrition_Flag ~ . -Attrition_Flag_1, data = bank.train,
                  method = "rf", metric = "Accuracy", trControl = fitcontrol,
                  tuneGrid = rf_grid, distribution = "binomial")
bank_rf

## Random Forest
##
## 8102 samples
##    26 predictor
##      2 classes: 'Attrited Customer', 'Existing Customer'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 6481, 6481, 6482, 6482, 6482
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1    0.8562083  0.1661097
##   2    0.9364355  0.7300094
##   3    0.9528515  0.8109719
##   4    0.9605037  0.8444480
##   5    0.9628489  0.8555243
##   6    0.9661817  0.8693155
##   7    0.9667984  0.8723158
##   8    0.9677863  0.8767752
##   9    0.9671687  0.8745574
##  10   0.9675389  0.8763248
##  11   0.96667985 0.8732763
##  12   0.9674155  0.8759080
##  13   0.9670450  0.8742694
##  14   0.9671686  0.8747819
##  15   0.9676624  0.8769163
##  16   0.96667983 0.8736001
##  17   0.96666749 0.8728622
##  18   0.9658109  0.8694679
##  19   0.9650701  0.8666856
##  20   0.9654404  0.8679438
##  21   0.9647000  0.8652807
##  22   0.9643293  0.8639794
##  23   0.9642062  0.8636656
##  24   0.9639595  0.8626442
##  25   0.9635890  0.8613103
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
best_mtry = unname(unlist(bank_rf$bestTune))
```

Vemos que el modelo mismo nos devuelve el valor óptimo que maximiza la precisión de los modelos, con lo cual `mtry = 8` es el valor elegido. Vamos a graficarlo para verlo de forma más visual.

```
plot(bank_rf)
```



En efecto, el `mtry` que maximiza la precisión es 8.

Realizamos la predicción del modelo con el conjunto de test y creamos la matriz de confusión.

```
rf_pred <- predict(bank_rf, newdata = bank.test)
rf_cm <- confusionMatrix(rf_pred, bank.test$Attrition_Flag)
rf_cm
```

```
## Confusion Matrix and Statistics
##
##                               Reference
## Prediction              Attrited Customer Existing Customer
##   Attrited Customer          270             32
##   Existing Customer           55            1668
##
##                               Accuracy : 0.957
##                               95% CI  : (0.9473, 0.9654)
##                               No Information Rate : 0.8395
##                               P-Value [Acc > NIR]  : < 2e-16
##
##                               Kappa : 0.8359
##
##   Mcnemar's Test P-Value : 0.01834
##
##                               Sensitivity : 0.8308
##                               Specificity  : 0.9812
##                               Pos Pred Value : 0.8940
##                               Neg Pred Value : 0.9681
##                               Prevalence    : 0.1605
##                               Detection Rate : 0.1333
##   Detection Prevalence : 0.1491
```

```

##      Balanced Accuracy : 0.9060
##
##      'Positive' Class : Attrited Customer
##

```

Nos proporciona una precisión de 95.7037037% y 55 falsos negativos. La precisión mejora sustancialmente respecto a los otros métodos, pero no llega a la precisión de K-Nearest Neighbors.

3.4.2 Gradient Boosting

Gradient Boosting se trata de otro método de ensamble para problemas de regresión y clasificación. Al igual que Random Forest, consiste en construir un grupo de modelos débiles para transformarlos en uno robusto. Es un método de boosting que tiene como objetivo optimizar una función coste arbitraria (la precisión en nuestro caso). Como hemos hecho con Random Forest, haremos tuning por grid search y encontraremos, con la función `train`, los valores de los hiperparámetros `interaction.depth`, `n.trees`, `shrinkage` y `n.minobsinnode`. Cogemos el conjunto de entrenamiento y construimos el modelo.

```

gbm_grid = expand.grid(interaction.depth = c(1,4,7,10),
                      n.trees = c(500,1000,2000),
                      shrinkage = c(0.005, 0.02,0.05),
                      n.minobsinnode = 10)

bank_gbm <- train(Attrition_Flag ~ . -Attrition_Flag_1, data = bank.train, method = "gbm",
                    metric = "Accuracy", trControl = fitcontrol, tuneGrid = gbm_grid)

```

```
bank_gbm
```

```

## Stochastic Gradient Boosting
##
## 8102 samples
##    26 predictor
##    2 classes: 'Attrited Customer', 'Existing Customer'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 6481, 6482, 6482, 6481, 6482
## Resampling results across tuning parameters:
##
##     shrinkage  interaction.depth  n.trees  Accuracy  Kappa
##     0.005       1                  500      0.8805230  0.3959710
##     0.005       1                  1000     0.9005180  0.5316808
##     0.005       1                  2000     0.9216243  0.6549555
##     0.005       4                  500      0.9302644  0.7020339
##     0.005       4                  1000     0.9507540  0.8020509
##     0.005       4                  2000     0.9645777  0.8624437
##     0.005       7                  500      0.9454467  0.7779609
##     0.005       7                  1000     0.9606280  0.8456273
##     0.005       7                  2000     0.9677868  0.8757568
##     0.005       10                 500      0.9522348  0.8083602
##     0.005       10                 1000     0.9630965  0.8562810
##     0.005       10                 2000     0.9703784  0.8862184
##     0.020       1                  500      0.9216243  0.6547643
##     0.020       1                  1000     0.9406324  0.7542756
##     0.020       1                  2000     0.9556914  0.8258924
##     0.020       4                  500      0.9637135  0.8590392

```

```

##   0.020      4          1000    0.9702551  0.8864038
##   0.020      4          2000    0.9713658  0.8912955
##   0.020      7          500     0.9686509  0.8793438
##   0.020      7          1000    0.9711188  0.8896359
##   0.020      7          2000    0.9721064  0.8939203
##   0.020      10         500     0.9693910  0.8826213
##   0.020      10         1000    0.9713654  0.8907302
##   0.020      10         2000    0.9732170  0.8982247
##   0.050      1          500     0.9468041  0.7852623
##   0.050      1          1000    0.9577894  0.8357305
##   0.050      1          2000    0.9609982  0.8511037
##   0.050      4          500     0.9705019  0.8876260
##   0.050      4          1000    0.9713654  0.8914989
##   0.050      4          2000    0.9717357  0.8928587
##   0.050      7          500     0.9716126  0.8918270
##   0.050      7          1000    0.9718595  0.8932135
##   0.050      7          2000    0.9724766  0.8957672
##   0.050      10         500     0.9727231  0.8961306
##   0.050      10         1000    0.9716125  0.8919489
##   0.050      10         2000    0.9725997  0.8959325
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 2000, interaction.depth
## = 10, shrinkage = 0.02 and n.minobsinnode = 10.

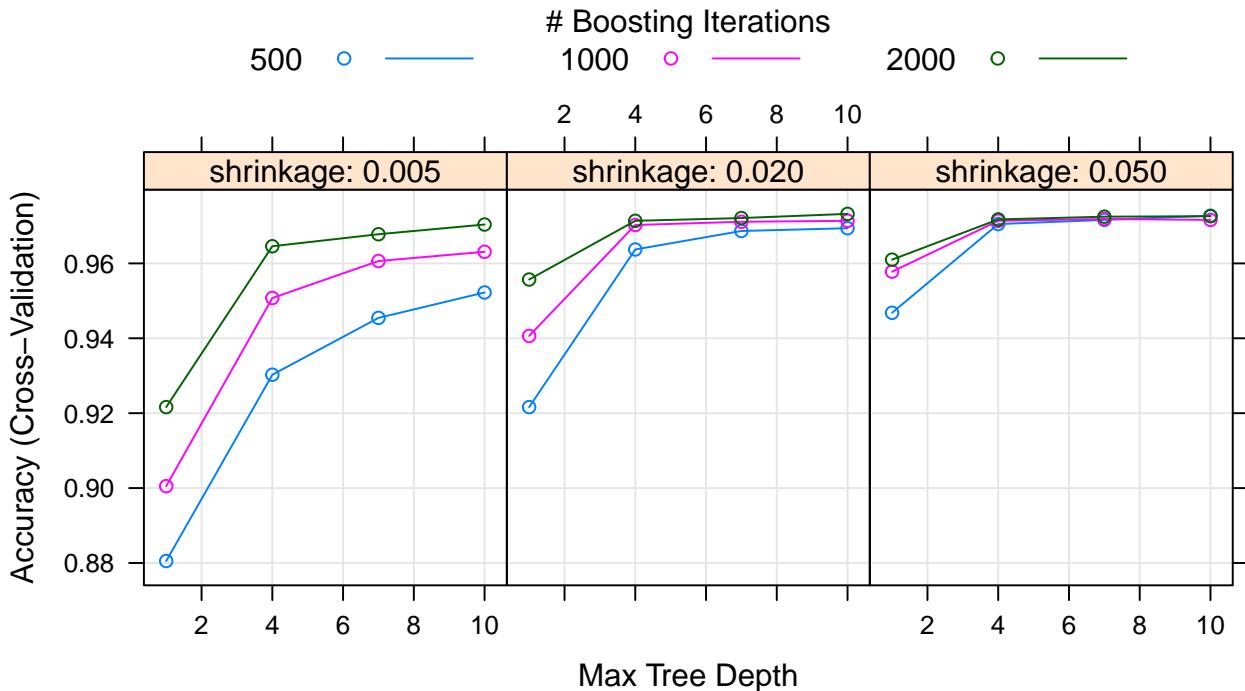
```

Los valores de los hiperparámetros que maximizan la precisión son:

- `n.trees` = 2000
- `interaction.depth` = 10
- `shrinkage` = 0.02
- `n.minobsinnode` = 10

Graficamos estos hiperparámetros para corroborar el resultado obtenido por el algoritmo. A continuación, predecimos con el conjunto de test y construimos la matriz de confusión.

```
plot(bank_gbm)
```



```
gbm_pred <- predict(bank_gbm, bank.test)
gbm_cm <- confusionMatrix(gbm_pred, bank.test$Attrition_Flag)
gbm_cm
```

```
## Confusion Matrix and Statistics
##
##                               Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer              292           27
##   Existing Customer               33        1673
##
##                               Accuracy : 0.9704
##                               95% CI  : (0.962, 0.9773)
##   No Information Rate : 0.8395
##   P-Value [Acc > NIR]  : <2e-16
##
##                               Kappa : 0.8892
##
##   Mcnemar's Test P-Value : 0.5186
##
##                               Sensitivity : 0.8985
##                               Specificity  : 0.9841
##   Pos Pred Value  : 0.9154
##   Neg Pred Value  : 0.9807
##   Prevalence     : 0.1605
##   Detection Rate  : 0.1442
##   Detection Prevalence : 0.1575
##   Balanced Accuracy : 0.9413
##
##   'Positive' Class : Attrited Customer
##
```

Obtenemos una precisión de 97.037037% y 33 falsos negativos.