Trabajo_aprendizaje_supervisado

Ignacio Scuderi

2/1/2022

1.Introducción

El presente trabajo busca analizar los datos relativos al dataset *CreditCard*, realizando en primer lugar un análisis cualitativo, y luego uno cuantitativo de los datos. El mismo se encuentra estructurado de la siguiente forma: luego de la presente introducción, continua la segunda parte correspondiente al importado y la limpieza de los datos. En la tercera parte, se indagan los datos. En la cuarta sección se realiza el análisis cualitativo, mientras que en la quinta se hace lo propio con el cualitativo. Por último, se enumeran algunas conclusiones.

El dataset *CreditCard* contiene datos transversales sobre el historial de crédito de una muestra de solicitantes de un tipo específico de tarjeta de crédito. El dataset posee 1.319 observaciones y 12 variables que se enumeran a continuación:

Card: Factor. ¿Se aceptó la solicitud de la tarjeta de crédito? (character)

Reports: Número de informes derogatorios de importancia. (integer)

Age: Edad en años más doceavos de año. (numeric)

Income: Ingresos anuales (en 10.000 dólares). (numeric)

Share: Relación entre los gastos mensuales de la tarjeta de crédito y los ingresos anuales. (numeric)

Expenditure: Gasto medio mensual de la tarjeta de crédito. (numeric)

Owner: Factor. ¿El individuo es propietario de su vivienda? (character)

Selfemp: Factor. ¿El individuo trabaja por cuenta propia? (character)

Dependents: Número de personas a cargo. (integer)

Months: Meses que vive en el domicilio actual. (integer)

Majorcards: Número de tarjetas de crédito principales. (integer)

Active: Número de cuentas de crédito activas. (integer)

Los datos fueron extraídos de *Greene, W.H. (2003).* Análisis econométrico, quinta edición. Link de los datos: *https://vincentarelbundock.github.io/Rdatasets/doc/AER/CreditCard.html*

2.Importado y limpieza de los datos

En primer lugar se carga el archivo y se procede a realizar una visualización de los datos:

```
df= read.csv("CreditCard.csv")
```

Se procede a eliminar la columna "X" que contiene un índice de las observaciones:

```
df$X= NULL
```

Se corrobora que el dataframe no posee valores *NA*:

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

```
       card     reports         age      income       share expenditure
          0           0           0           0           0           0
      owner     selfemp  dependents      months  majorcards      active
          0           0           0           0           0           0
```

Debido a la ausencia de *NA*, no es necesario eliminar observaciones o realizar imputaciones a dichos valores.

Cambio del formato de las variables *card*, *owner*, y *selfemp* de character a logical de modo de operar con las mismas:

```
str(df)
```

```
'data.frame':   1319 obs. of  12 variables:
 $ card       : chr  "yes" "yes" "yes" "yes" ...
 $ reports    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ age        : num  37.7 33.2 33.7 30.5 32.2 ...
 $ income     : num  4.52 2.42 4.5 2.54 9.79 ...
 $ share      : num  0.03327 0.00522 0.00416 0.06521 0.06705 ...
 $ expenditure: num  124.98 9.85 15 137.87 546.5 ...
 $ owner      : chr  "yes" "no" "yes" "no" ...
 $ selfemp    : chr  "no" "no" "no" "no" ...
 $ dependents : int  3 3 4 0 2 0 2 0 0 0 ...
 $ months     : int  54 34 58 25 64 54 7 77 97 65 ...
 $ majorcards : int  1 1 1 1 1 1 1 1 1 1 ...
 $ active     : int  12 13 5 7 5 1 5 3 6 18 ...
```

```
df$card= ifelse(df$card=="yes", 1, 0)
df$owner= ifelse(df$owner=="yes", 1, 0)
df$selfemp= ifelse(df$selfemp=="yes", 1, 0)
```

Se analizan los principales estadísticos del dataframe:

```
summary(df)
```

```
      card           reports            age              income
 Min.   :0.0000   Min.   : 0.0000   Min.   : 0.1667   Min.   : 0.210
 1st Qu.:1.0000   1st Qu.: 0.0000   1st Qu.:25.4167   1st Qu.: 2.244
 Median :1.0000   Median : 0.0000   Median :31.2500   Median : 2.900
 Mean   :0.7756   Mean   : 0.4564   Mean   :33.2131   Mean   : 3.365
 3rd Qu.:1.0000   3rd Qu.: 0.0000   3rd Qu.:39.4167   3rd Qu.: 4.000
 Max.   :1.0000   Max.   :14.0000   Max.   :83.5000   Max.   :13.500
     share            expenditure          owner           selfemp
 Min.   :0.0001091   Min.   :   0.000   Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.0023159   1st Qu.:   4.583   1st Qu.:0.0000   1st Qu.:0.00000
 Median :0.0388272   Median : 101.298   Median :0.0000   Median :0.00000
 Mean   :0.0687322   Mean   : 185.057   Mean   :0.4405   Mean   :0.06899
 3rd Qu.:0.0936168   3rd Qu.: 249.036   3rd Qu.:1.0000   3rd Qu.:0.00000
 Max.   :0.9063205   Max.   :3099.505   Max.   :1.0000   Max.   :1.00000
   dependents         months         majorcards         active
 Min.   :0.0000   Min.   :  0.00   Min.   :0.0000   Min.   : 0.000
 1st Qu.:0.0000   1st Qu.: 12.00   1st Qu.:1.0000   1st Qu.: 2.000
 Median :1.0000   Median : 30.00   Median :1.0000   Median : 6.000
 Mean   :0.9939   Mean   : 55.27   Mean   :0.8173   Mean   : 6.997
 3rd Qu.:2.0000   3rd Qu.: 72.00   3rd Qu.:1.0000   3rd Qu.:11.000
 Max.   :6.0000   Max.   :540.00   Max.   :1.0000   Max.   :46.000
```

Se observa la existencia de valores muy reducidos para la variable *age*, por lo que se procede a estudiar dichos casos:

```
df %>% filter(age<18)
```

```
  card reports       age income      share expenditure owner selfemp dependents
1    1       0 0.5000000   3.05 0.10172430   258.54920     0       0          1
2    1       0 0.1666667   3.24 0.18436640   497.70580     1       0          3
3    1       0 0.5833333   2.50 0.08317120   173.02330     0       0          0
4    0       0 0.7500000   3.00 0.00040000     0.00000     0       0          0
5    1       0 0.5833333   4.00 0.07266350   242.12830     1       0          3
6    1       1 0.5000000   3.70 0.01063703    32.46416     0       0          0
7    1       0 0.7500000   1.60 0.15419060   205.25420     0       0          0
  months majorcards active
1     94          1      5
2     25          1     16
3    150          1      5
4     18          0      2
5     24          1      4
6    186          0      5
7      1          1      9
```

Al ser de 18 años la edad mínima para obtener una tarjeta de crédito y considerando que sólo hay 7 observaciones que registran una edad inferior, se procede a excluir dichas filas del dataset. De igual modo, se procede a redondear la edad a enteros.

```
df2= df %>% filter(age>=18)
df2$age= round((df2$age),0)
```

Se multiplica la columna income por 10.000 para obtener el valor real de ingreso:

```
df2$income=df2$income*10000
```

Se añade un campo adicional, *years*, de manera de visualizar los datos de la variable *months* con dicho corte temporal:

```
df2$years= round(((df2$months)/12),0)
```

Se procede a indagar los valores únicos de cada variable, así como la cantidad:

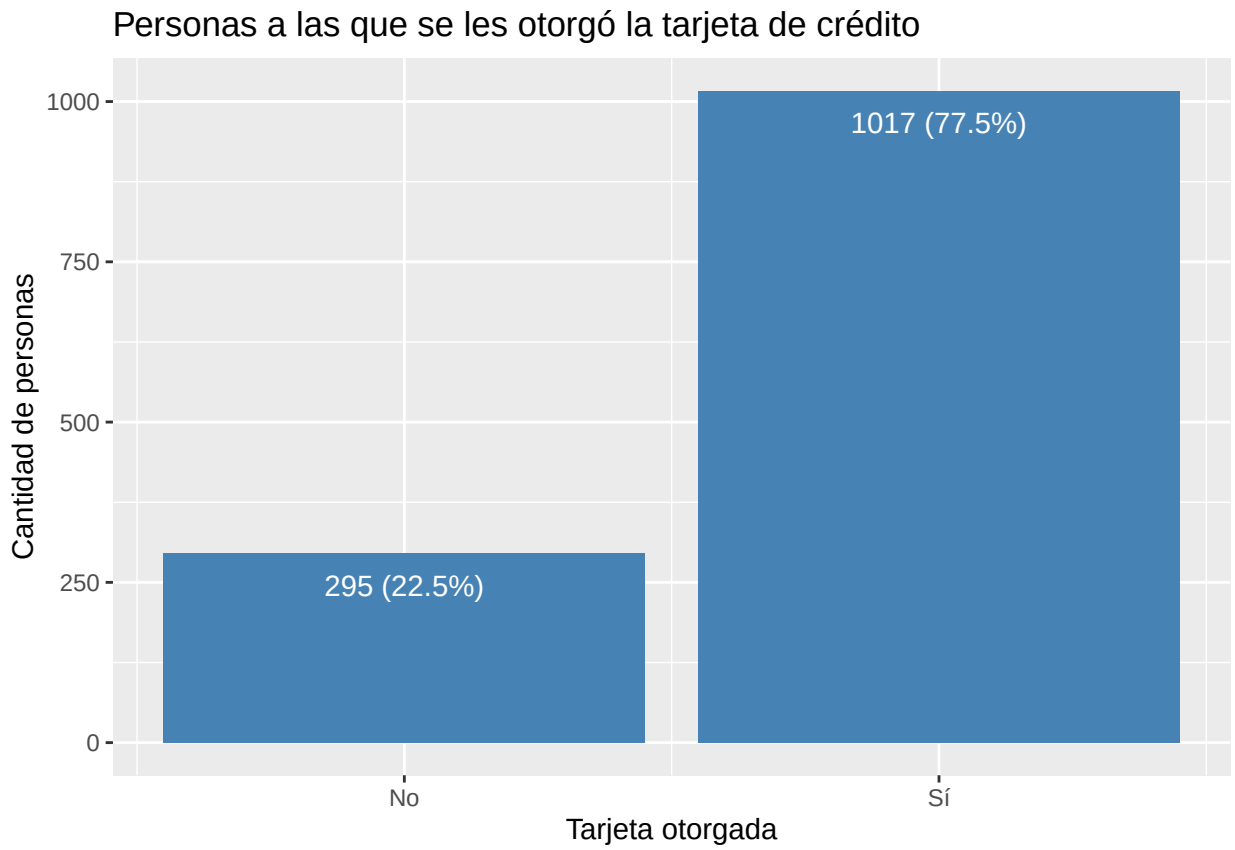
```
#df2 %>% lapply(unique)
```

Un primer barrido de los datos pareciera indicar que están correctamente cargados.

3.Análisis exploratorio

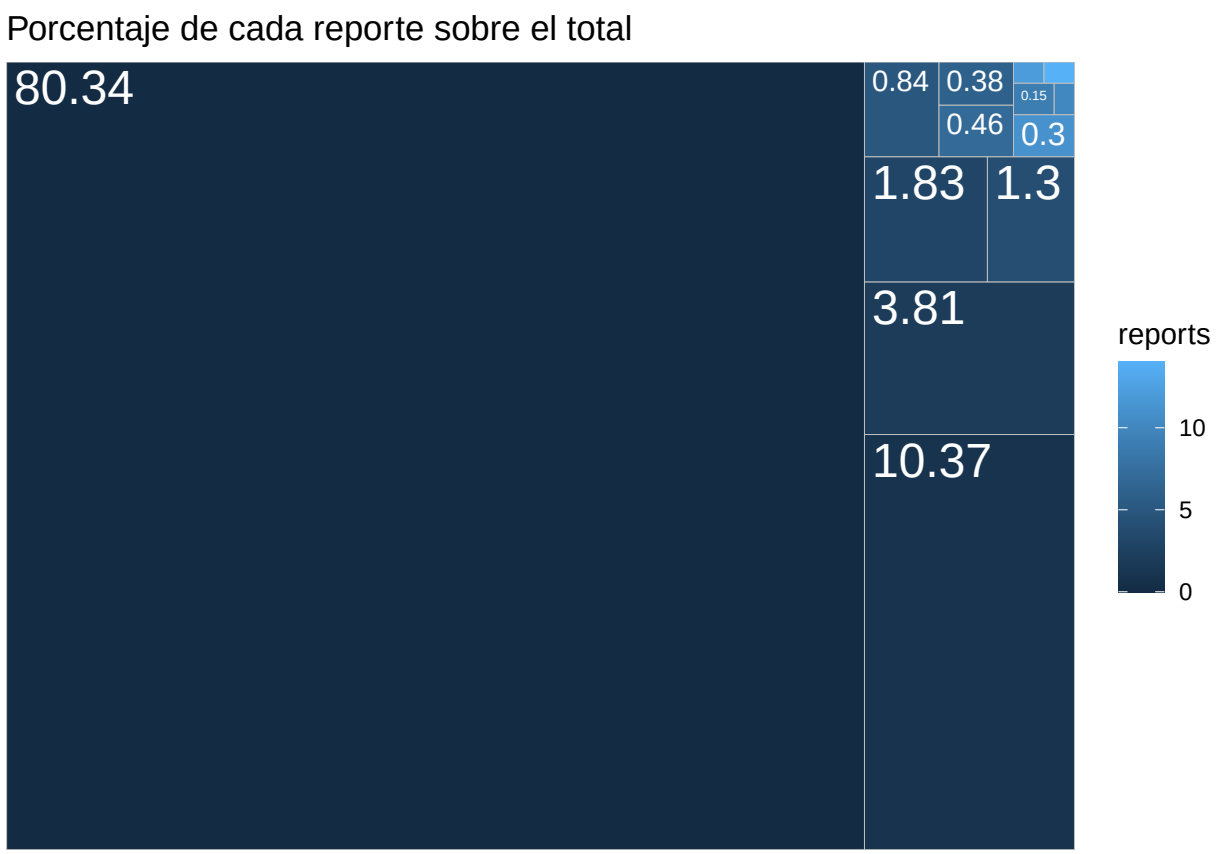
Se procede a analizar cada una de las variables:

Gráfico de histograma de *card*:



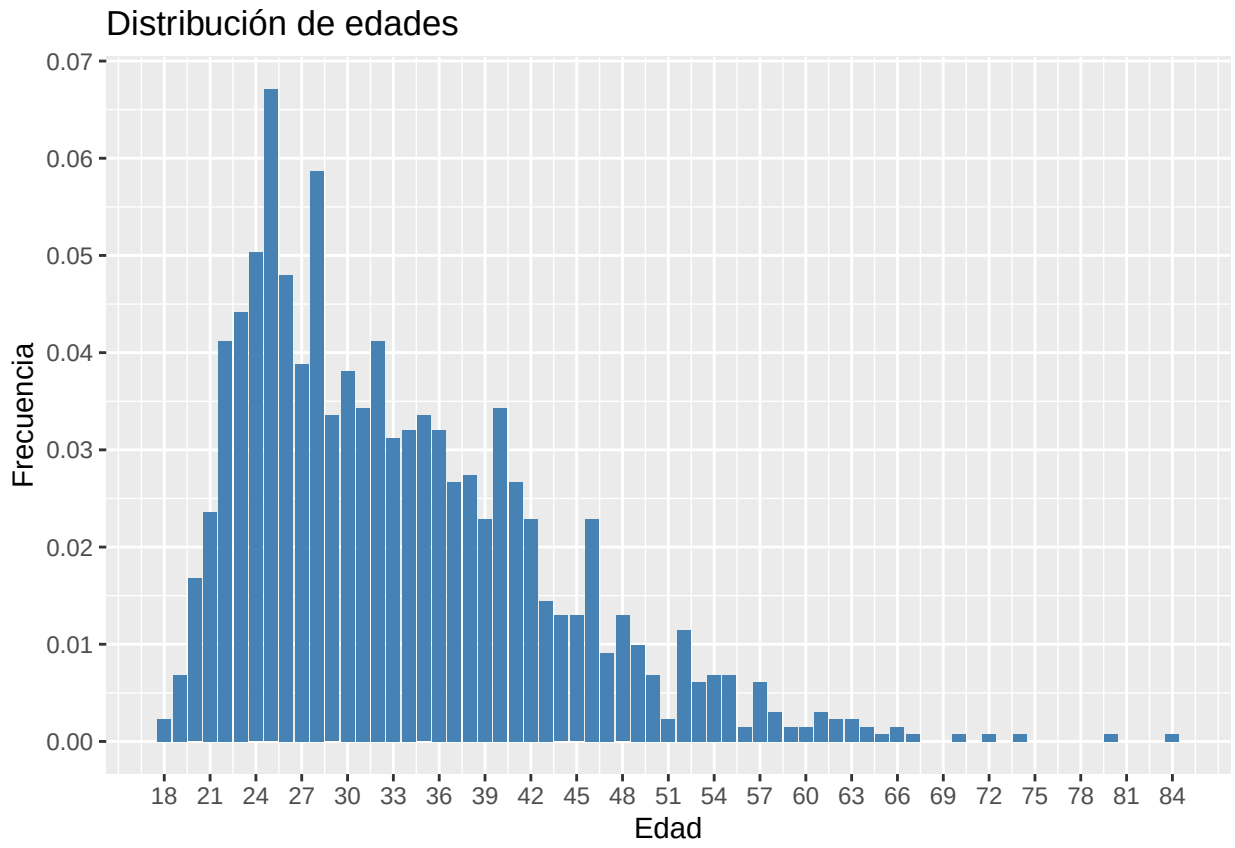
card (variable dependiente): Se releva que 295 solicitudes de tarjeta de crédito (22,5%) han sido denegadas, mientras que 1017 fueron aceptadas (77,5%), por lo que hay una cantidad aceptable para ambas categorias, lo que facilitará el armado de modelos.

Gráfico de mosaico de *reports*:



reports: Se ve a raíz del gráfico anterior que el 80,34% de las personas no poseen informes derogatorios de importancia, mientras que el 10,37% tienen uno, y el 3,81% poseen dos.

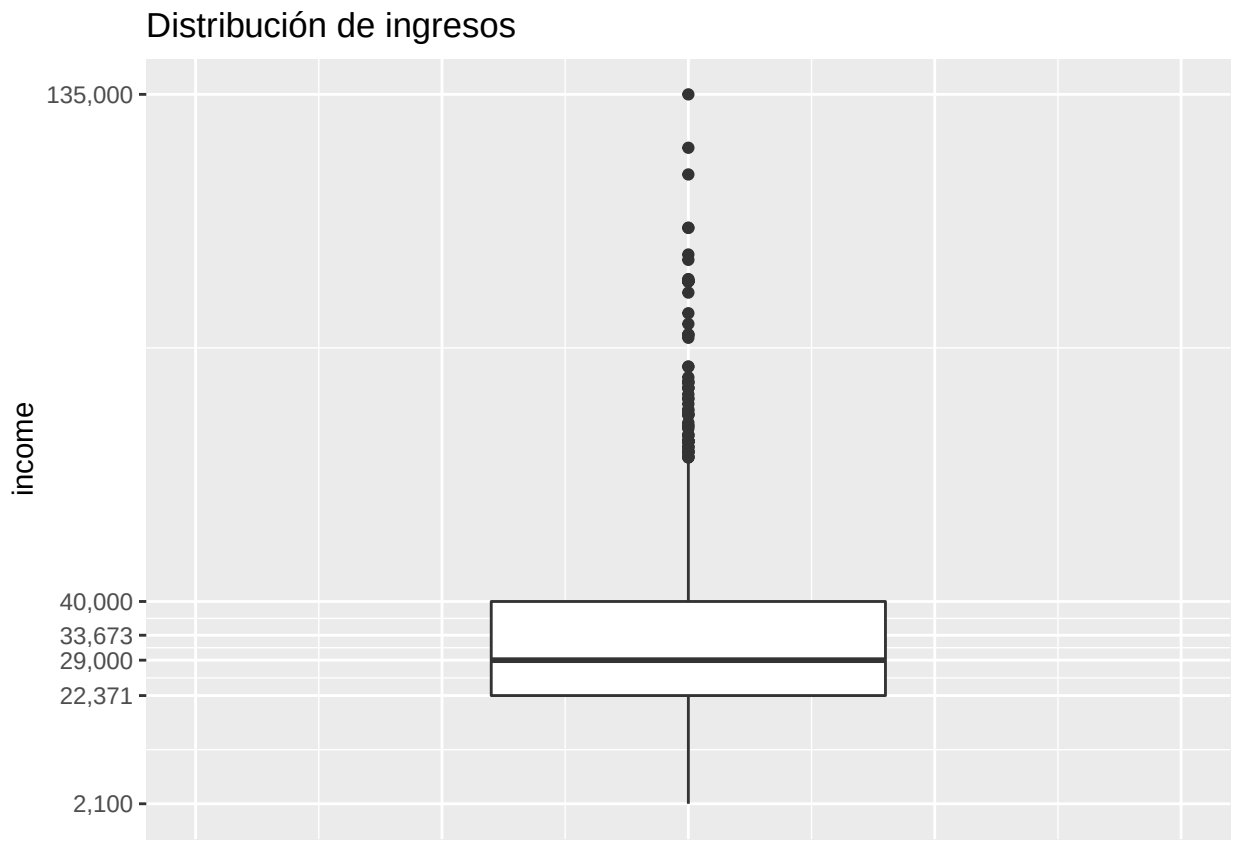
Gráfico de la variable *edad*:




```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   25.00   31.00   33.38   39.00   84.00
```

edad: Analizando la tabla de frecuencias y el gráfico, se observa que la moda es de 25 años, mientras que la media es de 33 años. Se observa asimismo una concentración en edades correspondientes a jóvenes. En este sentido, se pone de manifiesto que más del 50% de los registros son de personas de 31 años o menos. Se ve también que menos del 10% de los valores se corresponden con personas de 47 años o más.

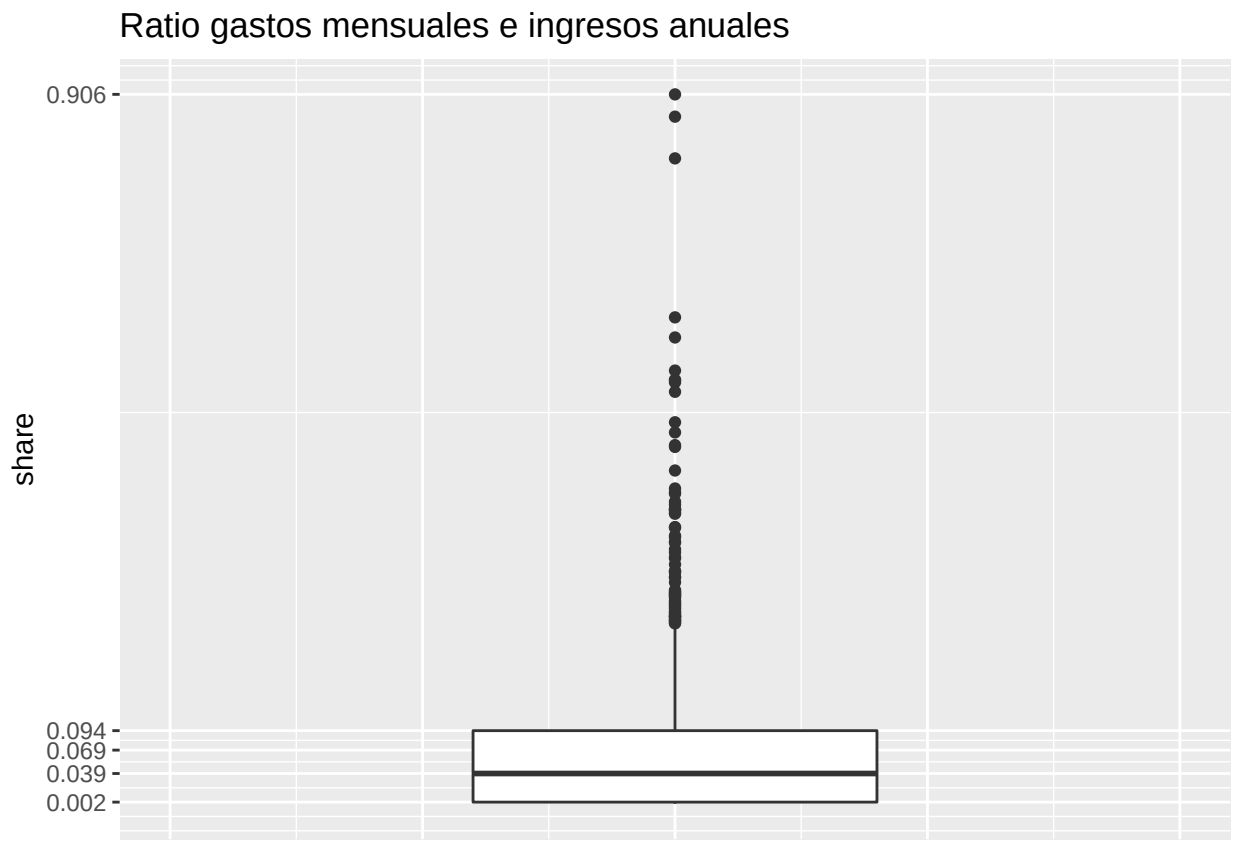
Gráfico de *income*:




```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2100   22371   29000   33673   40000  135000
```

income: El valor mínimo de ingresos es 2.100, mientras que el máximo se ubica en 135.000. Se nota la presencia de varios outliers que se corresponden a sujetos de ingresos altos. La media de ingresos es de 33.673, mientras que el primer rango intercuartílico se ubica en 22.371, la mediana en 29.000, y el tercer rango intercuartílico en 40.000.

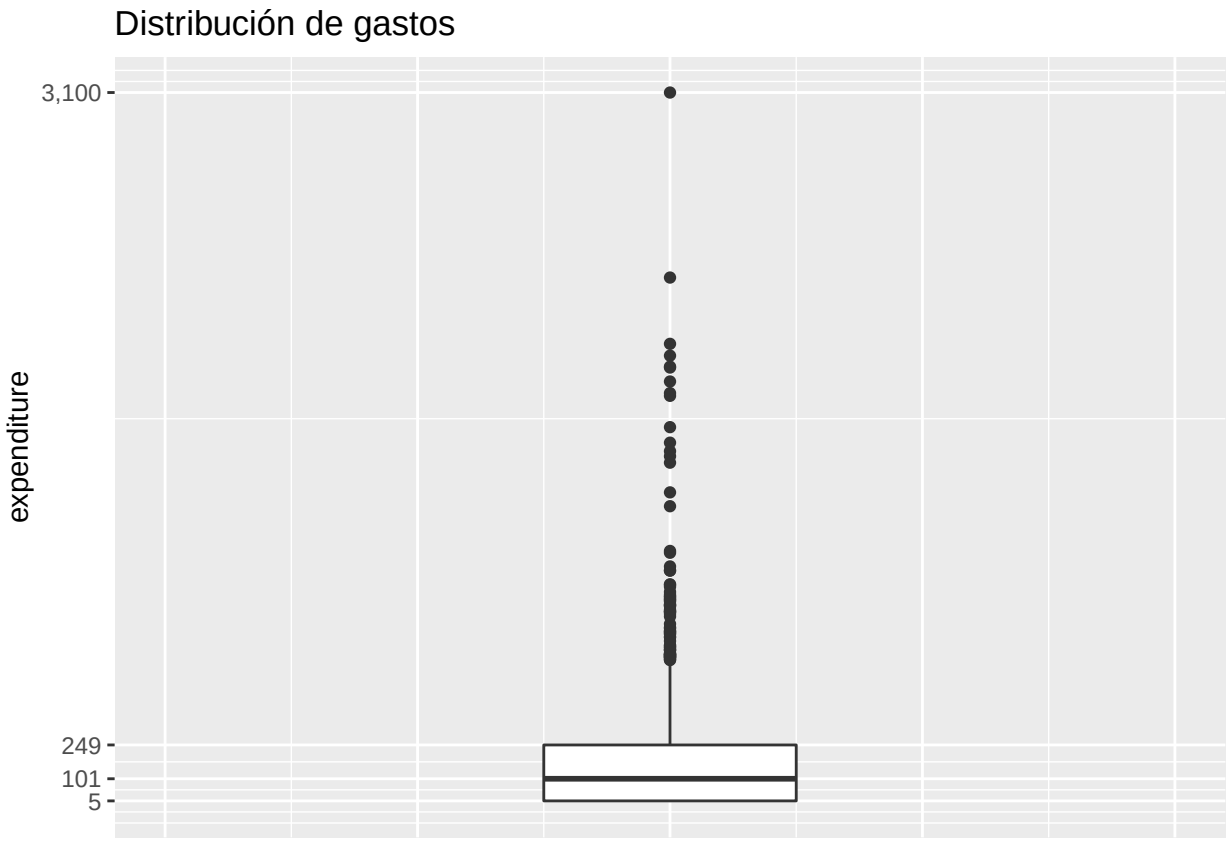
Gráfico de *share*:




```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0001091 0.0022080 0.0387754 0.0686361 0.0935162 0.9063205
```

share: Se observa un diagrama de caja con una relación entre los gastos mensuales de la tarjeta de crédito y los ingresos anuales compacta. La mediana es de 0.038, mientras que la media es de 0.068, empujada esta última hacia arriba debido a algunos valores atípicos con ratios superiores. En este sentido, el valor máximo obtiene un ratio de 0.90.

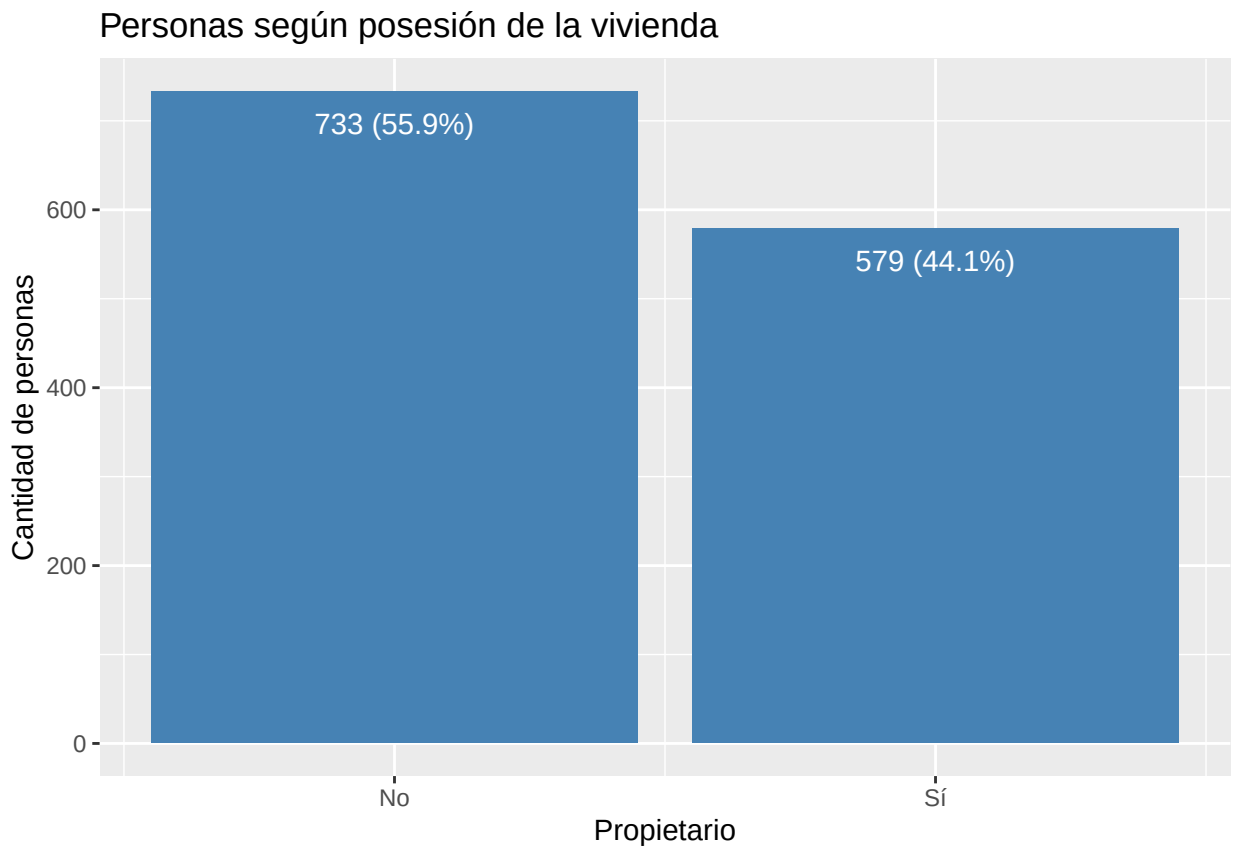
Gráfico de *expenditure*:




```
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  0.000    4.583  101.232  184.970  248.971 3099.505
```

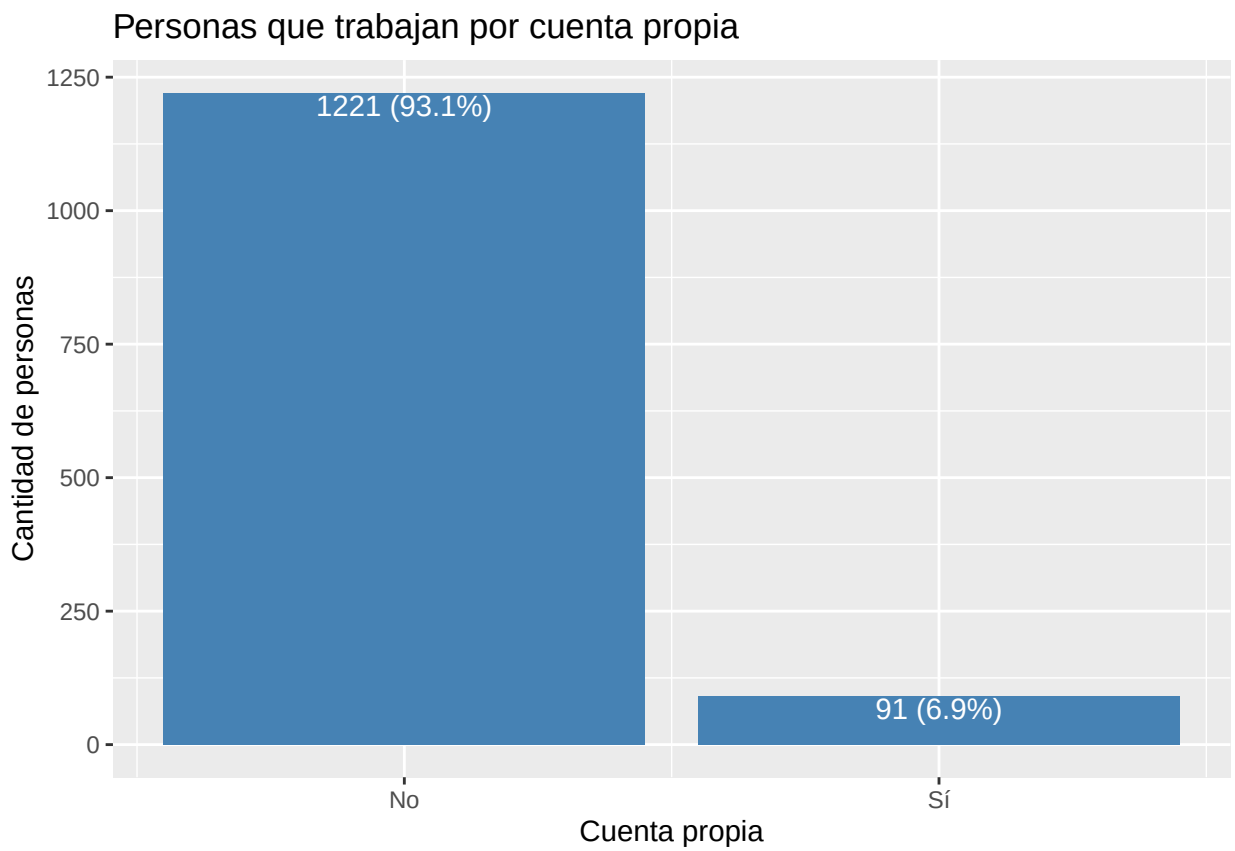
expenditure: el gasto medio mensual mínimo es 0, mientras que el máximo es de 3099,5. El primer rango intercuartílico es 4.5, la media es 101,23, el tercer rango es de 248,97. Por último la media se ubica en 184.97. Se observa un gasto medio mensual reducido y bastante homogeneo.

Gráfico de *owner*:



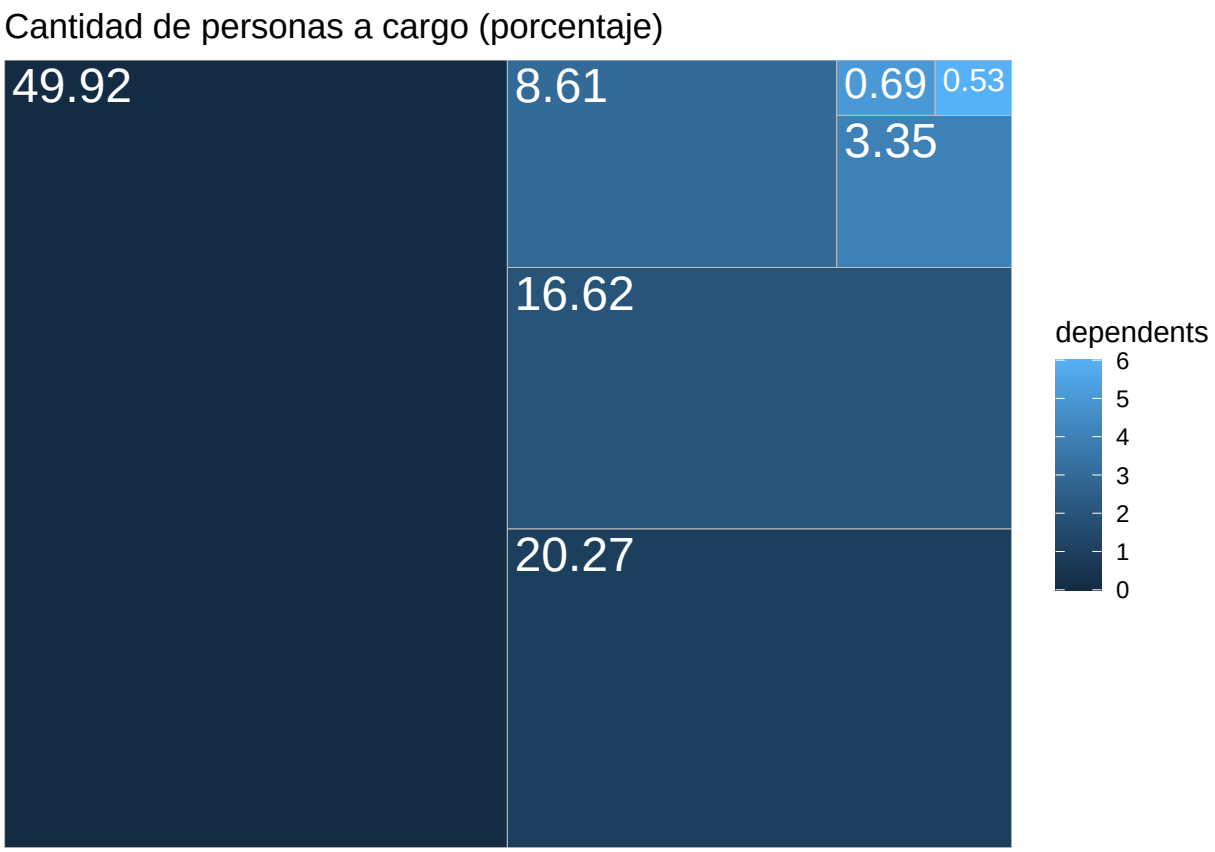
owner: En cuanto a la propiedad de la vivienda, se observa que el 55,9% de las personas no son propietarias, mientras que el 44,1% lo es.

Gráfico de *selfemp*:



selfemp: En cuanto al cuentapropismo, se observa que la abrumadora mayoria de las personas no registran dicha condición (93.1%).

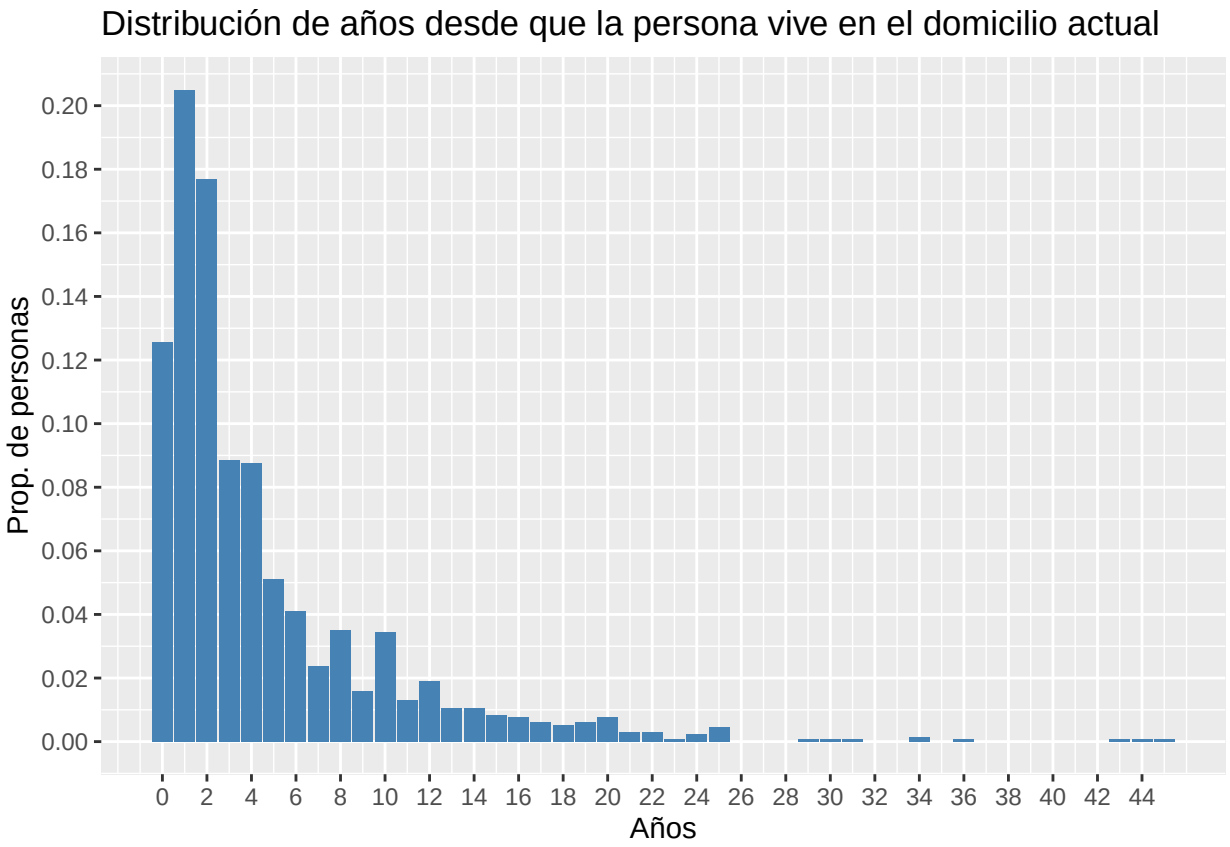
Gráfico de *dependents*:




```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0000  0.0000  1.0000  0.9939  2.0000  6.0000
```

dependents: se releva que casi el 50% de las observaciones se corresponden a gente que no tiene personas a cargo, mientras que poco más del 20% posee una persona a cargo, 16,6% a dos, y 8,6% a tres.

Gráfico de *years*:

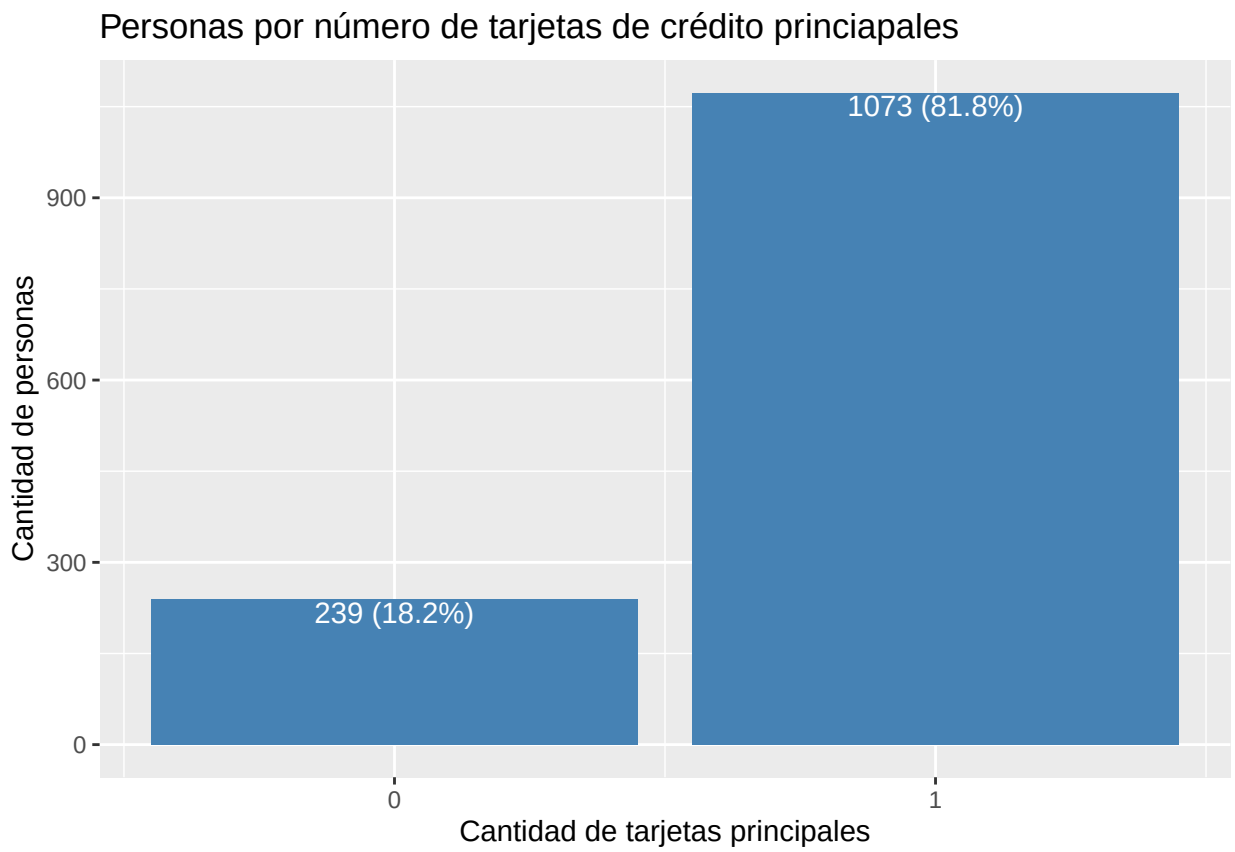



```
   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
0.13 0.21 0.18 0.09 0.09 0.05 0.04 0.02 0.04 0.02 0.03 0.01 0.02 0.01 0.01 0.01
  16   17   18   19   20   21   22   23   24   25   29   30   31   34   36   43
0.01 0.01 0.01 0.01 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  44   45
0.00 0.00
```

```
   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
0.13 0.33 0.51 0.60 0.68 0.73 0.78 0.80 0.83 0.85 0.88 0.90 0.92 0.93 0.94 0.95
  16   17   18   19   20   21   22   23   24   25   29   30   31   34   36   43
0.95 0.96 0.97 0.97 0.98 0.98 0.99 0.99 0.99 0.99 0.99 0.99 1.00 1.00 1.00 1.00
  44   45
1.00 1.00
```

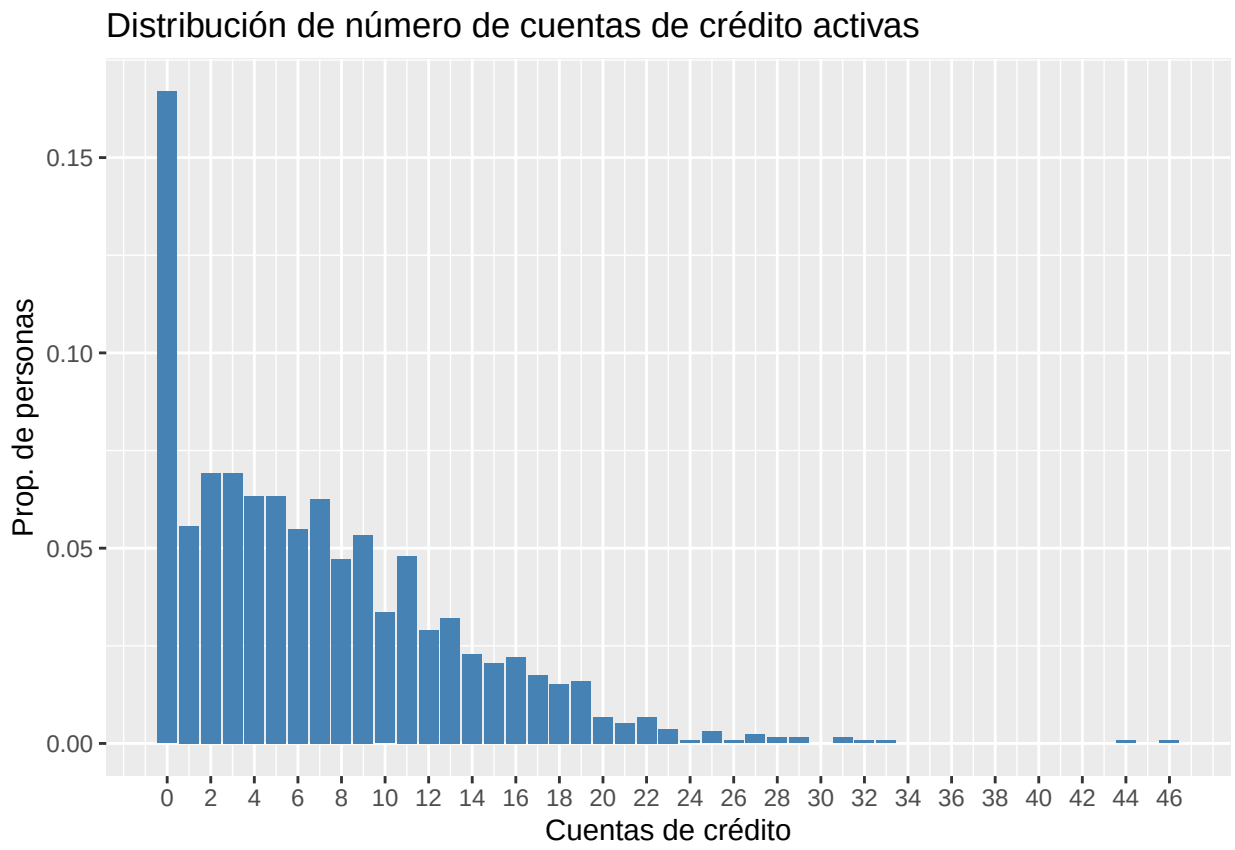
years: Se observa que la moda es de un año, mientras que la mediana se corresponde con personas que han habitado 2 años el mismo domicilio. Asimismo, casi el 80% de los valores se corresponden a personas que han vivido 7 años o menos en el mismo domicilio. Si bien se registran valores extremos, personas que han vivido más de 25 años o más en el mismo domicilio, estos representan menos del 1% de las observaciones.

Gráfico de *majorcards*:



majorcards: se observa que el 81,8% de las personas tienen una tarjeta principal, mientras que el 18,2% no poseen ninguna.

Gráfico de *active*:




```
   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
0.17 0.06 0.07 0.07 0.06 0.06 0.05 0.06 0.05 0.05 0.03 0.05 0.03 0.03 0.02 0.02
  16   17   18   19   20   21   22   23   24   25   26   27   28   29   31   32
0.02 0.02 0.02 0.02 0.01 0.01 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  33   44   46
0.00 0.00 0.00
```

```
   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
0.17 0.22 0.29 0.36 0.42 0.49 0.54 0.61 0.65 0.71 0.74 0.79 0.82 0.85 0.87 0.89
  16   17   18   19   20   21   22   23   24   25   26   27   28   29   31   32
0.91 0.93 0.95 0.96 0.97 0.97 0.98 0.99 0.99 0.99 0.99 0.99 0.99 1.00 1.00 1.00
  33   44   46
1.00 1.00 1.00
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   2.000   6.000   6.999  11.000  46.000
```

active: Se observa que el 16,6% de las personas no poseen cuentas de crédito activas (moda), mientras que la mediana se ubica en 6 cuentas. Por último, la media es de 7 cuentas.

4. Análisis cualitativo

En primer lugar debemos analizar la correlación existente entre las variables, para asegurarnos que no hay multicolinealidad perfecta entre las mismas.

```
hetcor(df2)[1]
```

```
$correlations
                    card      reports          age      income       share
card         1.000000000 -0.453567577  0.003256061  0.09475555  0.38742740
reports     -0.453567577  1.000000000  0.040657171  0.01054498 -0.15857393
age          0.003256061  0.040657171  1.000000000  0.33145672 -0.11773028
income       0.094755546  0.010544976  0.331456725  1.00000000 -0.05367639
share        0.387427403 -0.158573929 -0.117730284 -0.05367639  1.00000000
expenditure  0.365434629 -0.136272542  0.015286662  0.28157133  0.83873070
owner        0.147758963 -0.053783488  0.374904841  0.32433074 -0.01685859
selfemp     -0.054173911  0.018506070  0.098300457  0.11208268 -0.07874442
dependents  -0.037773282  0.020254830  0.218977359  0.31740910 -0.08468480
months      -0.001985794  0.048445601  0.453737731  0.13051897 -0.05428677
majorcards   0.105297159 -0.006620495  0.005839761  0.10810229  0.04861962
active       0.079472626  0.208139368  0.185816629  0.18109389 -0.02552547
years       -0.002120207  0.049166484  0.452615301  0.13028147 -0.05554308
            expenditure       owner      selfemp   dependents       months
card         0.36543463  0.14775896 -0.054173911 -0.037773282 -0.001985794
reports     -0.13627254 -0.05378349  0.018506070  0.020254830  0.048445601
age          0.01528666  0.37490484  0.098300457  0.218977359  0.453737731
income       0.28157133  0.32433074  0.112082684  0.317409096  0.130518966
share        0.83873070 -0.01685859 -0.078744417 -0.084684798 -0.054286767
expenditure  1.00000000  0.09168941 -0.035587911  0.050464933 -0.028116533
owner        0.09168941  1.00000000  0.041329929  0.305946458  0.242505197
selfemp     -0.03558791  0.04132993  1.000000000  0.042235371  0.066467420
dependents   0.05046493  0.30594646  0.042235371  1.000000000  0.049151089
months      -0.02811653  0.24250520  0.066467420  0.049151089  1.000000000
majorcards   0.07528103  0.06153573  0.004484547  0.007194003 -0.039544776
active       0.05315996  0.27416642  0.029499206  0.105743820  0.101593570
years       -0.02877484  0.24291898  0.067593793  0.049353071  0.998807191
             majorcards      active        years
card         0.105297159  0.07947263 -0.002120207
reports     -0.006620495  0.20813937  0.049166484
age          0.005839761  0.18581663  0.452615301
income       0.108102285  0.18109389  0.130281471
share        0.048619620 -0.02552547 -0.055543080
expenditure  0.075281028  0.05315996 -0.028774838
owner        0.061535729  0.27416642  0.242918984
selfemp      0.004484547  0.02949921  0.067593793
dependents   0.007194003  0.10574382  0.049353071
months      -0.039544776  0.10159357  0.998807191
majorcards   1.000000000  0.11819395 -0.040730058
active       0.118193947  1.00000000  0.100857812
years       -0.040730058  0.10085781  1.000000000
```

En base a los resultados obtenidos en el cuadro anterior, vemos que hay multicolinealidad perfecta entre las variables *expenditure* y *share*, por lo que procedemos a eliminar la primera variable. Se ha tomado esta decisión, ya que que *share* nos provee de más información al relacionar ingresos y gastos. Podría ocurrir que alguien posea muchos gastos, pero que esto esté justificado por ingresos que también lo sean, y dicha relación

no es recogida por *expenditure*. Asimismo y como era de esperar, hay multicolinealidad perfecta entre la variable *months* y el campo calculado *years*. Se decide eliminar el campo *months* por considerar más útil ver la información agrupada por años.

```
df3= df2
df2 = subset(df2, select = -c(expenditure, months))
write.csv(df2,"df2.csv")
df2= read.csv("df2.csv")
df2$X= NULL
```

En primer lugar se deben crear dos subsets, uno de *training* y otro de *testing*, para entrenar y testear los modelos. Asimismo, se aplica una semilla al inicio, de manera de hacer los resultados replicables.

Divido los datos de la siguiente manera:

- 50% train
- 50% test

```
set.seed(2022)
tamano_sample= floor(0.5*nrow(df2))
training <- sample(seq_len(nrow(df2)), size = tamano_sample)
df2.train= df2[training, ]
df2.test=df2[-training, ]
```

Se ha decidido predecir si a la persona se le ha otorgado tarjeta de crédito o no.

4.1 Evaluación de modelos

Los modelos a utlizar en esta parte del trabajo son:

-Linear Discriminant Analysis (LDA)

-Logistic Regression

-Quadratic Discriminant Analysis (QDA)

-Naive Bayes

-Árboles

-Random Forest

-Boosting

A continuación se detalla una breve descripción de cada modelo.

Linear Discriminant Analysis (LDA): El Análisis Discriminante Lineal es un método de clasificación supervisado de variables cualitativas. Haciendo uso del teorema de Bayes, LDA estima la probabilidad de que una observación, dado un determinado valor de los predictores, pertenezca a cada una de las clases de la variable cualitativa, asignando la observación a la clase k para la que la probabilidad predicha es mayor.

Logistic Regression: Este método permite estimar la probabilidad de una variable cualitativa binaria en función de una variable cuantitativa. La regresión logística permite calcular la probabilidad de que la variable dependiente pertenezca a cada una de las dos categorías en función del valor que adquiera la variable independiente.

Quadratic Discriminant Analysis (QDA): Esta metodología es similar a LDA, con la diferencia de que el QDA considera que cada clase k tiene su propia matriz de covarianza y, como consecuencia, la función discriminante toma forma cuadrática.

Naive Bayes: El algoritmo Bayes naive es un algoritmo de clasificación de variables cualitativas basado en los teoremas de Bayes. Este algoritmo es llamado "ingenuo" porque calcula las probabilidades condicionales por separado, como si fueran independientes una de otra. Una vez que se obtienen las probabilidades condicionales

por separado, se calcula la probabilidad conjunta de todas ellas, mediante un producto, para determinar la probabilidad de que pertenezca a la categoría. Luego se itera dicho proceso para cada observación.

Árboles: Mediante este modelo se intenta predecir una variable dependiente a partir de variables independientes. Existen árboles de clasificación (variable discreta) y árboles de regresión (variable continua). Lo que hace este algoritmo es encontrar la variable independiente que mejor separa nuestros datos en grupos, que corresponden con las categorías de la variable objetivo. Esta mejor separación es expresada con una regla. A cada regla corresponde un nodo.

Una vez hecho esto, los datos son separados (particionados) en grupos a partir de la regla obtenida. Después, para cada uno de los grupos resultantes, se repite el mismo proceso. Se busca la variable que mejor separa los datos en grupos, se obtiene una regla, y se separan los datos. Hacemos esto de manera recursiva hasta que nos es imposible obtener una mejor separación. Cuando esto ocurre, el algoritmo se detiene. Cuando un grupo no puede ser partido mejor, se le llama nodo terminal u hoja.

Random Forest: Los Random forests promedian múltiples árboles de decisión, entrenados en diferentes partes del mismo set de entrenamiento, con el objetivo de reducir la varianza. Esto se produce a expensas de un pequeño aumento en el sesgo y cierta pérdida de interpretabilidad, pero en general aumenta considerablemente el rendimiento del modelo final.

Boosting: El Boosting es una técnica de ensamble secuencial. En cada nuevo paso intentará aprender de los errores cometidos en los pasos previos. Trabaja sobre los errores del modelo anterior o bien usándolos para cambiar la ponderación en el siguiente modelo o bien entrenando un modelo que prediga los mismos.

Linear Discriminant Analysis (LDA)

Se comienza entrenando el modelo LDA y luego prediciendo con los datos de testeo:

```
set.seed(2022)
lda.fit=lda(card~.,data=df2,subset=training)
lda.pred=predict(lda.fit,newdata=df2.test)
```

Luego se observan y analizan los resultados que evaluan el modelo:

```
df2.test$card= as.factor(df2.test$card)
conf_matrix_lda <- confusionMatrix(lda.pred$class, df2.test$card, mode = "prec_recall", positive = "1")
conf_matrix_lda
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  57   2
         1  82 515

               Accuracy : 0.872
                 95% CI : (0.8439, 0.8966)
    No Information Rate : 0.7881
    P-Value [Acc > NIR] : 1.829e-08

                  Kappa : 0.5144

 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.8626
                 Recall : 0.9961
                     F1 : 0.9246
             Prevalence : 0.7881
```

```
         Detection Rate : 0.7851
   Detection Prevalence : 0.9101
      Balanced Accuracy : 0.7031

       'Positive' Class : 1
```

Una primera aproximación a la evaluación del modelo nos la da la métrica *Accuracy*:

$$\text{Accuracy} = (\text{Verdaderos positivos} + \text{Verdaderos negativos}) \ / \ \text{Total}$$

En este caso se observa que dicha metrica obtiene un valor de 0.872, lo cual a priori es un buen indicador de nuestro modelo. Se pone de manifiesto al relevar el valor de la métrica *No Information Rate (NIR)* que el modelo realiza una predicción sustancialmente mejor que la que se podría obtener producto del azar (*NIR* calcula el porcentaje de casos mayoritarios dentro de la variable dependiente).

A continuación se evalua la métrica de *Recall (TPR)*:

$$\text{Recall} = \text{Verdaderos positivos} \ / \ (\text{Verdaderos positivos} + \text{Falsos negativos})$$

Es una forma de analizar la tasa de verdaderos positivos. Si su valor es bajo, quiere decir que hay muchos falsos negativos, es decir gente a la que se podría haberle dado tarjeta y se le negó. La tasa penaliza no detectar los casos positivos. En este caso, se observa que el *Recall* es de 0.996, por lo que se puede concluir que el modelo casi siempre detecta los casos positivos correctamente.

Posteriormente se observa la métrica *Precision*:

$$\text{Precision} = \text{Verdaderos positivos} \ / \ (\text{Verdaderos positivos} + \text{Falsos positivos})$$

Esta métrica es de vital importancia en nuestro caso, se desea tener un *Precision* alto. Si se obtiene un valor reducido, querrá decir que hay muchos falsos positivos, es decir que le hemos dado una tajeta a gente a la que no deseabamos otorgarsela. Esto posee un impacto económico muy alto, por lo es imprescindible evitarlo. Se observa que dicha métrica tiene un valor de 0.862. Es un valor elevado, similar a lo obtenido en *Accuracy*.

Analizo finalmente la métrica de *Balanced accuracy*:

$$\text{Balanced accuracy} = (\text{TPR} + \text{TNR}) \ / \ 2$$

Esta métrica tiene en consideración el desbalance entre casos positivos y negativos en la variable dependiente. En el dataset seleccionado hay más casos con tarjeta que sin tarjeta, por lo que esta forma de calcular rectifica dicho problema. El *Balanced accuracy* es de 0.703, sensiblemente inferior al *Accuracy*.

Logistic regression

En primer término se procede a entrenar el modelo y a evaluar sus coeficientes:

```
set.seed(2022)
glm.fit=glm(card~.,data=df2.train,
            family=binomial)
summary(glm.fit)
```

```
Call:
glm(formula = card ~ ., family = binomial, data = df2.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
```

```
-1.629   0.000   0.000   0.000   2.891

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.570e+00  1.797e+00  -4.212 2.53e-05 ***
reports     -2.561e+00  1.149e+00  -2.230   0.0258 *
age          8.713e-02  4.593e-02   1.897   0.0578 .
income       5.323e-05  2.692e-05   1.977   0.0480 *
share        3.406e+03  7.960e+02   4.279 1.88e-05 ***
owner        2.620e-02  8.623e-01   0.030   0.9758
selfemp      1.760e+00  9.758e-01   1.803   0.0713 .
dependents  -1.019e+00  5.043e-01  -2.020   0.0434 *
majorcards  -8.699e-01  8.101e-01  -1.074   0.2829
active       9.547e-02  4.668e-02   2.045   0.0408 *
years       -7.737e-02  7.183e-02  -1.077   0.2814
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 719.680  on 655  degrees of freedom
Residual deviance:  56.468  on 645  degrees of freedom
AIC: 78.468

Number of Fisher Scoring iterations: 16
```

Haciendo el summary de *glm.fit* se observa que hay variables no significativas con un alfa al 5%. Se procede a realizar el modelado nuevamente sin incluirlas:

```
glm.fit=glm(card~reports+age+income+share+selfemp+dependents+active,data=df2.train,
            family=binomial)
summary(glm.fit)
```

```
Call:
glm(formula = card ~ reports + age + income + share + selfemp +
    dependents + active, family = binomial, data = df2.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.719   0.000   0.000   0.000   2.952

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.071e+00  1.464e+00  -4.831 1.36e-06 ***
reports     -2.190e+00  1.064e+00  -2.058   0.0396 *
age          5.101e-02  2.777e-02   1.837   0.0662 .
income       5.189e-05  2.609e-05   1.989   0.0467 *
share        3.165e+03  7.415e+02   4.269 1.97e-05 ***
selfemp      1.538e+00  9.117e-01   1.687   0.0916 .
dependents  -8.780e-01  4.623e-01  -1.899   0.0575 .
active       8.178e-02  4.335e-02   1.886   0.0592 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 719.68  on 655  degrees of freedom
Residual deviance:  58.99  on 648  degrees of freedom
AIC: 74.99

Number of Fisher Scoring iterations: 16
```

Se pone de manifiesto que en el nuevo modelo son todas variables relevantes al 5% de significatividad.

Se procede a realizar la predicción con los datos de testeo:

```
glm.probs = predict(glm.fit, newdata = df2.test, type = "response")
glm.pred = as.factor(ifelse(glm.probs>0.5,1,0))
```

Se evalúa la matriz de confusión del modelo:

```
card.test = as.factor(df2.test$card)

conf_matrix_glm <- confusionMatrix(glm.pred, card.test, mode = "prec_recall", positive = "1")
conf_matrix_glm
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 136  13
         1   3 504

               Accuracy : 0.9756
                 95% CI : (0.9607, 0.986)
    No Information Rate : 0.7881
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9288

 Mcnemar's Test P-Value : 0.02445

              Precision : 0.9941
                 Recall : 0.9749
                     F1 : 0.9844
             Prevalence : 0.7881
         Detection Rate : 0.7683
   Detection Prevalence : 0.7729
      Balanced Accuracy : 0.9766

       'Positive' Class : 1
```

Una primera aproximación a la evaluación del modelo nos la da la métrica *Accuracy*: Se observa un valor de 0.975 elevado, que lo es aún más cuando se observa el *Balanced accuracy*, de 0.976. Por otra parte, el *Recall*, clave en nuestro modelo, tiene un valor de 0.974, mientras que la *Precision* es de 0.9941.

Quadratic Discriminant Analysis

Se comienza entrenando el modelo y luego prediciendo con los datos de testeo:

```r
set.seed(2022)
qda.fit=qda(card~.,data=df2, subset=training)
qda.pred=predict(qda.fit,newdata=df2.test)

conf_matrix_qda <- confusionMatrix(qda.pred$class, df2.test$card, mode = "prec_recall", positive = "1")
conf_matrix_qda
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 135  13
         1   4 504

               Accuracy : 0.9741
                 95% CI : (0.9588, 0.9848)
    No Information Rate : 0.7881
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9242

 Mcnemar's Test P-Value : 0.05235

              Precision : 0.9921
                 Recall : 0.9749
                     F1 : 0.9834
             Prevalence : 0.7881
         Detection Rate : 0.7683
   Detection Prevalence : 0.7744
      Balanced Accuracy : 0.9730

       'Positive' Class : 1
```

Se observa un valor elevado de *Accuracy* y *Balanced accuracy*, 0.9741 y 0.9730 respectivamente. De igual modo, el *Recall* y la *Precision*, con valores de 0.9749 y 0.9921, demuestran valores muy altos.

Naive Bayes

Se entrena el modelo con los datos de *training*, luego se predice con los datos de *testing*, y por último se evalúa la matriz de confusión:

```r
set.seed(2022)
nb.fit <- naiveBayes(card~. , data = df2 , subset = training )
nb.pred <- predict (nb.fit , newdata=df2.test)


conf_matrix_nb <- confusionMatrix(nb.pred, df2.test$card, mode = "prec_recall", positive = "1")
conf_matrix_nb
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 135  17
```

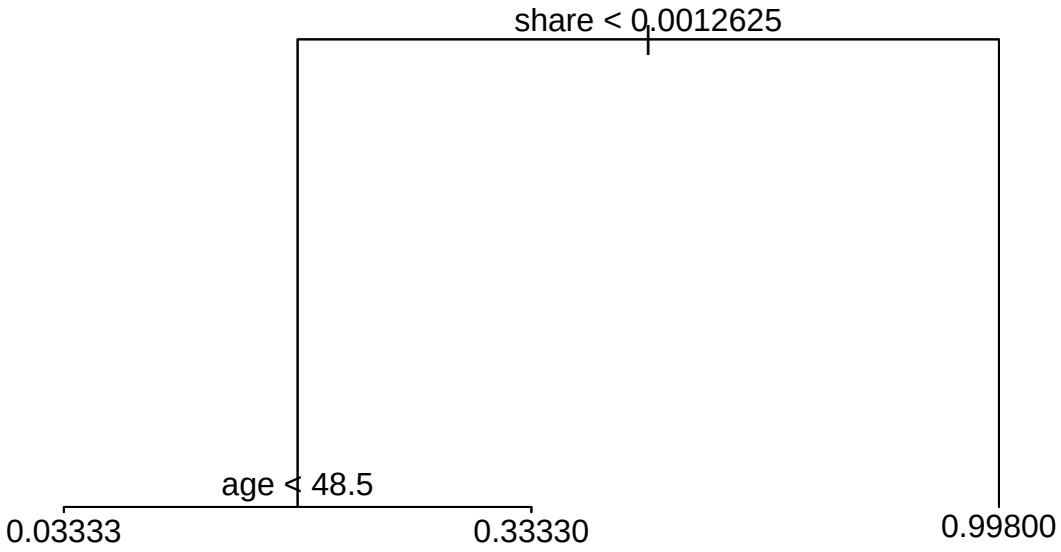
```
         1   4 500

               Accuracy : 0.968
                 95% CI : (0.9515, 0.9801)
    No Information Rate : 0.7881
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9073

 Mcnemar's Test P-Value : 0.008829

              Precision : 0.9921
                 Recall : 0.9671
                     F1 : 0.9794
             Prevalence : 0.7881
         Detection Rate : 0.7622
   Detection Prevalence : 0.7683
      Balanced Accuracy : 0.9692

       'Positive' Class : 1
```

Se observan buenas métricas para el presente modelo. El *Accuracy* es de 0.968, valor que se mantiene practicamente idéntico al observar el *Balanced accuracy*, de 0.969. La *Precision* es muy elevada, de 0.9921, siendo esta una métrica central del análisis. El *Recall* es muy elevado también, de 0.9671.

Árboles

A continuación se procede a entrenar, predecir, y evaluar, el modelo de árboles de clasificación:

```
set.seed(2022)
tree.fit <- tree(card~., df2, subset = training)

plot(tree.fit)
text(tree.fit, pretty=0)
```




```
summary(tree.fit)
```

```
Regression tree:
tree(formula = card ~ ., data = df2, subset = training)
Variables actually used in tree construction:
[1] "share" "age"
Number of terminal nodes:  3
Residual mean deviance:  0.01403 = 9.165 / 653
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.998000  0.002037  0.002037  0.000000  0.002037  0.966700
```

```
tree.probs = predict(tree.fit, df2.test)
tree.pred = as.factor(ifelse(tree.probs>0.5,1,0))

conf_matrix_tree = confusionMatrix(tree.pred, df2.test$card, mode = "prec_recall", positive = "1")
conf_matrix_tree
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 139  15
         1   0 502

               Accuracy : 0.9771
```

```
               95% CI : (0.9626, 0.9871)
    No Information Rate : 0.7881
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9341

 Mcnemar's Test P-Value : 0.0003006

              Precision : 1.0000
                 Recall : 0.9710
                     F1 : 0.9853
             Prevalence : 0.7881
         Detection Rate : 0.7652
   Detection Prevalence : 0.7652
      Balanced Accuracy : 0.9855

       'Positive' Class : 1
```

Simplemente utilizando dos variables y con tres nodos terminales, se ha obtenido un *Accuracy* de 0.977, mientras que el *Balanced accuracy* es de 0.9854. El *Recall* alcanza el valor de 0.970, y la *Precision* es de 1, por lo que el modelo no le da tarjeta a nadie que no debería poseerla. Para el armado del árbol, se puede observar en el plot realizado que la primera variable utilizada fue *share*. Si para dicha variable (que mide ratio *expenditure* e *income*), el valor no se encontraba por debajo de 0.0012625, entonces con probabilidad 0.998 se le otorga tarjeta de crédito. En el caso de que el *share* fuera menor, el árbol decanta a una nueva evaluación con la variable *age*. Si la edad es menor a 48.5, entonces la probabilidad asignada a tener una tarjeta es de 0.033, caso contrario 0.33. Vemos en este sentido que ser joven sería una penalidad.

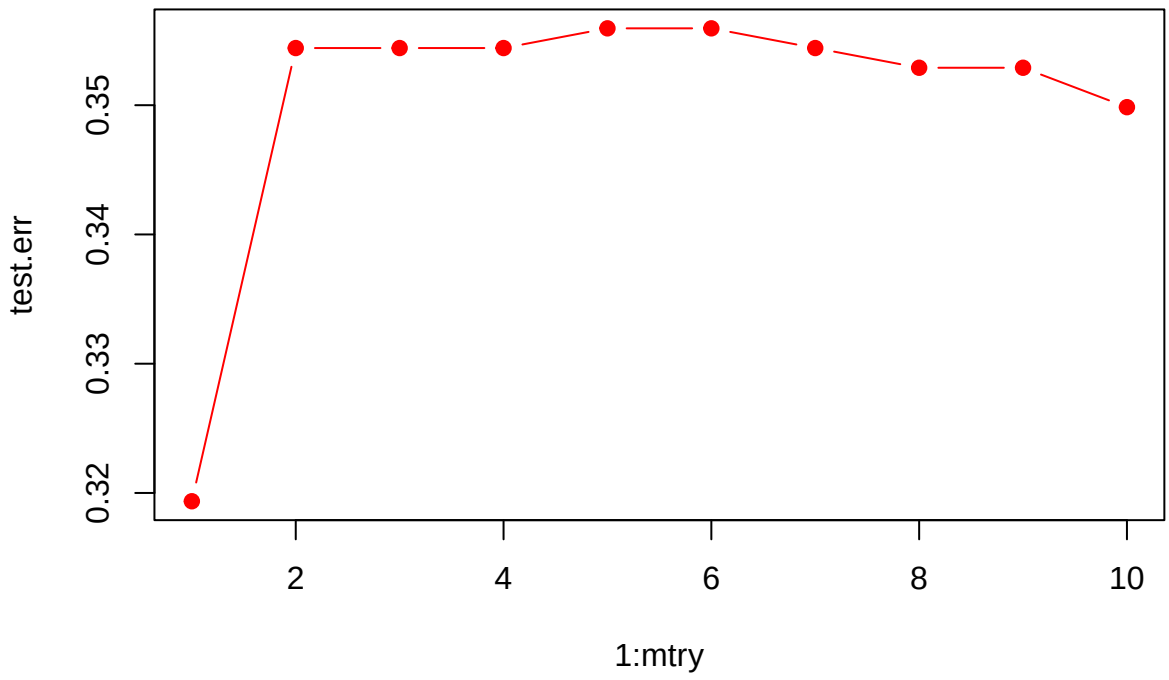
Random Forest

Se procede a modelar un *Random forest* con 10.000 árboles y se calcula la cantidad de variables que minimizan el error (evaluando todas las variables posibles).

```
set.seed(2022)
df2$card=as.factor(df2$card)
rf.fit <- randomForest(card~.,data=df2,subset=training)
NVariables <- ncol(df2)-1
test.err <- double(NVariables)

set.seed(2022)
for(mtry in 1:NVariables){
  fit <- randomForest(card~.,data=df2,subset=training,mtry=mtry,ntree=10000)
  pred <- predict(fit,df2.test)
  test.err[mtry]<- with(df2.test,1-mean(df2$card==pred))
  cat(mtry," ")
}
```

```
1  2  3  4  5  6  7  8  9  10
```

```
plot(1:mtry,test.err,pch=19,col="red",type="b")
```




```
set.seed(2022)
conf_matrix_rf = confusionMatrix(pred, df2.test$card, mode = "prec_recall", positive = "1")
conf_matrix_rf
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 135  15
         1   4 502

               Accuracy : 0.971
                 95% CI : (0.9551, 0.9825)
    No Information Rate : 0.7881
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9157

 Mcnemar's Test P-Value : 0.02178

              Precision : 0.9921
                 Recall : 0.9710
                     F1 : 0.9814
             Prevalence : 0.7881
         Detection Rate : 0.7652
   Detection Prevalence : 0.7713
```

```
      Balanced Accuracy : 0.9711

       'Positive' Class : 1
```

```
fit[9]
```

```
$importance
           MeanDecreaseGini
reports           1.8141501
age               6.0738057
income            5.1299787
share           218.8722155
owner             0.6845090
selfemp           0.4765829
dependents        0.4442087
majorcards        0.2443260
active            2.4200107
years             1.2573216
```

De acuerdo a los resultados obtenidos, se pone de manifiesto que el error se minimiza al utilizar una sola variable. Se observa en la última línea del código que este valor se debe a que la variable *share* posee una relevancia mucho mayor en la determinación de si se le da tarjeta a la persona o no, seguido por la variable *age*, aunque muy por detras.

En la *confusion matrix* se observa que el *Accuracy* del modelo es de 0.971, y que el *Balanced accuracy* es de 0.9711. El *Recall* es de 0.9710, mientras que la *Precision* es de 0.9921.

Bosting

A continuación se hace el modelado con *Boosting*, utilizando CV. Luego se procede a ajustar los parámetros, indicando una profundidad de 1,2, y 4. Se especifican dos *shrinkage*, de 0.01 y 0.001. Asimismo, se realizan pruebas con árboles de tamaño 100 a 10.000, con saltos de a 100. Por último, se especifican dos tamaños mínimos de observaciones permitidas, 10 y 30. Esto nos da como resultado 1200 modelos a testear.

```
set.seed(2022)
df2.test$card=as.factor(df2.test$card)
df2.train$card=as.factor(df2.train$card)
#fitControl <- trainControl(method = 'cv', number = 10, summaryFunction=defaultSummary)

set.seed(2022)
#getModelInfo()$gbm$parameters
#gbmGrid <-  expand.grid(interaction.depth = c(1,2,4),
#                        n.trees = seq(from=100,to=10000,by=100),
#                        shrinkage = c(0.001,0.01),
#                        n.minobsinnode= c(10,30))

#set.seed(2022)
#fit.gbm <- train(card~., data = df2.train, method = 'gbm', trControl = fitControl, tuneGrid = gbmGrid,

#saveRDS(fit.gbm, "fit.gbm.csv")
fit.gbm <- readRDS("fit.gbm.csv")

fit.gbm$bestTune
```

```
    n.trees interaction.depth shrinkage n.minobsinnode
972    7200                 2      0.01             30
```

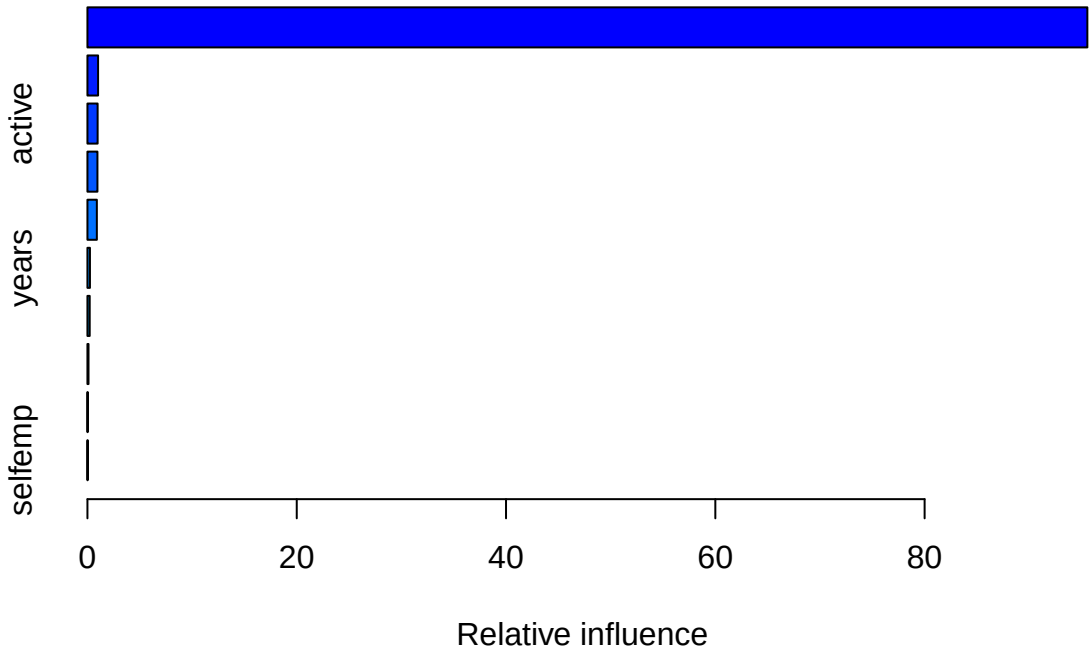
```r
res_gbm <- fit.gbm$results
acc_gbm <- subset(res_gbm[5])
max(acc_gbm)
```

```
[1] 0.9862937
```

```r
set.seed(2022)
boost.caret.pred <- predict(fit.gbm,df2.test)

set.seed(2022)
conf_matrix_gbm <- confusionMatrix(boost.caret.pred, df2.test$card, mode = "prec_recall",  positive = "
conf_matrix_gbm
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 136  13
         1   3 504

               Accuracy : 0.9756
                 95% CI : (0.9607, 0.986)
    No Information Rate : 0.7881
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9288

 Mcnemar's Test P-Value : 0.02445

              Precision : 0.9941
                 Recall : 0.9749
                     F1 : 0.9844
             Prevalence : 0.7881
         Detection Rate : 0.7683
   Detection Prevalence : 0.7729
      Balanced Accuracy : 0.9766

       'Positive' Class : 1
```

```r
summary(fit.gbm)
```




```
                  var     rel.inf
share           share 95.55050049
age               age  1.01644664
active         active  0.97287210
income         income  0.95471783
reports       reports  0.89940608
years           years  0.24044231
dependents dependents  0.21617627
majorcards majorcards  0.10113241
owner           owner  0.02553906
selfemp       selfemp  0.02276681
```

Se obtiene como resultado que 7200 árboles, con una profundidad de 2, un *shrinkage* de 0.01, y un tamaño mínimo de 30, es el modelo que mejor ajusta. El *share* es la variable más significativa, seguido por la variable *age*, aunque muy por detras. Al analizar la matriz de confusión, se observan buenos resultados para *Accuracy*, *Balanced Accuracy*, y *Recall*, con valores de 0.975, 0.976, y 0.975 respectivamente. Es especialmente importante destacar que la *Precision*, métrica fundamental en nuestro análisis, alcanzó el valor de 0.994, lo cual indica que se le dió tarjeta de crédito solo a tres personas que no debían tenerla.

4.2 Conclusiones - Análisis cualitativo

Para analizar de manera más sencilla los diversos modelos y sus resultados, se crea un dataframe que agrupe los resultados obtenidos:

```
metricas= c("Accuracy","Recall","Precision","Balanced accuracy" )

lda= c(conf_matrix_lda$overall[[1]], conf_matrix_lda$byClass[[6]],
```

```
       conf_matrix_lda$byClass[[5]], conf_matrix_lda$byClass[[11]])

glm= c(conf_matrix_glm$overall[[1]], conf_matrix_glm$byClass[[6]],
       conf_matrix_glm$byClass[[5]], conf_matrix_glm$byClass[[11]])

qda= c(conf_matrix_qda$overall[[1]], conf_matrix_qda$byClass[[6]],
       conf_matrix_qda$byClass[[5]], conf_matrix_qda$byClass[[11]])

nb= c(conf_matrix_nb$overall[[1]], conf_matrix_nb$byClass[[6]],
       conf_matrix_nb$byClass[[5]], conf_matrix_nb$byClass[[11]])

tree= c(conf_matrix_tree$overall[[1]], conf_matrix_tree$byClass[[6]],
       conf_matrix_tree$byClass[[5]], conf_matrix_tree$byClass[[11]])

rf= c(conf_matrix_rf$overall[[1]], conf_matrix_rf$byClass[[6]],
       conf_matrix_rf$byClass[[5]], conf_matrix_rf$byClass[[11]])

gbm= c(conf_matrix_gbm$overall[[1]], conf_matrix_gbm$byClass[[6]],
       conf_matrix_gbm$byClass[[5]], conf_matrix_gbm$byClass[[11]])

df.results= data.frame(metricas,lda,glm,qda,nb,tree,rf,gbm)
rownames(df.results) = df.results$metricas
df.results$metricas= NULL
df.results
```

```
                     lda       glm       qda        nb      tree        rf
Accuracy          0.8719512 0.9756098 0.9740854 0.9679878 0.9771341 0.9710366
Recall            0.9961315 0.9748549 0.9748549 0.9671180 0.9709865 0.9709865
Precision         0.8626466 0.9940828 0.9921260 0.9920635 1.0000000 0.9920949
Balanced accuracy 0.7031017 0.9766361 0.9730390 0.9691705 0.9854932 0.9711047
                      gbm
Accuracy          0.9756098
Recall            0.9748549
Precision         0.9940828
Balanced accuracy 0.9766361
```

Al analizar los resultados, se ve que el modelo de *árbol* es el que posee un *Accuracy*, *Balanced accuracy*, y *Precision* más elevados. Se destaca especialmente el valor registrado para *Presicion*, ya que como fuera anteriormente mencionado, significa que no se le ha dado a nadie que no debía poseerla, una tarjeta de crédito. Si bien el modelo de *lda* posee la *Recall* más elevada, esta métrica no es tan importante en el caso que analizado. Si la métrica es elevada, esto significa que hay pocos falsos negativos, por lo que se le ha dado tarjeta a casi todas las personas suceptibles de recibirla. Ahora bien, se releva que dicho modelo es el que peor se comporta en todas las otras métricas, incluida la *Precision*, con el elevado costo económico que esto conlleva. Si se estuviera analizando el caso de la detección de una enfermedad en cambio, esta métrica sería fundamental.

Como comentario general, es importante destacar que la métrica más importante para la predicción es *share*. Se pone de manifiesto en diversas partes del análisis que el resto de las variables ocupan un lugar decididamente secundario.

5. Análisis cuantitavo

Para el análisis cuantitativo se opta por utilizar la variable *Expenditure* como dependiente para el modelado. Se procede a exlcuir la variable *share*, ya que la misma tiene incluida en su definición, el valor de la variable

dependiente. De igual modo y replicando lo realizado para el análisis cualitativo, se elimina la variable *months* para evitar multicolinealidad:

```
df3 = subset(df3, select = -c(share,months))
write.csv(df3,"df3.csv")
df3= read.csv("df3.csv")
df3$X= NULL
```

A continuación, se evalúa la correlación entre las variables seleccionadas:

```
hetcor(df3)[1]
```

```
$correlations
                    card      reports          age     income expenditure
card         1.000000000 -0.453567577 0.003256061 0.09475555  0.36543463
reports     -0.453567577  1.000000000 0.040657171 0.01054498 -0.13627254
age          0.003256061  0.040657171 1.000000000 0.33145672  0.01528666
income       0.094755546  0.010544976 0.331456725 1.00000000  0.28157133
expenditure  0.365434629 -0.136272542 0.015286662 0.28157133  1.00000000
owner        0.147758963 -0.053783488 0.374904841 0.32433074  0.09168941
selfemp     -0.054173911  0.018506070 0.098300457 0.11208268 -0.03558791
dependents  -0.037773282  0.020254830 0.218977359 0.31740910  0.05046493
majorcards   0.105297159 -0.006620495 0.005839761 0.10810229  0.07528103
active       0.079472626  0.208139368 0.185816629 0.18109389  0.05315996
years       -0.002120207  0.049166484 0.452615301 0.13028147 -0.02877484
                   owner      selfemp   dependents   majorcards     active
card         0.14775896 -0.054173911 -0.037773282  0.105297159 0.07947263
reports     -0.05378349  0.018506070  0.020254830 -0.006620495 0.20813937
age          0.37490484  0.098300457  0.218977359  0.005839761 0.18581663
income       0.32433074  0.112082684  0.317409096  0.108102285 0.18109389
expenditure  0.09168941 -0.035587911  0.050464933  0.075281028 0.05315996
owner        1.00000000  0.041329929  0.305946458  0.061535729 0.27416642
selfemp      0.04132993  1.000000000  0.042235371  0.004484547 0.02949921
dependents   0.30594646  0.042235371  1.000000000  0.007194003 0.10574382
majorcards   0.06153573  0.004484547  0.007194003  1.000000000 0.11819395
active       0.27416642  0.029499206  0.105743820  0.118193947 1.00000000
years        0.24291898  0.067593793  0.049353071 -0.040730058 0.10085781
                    years
card        -0.002120207
reports      0.049166484
age          0.452615301
income       0.130281471
expenditure -0.028774838
owner        0.242918984
selfemp      0.067593793
dependents   0.049353071
majorcards  -0.040730058
active       0.100857812
years        1.000000000
```

Se observa que no hay variables con multicolinealidad, por lo que se procede a realizar la división de los datos entre *train* y *test*, en partes iguales:

```
set.seed(2022)
tamano_sample= floor(0.5*nrow(df3))
train <- sample(seq_len(nrow(df3)), size = tamano_sample)
```

```
df3.train= df3[train, ]
df3.test=df3[-train, ]
```

5.1 Evaluación de modelos

Los modelos a utlizar en esta parte del trabajo son:

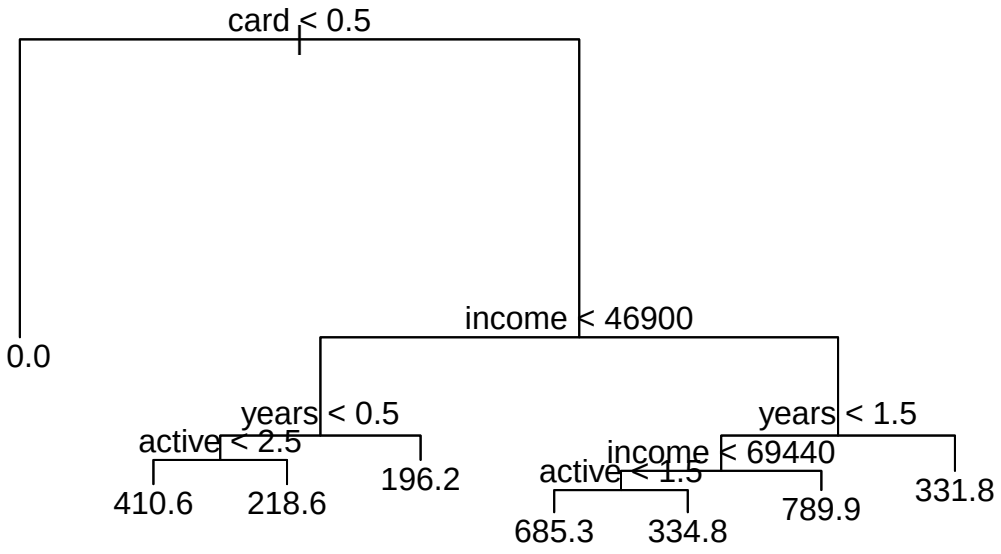
-Árboles

-Random Forest

-Boosting

Arboles

```
set.seed(2022)
tree.fit2 <- tree(expenditure~., df3, subset = train)

plot(tree.fit2)
text(tree.fit2, pretty = 0)
```




```
summary(tree.fit2)
```

```
Regression tree:
tree(formula = expenditure ~ ., data = df3, subset = train)
Variables actually used in tree construction:
```

```
[1] "card"   "income" "years"  "active"
Number of terminal nodes:  8
Residual mean deviance:  49550 = 32110000 / 648
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -607.1  -129.2     0.0     0.0    25.4  1641.0
```

```
tree.fit2
```

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 656 44480000 185.8
   2) card < 0.5 156        0   0.0 *
   3) card > 0.5 500 37410000 243.8
     6) income < 46900 412 22910000 212.2
      12) years < 0.5 64  5779000 299.6
        24) active < 2.5 27  4018000 410.6 *
        25) active > 2.5 37  1186000 218.6 *
      13) years > 0.5 348 16550000 196.2 *
     7) income > 46900 88 12180000 391.5
      14) years < 1.5 24  6758000 550.8
        28) income < 69440 18  3896000 471.1
          56) active < 1.5 7  2490000 685.3 *
          57) active > 1.5 11   880200 334.8 *
        29) income > 69440 6  2405000 789.9 *
      15) years > 1.5 64  4583000 331.8 *
```

```
tree.pred2 = predict(tree.fit2, df3.test)
tree= postResample(tree.pred2,df3.test$expenditure)
tree
```

```
       RMSE    Rsquared         MAE
260.5189822   0.1674371 146.7490101
```

Se observa que el árbol está compuesto de 4 variables y 8 nodos terminales. En primer lugar se evalúa el valor de la variable *card*. Si la persona no tiene tarjeta, naturalemente su gasto será de 0. En caso contrario pasa a evaluarse el ingreso. Si la persona posee un ingreso menor a 46.900, se decanta a analizar si la cantidad de años en la vivienda es menor a 0.5. En caso negativo, los gastos estimados son de 196.2, caso contrario, evalúa el número de cuentas activas. Si posee menos de 2.5 cuentas, el gasto será de 410.6, mientras que si es mayor, será de 218.6.

Si la persona posee un ingreso mayor a 46.900, se analiza mediante la variable *years* si hace más de 1.5 años habita la misma vivienda. En caso que sea así, el gasto será de 331.8, mientras que en el caso negativo se evalua nuevamente el ingreso. En este caso se analiza si el mismo es menor a 69.440. En caso negativo el gasto será de 789.9, mientas que en caso positivo, se evalua si la persona posee menos de 1.5 cuentas activas. En el caso de que la respuesta sea positiva, el gasto estipulado por el modelo es de 685.3, caso contrario es de 334.8.

Al analizar los resultados obtenidos vemos que las variables no predicen bien la variable dependiente. Se ha obtenido un R cuadrado de solo 0.167. La raíz del error cuadrático medio es de 260,51.

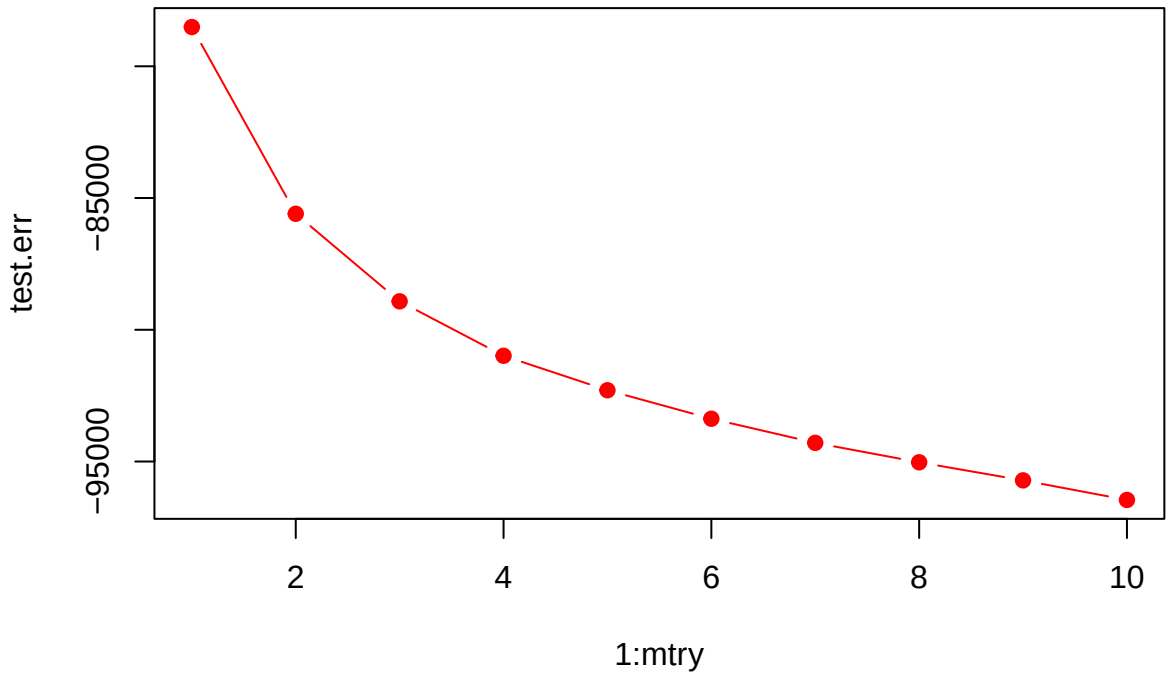
Random forest

Se procede a modelar un *Random forest* con 10.000 árboles y se calcula la cantidad de variables que minimizan el error (evaluando todas las variables posibles).

```
set.seed(2022)
NVariables <- ncol(df3)-1
test.err <- double(NVariables)

set.seed(2022)
for(mtry in 1:NVariables){
  fit <- randomForest(expenditure~.,data=df3,subset=train,mtry=mtry,ntree=10000)
  pred <- predict(fit,df3.test)
  test.err[mtry]<- with(df3.test,1-mean((df3$expenditure-pred)^2))
  cat(mtry," ")
}
```

```
1  2  3  4  5  6  7  8  9  10
```

```
plot(1:mtry,test.err,pch=19,col="red",type="b")
```




```
set.seed(2022)
rf2=postResample(pred,df3.test$expenditure)
rf2
```

```
       RMSE    Rsquared         MAE
259.9877258   0.1787594 148.3736589
```

Se observa en base al plot, que el error se minimiza con 10 variables.

Se observan magros resultados predictivos. El *R-squared* alcanza un valor de 0.178. La raíz del error cuadrático medio es de 259,987.

Boosting

A continuación se hace el modelado con *Boosting*, utilizando CV. Luego se procede a ajustar los parámetros, indicando una profundidad de 1,2, y 4. Se especifican dos *shrinkage*, de 0.01 y 0.001. Asimismo, se realizan pruebas con árboles de tamaño 100 a 10.000, con saltos de a 100. Por último, se especifican dos tamaños mínimos de observaciones permitidas, 10 y 30. Esto nos da como resultado 1200 modelos a testear.

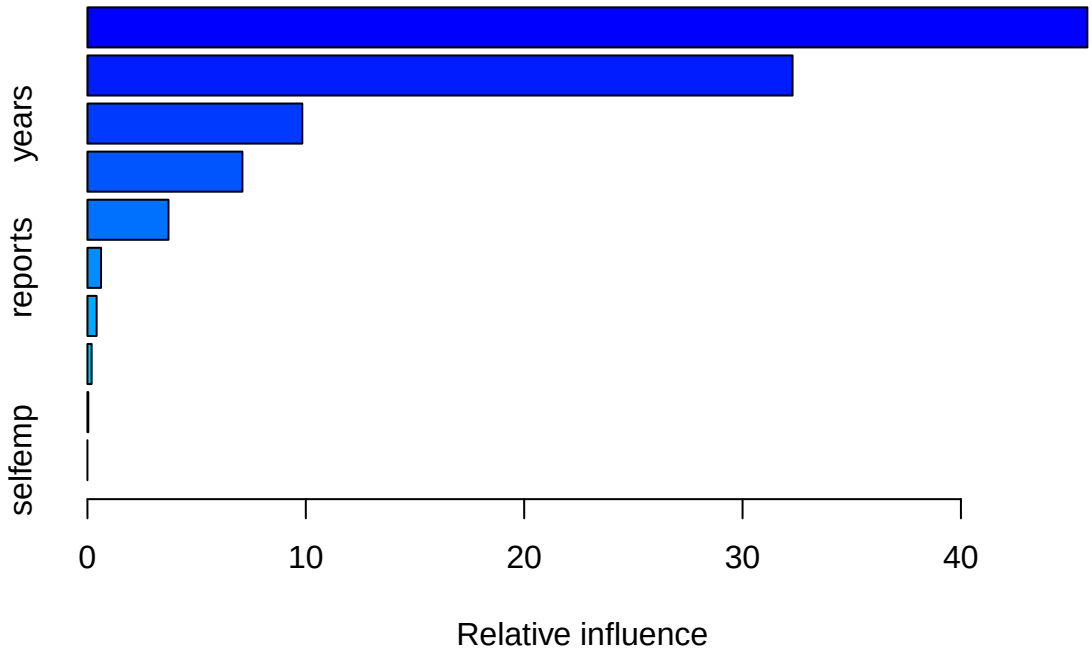
```
set.seed(2022)
#fitControl2 <- trainControl(method = 'cv', number = 10, summaryFunction=defaultSummary)

#set.seed(2022)
#getModelInfo()$gbm$parameters
#gbmGrid <-  expand.grid(interaction.depth = c(1,2,4),
#                        n.trees = seq(from=100,to=10000,by=100),
#                        shrinkage = c(0.001,0.01),
#                        n.minobsinnode= c(10,30))

#set.seed(2022)
#fit.gbm2 <- train(expenditure~., data = df3.train, method = 'gbm', trControl = fitControl2, tuneGrid =

#saveRDS(fit.gbm2, "fit.gbm2.csv")
fit.gbm2 <- readRDS("fit.gbm2.csv")

fit.gbm2$bestTune
```

```
    n.trees interaction.depth shrinkage n.minobsinnode
328    2800                 2     0.001             30
```

```
res_gbm2 <- fit.gbm2$results
acc_gbm2 <- subset(res_gbm[5])
max(acc_gbm2)
```

```
[1] 0.9862937
```

```
set.seed(2022)
boost.pred2 <- predict(fit.gbm2,df3.test)

set.seed(2022)
boosting=postResample(boost.pred2,df3.test$expenditure)
summary(fit.gbm2)
```




```
                  var     rel.inf
card             card 45.78994355
income         income 32.28970777
years           years  9.83842124
age               age  7.09619441
active         active  3.71111696
reports       reports  0.62326772
dependents dependents  0.42066736
owner           owner  0.19159837
majorcards majorcards  0.03908261
selfemp       selfemp  0.00000000
```

El mejor modelo es con 2800 árboles, profundidad 2, *shrinkage* 0.001, y un mínimo de observaciones de 30. La variable más relevante a la hora de predecir los gastos es, como era de esperar, *card*, seguido de *income*. En tercer lugar aparece *years*, y en cuarto lugar *age*.

Se observan nuevamente malos resultados. El *R-squared* alcanza el valor de 0.188. La raíz del error cuadrático medio es de 256,343.

5.2 Conclusiones - Análisis cuantitativo

Para analizar de manera más sencilla los diversos modelos y sus resultados, se crea un dataframe que agrupe los resultados obtenidos:

```
metricas2= c("RMSE","Rsquared","MAE")

tree= c(tree[[1]], tree[[2]],tree[[3]])
```

```
rf2= c(rf2[[1]], rf2[[2]],rf2[[3]])

boosting= c(boosting[[1]], boosting[[2]],boosting[[3]])


df.results2= data.frame(metricas2, tree, rf2, boosting)
rownames(df.results2) = df.results2$metricas2
df.results2$metricas2= NULL
df.results2
```

```
               tree         rf2    boosting
RMSE     260.5189822 259.9877258 257.3431875
Rsquared   0.1674371   0.1787594   0.1886001
MAE      146.7490101 148.3736589 146.4100274
```

Se observa que los tres modelos no logran presentar buenas métricas. A modo de ejemplo, el *R-squared* no supera en ningun caso el valor de 0.188. De los tres, el que obtiene un menor error y un R2 más elevado es el modelo de *boosting*, aunque con resultados decepcionantes. Es muy probable en este sentido, que la ausencia de más variables cuantitativas haya sido sustancial en el mal comportamiento de los modelos.