Proyecto FMAD

ICAI. Máster en Big Data. Fundamentos Matemáticos del Análisis de Datos (FMAD).

Curso 2021-22. Última actualización: 2021-11-13

Índice

1. Introducción

Lo primero que haremos será cargar las diferentes librerias que usaremos para nuestro proyecto. Entre ellas encontraremos librerias públicas como tidyverse y algunas privadas como MLTools:

```
library(tidyverse)
library(lubridate)
library(caret)
library(grid)
library(gridExtra)
library(ROCR)
library(MLTools)
library(GGally)
library(rpart)
library(rpart.plot)
library(partykit)
library(kernlab)
library(NeuralNetTools)
library(NeuralSens)
library(nnet)
library(ROSE)
library(randomForest)
```

A continuación leeremos los datos con los que trabajaremos:

```
datos <- read.csv("marketing_campaign.csv", header = TRUE, sep = "")
```

2. Definición de las variables

Antes de comenzar con el preprocesamiento de los datos lo que haremos será listar las variables y lo que representa cada una de ella:

- **ID**: El ID del cliente.
- **Year_Birth:** Indica el año de nacimiento del cliente.
- **Education:** Indica el nivel de educación del cliente.
- **Marital_Status:** Indica el estado civil del cliente.
- **Income:** Presenta el ingreso familiar anual del cliente.
- **Kidhome:** Indica el número de niños pequeños en casa del cliente.
- **Teenhome:** Indica el número de adolescentes en el hogar del cliente.
- **Dt_Customer:** Muestra la fecha de inscripción del cliente en la empresa.
- **Recency:** El número de días desde la última compra.
- **MntWines:** El gasto en productos vitivinícolas en los últimos 2 años.
- **MntGoldProds:** El gasto en productos premium en los últimos 2 años.
- **NumDealsPurchases:** El número de compras con uso de descuento.
- **NumWebPurchases:** El número de compras a través de la web.
- **NumCatalogPurchases:** El número de compras usando catalogo.
- **NumWebVisitsMonth:** El número de visitas por mes a la web.
- **AcceptedCmp1:** 1 si el cliente acepta la oferta en la 1ra campaña, 0 si no lo acepta.
- **AcceptedCmp2:** 1 si el cliente acepta la oferta en la 2nd campaña, 0 si no lo acepta.
- **Complain:** 1 si el cliente se ha quejado en los dos últimos años.
- **Z_CostContact:** El coste de contactar con cliente.
- **Z_Revenue:** Los ingresos/beneficios después de que el cliente acepte la campaña.
- **Response:** 1 si el cliente acepta la oferta en la última campaña y 0 si no la acepta.

3. Preprocesamiento

3.1. Resumen de datos

Lo primero que haremos será ver como esta estructurado nuestro dataset. Para ello veremos que tamaño tiene, tanto filas como columnas. A su vez también veremos con que tipo de datos estamos trabajando.

```
cat(cat(cat(cat("El conjunto de datos tiene", nrow(datos)), "filas y"),
        ncol(datos)), "columnas")
```

```
## El conjunto de datos tiene 2440 filas y 29 columnas
```

```
str(datos)
```

```
## 'data.frame':    2440 obs. of  29 variables:
##  $ ID                 : int  5524 2174 4141 6182 5324 7446 965 6177 4855 5899 ...
##  $ Year_Birth         : int  1957 1954 1965 1984 1981 1967 1971 1985 1974 1950 ...
##  $ Education          : chr  "Graduation" "Graduation" "Graduation" "Graduation" ...
##  $ Marital_Status     : chr  "Single" "Single" "Together" "Together" ...
##  $ Income             : chr  "58138" "46344" "71613" "26646" ...
##  $ Kidhome            : int  0 1 0 1 1 0 0 1 1 1 ...
##  $ Teenhome           : chr  "0" "1" "0" "0" ...
##  $ Dt_Customer        : chr  "04-09-2012" "08-03-2014" "21-08-2013" "10-02-2014" ...
##  $ Recency            : chr  "58" "38" "26" "26" ...
##  $ MntWines           : int  635 11 426 11 173 520 235 76 14 28 ...
##  $ MntFruits          : int  88 1 49 4 43 42 65 10 0 0 ...
##  $ MntMeatProducts    : int  546 6 127 20 118 98 164 56 24 6 ...
##  $ MntFishProducts    : int  172 2 111 10 46 0 50 3 3 1 ...
##  $ MntSweetProducts   : int  88 1 21 3 27 42 49 1 3 1 ...
##  $ MntGoldProds       : int  88 6 42 5 15 14 27 23 2 13 ...
##  $ NumDealsPurchases  : int  3 2 1 2 5 2 4 2 1 1 ...
##  $ NumWebPurchases    : int  8 1 8 2 5 6 7 4 3 1 ...
##  $ NumCatalogPurchases: int  10 1 2 0 3 4 3 0 0 0 ...
##  $ NumStorePurchases  : int  4 2 10 4 6 10 7 4 2 0 ...
##  $ NumWebVisitsMonth  : int  7 5 4 6 5 6 6 8 9 20 ...
##  $ AcceptedCmp3       : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ AcceptedCmp4       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp5       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp1       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp2       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Complain           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Z_CostContact      : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Z_Revenue          : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ Response           : int  1 0 0 0 0 0 0 0 1 0 ...
```

Una vez visto el tipo de variables con las que trabajamos es facilmente observable la necesidad de realizar algunas modificaciones en algunas de ellas.

Ahora veremos un resumen de las variables que tenemos:

```
summary(datos)
```

```
##        ID          Year_Birth    Education         Marital_Status
##  Min.   :    0   Min.   :1893   Length:2440        Length:2440
##  1st Qu.: 2108   1st Qu.:1959   Class :character   Class :character
##  Median : 5048   Median :1970   Mode  :character   Mode  :character
##  Mean   : 5134   Mean   :1969
##  3rd Qu.: 8147   3rd Qu.:1977
##  Max.   :11191   Max.   :1996
##                  NA's   :200
##     Income             Kidhome        Teenhome         Dt_Customer
##  Length:2440        Min.   :    0   Length:2440        Length:2440
##  Class :character   1st Qu.:    0   Class :character   Class :character
##  Mode  :character   Median :    0   Mode  :character   Mode  :character
##                     Mean   : 4253
##                     3rd Qu.:    1
##                     Max.   :96547
##                     NA's   :200
##    Recency             MntWines        MntFruits       MntMeatProducts
##  Length:2440        Min.   :   0.0   Min.   :   0.00   Min.   :   0.0
##  Class :character   1st Qu.:  25.0   1st Qu.:   2.00   1st Qu.:  14.0
##  Mode  :character   Median : 138.0   Median :   9.00   Median :  57.0
##                     Mean   : 288.5   Mean   :  42.39   Mean   : 156.8
##                     3rd Qu.: 476.5   3rd Qu.:  38.00   3rd Qu.: 211.5
##                     Max.   :1493.0   Max.   :1215.00   Max.   :1725.0
##                     NA's   :200      NA's   :200       NA's   :200
##  MntFishProducts  MntSweetProducts  MntGoldProds    NumDealsPurchases
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.000
##  1st Qu.:  3.00   1st Qu.:  1.00   1st Qu.:  7.00   1st Qu.:  1.000
##  Median : 13.00   Median :  9.00   Median : 22.00   Median :  2.000
##  Mean   : 45.34   Mean   : 28.44   Mean   : 42.45   Mean   :  6.326
##  3rd Qu.: 55.00   3rd Qu.: 35.00   3rd Qu.: 54.00   3rd Qu.:  4.000
##  Max.   :974.00   Max.   :362.00   Max.   :321.00   Max.   :246.000
##  NA's   :200      NA's   :200      NA's   :200      NA's   :200
##  NumWebPurchases  NumCatalogPurchases NumStorePurchases NumWebVisitsMonth
##  Min.   : 0.000   Min.   : 0.000      Min.   : 0.000    Min.   : 0.000
##  1st Qu.: 2.000   1st Qu.: 1.000      1st Qu.: 3.000    1st Qu.: 3.000
##  Median : 3.000   Median : 2.000      Median : 5.000    Median : 6.000
##  Mean   : 3.929   Mean   : 2.817      Mean   : 5.503    Mean   : 5.278
##  3rd Qu.: 6.000   3rd Qu.: 4.000      3rd Qu.: 8.000    3rd Qu.: 7.000
##  Max.   :27.000   Max.   :28.000      Max.   :13.000    Max.   :20.000
##  NA's   :200      NA's   :200         NA's   :200       NA's   :200
```

```
##   AcceptedCmp3      AcceptedCmp4       AcceptedCmp5       AcceptedCmp1
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.0000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.5545   Mean   :0.07679   Mean   :0.07277   Mean   :0.06161
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :9.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##  NA's   :200      NA's   :200       NA's   :200       NA's   :200
##   AcceptedCmp2         Complain       Z_CostContact      Z_Revenue
##  Min.   :0.00000   Min.   :0.00000   Min.   : 0.000   Min.   : 0.00
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.: 3.000   1st Qu.:11.00
##  Median :0.00000   Median :0.00000   Median : 3.000   Median :11.00
##  Mean   :0.01875   Mean   :0.03661   Mean   : 2.809   Mean   :10.18
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.: 3.000   3rd Qu.:11.00
##  Max.   :1.00000   Max.   :3.00000   Max.   :11.000   Max.   :11.00
##  NA's   :200       NA's   :200       NA's   :200      NA's   :200
##     Response
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 0.000
##  Mean   : 1.132
##  3rd Qu.: 0.000
##  Max.   :11.000
##  NA's   :221
```

Observamos que existen numerosos valores nulos en nuestras variables. En el punto siguiente veremos que hacer con estos casos.

3.2. Análisis de las variables

Lo primero que haremos será eliminar las filas que contienen datos nulos. Esto podemos hacerlo ya que disponemos de una muestra muy grande y eliminar los valores nulos no afectará para nuestro trabajo.

```
datos <- na.omit(datos)
```

Además, también observamos que hay algunos datos erróneos por lo que por el mismo motivo que antes procederemos a eliminarlos.

```
datos <- datos %>%
  filter(ID != 0 & ID != 1 & Education != "2n" & Income > 10 & Income != "2")
```

Además el conjunto de datos tiene muchas columnas las cuales no nos resultan interesantes, por ello vamos a eliminar algunas de ellas: “NumDealsPurchases”, “Receny”, “AcceptedCmp1”, “AcceptedCmp2”, “AcceptedCmp3”, “AcceptedCmp4” y “AcceptedCmp5”, “Z_CostContact” y “Z_Revenue”.

```
datos <- datos %>%
  select(-c(AcceptedCmp3:AcceptedCmp2), -NumDealsPurchases, -Recency,
```

```
         -Z_CostContact, -Z_Revenue)
```

También como vimos cuando hicimos la visión general de las variables y su tipo, nos dimos cuenta de que algunas de ellas estaban mal tipadas. Por ello cambiaremos el tipado de algunas columnas.

```
datos$Teenhome <- as.numeric(datos$Teenhome)
datos$Income <- as.numeric(datos$Income)
datos$Complain <- as.factor(datos$Complain)
datos$Education <- as.factor(datos$Education)
datos$Response <- as.factor(datos$Response)
```

A su vez hemos observado que algunas columnas podrían tener un formato más útil o sencillo, como es el caso del año de nacimiento, donde es mas cómodo trabajar con edades. Por tanto, para un mejor procesamiento y una mayor útilidad realizaremos un mutate para generar una nueva columna formada por la edad de los clientes. A su vez eliminaremos la columna de año de nacimiento.

```
datos <- datos %>%
  mutate(edad = 2021 - Year_Birth) %>%
  select(-Year_Birth)
```

También nos pareció interesante en vez de distinguir entre número de hijos los cuales son pequeños o son adolescentes, tomarlos como una única variable que nos indique el número de hijos que hay en cada hogar. Para ello sumaremos el total de niños de cada cliente agrupando las columnas Kidhome y Teenhome.

```
datos <- datos %>%
  mutate(totalHijos = Kidhome + Teenhome) %>%
  select(-Kidhome, -Teenhome)
```

Como en el caso de los hijos para las compras haremos algo similar, donde cogeremos las columnas "NumWebPurchases", "NumCatalogPurchases" y "NumStorePurchases" que indican el número de compras hechas en cada sitio, en tiendas, por catalogo y por la web y las sumaremos todas en una única columna que indique el total de compras que ha realizado el cliente.

```
datos <- datos %>%
  rowwise(ID) %>%
  mutate(suma_compras = sum(c(NumWebPurchases, NumCatalogPurchases,
                              NumStorePurchases))) %>%
  select(-c(NumWebPurchases, NumCatalogPurchases, NumStorePurchases))
```

A su vez, para el gasto en los diferentes tipos de producto sumaremos las columans: "MntWines", "MntFruits", "MntMeatProducts", "MntFishProducts", "MntSweetProducts", "MntGoldProds" lo cual nos indicara cuánto dinero se ha gastado un cliente en total.

```
datos <- datos %>%
  rowwise(ID) %>%
  mutate(Dinero_Gastado = sum(c(MntWines, MntFruits, MntMeatProducts, MntFishProducts,
                                MntSweetProducts, MntGoldProds)))
```

También existe una variable que nos indica el estado civil del cliente, al existir numerosas situaciones nosotros agruparemos el estado civil de cada cliente y lo simplificamos para ver si vive solo o en pareja. Ya que esto nos podrá resultar interesante para análisis posteriores.

```
datos <- datos %>%
  mutate(Marital_Status = factor(Marital_Status== "Single" | Marital_Status== "Divorced",
                           levels = c(TRUE, FALSE), labels = c('Single','Not single')))
```

Por último cambiaremos la columna Dt_Customer y estableceremos 3 grupos que representan la longevidad del cliente en la empresa.

```
datos$Dt_Customer = as.Date(datos$Dt_Customer, "%d-%m-%Y")
fechaMinima = min(datos$Dt_Customer)
datos$Dt_Customer <-  factor(cut_number(as.duration(datos$Dt_Customer-fechaMinima),
                      n = 3), labels = c("Nuevo", "Con Experiencia", "Muy Antiguo"),
                      ordered = TRUE)
```

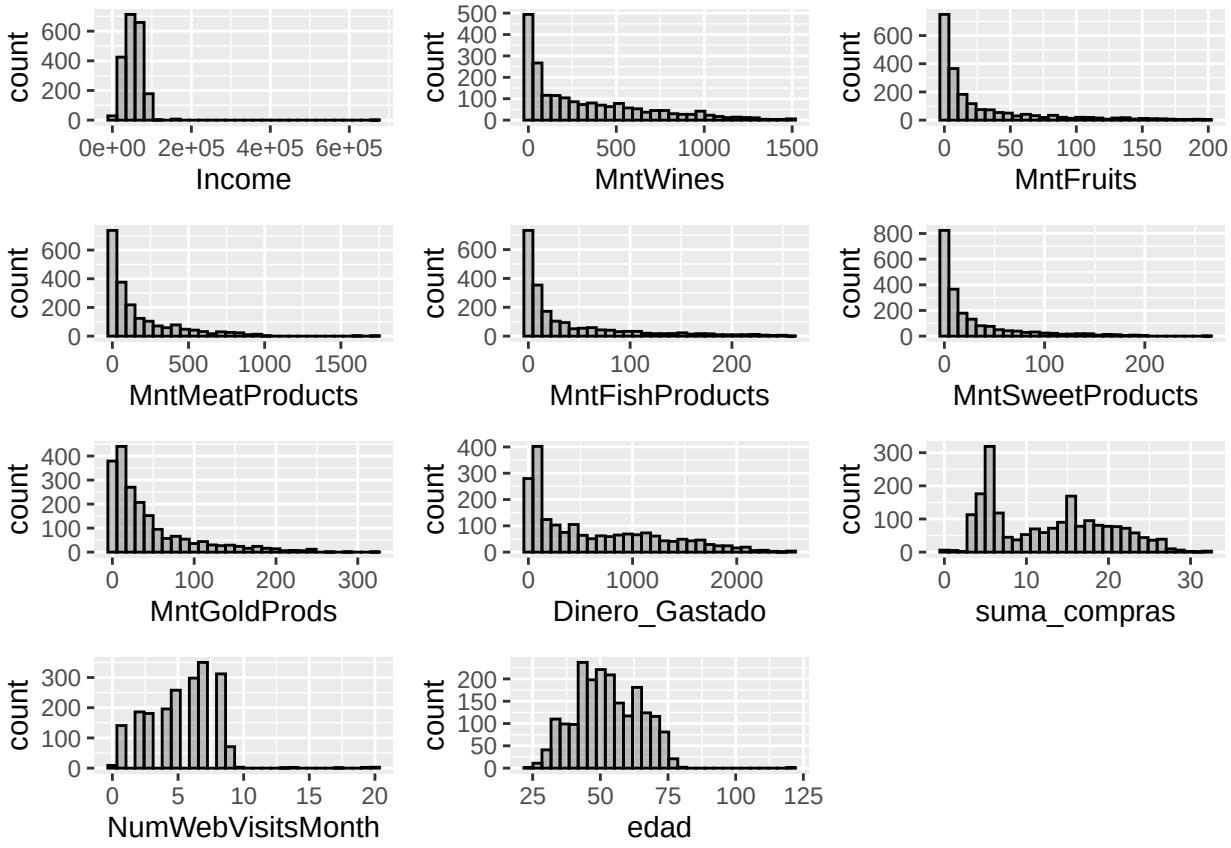
3.3. Visualización de los datos

En este apartado lo que realizaremos será un análisis mediante gráficas de las variables según su tipo, viendo si siguen una distribución normal y si presentarían outliers entre otros factores.

Comenzaremos con las variables continuas, que son las siguientes:

```
h1 <- ggplot(datos)+
  geom_histogram(aes(x = Income), color = "black", alpha = 0.35)
h2 <- ggplot(datos)+
  geom_histogram(aes(x = MntWines ), color = "black", alpha = 0.35)
h3 <- ggplot(datos)+
  geom_histogram(aes(x = MntFruits ), color = "black", alpha = 0.35)
h4 <- ggplot(datos)+
  geom_histogram(aes(x = MntMeatProducts ), color = "black", alpha = 0.35)
h5 <- ggplot(datos)+
  geom_histogram(aes(x = MntFishProducts ), color = "black", alpha = 0.35)
h6 <- ggplot(datos)+
  geom_histogram(aes(x = MntSweetProducts ), color = "black", alpha = 0.35)
h7 <- ggplot(datos) +
  geom_histogram(aes(x = MntGoldProds ), color = "black", alpha = 0.35)
h8 <- ggplot(datos) +
  geom_histogram(aes(x = Dinero_Gastado), color = "black", alpha = 0.35)
h9 <- ggplot(datos)+
  geom_histogram(aes(x = suma_compras ), color = "black", alpha = 0.35)
h10 <- ggplot(datos)+
  geom_histogram(aes(x = NumWebVisitsMonth ), color = "black", alpha = 0.35)
h12 <- ggplot(datos)+
  geom_histogram(aes(x = edad ), color = "black", alpha = 0.35)
```

```
grid.arrange(h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h12)
```

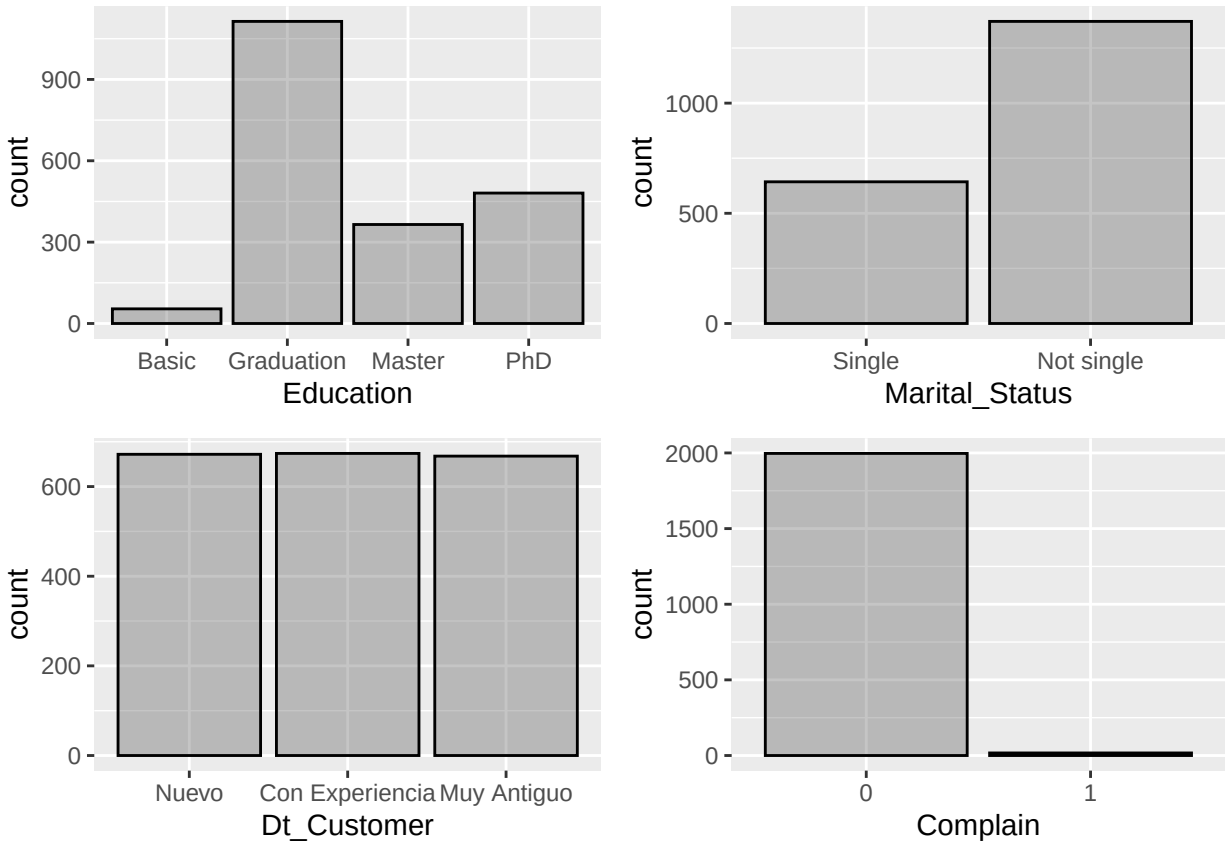


Podemos observar en los histogramas que ninguna de las variables sigue una distribución normal. El patrón más común es un gran número de datos con valores pequeños y muchos menos datos a medida que el valor de la variable del eje x aumenta. Por este motivo, aunque no los mostremos, podemos deducir que existen outliers en casi todas las variables. Un caso bastante claro de outlier se puede ver en la variable edad donde vemos que el eje x llega a 125 lo cual nos indica que debe existir alguna observación con valor por encima de 115 y menor de 125.

En cuanto a las variables discretas tenemos lo siguiente:

```
hb1 <- ggplot(datos)+
  geom_bar(aes(x = Education), color = "black", alpha = 0.35)
hb2 <- ggplot(datos) +
  geom_bar(aes(x = Marital_Status), color = "black", alpha = 0.35)
hb3 <- ggplot(datos) +
  geom_bar(aes(x = Dt_Customer), color = "black", alpha = 0.35)
hb4 <- ggplot(datos) +
  geom_bar(aes(x = Complain), color = "black", alpha = 0.35)

grid.arrange(hb1, hb2, hb3, hb4, ncol = 2)
```



En estas variables podemos observar diferentes patrones. Si miramos la educación de los clientes nos damos cuenta que hay mucha diferencia entre el número de clientes que tienen una educación básica y el resto de tipos, sobre todo los clientes graduados. De la misma forma vemos que hay más del doble de clientes not single que single. Por otro lado tenemos que la longividad de los clientes es uniforme. En cuanto a la variable complain observamos que unicamente un porcentaje muy pequeño de los clientes se han quejado durante los dos últimos años.

3. Análisis Predictivo y Analítica Avanzada

3.1. Preprocesamiento

En esta sección nuestro objetivo es usar técnicas de Machine Learning para predecir tendencias y comportamientos sobre diferentes variables que nos puedan resultar interesantes.

Para ello crearemos un nuevo conjunto de datos sacados de los datos que hemos refinado en la parte anterior. A su vez en un primer momento hemos analizado la relación entre tener hijos y los gastos en compras de cada tipo.

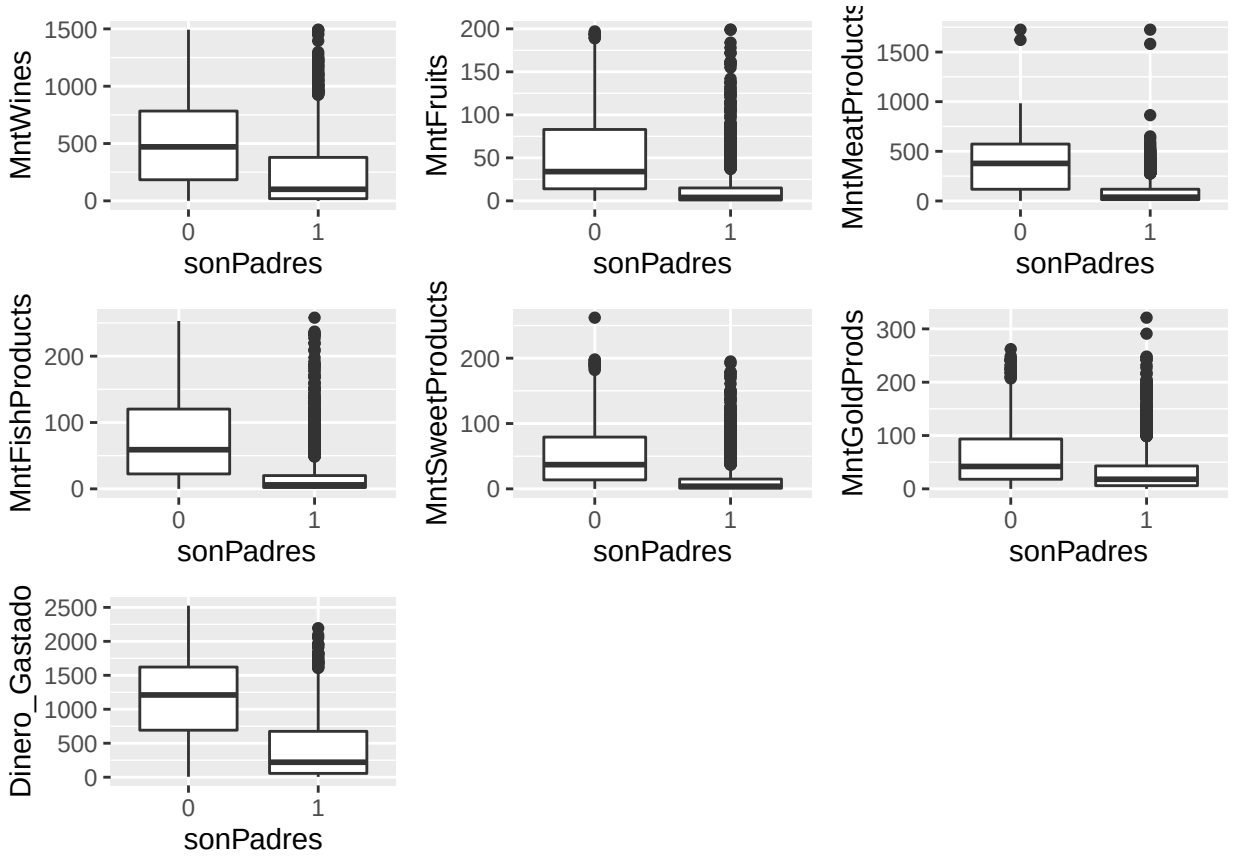
```
datos_ML<-datos %>%
  mutate(sonPadres = factor(totalHijos>0, levels = c(FALSE, TRUE), labels = c(0,1)))
borrar <- c("MntWines", "MntFruits", "MntMeatProducts",  "MntFishProducts", "MntSweetProducts", "MntGol
datos_ML <- datos_ML[(names(datos_ML) %in% borrar)]
head(datos_ML)
```

```
## # A tibble: 6 x 9
## # Rowwise:
##   MntWines MntFruits MntMeatProducts MntFishProducts MntSweetProducts
##      <int>     <int>           <int>           <int>             <int>
## 1      635        88             546             172                88
## 2       11         1               6               2                 1
## 3      426        49             127             111                21
## 4       11         4              20              10                 3
## 5      173        43             118              46                27
## 6      520        42              98               0                42
## # ... with 4 more variables: MntGoldProds <int>, totalHijos <dbl>,
## #   Dinero_Gastado <int>, sonPadres <fct>
```

Lo primero que haremos sera ver como se relacionan las diferentes variables con las variables de output, empezando por si tienen hijos.

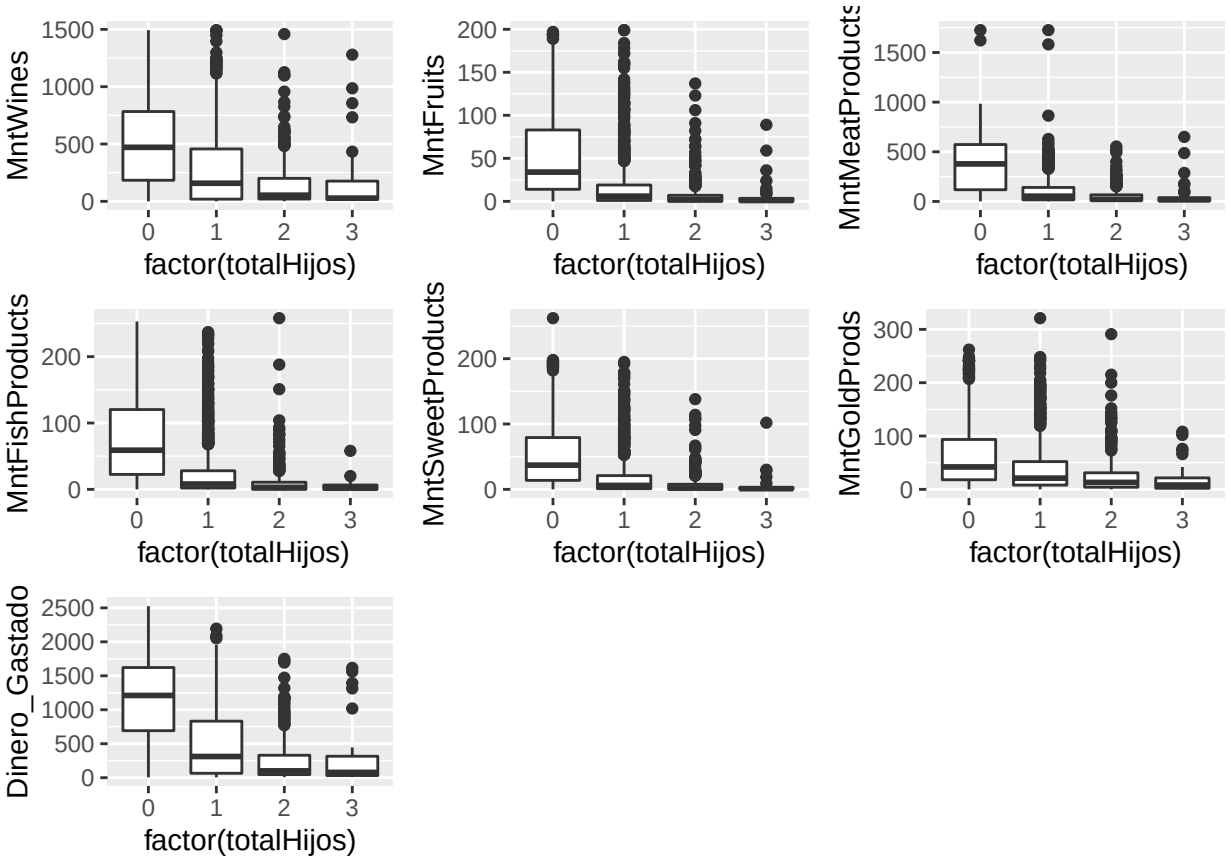
```
g1 <-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntWines))
g2<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntFruits))
g3<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntMeatProducts))
g4<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntFishProducts))
g5<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntSweetProducts))
g6<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntGoldProds))
g7<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = Dinero_Gastado))
```

```
gridExtra::grid.arrange(g1,g2,g3,g4, g5, g6, g7)
```



Vemos que si los clientes tienen hijos, el gasto en todos los productos se reduce. Pero ya que hemos llegado hasta aquí, queremos además observar la relación entre el número de hijos y dichos gastos. Veamos los siguientes gráficos.

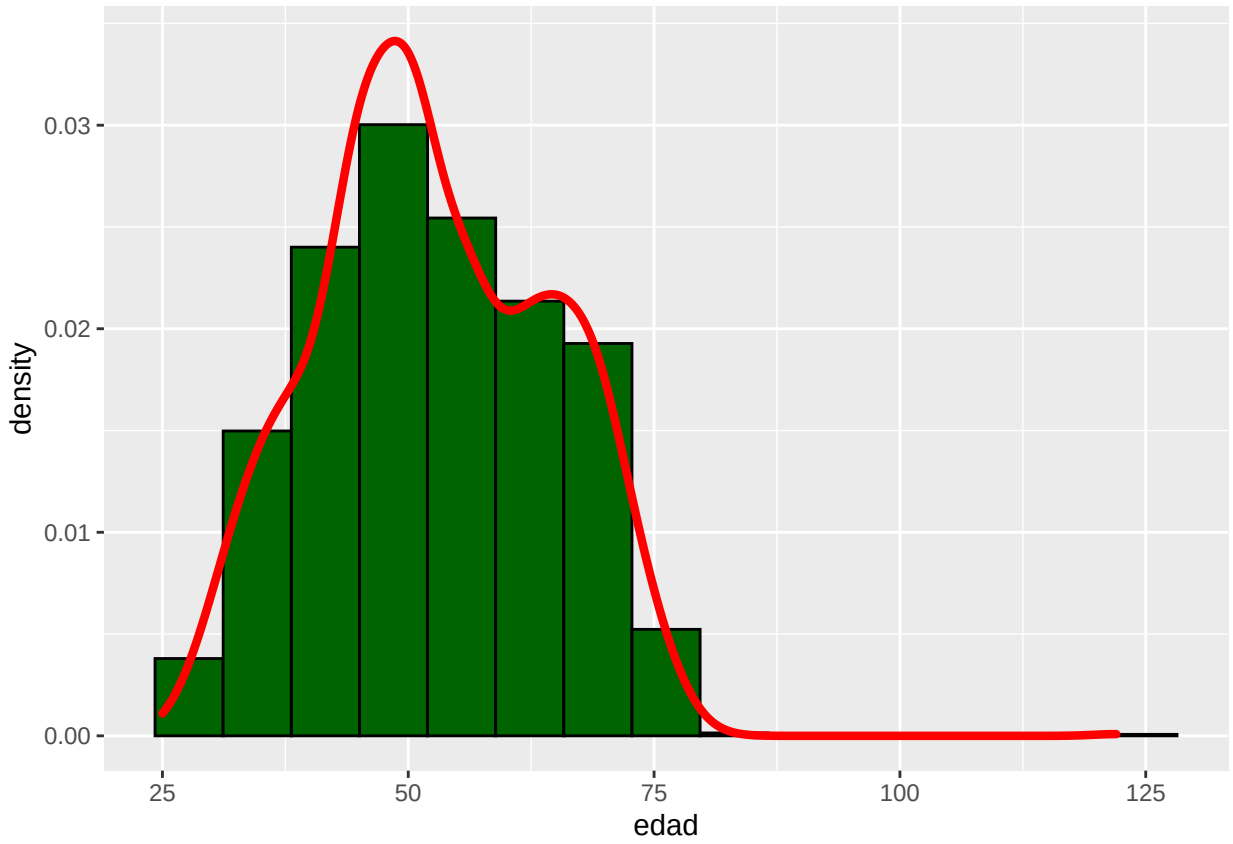
```
gb1 <-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntWines))
gb2<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntFruits))
gb3<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntMeatProducts))
gb4<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntFishProducts))
gb5<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntSweetProducts))
gb6<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntGoldProds))
gb7<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = Dinero_Gastado))
gridExtra::grid.arrange(gb1,gb2,gb3,gb4, gb5, gb6, gb7)
```



Siguiendo con la relación anterior, observamos que a más hijos menor es el gasto en todos los productos.

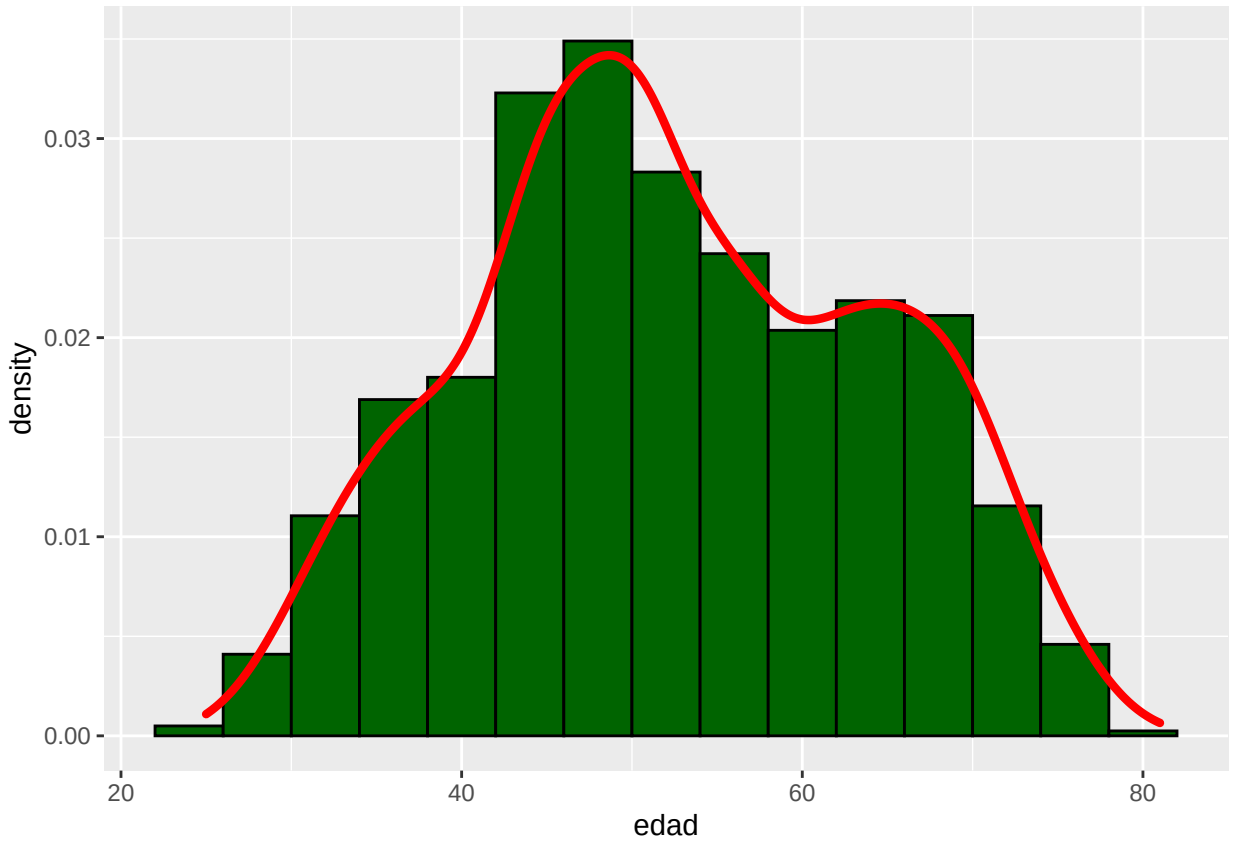
Veamos la relación entre la edad y los gastos de compras. Primero vamos a ver la distribución de la edad.

```
ggplot(datos) +
  geom_histogram(aes(x = edad, y =stat(density)), bins = 15, fill = "darkgreen",
                 color = "black") +
  geom_density(aes(x = edad), color="red", size=1.5)
```



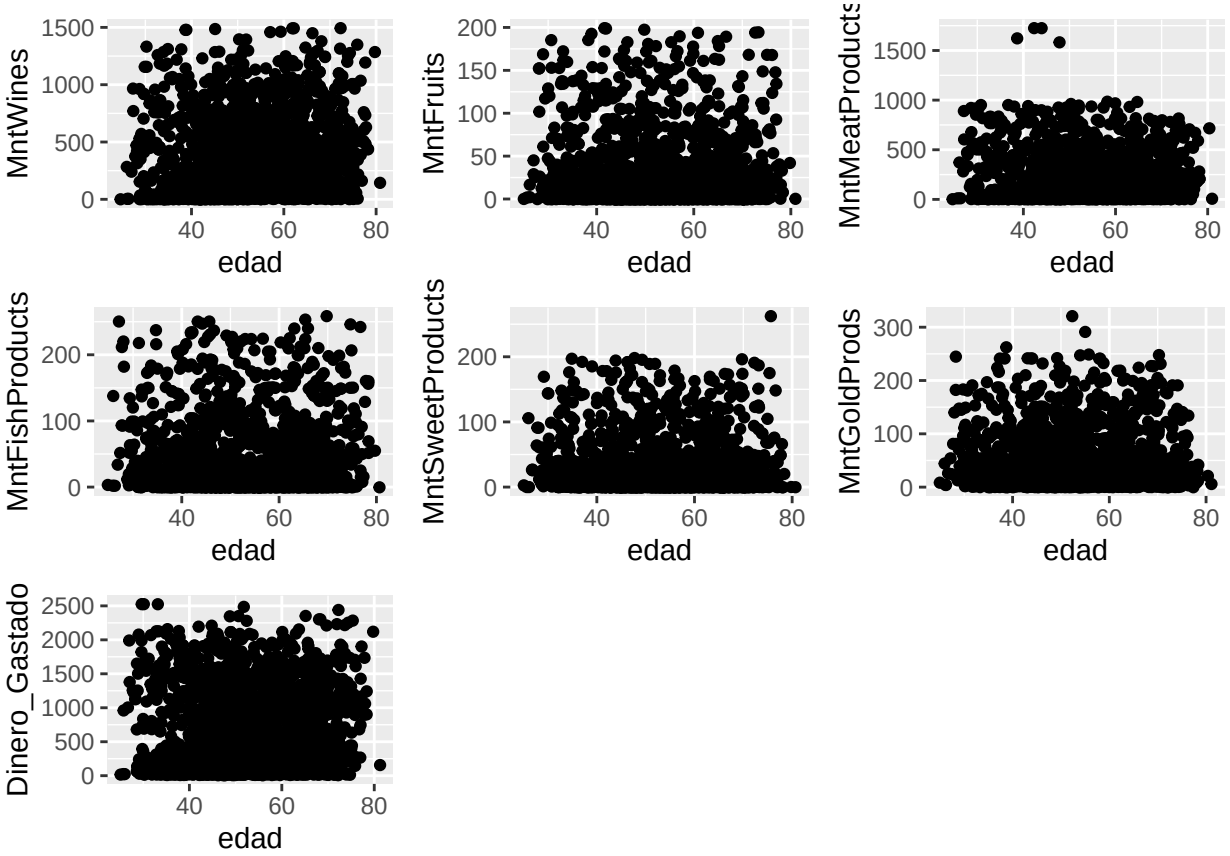
Como hemos comentado anteriormente existe un valor atípico dentro de nuestra variable, por lo que al unicamente ser uno lo eliminaremos (ya que no afectará a la información) y volvemos a examinar los datos.

```
datos <- datos %>%
  filter(edad < 100)
ggplot(datos) +
  geom_histogram(aes(x = edad, y =stat(density)), bins = 15, fill = "darkgreen",
                 color ='black') +
  geom_density(aes(x = edad), color="red", size=1.5)
```



Observamos que nuestros datos ya tienen una forma que es plausible, por lo que podemos iniciar el análisis de los datos.

```
g1 <-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntWines))
g2<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntFruits))
g3<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntMeatProducts))
g4<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntFishProducts))
g5<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntSweetProducts))
g6<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntGoldProds))
g7<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = Dinero_Gastado))
gridExtra::grid.arrange(g1,g2,g3,g4, g5, g6, g7)
```



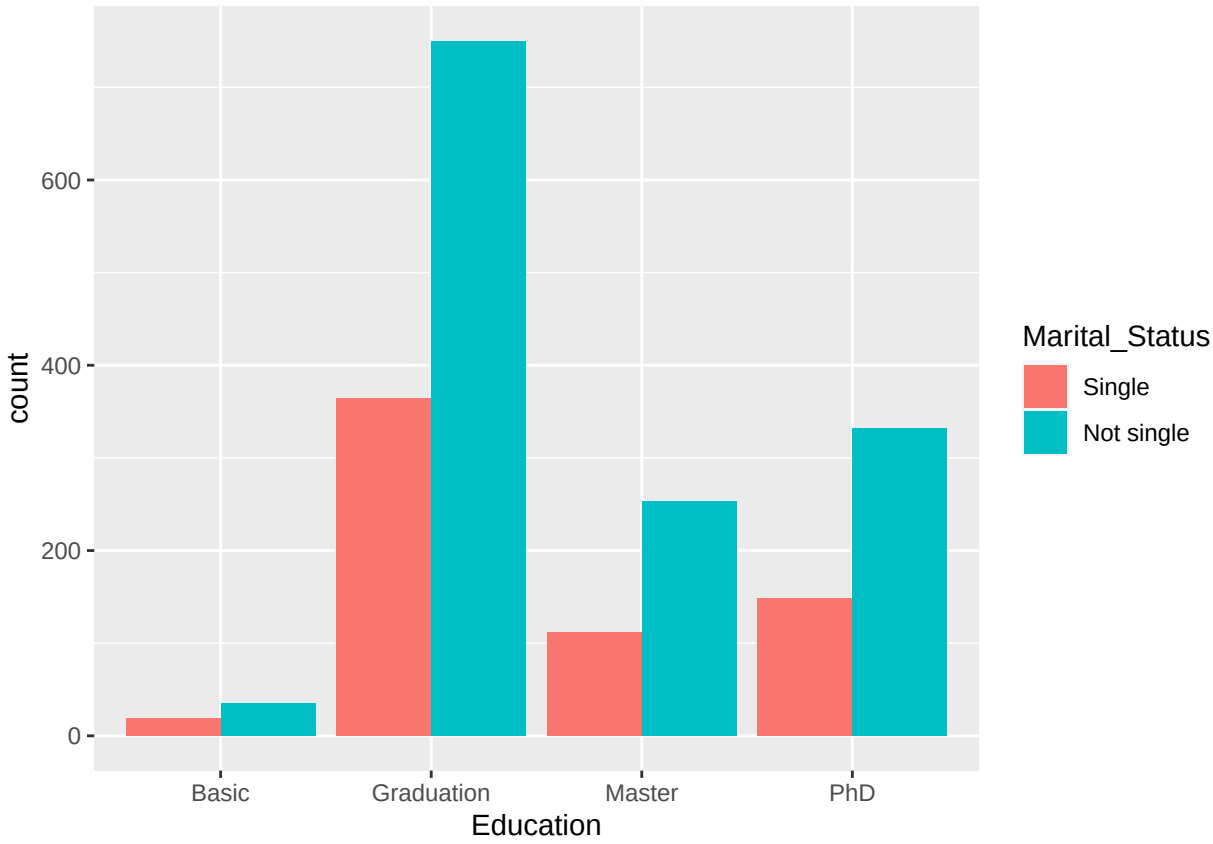
En una primera observación podemos ver que el gasto en vino es mucho mayor que en cualquiera de las otras secciones y para cualquier edad. Por otra parte, noo parece que existan patrones muy claros en ninguna de las variables. Sin embargo, podríamos decir que las personas de mayor edad gastan más dinero en vino y eso probablemente repercuten en que gasten más dinero en general.

Realizamos un último estudio en función del nivel de estudios y el estado sentimental. Pero primero vamos a ver cuantos datos hay de cada tipo.

```
table(datos$Education, datos$Marital_Status)
```

```
##
##              Single Not single
##   Basic          19         35
##   Graduation    364        750
##   Master        112        253
##   PhD           148        332
```

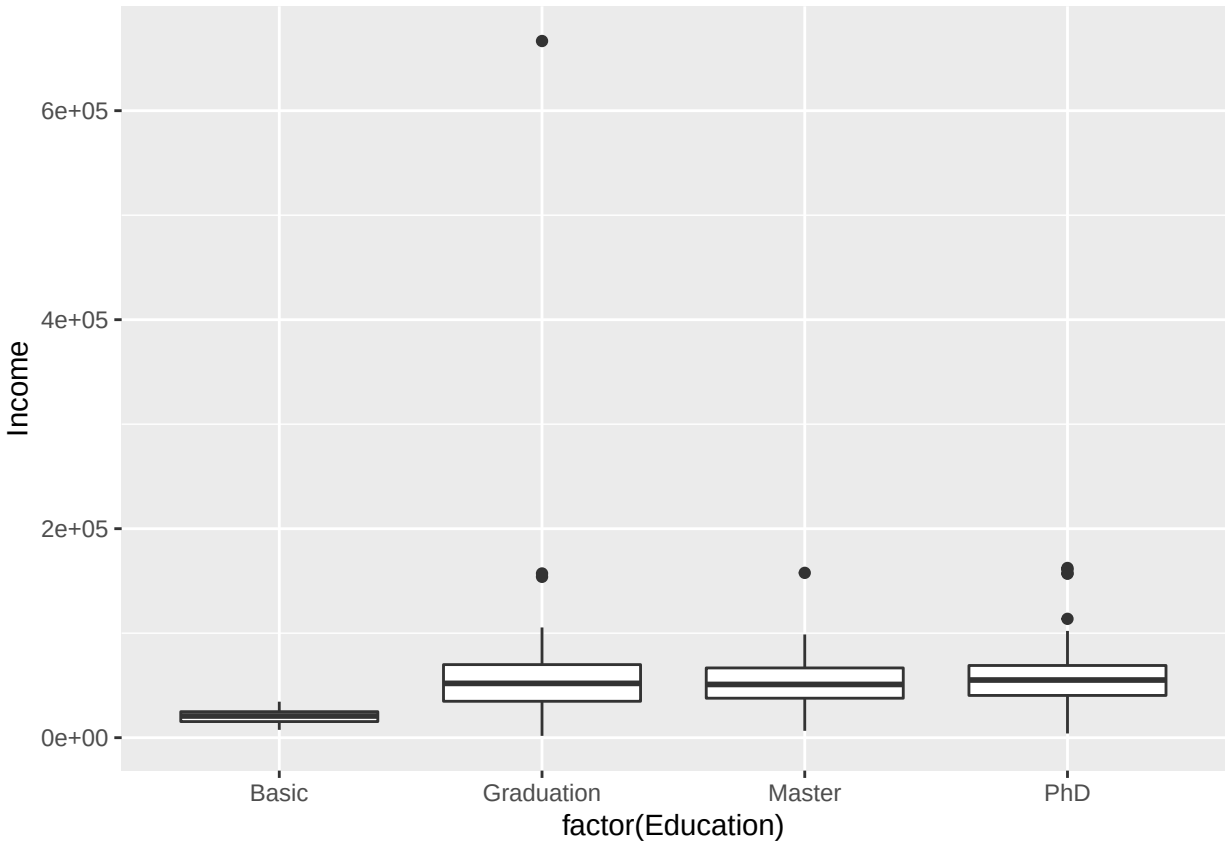
```
ggplot(datos) +
  geom_bar(aes(x = Education, fill = Marital_Status), position = "dodge")
```



Anteriormente hemos visto de forma general que el número de clientes con pareja son el doble de los sin pareja, pero ahora vemos esta relación se cumple además, para todos los grupos de niveles de estudio

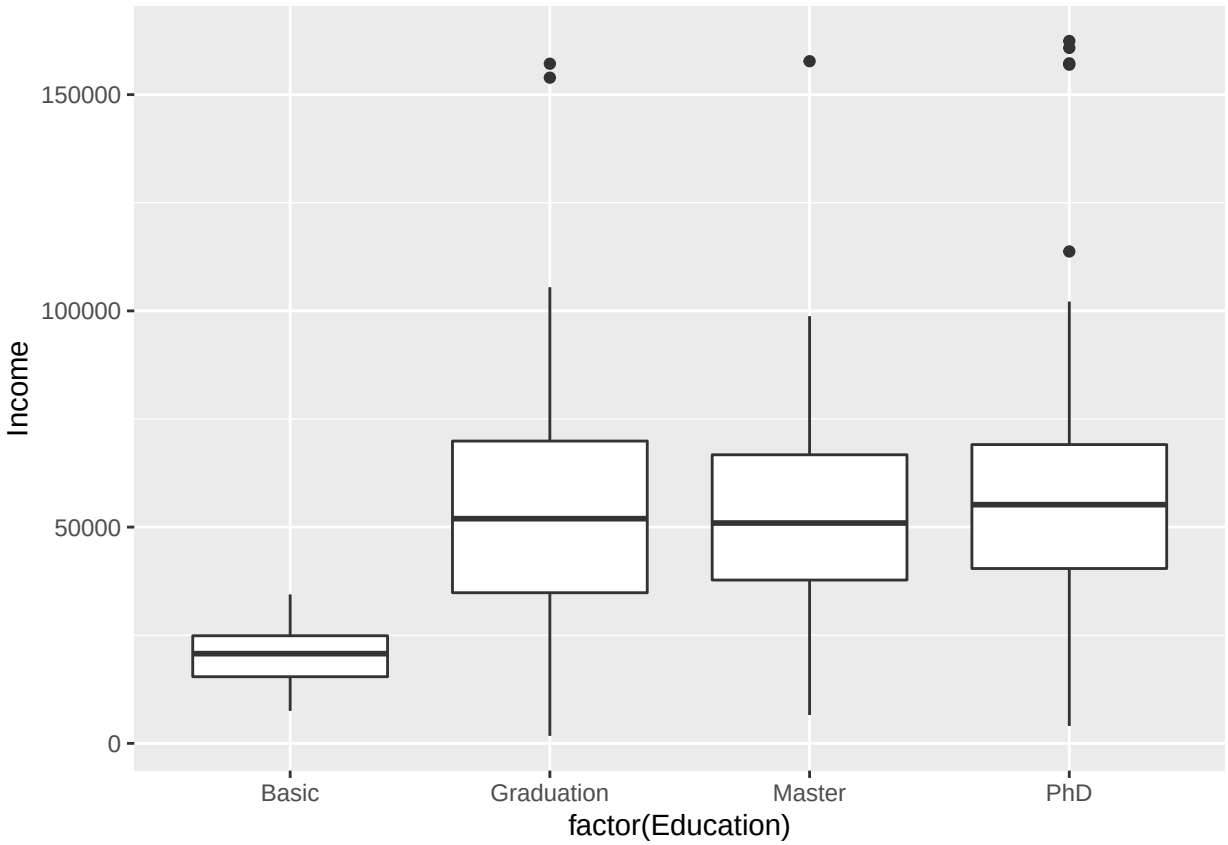
Veamos la relación entre el salario y el nivel de estudios:

```
ggplot(datos) +
  geom_boxplot(aes(x = factor(Education), y = Income))
```



Vemos que hay un outlier que no nos permite ver correctamente los gráficos. Por este motivo lo quitamos.

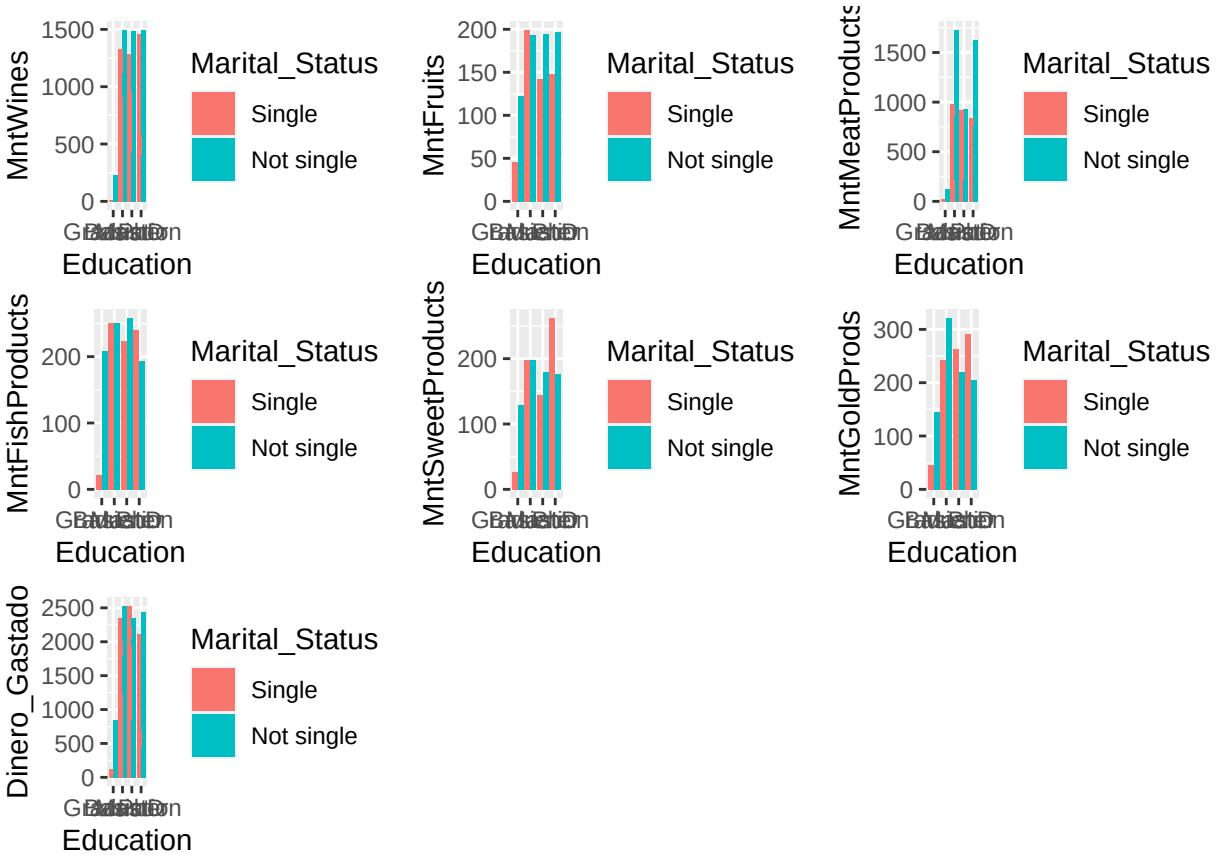
```
aux <- datos %>%
  filter(Income < 500000)
ggplot(aux) +
  geom_boxplot(aes(x = factor(Education), y = Income))
```



Se puede observar cláramente que las personas con un nivel de estudio “Basic” ganan mucho menos que cualquiera de los otros 3 grupos. Otra cosa que hay que tener en cuenta es que entre los 3 grupos restantes no existen deiferencias significativas.

Por último vemos la relación entre el gasto y el nivel de estudios y la situación sentimental.

```
g1 <- ggplot(datos) +
  geom_col(aes(x = Education, y = MntWines, fill = Marital_Status), position = "dodge")
g2<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntFruits, fill = Marital_Status), position = "dodge")
g3<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntMeatProducts, fill = Marital_Status), position = "dodge")
g4<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntFishProducts, fill = Marital_Status), position = "dodge")
g5<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntSweetProducts, fill = Marital_Status), position = "dodge")
g6<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntGoldProds, fill = Marital_Status), position = "dodge")
g7<-ggplot(datos) +
  geom_col(aes(x = Education, y = Dinero_Gastado, fill = Marital_Status), position = "dodge")
gridExtra::grid.arrange(g1,g2,g3,g4, g5, g6, g7)
```



Estos gráficos reflejan lo visto anteriormente, ya que en todas las secciones, los clientes con un nivel de estudio "Basic" gastan mucho menos dinero que cualquiera de los otros 3 grupos, lo que concuerda con que su salario sea menor.

3.2. Ajuste parámetros de control

Lo que haremos para para tener siempre un mismo resultado es usar una semilla, en nuestro caso será 2021. Por otro lado para el tema de control usaremos el método de cross-validation con un fold de 10.

```
ctrl <- trainControl(method = "cv", number = 10,summaryFunction = defaultSummary, classProbs = TRUE)
```

3.3. Análisis sobre variable sonPadres

Lo primero que haremos será dividir el conjunto de datos en dos, uno para entrenamiento y otro para test.

```
set.seed(2021)
trainIndex <- createDataPartition(datos_ML$sonPadres, p = 0.8, list = FALSE, times = 1)
train_set <- datos_ML[trainIndex,]
test_set <- datos_ML[-trainIndex,]
train_set_eval <- train_set
test_set_eval <- test_set
```

Una vez ya definidos los parámetros de control podemos comenzar con los

```r
set.seed(2021)
train_set <- train_set %>%
  mutate(sonPadres = factor(sonPadres == 1, levels = c(TRUE, FALSE), labels = c("Si", "No")))
LogReg.fit <- train(form = sonPadres ~ ., data = train_set, method = "glm",
                    preProcess = c("center","scale"), trControl = ctrl, metric = "Accuracy")
```

```r
LogReg.fit
```

```
## Generalized Linear Model
##
## 1612 samples
##    8 predictor
##    2 classes: 'Si', 'No'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1451, 1452, 1451, 1451, 1450, 1450, ...
## Resampling results:
##
##   Accuracy  Kappa
##   1         1
```

```r
summary(LogReg.fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.136e-05  -4.085e-06  -3.815e-06   3.224e-06   5.930e-06
##
## Coefficients: (1 not defined because of singularities)
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.276e+01  6.689e+03  -0.003    0.997
## MntWines         -4.479e-02  6.383e+03   0.000    1.000
## MntFruits         4.815e-02  7.122e+03   0.000    1.000
## MntMeatProducts   2.720e-01  6.831e+03   0.000    1.000
## MntFishProducts   9.437e-02  7.409e+03   0.000    1.000
## MntSweetProducts -6.553e-03  6.978e+03   0.000    1.000
## MntGoldProds     -4.992e-02  6.243e+03   0.000    1.000
## totalHijos       -3.759e+01  9.236e+03  -0.004    0.997
## Dinero_Gastado           NA         NA      NA       NA
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 1.9185e+03  on 1611  degrees of freedom
## Residual deviance: 2.3419e-08  on 1604  degrees of freedom
## AIC: 16
##
## Number of Fisher Scoring iterations: 25
```

```
str(LogReg.fit)
```

```
## List of 23
##  $ method      : chr "glm"
##  $ modelInfo   :List of 15
##   ..$ label     : chr "Generalized Linear Model"
##   ..$ library   : NULL
##   ..$ loop      : NULL
##   ..$ type      : chr [1:2] "Regression" "Classification"
##   ..$ parameters:'data.frame':   1 obs. of  3 variables:
##   .. ..$ parameter: chr "parameter"
##   .. ..$ class    : chr "character"
##   .. ..$ label    : chr "parameter"
##   ..$ grid      :function (x, y, len = NULL, search = "grid")
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 8 26 8 99 26 99 8 8
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
##   ..$ fit       :function (x, y, wts, param, lev, last, classProbs, ...)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 9 25 30 19 25 19 9 30
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
##   ..$ predict   :function (modelFit, newdata, submodels = NULL)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 31 29 42 19 29 19 31 42
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
##   ..$ prob      :function (modelFit, newdata, submodels = NULL)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 43 26 51 19 26 19 43 51
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
##   ..$ varImp    :function (object, ...)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 52 28 59 19 28 19 52 59
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
##   ..$ predictors:function (x, ...)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 60 32 60 67 32 67 60 60
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
##   ..$ levels    :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 61 28 61 93 28 93 61 61
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
##   ..$ trim      :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 62 26 85 19 26 19 62 85
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
```

```
##   ..$ tags      : chr [1:4] "Generalized Linear Model" "Linear Classifier" "Two Class Only" "Accepts
##   ..$ sort      :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 88 26 88 38 26 38 88 88
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fd80d8ee528>
##  $ modelType   : chr "Classification"
##  $ results     :'data.frame':    1 obs. of  5 variables:
##   ..$ parameter : chr "none"
##   ..$ Accuracy  : num 1
##   ..$ Kappa     : num 1
##   ..$ AccuracySD: num 0
##   ..$ KappaSD   : num 0
##  $ pred        : NULL
##  $ bestTune    :'data.frame':    1 obs. of  1 variable:
##   ..$ parameter: chr "none"
##  $ call        : language train.formula(form = sonPadres ~ ., data = train_set, method = "glm", preP
##  $ dots        : list()
##  $ metric      : chr "Accuracy"
##  $ control     :List of 27
##   ..$ method           : chr "cv"
##   ..$ number           : num 10
##   ..$ repeats          : logi NA
##   ..$ search           : chr "grid"
##   ..$ p                : num 0.75
##   ..$ initialWindow    : NULL
##   ..$ horizon          : num 1
##   ..$ fixedWindow      : logi TRUE
##   ..$ skip             : num 0
##   ..$ verboseIter      : logi FALSE
##   ..$ returnData       : logi TRUE
##   ..$ returnResamp     : chr "final"
##   ..$ savePredictions  : chr "none"
##   ..$ classProbs       : logi TRUE
##   ..$ summaryFunction  :function (data, lev = NULL, model = NULL)
##   ..$ selectionFunction: chr "best"
##   ..$ preProcOptions   :List of 6
##   .. ..$ thresh   : num 0.95
##   .. ..$ ICAcomp  : num 3
##   .. ..$ k        : num 5
##   .. ..$ freqCut  : num 19
##   .. ..$ uniqueCut: num 10
##   .. ..$ cutoff   : num 0.9
##   ..$ sampling         : NULL
##   ..$ index            :List of 10
```

```
##   .. ..$ Fold01: int [1:1451] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ Fold02: int [1:1452] 1 2 3 4 6 7 8 10 11 12 ...
##   .. ..$ Fold03: int [1:1451] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ Fold04: int [1:1451] 1 2 3 4 5 6 7 9 10 11 ...
##   .. ..$ Fold05: int [1:1450] 1 2 3 4 5 8 9 10 11 12 ...
##   .. ..$ Fold06: int [1:1450] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ Fold07: int [1:1451] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ Fold08: int [1:1451] 1 3 4 5 6 7 8 9 11 12 ...
##   .. ..$ Fold09: int [1:1451] 2 3 5 6 7 8 9 10 11 12 ...
##   .. ..$ Fold10: int [1:1450] 1 2 4 5 6 7 8 9 10 12 ...
##   ..$ indexOut         :List of 10
##   .. ..$ Resample01: int [1:161] 27 32 33 56 65 66 86 90 96 100 ...
##   .. ..$ Resample02: int [1:160] 5 9 15 18 47 48 53 58 71 73 ...
##   .. ..$ Resample03: int [1:161] 13 16 25 39 40 54 57 62 63 74 ...
##   .. ..$ Resample04: int [1:161] 8 14 24 59 64 72 83 87 124 128 ...
##   .. ..$ Resample05: int [1:162] 6 7 41 45 55 80 109 116 123 142 ...
##   .. ..$ Resample06: int [1:162] 30 31 34 35 38 49 79 84 88 93 ...
##   .. ..$ Resample07: int [1:161] 12 20 21 26 42 50 78 85 89 97 ...
##   .. ..$ Resample08: int [1:161] 2 10 29 43 44 46 51 60 68 69 ...
##   .. ..$ Resample09: int [1:161] 1 4 19 22 28 36 52 61 67 75 ...
##   .. ..$ Resample10: int [1:162] 3 11 17 23 37 82 99 108 130 135 ...
##   ..$ indexFinal       : NULL
##   ..$ timingSamps      : num 0
##   ..$ predictionBounds : logi [1:2] FALSE FALSE
##   ..$ seeds            :List of 11
##   .. ..$ : int 876986
##   .. ..$ : int 439180
##   .. ..$ : int 214086
##   .. ..$ : int 710246
##   .. ..$ : int 889157
##   .. ..$ : int 933991
##   .. ..$ : int 581143
##   .. ..$ : int 756659
##   .. ..$ : int 783506
##   .. ..$ : int 672956
##   .. ..$ : int 846211
##   ..$ adaptive         :List of 4
##   .. ..$ min     : num 5
##   .. ..$ alpha   : num 0.05
##   .. ..$ method  : chr "gls"
##   .. ..$ complete: logi TRUE
##   ..$ trim             : logi FALSE
##   ..$ allowParallel    : logi TRUE
```

```
##  $ finalModel  :List of 34
##   ..$ coefficients     : Named num [1:9] -22.7628 -0.0448 0.0481 0.272 0.0944 ...
##   .. ..- attr(*, "names")= chr [1:9] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   ..$ residuals        : Named num [1:1612] 1 -1 -1 -1 -1 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ fitted.values    : Named num [1:1612] 1.00 8.40e-12 1.02e-11 8.74e-12 2.22e-16 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ effects          : Named num [1:1612] 0.001301 0.000917 -0.00075 0.00083 0.000358 ...
##   .. ..- attr(*, "names")= chr [1:1612] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   ..$ R                : num [1:9, 1:9] -0.000178 0 0 0 0 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:9] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. .. ..$ : chr [1:9] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   ..$ rank             : int 8
##   ..$ qr               :List of 5
##   .. ..$ qr   : num [1:1612, 1:9] -1.78e-04 2.68e-02 2.95e-02 2.73e-02 8.35e-05 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:1612] "X1" "X2" "X3" "X4" ...
##   .. .. .. ..$ : chr [1:9] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. ..$ rank : int 8
##   .. ..$ qraux: num [1:9] 1.02 1.03 1.01 1.02 1 ...
##   .. ..$ pivot: int [1:9] 1 2 3 4 5 6 7 8 9
##   .. ..$ tol  : num 1e-11
##   .. ..- attr(*, "class")= chr "qr"
##   ..$ family           :List of 12
##   .. ..$ family    : chr "binomial"
##   .. ..$ link      : chr "logit"
##   .. ..$ linkfun   :function (mu)
##   .. ..$ linkinv   :function (eta)
##   .. ..$ variance  :function (mu)
##   .. ..$ dev.resids:function (y, mu, wt)
##   .. ..$ aic       :function (y, n, mu, wt, dev)
##   .. ..$ mu.eta    :function (eta)
##   .. ..$ initialize: language {     if (NCOL(y) == 1) { ...
##   .. ..$ validmu   :function (mu)
##   .. ..$ valideta  :function (eta)
##   .. ..$ simulate  :function (object, nsim)
##   .. ..- attr(*, "class")= chr "family"
##   ..$ linear.predictors: Named num [1:1612] 25.9 -25.5 -25.3 -25.5 -76.1 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ deviance         : num 2.34e-08
##   ..$ aic              : num 16
##   ..$ null.deviance    : num 1919
```

```
##   ..$ iter             : int 25
##   ..$ weights          : Named num [1:1612] 1.59e-11 2.28e-11 2.78e-11 2.38e-11 2.22e-16 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ prior.weights    : Named num [1:1612] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ df.residual      : int 1604
##   ..$ df.null          : int 1611
##   ..$ y                : Named num [1:1612] 1 0 0 0 0 1 1 0 1 1 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ converged        : logi FALSE
##   ..$ boundary         : logi FALSE
##   ..$ model            :'data.frame':    1612 obs. of  9 variables:
##   .. ..$ .outcome        : Factor w/ 2 levels "Si","No": 2 1 1 1 1 2 2 1 2 2 ...
##   .. ..$ MntWines        : num [1:1612] 0.934 -0.887 -0.414 0.598 -0.838 ...
##   .. ..$ MntFruits       : num [1:1612] 1.549 -0.543 0.429 0.404 -0.642 ...
##   .. ..$ MntMeatProducts : num [1:1612] 1.672 -0.656 -0.222 -0.311 -0.718 ...
##   .. ..$ MntFishProducts : num [1:1612] 2.616 -0.489 0.201 -0.681 -0.662 ...
##   .. ..$ MntSweetProducts: num [1:1612] 1.5853 -0.5729 0.0365 0.4173 -0.6237 ...
##   .. ..$ MntGoldProds    : num [1:1612] 0.853 -0.746 -0.554 -0.573 -0.592 ...
##   .. ..$ totalHijos      : num [1:1612] -1.2757 0.0684 0.0684 0.0684 1.4124 ...
##   .. ..$ Dinero_Gastado  : num [1:1612] 1.647 -0.921 -0.315 0.168 -0.927 ...
##   .. ..- attr(*, "terms")=Classes 'terms', 'formula'  language .outcome ~ MntWines + MntFruits + Mntl
##   .. .. .. ..- attr(*, "variables")= language list(.outcome, MntWines, MntFruits, MntMeatProducts, Mn
##   .. .. .. ..- attr(*, "factors")= int [1:9, 1:8] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:9] ".outcome" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. .. .. .. .. ..$ : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. .. .. ..- attr(*, "term.labels")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishPro
##   .. .. .. ..- attr(*, "order")= int [1:8] 1 1 1 1 1 1 1 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: 0x7fd7f997d9a8>
##   .. .. .. ..- attr(*, "predvars")= language list(.outcome, MntWines, MntFruits, MntMeatProducts, Mnt
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:9] "factor" "numeric" "numeric" "numeric" ...
##   .. .. .. .. ..- attr(*, "names")= chr [1:9] ".outcome" "MntWines" "MntFruits" "MntMeatProducts" ..
##   ..$ formula          :Class 'formula'  language .outcome ~ .
##   .. .. ..- attr(*, ".Environment")=<environment: 0x7fd7f997d9a8>
##   ..$ terms            :Classes 'terms', 'formula'  language .outcome ~ MntWines + MntFruits + MntMea
##   .. .. ..- attr(*, "variables")= language list(.outcome, MntWines, MntFruits, MntMeatProducts, MntFi
##   .. .. ..- attr(*, "factors")= int [1:9, 1:8] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:9] ".outcome" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. .. .. .. ..$ : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
```

```
##   .. .. ..- attr(*, "term.labels")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProduc
##   .. .. ..- attr(*, "order")= int [1:8] 1 1 1 1 1 1 1 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: 0x7fd7f997d9a8>
##   .. .. ..- attr(*, "predvars")= language list(.outcome, MntWines, MntFruits, MntMeatProducts, MntFis
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:9] "factor" "numeric" "numeric" "numeric" ...
##   .. .. .. ..- attr(*, "names")= chr [1:9] ".outcome" "MntWines" "MntFruits" "MntMeatProducts" ...
##   ..$ data             :'data.frame':    1612 obs. of  9 variables:
##   .. ..$ MntWines        : num [1:1612] 0.934 -0.887 -0.414 0.598 -0.838 ...
##   .. ..$ MntFruits       : num [1:1612] 1.549 -0.543 0.429 0.404 -0.642 ...
##   .. ..$ MntMeatProducts : num [1:1612] 1.672 -0.656 -0.222 -0.311 -0.718 ...
##   .. ..$ MntFishProducts : num [1:1612] 2.616 -0.489 0.201 -0.681 -0.662 ...
##   .. ..$ MntSweetProducts: num [1:1612] 1.5853 -0.5729 0.0365 0.4173 -0.6237 ...
##   .. ..$ MntGoldProds    : num [1:1612] 0.853 -0.746 -0.554 -0.573 -0.592 ...
##   .. ..$ totalHijos      : num [1:1612] -1.2757 0.0684 0.0684 0.0684 1.4124 ...
##   .. ..$ Dinero_Gastado  : num [1:1612] 1.647 -0.921 -0.315 0.168 -0.927 ...
##   .. ..$ .outcome        : Factor w/ 2 levels "Si","No": 2 1 1 1 1 2 2 1 2 2 ...
##   ..$ offset           : NULL
##   ..$ control          :List of 3
##   .. ..$ epsilon: num 1e-08
##   .. ..$ maxit  : num 25
##   .. ..$ trace  : logi FALSE
##   ..$ method           : chr "glm.fit"
##   ..$ contrasts        : NULL
##   ..$ xlevels          : Named list()
##   ..$ xNames           : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   ..$ problemType      : chr "Classification"
##   ..$ tuneValue        :'data.frame':    1 obs. of  1 variable:
##   .. ..$ parameter: chr "none"
##   ..$ obsLevels        : chr [1:2] "Si" "No"
##   .. ..- attr(*, "ordered")= logi FALSE
##   ..$ param            : list()
##   ..- attr(*, "class")= chr [1:2] "glm" "lm"
##  $ preProcess  :List of 22
##   ..$ dim              : int [1:2] 1612 8
##   ..$ bc               : NULL
##   ..$ yj               : NULL
##   ..$ et               : NULL
##   ..$ invHyperbolicSine: NULL
##   ..$ mean             : Named num [1:8] 315 25.8 168.2 35.5 25.6 ...
##   .. ..- attr(*, "names")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   ..$ std              : Named num [1:8] 342.6 40.2 226 52.2 39.4 ...
```

```
##   .. ..- attr(*, "names")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   ..$ ranges           : NULL
##   ..$ rotation         : NULL
##   ..$ method           :List of 3
##   .. ..$ center: chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. ..$ scale : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. ..$ ignore: chr(0)
##   ..$ thresh           : num 0.95
##   ..$ pcaComp          : NULL
##   ..$ numComp          : NULL
##   ..$ ica              : NULL
##   ..$ wildcards        :List of 2
##   .. ..$ PCA: chr(0)
##   .. ..$ ICA: chr(0)
##   ..$ k                : num 5
##   ..$ knnSummary       :function (x, ...)
##   ..$ bagImp           : NULL
##   ..$ median           : NULL
##   ..$ data             : NULL
##   ..$ rangeBounds      : num [1:2] 0 1
##   ..$ call             : chr "scrubed"
##   ..- attr(*, "class")= chr "preProcess"
##  $ trainingData: rowwise_df [1,612 x 9] (S3: rowwise_df/tbl_df/tbl/data.frame)
##   ..$ .outcome        : Factor w/ 2 levels "Si","No": 2 1 1 1 1 2 2 1 2 2 ...
##   ..$ MntWines        : int [1:1612] 635 11 173 520 28 6 194 233 3 1006 ...
##   ..$ MntFruits       : int [1:1612] 88 4 43 42 0 16 61 2 14 22 ...
##   ..$ MntMeatProducts : int [1:1612] 546 20 118 98 6 11 480 53 17 115 ...
##   ..$ MntFishProducts : int [1:1612] 172 10 46 0 1 11 225 3 6 59 ...
##   ..$ MntSweetProducts: int [1:1612] 88 3 27 42 1 1 112 5 1 68 ...
##   ..$ MntGoldProds    : int [1:1612] 88 5 15 14 13 16 30 14 5 45 ...
##   ..$ totalHijos      : num [1:1612] 0 1 1 1 2 0 0 2 0 0 ...
##   ..$ Dinero_Gastado  : int [1:1612] 1617 53 422 716 49 61 1102 310 46 1315 ...
##   ..- attr(*, "groups")= tibble [1,612 x 1] (S3: tbl_df/tbl/data.frame)
##   .. ..$ .rows: list<int> [1:1612]
##   .. .. ..$ : int 1
##   .. .. ..$ : int 2
##   .. .. ..$ : int 3
##   .. .. ..$ : int 4
##   .. .. ..$ : int 5
##   .. .. ..$ : int 6
##   .. .. ..$ : int 7
##   .. .. ..$ : int 8
##   .. .. ..$ : int 9
```

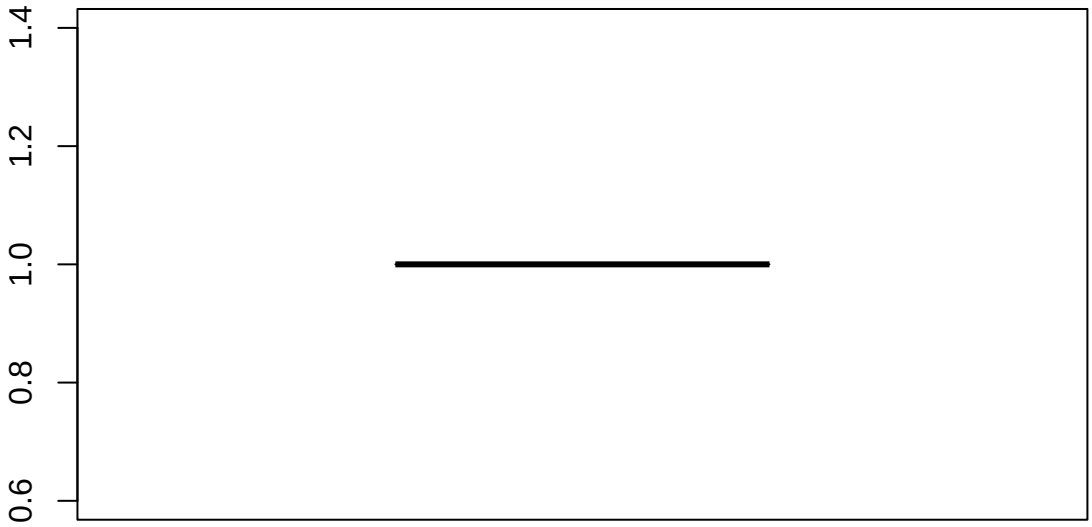
```
##   .. .. ..$ : int 10
##   .. .. ..$ : int 11
##   .. .. ..$ : int 12
##   .. .. ..$ : int 13
##   .. .. ..$ : int 14
##   .. .. ..$ : int 15
##   .. .. ..$ : int 16
##   .. .. ..$ : int 17
##   .. .. ..$ : int 18
##   .. .. ..$ : int 19
##   .. .. ..$ : int 20
##   .. .. ..$ : int 21
##   .. .. ..$ : int 22
##   .. .. ..$ : int 23
##   .. .. ..$ : int 24
##   .. .. ..$ : int 25
##   .. .. ..$ : int 26
##   .. .. ..$ : int 27
##   .. .. ..$ : int 28
##   .. .. ..$ : int 29
##   .. .. ..$ : int 30
##   .. .. ..$ : int 31
##   .. .. ..$ : int 32
##   .. .. ..$ : int 33
##   .. .. ..$ : int 34
##   .. .. ..$ : int 35
##   .. .. ..$ : int 36
##   .. .. ..$ : int 37
##   .. .. ..$ : int 38
##   .. .. ..$ : int 39
##   .. .. ..$ : int 40
##   .. .. ..$ : int 41
##   .. .. ..$ : int 42
##   .. .. ..$ : int 43
##   .. .. ..$ : int 44
##   .. .. ..$ : int 45
##   .. .. ..$ : int 46
##   .. .. ..$ : int 47
##   .. .. ..$ : int 48
##   .. .. ..$ : int 49
##   .. .. ..$ : int 50
##   .. .. ..$ : int 51
##   .. .. ..$ : int 52
```

```
##   .. .. ..$ : int 53
##   .. .. ..$ : int 54
##   .. .. ..$ : int 55
##   .. .. ..$ : int 56
##   .. .. ..$ : int 57
##   .. .. ..$ : int 58
##   .. .. ..$ : int 59
##   .. .. ..$ : int 60
##   .. .. ..$ : int 61
##   .. .. ..$ : int 62
##   .. .. ..$ : int 63
##   .. .. ..$ : int 64
##   .. .. ..$ : int 65
##   .. .. ..$ : int 66
##   .. .. ..$ : int 67
##   .. .. ..$ : int 68
##   .. .. ..$ : int 69
##   .. .. ..$ : int 70
##   .. .. ..$ : int 71
##   .. .. ..$ : int 72
##   .. .. ..$ : int 73
##   .. .. ..$ : int 74
##   .. .. ..$ : int 75
##   .. .. ..$ : int 76
##   .. .. ..$ : int 77
##   .. .. ..$ : int 78
##   .. .. ..$ : int 79
##   .. .. ..$ : int 80
##   .. .. ..$ : int 81
##   .. .. ..$ : int 82
##   .. .. ..$ : int 83
##   .. .. ..$ : int 84
##   .. .. ..$ : int 85
##   .. .. ..$ : int 86
##   .. .. ..$ : int 87
##   .. .. ..$ : int 88
##   .. .. ..$ : int 89
##   .. .. ..$ : int 90
##   .. .. ..$ : int 91
##   .. .. ..$ : int 92
##   .. .. ..$ : int 93
##   .. .. ..$ : int 94
##   .. .. ..$ : int 95
```

```
##   .. .. ..$ : int 96
##   .. .. ..$ : int 97
##   .. .. ..$ : int 98
##   .. .. ..$ : int 99
##   .. .. .. [list output truncated]
##   .. .. ..@ ptype: int(0)
##  $ resample    :'data.frame':    10 obs. of  3 variables:
##   ..$ Accuracy: num [1:10] 1 1 1 1 1 1 1 1 1 1
##   ..$ Kappa   : num [1:10] 1 1 1 1 1 1 1 1 1 1
##   ..$ Resample: chr [1:10] "Fold01" "Fold02" "Fold03" "Fold04" ...
##  $ resampledCM :'data.frame':    10 obs. of  6 variables:
##   ..$ cell1    : num [1:10] 116 115 115 116 116 116 116 115 116 116
##   ..$ cell2    : num [1:10] 0 0 0 0 0 0 0 0 0 0
##   ..$ cell3    : num [1:10] 0 0 0 0 0 0 0 0 0 0
##   ..$ cell4    : num [1:10] 45 45 46 45 46 46 45 46 45 46
##   ..$ parameter: chr [1:10] "none" "none" "none" "none" ...
##   ..$ Resample : chr [1:10] "Fold01" "Fold02" "Fold03" "Fold04" ...
##  $ perfNames   : chr [1:2] "Accuracy" "Kappa"
##  $ maximize    : logi TRUE
##  $ yLimits     : NULL
##  $ times       :List of 3
##   ..$ everything: 'proc_time' Named num [1:5] 0.984 0.058 1.066 0 0
##   .. ..- attr(*, "names")= chr [1:5] "user.self" "sys.self" "elapsed" "user.child" ...
##   ..$ final     : 'proc_time' Named num [1:5] 0.026 0.003 0.028 0 0
##   .. ..- attr(*, "names")= chr [1:5] "user.self" "sys.self" "elapsed" "user.child" ...
##   ..$ prediction: logi [1:3] NA NA NA
##  $ levels      : chr [1:2] "Si" "No"
##   ..- attr(*, "ordered")= logi FALSE
##  $ terms       :Classes 'terms', 'formula'  language sonPadres ~ MntWines + MntFruits + MntMeatProduc
##   .. ..- attr(*, "variables")= language list(sonPadres, MntWines, MntFruits, MntMeatProducts, MntFish
##   .. ..- attr(*, "factors")= int [1:9, 1:8] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:9] "sonPadres" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. .. .. ..$ : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. ..- attr(*, "term.labels")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts
##   .. ..- attr(*, "order")= int [1:8] 1 1 1 1 1 1 1 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(sonPadres, MntWines, MntFruits, MntMeatProducts, MntFish
##   .. ..- attr(*, "dataClasses")= Named chr [1:9] "factor" "numeric" "numeric" "numeric" ...
##   .. .. ..- attr(*, "names")= chr [1:9] "sonPadres" "MntWines" "MntFruits" "MntMeatProducts" ...
##  $ coefnames   : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
```

```
##  $ xlevels     : Named list()
##  - attr(*, "class")= chr [1:2] "train" "train.formula"
```

```
boxplot(LogReg.fit$resample$Accuracy, xlab = "Accuracy")
```



Nacho por favor explicame por que creamos un cjto de datos nuevos que en si no hace falta y termina tu este modelo que no

3.4. Análisis sobre variable Complain

Otro análisis que se puede realizar mediante técnicas de machine learning es encontrar aquellas variables que son claves a la hora de detectar de forma anticipada que clientes se pueden quejar. Para ello trabajaremos y realizaremos diferentes modelos de clasificación.

Lo primero antes de iniciar cualquier modelo lo que haremos será ver como está distribuida la variable complain y también cambiarla para poder trabajar con ella:

```
datos$Complain <- ifelse(datos$Complain == 1, 'Yes','No')
table(datos$Complain)
```

```
##
##   No  Yes
## 1996   17
```

Tenemos que el dataset está totalmente desbalanceado, para evitar este problema lo que haremos será rebalancear nuestros datos.

```
set.seed(2021)
datos_comp_rebal <- ovun.sample(Complain ~ ., data = datos, method = 'both',
                                N = table(datos$Complain)[1]*2)$data
```

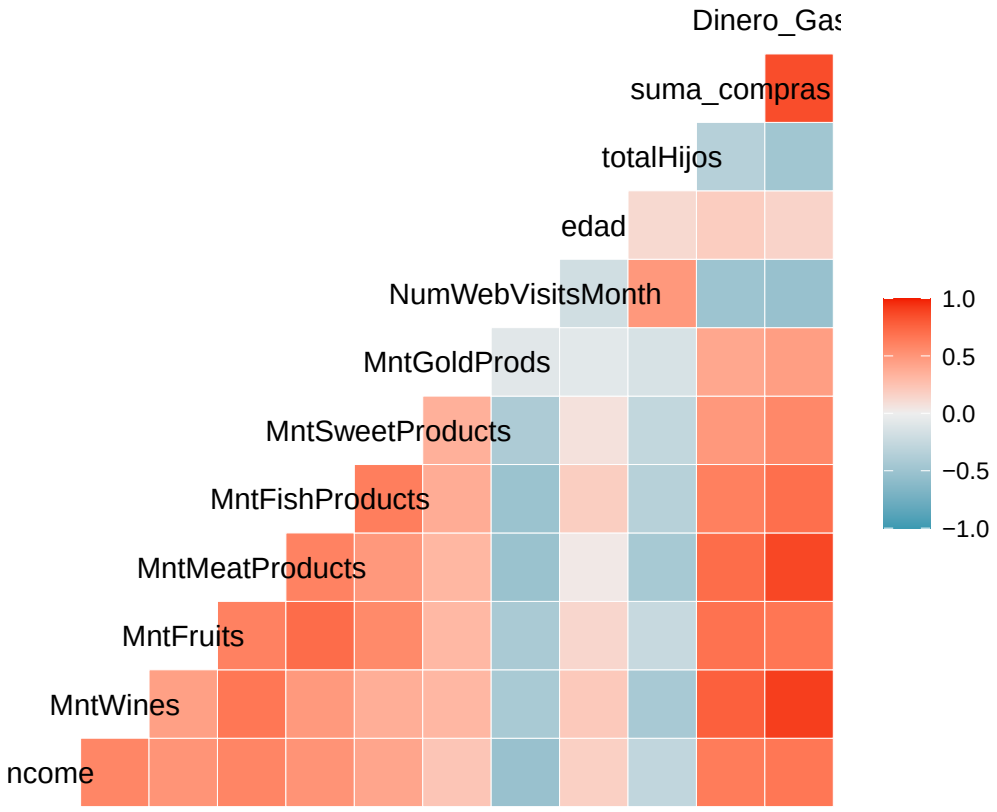
```
table(datos_comp_rebal$Complain)
```

```
##
##   No  Yes
## 2027 1965
```

```
datos_comp_rebal <- datos_comp_rebal %>%
  select(-ID)
datos_comp_rebal$Complain <- as.factor(datos_comp_rebal$Complain)
```

Una vez que tenemos los datos rebalanceados podemos pasar al siguiente paso, que consistirá en mirar si tenemos variables que tengan una alta correlación:

```
ggcorr(datos_comp_rebal)
```



Vemos que muchas variables tienen una alta correlación entre si, por ahora no haremos nada pero posteriormente veremos si es necesario eliminar alguna o no.

Una vez hecho esto podemos pasar a la parte de modelos, lo primero que haremos será dividir el conjunto de datos entre entrenamiento y test.

```
set.seed(2021)
trainIndex2 <- createDataPartition(datos_comp_rebal$Complain, p = 0.8, list = FALSE, times = 1)
fTR2 <- datos_comp_rebal[trainIndex2,]
fTS2 <- datos_comp_rebal[-trainIndex2,]
fTR2_eval <- fTR2
```

```r
fTS2_eval <- fTS2
```

Una vez definido tanto el conjunto de entrenamiento como el de test realizaremos un modelo sencillo para ver como se comporta todo, este será una regresión lineal.

```r
set.seed(2021)
LogReg2.fit <- train(form = Complain ~ ., data = fTR2, method = "glm",
                     trControl = ctrl, metric = "Accuracy")
LogReg2.fit
```

```
## Generalized Linear Model
##
## 3194 samples
##   16 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2875, 2874, 2875, 2875, 2875, 2874, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7069357  0.4140293
```

```r
summary(LogReg2.fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.79747  -0.83460  -0.00053   0.83918   1.34729
##
## Coefficients: (1 not defined because of singularities)
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -1.495e+01  1.980e+02  -0.075 0.939825
## EducationGraduation         1.637e+01  1.980e+02   0.083 0.934103
## EducationMaster             1.567e+01  1.980e+02   0.079 0.936912
## EducationPhD                1.380e+01  1.980e+02   0.070 0.944443
## `Marital_StatusNot single` -7.128e-01  1.124e-01  -6.344 2.24e-10 ***
## Income                     -1.780e-06  2.161e-06  -0.824 0.410176
## Dt_Customer.L              -1.058e+00  1.006e-01 -10.518  < 2e-16 ***
## Dt_Customer.Q               3.357e-01  8.705e-02   3.857 0.000115 ***
## MntWines                   -2.759e-03  3.899e-04  -7.077 1.47e-12 ***
```

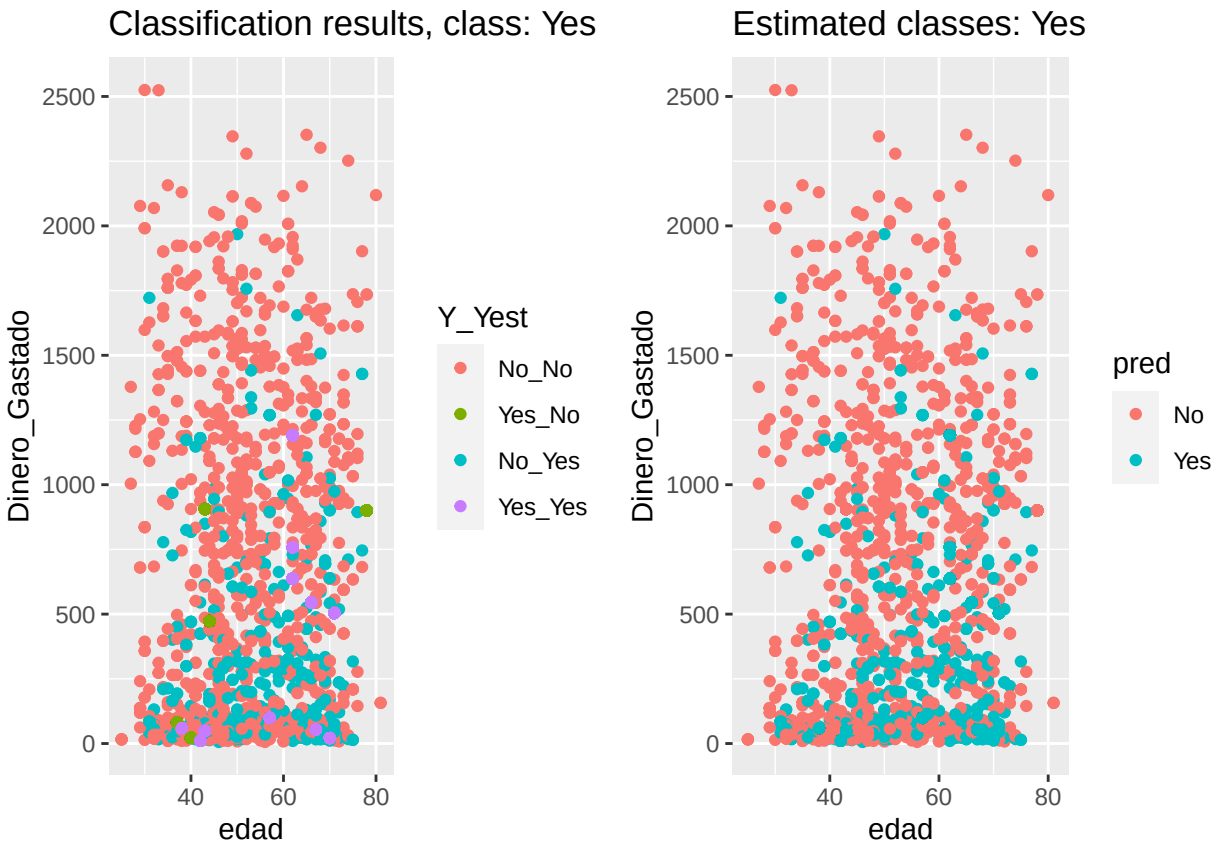
```
## MntFruits                   1.757e-02  2.448e-03   7.178 7.06e-13 ***
## MntMeatProducts            -2.869e-03  5.185e-04  -5.533 3.15e-08 ***
## MntFishProducts            -1.932e-03  1.858e-03  -1.040 0.298366
## MntSweetProducts           -2.358e-02  2.563e-03  -9.199  < 2e-16 ***
## MntGoldProds               -1.435e-02  1.409e-03 -10.183  < 2e-16 ***
## NumWebVisitsMonth          -2.455e-01  3.163e-02  -7.761 8.45e-15 ***
## Response1                   1.110e+00  1.802e-01   6.162 7.18e-10 ***
## edad                        2.517e-02  4.269e-03   5.895 3.75e-09 ***
## totalHijos                  1.233e-01  7.844e-02   1.572 0.116008
## suma_compras                6.074e-02  1.505e-02   4.036 5.43e-05 ***
## Dinero_Gastado                     NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4427.0  on 3193  degrees of freedom
## Residual deviance: 3266.7  on 3175  degrees of freedom
## AIC: 3304.7
##
## Number of Fisher Scoring iterations: 14
```

Vemos que las variables importantes en nuestro modelo son el estado civil, la antigüedad del cliente, el consumo en diferentes productos, las veces que visitan la web, si respondes a la ofertas, la edad y el total de compras. Cabe destacar que este modelo solo detecta importancia de variables lineales con el output.

Evaluamos nuestro modelo para obtener datos más claros.

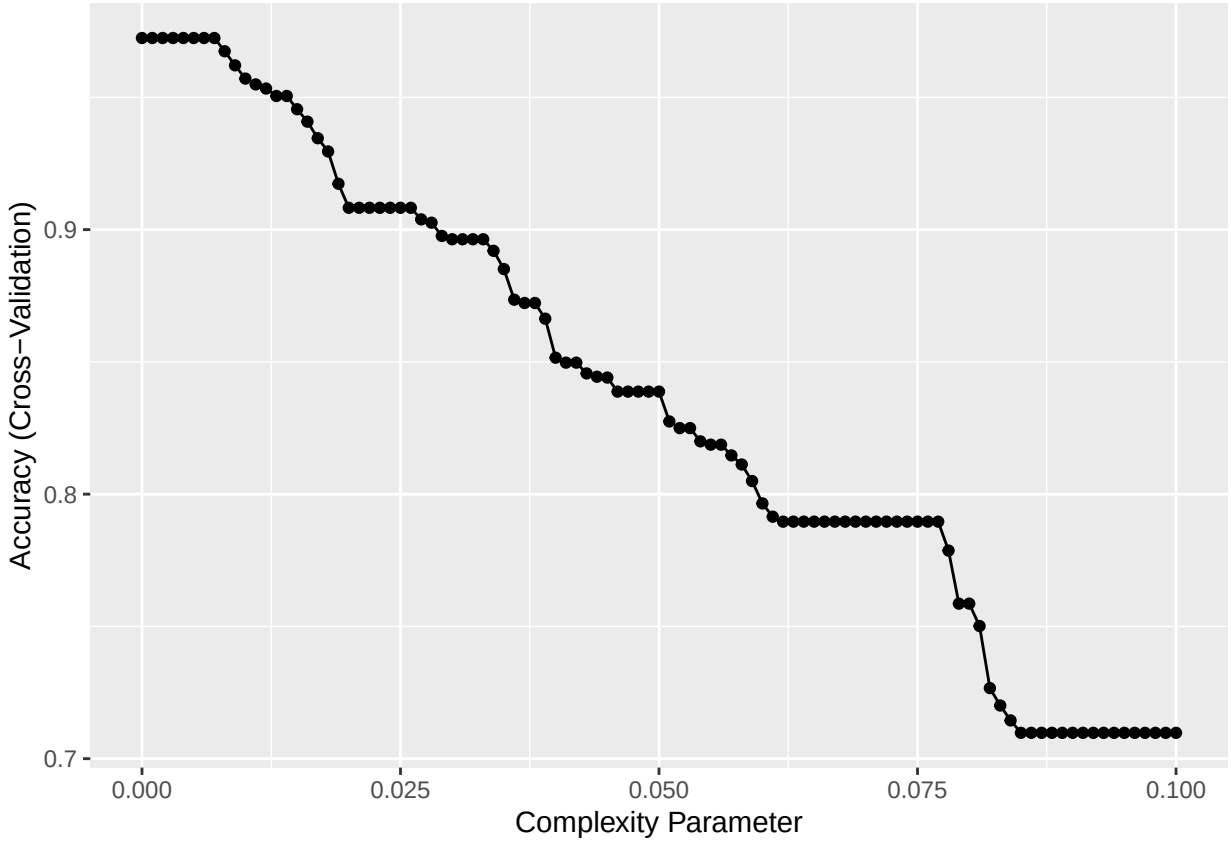
```
set.seed(2021)
fTR2_eval$LRprob <- predict(LogReg2.fit, type="prob", newdata = fTR2)
fTR2_eval$LRpred <- predict(LogReg2.fit, type="raw", newdata = fTR2)
fTS2_eval$LRprob <- predict(LogReg2.fit, type="prob", newdata = fTS2)
fTS2_eval$LRpred <- predict(LogReg2.fit, type="raw", newdata = fTS2)
```

```
Plot2DClass(fTR2, fTR2$Complain,LogReg2.fit, var1 = "edad", var2 = "Dinero_Gastado",
            selClass = "Yes")
```

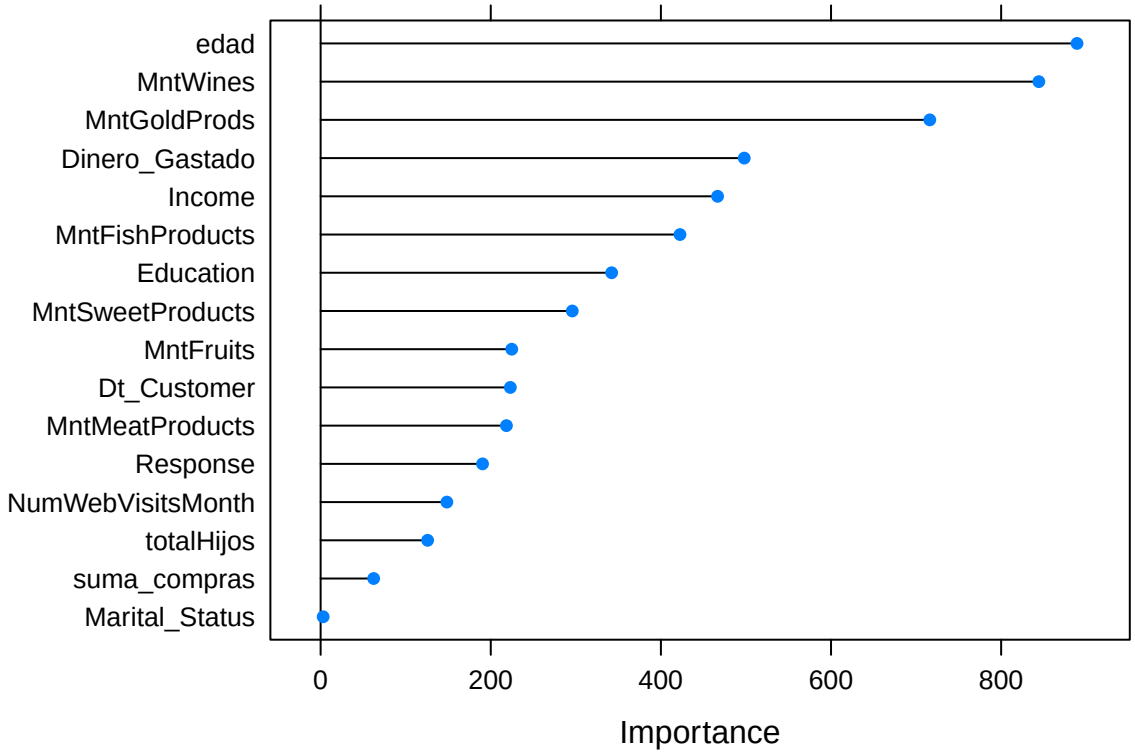


Ahora implementaremos otro modelo, que será el de arbol de decisiones. Este tiene la ventaja de que si nos dará las variables más importantes incluso si tienen relación no lineal con el output.

```
set.seed(2021)
tree2.fit <- train(x = fTR2[,c(seq(1,11),seq(13,17))],y = fTR2$Complain,method ="rpart",
            control=rpart.control(minsplit=20,minbucket = 20), parms = list(split = "gini"),
            tuneGrid = data.frame(cp = seq(0,0.1,0.001)), trControl = ctrl,metric = "Accuracy")
ggplot(tree2.fit)
```



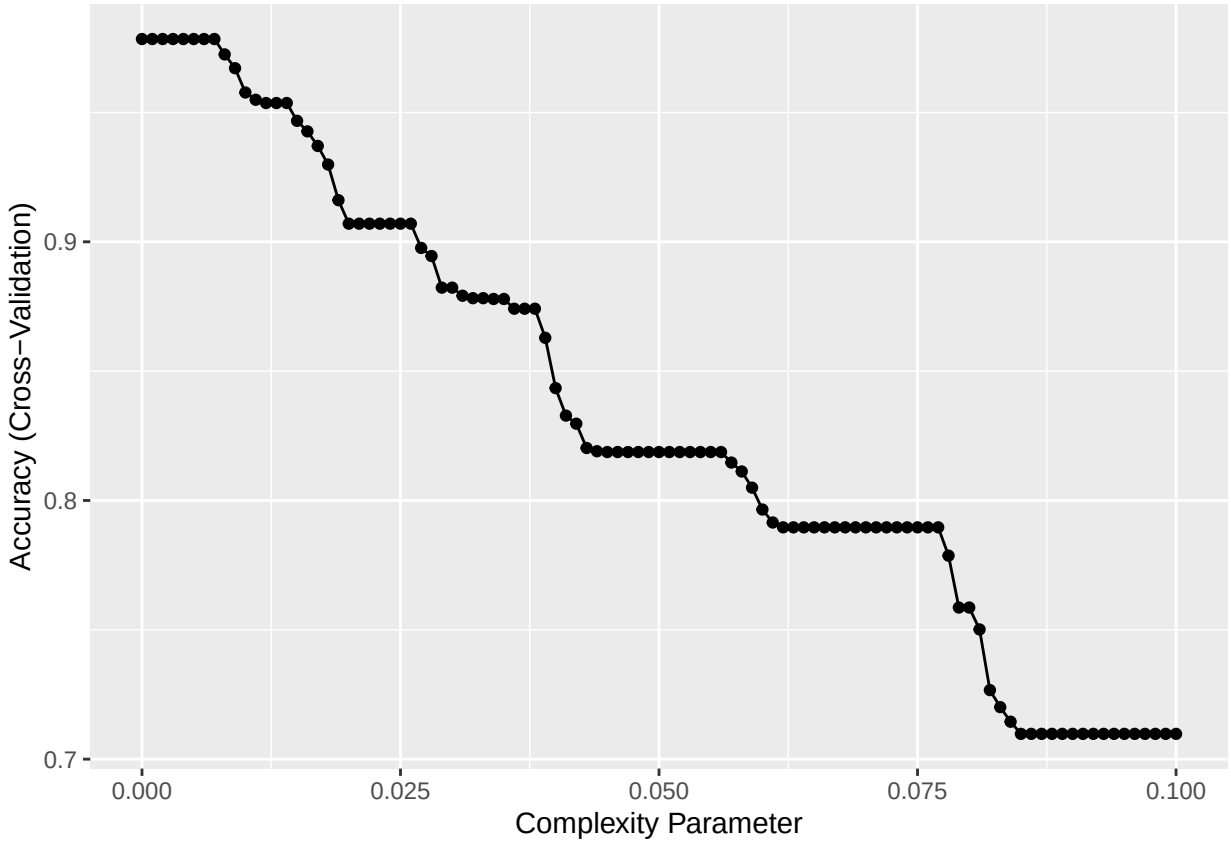

```
plot(varImp(tree2.fit,scale = FALSE))
```



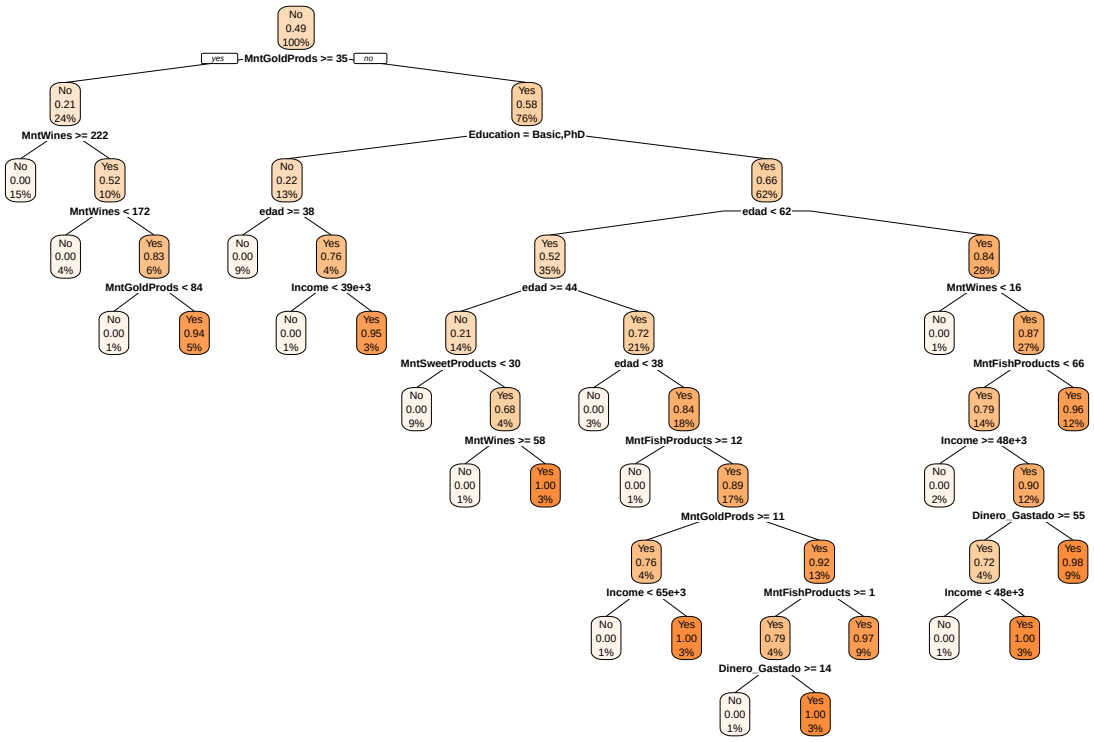
```
set.seed(2021)
fTR2_eval$tree_prob <- predict(tree2.fit, type="prob", newdata = fTR2)
fTR2_eval$tree_pred <- predict(tree2.fit, type="raw", newdata = fTR2)
fTS2_eval$tree_prob <- predict(tree2.fit, type="prob", newdata = fTS2)
fTS2_eval$tree_pred <- predict(tree2.fit, type="raw", newdata = fTS2)
```

Por último haremos un modelo solo con las variables más importantes, tanto aquellas que tienen una relación lineal como aquellas que son no lineales. Para una mayor facilidad de comprensión del modelo y viendo como ha salido el último, realizaremos un arbol de decisión.

```
set.seed(2021)
tree2_1.fit <- train(x = fTR2[,c(1,3,5,8,9,10,14,17)],y = fTR2$Complain,method ="rpart",
            control=rpart.control(minsplit=20,minbucket = 20), parms = list(split = "gini"),
            tuneGrid = data.frame(cp = seq(0,0.1,0.001)), trControl = ctrl,metric = "Accuracy")
ggplot(tree2_1.fit)
```

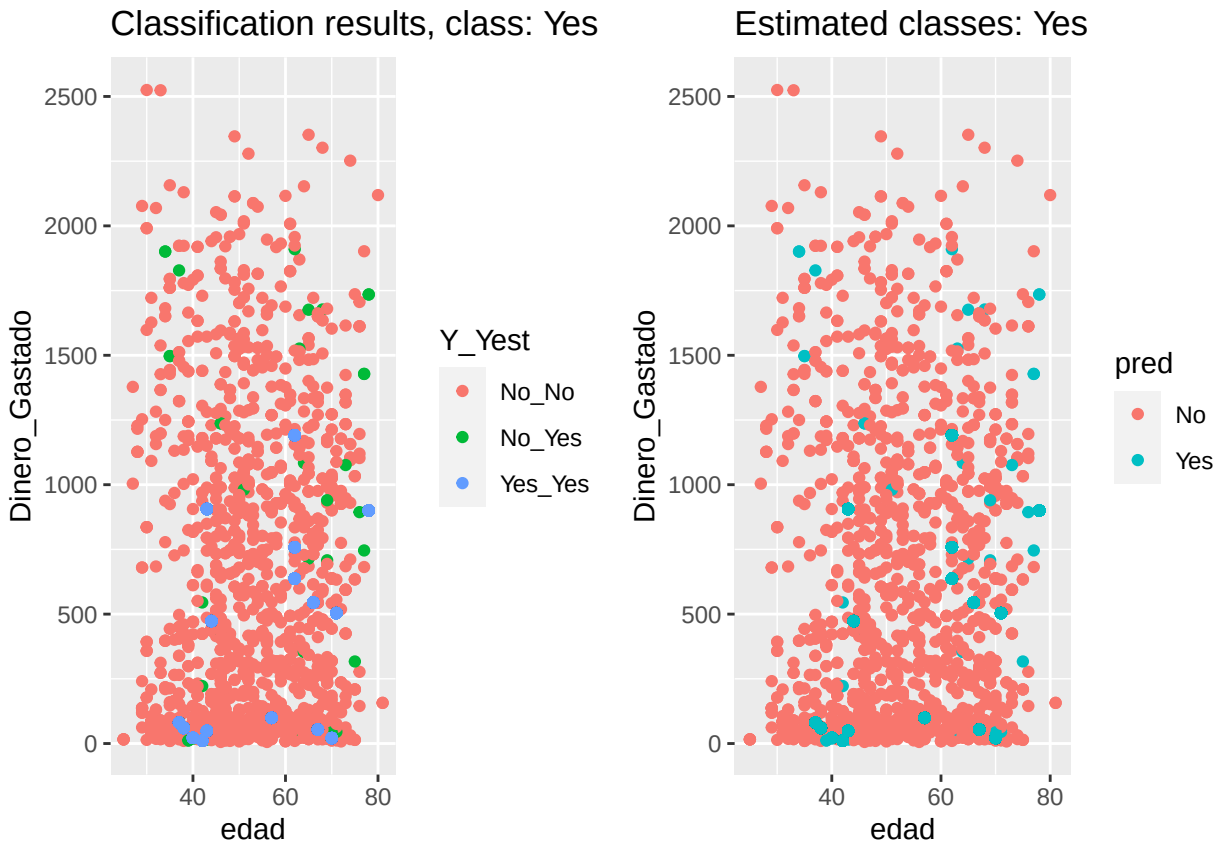



```
rpart.plot(tree2_1.fit$finalModel, type = 2, fallen.leaves = FALSE, box.palette = "Oranges")
```




```
set.seed(2021)
fTR2_eval$tree2_1_prob <- predict(tree2_1.fit, type="prob", newdata = fTR2)
fTR2_eval$tree2_1_pred <- predict(tree2_1.fit, type="raw", newdata = fTR2)
fTS2_eval$tree2_1_prob <- predict(tree2_1.fit, type="prob", newdata = fTS2)
fTS2_eval$tree2_1_pred <- predict(tree2_1.fit, type="raw", newdata = fTS2)
```

```
Plot2DClass(fTR2[,c(1,3,5,8,9,10,14,17)],fTR2$Complain,tree2_1.fit,var1 = "edad", var2 = "Dinero_Gastado
```




```
confusionMatrix(data = fTR2_eval$tree2_1_pred,reference = fTR2_eval$Complain,positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1580    0
##        Yes   42 1572
##
##                Accuracy : 0.9869
##                  95% CI : (0.9823, 0.9905)
##     No Information Rate : 0.5078
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9737
##
##  Mcnemar's Test P-Value : 2.509e-10
##
##             Sensitivity : 1.0000
##             Specificity : 0.9741
##          Pos Pred Value : 0.9740
##          Neg Pred Value : 1.0000
```

```
##              Prevalence : 0.4922
##          Detection Rate : 0.4922
##    Detection Prevalence : 0.5053
##       Balanced Accuracy : 0.9871
##
##        'Positive' Class : Yes
##
```

```
set.seed(2021)
confusionMatrix(data = fTS2_eval$tree2_1_pred,reference = fTS2_eval$Complain,positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  395   0
##        Yes  10 393
##
##                Accuracy : 0.9875
##                  95% CI : (0.9771, 0.994)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9749
##
##  Mcnemar's Test P-Value : 0.004427
##
##             Sensitivity : 1.0000
##             Specificity : 0.9753
##          Pos Pred Value : 0.9752
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4925
##          Detection Rate : 0.4925
##    Detection Prevalence : 0.5050
##       Balanced Accuracy : 0.9877
##
##        'Positive' Class : Yes
##
```