TFM_UOC

Ariel Cariaga Martínez

2024-12-28

```r
# Librerías
library(sessioninfo)
library(knitr)
library(kableExtra)
library(fastDummies)
library(xgboost)
library(tidyverse)
library(naniar)
library(corrplot)
library(caret)
library(DMwR2) #imputación knn
library(reshape2)
library(FactoMineR)
library(factoextra)
library(ggrepel)
library(MASS) # LDA
library(car)
library(multcomp)
library(pROC)
library(fastshap)
```

OBTENCIÓN DEL DATASET FINAL

Tras reunión de coordinación, seleccionaremos demográficas y clínicas relevantes.

La columna del espectro clínico de la primera infección tiene NAs, con una proporción de 10 asintomáticos, 35 hospitalizados, 185 en mild-moderated y 7 en UCI. La imputación que generaré es con el valor más común.

```r
#Seleccionamos el espectro clínico de la primera infección
column_spec <- bbddAECM$spec1
selected_spec <- data.frame(spec1 = column_spec)

#Imputación de valores faltantes en la columna 'spec1'
most_frequent <- names(sort(table(selected_spec), decreasing = TRUE))[1]

#Imputamos los valores faltantes con el más frecuente
selected_spec[is.na(selected_spec)] <- most_frequent

#Generamos dummies solo para 'spec1'
selected_spec1_dummy <- dummy_cols(selected_spec,
                                   remove_first_dummy = TRUE,
                                   remove_selected_columns = TRUE)
```

Ahora seleccionamos las demográficas de interés.

```r
#Variables sociodemográficas indicadas en la reunión de coordinación.
selected_sociodemographic <- bbddAECM %>%
  dplyr::select(sex, ag, el, ptg19)

#Creamos dummies para las variables demográficas seleccionadas.
selected_sociodemographic_dummies <- dummy_cols(
  selected_sociodemographic,
  remove_first_dummy = TRUE,
  remove_selected_columns = TRUE
)
```

Ahora seleccionamos las variables neuropsicológicas de interés.

```r
#Realizmos selección de tests neuropsicológicos de interés.
selected_neuropsychological <- bbddAECM %>%
  dplyr::select(tn46, tn52, tn36,
                tn38, tn40, tn42,
                tn22, tn44, tn14,
                tn24, tn12, tn6,
                tn30, tn34, tn8,
                tn48, tn50)
```

Y ahora que ya tenemos todas la variables de interés las combinamos en un único dataset.

```r
#Combinamos las variables seleccionadas en un solo dataset.
df_def <- bind_cols(selected_sociodemographic_dummies, selected_spec1_dummy, selected_neuropsychological
```

Resta crear la variable Target sobre la que generaremos el estudio.

```r
#Seleccionamos NoCOVID y No LongCOVID (que será un grupo) vs LongCovid Cog
column_cluster <- bbddAECM$cluster
selected_cluster <- data.frame(cluster = column_cluster)

#Creamos la nueva variable Target
selected_cluster$cluster <- ifelse(
  selected_cluster$cluster %in% c("No COVID", "No LongCOVID"), "No_LC",
  ifelse(selected_cluster$cluster == "LongCOVID Cog", "LC_Cog", NA)  # Excluir "LongCOVID NoCog"
)

#Convertimos Target a factor
selected_cluster$cluster <- factor(selected_cluster$cluster, levels = c("No_LC", "LC_Cog"))

#Convertimos a numérico restando 1 para que los niveles sean 0 y 1
selected_cluster$cluster_binary <- as.numeric(selected_cluster$cluster) - 1
```

Finalmente unimos todas las variables definitivamente seleccionadas.

```r
#Unimos la columna de los targets al dataframe final
df_def <- cbind(df_def, Target = selected_cluster$cluster_binary)

#------------------------------------------------------------------------------
##COMPROBAMOS QUE LOS INDICES COINCIDEN EN LA BBDD ORIGINAL Y EN LA RE-CREADA
#------------------------------------------------------------------------------

#Obtenemos los índices de las observaciones donde cluster es "LongCOVID NoCog"
indices <- which(bbddAECM$cluster == "LongCOVID NoCog")
```

```r
#Verificamos
print(indices)
```

```
 [1]   1  22  42  55  58  60  61  62  63  64  66  68 184 192 195 199 205 209 211
[20] 230 234 237 238 241 243 250 263
```

```r
#Obtenemos los índices de las observaciones donde Target es NA en el nuevo dataset
indices_na <- which(is.na(df_def$Target))

#Verificamos
print(indices_na)
```

```
 [1]   1  22  42  55  58  60  61  62  63  64  66  68 184 192 195 199 205 209 211
[20] 230 234 237 238 241 243 250 263
```

```r
#ELIMINAMOS TODAS LAS OBSERVACIONES NAS EN TARGET
df_def <- df_def[!is.na(df_def$Target), ]

#Dado que los índices coinciden, df_def será el dataset que utilizaremos para realizar el TFM

#Aquí remodificamos los nombres para evitar problemas
df_def <- df_def %>% rename_with(make.names)
```

```r
# ----------------------
# Análisis general del dataset de trabajo final generado
# ----------------------

#Vemos tipos de variables y estructura
str(df_def)
```

```
'data.frame':   241 obs. of  29 variables:
 $ ag                    : num  26.9 60.5 33.1 44.6 61.9 43 55.6 53.6 46.3 51.2 ...
 $ ptg19                 : num  38.2 26.3 18.2 24.1 32.8 ...
 $ sex_Woman             : int  1 0 1 1 0 1 1 1 1 1 ...
 $ el_Elementary         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ el_High.School        : int  1 0 0 0 0 0 0 0 0 0 ...
 $ el_Secondary          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ el_Specialist.Master  : int  0 1 0 0 1 1 1 0 1 0 ...
 $ el_University.Deg.    : int  0 0 0 1 0 0 0 1 0 1 ...
 $ spec1_Hospitalization: int  0 1 0 0 1 0 0 0 0 1 ...
 $ spec1_Mild.Moderated : int  1 0 1 1 0 1 1 1 1 0 ...
 $ spec1_UCI             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ tn46                  : num  0.3 1 0.7 0 0 0.3 -0.3 -0.3 -0.7 -0.3 ...
 $ tn52                  : num  1 0 2 -0.6 0 ...
 $ tn36                  : num  -1 0 1 0.3 0.3 -0.7 -0.3 -0.3 -1.7 0 ...
 $ tn38                  : num  -0.3 0.3 0.3 0 -0.3 -0.3 -0.7 0 -2.3 -0.3 ...
 $ tn40                  : num  0 0 1 1 0.3 0.3 -0.3 -0.3 -0.3 -0.3 ...
 $ tn42                  : num  -1.3 0 0.3 1.3 -0.3 -0.3 -1 -1.7 -0.7 0.3 ...
 $ tn22                  : num  0.4 0.4 -0.5 -0.9 -0.2 1 0.4 0.3 0.3 0.1 ...
 $ tn44                  : num  -0.7 -1.3 -1 1 -0.3 0 -0.7 1 -2 -0.7 ...
 $ tn14                  : num  -1.7 1 0.3 -0.3 1.3 0.7 0.3 0.7 -0.3 0.7 ...
 $ tn24                  : num  -3 0 -1 -0.7 -0.3 -0.7 0.4 -0.7 -2.7 -0.7 ...
 $ tn12                  : num  -2 0 1.3 -0.3 1.3 0 1 0 0.7 0 ...
 $ tn6                   : num  -2.06 0.86 1.7 -0.82 -0.57 1.38 -1.18 -0.2 -0.01 -0.32 ...
 $ tn30                  : num  -2.3 0.7 0.32 -0.43 0.4 1.71 -1.22 0.03 -0.07 0.03 ...
 $ tn34                  : num  -1 0.3 -1 -1 0.3 0 -0.3 -1 1 -2 ...
```

```
 $ tn8                      : num  2 -0.3 0.3 -0.7 -0.3 2.7 0 -0.3 -0.3 0.3 ...
 $ tn48                     : num  1 3.1 3.1 1.3 3.1 1.3 1.3 3.1 1 0.8 ...
 $ tn50                     : num  -1.7 1.3 -0.3 -0.3 -0.7 0.7 1.3 0.7 0.3 0.7 ...
 $ Target                   : num  1 1 0 1 1 1 1 1 1 1 ...
```

```r
#Resumen estadístico
summary(df_def)
```

```
       ag            ptg19         sex_Woman      el_Elementary    
 Min.   :25.40   Min.   :16.05   Min.   :0.0000   Min.   :0.00000  
 1st Qu.:42.40   1st Qu.:23.01   1st Qu.:1.0000   1st Qu.:0.00000  
 Median :48.80   Median :26.19   Median :1.0000   Median :0.00000  
 Mean   :48.72   Mean   :27.69   Mean   :0.8008   Mean   :0.04149  
 3rd Qu.:55.40   3rd Qu.:30.60   3rd Qu.:1.0000   3rd Qu.:0.00000  
 Max.   :70.80   Max.   :87.31   Max.   :1.0000   Max.   :1.00000  
                 NA's   :6                                          
 el_High.School    el_Secondary     el_Specialist.Master el_University.Deg.
 Min.   :0.0000   Min.   :0.00000   Min.   :0.0000       Min.   :0.0000    
 1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000       1st Qu.:0.0000    
 Median :0.0000   Median :0.00000   Median :0.0000       Median :0.0000    
 Mean   :0.3278   Mean   :0.03734   Mean   :0.1494       Mean   :0.3776    
 3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.0000       3rd Qu.:1.0000    
 Max.   :1.0000   Max.   :1.00000   Max.   :1.0000       Max.   :1.0000    
                                                                           
 spec1_Hospitalization spec1_Mild.Moderated   spec1_UCI           tn46        
 Min.   :0.0000        Min.   :0.0000       Min.   :0.0000   Min.   :-2.7000  
 1st Qu.:0.0000        1st Qu.:1.0000       1st Qu.:0.0000   1st Qu.:-0.7000  
 Median :0.0000        Median :1.0000       Median :0.0000   Median : 0.0000  
 Mean   :0.1411        Mean   :0.7967       Mean   :0.0249   Mean   :-0.1021  
 3rd Qu.:0.0000        3rd Qu.:1.0000       3rd Qu.:0.0000   3rd Qu.: 0.3000  
 Max.   :1.0000        Max.   :1.0000       Max.   :1.0000   Max.   : 2.7000  
                                                             NA's   :3        
      tn52               tn36              tn38              tn40        
 Min.   :-2.30000   Min.   :-3.0000   Min.   :-2.3000   Min.   :-3.0000  
 1st Qu.:-0.40000   1st Qu.:-0.7000   1st Qu.:-0.7000   1st Qu.:-0.9250  
 Median : 0.00000   Median :-0.3000   Median :-0.3000   Median :-0.3000  
 Mean   : 0.02143   Mean   :-0.2453   Mean   :-0.1378   Mean   :-0.2483  
 3rd Qu.: 0.40000   3rd Qu.: 0.3000   3rd Qu.: 0.3000   3rd Qu.: 0.3000  
 Max.   : 2.00000   Max.   : 2.3000   Max.   : 2.7000   Max.   : 2.7000  
 NA's   :3          NA's   :3         NA's   :3         NA's   :3        
      tn42              tn22              tn44             tn14        
 Min.   :-3.1000   Min.   :-2.3000   Min.   :-3.000   Min.   :-2.3000  
 1st Qu.:-1.3000   1st Qu.:-0.1000   1st Qu.:-0.700   1st Qu.:-0.3000  
 Median :-0.7000   Median : 0.5000   Median :-0.300   Median : 0.3000  
 Mean   :-0.7286   Mean   : 0.3982   Mean   :-0.363   Mean   : 0.4055  
 3rd Qu.: 0.0000   3rd Qu.: 0.9000   3rd Qu.: 0.225   3rd Qu.: 1.0000  
 Max.   : 2.3000   Max.   : 3.0000   Max.   : 2.300   Max.   : 2.7000  
 NA's   :3         NA's   :4         NA's   :3        NA's   :3        
      tn24              tn12              tn6               tn30         
 Min.   :-3.0000   Min.   :-2.0000   Min.   :-5.9400   Min.   :-2.53000  
 1st Qu.:-1.0000   1st Qu.:-0.3000   1st Qu.:-1.0500   1st Qu.:-0.78000  
 Median :-0.3000   Median : 0.1500   Median :-0.1900   Median : 0.03000  
 Mean   :-0.4887   Mean   : 0.2479   Mean   :-0.1951   Mean   : 0.03378  
 3rd Qu.: 0.0000   3rd Qu.: 1.0000   3rd Qu.: 0.8575   3rd Qu.: 0.97000  
 Max.   : 2.3000   Max.   : 2.3000   Max.   : 2.2700   Max.   : 1.73000  
```

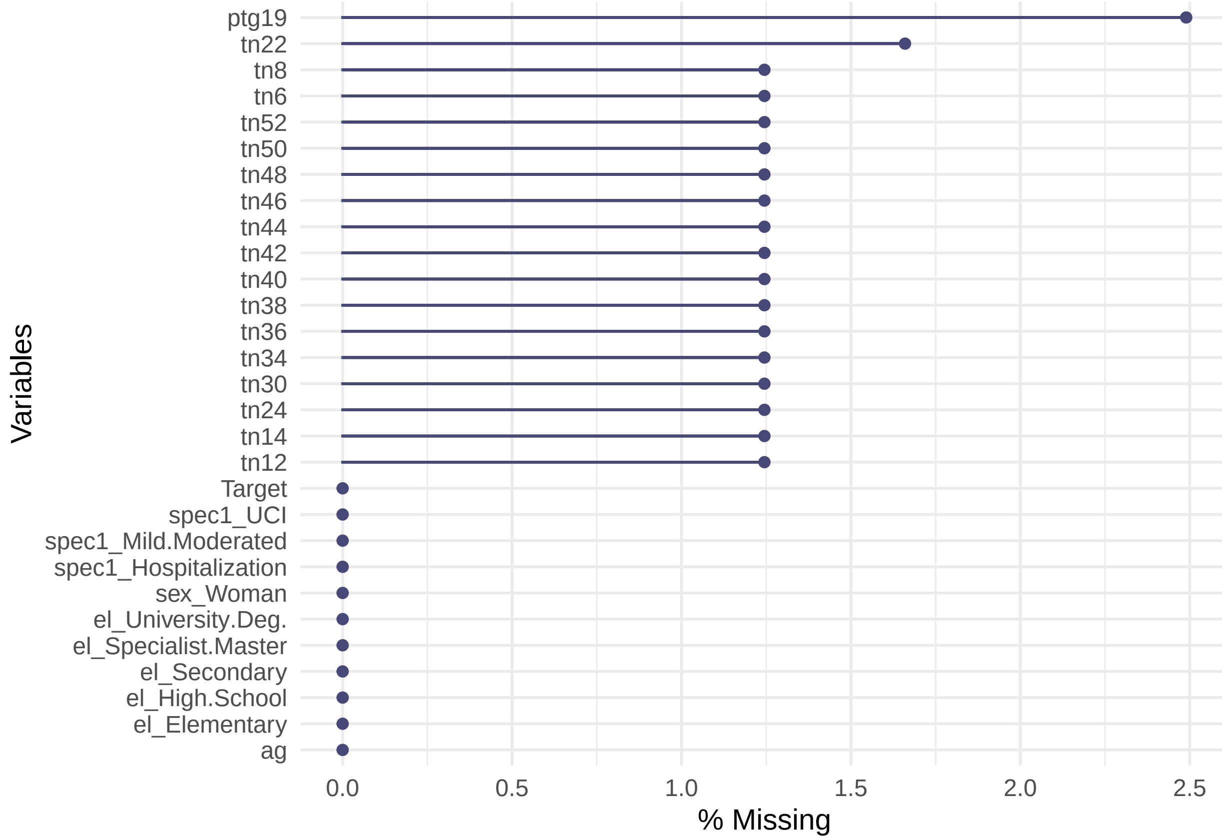
```
 NA's   :3          NA's   :3          NA's   :3          NA's   :3
      tn34               tn8                tn48             tn50
 Min.   :-2.7000   Min.   :-2.0000   Min.   :0.300   Min.   :-2.0000
 1st Qu.:-0.7000   1st Qu.:-0.7000   1st Qu.:1.000   1st Qu.: 0.0000
 Median :-0.3000   Median :-0.3000   Median :1.300   Median : 0.3000
 Mean   :-0.3571   Mean   :-0.1202   Mean   :1.615   Mean   : 0.2294
 3rd Qu.: 0.0000   3rd Qu.: 0.3000   3rd Qu.:3.000   3rd Qu.: 0.7000
 Max.   : 2.7000   Max.   : 2.7000   Max.   :3.100   Max.   : 3.0000
 NA's   :3         NA's   :3         NA's   :3       NA's   :3
     Target
 Min.   :0.000
 1st Qu.:0.000
 Median :1.000
 Mean   :0.722
 3rd Qu.:1.000
 Max.   :1.000
```

```r
# if (!requireNamespace("webshot", quietly = TRUE)) {
#   install.packages("webshot")
#   webshot::install_phantomjs()
# }
# library(webshot)
#
# # Crear un data frame a partir del summary
# summary_table <- as.data.frame(summary(df_def))
#
# save_kable(
#   kable(summary_table, format = "html", caption = "Resumen de df_def") %>%
#     kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover", "condensed")),
#   file = "tabla_resumen.html"
# )
#
# # Convertir el archivo HTML a JPG
# webshot("tabla_resumen.html", file = "tabla_resumen.jpg", vwidth = 800, vheight = 600)

#Ver las dimensiones y primeras filas
cat("Dimensiones del dataset:", dim(df_def), "\n")
```

```
Dimensiones del dataset: 241 29
```

Vamos a verificar si hay algún valor faltante

```r
# ----------------------
# Valores faltantes
# ----------------------

#Resumen de valores faltantes
gg_miss_var(df_def, show_pct = TRUE)
```



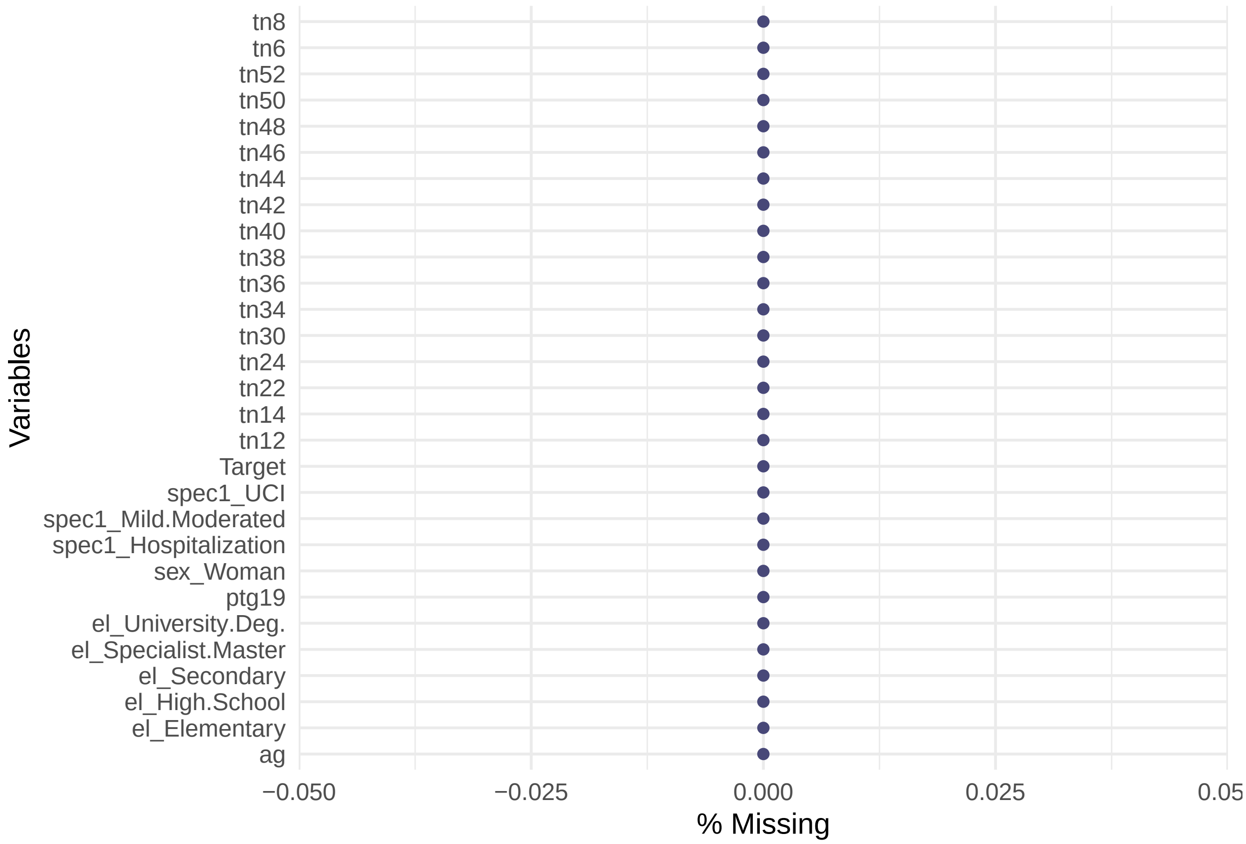

```
#-----------------------------------------------------------------------------
# IMAGEN 1
#-----------------------------------------------------------------------------
pdf("outputs/images/01_NAs.pdf", width = 16, height = 10)
gg_miss_var(df_def, show_pct = TRUE)
dev.off()
```

```
pdf
  2
```

Como no tenemos una idea global del tipo de valores faltantes, vamos a hacer una imputación con kNN para estos.

```
# ----------------------
# Imputación de valores faltantes
# ----------------------

#Excluimos columnas específicas
excluded_cols <- c("Target")

#Separamos las columnas excluidas
excluded_data <- df_def %>%
  dplyr::select(all_of(excluded_cols))

#Convertimos todas las columnas numéricas a tipo double
data_for_imputation_selected <- df_def %>% dplyr::select(-all_of(excluded_cols))
data_for_imputation_selected <- data_for_imputation_selected %>%
  mutate(across(where(is.numeric), as.numeric))
```

```r
#Aplicamos la imputación
selected_data_imputed <- knnImputation(data_for_imputation_selected, k = 10)

#Combinamos las columnas imputadas con las excluidas
df_def_clean <- bind_cols(selected_data_imputed, excluded_data)

#Gráfico tras imputación
gg_miss_var(df_def_clean, show_pct = TRUE)
```




```r
#---------------------------------------------------------------------------------
# IMAGEN 2
#---------------------------------------------------------------------------------
pdf("outputs/images/02_NAs_tras_imputacion.pdf", width = 16, height = 10)
gg_miss_var(df_def_clean, show_pct = TRUE)
dev.off()
```

```
pdf
  2
```

Vamos a dejar el código del balanceo del dataset y guardamos el dataset balanceado por si fuera de interés para utilizarlo posteriormente.

```r
# ----------------------
# Balancear el dataset (no se utiliza el dataset balanceado pero se deja para constancia futura)
# ----------------------

table(df_def_clean$Target) #Hay 64 vs 174
```

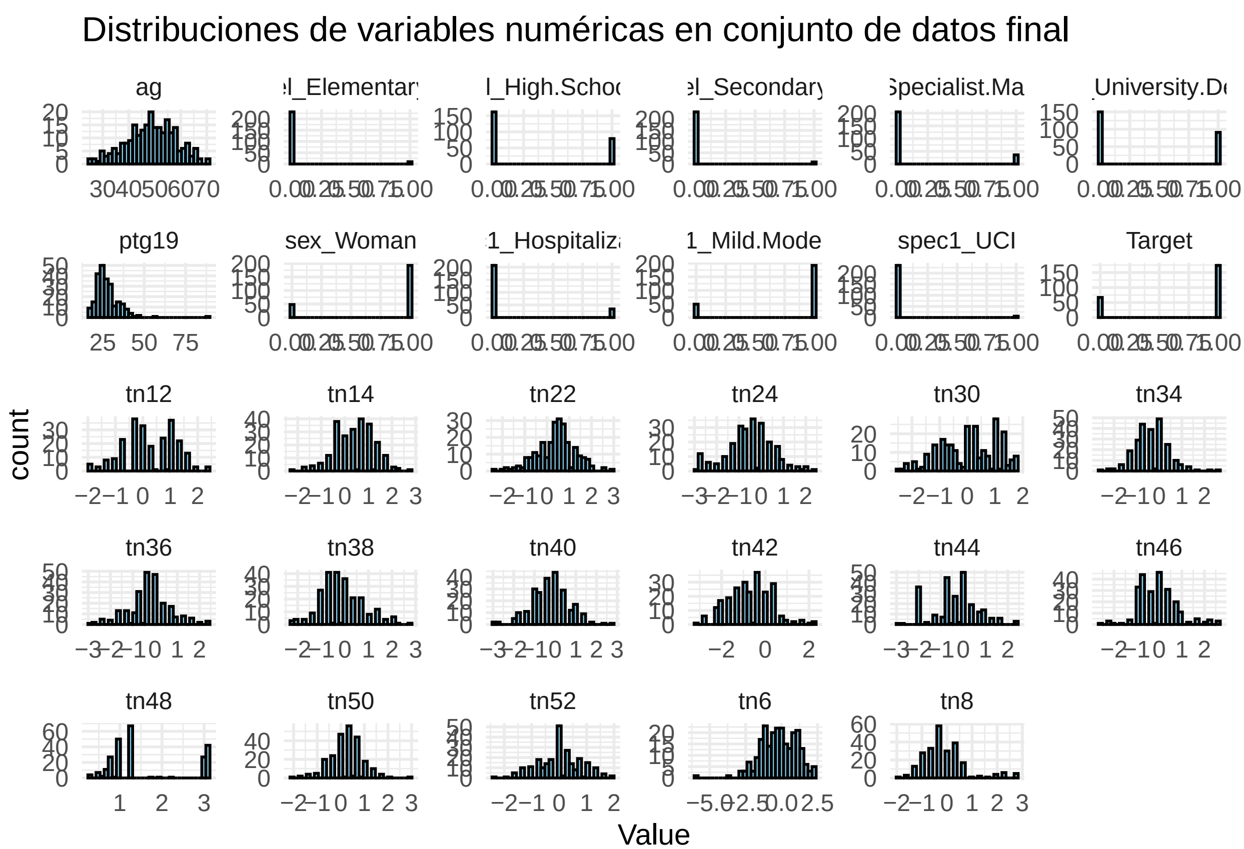
```
  0   1
 67 174
```

```
#Si aplicmos ROSE: OJO: DATOS SINTÉTICOS
#df_def_clean_balanced_ROSE <- ROSE(Target ~ ., data = df_combined_clean, seed = 123, method = "under")

#Oversampling manual
minority_class <- df_def_clean %>% filter(Target == "0")
oversampled <- minority_class %>% sample_n(size = nrow(df_def_clean[df_def_clean$Target == "1", ]), repl

#Combinamos con la clase mayoritaria
df_def_clean_balanced <- bind_rows(
  df_def_clean %>% filter(Target == "1"),
  oversampled
)

table(df_def_clean_balanced$Target)
```

```
  0   1
174 174
```

Análisis univariado de variables numéricas

```
# ----------------------
# Análisis univariado
# ----------------------

# a) Seleccionar variables numéricas
numeric_vars_def <- df_def_clean %>% dplyr::select(where(is.numeric))

# Histogramas para variables numéricas
numeric_vars_def %>%
  gather(key = "Variable", value = "Value") %>%
  ggplot(aes(x = Value)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  facet_wrap(~ Variable, scales = "free") +
  theme_minimal() +
  labs(title = "Distribuciones de variables numéricas en conjunto de datos final")
```



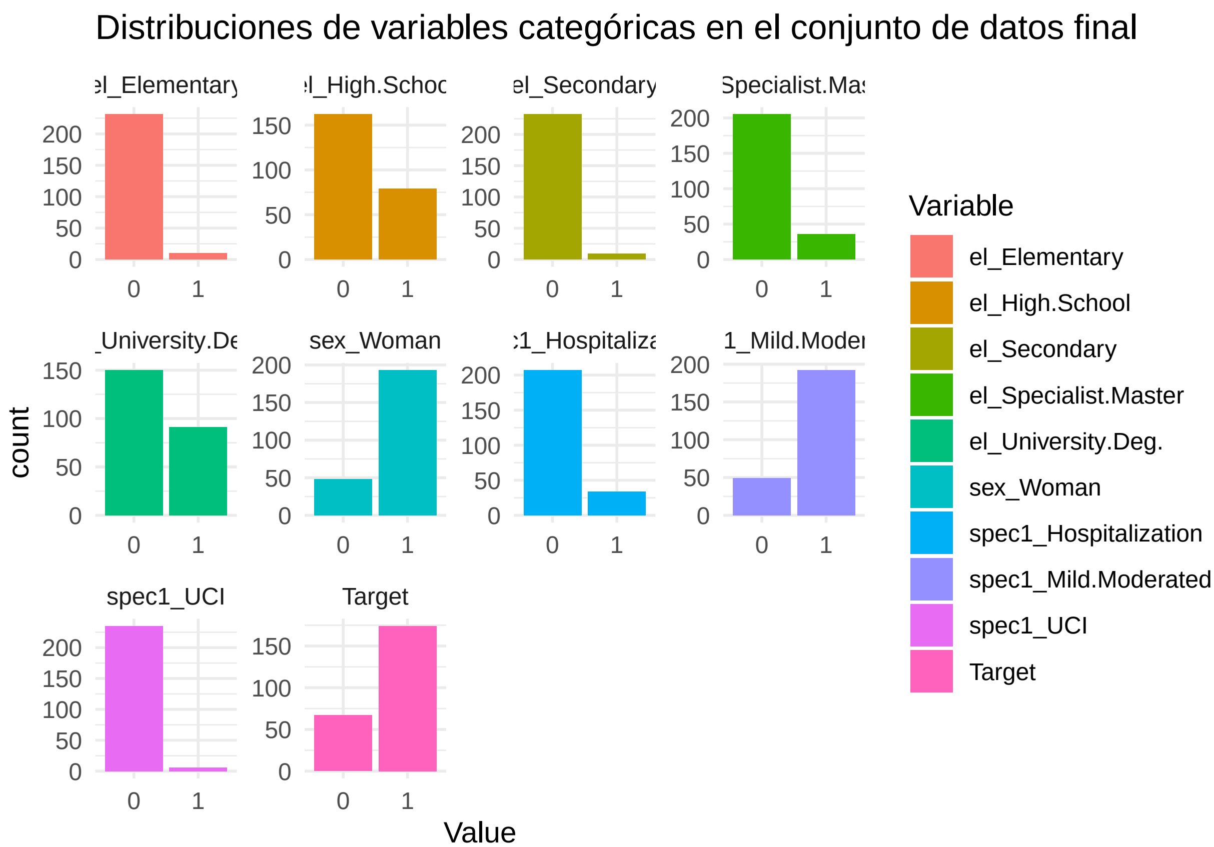

```
#-----------------------------------------------------------------------------
# IMAGEN 3
#-----------------------------------------------------------------------------
pdf("outputs/images/03_univariado_num.pdf", width = 16, height = 10)
# Histogramas para variables numéricas
numeric_vars_def %>%
  gather(key = "Variable", value = "Value") %>%
  ggplot(aes(x = Value)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  facet_wrap(~ Variable, scales = "free") +
  theme_minimal() +
  labs(title = "Distribuciones de variables numéricas en conjunto de datos final")
dev.off()
```

```
pdf
  2
```

Análisis univariado de variables categóricas

```
# b) Variables categóricas
# Seleccionar columnas específicas y transformarlas a factor: mantenemos el nombre cambiado
# para evitar afectar el dataset generado con columnas numéricas que usaremos en los modelos
df_def_clean_factors <- df_def_clean %>%
  mutate(across(
    where(~ is.numeric(.) && all(. %in% c(0, 1)) && n_distinct(.) == 2),
    as.factor
  ))
```

```r
# b.1) Selección de variables categóricas
categorical_vars_def <- df_def_clean_factors %>% dplyr::select(where(is.factor))

# Gráficos de barras para variables categóricas
categorical_vars_def %>%
  gather(key = "Variable", value = "Value") %>%
  ggplot(aes(x = Value, fill = Variable)) +
  geom_bar() +
  facet_wrap(~ Variable, scales = "free") +
  theme_minimal() +
  labs(title = "Distribuciones de variables categóricas en el conjunto de datos final")
```



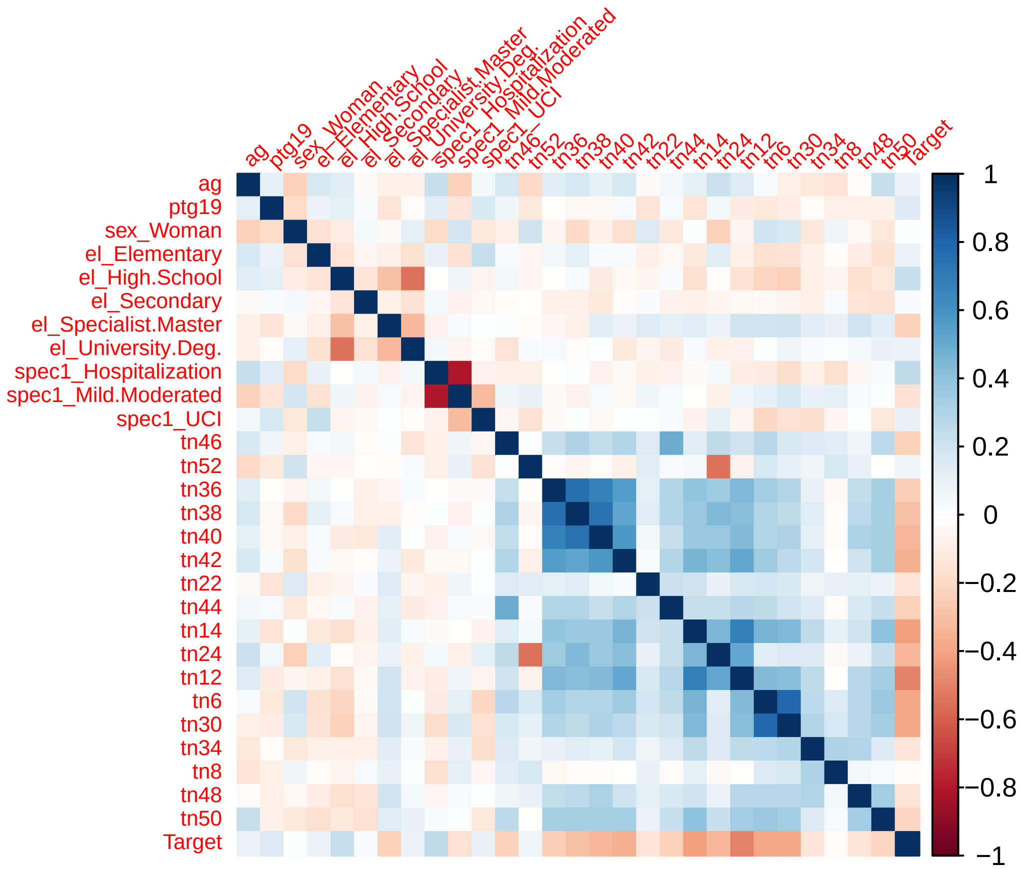

```r
#---------------------------------------------------------------------------------
# IMAGEN 4
#---------------------------------------------------------------------------------
pdf("outputs/images/04_categóricas.pdf", width = 16, height = 10)
#Gráficos de barras para variables categóricas
categorical_vars_def %>%
  gather(key = "Variable", value = "Value") %>%
  ggplot(aes(x = Value, fill = Variable)) +
  geom_bar() +
  facet_wrap(~ Variable, scales = "free") +
  theme_minimal() +
  labs(title = "Distribuciones de variables categóricas en el conjunto de datos final")
dev.off()
```

```
pdf
```

2

Correlación y relaciones multivariadas

```r
# -----------------------
# Correlación y relaciones multivariadas
# -----------------------

# a) Correlación entre variables numéricas
cor_matrix_def <- cor(numeric_vars_def, use = "pairwise.complete.obs")
corrplot(cor_matrix_def, method = "color", type = "full", tl.cex = 0.7, tl.srt = 45)
```




```r
#Filtramos correlaciones mayores o iguales a 0.6 (sin incluir la diagonal)
threshold_def <- 0.6
high_correlations_def <- which(abs(cor_matrix_def) >= threshold_def & abs(cor_matrix_def) < 1, arr.ind =

#Creamos un dataframe con las variables altamente correlacionadas: enviar a coordinación
cor_df_def <- data.frame(
  Var1 = rownames(cor_matrix_def)[high_correlations_def[, 1]],
  Var2 = colnames(cor_matrix_def)[high_correlations_def[, 2]],
  Correlation = cor_matrix_def[high_correlations_def]
)

#Eliminamos duplicados (porque la matriz de correlación es simétrica) y creamos csv
cor_df_def <- cor_df_def[!duplicated(t(apply(cor_df_def, 1, sort))), ]
write.csv(cor_df_def, "./outputs/correlaciones.csv")

#Mostramos las correlaciones
```

```r
print(cor_df_def)
```

```
                   Var1                    Var2 Correlation
1        el_High.School          el_High.School   1.0000000
2          el_Secondary            el_Secondary   1.0000000
3  spec1_Mild.Moderated spec1_Hospitalization  -0.8022447
5                  tn52                    tn52   1.0000000
6                  tn38                    tn36   0.7554991
7                  tn40                    tn36   0.6749011
9                  tn38                    tn38   1.0000000
10                 tn40                    tn38   0.7465677
13                 tn12                    tn14   0.6849510
15                  tn6                     tn6   1.0000000
16                 tn30                     tn6   0.7811306
18                 tn30                    tn30   1.0000000
19                 tn34                    tn34   1.0000000
20                 tn50                    tn50   1.0000000
```

```r
#-------------------------------------------------------------------------------
# IMAGEN 5
#-------------------------------------------------------------------------------

pdf("outputs/images/05_correlacion.pdf", width = 16, height = 10)

# a) Correlación entre variables numéricas
cor_matrix_def <- cor(numeric_vars_def, use = "pairwise.complete.obs")
corrplot(cor_matrix_def, method = "color", type = "full", tl.cex = 0.7, tl.srt = 45)

#Filtramos correlaciones mayores o iguales a 0.6 (sin incluir la diagonal)
threshold_def <- 0.6
high_correlations_def <- which(abs(cor_matrix_def) >= threshold_def & abs(cor_matrix_def) < 1, arr.ind =

#Creamos un dataframe con las variables altamente correlacionadas: enviar a coordinación
cor_df_def <- data.frame(
  Var1 = rownames(cor_matrix_def)[high_correlations_def[, 1]],
  Var2 = colnames(cor_matrix_def)[high_correlations_def[, 2]],
  Correlation = cor_matrix_def[high_correlations_def]
)

#Eliminamos duplicados (porque la matriz de correlación es simétrica) y creamos csv
cor_df_def <- cor_df_def[!duplicated(t(apply(cor_df_def, 1, sort))), ]
#write.csv(cor_df_def, "./outputs/correlaciones.csv")

#Mostramos las correlaciones
print(cor_df_def)
```

```
                   Var1                    Var2 Correlation
1        el_High.School          el_High.School   1.0000000
2          el_Secondary            el_Secondary   1.0000000
3  spec1_Mild.Moderated spec1_Hospitalization  -0.8022447
5                  tn52                    tn52   1.0000000
6                  tn38                    tn36   0.7554991
7                  tn40                    tn36   0.6749011
9                  tn38                    tn38   1.0000000
```

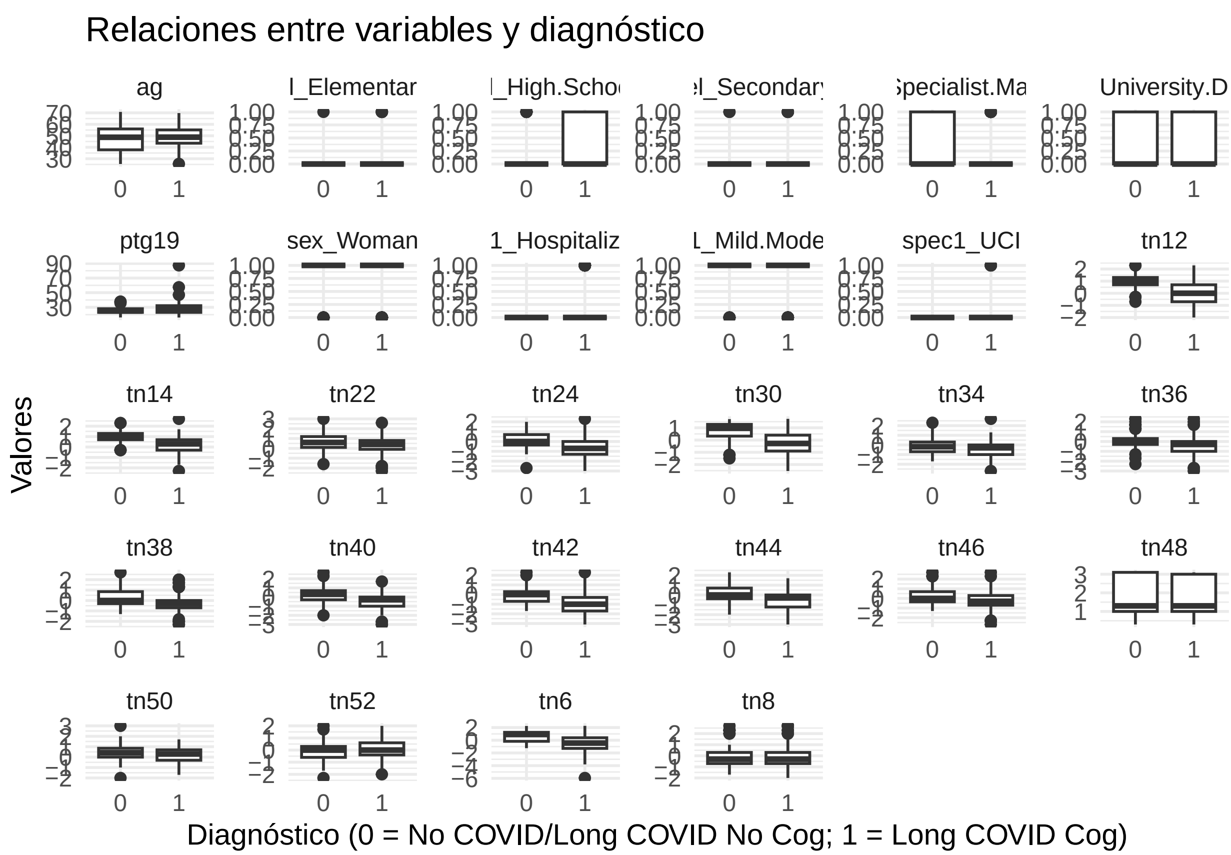
```
10                  tn40                 tn38   0.7465677
13                  tn12                 tn14   0.6849510
15                   tn6                  tn6   1.0000000
16                  tn30                  tn6   0.7811306
18                  tn30                 tn30   1.0000000
19                  tn34                 tn34   1.0000000
20                  tn50                 tn50   1.0000000
```

```r
dev.off()
```

```
pdf
  2
```

Relaciones numéricas-categóricas

```r
# ----------------------
# b) Relaciones numéricas-categóricas
# ----------------------

#Ajustando "Target" como variable objetivo
if ("Target" %in% colnames(df_def_clean)) {
  df_def_clean %>%
    mutate(Target = as.factor(Target)) %>% #Considerando que Target es binaria
    gather(key = "Variable", value = "Value", -Target) %>%
    filter(!is.na(Value)) %>%
    ggplot(aes(x = Target, y = Value)) +
    geom_boxplot() +
    facet_wrap(~ Variable, scales = "free") +
    theme_minimal() +
    labs(title = "Relaciones entre variables y diagnóstico",
         x = "Diagnóstico (0 = No COVID/Long COVID No Cog; 1 = Long COVID Cog)",
         y = "Valores",
         fill = "Diagnóstico") # Título de la leyenda)
}
```




```r
#-------------------------------------------------------------------------------
# IMAGEN 6
#-------------------------------------------------------------------------------
pdf("outputs/images/06_relaciones.pdf", width = 16, height = 10)

# b) Relaciones numéricas-categóricas
# Ajustando "Target" como variable objetivo
if ("Target" %in% colnames(df_def_clean)) {
  df_def_clean %>%
    mutate(Target = as.factor(Target)) %>% #Considerando que Target es binaria
    gather(key = "Variable", value = "Value", -Target) %>%
    filter(!is.na(Value)) %>%
    ggplot(aes(x = Target, y = Value)) +
    geom_boxplot() +
    facet_wrap(~ Variable, scales = "free") +
    theme_minimal() +
    labs(title = "Relaciones entre variables y Target",
         x = "Diagnóstico (0 = No COVID/Long COVID No Cog; 1 = Long COVID Cog)",
         y = "Valores",
         fill = "Diagnóstico") # Título de la leyenda
}
dev.off()
```

```
pdf
  2
```

Reducción de la dimensionalidad

```r
#Realmente no vamos a realizar una eliminación de variables "autómatica" pero la función del paquete pu

# a) Eliminar variables con baja variabilidad: esto es "automático" con el paquete caret
nzv <- nearZeroVar(df_def_clean, saveMetrics = TRUE)
low_variability_vars <- rownames(nzv[nzv$nzv == TRUE, ])
cat("Variables con baja variabilidad eliminadas:", low_variability_vars, "\n")
```

```
Variables con baja variabilidad eliminadas: el_Elementary el_Secondary spec1_UCI
```

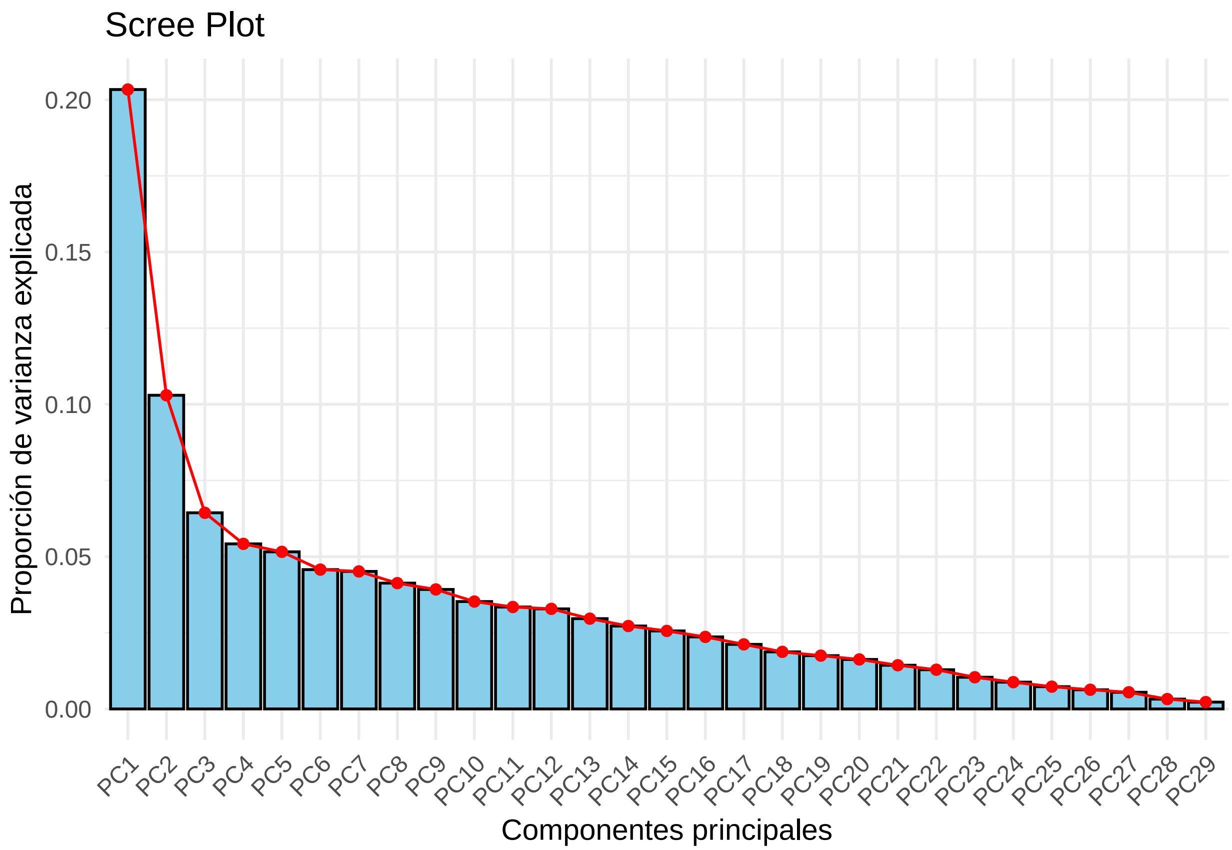
```r
df_def_clean_low_var <- df_def_clean %>% dplyr::select(-all_of(low_variability_vars))
```

```r
# b) Análisis de componentes principales (PCA)

pca_data_scaled <- scale(df_def_clean)
pca <- prcomp(pca_data_scaled, center = TRUE)
summary(pca)  #Verificar proporción de la varianza explicada
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     2.4283 1.7280 1.36641 1.25357 1.22288 1.15161 1.14405
Proportion of Variance 0.2033 0.1030 0.06438 0.05419 0.05157 0.04573 0.04513
Cumulative Proportion  0.2033 0.3063 0.37067 0.42486 0.47643 0.52216 0.56729
                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
Standard deviation     1.09425 1.06641 1.01097 0.98501 0.97596 0.92702 0.88838
Proportion of Variance 0.04129 0.03922 0.03524 0.03346 0.03284 0.02963 0.02721
Cumulative Proportion  0.60858 0.64780 0.68304 0.71650 0.74934 0.77898 0.80619
                         PC15    PC16    PC17    PC18    PC19    PC20    PC21
Standard deviation     0.8617 0.82799 0.78399 0.73707 0.71217 0.68631 0.64483
Proportion of Variance 0.0256 0.02364 0.02119 0.01873 0.01749 0.01624 0.01434
Cumulative Proportion  0.8318 0.85543 0.87663 0.89536 0.91285 0.92909 0.94343
                          PC22   PC23    PC24    PC25    PC26    PC27    PC28
Standard deviation     0.61088 0.5491 0.50475 0.46039 0.42703 0.39811 0.30577
Proportion of Variance 0.01287 0.0104 0.00879 0.00731 0.00629 0.00547 0.00322
Cumulative Proportion  0.95630 0.9667 0.97548 0.98279 0.98908 0.99454 0.99776
                          PC29
Standard deviation     0.25462
Proportion of Variance 0.00224
Cumulative Proportion  1.00000
```

```r
#Extraemos la proporción de la varianza explicada
pca_var <- pca$sdev^2  # Varianza explicada por cada componente
pca_var_exp <- pca_var / sum(pca_var)  # Proporción de varianza explicada

#Creamos un dataframe para el plot (sin reordenar manualmente)
scree_data <- data.frame(
  PC = factor(paste0("PC", seq_along(pca_var_exp)),
              levels = paste0("PC", seq_along(pca_var_exp))),  #Forzar el orden
  Variance_Explained = pca_var_exp
)

#Scree Plot
library(ggplot2)
ggplot(scree_data, aes(x = PC, y = Variance_Explained)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  geom_line(aes(group = 1), color = "red") + #Línea conectando las barras
```

```r
  geom_point(color = "red") + #Puntos en la línea
  theme_minimal() +
  labs(
    title = "Scree Plot",
    x = "Componentes principales",
    y = "Proporción de varianza explicada"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

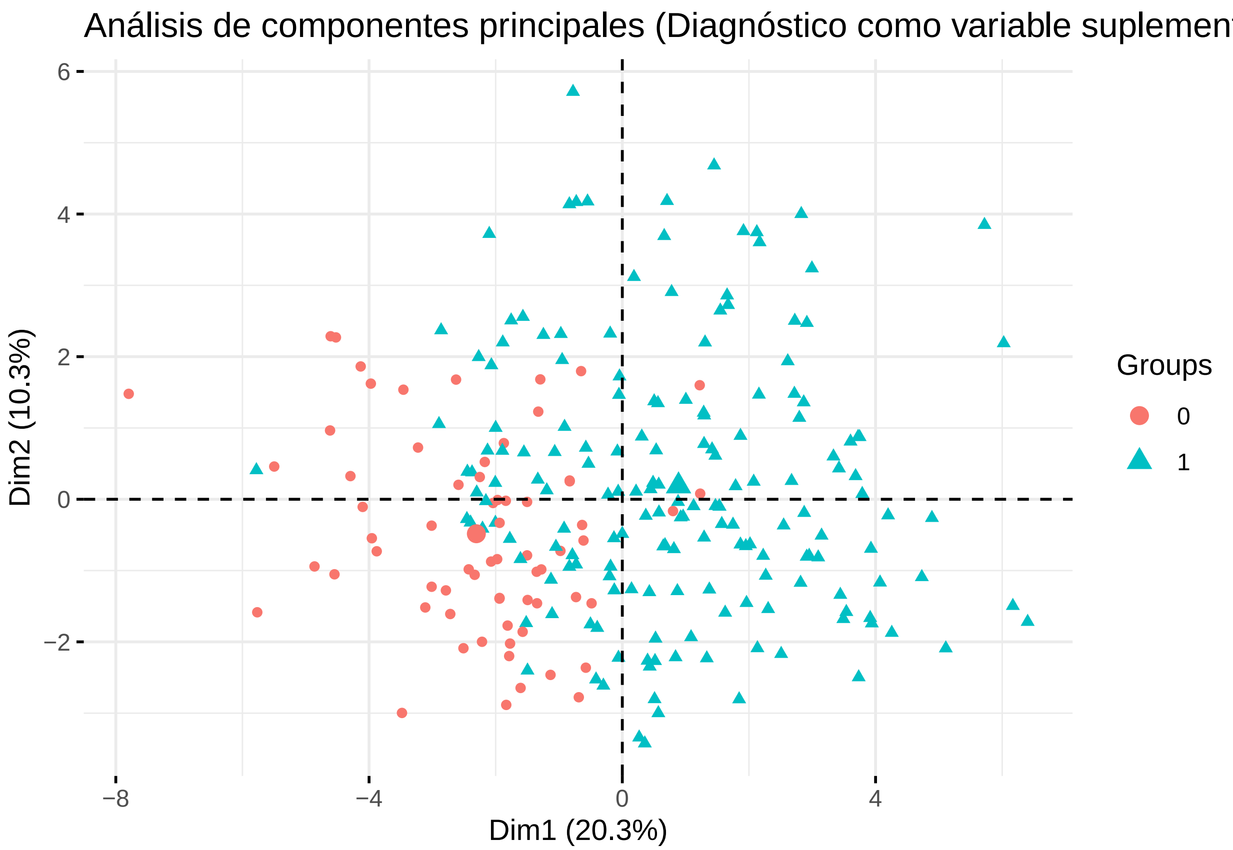



```r
#-------------------------------------------------------------------------------
# IMAGEN 7
#-------------------------------------------------------------------------------
pdf("outputs/images/07_screeplot.pdf", width = 16, height = 10)
# Scree Plot
library(ggplot2)
ggplot(scree_data, aes(x = PC, y = Variance_Explained)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  geom_line(aes(group = 1), color = "red") + # Línea conectando las barras
  geom_point(color = "red") + # Puntos en la línea
  theme_minimal() +
  labs(
    title = "Scree Plot",
    x = "Componentes principales",
    y = "Proporción de varianza explicada"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
dev.off()
```

```
pdf
  2
```

Necesitamos unas 13 componentes para explicar el 80% de la varianza. Las 2 primeras solo explican un 50% de la varianza.

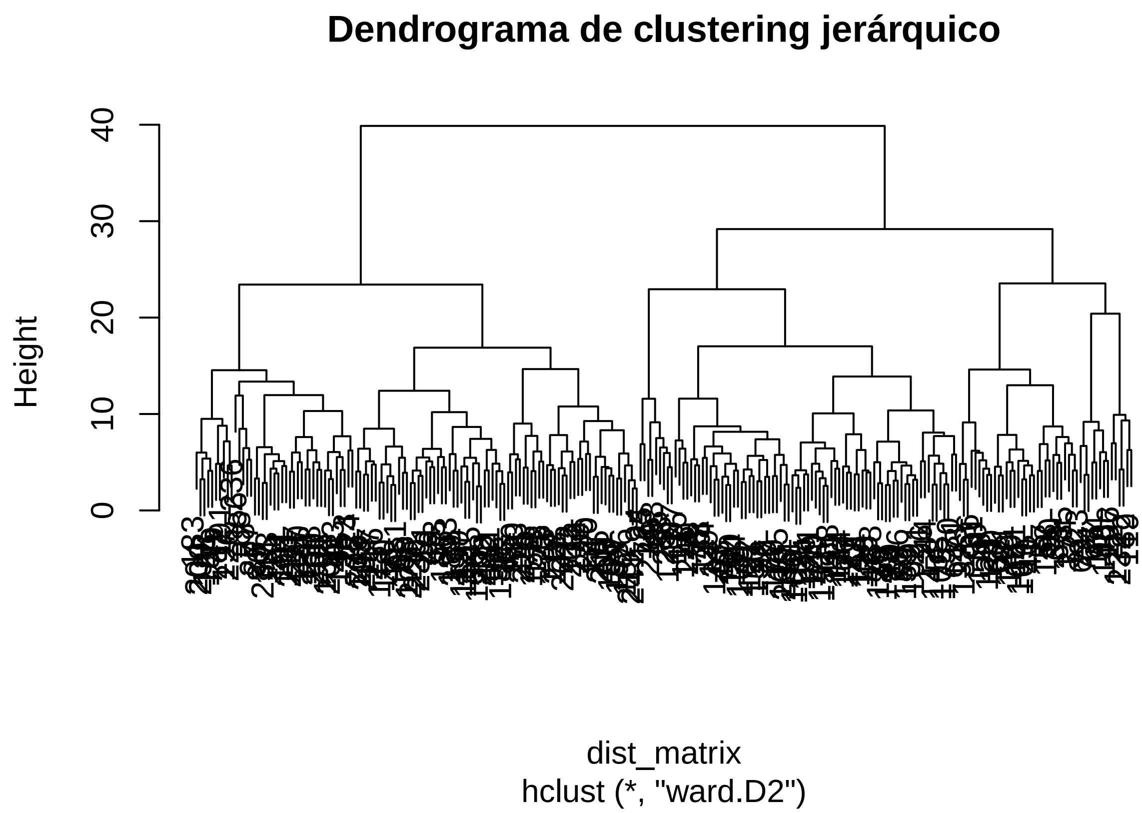
```
# Visualzación más sencilla
fviz_pca_ind(pca,
             habillage = df_def_clean$Target,
             label = "none",
             repel = TRUE,
             title = "Análisis de componentes principales (Diagnóstico como variable suplementaria)")
```




```
#--------------------------------------------------------------------------------
# IMAGEN 8
#--------------------------------------------------------------------------------
pdf("outputs/images/08_PCA.pdf", width = 16, height = 10)
# Visualzación más sencilla
fviz_pca_ind(pca,
             habillage = df_def_clean$Target,
             label = "none",
             repel = TRUE,
             title = "Análisis de componentes principales (Diagnóstico como variable de agrupación)")
```

```
# ----------------------
# Clustering para patrones
# ----------------------
#Generamos matriz de distancias y dendrograma
dist_matrix <- dist(scale(numeric_vars_def))
```

```r
hclust_model <- hclust(dist_matrix, method = "ward.D2")
plot(hclust_model, main = "Dendrograma de clustering jerárquico")
```




```r
#Cortamos el dendrograma en 2 clústeres
clusters <- cutree(hclust_model, k = 2)  # Cambiar "k" al número de clústeres sobre el que se cortará

#Ver asignaciones de clústeres para las observaciones
print(clusters)
```

```
  [1] 1 1 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 1 1 2 1 2 1 2 2 2 1 1 1 1 2 1 1 2 1
 [38] 1 1 2 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1
 [75] 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2 1 1 1 1 2 1 1 1
[112] 1 2 1 2 1 2 2 1 1 1 1 1 2 1 1 1 2 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
[149] 1 2 2 1 2 1 2 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 2 2 2 1 1 1 1 2 1 2 2 2 2 2 2
[186] 2 2 1 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
[223] 2 2 2 2 1 2 2 2 2 1 1 2 2 2 2 2 2 1 2
```

Verificar como funciona el clúster.

```r
#Asociamos observaciones con cada clúster
cluster_observations <- split(row.names(numeric_vars_def), clusters)

#Ver observaciones de cada clúster
print("Observaciones en el clúster 1:")
```

```
[1] "Observaciones en el clúster 1:"
```

```r
print(cluster_observations[[1]])
```

```
  [1] "1"    "2"    "4"    "5"    "10"   "12"   "14"   "17"   "18"   "20"   "21"   "23"
```

```
 [13] "25"  "29"  "30"  "31"  "32"  "34"  "35"  "37"  "38"  "39"  "41"  "42" 
 [25] "43"  "44"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "66"  "67"  "69" 
 [37] "70"  "71"  "72"  "73"  "74"  "76"  "77"  "78"  "79"  "81"  "82"  "83" 
 [49] "84"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"  "96"  "97" 
 [61] "98"  "99"  "100" "102" "104" "105" "106" "107" "109" "110" "111" "112"
 [73] "114" "116" "119" "120" "121" "122" "123" "125" "126" "127" "129" "130"
 [85] "134" "135" "136" "137" "138" "139" "140" "141" "142" "143" "144" "146"
 [97] "147" "148" "149" "152" "154" "156" "157" "158" "159" "160" "161" "162"
[109] "164" "165" "166" "168" "169" "170" "174" "175" "176" "177" "179" "188"
[121] "194" "198" "221" "227" "232" "233" "240"
```

```r
print("Observaciones en el clúster 2:")
```

```
[1] "Observaciones en el clúster 2:"
```

```r
print(cluster_observations[[2]])
```

```
  [1] "3"   "6"   "7"   "8"   "9"   "11"  "13"  "15"  "16"  "19"  "22"  "24" 
 [13] "26"  "27"  "28"  "33"  "36"  "40"  "45"  "46"  "47"  "48"  "49"  "50" 
 [25] "51"  "52"  "53"  "54"  "55"  "56"  "57"  "65"  "68"  "75"  "80"  "85" 
 [37] "95"  "101" "103" "108" "113" "115" "117" "118" "124" "128" "131" "132"
 [49] "133" "145" "150" "151" "153" "155" "163" "167" "171" "172" "173" "178"
 [61] "180" "181" "182" "183" "184" "185" "186" "187" "189" "190" "191" "192"
 [73] "193" "195" "196" "197" "199" "200" "201" "202" "203" "204" "205" "206"
 [85] "207" "208" "209" "210" "211" "212" "213" "214" "215" "216" "217" "218"
 [97] "219" "220" "222" "223" "224" "225" "226" "228" "229" "230" "231" "234"
[109] "235" "236" "237" "238" "239" "241"
```

```r
#Filtramos datos por clústeres (trabajar con subconjuntos)
cluster_1 <- numeric_vars_def[clusters == 1, ]
cluster_2 <- numeric_vars_def[clusters == 2, ]

#Ver los subconjuntos
print("Datos del clúster 1:")
```

```
[1] "Datos del clúster 1:"
```

```r
print(cluster_1)
```

```
      ag ptg19 sex_Woman el_Elementary el_High.School el_Secondary
1   26.9 38.23         1             0              1            0
2   60.5 26.34         0             0              0            0
4   44.6 24.14         1             0              0            0
5   61.9 32.80         0             0              0            0
10  51.2 28.66         1             0              0            0
12  36.4 28.45         1             0              1            0
14  48.8 30.98         1             0              0            0
17  54.7 35.75         1             0              0            0
18  47.5 25.44         1             0              1            0
20  46.3 33.01         1             0              1            0
21  33.5 25.10         1             0              0            0
23  53.9 26.57         0             1              0            0
25  55.4 22.68         1             0              0            1
29  39.7 28.13         1             0              0            0
30  46.1 26.41         1             0              0            0
31  63.0 26.47         0             0              0            0
32  42.6 22.21         1             0              0            0
```

34	63.0	19.70	1	0	0	0
35	39.2	27.71	1	0	1	0
37	59.7	22.10	1	0	1	0
38	43.0	26.25	0	0	0	0
39	44.7	22.62	1	0	1	0
41	50.4	39.94	1	0	0	0
42	56.6	25.80	1	0	0	0
43	46.7	21.95	1	0	0	0
44	48.8	24.01	1	0	1	0
58	47.0	21.29	1	0	1	0
59	51.4	23.05	0	0	1	0
60	67.2	32.21	1	0	1	0
61	53.8	25.71	1	0	1	0
62	55.3	28.67	0	0	1	0
63	57.8	39.08	1	0	0	0
64	57.9	25.44	1	1	0	0
66	56.7	29.97	0	0	1	0
67	40.3	26.89	1	0	0	0
69	65.7	36.88	1	1	0	0
70	43.6	17.62	1	0	0	0
71	48.5	32.25	0	0	0	0
72	41.8	23.07	1	0	1	0
73	69.8	33.24	0	1	0	0
74	54.4	32.81	1	0	1	0
76	48.1	28.52	1	0	0	1
77	45.5	28.19	1	1	0	0
78	48.3	34.36	1	0	0	0
79	53.7	21.85	1	0	0	0
81	61.8	27.21	1	0	0	0
82	54.9	37.80	1	1	0	0
83	49.8	23.65	1	0	1	0
84	50.3	29.34	1	0	1	0
86	42.0	25.64	1	0	0	1
87	49.5	31.92	1	0	0	0
88	51.4	30.61	1	0	0	1
89	55.6	22.83	1	0	1	0
90	55.6	35.77	0	0	0	0
91	44.8	21.33	1	0	1	0
92	50.2	40.32	1	0	0	0
93	43.3	16.05	1	0	0	0
94	37.9	33.67	1	0	0	1
96	26.4	42.40	1	0	1	0
97	56.4	26.71	0	0	1	0
98	53.3	33.21	0	0	1	0
99	62.6	36.92	0	1	0	0
100	39.8	20.56	1	0	1	0
102	48.2	24.35	1	0	1	0
104	47.2	17.38	1	0	0	0
105	41.5	22.81	1	0	1	0
106	49.1	29.40	0	0	0	0
107	42.2	57.46	1	0	0	0
109	47.5	29.26	1	0	1	0
110	56.9	24.44	0	0	1	0
111	54.3	33.45	1	0	1	0

112	40.5	25.64	0	0	1	0
114	48.6	38.29	1	0	0	0
116	37.9	41.27	0	0	1	0
119	56.5	27.24	0	1	0	0
120	53.2	34.17	1	0	0	0
121	50.7	27.21	1	0	0	1
122	61.3	30.59	0	0	0	0
123	55.4	25.75	0	0	1	0
125	39.1	33.57	0	0	0	1
126	46.0	44.23	0	0	1	0
127	47.6	33.87	1	0	0	0
129	59.9	28.93	1	0	1	0
130	40.0	20.00	1	0	0	0
134	59.6	37.97	1	0	1	0
135	44.6	20.90	1	0	0	0
136	48.4	20.62	1	0	1	0
137	52.4	36.31	1	0	0	0
138	48.8	30.21	0	0	1	0
139	48.4	22.31	1	0	1	0
140	48.4	27.97	1	0	1	0
141	51.7	17.94	1	0	1	0
142	53.9	21.15	1	0	1	0
143	45.2	24.16	0	0	1	0
144	41.9	28.95	1	0	0	0
146	50.6	22.32	1	1	0	0
147	37.0	25.30	1	0	1	0
148	57.6	32.78	1	0	0	0
149	57.7	37.72	1	0	0	0
152	54.5	23.84	1	0	1	0
154	49.0	28.76	1	0	0	0
156	50.6	22.50	1	0	1	0
157	43.2	29.30	1	0	0	0
158	57.2	29.34	1	0	0	0
159	54.2	22.82	1	0	1	0
160	52.3	30.35	1	0	0	0
161	39.8	22.34	1	0	0	0
162	54.9	19.54	1	0	1	0
164	44.3	30.51	0	1	0	0
165	40.6	27.79	0	0	0	0
166	54.8	29.79	1	0	0	0
168	47.3	36.06	1	0	1	0
169	50.2	24.01	1	0	0	0
170	38.2	19.14	1	0	0	0
174	44.2	28.28	1	0	0	0
175	39.5	28.26	1	0	1	0
176	46.9	22.47	1	0	0	0
177	35.2	26.40	1	0	1	0
179	53.1	21.21	1	0	0	0
188	31.7	23.18	1	0	1	0
194	43.5	19.43	0	0	0	0
198	67.5	21.88	1	0	0	1
221	34.5	33.24	1	0	0	1
227	35.8	47.02	1	0	1	0
232	54.5	39.36	1	0	0	0

```
233 48.6 35.23         1            0            0            0
240 51.1 23.83         0            0            0            0
    el_Specialist.Master el_University.Deg. spec1_Hospitalization
1                      0                  0                     0
2                      1                  0                     1
4                      0                  1                     0
5                      1                  0                     1
10                     0                  1                     1
12                     0                  0                     0
14                     0                  1                     0
17                     0                  1                     0
18                     0                  0                     0
20                     0                  0                     0
21                     0                  1                     0
23                     0                  0                     1
25                     0                  0                     0
29                     0                  1                     0
30                     1                  0                     0
31                     0                  1                     1
32                     0                  1                     0
34                     0                  1                     1
35                     0                  0                     1
37                     0                  0                     1
38                     0                  1                     0
39                     0                  0                     0
41                     0                  1                     1
42                     0                  1                     0
43                     0                  1                     0
44                     0                  0                     0
58                     0                  0                     0
59                     0                  0                     1
60                     0                  0                     1
61                     0                  0                     0
62                     0                  0                     1
63                     0                  1                     1
64                     0                  0                     0
66                     0                  0                     1
67                     0                  1                     0
69                     0                  0                     1
70                     0                  1                     0
71                     0                  1                     0
72                     0                  0                     1
73                     0                  0                     1
74                     0                  0                     0
76                     0                  0                     0
77                     0                  0                     0
78                     0                  1                     0
79                     0                  1                     0
81                     0                  1                     1
82                     0                  0                     0
83                     0                  0                     0
84                     0                  0                     0
86                     0                  0                     0
87                     0                  1                     1
```

88	0	0	1
89	0	0	0
90	0	1	1
91	0	0	0
92	0	1	0
93	0	1	0
94	0	0	0
96	0	0	0
97	0	0	0
98	0	0	1
99	0	0	0
100	0	0	0
102	0	0	0
104	0	1	1
105	0	0	0
106	1	0	0
107	0	1	1
109	0	0	0
110	0	0	1
111	0	0	0
112	0	0	0
114	0	1	0
116	0	0	0
119	0	0	0
120	0	1	0
121	0	0	0
122	0	1	0
123	0	0	1
125	0	0	1
126	0	0	0
127	0	1	0
129	0	0	0
130	0	1	0
134	0	0	0
135	0	1	0
136	0	0	0
137	0	1	0
138	0	0	0
139	0	0	0
140	0	0	1
141	0	0	0
142	0	0	0
143	0	0	0
144	0	1	0
146	0	0	0
147	0	0	0
148	0	1	1
149	0	1	1
152	0	0	0
154	0	1	0
156	0	0	0
157	0	1	0
158	1	0	1
159	0	0	0

160	0	1	0
161	0	1	1
162	0	0	0
164	0	0	0
165	0	0	0
166	0	1	0
168	0	0	0
169	0	1	1
170	0	1	0
174	0	1	1
175	0	0	0
176	0	1	0
177	0	0	0
179	0	1	0
188	0	0	0
194	0	1	0
198	0	0	0
221	0	0	0
227	0	0	0
232	0	1	0
233	0	1	0
240	0	1	0

	spec1_Mild.Moderated	spec1_UCI	tn46	tn52	tn36	tn38	tn40	tn42	tn22	tn44
1	1	0	0.3	1.0	-1.00	-0.3	0.0	-1.3	0.4	-0.7
2	0	0	1.0	0.0	0.00	0.3	0.0	0.0	0.4	-1.3
4	1	0	0.0	-0.6	0.30	0.0	1.0	1.3	-0.9	1.0
5	0	0	0.0	0.0	0.30	-0.3	0.3	-0.3	-0.2	-0.3
10	0	0	-0.3	0.7	0.00	-0.3	-0.3	0.3	0.1	-0.7
12	1	0	-0.7	0.7	-2.00	-1.0	-1.0	-1.7	0.1	0.0
14	1	0	-0.7	-0.3	-0.30	-0.3	-1.0	-0.7	1.3	-0.7
17	1	0	-0.7	-2.0	-2.00	-1.3	-1.3	-1.0	-0.5	-2.0
18	1	0	-0.7	0.0	1.30	2.0	0.3	-1.3	-0.9	-0.7
20	1	0	-1.0	0.0	0.00	-0.3	-0.3	-0.3	0.2	0.0
21	1	0	0.3	1.0	-0.70	-0.3	-0.7	-2.7	-0.1	0.0
23	0	0	-2.0	0.0	0.30	1.0	1.3	0.3	0.2	-1.3
25	1	0	0.7	0.0	-0.30	-0.3	-0.3	-0.7	0.8	-0.7
29	1	0	-0.7	-1.3	0.70	-0.7	-0.7	-2.0	0.3	1.0
30	1	0	0.7	0.0	0.70	-0.7	0.3	-2.0	1.2	1.7
31	0	0	-0.7	-1.0	-0.70	0.0	-0.7	-0.7	-0.2	1.0
32	1	0	0.3	-0.3	1.70	-0.3	-0.7	-0.7	-0.2	-2.0
34	0	0	0.0	-1.7	-0.30	0.3	1.0	-0.3	0.7	-0.3
35	0	0	1.3	0.0	0.70	1.7	0.0	0.3	-1.2	0.0
37	0	0	0.0	-0.7	1.30	0.3	0.0	0.0	0.2	0.0
38	0	0	0.0	0.7	0.30	-0.3	0.0	-0.3	-1.3	-1.3
39	1	0	0.3	0.6	0.00	0.0	-0.3	-1.0	1.4	-0.7
41	0	0	-0.3	0.4	-0.30	-0.3	-0.3	-1.0	0.5	-0.7
42	1	0	-0.3	0.4	-1.70	-1.0	-1.0	-2.0	0.5	-0.3
43	1	0	-1.0	1.4	-1.00	-1.0	-1.0	-2.7	-0.6	-0.7
44	1	0	0.3	1.0	0.00	-0.3	-1.3	-0.3	1.4	0.0
58	1	0	0.3	0.0	-1.00	-0.7	-1.3	-2.7	1.6	0.0
59	0	0	0.0	-0.7	-0.30	-0.7	0.0	0.0	-0.1	0.3
60	0	0	0.3	0.0	-0.70	-0.7	-1.3	-1.7	0.7	-0.7
61	1	0	-0.7	0.6	-2.30	-2.0	-2.0	-2.3	0.7	-2.0
62	0	0	0.0	0.4	0.00	0.3	-0.3	-1.7	0.6	-0.3

63	0	0	-0.3	-0.6	1.00	0.7	0.0	0.0	-0.2	0.3
64	1	0	0.0	0.0	-0.70	-0.7	-0.3	-1.0	-0.6	-2.7
66	0	0	-0.3	1.0	0.00	1.3	1.0	0.3	-0.1	1.0
67	1	0	-1.0	1.3	0.30	0.0	-0.7	-1.0	0.5	-0.7
69	0	0	0.3	-1.3	-0.70	-0.3	-0.7	-2.0	0.5	-1.0
70	1	0	-1.0	-1.0	1.30	0.3	0.0	-0.3	-1.8	-2.0
71	1	0	0.3	0.3	-0.70	-0.7	-0.7	-2.3	2.6	0.0
72	0	0	-0.7	-0.7	0.70	0.3	-0.7	-0.7	0.4	0.0
73	0	0	0.7	-0.4	-1.00	0.0	-1.0	-1.3	0.7	-0.7
74	1	0	0.0	-0.3	-1.70	-1.0	-1.0	-0.7	-0.1	0.0
76	1	0	-1.0	-0.3	-1.00	-0.7	-1.0	-2.3	1.3	-2.0
77	1	0	-0.7	1.0	0.00	0.7	0.7	-1.0	1.0	-1.7
78	1	0	0.3	-0.3	-0.30	-0.7	-1.0	0.3	-2.3	-0.7
79	1	0	0.0	-0.3	-1.00	-0.7	-0.7	-0.7	0.0	-0.7
81	0	0	-0.7	0.4	-0.70	-0.7	-0.3	-2.3	0.5	-3.0
82	0	1	0.0	0.0	0.00	-1.0	-1.3	-0.3	-0.8	-0.3
83	1	0	0.3	0.0	-0.70	-0.3	0.0	-1.7	0.3	-0.7
84	1	0	-0.7	0.7	-0.30	-1.0	-0.7	-0.3	-0.1	-0.7
86	1	0	-0.7	2.0	0.00	-0.7	-0.7	-0.3	0.7	-0.7
87	0	0	1.7	0.7	0.70	-0.3	-0.7	-1.3	0.3	1.7
88	0	0	0.0	0.3	0.00	0.7	0.3	0.7	0.5	-0.3
89	1	0	0.0	0.7	-0.30	0.0	0.0	-0.3	0.6	-0.3
90	0	0	-2.7	-0.4	-0.30	0.3	0.0	-0.7	0.9	-2.0
91	1	0	-2.3	0.7	-0.30	-1.0	-2.0	-1.0	0.5	0.0
92	1	0	-0.3	-0.6	0.00	0.0	-0.3	-1.0	-0.5	-2.0
93	1	0	-1.0	-0.3	-1.70	-1.0	-1.0	-2.7	0.4	-0.7
94	1	0	-1.0	0.6	-2.30	-2.3	-3.0	-2.7	0.3	0.7
96	1	0	0.7	0.4	-0.70	-1.0	-1.7	-2.0	0.5	0.0
97	1	0	0.0	-0.7	0.00	0.0	0.0	0.0	-0.9	-0.3
98	0	0	0.0	-1.0	-1.00	-0.7	-1.7	0.0	-0.5	0.3
99	1	0	0.7	0.0	1.00	1.3	1.0	-0.3	-1.1	0.3
100	1	0	-2.3	0.3	-2.30	-1.8	-2.7	-1.0	0.0	-0.7
102	1	0	-1.0	0.7	0.00	0.0	0.0	-2.0	1.3	0.0
104	0	0	-0.7	0.3	-0.30	-0.3	-1.0	-1.3	1.0	-0.7
105	1	0	0.3	0.0	-1.30	-1.0	-1.0	-2.0	-0.9	-2.0
106	0	1	-1.3	-1.0	-1.07	-1.0	-1.3	-1.3	1.1	-1.0
107	0	0	-1.3	-0.7	-1.30	-1.0	-1.0	-2.0	-0.4	0.0
109	1	0	0.3	0.0	-1.00	-0.7	-1.3	-1.7	0.6	-0.7
110	0	0	-0.3	0.0	0.00	-0.7	-0.3	-0.3	-1.0	-2.0
111	1	0	1.0	-0.7	-1.70	-1.0	-1.7	-2.0	0.5	0.3
112	1	0	-1.0	0.3	-1.00	-0.7	-1.0	-1.0	-0.1	-2.0
114	1	0	-1.3	0.3	-0.30	-2.0	-0.3	-2.3	-0.9	-2.0
116	1	0	-0.7	-1.3	0.00	-0.3	-0.7	-1.3	-2.0	0.0
119	0	1	0.3	-1.3	0.30	0.7	0.0	0.3	1.8	0.3
120	1	0	-0.7	-0.3	-1.30	-1.0	-1.0	-3.1	-0.6	-2.0
121	1	0	2.7	0.6	0.30	0.0	0.0	0.3	1.8	1.3
122	0	1	0.0	-0.6	0.30	1.7	0.3	-1.0	1.0	0.3
123	0	0	-0.3	0.3	-0.70	-0.3	-0.3	-1.3	0.6	-2.0
125	0	0	-1.0	-0.7	-1.30	-1.7	-2.0	-2.3	-1.2	-2.0
126	1	0	-1.0	-1.0	-1.70	-1.7	-1.7	-1.7	-0.3	-2.0
127	1	0	-0.7	0.3	-0.30	-0.7	0.0	-0.3	0.7	-0.7
129	1	0	-0.3	0.4	-0.70	-1.0	-0.7	-2.0	-0.6	0.0
130	1	0	-1.0	0.0	-0.70	0.0	0.0	-0.3	-0.9	-2.0
134	1	0	0.0	0.3	-1.70	-1.0	-1.7	-1.3	0.0	-1.3

135	1	0	-0.7	0.3	0.70	-0.3	0.0	-1.0	0.6	-2.0
136	1	0	-0.7	-0.6	-0.30	-0.3	-1.0	-1.0	0.7	-2.0
137	1	0	-0.7	0.0	0.30	0.3	-0.3	-1.0	0.4	-0.7
138	1	0	-1.0	-1.4	-0.30	-0.3	-0.3	-0.3	-0.8	-2.0
139	1	0	-1.0	-0.4	0.70	1.3	0.3	-1.3	0.2	0.0
140	0	0	-0.7	0.0	-1.70	-0.7	-1.0	-2.3	0.7	-2.0
141	1	0	-0.3	-0.4	-2.00	-1.0	-1.3	-2.3	0.6	0.0
142	1	0	0.0	1.0	-0.70	-1.0	-0.3	0.3	0.7	0.3
143	1	0	-0.7	-0.7	0.00	0.3	0.0	-1.0	0.3	0.0
144	1	0	-1.0	0.0	-2.70	-2.3	-3.0	-1.0	-0.8	-2.0
146	1	0	0.3	-0.4	1.30	1.7	0.3	1.0	-0.3	1.7
147	1	0	-0.7	0.3	-0.30	-0.7	-1.7	-1.3	0.4	-0.7
148	0	0	-0.7	-1.0	-0.70	-1.3	-0.7	-0.7	-0.5	-0.3
149	0	0	-0.7	0.0	-1.30	-1.3	-2.0	-2.3	-0.1	-1.7
152	1	0	-0.3	0.0	-1.70	-1.0	-0.7	-2.0	0.2	0.3
154	1	0	-1.0	0.0	-2.70	-1.3	-1.0	-2.7	0.0	-2.0
156	1	0	-0.3	0.3	-1.30	-1.3	-1.3	-2.3	0.8	-0.7
157	1	0	-1.0	-0.4	-0.30	-0.7	0.0	-1.0	-0.8	-2.0
158	0	0	1.0	0.7	-0.30	0.3	-0.3	-0.3	1.7	1.3
159	1	0	0.0	-0.4	0.00	0.7	0.0	-1.0	1.2	0.3
160	1	0	-0.7	-1.6	-1.30	-0.7	-0.7	-2.0	0.5	-0.7
161	0	0	0.3	0.7	0.00	-0.3	0.0	0.7	0.7	-0.7
162	1	0	-0.3	0.0	-1.00	-0.7	-0.3	-1.7	0.3	-0.7
164	1	0	0.7	1.0	-0.70	0.0	-1.0	-1.3	-0.9	0.3
165	1	0	-1.0	1.3	-0.70	-0.3	0.0	-1.3	-0.3	-1.0
166	1	0	-0.3	0.6	-0.70	-0.3	-0.3	-0.7	0.0	-2.0
168	1	0	-0.7	0.7	-1.30	-1.0	-0.7	-1.7	0.8	-2.0
169	0	0	-0.7	0.4	-0.70	-0.7	-0.7	-1.3	2.1	0.0
170	1	0	0.0	1.7	1.00	0.0	0.0	-1.0	1.5	0.0
174	0	0	-1.0	-0.3	0.70	1.3	0.3	-1.3	-0.1	0.0
175	1	0	0.3	0.3	-0.30	-0.3	-0.3	-2.0	0.7	0.0
176	1	0	0.7	0.7	-1.30	-1.7	-1.0	-2.0	0.4	-2.0
177	1	0	-1.0	0.0	-2.30	-2.0	-2.7	-2.3	0.3	-2.0
179	1	0	-0.3	-0.7	0.00	0.0	0.0	-2.3	0.3	-2.0
188	1	0	-1.0	0.0	0.00	0.3	0.3	-0.3	-1.5	0.0
194	1	0	-0.7	-0.4	-0.30	0.0	-1.0	-1.7	-0.2	-0.7
198	0	0	-0.3	-1.4	0.00	0.3	0.3	0.3	-1.1	-0.7
221	1	0	-1.0	-1.0	-1.00	0.0	-0.3	-0.3	1.4	-2.0
227	0	1	-0.7	-1.0	-1.70	-1.0	-1.3	-2.0	-1.4	-0.7
232	0	1	-0.3	-0.4	-0.30	0.3	1.0	0.3	1.0	0.0
233	1	0	-0.7	1.0	-0.70	-0.7	-1.0	-1.0	0.0	-2.0
240	1	0	0.7	0.0	-3.00	-1.0	-0.3	0.3	-1.8	0.0

	tn14	tn24	tn12	tn6	tn30	tn34	tn8	tn48	tn50	Target
1	-1.7	-3.0	-2.0	-2.06	-2.30	-1.0	2.0	1.0	-1.7	1
2	1.0	0.0	0.0	0.86	0.70	0.3	-0.3	3.1	1.3	1
4	-0.3	-0.7	-0.3	-0.82	-0.43	-1.0	-0.7	1.3	-0.3	1
5	1.3	-0.3	1.3	-0.57	0.40	0.3	-0.3	3.1	-0.7	1
10	0.7	-0.7	0.0	-0.32	0.03	-2.0	0.3	0.8	0.7	1
12	0.3	-1.0	-0.3	-1.03	-0.03	0.0	-0.3	3.1	-0.7	1
14	0.7	-1.7	-0.7	-2.22	-1.14	-1.0	0.3	1.3	0.7	1
17	-0.3	0.7	0.0	0.04	-0.28	-0.3	0.0	0.7	-1.0	1
18	-0.3	-0.3	0.0	0.01	-0.42	-0.7	-0.3	1.0	-0.7	1
20	-0.3	-0.3	0.0	-1.52	-1.14	-0.7	-0.3	1.3	0.0	1
21	-0.7	-2.0	-1.3	-2.36	-1.46	0.0	0.3	0.8	-0.7	1

23	0.0	0.0	0.0	-1.80	-0.59	0.3	0.3	1.3	-1.3	1
25	1.0	-0.3	1.0	-0.93	-0.90	-0.7	-0.3	0.5	0.7	1
29	0.7	-1.0	0.3	-1.39	-0.75	-1.3	-0.3	1.0	-0.7	1
30	-0.7	-0.7	-0.7	-1.87	-0.78	0.0	0.3	1.3	0.0	1
31	0.7	-0.7	0.7	-0.96	-1.60	-0.7	-2.0	0.7	1.3	1
32	0.3	-0.7	0.7	0.24	-0.42	-0.3	-0.3	1.3	0.7	1
34	1.0	1.0	1.3	0.07	0.06	0.0	-0.3	0.8	0.7	1
35	1.3	1.0	1.0	0.98	0.40	1.3	0.7	0.3	0.3	1
37	1.0	0.0	1.0	0.79	0.03	0.0	0.3	3.0	0.7	1
38	0.7	-1.0	0.7	-1.48	-1.60	-0.3	-0.3	1.3	-0.3	1
39	0.7	-0.3	1.0	0.22	-0.78	-0.3	-0.3	1.0	0.0	1
41	-0.3	-0.7	-0.3	0.04	0.03	-0.3	-0.3	1.3	0.0	1
42	-0.3	-1.7	-0.3	-0.93	-0.90	-0.3	0.0	1.0	-0.3	1
43	-0.3	-2.7	-1.3	-1.29	-1.50	0.0	2.3	1.0	0.0	1
44	-0.3	-2.3	-1.0	0.80	1.35	1.3	0.7	3.0	-1.0	1
58	-1.0	-1.0	-0.7	-0.59	-0.07	-0.3	0.7	3.1	0.3	1
59	-0.3	-0.3	-1.0	-2.91	-1.21	-0.3	-1.3	1.3	0.0	1
60	-0.3	-0.3	-1.0	-0.31	-0.93	-1.7	-0.3	0.3	0.3	1
61	0.3	-1.3	-0.7	-1.18	-0.90	-0.7	-0.3	0.5	0.5	1
62	0.0	-0.7	-1.3	-0.69	-0.90	-1.3	0.0	0.8	0.7	1
63	1.3	0.3	2.0	0.42	0.65	0.3	-0.7	3.1	1.7	1
64	0.0	-1.0	-0.3	-0.93	-0.59	-2.3	-0.7	0.5	-0.3	1
66	1.3	0.7	1.0	-0.36	-1.86	-1.0	-1.0	1.0	-0.7	1
67	-0.7	-2.3	-0.3	-0.71	-0.43	-1.0	-0.3	3.0	0.0	1
69	-0.3	0.0	-1.0	-1.22	-0.93	-1.0	0.0	0.7	-0.7	1
70	-0.3	-1.0	0.3	-0.01	-0.07	-1.0	0.7	0.8	0.3	1
71	-1.7	-1.0	-0.7	-0.24	-0.07	-0.7	0.7	0.9	0.3	1
72	0.3	-1.0	0.0	0.22	1.00	-1.3	0.0	1.3	0.0	1
73	0.0	0.7	-0.3	-1.22	-0.93	-0.3	-0.3	0.7	0.3	1
74	-0.3	-2.0	-2.0	-1.43	-0.59	-1.7	-1.0	0.7	0.3	1
76	-0.3	-1.0	-0.7	0.33	-0.42	-0.7	0.7	0.7	-1.7	1
77	-0.3	-1.7	0.0	-0.24	0.28	-1.3	-0.7	3.1	0.3	1
78	-0.7	0.0	-0.7	1.03	1.35	-1.0	-0.7	1.0	0.3	1
79	-0.3	-0.7	0.7	-0.44	-0.59	0.0	-0.3	1.0	-0.3	1
81	0.3	-0.7	0.0	-1.22	-0.60	-0.7	0.3	3.1	0.7	1
82	-0.3	0.0	-0.3	-0.32	0.03	-0.7	-0.3	1.0	-1.0	1
83	0.0	-0.7	-0.3	0.33	-0.07	-2.0	-0.3	0.8	0.0	1
84	0.3	-0.7	-0.3	0.54	0.03	-0.3	0.3	1.3	0.7	1
86	0.7	-2.7	0.3	1.50	1.35	0.0	0.7	1.0	-0.3	1
87	0.3	0.3	0.0	1.84	1.00	0.0	0.3	1.3	0.3	1
88	0.7	-0.3	-0.3	-0.44	-0.59	-1.0	1.7	1.3	0.3	1
89	0.0	-1.7	-0.3	-0.32	-1.22	-0.3	0.3	1.3	0.0	1
90	1.3	-0.3	1.7	0.54	0.34	0.0	-0.3	3.1	1.3	1
91	-0.3	-2.0	-0.3	-1.05	-0.78	-1.7	-1.0	1.3	0.3	1
92	-0.3	0.3	-0.3	1.03	0.34	-0.7	-0.3	0.8	0.8	1
93	0.0	-1.0	-0.3	-0.59	0.28	0.7	0.0	1.3	-0.7	1
94	0.0	-2.3	-0.7	-1.27	-0.75	-0.3	-1.0	0.7	-0.7	1
96	-1.0	-2.7	-2.0	-0.70	0.30	-0.3	0.3	1.0	0.0	1
97	1.0	0.7	0.3	-1.80	-1.21	0.3	-0.3	1.0	0.7	1
98	-0.7	-0.7	2.0	-1.80	-1.53	-0.3	-1.0	1.0	-0.7	1
99	-1.0	0.7	1.0	-2.77	-0.93	0.0	0.0	1.0	0.3	1
100	0.0	-1.3	-0.7	-1.03	-0.39	0.0	0.3	0.6	-0.7	1
102	0.3	0.0	-0.3	-1.40	-0.42	-1.0	-0.3	0.8	-0.7	1
104	0.0	0.0	-0.7	-1.75	-1.85	-1.0	-1.0	1.3	0.7	1

105	0.3	-1.0	-0.3	-2.33	-0.78	0.3	0.7	0.8	0.0	1
106	-0.3	-1.0	0.0	-1.40	-0.78	-0.7	-1.3	1.0	-0.7	1
107	-0.3	-1.0	-0.3	-0.36	1.00	0.0	-1.3	1.0	-0.7	1
109	-0.3	-1.0	-1.3	-0.36	0.28	-0.7	0.3	1.3	0.3	1
110	1.0	-0.7	0.7	-1.18	-0.59	0.3	-0.7	1.0	0.7	1
111	-1.3	-0.3	-0.3	-2.66	-1.84	-0.3	-0.3	1.3	-0.7	1
112	0.0	-1.0	-1.3	-1.63	-1.14	-1.3	-1.0	1.0	0.0	1
114	1.0	-1.0	0.3	-1.52	-1.14	-0.3	-0.7	1.0	-1.3	1
116	-2.3	0.0	-0.7	-0.67	-0.03	-0.7	-1.0	1.3	-0.3	1
119	1.0	2.3	0.0	-1.30	-0.90	-1.0	0.3	1.3	0.0	1
120	-1.3	-2.0	-2.0	-1.06	-0.59	-1.7	-1.7	0.7	0.0	1
121	0.3	-0.3	1.0	-0.07	0.34	-0.7	1.3	1.3	0.7	1
122	0.7	0.3	0.7	-1.48	-0.93	-1.0	-0.3	3.1	0.3	1
123	0.3	-1.0	-0.3	-2.91	-1.84	-1.0	-1.0	0.8	-0.3	1
125	-1.3	-2.0	-2.0	-1.63	-1.82	-1.3	-1.7	1.0	0.3	1
126	-0.3	-0.7	-0.3	-1.17	0.28	0.3	0.7	0.5	-1.0	1
127	0.7	-0.3	-1.0	-0.12	0.64	-0.3	-0.7	0.8	0.0	1
129	0.0	-1.7	-0.7	-2.54	-2.15	-2.7	0.0	0.7	0.0	1
130	-0.3	-1.0	-1.0	-1.63	-1.46	0.0	-0.3	3.0	-0.3	1
134	1.0	0.0	0.3	-1.80	-0.90	0.0	0.3	1.0	0.3	1
135	1.0	0.0	-1.0	-0.12	0.28	0.0	0.7	1.0	0.7	1
136	-0.7	-0.7	-0.7	-1.29	-0.07	-0.7	0.3	0.3	-1.3	1
137	0.0	-0.3	-0.3	-1.43	-2.15	-0.3	-0.7	1.3	0.3	1
138	0.0	0.7	0.7	-2.33	-0.78	-0.3	-0.3	1.3	0.0	1
139	0.0	-0.3	-0.7	0.80	1.00	-0.7	-0.3	1.3	0.0	1
140	-1.0	-1.0	-1.3	-0.94	-0.07	-0.3	-0.3	1.3	0.0	1
141	-0.7	-1.3	-1.7	-2.29	-1.53	-1.0	-0.3	0.8	0.0	1
142	-0.7	-2.7	0.0	-0.19	-0.90	-1.3	-0.3	1.0	-0.3	1
143	-0.3	0.7	1.0	-5.94	-1.50	0.0	-1.0	1.0	0.0	1
144	-0.7	-2.7	-1.3	-2.33	-1.85	-0.7	-0.3	0.8	-0.7	1
146	0.7	0.7	0.0	-0.69	-1.21	0.0	0.0	1.3	0.0	1
147	-0.3	-1.3	-0.3	-1.03	-0.39	-0.7	-0.3	3.0	-0.3	1
148	1.7	0.0	-0.7	-1.80	-1.53	-1.0	-1.3	1.3	1.0	1
149	0.7	-0.3	0.0	-0.56	0.03	-0.7	-0.7	0.8	0.3	1
152	-1.3	-1.7	-0.7	-1.92	-0.59	-1.7	-1.0	0.7	-0.3	1
154	-0.3	-1.3	-1.7	-1.52	0.07	0.0	0.3	1.3	0.3	1
156	-0.3	-1.0	-0.3	-1.43	-0.59	0.0	-0.7	1.0	0.0	1
157	-0.7	-1.3	-0.7	-1.40	-1.14	-0.7	0.3	1.3	0.3	1
158	1.3	-0.7	0.0	2.27	1.59	0.3	-0.7	3.0	0.0	1
159	-0.3	-0.3	-0.3	0.54	-1.21	-1.3	-0.3	1.0	0.0	1
160	0.3	-0.7	0.0	-0.56	-0.90	-0.3	0.0	3.0	0.0	1
161	-0.3	-1.7	0.0	-1.03	-2.53	-0.7	-0.7	3.0	-0.3	1
162	0.0	0.3	-0.7	-1.18	-0.90	-0.3	-0.7	1.3	0.3	1
164	0.0	0.0	0.0	-1.05	-1.50	-0.3	-0.3	0.8	-0.7	0
165	0.3	-2.3	-1.0	-0.12	1.00	0.0	0.0	1.0	0.3	1
166	-0.3	-1.3	-0.3	1.65	0.34	0.0	0.0	1.0	0.3	1
168	-1.7	-2.7	-1.3	-0.82	-1.50	-1.0	-1.0	3.1	0.3	1
169	0.3	-0.7	-0.3	0.42	-0.59	-0.7	-1.0	0.8	0.0	1
170	-0.3	-1.7	-0.7	-0.19	-0.39	-0.3	0.3	1.0	0.3	1
174	-1.0	-0.7	-0.7	0.10	1.35	-1.0	-1.3	3.0	1.0	1
175	0.3	-1.0	-0.3	0.53	1.39	-0.7	0.3	1.0	0.7	1
176	-1.0	-2.7	-1.0	-1.52	-0.78	-1.3	-0.3	1.0	0.3	1
177	-0.7	-2.7	-1.7	-2.60	-2.17	0.0	0.7	0.5	-0.7	1
179	0.3	0.0	-0.3	-0.69	-0.28	-0.7	0.0	1.3	0.7	1

```
188  0.7  0.0  0.7  1.13  0.68 -1.3 -1.0  0.8 -0.3      0
194  0.0 -1.3  0.7 -1.05 -0.78  0.0 -1.0  3.0  0.0      1
198  0.3  0.7  1.0  0.07  1.06  0.0 -0.3  0.8  0.0      0
221 -0.3  1.3  1.3 -0.79 -0.39 -1.3 -0.7  1.0 -2.0      0
227 -0.7 -0.3 -0.3 -3.80 -2.17 -2.3 -1.0  0.5 -1.3      1
232  0.0 -0.3 -0.3 -2.29 -0.90 -1.3  0.3  3.1  1.0      1
233  0.0 -2.7 -0.7  0.10  0.64  0.3  0.7  0.8  0.0      1
240  0.0  0.3  0.0 -0.32  0.65 -0.3 -0.3  1.3  0.0      1
```

```r
print("Datos del clúster 2:")
```

```
[1] "Datos del clúster 2:"
```

```r
print(cluster_2)
```

```
      ag    ptg19 sex_Woman el_Elementary el_High.School el_Secondary
3   33.1 18.24000         1             0              0            0
6   43.0 32.70000         1             0              0            0
7   55.6 25.44000         1             0              0            0
8   53.6 18.24000         1             0              0            0
9   46.3 23.23000         1             0              0            0
11  58.7 26.01000         1             0              1            0
13  42.7 19.37000         1             0              0            0
15  48.7 28.31000         1             0              0            0
16  53.6 20.97000         1             0              0            0
19  55.5 23.65000         1             0              0            0
22  64.6 30.90000         1             0              0            0
24  46.6 20.51000         1             0              0            0
26  29.9 17.54000         1             0              0            0
27  56.9 27.41000         1             0              1            0
28  47.6 23.67000         0             0              1            0
33  45.3 23.53000         1             0              0            0
36  61.6 23.87000         1             0              0            0
40  48.7 25.28000         1             0              0            0
45  36.5 29.40000         1             0              0            0
46  65.1 21.64000         1             0              0            0
47  62.8 24.40000         1             0              0            0
48  52.3 19.43000         1             0              0            0
49  53.0 25.40000         0             0              0            0
50  57.6 23.28000         1             0              1            0
51  51.1 27.57000         1             0              0            0
52  45.2 26.25000         1             0              0            0
53  65.9 24.69000         1             0              0            0
54  31.1 22.97000         1             0              0            0
55  42.6 24.58000         0             0              0            0
56  33.3 27.19237         0             0              0            0
57  61.6 29.72000         0             0              0            0
65  55.2 27.70000         1             0              1            0
68  57.2 45.79000         0             0              1            0
75  25.5 22.21000         1             0              0            0
80  44.8 28.12000         1             0              1            0
85  65.9 25.92000         1             0              1            0
95  33.7 39.51000         1             0              0            0
101 52.7 35.62000         1             0              1            0
103 42.7 20.95000         1             0              0            0
```

108	39.9	35.13000	1	0	0	0
113	58.3	23.06000	1	0	1	0
115	44.3	27.55000	1	0	0	0
117	47.1	24.17000	1	0	0	0
118	50.8	22.18000	1	0	1	0
124	57.0	40.61000	0	0	1	0
128	40.1	30.63000	1	0	0	0
131	46.1	30.76000	1	0	0	0
132	58.3	22.97000	1	0	1	0
133	59.7	24.46000	1	0	1	0
145	52.0	24.17000	1	0	0	0
150	43.1	21.91000	1	0	0	0
151	45.9	41.29000	1	0	1	0
153	41.9	20.42000	1	0	0	0
155	59.2	25.38000	1	0	0	0
163	45.8	27.87000	1	0	0	0
167	30.2	24.14000	1	0	0	0
171	38.4	24.19000	1	0	0	0
172	49.6	25.37000	0	0	0	0
173	58.7	28.55000	1	0	0	0
178	38.2	30.56000	1	0	0	0
180	29.4	17.95000	1	0	0	0
181	42.3	33.41000	1	0	0	0
182	41.5	23.57000	1	0	0	0
183	29.4	24.12000	1	0	0	0
184	40.8	25.43000	1	0	0	0
185	65.3	35.22000	0	0	1	0
186	56.1	23.42000	1	0	0	0
187	33.8	20.78000	1	0	0	0
189	61.0	37.46000	1	0	0	0
190	50.7	25.96000	1	0	0	0
191	62.6	28.57000	0	0	0	0
192	54.3	23.95000	1	0	0	0
193	28.9	26.75000	1	0	0	0
195	65.8	28.06000	1	0	1	0
196	48.7	22.63000	1	0	1	0
197	70.8	24.39000	0	0	0	0
199	40.7	27.42241	1	0	0	0
200	64.6	27.85000	0	0	1	0
201	55.3	26.19000	0	0	1	0
202	54.1	24.85000	0	0	0	0
203	57.8	19.17000	1	0	1	0
204	30.1	21.73000	1	0	0	0
205	57.7	38.63000	1	0	1	0
206	31.9	21.80000	1	0	0	0
207	41.1	27.36000	1	0	0	0
208	63.5	29.96000	1	0	0	0
209	40.9	26.15119	1	0	0	0
210	48.3	25.08000	0	0	0	0
211	49.0	23.54000	1	0	0	0
212	50.1	30.64000	0	0	0	0
213	49.0	22.04000	1	0	0	0
214	55.8	34.02000	1	0	1	0
215	43.3	23.44000	1	0	0	0

```
216 35.9 20.18000         1             0             0            0
217 51.6 22.79000         1             0             0            0
218 51.8 36.18000         1             0             0            0
219 34.8 28.10000         1             0             0            0
220 51.4 35.75000         1             0             1            0
222 52.6 21.08000         1             0             0            0
223 50.7 25.11000         1             0             0            0
224 45.4 37.82000         1             0             0            0
225 33.0 20.02000         1             0             0            0
226 57.5 30.02000         1             0             0            0
228 37.9 24.93343         1             0             0            0
229 34.7 24.72016         1             0             0            0
230 42.4 26.62936         1             0             0            0
231 46.3 23.24000         1             0             1            0
234 37.0 21.33000         1             0             0            0
235 37.6 24.42000         1             0             0            0
236 51.0 87.31000         0             0             1            0
237 62.1 29.67000         1             0             0            0
238 63.2 27.35000         0             0             1            0
239 47.1 24.10000         0             0             0            0
241 25.4 16.14000         1             0             0            0
    el_Specialist.Master el_University.Deg. spec1_Hospitalization
3                      0                  0                     0
6                      1                  0                     0
7                      1                  0                     0
8                      0                  1                     0
9                      1                  0                     0
11                     0                  0                     0
13                     0                  1                     0
15                     0                  1                     0
16                     1                  0                     0
19                     1                  0                     0
22                     1                  0                     0
24                     1                  0                     0
26                     1                  0                     0
27                     0                  0                     0
28                     0                  0                     0
33                     1                  0                     0
36                     0                  1                     0
40                     0                  1                     0
45                     0                  1                     0
46                     0                  0                     0
47                     0                  0                     0
48                     0                  0                     0
49                     0                  1                     0
50                     0                  0                     0
51                     0                  0                     0
52                     0                  1                     0
53                     1                  0                     0
54                     1                  0                     0
55                     1                  0                     0
56                     1                  0                     0
57                     0                  1                     0
65                     0                  0                     0
```

68	0	0	0
75	0	1	0
80	0	0	0
85	0	0	0
95	0	1	0
101	0	0	0
103	0	1	0
108	0	1	0
113	0	0	0
115	0	1	0
117	1	0	0
118	0	0	0
124	0	0	0
128	0	0	0
131	0	1	0
132	0	0	0
133	0	0	0
145	0	1	0
150	0	1	0
151	0	0	0
153	1	0	0
155	0	1	0
163	0	1	0
167	0	0	0
171	0	1	0
172	0	1	0
173	1	0	0
178	0	0	0
180	0	1	0
181	0	1	0
182	0	1	0
183	0	1	0
184	0	1	0
185	0	0	0
186	0	1	0
187	1	0	0
189	0	1	0
190	1	0	0
191	0	1	0
192	0	1	0
193	1	0	0
195	0	0	0
196	0	0	0
197	1	0	0
199	0	1	0
200	0	0	0
201	0	0	0
202	1	0	0
203	0	0	0
204	0	1	0
205	0	0	0
206	0	1	0
207	1	0	0
208	0	1	0

209	0	1	0
210	0	1	0
211	1	0	0
212	0	1	0
213	0	1	0
214	0	0	0
215	1	0	0
216	0	0	0
217	0	0	0
218	1	0	0
219	1	0	0
220	0	0	0
222	0	1	0
223	0	0	0
224	0	0	0
225	1	0	0
226	1	0	0
228	0	0	0
229	0	0	0
230	0	0	0
231	0	0	0
234	0	1	0
235	1	0	0
236	0	0	0
237	0	1	0
238	0	0	0
239	1	0	0
241	1	0	0

	spec1_Mild.Moderated	spec1_UCI	tn46	tn52	tn36
3	1	0	0.7000000	2.000000000	1.0000000
6	1	0	0.3000000	0.400000010	-0.7000000
7	1	0	-0.3000000	-0.699999990	-0.3000000
8	1	0	-0.3000000	0.400000010	-0.3000000
9	1	0	-0.7000000	1.000000000	-1.7000000
11	1	0	0.0000000	-0.400000010	0.3000000
13	1	0	-0.7000000	1.000000000	-1.7000000
15	1	0	0.0000000	0.699999990	0.0000000
16	1	0	-0.3000000	-1.700000000	-0.3000000
19	1	0	-0.7000000	0.300000010	-0.3000000
22	1	0	-0.3000000	0.000000000	-0.7000000
24	1	0	0.3000000	1.300000000	0.0000000
26	1	0	-0.7000000	0.699999990	-1.3000000
27	1	0	0.0000000	1.000000000	0.0000000
28	1	0	0.3000000	-0.699999990	0.0000000
33	0	0	0.3000000	1.000000000	-1.3000000
36	1	0	0.0000000	0.699999990	1.0000000
40	1	0	0.3000000	-0.400000010	0.0000000
45	1	0	-0.7000000	1.400000000	-0.7000000
46	1	0	-0.3000000	0.000000000	0.0000000
47	1	0	0.7000000	1.400000000	-0.3000000
48	1	0	0.3000000	0.300000010	-0.7000000
49	1	0	1.3000000	-0.699999990	1.7000000
50	1	0	0.0000000	0.000000000	-0.3000000
51	1	0	2.0000000	0.000000000	-0.7000000

52	1	0	-0.7000000	0.000000000	-0.3000000
53	1	0	0.7000000	-2.300000000	0.0000000
54	1	0	0.0000000	-0.600000020	-0.3000000
55	1	0	-1.0000000	-1.300000000	-1.7000000
56	1	0	2.3000000	-0.400000010	-0.3000000
57	1	0	0.0000000	0.300000010	-0.3000000
65	1	0	1.0000000	1.300000000	1.7000000
68	1	0	0.7000000	0.899999980	0.0000000
75	1	0	1.7000000	-0.300000010	-1.3000000
80	1	0	-0.7000000	-1.700000000	0.7000000
85	1	0	0.0000000	0.600000020	-0.7000000
95	1	0	-1.0000000	0.600000020	-0.7000000
101	1	0	-0.3000000	-0.800000010	0.3000000
103	1	0	0.7000000	0.300000010	-0.3000000
108	1	0	0.7000000	-0.300000010	-0.3000000
113	1	0	0.7000000	0.000000000	2.3000000
115	1	0	-1.0000000	0.000000000	0.7000000
117	1	0	-1.0000000	1.700000000	-0.3000000
118	1	0	1.0000000	-0.400000010	0.3000000
124	1	0	2.3000000	0.300000010	0.0000000
128	1	0	-1.0000000	-0.600000020	1.3000000
131	1	0	-0.7000000	1.000000000	0.0000000
132	1	0	0.0000000	1.400000000	1.0000000
133	1	0	-1.7000000	-0.600000020	-0.7000000
145	1	0	1.0000000	0.300000010	0.3000000
150	1	0	1.7000000	0.300000010	-0.3000000
151	1	0	-0.7000000	-0.600000020	0.7000000
153	1	0	0.0000000	0.000000000	-2.0000000
155	1	0	0.3000000	0.000000000	0.7000000
163	1	0	0.0000000	0.000000000	0.7000000
167	1	0	0.0000000	0.400000010	-0.7000000
171	1	0	0.0000000	1.600000000	0.0000000
172	1	0	-1.3000000	0.000000000	-0.7000000
173	1	0	0.3000000	0.699999990	-1.3000000
178	1	0	0.7000000	0.300000010	-2.3000000
180	0	0	-1.0000000	0.600000020	0.3000000
181	1	0	0.3000000	0.000000000	-0.3000000
182	1	0	0.3000000	0.300000010	0.0000000
183	1	0	0.3000000	0.300000010	2.0000000
184	1	0	0.0000000	0.000000000	1.3000000
185	1	0	0.3000000	0.000000000	0.0000000
186	1	0	-0.3000000	1.700000000	0.3000000
187	1	0	-1.0000000	0.300000010	-0.7000000
189	1	0	0.7000000	-0.400000010	-0.3000000
190	1	0	0.7000000	-0.699999990	0.0000000
191	1	0	-0.7000000	-0.699999990	1.0000000
192	1	0	-0.3000000	0.000000000	0.7000000
193	1	0	-1.0000000	-0.600000020	0.3000000
195	1	0	2.7000000	-1.000000000	1.7000000
196	1	0	1.7000000	1.000000000	0.0000000
197	1	0	0.0000000	0.400000010	0.7000000
199	1	0	-2.3000000	0.300000010	0.3000000
200	1	0	0.3000000	-0.300000010	0.7000000
201	1	0	1.0000000	-1.000000000	1.7000000

202	1	0	1.0000000	-0.699999990	0.0000000
203	0	0	-0.3000000	1.000000000	0.0000000
204	1	0	0.0000000	-0.699999990	0.3000000
205	1	0	-0.3000000	0.400000010	2.0000000
206	1	0	1.0000000	0.699999990	-0.3000000
207	1	0	0.0000000	0.300000010	-0.3000000
208	0	0	0.7000000	0.000000000	-0.3000000
209	1	0	-0.7000000	-0.300000010	1.3000000
210	1	0	-1.0000000	-1.700000000	0.0000000
211	1	0	-1.0000000	-1.000000000	-0.3000000
212	1	0	2.3000000	0.300000010	2.3000000
213	1	0	-1.0000000	0.000000000	-0.3000000
214	1	0	1.0000000	0.000000000	1.3000000
215	0	0	0.0000000	1.300000000	0.0000000
216	1	0	0.0000000	-0.300000010	-0.3000000
217	1	0	2.0000000	-1.400000000	-0.7000000
218	1	0	-0.3000000	0.400000010	-0.3000000
219	1	0	-0.1936254	0.008326257	-0.3344785
220	1	0	2.3000000	-0.300000010	0.3000000
222	0	0	0.7000000	-0.300000010	-0.3000000
223	1	0	1.0000000	-0.699999990	0.0000000
224	0	0	0.0000000	0.699999990	2.3000000
225	1	0	0.0000000	0.400000010	0.3000000
226	1	0	-0.7000000	0.000000000	1.7000000
228	1	0	0.3389087	0.154617766	-0.3216190
229	1	0	0.0000000	1.300000000	-0.3000000
230	1	0	0.0000000	0.000000000	0.0000000
231	1	0	-0.7000000	-1.300000000	0.0000000
234	1	0	-0.7000000	0.699999990	-1.7000000
235	1	0	0.0000000	-0.300000010	0.0000000
236	1	0	1.7000000	0.000000000	0.7000000
237	1	0	0.0000000	-0.300000010	0.0000000
238	1	0	2.7000000	-1.300000000	0.0000000
239	1	0	0.0000000	0.000000000	-0.3000000
241	1	0	-0.4050728	0.211006487	-0.5695441

	tn38	tn40	tn42	tn22	tn44	tn14
3	0.3000000	1.00000000	0.3000000	-0.5000000	-1.0000000	0.3000000
6	-0.3000000	0.30000000	-0.3000000	1.0000000	0.0000000	0.7000000
7	-0.7000000	-0.30000000	-1.0000000	0.4000000	-0.7000000	0.3000000
8	0.0000000	-0.30000000	-1.7000000	0.3000000	1.0000000	0.7000000
9	-2.3000000	-0.30000000	-0.7000000	0.3000000	-2.0000000	-0.3000000
11	0.3000000	0.70000000	1.0000000	-0.3000000	-1.3000000	1.3000000
13	-1.3000000	-1.00000000	-1.7000000	0.9000000	0.0000000	1.3000000
15	-0.3000000	-1.00000000	-1.7000000	0.3000000	0.0000000	1.0000000
16	-0.7000000	0.30000000	0.0000000	-0.7000000	0.0000000	0.3000000
19	-0.7000000	0.00000000	-0.7000000	0.0000000	1.0000000	1.3000000
22	-0.7000000	-0.70000000	-1.3000000	0.2000000	1.0000000	1.0000000
24	0.7000000	0.70000000	-1.0000000	0.2000000	-0.7000000	-0.3000000
26	0.0000000	-1.00000000	-1.0000000	1.5000000	-0.7000000	1.7000000
27	-0.3000000	1.00000000	-0.3000000	-0.6000000	-0.3000000	0.7000000
28	1.3000000	1.30000000	-0.7000000	-0.1000000	0.0000000	0.7000000
33	-0.3000000	-0.30000000	0.0000000	0.7000000	-0.7000000	0.3000000
36	0.7000000	0.30000000	0.3000000	1.0000000	-1.3000000	1.0000000
40	-0.7000000	0.00000000	-2.0000000	1.0000000	1.0000000	0.3000000

45	-0.7000000	0.00000000	-0.3000000	0.4000000	-0.7000000	1.0000000
46	-0.7000000	-0.70000000	-0.3000000	1.8000000	-0.7000000	1.0000000
47	0.7000000	0.00000000	-0.3000000	-0.1000000	0.0000000	1.7000000
48	0.0000000	0.30000000	0.3000000	-0.7000000	-1.0000000	1.0000000
49	1.3000000	1.00000000	2.0000000	0.2000000	2.3000000	0.7000000
50	0.0000000	0.30000000	0.3000000	0.7000000	-1.3000000	1.3000000
51	0.0000000	0.00000000	-0.7000000	0.6000000	0.7000000	0.3000000
52	-0.3000000	-1.00000000	-1.7000000	0.6000000	0.0000000	1.0000000
53	0.3000000	-0.30000000	-1.0000000	1.2000000	0.3000000	0.7000000
54	0.0000000	0.00000000	0.0000000	1.0000000	-0.3000000	2.0000000
55	-0.7000000	-1.70000000	-1.0000000	1.6000000	-1.0000000	0.7000000
56	1.0000000	0.00000000	-0.3000000	0.6331074	2.3000000	0.7000000
57	0.7000000	-0.70000000	-0.3000000	1.5000000	0.3000000	0.0000000
65	2.0000000	1.70000000	-0.3000000	1.2000000	1.3000000	1.0000000
68	0.7000000	-1.70000000	0.3000000	0.9000000	1.7000000	1.3000000
75	-0.3000000	-1.00000000	-1.7000000	1.6000000	0.0000000	0.3000000
80	0.0000000	0.00000000	0.0000000	0.3000000	-0.7000000	1.3000000
85	-0.3000000	-0.30000000	-1.3000000	0.6000000	-0.3000000	0.3000000
95	-1.0000000	-0.30000000	-2.0000000	0.6000000	0.7000000	0.7000000
101	-0.3000000	0.00000000	-0.7000000	1.8000000	0.0000000	0.7000000
103	-0.3000000	-1.70000000	-1.3000000	0.3000000	-2.0000000	1.0000000
108	-1.0000000	0.30000000	-1.3000000	-0.1000000	0.7000000	-0.3000000
113	2.0000000	1.30000000	2.3000000	1.3000000	0.3000000	2.7000000
115	0.7000000	1.00000000	0.3000000	1.2000000	0.0000000	1.0000000
117	-1.0000000	0.00000000	-1.3000000	0.8000000	-0.3000000	0.3000000
118	-0.3000000	0.30000000	-0.3000000	1.7000000	1.3000000	1.3000000
124	1.0000000	0.30000000	-0.3000000	0.6000000	0.3000000	0.7000000
128	1.3000000	0.30000000	0.0000000	-0.4000000	-1.0000000	2.3000000
131	-0.3000000	-0.30000000	-0.3000000	0.8000000	-0.7000000	0.7000000
132	0.7000000	0.30000000	-0.3000000	1.0000000	0.0000000	1.7000000
133	-0.7000000	-0.30000000	-0.7000000	0.3000000	-1.3000000	1.0000000
145	0.3000000	0.00000000	-0.3000000	0.3000000	0.7000000	0.0000000
150	-0.7000000	-0.30000000	-1.7000000	1.0000000	1.0000000	1.0000000
151	0.0000000	0.00000000	0.3000000	1.3000000	-0.7000000	0.7000000
153	-2.0000000	-1.00000000	-1.3000000	0.4000000	-0.3000000	0.3000000
155	1.3000000	0.00000000	-1.0000000	1.0000000	-0.7000000	1.3000000
163	0.7000000	1.00000000	-0.3000000	1.3000000	0.0000000	0.7000000
167	-1.3000000	-1.00000000	-1.3000000	0.5000000	-0.3000000	1.7000000
171	-0.7000000	-0.70000000	-0.7000000	0.9000000	0.3000000	0.3000000
172	-0.7000000	-0.30000000	-1.0000000	0.2700000	1.0000000	0.7000000
173	-1.3000000	-1.30000000	-0.7000000	1.6000000	0.0000000	1.0000000
178	-0.7000000	-0.30000000	0.0000000	0.6000000	-0.3000000	1.3000000
180	0.0000000	0.00000000	-1.3000000	0.5000000	-0.3000000	2.3000000
181	0.0000000	0.30000000	-1.7000000	-0.4000000	-2.0000000	0.7000000
182	0.3000000	0.30000000	-1.3000000	0.7000000	-0.7000000	1.0000000
183	2.7000000	2.70000000	-1.0000000	0.8000000	0.0000000	1.3000000
184	0.0000000	0.00000000	0.7000000	-0.4000000	1.0000000	2.0000000
185	0.7000000	0.30000000	0.0000000	-0.7000000	0.0000000	0.0000000
186	0.3000000	0.70000000	0.3000000	0.2000000	0.0000000	1.3000000
187	-0.7000000	-0.70000000	-1.7000000	2.0000000	-0.3000000	0.3000000
189	-0.7000000	-0.70000000	-0.7000000	1.6000000	0.7000000	1.7000000
190	0.0000000	0.30000000	0.3000000	0.2000000	0.0000000	0.7000000
191	2.0000000	1.30000000	0.7000000	-1.6000000	-1.3000000	2.0000000
192	1.3000000	1.30000000	-1.7000000	0.4000000	-0.7000000	0.7000000

193	1.0000000	1.70000000	-0.3000000	1.2000000	-0.3000000	0.7000000
195	1.0000000	0.70000000	0.3000000	0.5000000	0.0000000	0.0000000
196	0.7000000	1.00000000	-1.0000000	0.6000000	1.0000000	1.7000000
197	0.7000000	1.00000000	0.0000000	0.0000000	1.0000000	1.7000000
199	1.3000000	1.00000000	-0.7000000	0.2000000	0.0000000	0.7000000
200	0.0000000	-1.00000000	-0.7000000	0.8000000	0.0000000	0.7000000
201	1.7000000	0.70000000	0.3000000	0.8000000	0.3000000	1.0000000
202	0.0000000	-0.70000000	-1.0000000	1.1000000	0.7000000	0.3000000
203	1.0000000	-0.30000000	0.3000000	1.6000000	0.3000000	1.0000000
204	0.0000000	0.30000000	0.3000000	1.8000000	0.0000000	0.7000000
205	0.7000000	0.00000000	0.0000000	1.4000000	-0.3000000	1.3000000
206	-0.3000000	-0.70000000	0.3000000	1.8000000	0.0000000	0.7000000
207	-0.3000000	0.70000000	-0.7000000	-0.4000000	-0.3000000	-0.3000000
208	0.0000000	0.00000000	0.0000000	0.1000000	1.0000000	1.0000000
209	0.7000000	1.00000000	1.0000000	1.4000000	-0.7000000	1.7000000
210	0.3000000	0.30000000	0.3000000	2.0000000	-2.0000000	1.0000000
211	-0.7000000	1.00000000	0.7000000	0.4000000	-0.3000000	0.3000000
212	2.3000000	2.30000000	2.3000000	1.0000000	2.3000000	1.7000000
213	1.0000000	0.70000000	-0.7000000	0.8000000	-2.0000000	1.7000000
214	2.0000000	1.30000000	0.0000000	0.6000000	-0.3000000	0.3000000
215	-0.3000000	-0.30000000	-1.7000000	0.2000000	0.7000000	1.3000000
216	-1.3000000	-2.00000000	-1.3000000	0.8000000	0.7000000	1.0000000
217	1.0000000	1.30000000	0.3000000	1.4000000	0.7000000	0.7000000
218	0.7000000	1.00000000	0.0000000	0.8000000	-2.0000000	0.0000000
219	-0.4843276	0.36919106	-0.3884471	0.6442551	-0.2315443	0.5955351
220	1.3000000	0.30000000	0.3000000	2.6000000	1.3000000	1.0000000
222	0.3000000	-0.30000000	0.0000000	0.1000000	-0.7000000	0.0000000
223	-1.0000000	-1.00000000	0.3000000	1.2000000	0.7000000	1.7000000
224	2.0000000	1.30000000	1.7000000	3.0000000	-0.3000000	1.3000000
225	0.0000000	0.70000000	-0.3000000	0.4000000	-0.3000000	1.0000000
226	0.7000000	1.00000000	0.7000000	1.2000000	-0.7000000	1.0000000
228	-0.2380424	-0.29804675	-0.3265751	0.3526449	-0.2024081	1.2087006
229	0.7000000	0.30000000	0.0000000	0.6000000	-0.3000000	1.3000000
230	-0.3000000	-0.30000000	0.0000000	0.8000000	0.7000000	-0.3000000
231	0.0000000	0.00000000	0.0000000	0.8000000	0.0000000	0.7000000
234	-0.7000000	-1.70000000	-1.3000000	0.6000000	0.0000000	1.0000000
235	-1.0000000	0.70000000	1.7000000	-0.2000000	-0.3000000	1.3000000
236	0.0000000	1.30000000	1.7000000	-0.8000000	1.3000000	0.7000000
237	-0.3000000	-0.30000000	-0.3000000	0.6000000	0.3000000	1.0000000
238	1.3000000	0.70000000	1.3000000	-0.6000000	1.7000000	1.0000000
239	0.0000000	0.70000000	0.0000000	1.8000000	1.7000000	0.7000000
241	-0.2685230	-0.03704518	-0.5917289	0.9404058	-0.4714624	1.0037005

	tn24	tn12	tn6	tn30	tn34	tn8	tn48
3	-1.00000000	1.3000000	1.7000000	0.3200000	-1.0000000	0.3000000	3.100000
6	-0.70000000	0.0000000	1.3800000	1.7100000	0.0000000	2.7000000	1.300000
7	0.40000000	1.0000000	-1.1800000	-1.2200000	-0.3000000	0.0000000	1.300000
8	-0.70000000	0.0000000	-0.2000000	0.0300000	-1.0000000	-0.3000000	3.100000
9	-2.70000000	0.7000000	-0.0100000	-0.0700000	1.0000000	-0.3000000	1.000000
11	-0.30000000	1.0000000	0.4200000	-0.5900000	1.0000000	0.3000000	1.300000
13	-0.30000000	1.0000000	1.2600000	1.7100000	-0.7000000	2.7000000	0.800000
15	-1.00000000	0.3000000	1.0300000	1.3600000	0.0000000	-0.7000000	3.100000
16	1.00000000	0.0000000	-0.8100000	0.6500000	-0.7000000	-0.3000000	0.800000
19	-1.00000000	1.0000000	-0.3200000	-1.2100000	-0.3000000	-0.7000000	1.300000
22	-0.30000000	1.3000000	0.5900000	0.4000000	-0.3000000	0.0000000	3.000000

24	-1.30000000	1.0000000	0.9100000	1.0000000	0.0000000	-1.0000000	3.000000
26	0.30000000	0.7000000	-0.1500000	-0.1300000	0.0000000	0.3000000	3.100000
27	-1.30000000	1.0000000	0.1700000	0.3400000	-0.3000000	0.3000000	3.100000
28	0.00000000	0.0000000	0.9200000	0.2800000	0.3000000	2.7000000	3.100000
33	-1.00000000	0.7000000	0.2200000	1.0000000	0.0000000	-0.3000000	1.300000
36	0.00000000	1.0000000	0.4600000	1.7300000	-0.7000000	0.0000000	1.300000
40	-0.30000000	0.0000000	-1.0500000	-0.7800000	-0.7000000	-1.0000000	3.000000
45	-1.70000000	-0.3000000	-0.4300000	1.0300000	0.0000000	2.7000000	1.300000
46	-0.30000000	1.0000000	0.9800000	0.0600000	-0.7000000	-0.3000000	3.000000
47	-2.70000000	2.0000000	1.2500000	0.7300000	-1.0000000	-0.7000000	1.000000
48	-1.00000000	-0.7000000	1.5300000	0.9600000	-0.7000000	-1.3000000	3.000000
49	0.70000000	0.0000000	1.1600000	0.3400000	-0.7000000	-0.7000000	1.300000
50	1.70000000	1.7000000	0.7900000	0.3400000	0.3000000	-0.7000000	3.000000
51	-0.30000000	1.0000000	-0.0700000	-1.2100000	-0.7000000	-1.3000000	1.300000
52	0.00000000	0.7000000	0.9100000	0.2800000	0.0000000	-0.3000000	1.300000
53	1.30000000	1.0000000	0.9800000	0.4000000	-1.0000000	-0.7000000	0.800000
54	0.30000000	1.7000000	-0.6700000	0.3200000	-1.3000000	-1.0000000	1.000000
55	2.00000000	-0.3000000	-0.5900000	-0.7800000	0.0000000	0.3000000	1.000000
56	0.70000000	1.3000000	1.9800000	1.3900000	2.3000000	2.3000000	3.100000
57	-1.30000000	0.3000000	0.5900000	1.4000000	0.7000000	2.0000000	3.100000
65	0.00000000	1.0000000	-0.0500000	-0.5900000	-1.0000000	0.0000000	1.300000
68	0.00000000	1.7000000	-0.5600000	-0.9000000	1.3000000	1.3000000	3.100000
75	-0.70000000	0.0000000	-1.5200000	-0.1300000	1.0000000	0.3000000	0.800000
80	1.00000000	1.3000000	0.9200000	1.3500000	-1.0000000	-1.0000000	3.100000
85	-1.30000000	1.0000000	0.8500000	1.7300000	0.0000000	-0.3000000	3.100000
95	-1.30000000	0.0000000	0.8900000	1.0300000	0.3000000	0.0000000	1.300000
101	1.80000000	0.7000000	-0.8100000	-1.2200000	0.3000000	0.3000000	3.100000
103	-0.30000000	-0.3000000	1.5000000	1.3500000	0.7000000	0.3000000	1.000000
108	-0.70000000	-0.3000000	0.5300000	0.3200000	1.0000000	0.3000000	3.000000
113	-0.30000000	0.7000000	2.2700000	1.2800000	0.3000000	0.3000000	3.100000
115	0.00000000	0.7000000	-0.7100000	-0.0700000	-1.3000000	-0.7000000	1.300000
117	-2.70000000	-0.7000000	0.1000000	0.2800000	-0.7000000	-1.0000000	0.700000
118	0.70000000	1.3000000	1.0300000	1.5900000	0.3000000	-0.3000000	1.000000
124	0.00000000	0.3000000	-1.0600000	-0.2800000	-0.7000000	-0.7000000	1.000000
128	0.30000000	1.7000000	0.6900000	1.3500000	0.7000000	0.0000000	3.100000
131	0.00000000	0.7000000	-0.3600000	-0.4200000	2.7000000	0.7000000	3.000000
132	0.30000000	1.7000000	0.0400000	0.0300000	-0.7000000	-0.3000000	1.300000
133	0.30000000	1.0000000	-0.3200000	0.6500000	0.0000000	-0.3000000	0.600000
145	-2.30000000	0.3000000	1.9000000	0.6500000	-0.7000000	0.0000000	1.000000
150	-1.30000000	-0.3000000	0.1000000	0.6400000	0.0000000	-0.3000000	3.000000
151	0.30000000	0.3000000	1.3800000	1.3500000	0.7000000	-0.7000000	1.300000
153	-1.00000000	-0.3000000	1.0300000	1.3500000	0.0000000	2.3000000	3.000000
155	0.00000000	1.7000000	1.2800000	1.5900000	-0.3000000	-0.7000000	1.300000
163	0.00000000	0.7000000	-0.5900000	0.2800000	0.0000000	0.0000000	3.000000
167	-1.70000000	1.3000000	1.0100000	0.3200000	-1.0000000	0.3000000	1.300000
171	-2.30000000	1.0000000	1.2500000	1.0300000	-0.3000000	-1.0000000	3.000000
172	-1.00000000	0.3000000	-0.8300000	1.3500000	0.0000000	-1.3000000	3.000000
173	-1.00000000	1.0000000	1.0300000	0.9600000	-1.3000000	2.0000000	0.300000
178	-0.30000000	1.7000000	1.7300000	1.3900000	0.0000000	-1.0000000	1.300000
180	-0.30000000	0.5000000	0.9400000	1.1700000	-1.0000000	2.3000000	1.300000
181	1.00000000	0.7000000	0.5600000	1.3500000	-1.0000000	0.3000000	3.000000
182	-0.30000000	1.0000000	0.1000000	0.6400000	-0.3000000	0.3000000	1.000000
183	0.00000000	1.0000000	-0.1500000	1.6000000	0.7000000	-0.3000000	3.100000
184	1.00000000	1.7000000	2.1900000	1.7100000	0.7000000	-1.0000000	3.100000

```
185 -1.30000000 -0.3000000 -0.3100000 -0.0600000 -0.3000000 -0.7000000 1.000000
186 -0.70000000  1.3000000  0.1700000  0.9700000 -0.3000000 -0.3000000 3.100000
187 -0.30000000  1.3000000  0.8900000  1.0300000 -0.7000000  0.0000000 1.300000
189 -0.30000000  1.3000000  0.7200000  1.4000000 -1.7000000 -1.7000000 0.500000
190  0.70000000  0.3000000 -0.1900000 -0.5900000  0.3000000  0.0000000 1.300000
191  2.00000000  2.3000000 -0.8300000  0.4000000 -0.7000000  0.7000000 3.100000
192 -0.70000000  0.7000000  2.1400000  0.9600000  0.3000000 -0.7000000 0.800000
193  0.30000000  1.0000000 -1.2400000  0.7300000 -1.3000000 -1.3000000 1.300000
195  0.30000000  0.3000000 -0.1800000  1.0600000 -1.0000000  0.0000000 1.000000
196  0.00000000  1.7000000  0.9100000  1.2800000 -0.7000000 -1.3000000 0.800000
197  0.30000000  2.3000000  2.2500000  1.6600000 -0.3000000 -1.3000000 3.000000
199 -1.30000000  0.0000000  0.5700000  1.3500000  0.0000000 -1.3000000 1.000000
200  0.00000000  1.0000000  1.2500000  0.7300000  0.0000000  1.0000000 1.300000
201  0.30000000  0.3000000  1.4000000  0.9700000  0.7000000 -0.7000000 1.000000
202 -0.30000000  0.7000000  1.2800000  0.9700000 -0.7000000 -0.7000000 3.100000
203 -1.30000000  0.7000000  1.6500000  0.0300000  0.3000000 -0.7000000 1.000000
204  0.70000000  1.3000000  1.3700000  1.3900000  0.3000000  0.0000000 1.300000
205 -0.70000000  2.3000000 -0.0700000  0.9700000 -0.7000000 -0.3000000 1.000000
206 -0.70000000  0.3000000 -0.1900000  1.0400000  0.7000000  0.7000000 0.800000
207 -1.30000000  0.7000000  1.2600000  1.7100000 -0.3000000  0.3000000 3.100000
208 -0.30000000  1.0000000  1.2400000  1.4000000  0.0000000  0.3000000 1.300000
209  0.30000000  1.3000000  1.3800000  1.7100000  0.0000000 -1.0000000 1.300000
210  1.00000000  1.3000000 -1.2000000 -0.7800000 -0.7000000  0.0000000 1.300000
211  0.00000000  1.0000000 -0.0100000  0.2800000  1.3000000  0.0000000 3.100000
212  1.70000000  1.7000000  1.5300000  1.2800000  0.3000000  0.0000000 3.100000
213 -0.30000000  1.0000000 -1.0500000  0.2900000  0.3000000  2.7000000 1.300000
214 -0.30000000  1.0000000  0.1700000  0.9700000 -0.3000000  0.0000000 3.100000
215 -1.30000000  0.3000000  1.6100000  1.7100000  1.0000000  2.3000000 3.100000
216  1.00000000  1.3000000 -0.0700000  0.6700000  0.0000000 -1.0000000 1.000000
217  1.70000000  1.3000000  1.1600000  0.9600000  0.0000000 -0.3000000 3.000000
218 -0.70000000  0.0000000  1.5300000  0.6500000 -1.3000000 -0.3000000 3.000000
219 -0.41269192  0.8447977  0.3163209  0.9025117 -0.3208913  0.3926552 1.753575
220  2.00000000  1.3000000 -0.6900000 -0.9000000  0.0000000 -0.3000000 1.000000
222 -0.70000000  0.0000000 -0.4400000  0.0300000 -0.7000000  0.0000000 3.100000
223  0.70000000  1.3000000  2.0200000  1.5900000  0.0000000  2.0000000 1.300000
224  0.00000000  1.0000000  1.0300000  1.0000000  0.0000000 -1.0000000 1.300000
225  0.30000000  1.7000000  1.1300000  1.3900000  1.7000000  2.3000000 3.100000
226  0.00000000  1.3000000  0.6600000  0.9700000 -1.0000000 -0.7000000 3.100000
228 -0.11222345  1.2872673  1.0044066  0.8061191 -0.1814728 -0.3374685 1.937584
229 -1.00000000  1.0000000  0.6500000  1.0300000  1.0000000 -0.7000000 3.100000
230 -1.00000000  1.0000000  1.3800000  0.6400000  0.3000000 -0.7000000 3.100000
231  0.00000000  0.3000000 -0.7000000  0.2900000 -1.3000000 -1.3000000 3.100000
234  0.00000000  0.7000000  0.7700000  1.0300000 -0.3000000 -0.3000000 0.800000
235  0.30000000  1.3000000  1.1300000  1.0300000 -0.7000000  0.3000000 1.300000
236  0.70000000  0.0000000  0.3000000 -0.2800000  0.3000000 -0.3000000 1.300000
237  1.30000000  1.7000000  0.9800000  0.4000000 -1.3000000  0.7000000 1.300000
238  1.30000000  1.0000000  1.2400000  0.4000000  0.7000000  0.7000000 1.300000
239  0.70000000  0.0000000 -1.0500000 -0.7800000  0.3000000 -1.0000000 3.100000
241 -0.09916293  1.0844352  0.2887386  0.6956717 -0.1530272  0.3005667 2.201543
          tn50 Target
3   -0.3000000      0
6    0.7000000      1
7    1.3000000      1
8    0.7000000      1
```

9	0.3000000	1
11	0.3000000	1
13	0.0000000	1
15	0.3000000	1
16	0.7000000	1
19	0.7000000	0
22	-0.3000000	1
24	0.7000000	1
26	-0.3000000	1
27	0.3000000	1
28	1.3000000	1
33	0.0000000	1
36	0.7000000	1
40	0.3000000	1
45	0.0000000	1
46	0.3000000	0
47	1.3000000	0
48	0.3000000	0
49	0.3000000	0
50	1.0000000	1
51	0.0000000	0
52	0.0000000	0
53	-0.7000000	0
54	0.3000000	0
55	-1.0000000	0
56	3.0000000	0
57	1.0000000	1
65	0.7000000	1
68	0.7000000	1
75	-0.3000000	1
80	0.3000000	1
85	1.0000000	1
95	-0.3000000	1
101	0.3000000	1
103	0.7000000	1
108	0.3000000	1
113	1.3000000	1
115	0.7000000	1
117	0.3000000	1
118	0.0000000	1
124	0.3000000	1
128	1.0000000	0
131	0.7000000	1
132	0.3000000	1
133	0.0000000	1
145	1.3000000	1
150	0.0000000	1
151	-0.3000000	1
153	1.0000000	1
155	0.3000000	1
163	0.7000000	1
167	0.0000000	1
171	0.0000000	1
172	0.3000000	0

173	0.3000000	0
178	1.0000000	0
180	1.0000000	0
181	0.7000000	0
182	0.7000000	0
183	1.0000000	0
184	0.7000000	0
185	0.7000000	0
186	0.7000000	0
187	0.3000000	0
189	0.3000000	0
190	0.7000000	0
191	1.7000000	0
192	-0.3000000	0
193	0.0000000	0
195	0.0000000	0
196	0.3000000	0
197	2.0000000	0
199	0.3000000	1
200	1.3000000	0
201	1.0000000	0
202	1.7000000	1
203	0.3000000	0
204	0.0000000	0
205	0.3000000	1
206	-0.3000000	0
207	0.7000000	0
208	1.0000000	0
209	0.7000000	0
210	0.3000000	0
211	0.7000000	0
212	1.3000000	0
213	-0.3000000	0
214	0.0000000	0
215	-0.7000000	0
216	0.7000000	0
217	0.7000000	0
218	0.3000000	1
219	0.4238885	0
220	1.0000000	0
222	0.7000000	0
223	0.7000000	0
224	0.7000000	0
225	-0.3000000	0
226	1.7000000	0
228	0.5523615	0
229	0.3000000	0
230	1.0000000	0
231	0.7000000	1
234	1.0000000	1
235	1.0000000	0
236	0.7000000	1
237	0.0000000	0
238	1.0000000	0

```
239  0.3000000      0
241  0.1655717      0
```

```r
#Agregamos la asignación de clústeres como columna en los datos originales
numeric_vars_def$cluster <- clusters

#Visualizamos los datos con la columna de clúster
print("Datos originales con asignación de clúster:")
```

```
[1] "Datos originales con asignación de clúster:"
```

```r
print(head(numeric_vars_def))
```

```
    ag ptg19 sex_Woman el_Elementary el_High.School el_Secondary
1 26.9 38.23         1             0              1            0
2 60.5 26.34         0             0              0            0
3 33.1 18.24         1             0              0            0
4 44.6 24.14         1             0              0            0
5 61.9 32.80         0             0              0            0
6 43.0 32.70         1             0              0            0
  el_Specialist.Master el_University.Deg. spec1_Hospitalization
1                    0                  0                     0
2                    1                  0                     1
3                    0                  0                     0
4                    0                  1                     0
5                    1                  0                     1
6                    1                  0                     0
  spec1_Mild.Moderated spec1_UCI tn46 tn52 tn36 tn38 tn40 tn42 tn22 tn44 tn14
1                    1         0  0.3  1.0 -1.0 -0.3  0.0 -1.3  0.4 -0.7 -1.7
2                    0         0  1.0  0.0  0.0  0.3  0.0  0.0  0.4 -1.3  1.0
3                    1         0  0.7  2.0  1.0  0.3  1.0  0.3 -0.5 -1.0  0.3
4                    1         0  0.0 -0.6  0.3  0.0  1.0  1.3 -0.9  1.0 -0.3
5                    0         0  0.0  0.0  0.3 -0.3  0.3 -0.3 -0.2 -0.3  1.3
6                    1         0  0.3  0.4 -0.7 -0.3  0.3 -0.3  1.0  0.0  0.7
  tn24 tn12   tn6  tn30 tn34  tn8 tn48 tn50 Target cluster
1 -3.0 -2.0 -2.06 -2.30 -1.0  2.0  1.0 -1.7      1       1
2  0.0  0.0  0.86  0.70  0.3 -0.3  3.1  1.3      1       1
3 -1.0  1.3  1.70  0.32 -1.0  0.3  3.1 -0.3      0       2
4 -0.7 -0.3 -0.82 -0.43 -1.0 -0.7  1.3 -0.3      1       1
5 -0.3  1.3 -0.57  0.40  0.3 -0.3  3.1 -0.7      1       1
6 -0.7  0.0  1.38  1.71  0.0  2.7  1.3  0.7      1       2
```

```r
#Resumen por clúster
cluster_summary <- aggregate(. ~ cluster, data = numeric_vars_def, mean)
print("Resumen por clúster:")
```

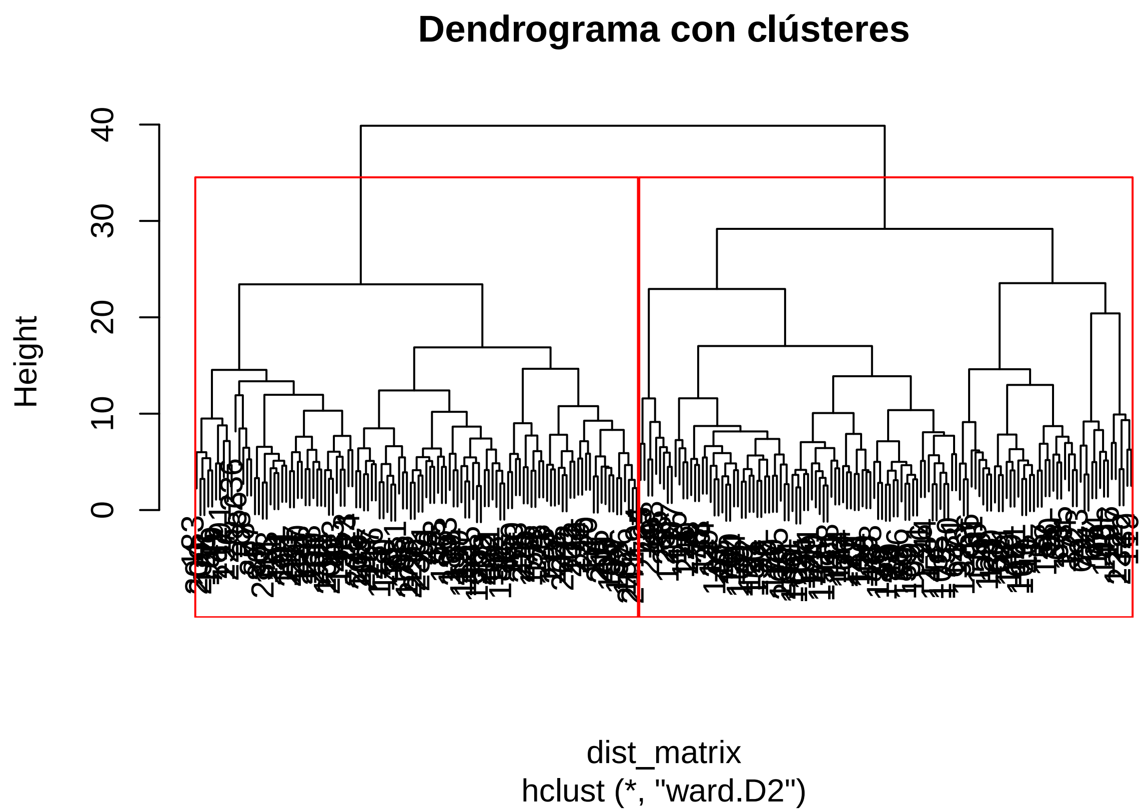
```
[1] "Resumen por clúster:"
```

```r
print(cluster_summary)
```

```
  cluster       ag    ptg19 sex_Woman el_Elementary el_High.School el_Secondary
1       1 49.16929 28.42748 0.7716535    0.07874016      0.4094488   0.07086614
2       2 48.21404 26.79841 0.8333333    0.00000000      0.2368421   0.00000000
  el_Specialist.Master el_University.Deg. spec1_Hospitalization
1           0.03937008          0.3937008             0.2677165
2           0.27192982          0.3596491             0.0000000
  spec1_Mild.Moderated  spec1_UCI       tn46        tn52        tn36       tn38
```

```
1            0.6692913 0.04724409 -0.3299213 -0.02440945 -0.50448819 -0.3881890
2            0.9385965 0.00000000  0.1521071  0.07521009  0.03924876  0.1360448
        tn40       tn42       tn22        tn44         tn14       tn24       tn12
1 -0.5763780 -1.0795276 0.1574803 -0.67165354 -0.02519685 -0.7598425 -0.2503937
2  0.1239833 -0.3298838 0.6749159 -0.01759136  0.89919242 -0.1791586  0.8247061
         tn6       tn30       tn34         tn8      tn48        tn50    Target
1 -0.8196063 -0.4996850 -0.5622047 -0.2188976 1.319685 -0.01732283 0.9685039
2  0.5199076  0.6482834 -0.1250473 -0.0038969 1.953445  0.50826159 0.4473684
```

```
#Agregamos rectángulos al dendrograma para visualizar los clústeres
plot(hclust_model, main = "Dendrograma con clústeres")
rect.hclust(hclust_model, k = 2, border = "red")
```



La organización por clústeres no parece ser de utilidad inicial.

Estadística descriptiva e inferial básica.

Vamos a seleccionar las variables que son factores (reales) y las vamos a contrastar con chi-cuadrado

```
#Creamos un dataset nuevo para evitar trabajar sobre el original y aquí hacemos las pruebas.
df_chi <- df_def_clean %>%
  dplyr::select(where(~ is.numeric(.) && all(. %in% c(0, 1)))) %>%
  mutate(Target = as.factor(df_def_clean$Target))  #Agregar Target como factor
```

```
# ---------------------------------------------------------------------------
# FILTRADO POR FRECUENCIA Y PRUEBAS DE CHI-CUADRADO
# ---------------------------------------------------------------------------

# 1. Calcular las frecuencias mínimas para cada variable dummy
low_freq_vars <- sapply(names(df_chi), function(var) {
```

```r
  freq_table <- table(df_chi[[var]])  # Calcular la tabla de frecuencias
  min(freq_table)  # Obtener la frecuencia mínima para cada variable
})

# 2. Definir el umbral de frecuencia mínima
threshold <- 5  # Se puede cambiar el umbral si es necesario

# 3. Seleccionar variables con frecuencias mínimas mayores o iguales al umbral
valid_vars <- names(low_freq_vars[low_freq_vars >= threshold])

# 4. Filtrar el dataset para mantener solo las variables con frecuencias aceptables
df_chi_filtered <- df_chi %>%
  dplyr::select(all_of(valid_vars))  # Seleccionar las variables válidas

# 5. Realizar pruebas de chi-cuadrado para las variables filtradas contra Target
chi_results_filtered <- sapply(names(df_chi_filtered), function(var) {
  table_var <- table(df_chi_filtered[[var]], df_def_clean$Target)  # Crear tabla de contingencia
  chi_test <- chisq.test(table_var)  # Realizar prueba de chi-cuadrado
  return(chi_test$p.value)  #Extraer el valor p
})

# 6. Convertir los resultados a dataframe
chi_results_filtered_df <- data.frame(
  Variable = names(chi_results_filtered),  # Nombre de las variables
  P_Value = chi_results_filtered  # Valores p
)

# 7. Ordenar los resultados por valor p
chi_results_filtered_df <- chi_results_filtered_df %>%
  arrange(P_Value)

# ----------------------------------------------------------------------------
# RESULTADOS Y VISUALIZACIÓN
# ----------------------------------------------------------------------------

#Mostramos el resumen de los resultados
print(chi_results_filtered_df)
```

```
                                    Variable      P_Value
Target                                Target 2.866752e-53
spec1_Hospitalization spec1_Hospitalization 2.176275e-04
el_High.School                el_High.School 4.464665e-04
el_Specialist.Master   el_Specialist.Master 6.144517e-04
spec1_Mild.Moderated   spec1_Mild.Moderated 2.872846e-02
el_University.Deg.       el_University.Deg. 2.598910e-01
spec1_UCI                          spec1_UCI 2.810929e-01
el_Elementary                  el_Elementary 3.560693e-01
sex_Woman                          sex_Woman 9.553274e-01
el_Secondary                    el_Secondary 9.987447e-01
```

```r
#Guardamos los resultados en un archivo CSV
# write.csv(chi_results_filtered_df, "chi_squared_results_filtered.csv", row.names = FALSE)

#Filtramos variables significativas (p < 0.05)
```

```
significant_vars_chi <- chi_results_filtered_df %>%
  filter(P_Value < 0.05)

#Mostramos las variables significativas
print(significant_vars_chi)
```

```
                                     Variable      P_Value
Target                                 Target 2.866752e-53
spec1_Hospitalization spec1_Hospitalization 2.176275e-04
el_High.School               el_High.School 4.464665e-04
el_Specialist.Master   el_Specialist.Master 6.144517e-04
spec1_Mild.Moderated   spec1_Mild.Moderated 2.872846e-02
```

Sobre Chi-cuadrado:

Un p-valor bajo significa que la distribución de las categorías de la variable explicativa difiere significativamente entre las categorías de Target. Es decir, las frecuencias de las categorías no son independientes entre sí, lo que sugiere una relación entre la variable explicativa y Target.

Por ejemplo, si spec1_Hospitalization tiene un p-valor bajo, esto implica que las frecuencias de hospitalización están distribuidas de manera diferente según la categoría de Target (No_LC vs. LC_Cog) (lo que podría ser relevante para el modelo). En este caso también habría que tener en cuenta cómo podría afectar el desbalance de las clases que se observa en el EDA.

Análisis de variables cuantitativas.

Hay diferentes opciones y modelos, sin embargo, las opciones paramétricas no serán fácilmente aplicables porque los supuestos no se terminan de cumplir en las variables. En consecuencia, aunque completemos el modelado, su aplicación no es correcta. Uno de dichos modelos es la regresión logística.

```
#Seleccionamos únicamente las variables numéricas continuas (excluyendo las dummy)
numeric_vars_log <- df_def_clean %>%
  dplyr::select(where(is.numeric)) %>%   #Seleccionar variables numéricas
  dplyr::select_if(~ length(unique(.)) > 2)  #Excluir las dummies (solo deja aquellas con más de 2 valo

numeric_vars_log$Target <- as.factor(df_def_clean$Target)
```

Siguiendo con el análisis inferencial sobre variables númericas, vamos a usar una comparación no paramétrica, media a media.

```
# Realizamos la prueba de Mann-Whitney U (Wilcoxon rank-sum test) para cada variable continua
mann_results <- sapply(colnames(numeric_vars_log)[-which(colnames(numeric_vars_log) == "Target")], func
  wilcox.test(as.formula(paste(var, "~ Target")), data = numeric_vars_log)$p.value
})

# Pasamos los resultados a un dataframe
mann_results_df <- data.frame(
  Variable = names(mann_results),
  P_Value = mann_results
)

# Ordenamos por valor p
mann_results_df <- mann_results_df %>% arrange(P_Value)

# Mostramos los resultados
print(mann_results_df)
```

```
      Variable      P_Value
tn12      tn12 1.358091e-15
tn14      tn14 7.126898e-11
tn6        tn6 5.048675e-10
tn30      tn30 9.730113e-10
tn42      tn42 2.679294e-08
tn40      tn40 1.876469e-07
tn24      tn24 3.592364e-07
tn38      tn38 2.274921e-06
tn50      tn50 8.197170e-04
tn36      tn36 8.996866e-04
tn44      tn44 1.185235e-03
tn46      tn46 2.355474e-03
tn48      tn48 4.839461e-03
ptg19    ptg19 3.631922e-02
tn22      tn22 3.698245e-02
tn34      tn34 6.040687e-02
tn52      tn52 2.040399e-01
ag          ag 2.761711e-01
tn8        tn8 7.219663e-01
```

```r
# Filtramos variables significativas (p < 0.05)
significant_vars_mann <- mann_results_df %>%
  filter(P_Value < 0.05)

# Mostramos las variables significativas
print(significant_vars_mann)
```

```
      Variable      P_Value
tn12      tn12 1.358091e-15
tn14      tn14 7.126898e-11
tn6        tn6 5.048675e-10
tn30      tn30 9.730113e-10
tn42      tn42 2.679294e-08
tn40      tn40 1.876469e-07
tn24      tn24 3.592364e-07
tn38      tn38 2.274921e-06
tn50      tn50 8.197170e-04
tn36      tn36 8.996866e-04
tn44      tn44 1.185235e-03
tn46      tn46 2.355474e-03
tn48      tn48 4.839461e-03
ptg19    ptg19 3.631922e-02
tn22      tn22 3.698245e-02
```

A partir de aquí ya podemos generar los modelos.

El modelo con más restricciones es el logístico, pero lo podemos generar para tener un valor de comparación con otros modelos más complejos.

MODELO LOGÍSTICO

```r
#Pasos:

# 1. Dividir el dataset en entrenamiento y prueba
set.seed(123) #2435
```

```r
train_idx <- sample(seq_len(nrow(numeric_vars_log)), size = 0.8 * nrow(numeric_vars_log))

train_data_log <- numeric_vars_log[train_idx, ]
test_data_log <- numeric_vars_log[-train_idx, ]

# 2. Ajustar el modelo de regresión logística
logistic_model <- glm(Target ~ ., data = train_data_log, family = binomial)

#Summary
summary(logistic_model)
```

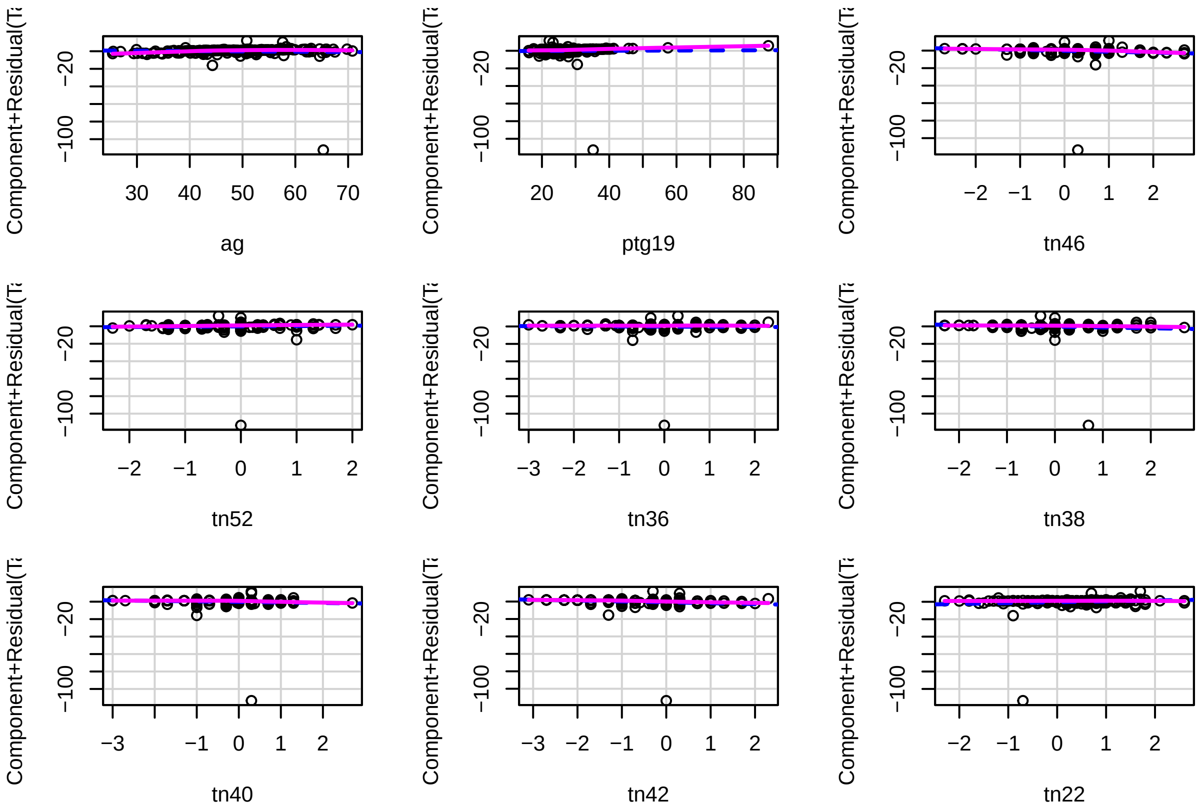
```

Call:
glm(formula = Target ~ ., family = binomial, data = train_data_log)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.20391    1.74810  -1.833   0.0668 .
ag           0.06381    0.02660   2.399   0.0164 *
ptg19        0.07858    0.04398   1.787   0.0740 .
tn46        -0.46675    0.31467  -1.483   0.1380
tn52         0.35693    0.42973   0.831   0.4062
tn36         0.12976    0.41432   0.313   0.7541
tn38        -0.00498    0.48037  -0.010   0.9917
tn40        -0.06963    0.42785  -0.163   0.8707
tn42        -0.40537    0.33627  -1.205   0.2280
tn22         0.09384    0.28107   0.334   0.7385
tn44        -0.36939    0.29955  -1.233   0.2175
tn14         0.30195    0.50426   0.599   0.5493
tn24        -0.89756    0.44689  -2.008   0.0446 *
tn12        -0.99289    0.40170  -2.472   0.0134 *
tn6         -0.66065    0.36061  -1.832   0.0670 .
tn30        -0.35100    0.38085  -0.922   0.3567
tn34         0.07754    0.36338   0.213   0.8310
tn8          0.32202    0.32012   1.006   0.3145
tn48        -0.54894    0.29347  -1.871   0.0614 .
tn50         0.46984    0.44177   1.064   0.2875
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 218.11  on 191  degrees of freedom
Residual deviance: 120.69  on 172  degrees of freedom
AIC: 160.69

Number of Fisher Scoring iterations: 6
```

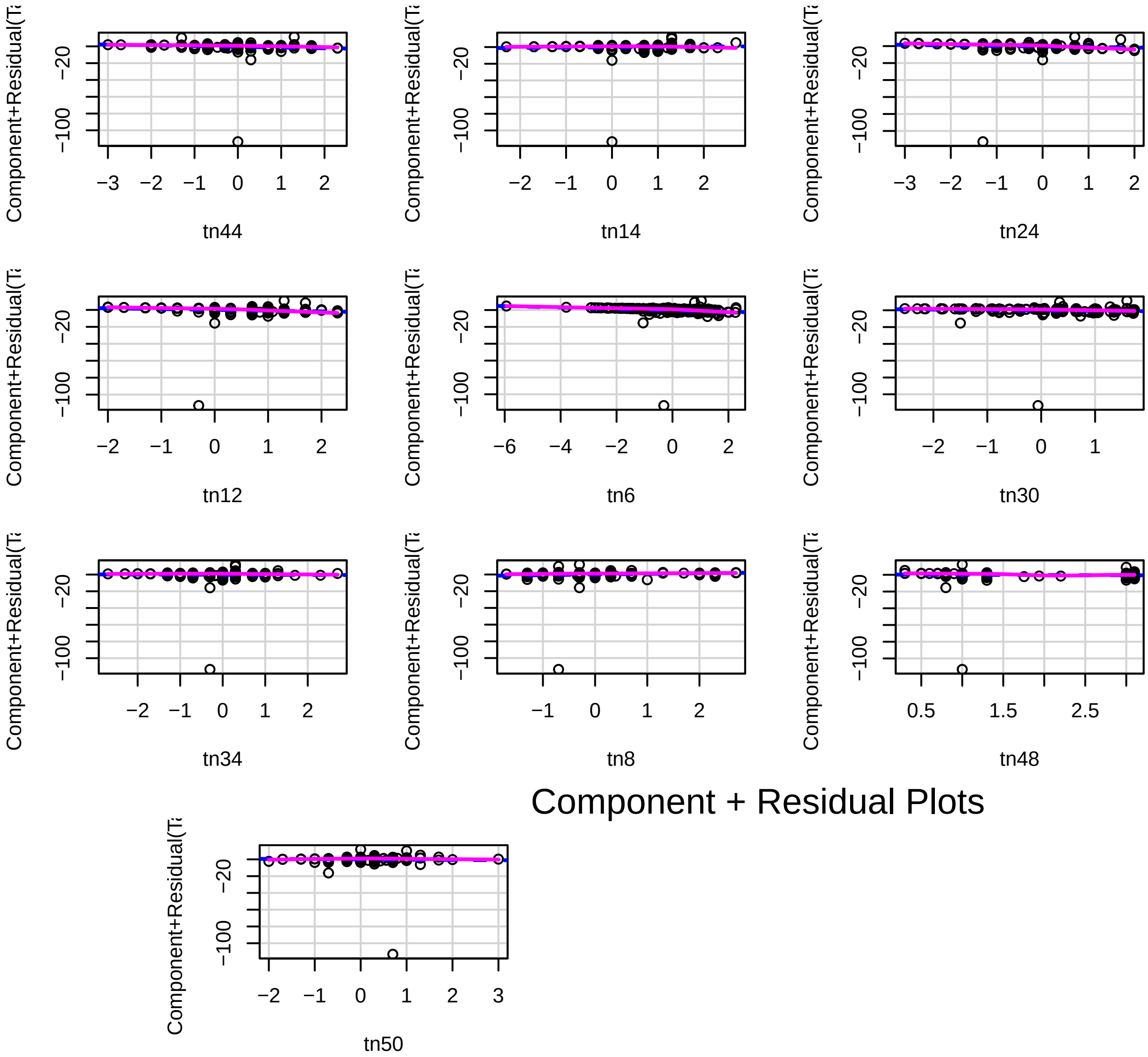
```r
vif_values <- vif(logistic_model)  # Calcular VIF
print(vif_values) #No hay valores VIF por encima de 5
```

```
      ag    ptg19     tn46     tn52     tn36     tn38     tn40     tn42
1.317334 1.249606 1.434816 2.122139 2.730130 3.067105 2.127563 1.550280
    tn22     tn44     tn14     tn24     tn12      tn6     tn30     tn34
```

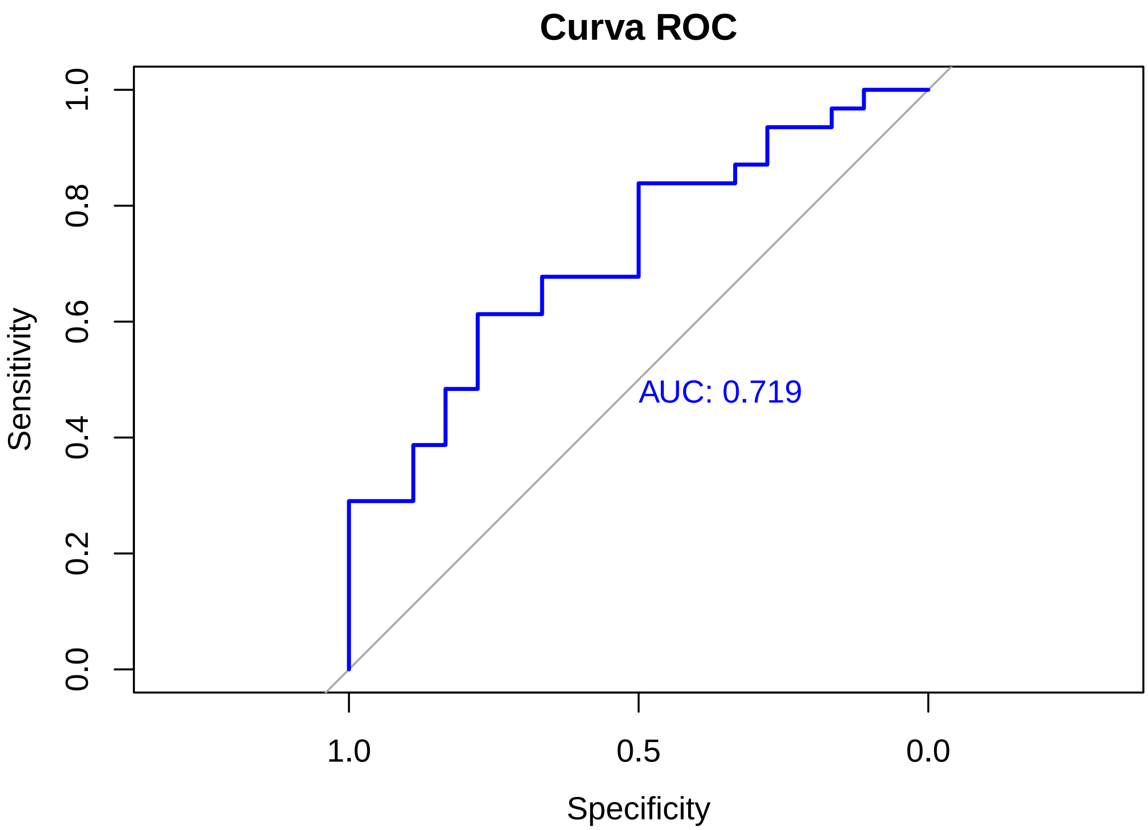
```
1.309288 1.321190 2.593691 2.665532 1.680649 2.222807 2.284425 1.277574
     tn8      tn48      tn50
1.287406 1.569359 1.481415
```

```r
crPlots(logistic_model) #De forma global, las variables son aptas para un modelo logístico. Quizás tn6 p
```






```r
# 3. Obtener predicciones en el conjunto de prueba
predicted_probs <- predict(logistic_model,
                           newdata = test_data_log,
                           type = "response")
predicted_class <- ifelse(predicted_probs > 0.5, 1, 0)

# 4. Evaluar el modelo: Calcular la curva ROC y el AUC
roc_curve_log <- roc(test_data_log$Target, predicted_probs)
auc_value_log <- auc(roc_curve_log)

# Mostrar la curva ROC
plot(roc_curve_log,
     main = "Curva ROC",
     col = "blue",
     lwd = 2,
     print.auc = TRUE)
```




```
cat("El valor del AUC es:", auc_value_log, "\n")
```

```
El valor del AUC es: 0.718638
```

```
# 5. Evaluar el modelo: matriz de confusión y métricas
conf_matrix_log <- confusionMatrix(as.factor(predicted_class),
                                   as.factor(test_data_log$Target),
                                   mode="everything",
                                   positive = "1")

# Mostrar resultados clave
print(conf_matrix_log)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0  6  4
         1 12 27

               Accuracy : 0.6735
                 95% CI : (0.5246, 0.8005)
    No Information Rate : 0.6327
    P-Value [Acc > NIR] : 0.33245

                  Kappa : 0.2253

 Mcnemar's Test P-Value : 0.08012
```

```
            Sensitivity : 0.8710
            Specificity : 0.3333
         Pos Pred Value : 0.6923
         Neg Pred Value : 0.6000
              Precision : 0.6923
                 Recall : 0.8710
                     F1 : 0.7714
             Prevalence : 0.6327
         Detection Rate : 0.5510
   Detection Prevalence : 0.7959
      Balanced Accuracy : 0.6022

       'Positive' Class : 1
```

```
#--------------------------------------------------------------------------------
# IMAGEN 9
#--------------------------------------------------------------------------------
pdf("outputs/images/09_roc_log.pdf", width = 16, height = 10)
# Mostrar la curva ROC
plot(roc_curve_log,
     main = "Curva ROC (Modelo logístico)",
     col = "blue",
     lwd = 2,
     print.auc = TRUE)
dev.off()
```

```
pdf
  2
```

Otra opción sería un análisis de discriminante lineal, pero no se cumplen los postulados requeridos, así que debe desecharse (se deja el código para constancia por si en algún momento se reescribe o ingresan nuevas variables de interés)

```
#Dividimos en train y test
set.seed(1234)
train_idx_lda <- sample(seq_len(nrow(numeric_vars_log)), size = 0.7 * nrow(numeric_vars_log))
train_data_lda <- numeric_vars_log[train_idx_lda, ]
test_data_lda <- numeric_vars_log[-train_idx_lda, ]

# Comprobación de supuestos:

#Seleccionamos únicamente las columnas numéricas (excluyendo Target ya que no es numérica)
predictors <- train_data_lda[, sapply(train_data_lda, is.numeric)]

#Se realiza la prueba de Shapiro-Wilk por grupo (según Target)
normality_results <- by(predictors, train_data_lda$Target, function(group_data) {
  apply(group_data, 2, function(column) shapiro.test(column)$p.value)
})

print(normality_results)
```

```
train_data_lda$Target: 0
          ag        ptg19         tn46         tn52         tn36         tn38
1.205373e-01 1.257356e-01 1.404048e-02 1.977717e-01 2.903471e-04 4.265649e-02
        tn40         tn42         tn22         tn44         tn14         tn24
```

```
2.651212e-01 3.976089e-02 6.099801e-01 9.428373e-02 1.166206e-01 4.772344e-01
        tn12          tn6         tn30         tn34          tn8         tn48
1.795141e-02 5.594861e-02 2.258908e-04 4.418935e-01 5.268126e-04 1.349912e-06
        tn50
2.666192e-03
------------------------------------------------------------
train_data_lda$Target: 1
          ag        ptg19         tn46         tn52         tn36         tn38
8.981599e-01 1.722421e-12 4.648055e-04 1.056511e-01 5.539881e-02 8.465676e-03
        tn40         tn42         tn22         tn44         tn14         tn24
1.384317e-03 3.183252e-02 1.244323e-01 2.128622e-05 2.475254e-02 4.532706e-03
        tn12          tn6         tn30         tn34          tn8         tn48
9.209390e-03 6.882700e-03 3.469233e-02 3.511927e-04 1.815631e-07 7.819760e-13
        tn50
1.257263e-03
```

```r
# Interpretación:
# - Si los p-valores son mayores a 0.05, no se rechaza la hipótesis de normalidad. HAY VARIAS VARIABLES

# b) Homogeneidad de varianzas con test de Levene
levene_results <- sapply(colnames(predictors), function(var) {
  leveneTest(as.formula(paste(var, "~ Target")), data = train_data_lda)$"Pr(>F)"[1]
})
print(levene_results)
```

```
         ag       ptg19        tn46        tn52        tn36        tn38
0.004930289 0.162780627 0.465236170 0.481237892 0.502903058 0.850096454
       tn40        tn42        tn22        tn44        tn14        tn24
0.821186954 0.707354729 0.152171605 0.382286994 0.021942633 0.685160296
       tn12         tn6        tn30        tn34         tn8        tn48
0.012068642 0.211756715 0.017134232 0.821181187 0.054407737 0.324029375
       tn50
0.517469560
```

```r
# Interpretación:
# - Si los p-valores son mayores a 0.05, no se rechaza la hipótesis de igualdad de varianzas.

# c) Multicolinealidad (VIF)
vif_values_lda <- vif(lm(Target ~ ., data = train_data_lda))
print(vif_values_lda)
```

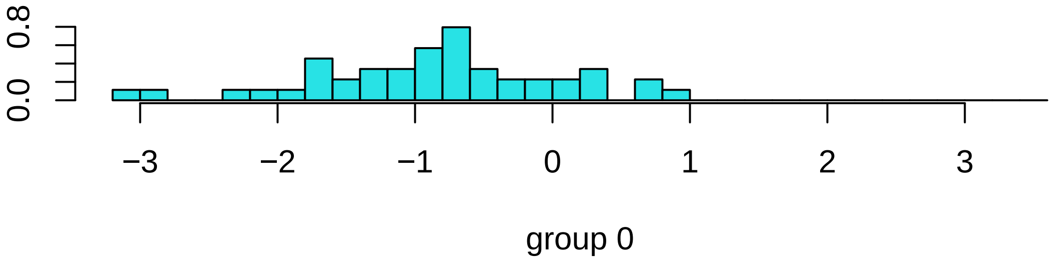
```
   ag ptg19  tn46  tn52  tn36  tn38  tn40  tn42  tn22  tn44  tn14  tn24  tn12
  NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
  tn6  tn30  tn34   tn8  tn48  tn50
  NaN   NaN   NaN   NaN   NaN   NaN
```

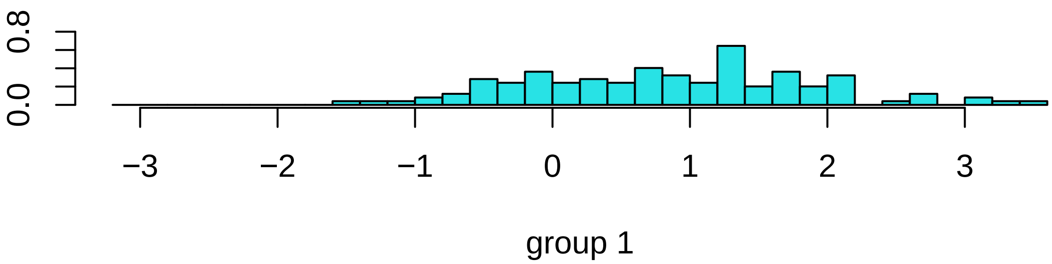
```r
# Interpretación:
# - VIF < 5: No hay problemas significativos de multicolinealidad.
# - VIF > 5: Multicolinealidad moderada.
# - VIF > 10: Multicolinealidad alta.

#Ajustamos modelo LDA
lda_model <- lda(Target ~ ., data = train_data_lda)
print(lda_model)
```

```
Call:
```

```
lda(Target ~ ., data = train_data_lda)

Prior probabilities of groups:
        0         1 
0.2619048 0.7380952 

Group means:
        ag    ptg19       tn46        tn52       tn36       tn38       tn40
0 49.25227 26.13636  0.4157938 -0.02929531  0.2097831  0.3098063  0.2559762
1 48.90968 28.39704 -0.2774194  0.06854839 -0.3804032 -0.2967742 -0.4991935
        tn42      tn22       tn44      tn14        tn24        tn12        tn6
0 -0.1202666 0.6047819  0.1051940 0.9069023  0.01138266  0.85419171  0.5442895
1 -0.9709677 0.3750000 -0.4604839 0.1540323 -0.74677419 -0.05645161 -0.4901613
        tn30       tn34         tn8     tn48      tn50
0  0.6796744 -0.1762052  0.01137652 1.822762 0.4401266
1 -0.2102419 -0.4040323 -0.12419355 1.553226 0.1685484

Coefficients of linear discriminants:
               LD1
ag     0.028832107
ptg19  0.030903295
tn46  -0.558540174
tn52   0.074761765
tn36   0.433106207
tn38   0.059068660
tn40  -0.506254751
tn42  -0.131401322
tn22   0.034172627
tn44  -0.005238356
tn14  -0.278620217
tn24  -0.088924788
tn12  -0.585428294
tn6   -0.039993922
tn30  -0.372490695
tn34   0.159675827
tn8    0.127791856
tn48   0.107149954
tn50   0.139400691
```

```r
#Visualizamos el modelo LDA
plot(lda_model)
```



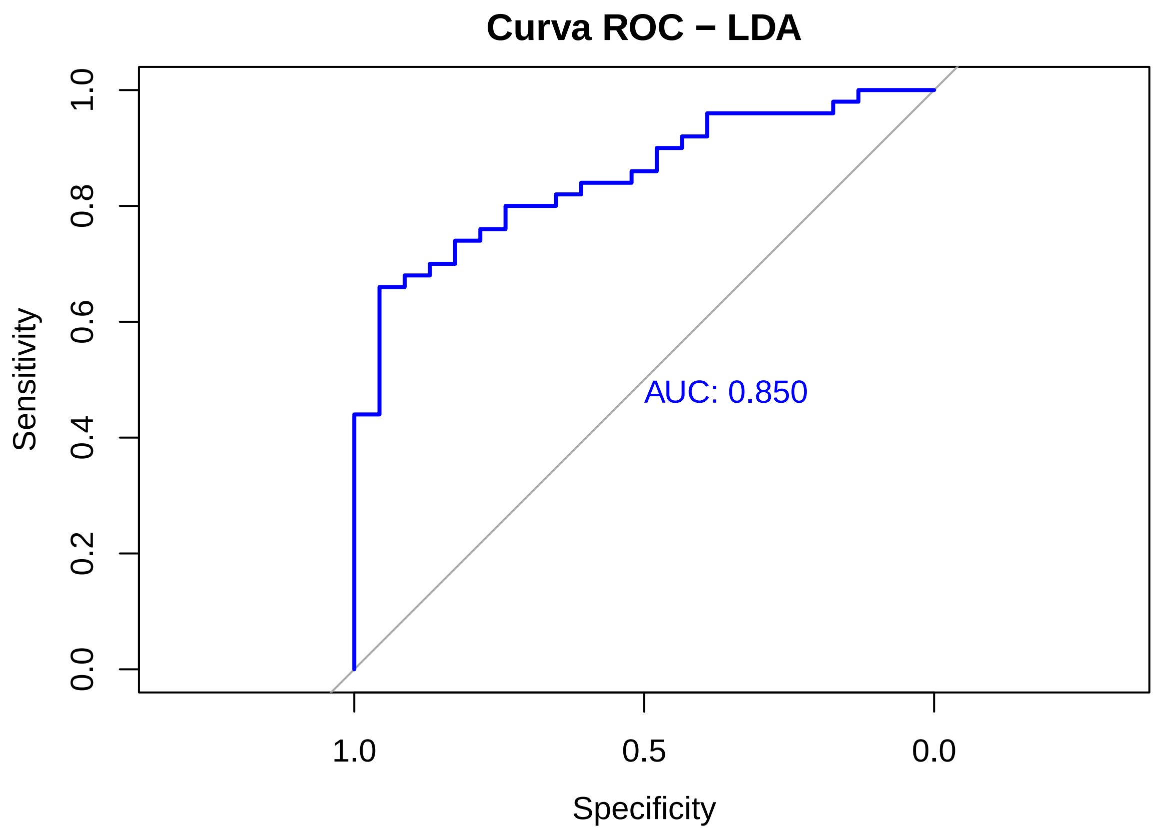



```r
#Realizamos las predicciones
lda_pred <- predict(lda_model, newdata = test_data_lda)
predicted_class <- lda_pred$class
posterior_probs <- lda_pred$posterior

#Matriz de confusión
conf_matrix <- confusionMatrix(as.factor(predicted_class), as.factor(test_data_lda$Target))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 16 10
         1  7 40

               Accuracy : 0.7671
                 95% CI : (0.6535, 0.8581)
    No Information Rate : 0.6849
    P-Value [Acc > NIR] : 0.0803

                  Kappa : 0.4788

 Mcnemar's Test P-Value : 0.6276

            Sensitivity : 0.6957
            Specificity : 0.8000
         Pos Pred Value : 0.6154
         Neg Pred Value : 0.8511
             Prevalence : 0.3151
         Detection Rate : 0.2192
   Detection Prevalence : 0.3562
```

```
      Balanced Accuracy : 0.7478

       'Positive' Class : 0
```

```
#Curva ROC y AUC (lda)
roc_curve_lda <- roc(test_data_lda$Target, posterior_probs[, 2])  # Probabilidad de la clase 1
plot(roc_curve_lda,
     main = "Curva ROC - LDA",
     col = "blue",
     lwd = 2,
     print.auc=TRUE)
```




```
auc_value_lda <- auc(roc_curve_lda)
cat("\nEl valor del AUC es:", auc_value_lda, "\n")
```

```
El valor del AUC es: 0.8495652
```

XGBOOST

Un modelo más complejo pero más generalizable y escalable con mayor precisión para uso clínico real.

```
# Dividir el dataset en entrenamiento y test
set.seed(2429)
#semillas interesantes: (0.8) 1234, 1237, 1238, 1339
#semillas interesantes: (0.7) 1339, 1441 por el recall
#semillas interesantes (0.75) 1234
#Tras mejoras (que es evitar el rol del desbalance y que la clase positiva sea 1), es decir,
#identificar los casos de long covid y 0.75: 1236, 1237, #1239
```

```r
# tras mejoras y 0,8: 2435, 2429, 2433, 2435

trainIndex <- createDataPartition(df_def_clean$Target, p = 0.8, list = FALSE)
train_data <- df_def_clean[trainIndex, ]
test_data <- df_def_clean[-trainIndex, ]

#Definimos la columna objetivo para entrenamiento y prueba
train_target <- train_data$Target
test_target <- test_data$Target

#Eliminamos la columna objetivo de los datos de entrenamiento y prueba
train_data <- train_data %>% dplyr::select(-Target)
test_data <- test_data %>% dplyr::select(-Target)

#Convertimos los datasets a matrices para XGBoost
train_matrix <- as.matrix(train_data)
test_matrix <- as.matrix(test_data)

#Creamos DMatrix para XGBoost
dtrain <- xgb.DMatrix(data = train_matrix, label = train_target)
dtest <- xgb.DMatrix(data = test_matrix, label = test_target)

#Calculamos el peso para la clase positiva (en este dataset hay desbalance)
num_negativos <- sum(train_target == 0)  # Casos de la clase 0
num_positivos <- sum(train_target == 1)  # Casos de la clase 1
scale_pos_weight <- num_negativos / num_positivos

#Generamos la lista de parámetros generales que usaremos para una primera validación (cross-validation)
params <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
    scale_pos_weight = scale_pos_weight  #Peso para manejar el desbalance de clases
)
```

VALIDACIÓN CRUZADA

```r
# Configuración de la validación cruzada con funciones del paquete xgboost
# set.seed(2429)  # Para reproducibilidad
# cv_results <- xgb.cv(
#   params = params,          # Parámetros del modelo
#   data = dtrain,            # DMatrix con los datos de entrenamiento
#   nrounds = 500,            # Número de iteraciones
#   nfold = 10,                # Número de pliegues para validación cruzada
#   metrics = "logloss",      # Métrica de evaluación
#   verbose = TRUE,           # Mostrar progreso
#   stratified = TRUE,        # Mantener proporción de clases en los pliegues
#   early_stopping_rounds = 10 # Detener si no mejora en 10 rondas
# )
#
# # Mejor número de rondas
# best_nrounds <- cv_results$best_iteration
# print(paste("Mejor número de rondas:", best_nrounds))
#
# # Resumen de métricas en los pliegues
```

```
# print(cv_results$evaluation_log)
```

VALIDACIÓN CON CARET

```
# Volver a agregar la columna objetivo a los datos de entrenamiento
train_data_caret <- train_data %>%
  mutate(Target = as.factor(ifelse(train_target == 1, "Yes", "No")))

# Asegúrate de que el conjunto de prueba también tenga Target para evaluación posterior
test_data_caret <- test_data %>%
  mutate(Target = as.factor(ifelse(test_target == 1, "Yes", "No")))

# Configurar control de validación cruzada
train_control <- trainControl(
  method = "cv",    # Validación cruzada
  number = 10,       # Número de pliegues
  verboseIter = TRUE,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)

# Definir los hiperparámetros que queremos probar
tune_grid <- expand.grid(
  nrounds = c(100, 200, 300),  # Número de iteraciones
  max_depth = c(3, 6, 9),      # Profundidad máxima
  eta = c(0.01, 0.1),          # Tasa de aprendizaje
  gamma = 0,                   # Complejidad mínima
  colsample_bytree = 0.8,
  min_child_weight = 1,
  subsample = 0.8
)

# Ejecutar la validación cruzada
xgb_model_caret <- train(
  Target ~ .,
  data = train_data_caret,
  method = "xgbTree",
  trControl = train_control,
  tuneGrid = tune_grid,
  metric = "ROC" # Optimizar el área bajo la curva ROC
)
```

```
+ Fold01: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold01: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
+ Fold01: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold01: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
+ Fold01: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrou
```

```
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold01: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold01: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold01: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold01: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold01: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold01: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold01: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold02: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold02: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold02: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold02: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold02: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold02: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold02: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold02: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold02: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold02: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold02: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
```

```
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold02: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold03: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold03: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold03: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold03: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold03: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold03: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold03: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold03: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold03: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold03: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold03: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold03: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold04: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold04: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold04: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold04: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold04: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
```

```
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold04: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold04: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold04: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold04: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold04: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold04: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold04: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold05: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold05: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold05: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold05: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold05: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold05: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold05: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold05: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold05: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold05: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold05: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
```

```
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold05: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold06: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold06: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold06: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold06: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold06: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold06: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold06: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold06: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold06: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold06: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold06: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold06: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold07: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold07: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold07: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold07: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
+ Fold07: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nroun
```

```
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold07: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold07: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold07: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold07: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold07: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold07: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold07: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold08: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold08: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold08: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold08: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold08: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold08: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold08: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold08: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold08: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold08: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold08: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
```

```
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold08: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold09: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold09: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold09: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold09: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold09: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold09: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold09: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold09: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold09: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold09: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold09: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold09: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold10: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold10: eta=0.01, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold10: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold10: eta=0.01, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold10: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
```

```
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold10: eta=0.01, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold10: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold10: eta=0.10, max_depth=3, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold10: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold10: eta=0.10, max_depth=6, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
+ Fold10: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[18:54:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
- Fold10: eta=0.10, max_depth=9, gamma=0, colsample_bytree=0.8, min_child_weight=1, subsample=0.8, nrour
Aggregating results
Selecting tuning parameters
Fitting nrounds = 300, max_depth = 3, eta = 0.01, gamma = 0, colsample_bytree = 0.8, min_child_weight =
```

```r
# Ver el mejor conjunto de hiperparámetros
print(xgb_model_caret$bestTune)
```

```
  nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
3     300         3 0.01     0              0.8                1       0.8
```

ENTRENAMIENTO DEL MODELO FINAL

```r
# Tras realizar la validación cruzada ampliamos los parámetros
set.seed(1239)

#buenas seeds: 2435, 1239

params_cv <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  scale_pos_weight = scale_pos_weight,  # Peso para manejar el desbalance
  max_depth = 6,                        # Mejor valor de max_depth
  eta = 0.01,                           # Mejor valor de eta
  gamma = 0,                            # Mejor valor de gamma
  colsample_bytree = 0.8,               # Mejor valor de colsample_bytree
  min_child_weight = 1,                 # Mejor valor de min_child_weight
  subsample = 0.8                       # Mejor valor de subsample
)

#Entrenamos el modelo de clasificación binaria con XGBoost
modelo_xgb <- xgboost(data = dtrain,
                      params = params_cv,
                      verbose = 0,
```

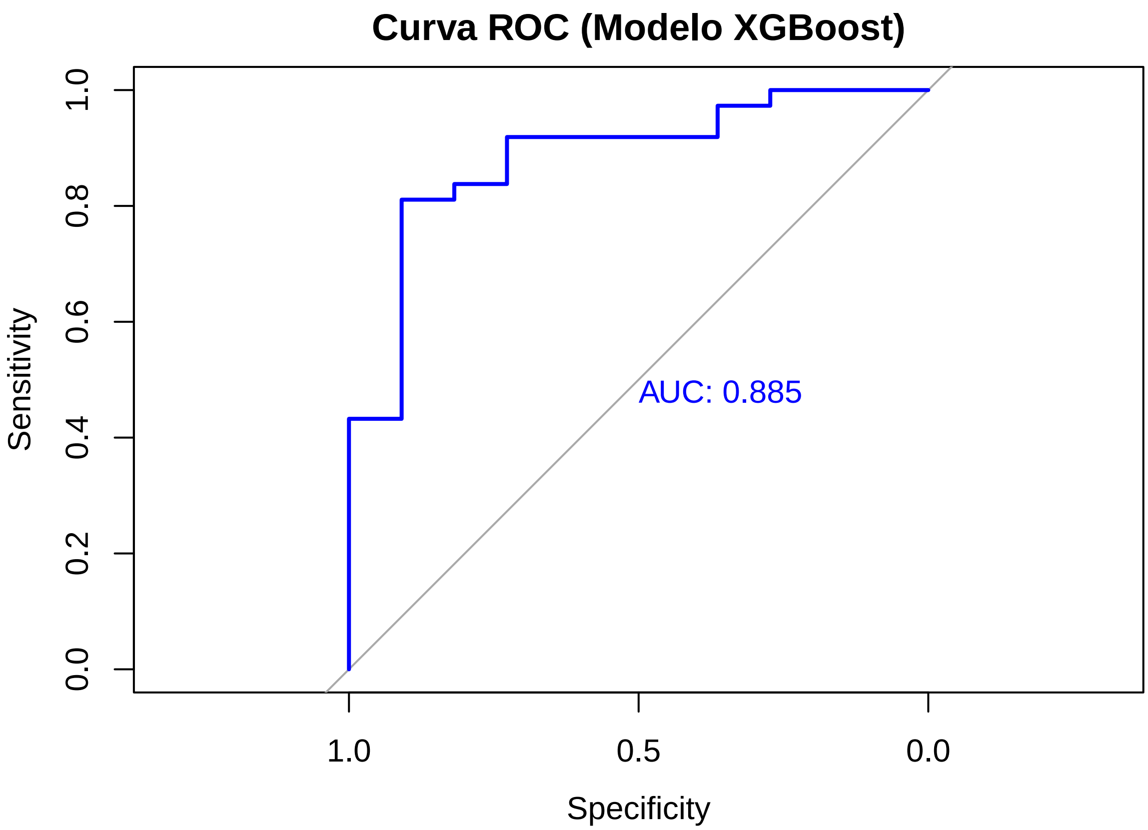
```r
                      nrounds = 100)

#Generamos predicciones en el conjunto de prueba
predicciones <- predict(modelo_xgb, test_matrix)

#Convertimos las probabilidades en clases
predicciones_clase <- ifelse(predicciones >= 0.5, 1, 0)

#Matriz de confusión
matriz_confusion <- table(Predicho = predicciones_clase, Real = test_target)
print(matriz_confusion)
```

```
        Real
Predicho  0  1
       0 10  8
       1  1 29
```

```r
#Matriz de confusión con caret para métricas adicionales
confusion <- confusionMatrix(factor(predicciones_clase),
                             factor(test_target),
                             mode="everything",
                             positive = "1")

print(confusion)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 10  8
         1  1 29

               Accuracy : 0.8125
                 95% CI : (0.6737, 0.9105)
    No Information Rate : 0.7708
    P-Value [Acc > NIR] : 0.3117

                  Kappa : 0.5663

 Mcnemar's Test P-Value : 0.0455

            Sensitivity : 0.7838
            Specificity : 0.9091
         Pos Pred Value : 0.9667
         Neg Pred Value : 0.5556
              Precision : 0.9667
                 Recall : 0.7838
                     F1 : 0.8657
             Prevalence : 0.7708
         Detection Rate : 0.6042
   Detection Prevalence : 0.6250
      Balanced Accuracy : 0.8464

       'Positive' Class : 1
```

```r
#predicciones <- predict(modelo_xgb, test_matrix)

#Creamos curva ROC
roc_xgboost <- roc(test_target, predicciones)

#Curva ROC
plot(roc_xgboost,
     col = "blue",
     main = "Curva ROC (Modelo XGBoost)",
     print.auc = TRUE)
```



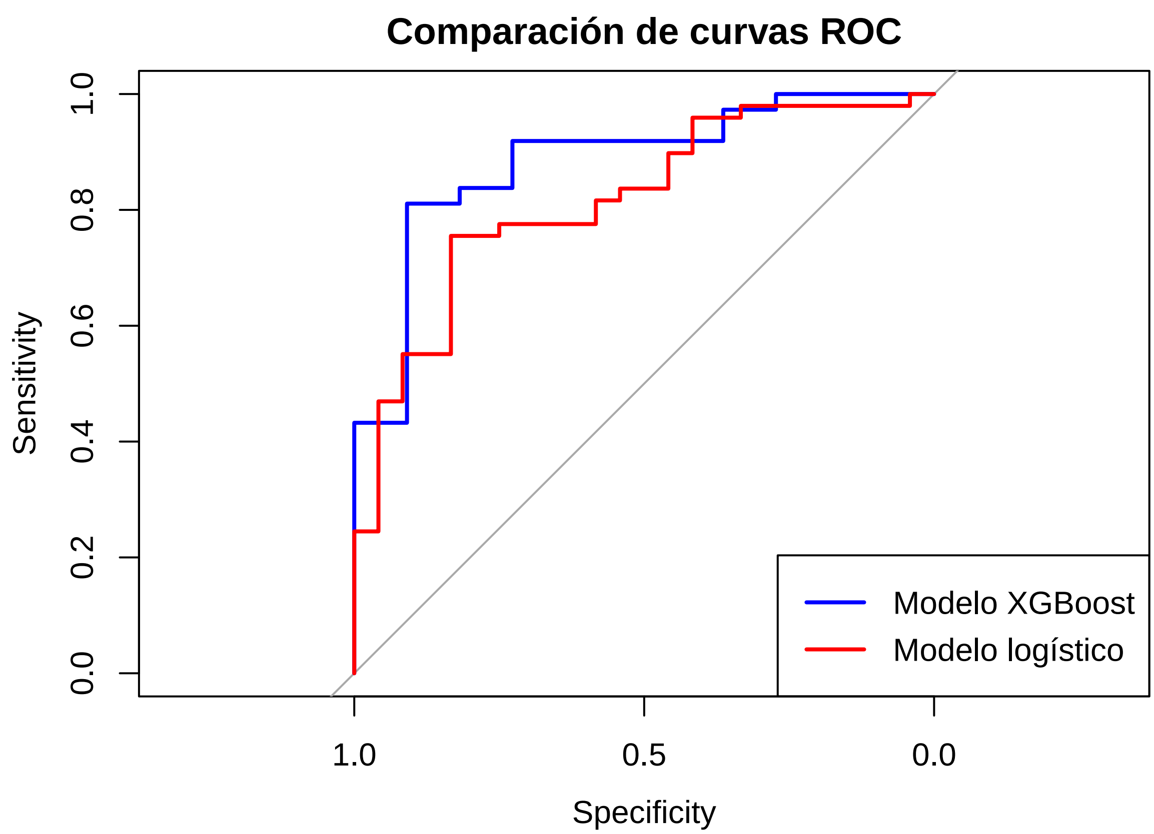

```r
#Calculamos AUC
auc_valor_xgboost <- auc(roc_xgboost)
print(paste("El AUC es:", auc_valor_xgboost))
```

```
[1] "El AUC es: 0.884520884520885"
```

```r
#------------------------------------------------------------------------------
# IMAGEN 10
#------------------------------------------------------------------------------
pdf("outputs/images/10_roc_xg.pdf", width = 16, height = 10)
#Curva ROC
plot(roc_xgboost,
     col = "blue",
     main = "Curva ROC (Modelo XGBoost)",
     print.auc = TRUE)
dev.off()
```

```
pdf
  2
```

```r
# Graficar la primera curva ROC
plot(roc_xgboost,
     col = "blue",
     main = "Comparación de curvas ROC")

# Agregar las demás curvas
lines(roc_curve_log, col = "red")
#lines(roc_curve_lda, col = "green")

# Añadir una leyenda
# legend("bottomright",
#        legend = c("Modelo XGBoost", "Modelo logístico", "Modelo LDA"),
#        col = c("blue", "red", "green"),
#        lwd = 2)

# Añadir una leyenda
legend("bottomright",
       legend = c("Modelo XGBoost", "Modelo logístico"),
       col = c("blue", "red"),
       lwd = 2)
```




```r
#---------------------------------------------------------------------------------
# IMAGEN OPCIONAL
#---------------------------------------------------------------------------------
pdf("outputs/images/00_COMPARACION_ROCS.pdf", width = 16, height = 10)
# Graficar la primera curva ROC
```

```r
plot(roc_xgboost,
     col = "blue",
     main = "Comparación de curvas ROC")

# Agregar la/s otra/s curva/s
lines(roc_curve_log, col = "red")

# Añadir una leyenda
legend("bottomright",
       legend = c("Modelo XGBoost", "Modelo logístico"),
       col = c("blue", "red"),
       lwd = 2)
dev.off()
```

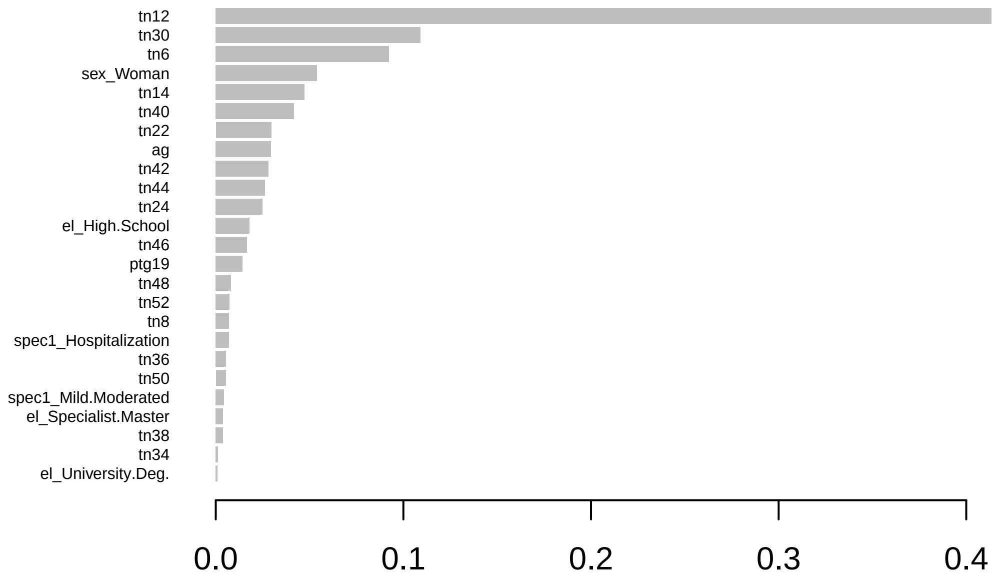
```
pdf 
  2 
```

```r
#Obtenemos la importancia de las variables
importancia <- xgb.importance(model = modelo_xgb, feature_names = colnames(train_matrix))

#Convertimos la importancia en un dataframe
importancia_df <- as.data.frame(importancia)

# Ordenamos las variables por importancia de mayor a menor
importancia_df <- importancia_df[order(-importancia_df$Gain), ]

#Mostramos la importancia de las variables
print(importancia)
```

```
                  Feature         Gain       Cover   Frequency
                   <char>        <num>       <num>       <num>
 1:                  tn12 0.4133791237 0.224203175 0.112305854
 2:                  tn30 0.1091563243 0.133012694 0.101553166
 3:                   tn6 0.0923888949 0.087693001 0.086021505
 4:             sex_Woman 0.0538703178 0.043077310 0.048984468
 5:                  tn14 0.0473565715 0.044630799 0.032258065
 6:                  tn40 0.0417528345 0.057770285 0.057347670
 7:                  tn22 0.0295833778 0.043149535 0.062126643
 8:                    ag 0.0295129718 0.048841068 0.074074074
 9:                  tn42 0.0281031807 0.039377200 0.046594982
10:                  tn44 0.0261445483 0.045458676 0.060931900
11:                  tn24 0.0250257897 0.027709153 0.038231780
12:        el_High.School 0.0180892814 0.045329456 0.043010753
13:                  tn46 0.0165544895 0.020101394 0.032258065
14:                 ptg19 0.0142982850 0.037507361 0.053763441
15:                  tn48 0.0082465546 0.013299920 0.023894863
16:                  tn52 0.0072661591 0.022957027 0.031063321
17:                   tn8 0.0071152291 0.009219339 0.020310633
18: spec1_Hospitalization 0.0071150675 0.007247230 0.004778973
19:                  tn36 0.0055713396 0.017960738 0.020310633
20:                  tn50 0.0053280253 0.008439648 0.016726404
21:  spec1_Mild.Moderated 0.0043082539 0.007092093 0.007168459
22:  el_Specialist.Master 0.0039277664 0.005307804 0.007168459
23:                  tn38 0.0038383871 0.007152710 0.009557945
24:                  tn34 0.0011140242 0.002373180 0.007168459
```

```
25:    el_University.Deg. 0.0009532023 0.001089205 0.002389486
                  Feature         Gain       Cover   Frequency
```

```r
#Graficamos la importancia de las variables
xgb.plot.importance(importance_matrix = importancia)
```



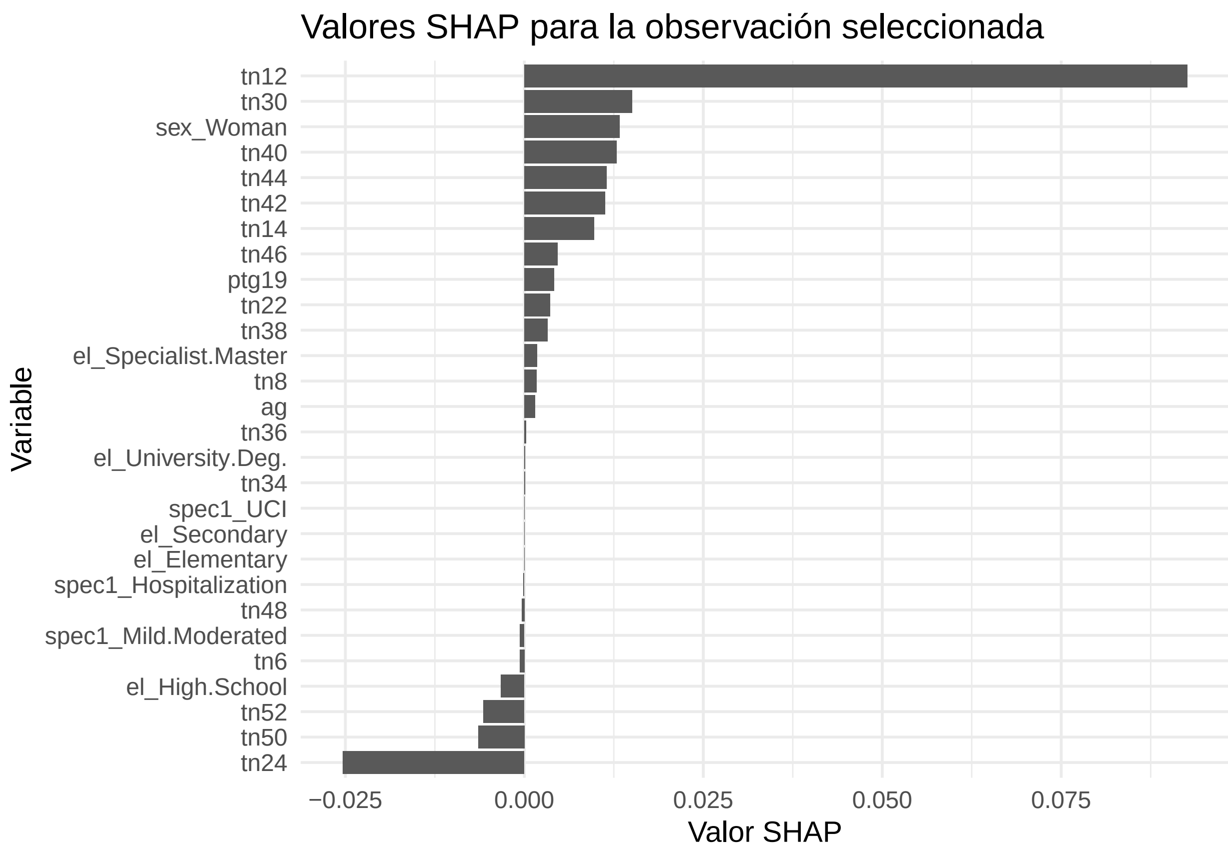

```r
#-------------------------------------------------------------------------------
# IMAGEN 11
#-------------------------------------------------------------------------------
pdf("outputs/images/11_importancia.pdf", width = 10, height = 10)
#Graficamos la importancia de las variables
xgb.plot.importance(importance_matrix = importancia)
dev.off()
```

```
pdf
  2
```

Para darle mayor explicabilidad al modelo vamos a calcular los valores SHAP de observaciones individuales

```r
#Definimos la función de predicción personalizada para el modelo xgboost
pred_fun <- function(object, newdata) {
  predict(object, newdata = xgb.DMatrix(data = as.matrix(newdata)))
}

#Convertimos test_data a dataframe y seleccionamos una observación para explicar
test_df <- as.data.frame(as.matrix(test_data))
new_observation <- test_df[2, , drop = FALSE]  # Seleccionar la observación número 2

#Calculamos los valores SHAP para la observación seleccionada
set.seed(2435)
shap_values_fast <- fastshap::explain(
  object = modelo_xgb,             # El modelo de xgboost
  feature_names = colnames(train_data), # Los nombres de las variables
  newdata = new_observation,       # La observación para la cual queremos valores SHAP
  pred_wrapper = pred_fun,         # La función de predicción personalizada
  X = train_data,                    # El conjunto de entrenamiento completo
  nsim = 100                       # Número de simulaciones
)
```

```r
#Convertimos los valores SHAP a un dataframe para graficar
shap_df <- data.frame(
  Variable = colnames(new_observation),
  SHAP = shap_values_fast[1, ]  # Solo la primera fila (si tienes más observaciones)
)

#Graficamos los valores SHAP
ggplot(shap_df, aes(x = reorder(Variable, SHAP), y = SHAP)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Valores SHAP para la observación seleccionada",
       x = "Variable",
       y = "Valor SHAP") +
  theme_minimal()
```

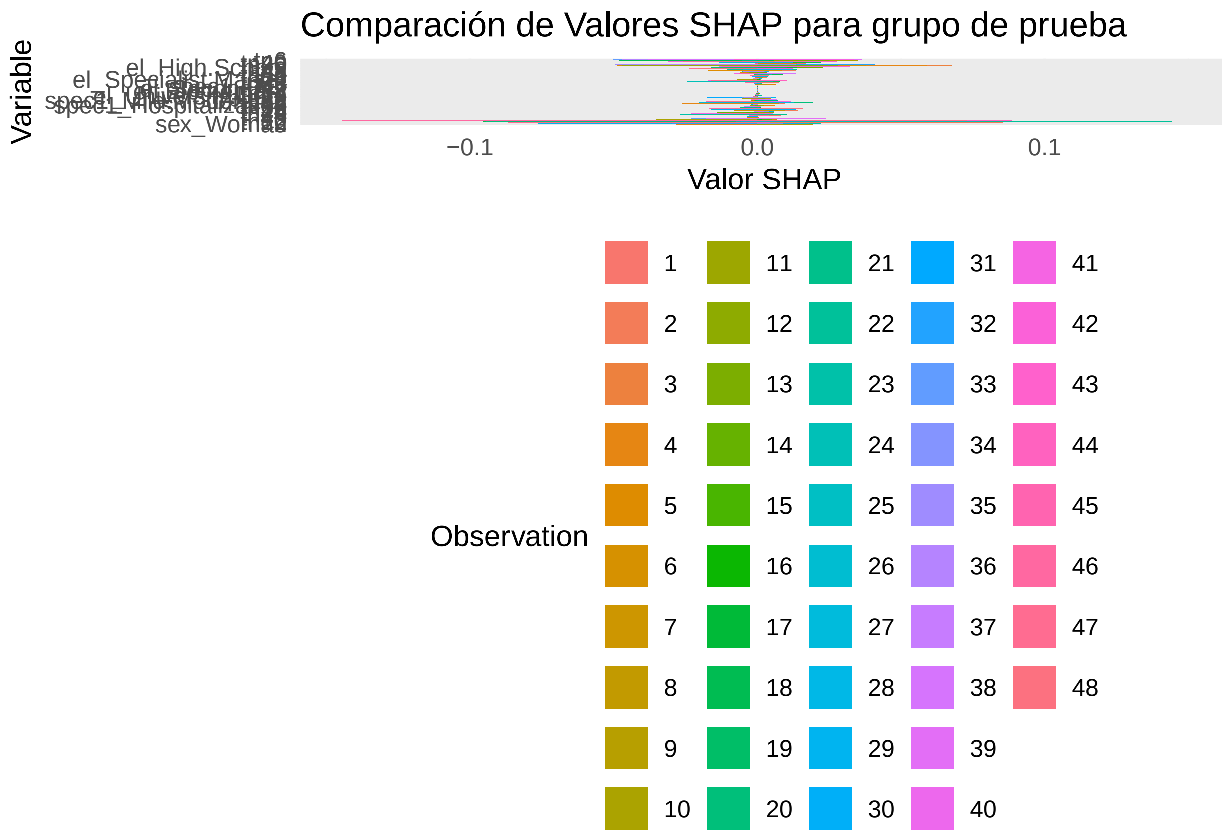



```r
#-----------------------------------------------------------------------------
# IMAGEN 12
#-----------------------------------------------------------------------------
pdf("outputs/images/12_ejemplo_SHAP.pdf", width = 16, height = 10)
#Graficamos los valores SHAP
ggplot(shap_df, aes(x = reorder(Variable, SHAP), y = SHAP)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Valores SHAP para la observación seleccionada",
       x = "Variable",
       y = "Valor SHAP") +
  theme_minimal()
```

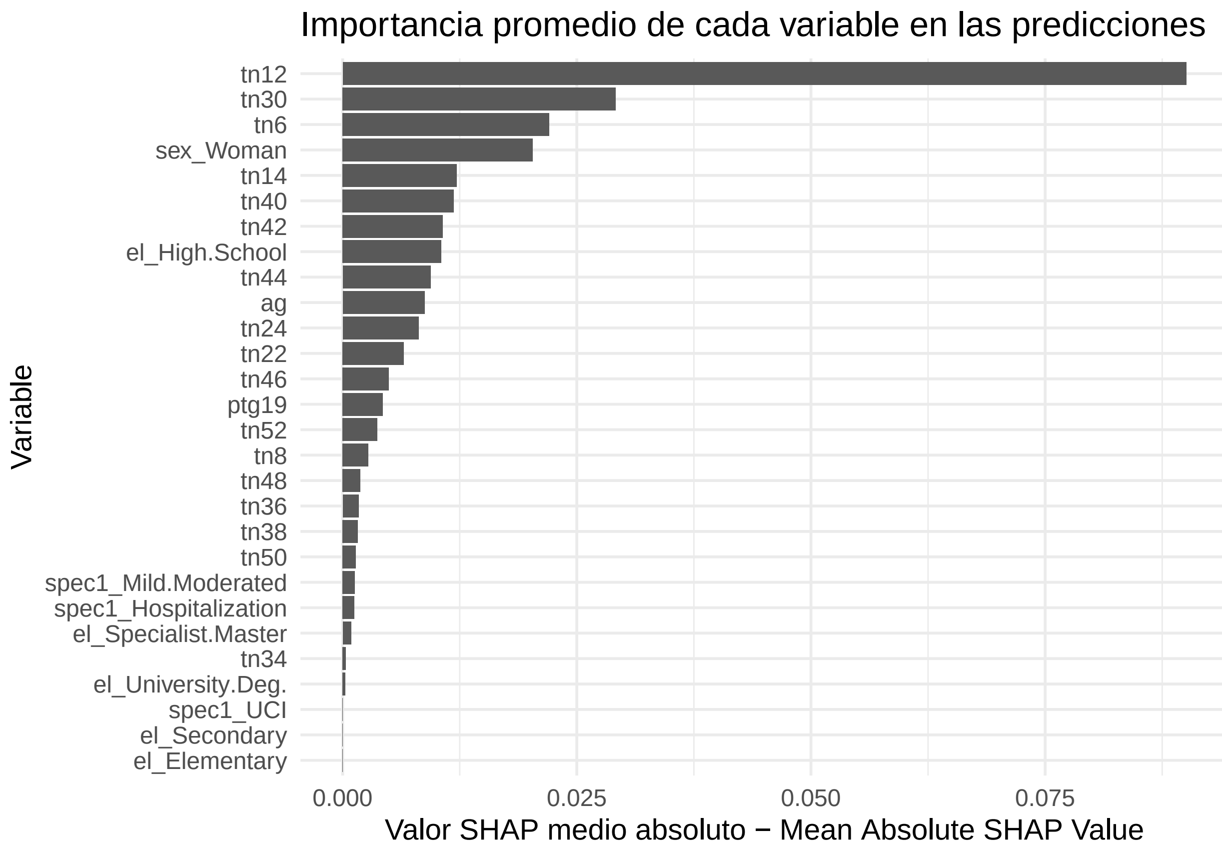
```r
dev.off()
```

```
pdf
  2
```

```r
########### VARIOS VALORES ################################

#Convertimos test_data a dataframe y seleccionamos múltiples observaciones para explicar. Usaremos todo
test_df <- as.data.frame(as.matrix(test_data))
new_observations <- test_df[1:48, , drop = FALSE]  #Seleccionamos todo el conjunto de dataset

#Calculamos los valores SHAP para las observaciones seleccionadas
set.seed(2435)
shap_values_fast_24 <- fastshap::explain(
  object = modelo_xgb,               # El modelo de xgboost
  feature_names = colnames(train_data), # Los nombres de las variables
  newdata = new_observations,         # Las observaciones seleccionadas
  pred_wrapper = pred_fun,            # La función de predicción personalizada
  X = train_data,                       # El conjunto de entrenamiento completo
  nsim = 100                          # Número de simulaciones
)

#Convertimos shap_values_fast a dataframe y añadimos un identificador de observación en el proceso de p
shap_df_24 <- as.data.frame(shap_values_fast_24)
colnames(shap_df_24) <- colnames(train_data)
shap_df_24$Observation <- factor(1:48)

#Convertimos a formato largo para visualización
shap_df_long <- shap_df_24 %>%
  pivot_longer(cols = -Observation, names_to = "Variable", values_to = "SHAP")

#Graficar los valores SHAP para múltiples observaciones
ggplot(shap_df_long, aes(x = reorder(Variable, SHAP), y = SHAP, fill = Observation)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  labs(title = "Comparación de Valores SHAP para grupo de prueba",
       x = "Variable",
       y = "Valor SHAP") +
  theme_minimal() +
  theme(legend.position = "bottom")
```




```r
############### COMPARAR TODOS LOS VALORES SHAP ################################

#Calculamos la importancia promedio absoluta de cada variable: la idea es tener un peso "absoluto" (en
importance_summary <- shap_df_24 %>%
  dplyr::select(-Observation) %>%  # Excluimos la columna de observación
  summarise_all(~ mean(abs(.))) %>%  # Calculamos la media absoluta para cada variable
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Mean_Absolute_SHAP") %>%
  arrange(desc(Mean_Absolute_SHAP))  # Ordenamos de mayor a menor importancia

# Graficar la importancia promedio de cada variable
ggplot(importance_summary, aes(x = reorder(Variable, Mean_Absolute_SHAP), y = Mean_Absolute_SHAP)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Importancia promedio de cada variable en las predicciones",
       x = "Variable",
       y = "Valor SHAP medio absoluto - Mean Absolute SHAP Value") +
  theme_minimal()
```




```r
#Mostramos el dataframe con la importancia promedio de cada variable
print(importance_summary)
```

```
# A tibble: 28 x 2
   Variable       Mean_Absolute_SHAP
   <chr>                       <dbl>
 1 tn12                      0.0901
 2 tn30                      0.0292
 3 tn6                       0.0221
 4 sex_Woman                 0.0203
 5 tn14                      0.0122
 6 tn40                      0.0119
 7 tn42                      0.0107
 8 el_High.School            0.0105
 9 tn44                      0.00938
10 ag                        0.00874
# i 18 more rows
```

```r
#---------------------------------------------------------------------------------
# IMAGEN 13
#---------------------------------------------------------------------------------
pdf("outputs/images/13_valores_SHAP_absolutos.pdf", width = 16, height = 10)
# Graficar la importancia promedio de cada variable
ggplot(importance_summary, aes(x = reorder(Variable, Mean_Absolute_SHAP), y = Mean_Absolute_SHAP)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Importancia promedio de cada variable en las predicciones",
       x = "Variable",
```

```r
       y = "Valor SHAP medio absoluto - Mean Absolute SHAP Value") +
  theme_minimal()
dev.off()
```

```
pdf
  2
```