Anexo III

Julio Ladron de Guevara Jimenez

2023-09-06

```
options(repos = c(CRAN = "https://cloud.r-project.org"))
```

```
library(kableExtra)
library(dplyr)
library(caret)

library(fastDummies)
library(smotefamily)

# l-regression
library(MASS)

# K-neighbours
library(pROC)
library(FNN)

# Decision tree
library(rpart.plot)
library(rpart)

# Random-Forest
library(randomForest)
```

3) Modelos de clasificacion

3.0) Pretratamiento de los datos

Una vez hemos hecho el análisis exploratorio de las variables y nos quedamos con las variables necesarias, hacemos los modelos de clasificacion

```
datos <- read.csv("/home/guincho/Desktop/stroke_final.csv")
```

Pasamos a factorial

Gracias a la función str(), vemos que las variables categóricas están guardadas en forma de int o chr. Esto puede causar fallos en el modelo de regresión logistica en R. Así que las pasamos todas a factor.

d_negativo significará diagnóstico negativo y d_positivo diangnóstico positivo

```
str(datos)
```

```
## 'data.frame':    5110 obs. of  9 variables:
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
```

```
##  $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married     : int  1 1 1 1 1 1 1 0 1 1 ...
##  $ work_type        : chr  "Private" "Self-employed" "Private" "Private" ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : num  36.6 28.1 32.5 34.4 24 29 27.4 22.8 28.1 24.2 ...
##  $ smoking_status   : chr  "formerly smoked" "never smoked" "never smoked" "smokes" ...
##  $ stroke           : int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
datos$smoking_status <- gsub(" ", "_", datos$smoking_status)
datos$work_type <- gsub(" ", "_", datos$work_type)
datos$work_type <- gsub("-", "_", datos$work_type)


datos$stroke = ifelse(test = datos$stroke == 0,
                      yes = "D_negativo", no = "D_positivo")
datos$stroke = as.factor(datos$stroke)


datos$hypertension = ifelse(test = datos$hypertension == 0,
                            yes = "No_hipertenso", no = "Hipertenso")
datos$hypertension = as.factor(datos$hypertension)


datos$heart_disease = ifelse(test = datos$heart_disease == 0,
                             yes = "Enf_coronaria_negativa",
                             no = "Enf_coronaria_positiva")
datos$heart_disease = as.factor(datos$heart_disease)


datos$ever_married = ifelse(test = datos$ever_married == 0,
                            yes = "Nunca_casado",
                            no = "Casado")
datos$ever_married = as.factor(datos$ever_married)

datos$work_type = as.factor(datos$work_type)

datos$smoking_status = as.factor(datos$smoking_status)


str(datos)
```

```
## 'data.frame':    5110 obs. of  9 variables:
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : Factor w/ 2 levels "Hipertenso","No_hipertenso": 2 2 2 2 1 2 1 2 2 2 ...
##  $ heart_disease    : Factor w/ 2 levels "Enf_coronaria_negativa",..: 2 1 2 1 1 1 2 1 1 1 ...
##  $ ever_married     : Factor w/ 2 levels "Casado","Nunca_casado": 1 1 1 1 1 1 1 2 1 1 ...
##  $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 5 4 4 5 4 4 4 4 4 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : num  36.6 28.1 32.5 34.4 24 29 27.4 22.8 28.1 24.2 ...
##  $ smoking_status   : Factor w/ 4 levels "formerly_smoked",..: 1 2 2 3 2 1 2 2 4 4 ...
##  $ stroke           : Factor w/ 2 levels "D_negativo","D_positivo": 2 2 2 2 2 2 2 2 2 2 ...
```

Creando variables dummy

Para algunos modelos de predicción las entradas deben ser numéricas. Es para ello que creamos variables dummy. Para representar información categórica en formato numérico.

Para ello vamos a utilizar un paquete llamado fastDummies

Dejamos Stroke como factor

```
datos_dummies  <- dummy_cols(datos, remove_first_dummy = TRUE,
                             select_columns = c("hypertension",
                                                "heart_disease", "ever_married",
                                                "work_type","smoking_status"),
                             remove_selected_columns = TRUE)


str(datos_dummies)
```

```
## 'data.frame':    5110 obs. of  14 variables:
##  $ age                                 : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ avg_glucose_level                   : num  229 202 106 171 174 ...
##  $ bmi                                 : num  36.6 28.1 32.5 34.4 24 29 27.4 22.8 28.1 24.2 ...
##  $ stroke                              : Factor w/ 2 levels "D_negativo","D_positivo": 2 2 2 2 2 2 2
##  $ hypertension_No_hipertenso          : int  1 1 1 1 0 1 0 1 1 1 ...
##  $ heart_disease_Enf_coronaria_positiva: int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married_Nunca_casado           : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ work_type_Govt_job                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Never_worked              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Private                   : int  1 0 1 1 0 1 1 1 1 1 ...
##  $ work_type_Self_employed             : int  0 1 0 0 1 0 0 0 0 0 ...
##  $ smoking_status_never_smoked         : int  0 1 1 0 1 0 1 1 0 0 ...
##  $ smoking_status_smokes               : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ smoking_status_Unknown              : int  0 0 0 0 0 0 0 0 1 1 ...
```

Dividimos en subconjunto train y test

Ahora dividimos nuestro dataset datos_dummies en un subconjunto train y otro test. Fijamos la semilla

Hacemos esta operación la primera. Porque vamos a manipular los datos de train. Sin embargo, no queremos manipular los datos de test. Vamos a usarlos al final para evaluar el rendimiento de nuestros modelos

```
set.seed(123)

train = createDataPartition(y = datos_dummies$stroke,
                            p = 0.8,
                            list = FALSE)

train_dummies = datos_dummies[train, ]
test_dummies = datos_dummies[-train, ]

table(train_dummies$stroke)
```

```
##
## D_negativo D_positivo
##       3889        200
```

```
table(test_dummies$stroke)
```

```
##
```

```
## D_negativo D_positivo
##        972         49
```

Escalar las variables continuas

La estandarización es una técnica comúnmente utilizada en la regresión logística para asegurarnos de que las variables independientes (predictoras) tengan la misma escala. Esto puede ser beneficioso porque la regresión logística se basa en la función logística, que es sensible a la escala de las variables. Al estandarizar las variables, podemos comparar más fácilmente el impacto relativo de cada variable en el modelo y garantizar que los coeficientes sean más interpretables.

- $X_{\text{estandarizado}}$ es el valor estandarizado
- X es el valor original de la variable.
- μ es la media de la variable.
- σ es la desviación estándar de la variable.

$$X_{\text{estandarizado}} = \frac{X - \mu}{\sigma}$$

```
train_dummies_N <- train_dummies %>%
  mutate(
    bmi = scale(bmi, center = TRUE, scale = TRUE),
    age = scale(age, center = TRUE, scale = TRUE),
    avg_glucose_level = scale(avg_glucose_level, center = TRUE, scale = TRUE)
  )


str(train_dummies_N)
```

```
## 'data.frame':    4089 obs. of  14 variables:
##  $ age                                 : num [1:4089, 1] 1.047 1.623 0.251 1.578 1.667 ...
##   ..- attr(*, "scaled:center")= num 43.3
##   ..- attr(*, "scaled:scale")= num 22.6
##  $ avg_glucose_level                   : num [1:4089, 1] 2.6929 -0.0158 1.4252 1.4889 1.7557 ...
##   ..- attr(*, "scaled:center")= num 107
##   ..- attr(*, "scaled:scale")= num 45.3
##  $ bmi                                 : num [1:4089, 1] 0.9905 0.4601 0.7059 -0.6395 0.0073 ...
##   ..- attr(*, "scaled:center")= num 28.9
##   ..- attr(*, "scaled:scale")= num 7.73
##  $ stroke                              : Factor w/ 2 levels "D_negativo","D_positivo": 2 2 2 2 2 2 2
##  $ hypertension_No_hipertenso          : int  1 1 1 0 1 0 1 1 1 0 ...
##  $ heart_disease_Enf_coronaria_positiva: int  1 1 0 0 0 1 0 0 0 0 ...
##  $ ever_married_Nunca_casado           : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ work_type_Govt_job                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Never_worked              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Private                   : int  1 1 1 0 1 1 1 1 1 1 ...
##  $ work_type_Self_employed             : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ smoking_status_never_smoked         : int  0 1 0 1 0 1 1 0 0 1 ...
##  $ smoking_status_smokes               : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ smoking_status_Unknown              : int  0 0 0 0 0 0 0 1 1 0 ...
```

```
test_dummies_N = test_dummies %>%
  mutate(
    bmi = scale(bmi, center = TRUE, scale = TRUE),
    age = scale(age, center = TRUE, scale = TRUE),
    avg_glucose_level = scale(avg_glucose_level, center = TRUE, scale = TRUE)
```

```
  )

str(test_dummies_N)
```

```
## 'data.frame':    1021 obs. of  14 variables:
##  $ age                                : num [1:1021, 1] 0.803 1.596 0.318 0.759 0.627 ...
##   ..- attr(*, "scaled:center")= num 42.8
##   ..- attr(*, "scaled:scale")= num 22.7
##  $ avg_glucose_level                  : num [1:1021, 1] 2.174 2.437 1.402 -0.332 2.504 ...
##   ..- attr(*, "scaled:center")= num 104
##   ..- attr(*, "scaled:scale")= num 45.1
##  $ bmi                                : num [1:1021, 1] -0.0575 -0.0443 0.3123 1.2236 -0.0575 ...
##   ..- attr(*, "scaled:center")= num 28.5
##   ..- attr(*, "scaled:scale")= num 7.57
##  $ stroke                             : Factor w/ 2 levels "D_negativo","D_positivo": 2 2 2 2 2 2 2
##  $ hypertension_No_hipertenso         : int  1 1 0 1 1 0 1 1 0 0 ...
##  $ heart_disease_Enf_coronaria_positiva: int  0 1 0 0 1 0 1 1 0 0 ...
##  $ ever_married_Nunca_casado          : int  0 0 0 1 1 0 0 0 0 0 ...
##  $ work_type_Govt_job                 : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ work_type_Never_worked             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Private                  : int  0 1 0 1 0 0 1 0 1 1 ...
##  $ work_type_Self_employed            : int  1 0 1 0 0 1 0 1 0 0 ...
##  $ smoking_status_never_smoked        : int  1 1 1 1 0 1 0 0 0 1 ...
##  $ smoking_status_smokes              : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ smoking_status_Unknown             : int  0 0 0 0 1 0 1 0 0 0 ...
```

Balancear

Como hemos visto, tenemos una cantidad mucho mayor de Diagnósticos negativos que positivos. Para arreglar esto es necesario rebalancear.

Vamos a usar la funcion SMOTE de smotefamily.

SMOTE (Synthetic Minority Over-sampling Technique) es una técnica de sobre-muestreo que ayuda a abordar el desequilibrio de clases en conjuntos de datos. En lugar de simplemente duplicar los datos de la clase minoritaria, SMOTE crea instancias sintéticas que son combinaciones lineales de ejemplos de la clase minoritaria existente. Esto ayuda a mitigar el sesgo que podría surgir si simplemente replicáramos los datos originales.

Al generar muestras sintéticas, SMOTE se asegura de mantener la estructura y las características de los datos originales, lo que resulta en un conjunto de datos balanceado

```
target <- train_dummies_N$stroke


X <- train_dummies_N[, !names(train_dummies_N) %in% "stroke"]


X_smote <- SMOTE(X, target, K = 5, dup_size = 0)
```

SMOTE genera una lista con 10 elementos. Entre ellos está:

- $data El conjunto de datos resultante consta de instancias minoritarias originales, instancias minoritarias sintéticas e instancias mayoritarias originales, con un vector de sus respectivas clases objetivo adjunto en la última columna.
- $syn_data Un conjunto de instancias minoritarias sintéticas con un vector de la clase objetivo minoritaria adjunto en la última columna

- $orig_N Un conjunto de instancias originales cuya clase no ha sido sobre-muestreada, con un vector de su clase objetivo adjunto en la última columna.
- $orig_P: Un conjunto de instancias originales cuya clase ha sido sobre-muestreada, con un vector de su clase objetivo adjunto en la última columna.

A nosotros el que mas nos interesa es data. Las nuevas están guardadas en la columna class

SMOTE también cambia el tipo de todas las variables a num

Queremos que las que eran int sigan siendolo

```
X_new = X_smote$data

table(X_new$class)
```

```
##
## D_negativo D_positivo
##       3889       3800
```

```
table(X_smote$syn_data$class) #Para ver cuantos nuevos hay
```

```
##
## D_positivo
##       3600
```

```
#Le damos el nombre stroke a la columna class
names(X_new)[names(X_new) == "class"] <- "stroke"

X_new$stroke = as.factor(X_new$stroke)

str(X_new)
```

```
## 'data.frame':    7689 obs. of  14 variables:
##  $ age                                : num [1:7689, 1] 1.578 0.87 1.534 0.87 0.649 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ avg_glucose_level                  : num [1:7689, 1] -0.19646 -0.34715 0.00233 2.25077 -1.03198
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ bmi                                : num [1:7689, 1] -0.963 -0.109 0.525 0.227 -0.122 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ hypertension_No_hipertenso         : num  1 1 1 1 1 0 1 1 1 1 ...
##  $ heart_disease_Enf_coronaria_positiva: num  0 0 0 0 0 0 0 1 0 0 ...
##  $ ever_married_Nunca_casado          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Govt_job                 : num  0 0 0 0 0 1 0 0 0 0 ...
##  $ work_type_Never_worked             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Private                  : num  1 1 1 1 1 0 1 1 1 0 ...
##  $ work_type_Self_employed            : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ smoking_status_never_smoked        : num  0 0 0 1 0 0 0 0 0 1 ...
##  $ smoking_status_smokes              : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ smoking_status_Unknown             : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ stroke                             : Factor w/ 2 levels "D_negativo","D_positivo": 2 2 2 2 2 2 2
```

3.1) Regresion logistica

3.1.a) Glm con y sin SMOTE

Vamos a ver que glm es mejor. Si la que tiene dummies y el SMOTE o la entrenamiento_dummies sin SMOTE.

```
#sin SMOTE
X_glm_1 = train_dummies_N

#Con SMOTE
X_glm_2 = X_new



logistico1 = glm(stroke~., data =  X_glm_1, family = binomial)

logistico2 = glm(stroke~., data =  X_glm_2, family = binomial)
```

```
logistico1
```

```
##
## Call:  glm(formula = stroke ~ ., family = binomial, data = X_glm_1)
##
## Coefficients:
##                           (Intercept)                                   age
##                             -2.854529                              1.682496
##                     avg_glucose_level                                   bmi
##                              0.192158                              0.008481
##           hypertension_No_hipertenso  heart_disease_Enf_coronaria_positiva
##                             -0.263458                              0.269506
##            ever_married_Nunca_casado                     work_type_Govt_job
##                              0.208079                             -1.015940
##               work_type_Never_worked                      work_type_Private
##                            -10.517299                             -0.829327
##              work_type_Self_employed           smoking_status_never_smoked
##                             -1.296910                             -0.269159
##                smoking_status_smokes                 smoking_status_Unknown
##                              0.198590                             -0.085732
##
## Degrees of Freedom: 4088 Total (i.e. Null);  4075 Residual
## Null Deviance:       1597
## Residual Deviance: 1282  AIC: 1310
```

```
logistico2
```

```
##
## Call:  glm(formula = stroke ~ ., family = binomial, data = X_glm_2)
##
## Coefficients:
##                           (Intercept)                                   age
##                               0.76921                              1.97708
##                     avg_glucose_level                                   bmi
##                               0.15705                              0.02868
##           hypertension_No_hipertenso  heart_disease_Enf_coronaria_positiva
##                             -0.27801                             -0.03541
##            ever_married_Nunca_casado                     work_type_Govt_job
```

```
##                              -0.19819                              -1.77523
##               work_type_Never_worked                    work_type_Private
##                            -11.06840                              -1.37909
##              work_type_Self_employed          smoking_status_never_smoked
##                             -1.92848                              -0.61260
##                smoking_status_smokes               smoking_status_Unknown
##                              0.02773                              -0.36223
##
## Degrees of Freedom: 7688 Total (i.e. Null);  7675 Residual
## Null Deviance:       10660
## Residual Deviance: 7266  AIC: 7294
```

```
summary(logistico1)
```

```
##
## Call:
## glm(formula = stroke ~ ., family = binomial, data = X_glm_1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0471  -0.3262  -0.1695  -0.0923   3.5442
##
## Coefficients:
##                                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -2.854529   0.817008  -3.494 0.000476 ***
## age                                 1.682496   0.146513  11.484  < 2e-16 ***
## avg_glucose_level                   0.192158   0.060218   3.191 0.001418 **
## bmi                                 0.008481   0.096986   0.087 0.930317
## hypertension_No_hipertenso         -0.263458   0.189391  -1.391 0.164202
## heart_disease_Enf_coronaria_positiva  0.269506   0.213834   1.260 0.207542
## ever_married_Nunca_casado           0.208079   0.253795   0.820 0.412290
## work_type_Govt_job                 -1.015940   0.860582  -1.181 0.237791
## work_type_Never_worked            -10.517299 458.867816  -0.023 0.981714
## work_type_Private                  -0.829327   0.839569  -0.988 0.323250
## work_type_Self_employed            -1.296910   0.866342  -1.497 0.134394
## smoking_status_never_smoked        -0.269159   0.196623  -1.369 0.171027
## smoking_status_smokes               0.198590   0.234476   0.847 0.397022
## smoking_status_Unknown             -0.085732   0.231053  -0.371 0.710601
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1597.1  on 4088  degrees of freedom
## Residual deviance: 1282.3  on 4075  degrees of freedom
## AIC: 1310.3
##
## Number of Fisher Scoring iterations: 14
```

Este modelo es el resultado de 14 iteraciones

El paquete MASS tiene una funcion llamada stepAIC

Esta función lleva a cabo una búsqueda automática para determinar el subconjunto óptimo de predictores que deben incluirse en el modelo final, con el objetivo de minimizar un criterio de selección específico, como el AIC

```
# stepAIC(logistico1)
# stepAIC(logistico2)
```

este es el AIC mas pequeño que ha salido

Call: glm(formula = stroke ~ age + avg_glucose_level + hypertension_No_hipertenso + heart_disease_Enf_coronaria_positiv
+ work_type_Self_employed + smoking_status_never_smoked, family = binomial, data = X_glm_1)

Coefficients: 1)intercept: -3.5678; 2)age: 1.5949; 3)avg_glucose_level: 0.1917; 4) Hypertension_No_hipertenso: -0.2825; 5)heart_disease_Enf_coronaria_positiva: 0.3072;
6) work_type_Self_employed: -0.4181; 7) smoking_status_never_smoked: -0.2974

Degrees of Freedom: 4088 Total (i.e. Null); 4082 Residual Null Deviance: 1597 Residual Deviance: 1286 AIC: 1300

Mejora un poco el AIC, pero la devianza residual empeora un poco comparado con logistico1

```
logistico = glm(formula = stroke ~ age + avg_glucose_level + hypertension_No_hipertenso +
                  heart_disease_Enf_coronaria_positiva + work_type_Self_employed +
                  smoking_status_never_smoked, family = binomial, data = X_glm_1,
                                  control = list(maxit = 10))


# Modelo sin SMOTE, pero con mejor AIC
summary(logistico)
```

```
##
## Call:
## glm(formula = stroke ~ age + avg_glucose_level + hypertension_No_hipertenso +
##     heart_disease_Enf_coronaria_positiva + work_type_Self_employed +
##     smoking_status_never_smoked, family = binomial, data = X_glm_1,
##     control = list(maxit = 10))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0334  -0.3324  -0.1750  -0.0834   3.7337
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -3.56785    0.23161 -15.404   <2e-16 ***
## age                                 1.59494    0.13274  12.016   <2e-16 ***
## avg_glucose_level                   0.19169    0.05875   3.263   0.0011 **
## hypertension_No_hipertenso         -0.28249    0.18729  -1.508   0.1315
## heart_disease_Enf_coronaria_positiva  0.30721    0.21232   1.447   0.1479
## work_type_Self_employed            -0.41811    0.18446  -2.267   0.0234 *
## smoking_status_never_smoked        -0.29743    0.16103  -1.847   0.0647 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1597.1  on 4088  degrees of freedom
## Residual deviance: 1285.9  on 4082  degrees of freedom
## AIC: 1299.9
##
## Number of Fisher Scoring iterations: 7
```

```r
# Modelo con SMOTE
summary(logistico2)
```

```
##
## Call:
## glm(formula = stroke ~ ., family = binomial, data = X_glm_2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5348  -0.6819  -0.1803   0.7658   2.7910
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           0.76921    0.27322   2.815  0.00487 **
## age                                   1.97708    0.05796  34.112  < 2e-16 ***
## avg_glucose_level                     0.15705    0.02653   5.920 3.21e-09 ***
## bmi                                   0.02868    0.03915   0.733  0.46384
## hypertension_No_hipertenso           -0.27801    0.08729  -3.185  0.00145 **
## heart_disease_Enf_coronaria_positiva -0.03541    0.10814  -0.327  0.74331
## ever_married_Nunca_casado            -0.19819    0.10799  -1.835  0.06646 .
## work_type_Govt_job                   -1.77523    0.28064  -6.326 2.52e-10 ***
## work_type_Never_worked              -11.06840  168.27645  -0.066  0.94756
## work_type_Private                    -1.37909    0.27144  -5.081 3.76e-07 ***
## work_type_Self_employed              -1.92848    0.28750  -6.708 1.98e-11 ***
## smoking_status_never_smoked          -0.61260    0.07941  -7.715 1.21e-14 ***
## smoking_status_smokes                 0.02773    0.09378   0.296  0.76744
## smoking_status_Unknown               -0.36223    0.09288  -3.900 9.63e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10658.2  on 7688  degrees of freedom
## Residual deviance:  7266.3  on 7675  degrees of freedom
## AIC: 7294.3
##
## Number of Fisher Scoring iterations: 12
```

3.1.b) Matriz de confusion y evaluacion del modelo

```r
predicciones1 <- predict(logistico, newdata = test_dummies_N, type = "response")

predicciones2 <- predict(logistico2, newdata = test_dummies_N, type = "response")
```

```r
predicciones_clases1 <- ifelse(predicciones1 > 0.5, 1, 0)

predicciones_clases2 <- ifelse(predicciones2 > 0.5, 1, 0)

# Crear la matriz de confusión
confusion_matrix1 <- table(Real = test_dummies_N$stroke, Prediccion = predicciones_clases1)

confusion_matrix2 <- table(Real = test_dummies_N$stroke, Prediccion = predicciones_clases2)

# Mostrar la matriz de confusión
```

```r
print(confusion_matrix1)
```

```
##             Prediccion
## Real           0
##   D_negativo 972
##   D_positivo  49
```

```r
print(confusion_matrix2)
```

```
##             Prediccion
## Real           0   1
##   D_negativo 723 249
##   D_positivo  11  38
```

```r
str(confusion_matrix2)
```

```
##  'table' int [1:2, 1:2] 723 11 249 38
##  - attr(*, "dimnames")=List of 2
##   ..$ Real      : chr [1:2] "D_negativo" "D_positivo"
##   ..$ Prediccion: chr [1:2] "0" "1"
```

3.1.c) Evaluacion del modelo

La primera matriz de confusion que toma los valores de las predicciones de la glm realizada con los datos sin SMOTE demuestra que al haber un desbalance tan grande de los datos es necesario hacer SMOTE.

La segunda matriz se ha hecho con los datos sintéticos del SMOTE.

Verdaderos negativos: Cantidad de negativos que fueron clasificados correctamente -> 723 Falsos negativos -> Cantidad de positivos que fueron clasificados incorrectamente como negativos -> 249

Falsos positivos -> Negativos clasificados incorrectamente como positivos Verdaderos positivos -> Positivos clasificados como positivos

Indicadores de evaluacion Metricas para comparar el rendimiento de modelos

Exactitud:: Accuracy = (VP+VN) / Total Sensibilidad = VP / (Total positivos) Especificidad = VN / (Total negativos)

```r
VP <- confusion_matrix2["D_positivo", "1"]
VN <- confusion_matrix2["D_negativo", "0"]
FP <- confusion_matrix2["D_negativo", "1"]
FN <- confusion_matrix2["D_positivo", "0"]

# Calcular la precisión, sensibilidad y especificidad
accuracy <- (VP + VN) / (FP + FN + VP + VN)
sensitivity <- VP / (VP + FN)
specificity <- VN / (VN + FP)

cat("Precisión (Accuracy):", round(accuracy * 100, 2), "%\n")
```

```
## Precisión (Accuracy): 74.53 %
```

```r
cat("Sensibilidad (Sensitivity):", round(sensitivity*100, 2), "%\n")
```

```
## Sensibilidad (Sensitivity): 77.55 %
```

```r
cat("Especificidad (Specificity):", round(specificity*100, 2), "%\n")
```

```
## Especificidad (Specificity): 74.38 %
```

```r
roc_obj <- roc(test_dummies_N$stroke, as.numeric(predicciones_clases2))

#str(roc_obj)

auc_value <- auc(roc_obj)

cat("Área bajo la curva ROC (AUC):", auc_value, "\n")
```

```
## Área bajo la curva ROC (AUC): 0.7596687
```

En resumen, este modelo tiene una precisión aceptable (74.53%) y un rendimiento relativamente equilibrado en términos de sensibilidad y especificidad. El AUC de 0.7596687 sugiere que el modelo tiene un rendimiento mejor que el azar, pero no es extremadamente alto.

3.2) K-Nearest Neighbors

3.2.a) Teoria

- Se buscan K registros con características similares:

En este paso, se identifican los K registros del conjunto de entrenamiento que tienen características más cercanas a las del nuevo registro. Esto se hace utilizando una métrica de distancia, como la distancia euclidiana o la distancia de Mahalanobis.

- Para clasificar, asignamos la clase mayoritaria entre los registros similares al nuevo registro:

Una vez que se han identificado los K registros más cercanos, se observa la clase a la que pertenecen. Luego, se asigna al nuevo registro la clase que es más común entre esos K registros.

- Para pronosticar, hallamos el promedio entre los registros similares y pronosticamos ese promedio para el nuevo registro:

Se calcula el promedio de la variable objetivo de los K registros más cercanos y se asigna ese valor como pronóstico para el nuevo registro.

Metricas de distancias Cuando se trabaja con variables dummy (one-hot encoded) en algoritmos como k-NN, la medida de distancia más comúnmente utilizada es la Distancia de Hamming.

Sin embargo, en algunos casos, cuando hay una combinación de variables numéricas y dummy, se puede considerar una métrica de distancia que combine medidas para ambos tipos de variables. Por ejemplo, la Distancia Euclidiana Ponderada.

Normalizacion

Eleccion de K

3.2.b) Implementando K-N Neighbors

La función que usaremos para K-NN es del paquete FNN.

Definimos cuáles son los valores que vamos a introducir en la funcion

```r
knn_train = X_new
knn_test = test_dummies_N

x_train = knn_train[, -which(names(knn_train) == "stroke")]
x_test = knn_test[, -which(names(knn_test) == "stroke")]

y_train = knn_train$stroke
```

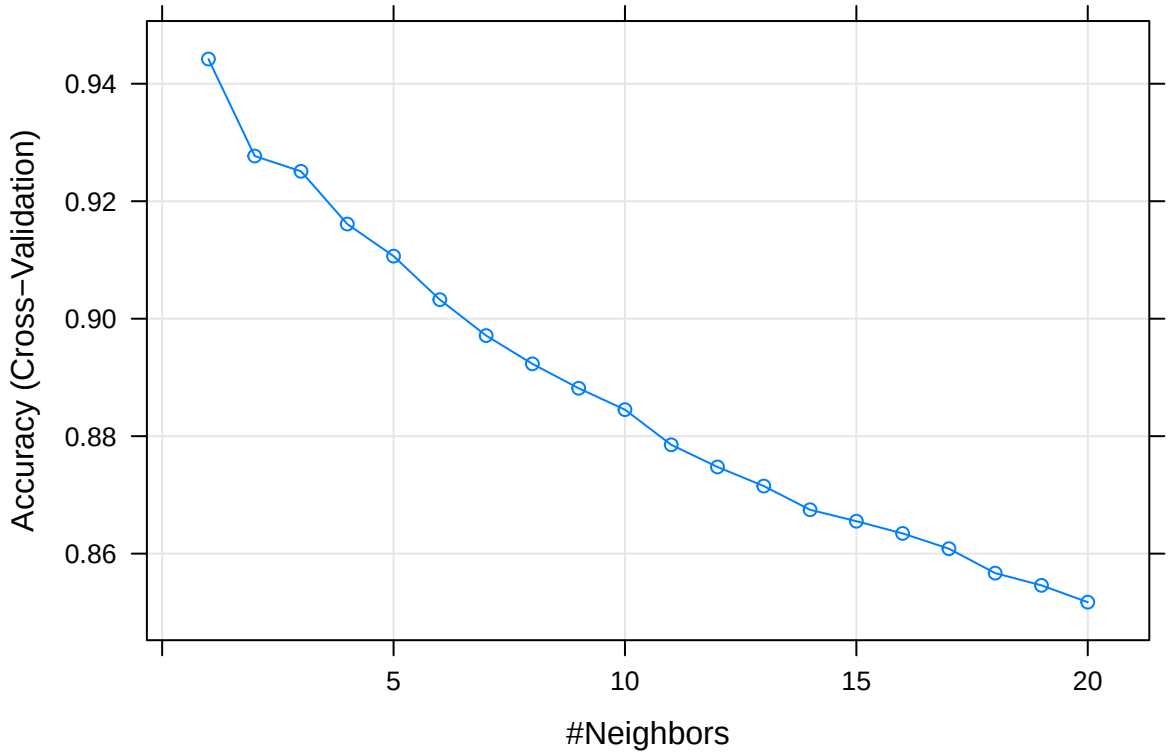
```r
y_test  = knn_test$stroke

#y_train = ifelse(y_train == "D_negativo", 0, 1)
#y_test = ifelse(y_test == "D_negativo", 0, 1)
```

Según la K que tomemos, cambia mucho la accuracy. Asi que vamos a usar un metodo para encontrar la K con mayor accuracy

Usaremos un metodo de validacion cruzada para encontrar la k

```r
set.seed(123)

kfold = trainControl(method = "cv", number = 10)
knn_model = train(x = x_train, y = y_train, method = "knn", trControl = kfold,
                  tuneGrid = expand.grid(k = 1:20))
```

```r
#knn_model

plot(knn_model)
```




```r
knn_stroke <- knn(train = x_train, test = x_test, cl = y_train, k = 1)

confusion_matrix <- confusionMatrix(knn_stroke, y_test)

confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   D_negativo D_positivo
##   D_negativo        887         37
```

```
##   D_positivo         85         12
##
##                Accuracy : 0.8805
##                  95% CI : (0.859, 0.8998)
##     No Information Rate : 0.952
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1075
##
##  Mcnemar's Test P-Value : 2.089e-05
##
##             Sensitivity : 0.9126
##             Specificity : 0.2449
##          Pos Pred Value : 0.9600
##          Neg Pred Value : 0.1237
##              Prevalence : 0.9520
##          Detection Rate : 0.8688
##    Detection Prevalence : 0.9050
##       Balanced Accuracy : 0.5787
##
##        'Positive' Class : D_negativo
##
```

```
accuracy <- confusion_matrix$overall["Accuracy"]
sensitivity <- confusion_matrix$byClass["Sensitivity"]
specificity <- confusion_matrix$byClass["Specificity"]

cat("Precisión (Accuracy):", round(accuracy * 100, 2), "%\n")
```

```
## Precisión (Accuracy): 88.05 %
```

```
cat("Sensibilidad (Sensitivity):", round(sensitivity*100, 2), "%\n")
```

```
## Sensibilidad (Sensitivity): 91.26 %
```

```
cat("Especificidad (Specificity):", round(specificity*100, 2), "%\n")
```

```
## Especificidad (Specificity): 24.49 %
```

```
roc_obj <- roc(test_dummies_N$stroke, as.numeric(knn_stroke))

auc_value <- auc(roc_obj)

cat("Área bajo la curva ROC (AUC):", auc_value, "\n")
```

```
## Área bajo la curva ROC (AUC): 0.5787247
```

La especificidad es muy baja, y el area bajo la curva ROC es muy cercana a 0.5

El AUC es relativamente bajo, lo que sugiere que el modelo podría no estar funcionando tan bien en la clasificación.

3.2.c) Optimizando

Vamos a optimizar la eleccion de K, para que salga la mejor AUC

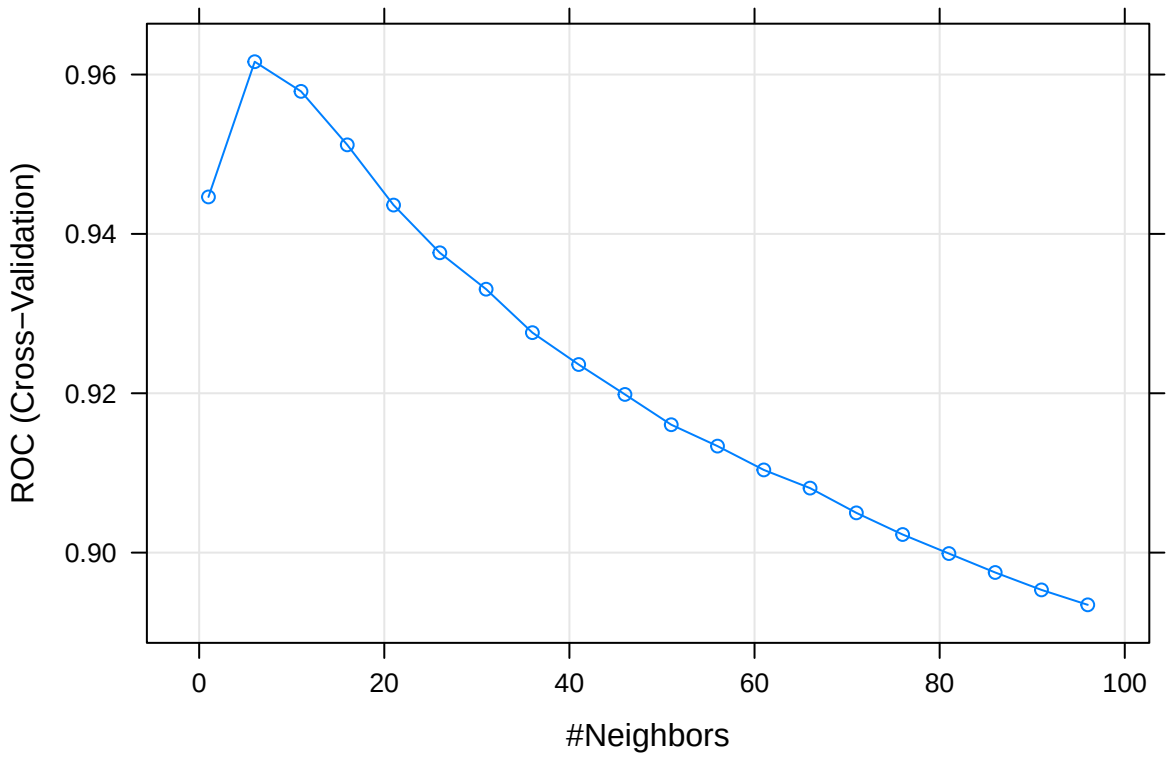
```
set.seed(123)

# Definir el control de entrenamiento
```

```r
kfold = trainControl(method = "cv", number = 10, summaryFunction = twoClassSummary, classProbs = TRUE,
                     savePredictions = TRUE)

# Entrenar el modelo k-NN con optimización de AUC y k de 1 a 100 de 5 en 5
knn_model = train(x = x_train, y = y_train, method = "knn", trControl = kfold,
                  tuneGrid = expand.grid(k = seq(1, 100, by = 5)), metric = "ROC")

# Ver los resultados
print(knn_model)
```

```
## k-Nearest Neighbors
##
## 7689 samples
##   13 predictor
##    2 classes: 'D_negativo', 'D_positivo'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 6920, 6920, 6920, 6920, 6920, 6920, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    1  0.9446403  0.9074384  0.9818421
##    6  0.9616123  0.8182135  0.9884211
##   11  0.9578893  0.7683241  0.9910526
##   16  0.9511745  0.7410668  0.9876316
##   21  0.9436191  0.7220384  0.9831579
##   26  0.9376313  0.7099555  0.9778947
##   31  0.9330596  0.7017233  0.9736842
##   36  0.9276083  0.6909204  0.9652632
##   41  0.9236074  0.6904063  0.9589474
##   46  0.9198521  0.6868080  0.9505263
##   51  0.9160515  0.6806350  0.9457895
##   56  0.9133658  0.6796054  0.9447368
##   61  0.9103795  0.6744633  0.9444737
##   66  0.9080960  0.6731780  0.9434211
##   71  0.9049897  0.6682930  0.9442105
##   76  0.9022725  0.6675225  0.9431579
##   81  0.8998836  0.6644356  0.9436842
##   86  0.8975088  0.6613521  0.9431579
##   91  0.8953228  0.6590378  0.9439474
##   96  0.8934479  0.6572390  0.9439474
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 6.
```

```r
# Obtener la mejor configuración de k
best_k <- knn_model$bestTune$k
```

```r
plot(knn_model)
```



Parece que la mejor nos la da k = 6

3.2.d) Matriz de confusion y evaluacion del modelo

```
knn_stroke <- knn(train = x_train, test = x_test, cl = y_train, k = 6)

confusion_matrix <- confusionMatrix(knn_stroke, y_test)

print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   D_negativo D_positivo
##   D_negativo        827         29
##   D_positivo        145         20
##
##                Accuracy : 0.8296
##                  95% CI : (0.8051, 0.8522)
##     No Information Rate : 0.952
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1219
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8508
##             Specificity : 0.4082
##          Pos Pred Value : 0.9661
##          Neg Pred Value : 0.1212
```

```
##              Prevalence : 0.9520
##          Detection Rate : 0.8100
##    Detection Prevalence : 0.8384
##       Balanced Accuracy : 0.6295
##
##        'Positive' Class : D_negativo
##
```

```
accuracy <- confusion_matrix$overall["Accuracy"]
sensitivity <- confusion_matrix$byClass["Sensitivity"]
specificity <- confusion_matrix$byClass["Specificity"]

cat("Precisión (Accuracy):", round(accuracy * 100, 2), "%\n")
```

```
## Precisión (Accuracy): 82.96 %
```

```
cat("Sensibilidad (Sensitivity):", round(sensitivity*100, 2), "%\n")
```

```
## Sensibilidad (Sensitivity): 85.08 %
```

```
cat("Especificidad (Specificity):", round(specificity*100, 2), "%\n")
```

```
## Especificidad (Specificity): 40.82 %
```

```
roc_obj <- roc(test_dummies_N$stroke, as.numeric(knn_stroke))

auc_value <- auc(roc_obj)

cat("Área bajo la curva ROC (AUC):", auc_value, "\n")
```

```
## Área bajo la curva ROC (AUC): 0.6294932
```

La precision y la sensibilidad han disminuido, pero han aumentado la especificidad y el área bajo la curva ROC

La especificidad sigue siendo relativamente baja, situándose en un 40.82%, lo que indica que el modelo tiene dificultades para identificar correctamente los casos negativos. El AUC de 0.6294932 sugiere que el modelo supera al azar en términos de rendimiento, pero sigue siendo más bajo que la regresion logarítmica

3.3) Arbol de decisión

Los modelos de árbol son un método de clasificación desarrollado por Leo Breiman en 1984.

Los datos se someten a particion repetidamente utilizando valores predictivos que hacen lo mejor posible el trabajo de separar los datos en particiones relativamente homogéneas.

Un árbol completamente desarrollado da como resultado hojas puras. 100% de acierto sobre los datos sobre los que ha sido entrenado => Hemos sobreajustado

Evitamos dividir una partición si una subpartición es demasiado pequeña o una hoja terminal es demasiado pequeña. En rpart estas restricciones se controlan mediante los parámetros minsplit y minbucket. Con valores predeterminados de 20 y 7.

No dividimos una particion si la nueva particion no reduce significativamente laimpureza. En rpart esta decision la controla el parametro de complejidad cp. Es una medida de lo complejo que es el árbol. En la práctica cp se usa para limitar el crecimiento de un árbol agregando una penalización a la complejidad adicional.

El valor predetermiando en rpart es 0.01

3.3.a) Implementando el árbol de decision

```
tree = rpart(stroke ~., data = X_new)
```

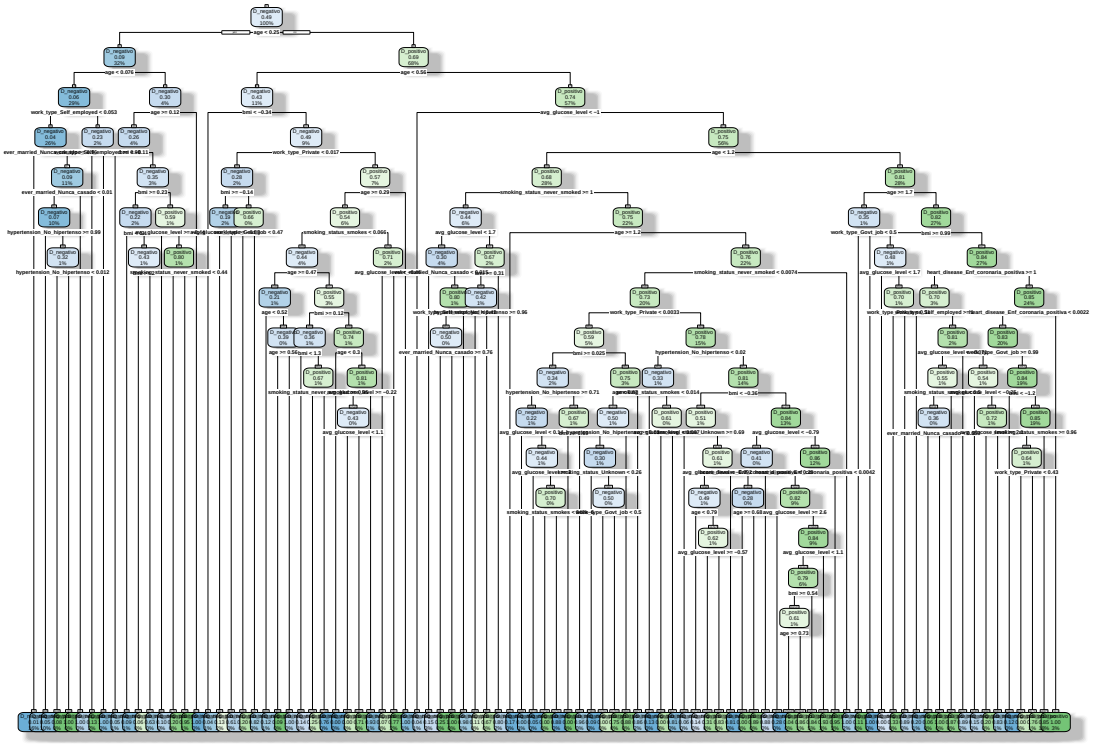
La forma más comun de estimar un buen valor de cp es mediante validación cruzada. En rpart podemos usar el argumento cptable para producir una tabla de valores de cp y su error de validacion cruzada asociado (xerror), a partir del cual podemos determinar el valor de cp que tiene el error de validacion cruzada mas bajo.

```
# Definir el control para el entrenamiento del modelo
control <- trainControl(method = "cv", number = 10)  # Validación cruzada con 10 folds

# Definir la cuadrícula de hiperparámetros para buscar
grid <- expand.grid(cp = seq(0.001, 0.1, by = 0.001))  # Valores de cp a considerar

# Entrenar el modelo con la selección de cp
model <- train(stroke ~ ., data = X_new, method = "rpart", trControl = control, tuneGrid = grid)

# Imprimir el mejor valor de cp
print(paste("El mejor valor de cp es:", model$bestTune$cp))
```

```
## [1] "El mejor valor de cp es: 0.001"
```

3.3.b) Matriz de confusion y evaluacion del modelo

```
tree = rpart(stroke ~. , data = X_new, control = rpart.control(cp = 0.001))


rpart.plot(tree, box.palette="auto", shadow.col="gray", nn=TRUE)
```



```r
predictions <- predict(tree, test_dummies_N, type = "class")

confusion_matrix <- confusionMatrix(predictions, test_dummies_N$stroke)

print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   D_negativo D_positivo
##   D_negativo        793         19
##   D_positivo        179         30
##
##                Accuracy : 0.8061
##                  95% CI : (0.7805, 0.8299)
##     No Information Rate : 0.952
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1679
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8158
##             Specificity : 0.6122
##          Pos Pred Value : 0.9766
##          Neg Pred Value : 0.1435
##              Prevalence : 0.9520
##          Detection Rate : 0.7767
##    Detection Prevalence : 0.7953
##       Balanced Accuracy : 0.7140
##
##        'Positive' Class : D_negativo
##
```

```r
accuracy <- confusion_matrix$overall["Accuracy"]
sensitivity <- confusion_matrix$byClass["Sensitivity"]
specificity <- confusion_matrix$byClass["Specificity"]

cat("Precisión (Accuracy):", round(accuracy * 100, 2), "%\n")
```

```
## Precisión (Accuracy): 80.61 %
```

```r
cat("Sensibilidad (Sensitivity):", round(sensitivity*100, 2), "%\n")
```

```
## Sensibilidad (Sensitivity): 81.58 %
```

```r
cat("Especificidad (Specificity):", round(specificity*100, 2), "%\n")
```

```
## Especificidad (Specificity): 61.22 %
```

```r
roc_obj <- roc(test_dummies_N$stroke, as.numeric(predictions))

auc_value <- auc(roc_obj)

cat("Área bajo la curva ROC (AUC):", auc_value, "\n")
```

```
## Área bajo la curva ROC (AUC): 0.7140443
```

Esto modelos tiene una precisión del 80.61% y una sensibilidad del 81.58% es bastante sólido. Sin embargo, la especificidad del 61.22% es un poco más baja que el primero modelo de regresion logística, lo que significa que el modelo tiene más dificultades para identificar correctamente los casos negativos.

El AUC de 0.714 es relativamente bueno. En general, un AUC por encima de 0.7 indica un rendimiento superior al azar

Los modelos de árboles de decision pueden proporcionar una idea visual para explorar los datos, para tener una idea de que variables son importantes y como se relacionan entre si. Los árboles pueden capturar relaciones no lineales entre variables predictoras.

En cuanto al pronóstico, utilizar varios árboles suele ser más eficaz. Entre este tipo de algoritmos con varios árboles se encuentra uno llamado bosque aleatorio

3.4) Random forest (bosque aleatorio)

3.4.a) Matriz de confusion y evaluacion del modelo

```
r_forest = randomForest(stroke ~. , data = X_new, ntree= 500)

predictions <- predict(r_forest, test_dummies_N, type = "class")

confusion_matrix <- confusionMatrix(predictions, test_dummies_N$stroke)

confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   D_negativo D_positivo
##   D_negativo        885         32
##   D_positivo         87         17
##
##                Accuracy : 0.8834
##                  95% CI : (0.8622, 0.9025)
##     No Information Rate : 0.952
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1679
##
##  Mcnemar's Test P-Value : 7.415e-07
##
##             Sensitivity : 0.9105
##             Specificity : 0.3469
##          Pos Pred Value : 0.9651
##          Neg Pred Value : 0.1635
##              Prevalence : 0.9520
##          Detection Rate : 0.8668
##    Detection Prevalence : 0.8981
##       Balanced Accuracy : 0.6287
##
##        'Positive' Class : D_negativo
##
```

```
accuracy <- confusion_matrix$overall["Accuracy"]
sensitivity <- confusion_matrix$byClass["Sensitivity"]
```

```r
specificity <- confusion_matrix$byClass["Specificity"]

cat("Precisión (Accuracy):", round(accuracy * 100, 2), "%\n")
```

```
## Precisión (Accuracy): 88.34 %
```

```r
cat("Sensibilidad (Sensitivity):", round(sensitivity*100, 2), "%\n")
```

```
## Sensibilidad (Sensitivity): 91.05 %
```

```r
cat("Especificidad (Specificity):", round(specificity*100, 2), "%\n")
```

```
## Especificidad (Specificity): 34.69 %
```

```r
roc_obj <- roc(test_dummies_N$stroke, as.numeric(predictions))

auc_value <- auc(roc_obj)

cat("Área bajo la curva ROC (AUC):", auc_value, "\n")
```

```
## Área bajo la curva ROC (AUC): 0.6287163
```

Es un modelo que no mejora al arbol de decision