MACHINE LEARNING EN RSTUDIO

REGRESION LOGISTICA

EDUARD LARA

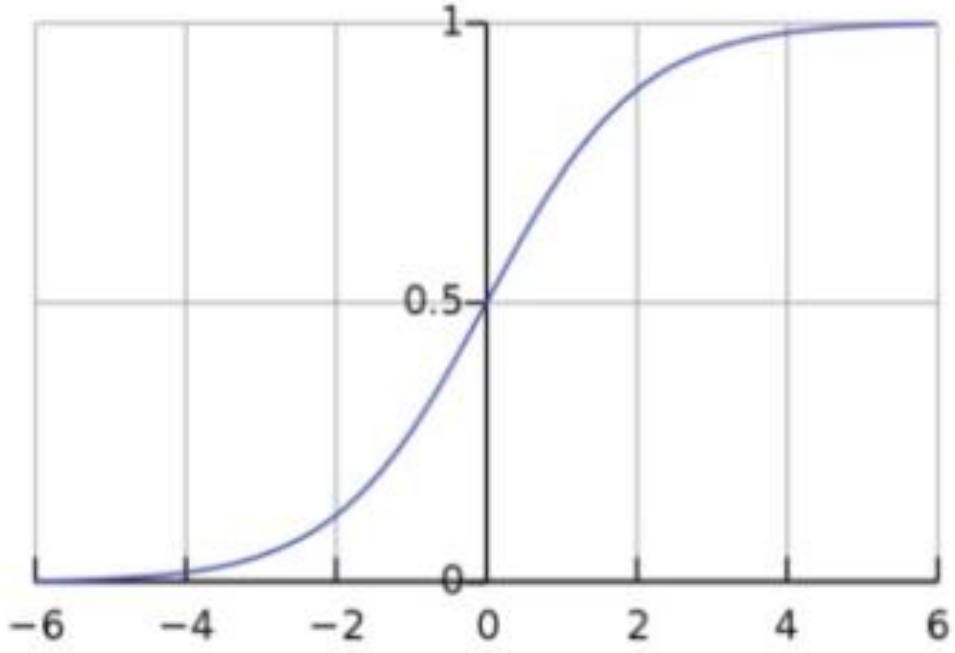
1. INDICE

1. REGRESION LOGISTICA

- La regresión logística es un tipo de análisis de regresión, utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un numero limitado de categorías) en función de otras variables independientes.
- Es útil para modelar la probabilidad de que un evento pueda ocurrir en función de otros factores.
- Es un método de clasificación, por ejemplo, para clasificar los correos según sean validos o no, para clasificar a las personas según tengan o no una enfermedad concreta, o para solicitar un préstamo según lo puedan pagar o no,
- Son ejemplos de clasificaciones binarias en las que sólo hay dos categorías (sí o no) (categoría 1 o categoría 2).

1. REGRESION LOGISTICA

- Aquí vemos un gráfico de la función de regresión logística, donde se recoge cualquier valor del eje X y devolverá siempre un valor entre 0 y 1 en el eje Y
- Si resultado es >= 0.5, la salida será 1
- Si resultado es < 0.5, la salida será 0



1. REGRESION LOGISTICA

Matriz de confusión

- Una matriz de confusión sirve para evaluar nuestro modelo de regresión logística, comparando el valor real con el valor de predicción.
- Si por ejemplo, el valor real es que si y la predicción es que sí, entonces tenemos un positivo correcto,
- Pero si el valor real es que si y nuestra predicción dice que no, entonces tenemos un falso negativo.
- Así podemos ver en qué cosas nos equivocamos y en que cosas acertamos.

1. REGRESION LOGISTICA

Matriz de confusión

- Ejemplo de una matriz de confusión para unas pruebas de detección de spam

correos totals = 200	Predicción (SI es spam)	Predicción (NO es spam)
Valor real = SI es spam	30 (PC)	15 (FN)
Valor real = NO es spam	5 (FP)	150 (NC)

PC (positivos correctos), NC (negativos correctos)

FP (falsos positivos, error tipo 1)

FN (falsos negativos, error tipo 2)

1. REGRESION LOGISTICA

- La precisión sirve para saber la probabilidad de acierto en la predicción.

Precisión = (positivos correctos + negativos correctos)/ Total

Precisión = (20+150)/200 = 0.9 (prob. de acierto del 90%)

- La tasa error sirve para saber la probabilidad de error en la predicción.

Tasa de error = (falsos positivos + falsos negativos)/Total

Tasa de error = (5 + 15)/200 = 0.1 (probabilidad de error del 10%)

2. EJEMPLO REGRESION LOGISTICA I

Paso 1. Utilizar los datos del fichero titanic.csv. Cargamos en la variable datos el fichero csv, que previamente hemos dejado en el directorio workspace. Vemos los primeros elementos y sus columnas

```
> datos = read.csv('titanic.csv')
> head(datos)
  PassengerId Survived Pclass
1           1        0      3
2           2        1      1
3           3        1      3
4           4        1      1
5           5        0      3
6           6        0      3
                                                 Name    Sex Age SibSp Parch
1                             Braund, Mr. Owen Harris   male  22     1     0
2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
3                              Heikkinen, Miss. Laina female  26     0     0
4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
5                            Allen, Mr. William Henry   male  35     0     0
6                                    Moran, Mr. James   male  NA     0     0
            Ticket     Fare Cabin Embarked
1        A/5 21171   7.2500              S
2         PC 17599 71.2833   C85        C
3 STON/O2. 3101282   7.9250              S
4           113803 53.1000  C123        S
5           373450   8.0500              S
6           330877   8.4583              Q
>
```

2. EJEMPLO REGRESION LOGISTICA I

Paso 2. Vemos las columnas: id del pasajero, si ha sobrevivido o no al accidente (0 no ha sobrevivido y 1 ha sobrevivido), el tipo de clase donde viajaba (primera clase, segunda clase), su nombre de la persona, sexo, edad, etc

En total tenemos 891 observaciones en 12 columnas

Data	
datos	891 obs. of 12 variables

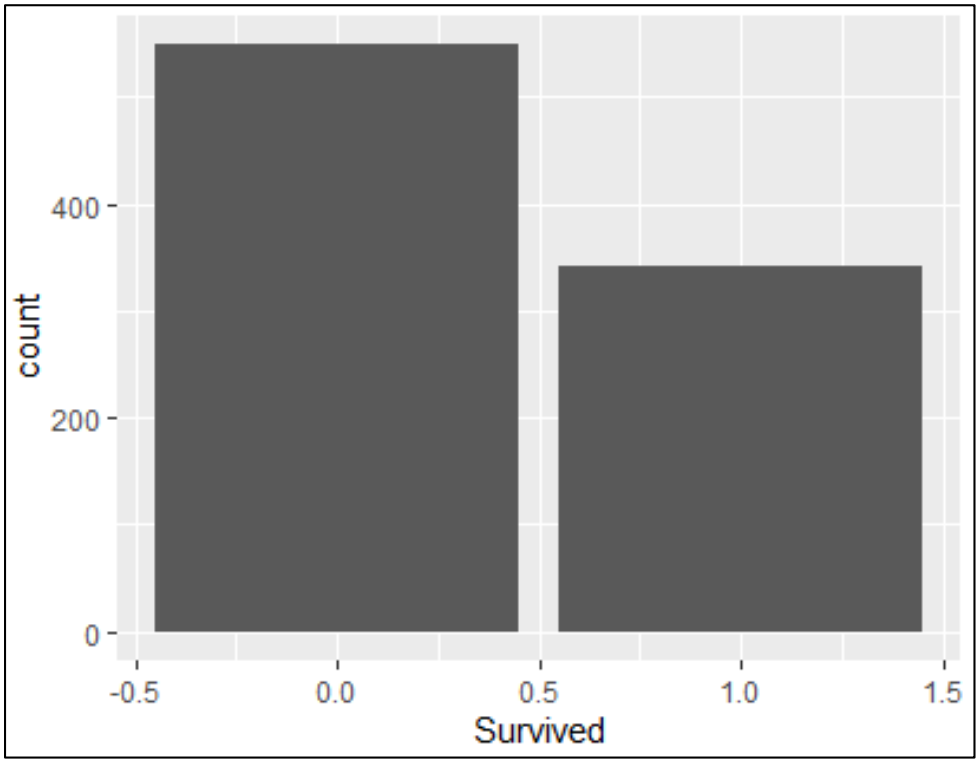
```
> str(datos)
'data.frame':   891 obs. of  12 variables:
 $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
 $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence
nen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
 $ Sex        : chr  "male" "female" "female" "female" ...
 $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
 $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
 $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
 $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
 $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin      : chr  "" "C85" "" "C123" ...
 $ Embarked   : chr  "S" "C" "S" "S" ...
>
```

Mediante str(datos) vemos lo mismo el nº de observaciones, el nº de filas y el nº de columnas que tiene este dataset

2. EJEMPLO REGRESION LOGISTICA I

Paso 3. Haremos una visualización gráfica de la columna Survived Utilizaremos la librería ggplot2.

```
> library(ggplot2)
> ggplot(datos,aes(Survived)) + geom_bar()
>
```

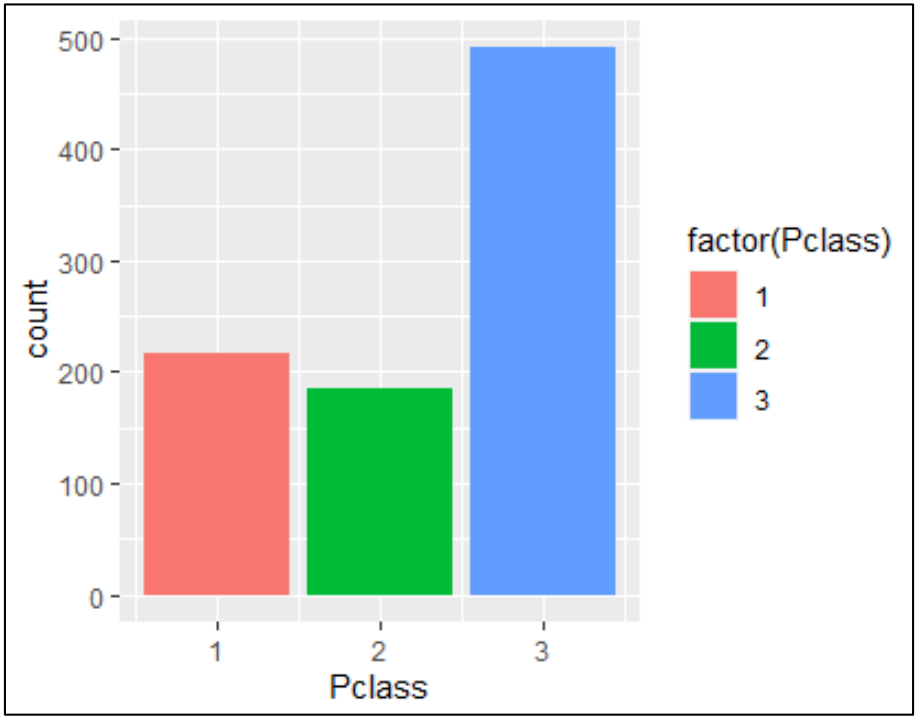


Ha generado un gráfico donde nos dice si ha sobrevivido o no. 0 indica que no ha sobrevivido (unos 600) y 1 es que ha sobrevivido (300 y pico).

2. EJEMPLO REGRESION LOGISTICA I

Paso 4. También podemos ver el tipo de clase en que está cada pasajero, primera, segunda o tercera.

```
> ggplot(datos,aes(Pclass)) + geom_bar(aes(fill=factor(Pclass)))
>
```



En el eje X hemos puesto Pclass y el color de relleno esta factorizado en función del valor de Pclass

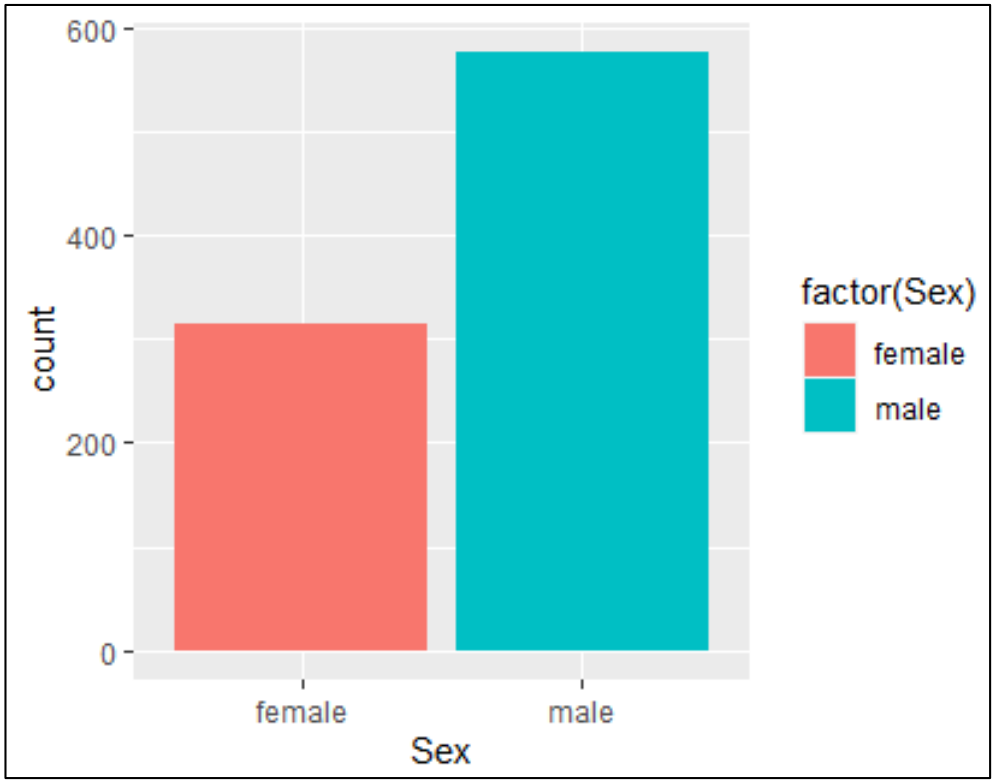
La mayor parte de la gente está en clase 3. En la clase 2 y 3 hay más o menos la misma gente.

La gente que trabaja en el barco pertenece a la Clase 3

2. EJEMPLO REGRESION LOGISTICA I

Paso 5. También podemos ver un gráfico por sexo

```
> ggplot(datos,aes(sex)) + geom_bar(aes(fill=factor(sex)))
>
```

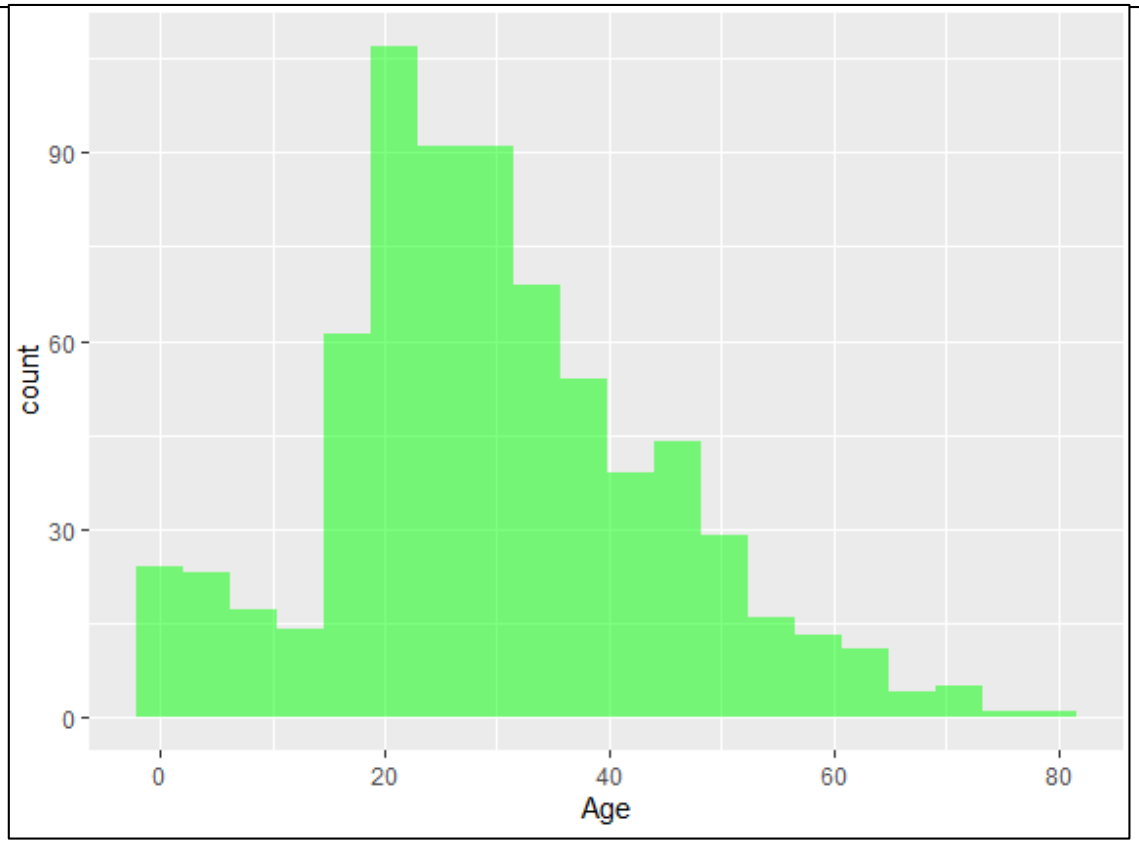


Según sea femenino o masculino, obtenemos el número de personas de cada sexo

2. EJEMPLO REGRESION LOGISTICA I

Paso 6. Podemos ver la edad de los pasajeros.

```
> ggplot(datos,aes(Age)) + geom_histogram(bins=20,alpha=0.5, fill='green')
Warning message:
Removed 177 rows containing non-finite values (`stat_bin()`).
>
```

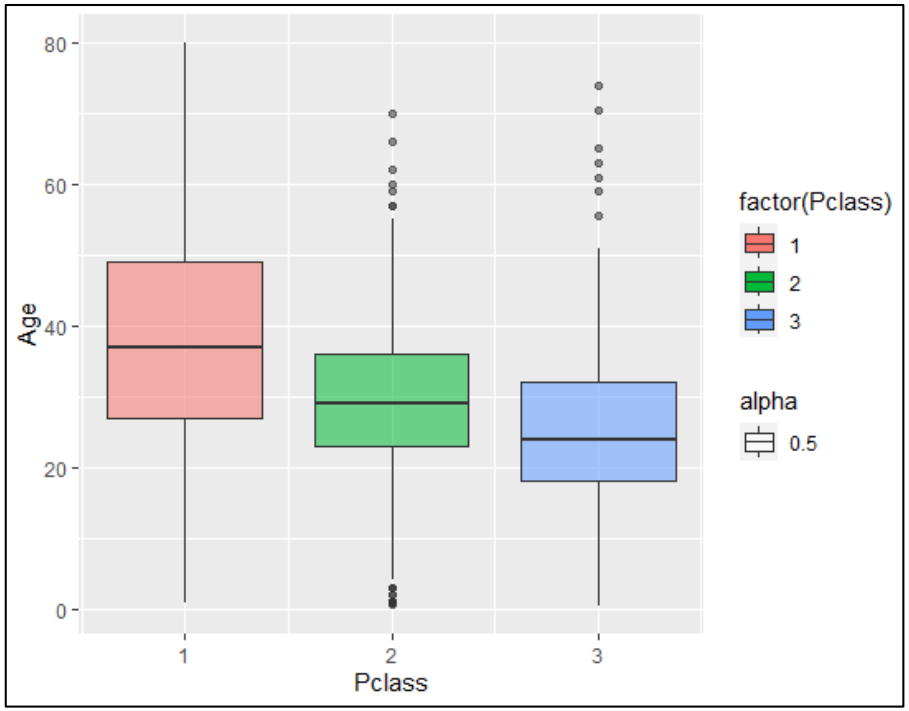


Hemos creado un histograma de la edad de los pasajeros. Hay niños pequeños también. La mayor parte de la gente tiene más de 20 años.

2. EJEMPLO REGRESION LOGISTICA I

Paso 7. Ahora compararemos 2 variables, clase y edad, usando el tipo grafico geom_boxplot

```
> grafico = ggplot(datos,aes(Pclass, Age))
> grafico = grafico + geom_boxplot(aes(group=Pclass, fill=factor(Pclass),alpha=0.5))
> print(grafico)
Warning message:
Removed 177 rows containing non-finite values (`stat_boxplot()`).
>
```



Se ha generado un gráfico de caja agrupado y factorizado por Pclass, donde los pasajeros de la clase 1, tienen entre 30 y 50 años. Los de la clase 2 tienen entre 20 y tantos hasta menos de 40. Y los de la clase 3 tiene la gente más joven entre los 18 y los 35

3. EJEMPLO REGRESION LOGISTICA II

Paso 1. Vamos a instalar un paquete para revisar valores nulos de las columnas de nuestro dataset. Instalamos el paquete Amelia y lo cargamos en memoria.

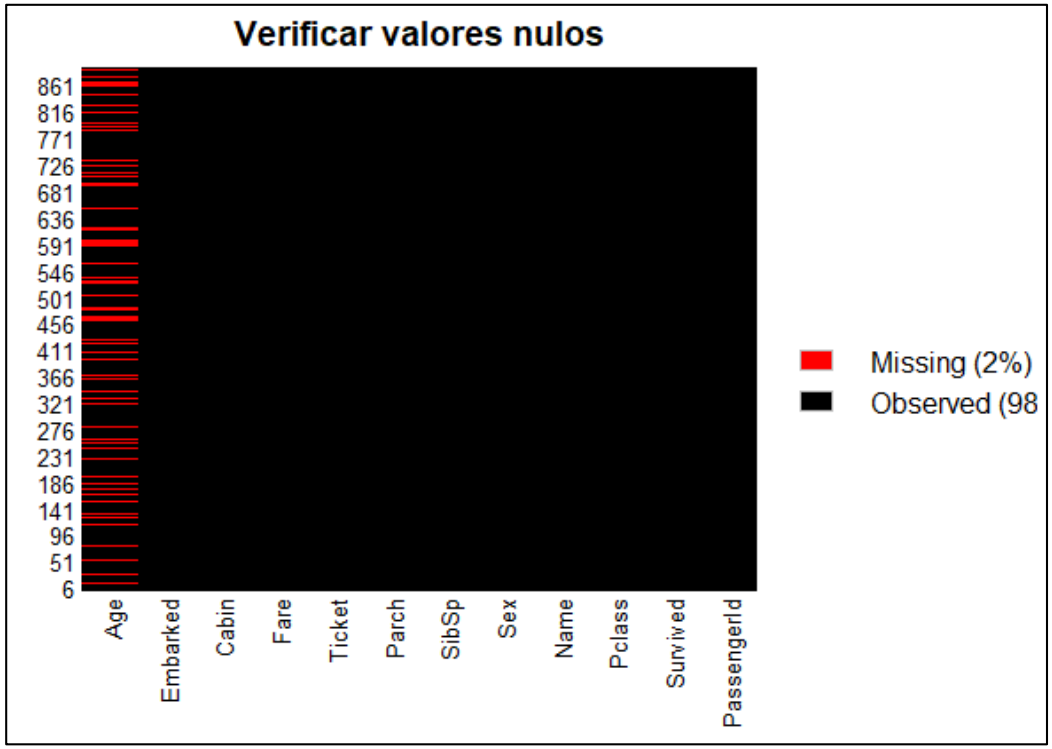
```
> install.packages('Amelia')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
also installing the dependencies 'Rcpp', 'RcppArmadillo'

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/Rcpp_
1.0.12.zip'
Content type 'application/zip' length 2877855 bytes (2.7 MB)
downloaded 2.7 MB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/RcppAr
madillo_0.12.6.6.1.zip'
Content type 'application/zip' length 2050522 bytes (2.0 MB)
downloaded 2.0 MB

probando la URL 'https://cran.rstudio.com
_1.8.1.zip'
Content type 'application/zip' length 187
downloaded 1.8 MB

package 'Rcpp' successfully unpacked and
package 'RcppArmadillo' successfully unpa
package 'Amelia' successfully unpacked an

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\RtmpkzCeFO\downloaded_packages
>
```

```
> library(Amelia)
Loading required package: Rcpp
##
## Amelia II: Multiple Imputation
## (Version 1.8.1, built: 2022-11-18)
## Copyright (C) 2005-2024 James Honaker, Gary King and Matthew Blackwel
l
## Refer to http://gking.harvard.edu/amelia/ for more information
##
>
```

3. EJEMPLO REGRESION LOGISTICA II

Paso 2. Vamos a usar de Amelia la función missmap que nos dice para cada una de las columnas si existen valores nulos

```
> missmap(datos, main="Verificar valores nulos", col=c('red','black'))
>
```



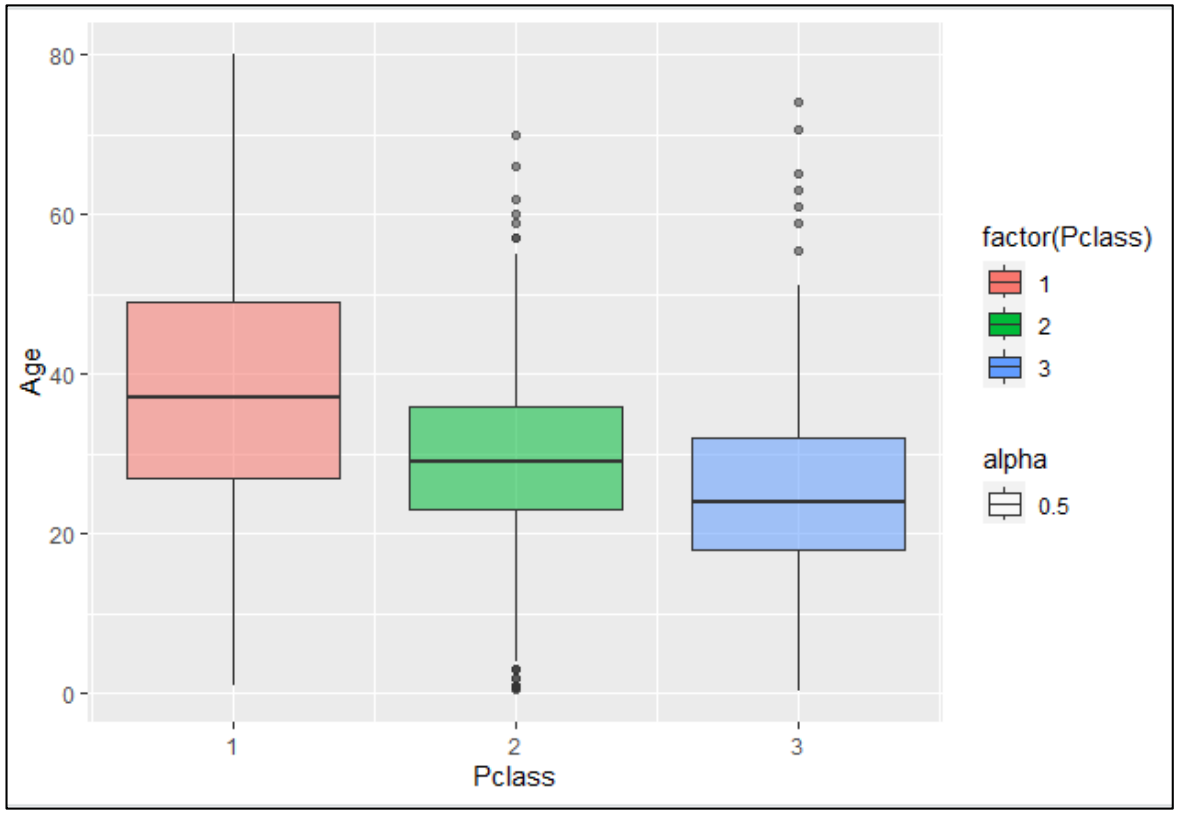
Las columnas de color red tiene un valor nulo, y las columnas de color black tienen un valor que no es nulo

Todas las columnas están en negro, tienen valores no nulos. En cambio, la columna Age tiene muchos valores nulos.

Tenemos que hacer una limpieza para eliminarlos.

3. EJEMPLO REGRESION LOGISTICA II

Paso 3. Si volvemos al gráfico anterior,



A los valores de la edad de la clase 1 que sean nulos, le pondremos el valor medio de 38 (valor medio entre las dos cajas)

Para los valores nulos de edad de la clase 2 le pondremos el valor de 29 y a los de la clase 3, le pondremos el valor medio de 23.

3. EJEMPLO REGRESION LOGISTICA II

Paso 4. Vamos a rellenar esta información que está en rojo, que no tiene valores con el valor medio según la clase,

Utilizaremos esta función para rellenar valores nulos de nuestro dataset

```
edad <- function(edad,clase) {
        salida <- edad
        for (i in 1:length(edad)) {
                if (is.na(edad[i])) {
                        if (clase[i] == 1) {
                                salida[i] <- 38
                        } else if (clase[i] == 2) {
                                salida[i] <- 29
                        } else {
                            salida[i] <- 23
                        }
                } else {
                        salida[i] <- edad[i]
                }
        }
        return(salida)
}
```

3. EJEMPLO REGRESION LOGISTICA II

Paso 5. Le pasamos a la función edad las columnas edad y clase. Asigna la entrada (edad) a la salida y luego hace un bucle for para recorrer todos los valores de la columna

```
>     edad <- function(edad,clase) {
+         salida <- edad
+         for (i in 1:length(edad)) {
+             if (is.na(edad[i])) {
+                 if (clase[i] == 1) {
+                     salida[i] <- 38
+                 } else if (clase[i] == 2) {
+                     salida[i] <- 29
+                 } else {
+                     salida[i] <- 23
+                 }
+             } else {
+                 salida[i] <- edad[i]
+             }
+         }
+         return(salida)
+     }
>
>
```

Si un elemento es nulo, según su clase ponemos en la salida los valores medios de edad antes indicados

Al final la función edad devuelve la variable salida

3. EJEMPLO REGRESION LOGISTICA II

Paso 6. Aplicaremos esta función a las columnas Age y Pclass de datos, que son parámetros de la función edad

```
> edades = edad (datos$Age, datos$Pclass)
> datos$Age = edades
```

Mediante el símbolo $ accedemos a las columnas del dataset datos usando la sintaxis **dataset$columna**

datos$Age → Le pasamos la columna Age del dataset

datos$Pclass → Le pasamos la columna Pclass del dataset

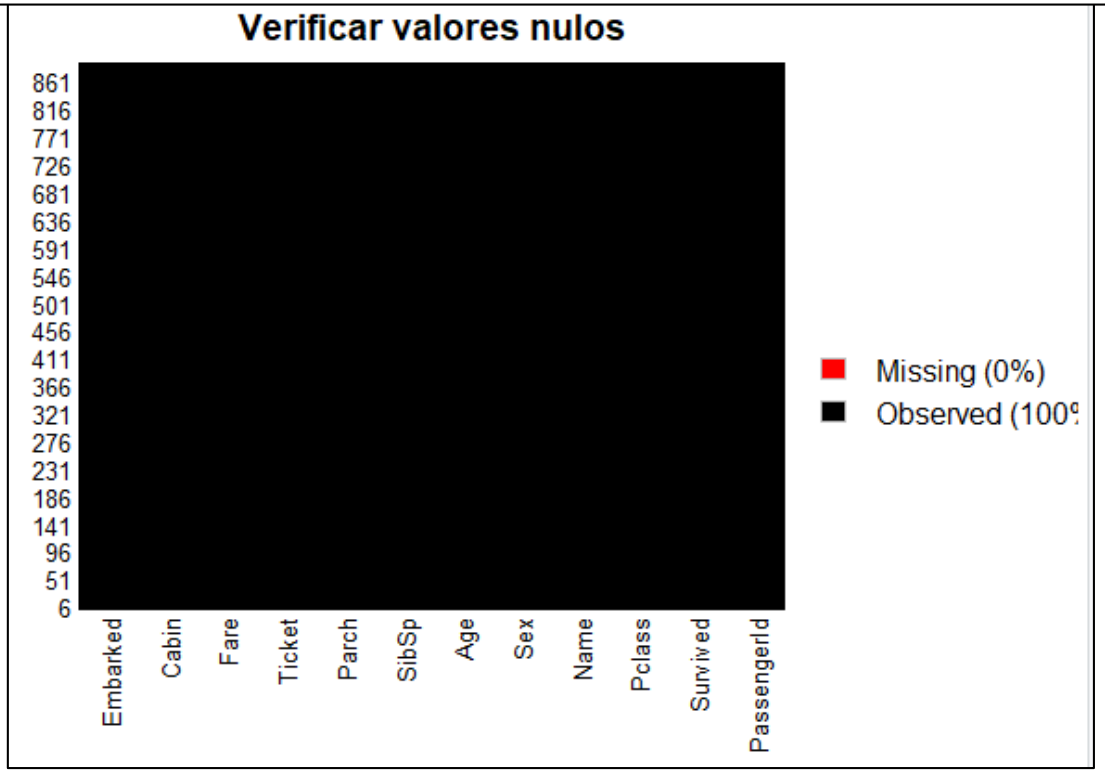
La salida de la función edad devolverá la lista de las edades del dataset rellenadas con datos. Edades contiene los datos de la columna Age limpiados de valores nulos

Finalmente modificamos la columna Age del dataset con los nuevos valores rellenados en la variable edades

3. EJEMPLO REGRESION LOGISTICA II

Paso 7. Si utilizamos la función missmap de nuevo, vemos que se han eliminado los valores rojos y están todos en negro. No hay ningún valor nulo y hemos limpiado los datos

```
> missmap(datos, main="Verificar valores nulos", col=c('red','black'))
>
```



3. EJEMPLO REGRESION LOGISTICA III

Paso 1. Ahora vamos a ejecutar el modelo y realizar las predicciones. Primero quitamos las columnas que no necesitamos. Para ello cargaremos la librería dplyr

```
> library(dplyr)

Attaching package:

The following obje

    filter, lag

The following obje

    intersect, set

>
```

```
> head(datos)
  PassengerId Survived Pclass
1           1        0      3
2           2        1      1
3           3        1      3
4           4        1      1
5           5        0      3
6           6        0      3
                                                 Name    Sex Age SibSp Parch
1                             Braund, Mr. Owen Harris   male  22     1     0
2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
3                              Heikkinen, Miss. Laina female  26     0     0
4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
5                            Allen, Mr. William Henry   male  35     0     0
6                                    Moran, Mr. James   male  23     0     0
            Ticket     Fare Cabin Embarked
1        A/5 21171   7.2500              S
2         PC 17599 71.2833   C85        C
3 STON/O2. 3101282   7.9250              S
4           113803 53.1000  C123        S
5           373450   8.0500              S
6           330877   8.4583              Q
>
```

3. EJEMPLO REGRESION LOGISTICA III

Paso 2. Vamos a eliminar algunas columnas que nos hacen falta : passengerId, name, ticket y cabin.

Para eliminarlas hacemos una selección, con la librería lpyr sobre la variable datos.

```
> datos = select(datos, -PassengerId, -Name, -Ticket, -Cabin)
> head(datos)
  Survived Pclass    Sex Age SibSp Parch    Fare Embarked
1        0      3   male  22     1     0  7.2500        S
2        1      1 female  38     1     0 71.2833        C
3        1      3 female  26     0     0  7.9250        S
4        1      1 female  35     1     0 53.1000        S
5        0      3   male  35     0     0  8.0500        S
6        0      3   male  23     0     0  8.4583        Q
>
```

Si ponemos el nombre de la columna con un menos, indicamos que no la queremos seleccionar.

Si hacemos un head vemos que ya no están esas columnas

3. EJEMPLO REGRESION LOGISTICA III

Paso 3. Ahora vamos a factorizar algunas columnas numéricas, con pocos valores, como Survived, Pclass, Parch, y SibSp. Mediante factor convertiremos la columnas numéricas en categóricas con valores concretos.

```
> datos$Survived  =factor(datos$Survived)
> datos$Pclass = factor(datos$Pclass)
> datos$Parch = factor(datos$Parch)
> datos$SibSp = factor(datos$SibSp)
> str(datos)
'data.frame':   891 obs. of  8 variables:
 $ Survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
 $ Pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
 $ Sex     : chr  "male" "female" "female" "female" ...
 $ Age     : num  22 38 26 35 35 23 54 2 27 14 ...
 $ SibSp   : Factor w/ 7 levels "0","1","2","3",..: 2 2 1 2 1 1 1 4 1 2 ...
 $ Parch   : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 2 3 1 ...
 $ Fare    : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Embarked: chr  "S" "C" "S" "S" ...
> |
```

3. EJEMPLO REGRESION LOGISTICA III

Paso 4. Vamos a dividir los datos del dataset en:

- datos de entrenamiento para entrenar al modelo
- datos de pruebas para realizar las predicciones.

```
> library(caTools)
> set.seed(90)
>
```

Cargamos la librería caTools y establecemos una semilla de 90.

3. EJEMPLO REGRESION LOGISTICA III

Paso 5. Usaremos la función sample.split que viene en caTools. Con ella creamos la variable división que nos va a marcar la división de los datos que vamos a hacer

```
> division = sample.split(datos$Survived, SplitRatio = 0.7)
> entrenamiento = subset(datos, division==TRUE)
> pruebas = subset(datos, division==FALSE)
>
```

La columna Survived es la que queremos estimar o predecir en función de los datos que tenemos. El SplitRatio de 0.7 indica que el 70% se usará para entrenamiento y otro 30% se utilizará para hacer las predicciones.

La variable entrenamiento va a ser un subconjunto del 70% de los datos (división==TRUE) y la variable pruebas tendrá el otro 30% (división==FALSE) que utilizaremos luego para predicciones

3. EJEMPLO REGRESION LOGISTICA III

Paso 6. Hacemos un head de entrenamiento y pruebas, donde vemos las 6 primeras filas de cada una y como el índice va cambiando de forma aleatoria 70% entrenamiento y 30% pruebas.

```
> head(entrenamiento)
  Survived Pclass    Sex Age SibSp Parch    Fare Embarked
1        0      3   male  22     1     0  7.2500        S
2        1      1 female  38     1     0 71.2833        C
3        1      3 female  26     0     0  7.9250        S
4        1      1 female  35     1     0 53.1000        S
5        0      3   male  35     0     0  8.0500        S
6        0      3   male  23     0     0  8.4583        Q
> head(pruebas)
   Survived Pclass    Sex Age SibSp Parch    Fare Embarked
7         0      1   male  54     0     0 51.8625        S
8         0      3   male   2     3     1 21.0750        S
10        1      2 female  14     1     0 30.0708        C
11        1      3 female   4     1     1 16.7000        S
21        0      2   male  35     0     0 26.0000        S
24        1      1   male  28     0     0 35.5000        S
>
```

3. EJEMPLO REGRESION LOGISTICA III

Paso 7. Ahora vamos a entrenar el modelo.

```
> modelo <- glm(Survived  ~ . , family=binomial(link='logit'), data=entrenamiento)
>
```

Generaremos una variable modelo mediante glm donde vamos a predecir la variable de supervivencia, si o no.

Mediante family=binomial(link='logit') le indicamos que va a ser una regresión logística y los datos serán los del entrenamiento.

Así que de esta forma entrenamos el modelo

3. EJEMPLO REGRESION LOGISTICA III

Paso 8. Hacemos un summary del modelo

```
> summary(modelo)

Call:
glm(formula = Survived ~ ., family = binomial(link = "logit"),
    data = entrenamiento)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.006e+01  2.705e+03   0.007 0.994083
Pclass2     -1.306e+00  3.810e-01  -3.427 0.000609 ***
Pclass3     -2.419e+00  3.818e-01  -6.336 2.35e-10 ***
Sexmale     -2.658e+00  2.539e-01 -10.472  < 2e-16 ***
Age         -5.048e-02  1.109e-02  -4.551 5.34e-06 ***
SibSp1       1.905e-02  2.693e-01   0.071 0.943589
SibSp2      -4.462e-01  6.499e-01  -0.687 0.492380
SibSp3      -2.237e+00  7.714e-01  -2.899 0.003740 **
SibSp4      -1.755e+01  1.089e+03  -0.016 0.987142
SibSp5      -1.703e+01  1.742e+03  -0.010 0.992197
SibSp8      -1.746e+01  1.978e+03  -0.009 0.992955
Parch1       6.155e-01  3.567e-01   1.725 0.084444 .
Parch2      -6.186e-02  4.426e-01  -0.140 0.888840
Parch3       5.638e-01  1.060e+00   0.532 0.594843
Parch4      -1.745e+01  2.758e+03  -0.006 0.994951
Parch5      -9.254e-01  1.183e+00  -0.782 0.434101
Parch6      -1.715e+01  3.956e+03  -0.004 0.996541
Fare         1.722e-03  2.712e-03   0.635 0.525421
EmbarkedC   -1.573e+01  2.705e+03  -0.006 0.995361
EmbarkedQ   -1.577e+01  2.705e+03  -0.006 0.995348
EmbarkedS   -1.598e+01  2.705e+03  -0.006 0.995286
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 829.60  on 622  degrees of freedom
Residual deviance: 523.55  on 602  degrees of freedom
AIC: 565.55

Number of Fisher Scoring iterations: 16
```

Nos salen estadísticas del entrenamiento: formula aplicada, los coeficientes, etc

Las columnas que tienen mas asteriscos, son las columnas clave que el modelo considera más importantes para hacer una predicción correcta: la clase 2, la clase 3, el sexo y la edad.

3. EJEMPLO REGRESION LOGISTICA III

Paso 9. Ahora vamos a realizar las predicciones.

```
> predicciones = predict(modelo, pruebas, type='response')
> head(predicciones)
        7         8        10        11        21        24
0.2283564 0.0638084 0.9160004 0.8927686 0.1667349 0.5166890
>
```

Creamos una variable predicciones usando la función predict y pasándole el modelo que acabamos de entrenar, junto con los datos de prueba (30%) y type=response

Con un head vemos las predicciones de las 6 primeras filas. Tenemos valores numéricos entre 0 y 1: cercano a 0 es que no ha sobrevivido y cercano al 1 es que ha sobrevivido)

- Persona 10 con 0.916 → muy cerca de haber sobrevivido
- Persona 8 con 0.06 → muy cerca de no haber sobrevivido

3. EJEMPLO REGRESION LOGISTICA III

Paso 10. Como se trata de una regresión logística, en lugar de poner los valores numéricos predichos, podemos poner "1" si se salva o "0" si no se salva

```
> resultados = ifelse(predicciones > 0.5, 1, 0)
>
> head(resultados)
 7  8 10 11 21 24
 0  0  1  1  0  1
>
```

Con la función **ifelse**, la variable resultados sólo va a tener

- "1" si los valores de predicciones son mayores que 0.5 (indica que sobrevive)
- "0" si esos valores son menores a 0.5 (indica que no sobrevive)

3. EJEMPLO REGRESION LOGISTICA III

Paso 11. Esto se corresponde mas con el original.

```
> head(resultados)
 7  8 10 11 21 24
 0  0  1  1  0  1
> head(datos)
  Survived Pclass    Sex Age SibSp Parch    Fare Embarked
1        0      3   male  22     1     0  7.2500        S
2        1      1 female  38     1     0 71.2833        C
3        1      3 female  26     0     0  7.9250        S
4        1      1 female  35     1     0 53.1000        S
5        0      3   male  35     0     0  8.0500        S
6        0      3   male  23     0     0  8.4583        Q
>
```

Los valores de supervivencia se han normalizado a los datos del dataset original

No podemos comparar visualmente nuestra predicción puesto que vemos personas diferentes en ambas muestras

No podemos decirque sean buenas nuestras predicciones

3. EJEMPLO REGRESION LOGISTICA III

Paso 12. Para calcular la precisión del modelo, calcularemos el error en la tasa de error. La variable error será la media de los valores de los resultados predichos que son distintos de los valores reales

```
> error = mean(resultados!=pruebas$survived)
> error
[1] 0.1791045
> precision = 1 - error
> precision
[1] 0.8208955
>
```

El error es de 0.17 y la precisión final = 1- error = 0.82. Se trata de una precisión del 81%. Este modelo acierta el 81% de las veces a la hora de predecir si un pasajero, en función de las características que tiene, va a sobrevivir o no al accidente del Titanic