Actividad Integradora 2

Héctor Hibran Tapia Fernández - A01661114

2024-11-19

Titanic

Base de datos del Titanic: Titanic.csv Base de datos de prueba: Titanic_test.csv

Las variables para la base de datos son las siguientes:

Name: Nombre del pasajero *PassengerId:* Ids del pasajero *Survived:* Si sobrevivió o no (No = 0, Sí = 1)
Ticket: Número de ticket
Cabin: Cabina en la que viajó
Pclass: Clase en la que viajó (1 = 1era, 2 = 2da, 3 = 3ra)
Sex: Masculino o Femenino (male/female)
Age: Edad
SibSp: Número de hermanos/conyuge a bordo
Parch: Número de padres/hijos a bordo
Fare: Tarifa que pagó
Embarked: Puerto de embarcación (C = Cherbourg, Q = Queenstown, S = Southampton)

1. Prepara la base de datos Titanic:

```
M = read.csv("Titanic.csv")
str(M)
```

```
## 'data.frame':    1309 obs. of  12 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Survived   : int  0 1 0 0 1 0 1 0 1 0 ...
##  $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
##  $ Name       : chr  "Kelly, Mr. James" "Wilkes, Mrs. James (Ellen Needs)" "Myles, M
r. Thomas Francis" "Wirz, Mr. Albert" ...
##  $ Sex        : chr  "male" "female" "male" "male" ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Ticket     : chr  "330911" "363272" "240276" "315154" ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Cabin      : chr  "" "" "" "" ...
##  $ Embarked   : chr  "Q" "S" "Q" "S" ...
```

Ajustando las variables

Variables de interés: Quitamos aquellas variables que de entrada no tengan que ver con la sobrevivencia del pasajero. Variables 4, 9 y 11.

```
M1 <- M[,c(-4,-9,-11)]
```

Ahora, las variables categóricas deben aparecer como factores: Survived, Pclass, Sex y Embarked

```
for(var in c('Survived','Pclass','Embarked','Sex'))
  M1[,var] <-as.factor(M1[,var])
```

```
#M
```

Análisis de datos faltantes

Detectamos si hay espacios vacíos en lugar de datos:

```
V = matrix(NA,ncol=1,nrow=9)
for(i in c(1:9)){
  V[i,] <- sum(with(M1,M1[,i])=="")}
V
```

```
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]   NA
##  [6,]    0
##  [7,]    0
##  [8,]   NA
##  [9,]   NA
```

Ninguna variable tiene espacios vacíos, pero las variables 5 (Age), 8 (Fare) y 9 (Embarked) tienen datos faltantes.

Vamos a contar los datos faltantes:

```
N = apply(X=is.na(M1),MARGIN = 2,FUN = sum)
P = round(100*N/length(M1[,2]),2)
NP = data.frame(as.numeric(N),as.numeric(P))
row.names(NP)= c("PassengerId", "Survived", "Pclass", "Sex", "Age", "SibSp", "Parch", "F
are", "Embarked")
names(NP)=c("Número","Porcentaje")
t(NP)
```

```
##            PassengerId Survived Pclass Sex    Age SibSp Parch Fare Embarked
## Número               0        0      0   0 263.00     0     0 1.00     2.00
## Porcentaje           0        0      0   0  20.09     0     0 0.08     0.15
```

En edad hay muchos datos faltantes, el 20% de los datos.

Realiza un análisis descriptivo

Observemos el patrón de los datos faltantes:

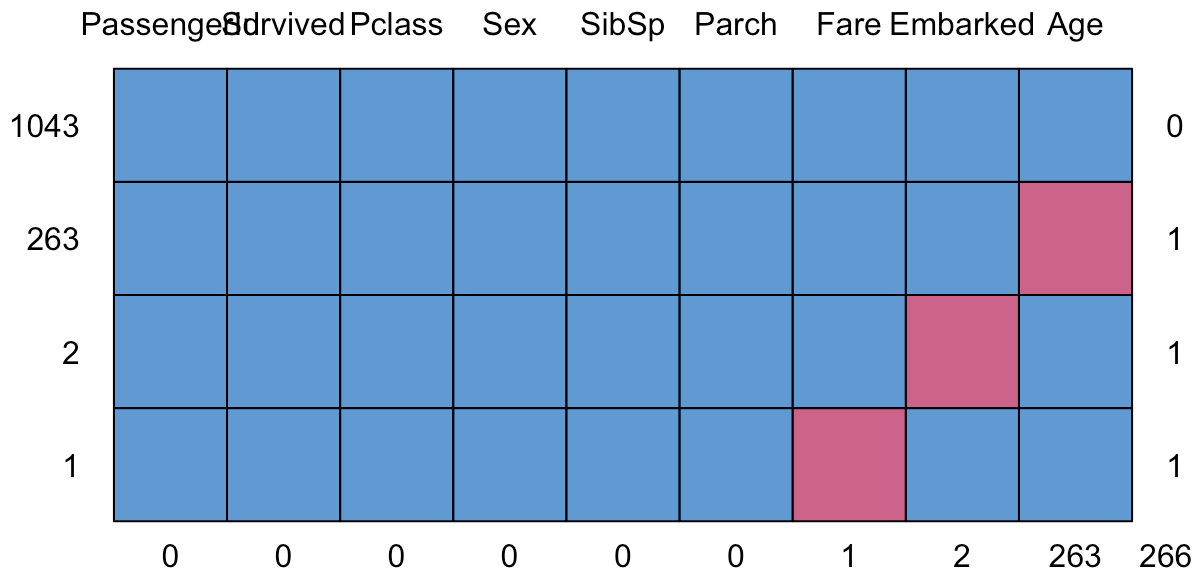
```
# install.packages("mice", type = "binary")
library("mice")
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

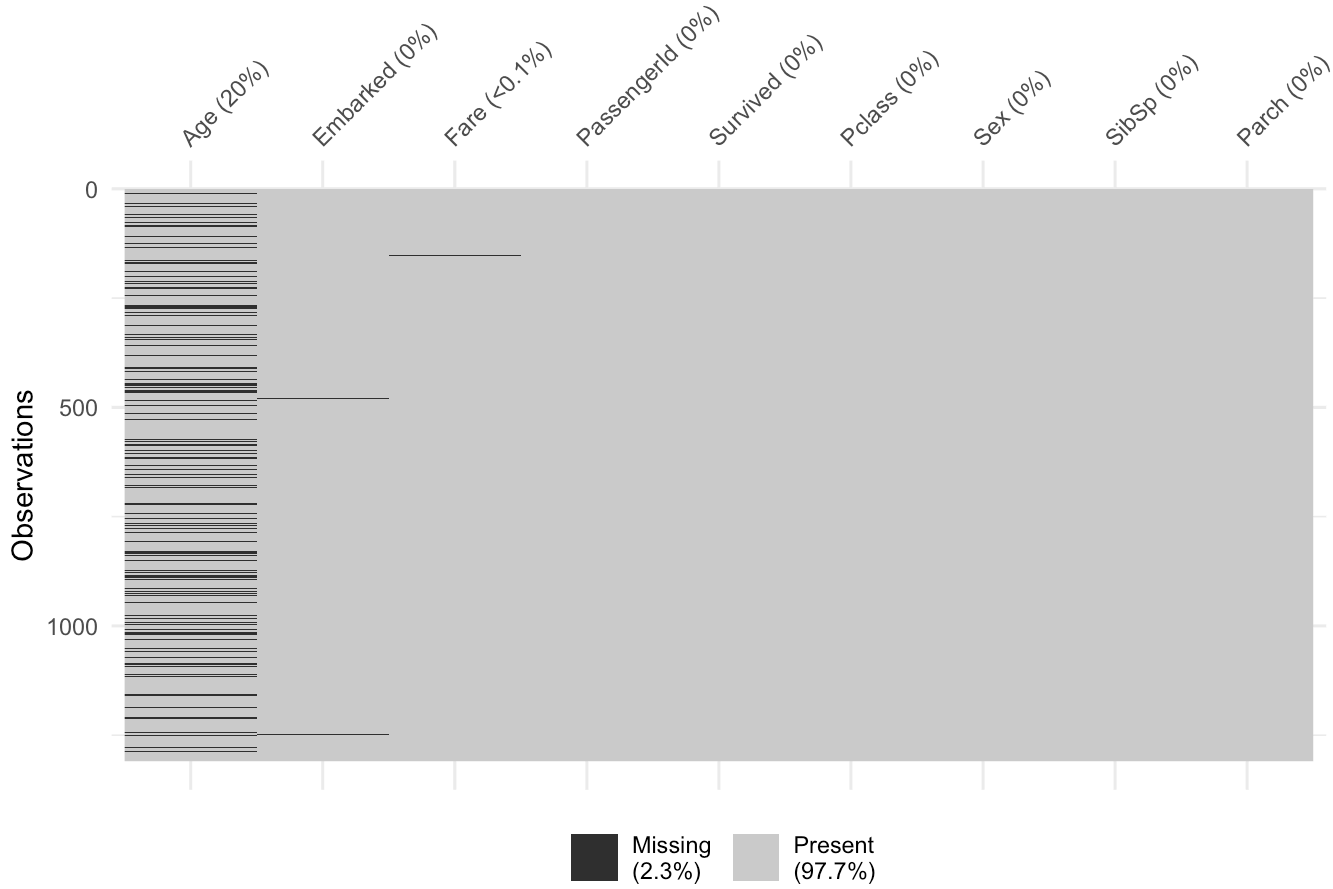
```
md.pattern(M1)
```



```
##      PassengerId Survived Pclass Sex SibSp Parch Fare Embarked Age
## 1043           1        1      1   1     1     1    1        1   1   0
## 263            1        1      1   1     1     1    1        1   0   1
## 2              1        1      1   1     1     1    1        0   1   1
## 1              1        1      1   1     1     1    0        1   1   1
##                0        0      0   0     0     0    1        2 263 266
```

Todos los datos faltantes son de distintos pasajeros (observaciones), por lo tanto, si se eliminan los NA, se eliminarían 266 observaciones y nos quedaríamos con 1043 observaciones.

```
#install.packages("naniar")
library(naniar)
vis_miss(M1,sort_miss = TRUE)
```



Análisis sobre datos faltantes

Medidas con datos faltantes

```
summary(M1[,-1])
```

```
##  Survived Pclass      Sex           Age            SibSp           Parch
##  0:815    1:323   female:466   Min.   : 0.17   Min.   :0.0000   Min.   :0.000
##  1:494    2:277   male  :843   1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000
##           3:709                Median :28.00   Median :0.0000   Median :0.000
##                                Mean   :29.88   Mean   :0.4989   Mean   :0.385
##                                3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:0.000
##                                Max.   :80.00   Max.   :8.0000   Max.   :9.000
##                                NA's   :263
##       Fare          Embarked
##  Min.   :  0.000   C   :270
##  1st Qu.:  7.896   Q   :123
##  Median : 14.454   S   :914
##  Mean   : 33.295   NA's:  2
##  3rd Qu.: 31.275
##  Max.   :512.329
##  NA's   :1
```

Medidas sin datos faltantes

```
M2 = na.omit(M1)
summary(M2[,-1])
```

```
##  Survived Pclass      Sex           Age            SibSp
##  0:628    1:282   female:386   Min.   : 0.17   Min.   :0.0000
##  1:415    2:261   male  :657   1st Qu.:21.00   1st Qu.:0.0000
##           3:500                Median :28.00   Median :0.0000
##                                Mean   :29.81   Mean   :0.5043
##                                3rd Qu.:39.00   3rd Qu.:1.0000
##                                Max.   :80.00   Max.   :8.0000
##      Parch             Fare        Embarked
##  Min.   :0.0000   Min.   :  0.00   C:212
##  1st Qu.:0.0000   1st Qu.:  8.05   Q: 50
##  Median :0.0000   Median : 15.75   S:781
##  Mean   :0.4219   Mean   : 36.60
##  3rd Qu.:1.0000   3rd Qu.: 35.08
##  Max.   :6.0000   Max.   :512.33
```

Sobrevivientes

```
t2c = 100*prop.table(table(M1[,2]))
t2s = 100*prop.table(table(M2[,2]))
t2p = c(t2s[1]/t2c[1],t2s[2]/t2c[2])
t2 = data.frame(as.numeric(t2c),as.numeric(t2s),as.numeric(t2p))
row.names(t2) = c("Murió","Sobrevivió")
names(t2) = c("Con NA (%)","Sin NA (%)","Pérdida (prop)")
round(t2,2)
```

```
##           Con NA (%) Sin NA (%) Pérdida (prop)
## Murió           62.26      60.21           0.97
## Sobrevivió      37.74      39.79           1.05
```

Clase en que viajó

```
t3c = 100*prop.table(table(M1[,3]))
t3s = 100*prop.table(table(M2[,3]))
t3p = c(t3s[1]/t3c[1],t3s[2]/t3c[2],t3s[3]/t3c[3])
t3 = data.frame(as.numeric(t3c),as.numeric(t3s),as.numeric(t3p))
row.names(t3) = c("Primera","Segunda","Tercera")
names(t3) = c("Con NA (%)","Sin NA (%)","Pérdida (prop)")
round(t3,2)
```

```
##         Con NA (%) Sin NA (%) Pérdida (prop)
## Primera      24.68      27.04           1.10
## Segunda      21.16      25.02           1.18
## Tercera      54.16      47.94           0.89
```

Sexo

```
t4c = 100*prop.table(table(M1[,4]))
t4s = 100*prop.table(table(M2[,4]))
t4p = c(t4s[1]/t4c[1],t4s[2]/t4c[2])
t4 = data.frame(as.numeric(t4c),as.numeric(t4s),as.numeric(t4p))
row.names(t4) = c("Mujer","Hombre")
names(t4) = c("Con NA (%)","Sin NA (%)","Pérdida (prop)")
round(t4,2)
```

```
##        Con NA (%) Sin NA (%) Pérdida (prop)
## Mujer        35.6      37.01           1.04
## Hombre       64.4      62.99           0.98
```

Puerto de embarcación

```
t9c = 100*prop.table(table(M1[,9]))
t9s = 100*prop.table(table(M2[,9]))
t9p = c(t9s[1]/t9c[1],t9s[2]/t9c[2],t9s[3]/t9c[3])
t9 = data.frame(as.numeric(t9c),as.numeric(t9s),as.numeric(t9p))
row.names(t9) = c("Cherbourg","Queenstown","Southampton")
names(t9) = c("Con NA (%)","Sin NA (%)","Pérdida (prop)")
round(t9,2)
```

```
##             Con NA (%) Sin NA (%) Pérdida (prop)
## Cherbourg        20.66      20.33           0.98
## Queenstown        9.41       4.79           0.51
## Southampton      69.93      74.88           1.07
```

```
table(M1$Survived) / nrow(M1) # Proporción de sobrevivientes
```

```
##
##         0         1
## 0.6226127 0.3773873
```

```
table(M1$Pclass) / nrow(M1)   # Proporción por clase
```

```
##
##         1         2         3
## 0.2467532 0.2116119 0.5416348
```

```
table(M1$Sex) / nrow(M1)      # Proporción por sexo
```

```
##
##    female      male
## 0.3559969 0.6440031
```

```
table(M1$Embarked) / nrow(M1) # Proporción por puerto de embarque
```

```
##
##          C          Q          S
## 0.20626432 0.09396486 0.69824293
```

Haz una partición de los datos (70-30) para el entrenamiento y la validación. Revisa la proporción de sobrevivientes para la partición y la base original.

```
set.seed(092784)
train_indices = sample(1:nrow(M1), size = 0.7 * nrow(M1))

train_data = M1[train_indices, ]
test_data = M1[-train_indices, ]

original_survived = table(M1$Survived) / nrow(M1)
train_survived = table(train_data$Survived) / nrow(train_data)
test_survived = table(test_data$Survived) / nrow(test_data)
```

```
train_data = na.omit(train_data)
```

2. Con la base de datos de entrenamiento, encuentra un modelo logístico para encontrar el mejor conjunto de predictores que auxilien a clasificar la dirección de cada observación.

```
full_model = glm(Survived ~ ., data = train_data, family = binomial) # Modelo con todas las variables
summary(full_model)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3055  -0.5154  -0.3403   0.4437   2.6266
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.0033983  0.6492179   7.707 1.29e-14 ***
## PassengerId -0.0003976  0.0003010  -1.321 0.186489
## Pclass2     -1.2462256  0.3681576  -3.385 0.000712 ***
## Pclass3     -2.1602908  0.3807241  -5.674 1.39e-08 ***
## Sexmale     -3.8826706  0.2739124 -14.175  < 2e-16 ***
## Age         -0.0435821  0.0091479  -4.764 1.90e-06 ***
## SibSp       -0.4902564  0.1509318  -3.248 0.001161 **
## Parch       -0.1470645  0.1323854  -1.111 0.266619
## Fare         0.0026008  0.0028917   0.899 0.368449
## EmbarkedQ   -0.1708229  0.5905027  -0.289 0.772364
## EmbarkedS    0.0307164  0.3077877   0.100 0.920505
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 969.16  on 728  degrees of freedom
## Residual deviance: 536.72  on 718  degrees of freedom
## AIC: 558.72
##
## Number of Fisher Scoring iterations: 5
```

Auxiliate del criterio de AIC para determinar cuál es

el mejor modelo.

```
best_model = step(full_model, direction = "both") # Modelo usando Stepwise (Backward y Forward)
```

```
## Start:  AIC=558.72
## Survived ~ PassengerId + Pclass + Sex + Age + SibSp + Parch +
##     Fare + Embarked
##
##               Df Deviance    AIC
## - Embarked     2   536.86 554.86
## - Fare         1   537.57 557.57
## - Parch        1   537.96 557.96
## - PassengerId  1   538.47 558.47
## <none>             536.72 558.72
## - SibSp        1   548.13 568.13
## - Age          1   561.30 581.30
## - Pclass       2   570.03 588.03
## - Sex          1   865.02 885.02
##
## Step:  AIC=554.86
## Survived ~ PassengerId + Pclass + Sex + Age + SibSp + Parch +
##     Fare
##
##               Df Deviance    AIC
## - Fare         1   537.71 553.71
## - Parch        1   538.05 554.05
## - PassengerId  1   538.75 554.75
## <none>             536.86 554.86
## + Embarked     2   536.72 558.72
## - SibSp        1   548.17 564.17
## - Age          1   561.69 577.69
## - Pclass       2   572.35 586.35
## - Sex          1   870.56 886.56
##
## Step:  AIC=553.71
## Survived ~ PassengerId + Pclass + Sex + Age + SibSp + Parch
##
##               Df Deviance    AIC
## - Parch        1   538.56 552.56
## - PassengerId  1   539.47 553.47
## <none>             537.71 553.71
## + Fare         1   536.86 554.86
## + Embarked     2   537.57 557.57
## - SibSp        1   548.43 562.43
## - Age          1   563.42 577.42
## - Pclass       2   602.50 614.50
## - Sex          1   877.07 891.07
##
## Step:  AIC=552.56
## Survived ~ PassengerId + Pclass + Sex + Age + SibSp
##
##               Df Deviance    AIC
## - PassengerId  1   540.30 552.30
## <none>             538.56 552.56
## + Parch        1   537.71 553.71
## + Fare         1   538.05 554.05
```

```
## + Embarked     2   538.45 556.45
## - SibSp        1   551.69 563.69
## - Age          1   564.05 576.05
## - Pclass       2   603.84 613.84
## - Sex          1   887.18 899.18
##
## Step:  AIC=552.3
## Survived ~ Pclass + Sex + Age + SibSp
##
##               Df Deviance    AIC
## <none>             540.30 552.30
## + PassengerId  1   538.56 552.56
## + Parch        1   539.47 553.47
## + Fare         1   539.88 553.88
## + Embarked     2   540.10 556.10
## - SibSp        1   552.34 562.34
## - Age          1   565.05 575.05
## - Pclass       2   604.23 612.23
## - Sex          1   887.21 897.21
```

```
summary(best_model)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp, family = binomial,
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2283  -0.4944  -0.3428   0.4607   2.5900
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.801075   0.513306   9.353  < 2e-16 ***
## Pclass2     -1.360143   0.313370  -4.340 1.42e-05 ***
## Pclass3     -2.319298   0.308325  -7.522 5.38e-14 ***
## Sexmale     -3.788652   0.256399 -14.776  < 2e-16 ***
## Age         -0.043149   0.009025  -4.781 1.74e-06 ***
## SibSp       -0.470568   0.141493  -3.326 0.000882 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 969.16  on 728  degrees of freedom
## Residual deviance: 540.30  on 723  degrees of freedom
## AIC: 552.3
##
## Number of Fisher Scoring iterations: 5
```

Propón por lo menos los dos que consideres mejores modelos.

```
model1 = best_model

model2 = glm(Survived ~ Pclass + Sex + Age + SibSp, data = train_data, family = binomia
l)  # Modelo con predictores clave

cat("\nAIC de los modelos:\n")
```

```
##
## AIC de los modelos:
```

```
cat("Modelo 1:", AIC(model1), "\n")
```

```
## Modelo 1: 552.2954
```

```
cat("Modelo 2:", AIC(model2), "\n")
```

```
## Modelo 2: 552.2954
```

3. Analiza los modelos a través de:

Identificación de la Desviación residual y Nula de cada modelo

```
null_deviance = full_model$null.deviance
residual_deviance = best_model$deviance

cat("Desviación nula:", null_deviance, "\n")
```

```
## Desviación nula: 969.1594
```

```
cat("Desviación residual:", residual_deviance, "\n")
```

```
## Desviación residual: 540.2954
```

Cálculo de la Desviación Explicada

```
deviance_explained = (null_deviance - residual_deviance) / null_deviance
cat("Proporción de desviación explicada:", round(deviance_explained * 100, 2), "%\n")
```

```
## Proporción de desviación explicada: 44.25 %
```

Prueba de la razón de verosimilitud

```
lr_test = null_deviance - residual_deviance
df_diff = full_model$df.null - best_model$df.residual
p_value_lr = pchisq(lr_test, df = df_diff, lower.tail = FALSE)

cat("Prueba de la razón de verosimilitud:\n")
```

```
## Prueba de la razón de verosimilitud:
```

```
cat("  Chi-cuadrado:", lr_test, "\n")
```

```
##   Chi-cuadrado: 428.864
```

```
cat("  Grados de libertad:", df_diff, "\n")
```

```
##   Grados de libertad: 5
```

```
cat("  Valor-p:", p_value_lr, "\n")
```

```
##   Valor-p: 1.777038e-90
```

Define cuál es el mejor modelo

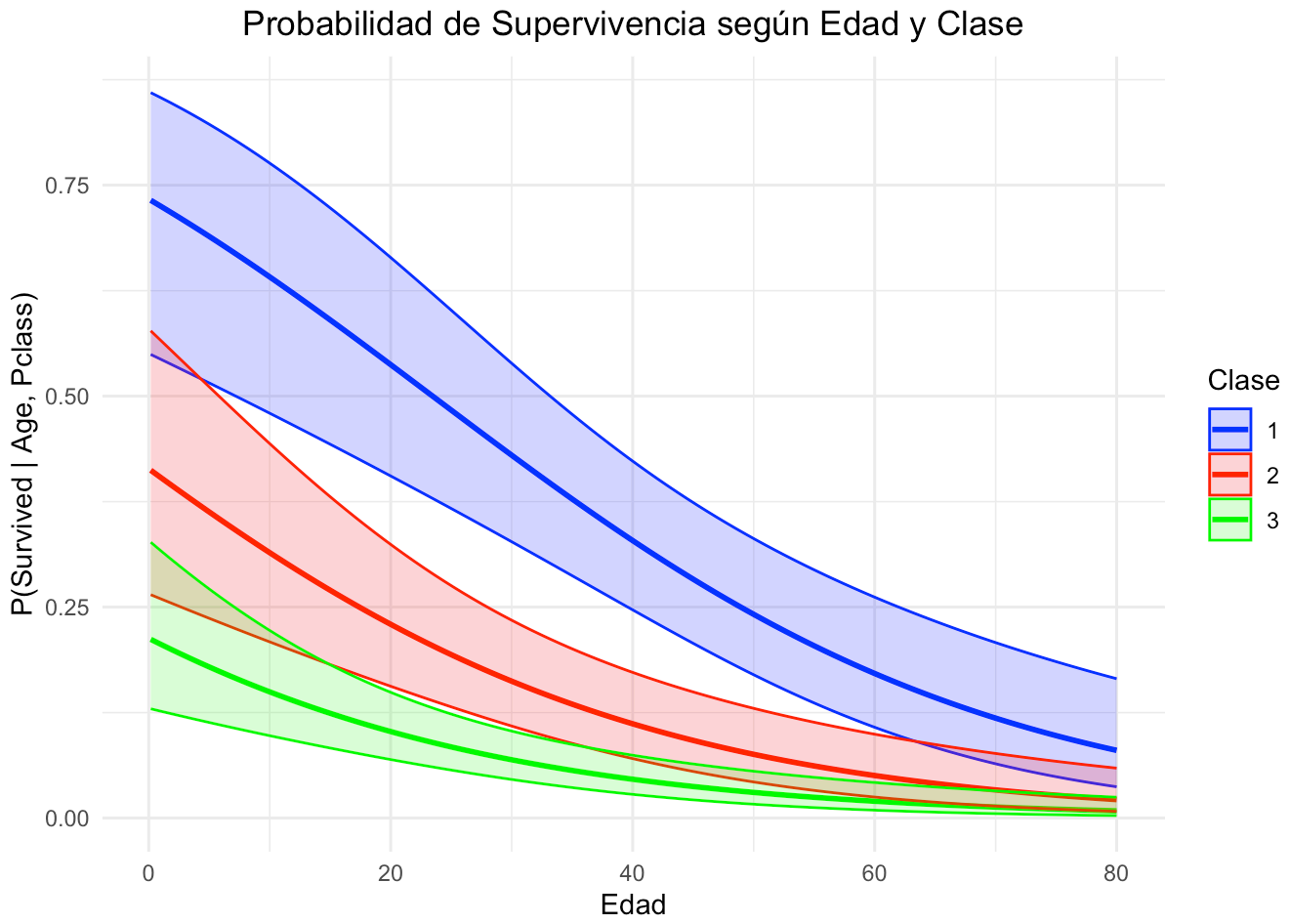
```
if(!require(ggplot2)) install.packages("ggplot2")
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
library(ggplot2)

valores_age = data.frame(
  Pclass = factor(rep(1:3, each = 100)),
  Sex = factor(rep("male", 300)),
  Age = seq(min(train_data$Age, na.rm = TRUE), max(train_data$Age, na.rm = TRUE), lengt
h.out = 100),
  SibSp = 0
)

predicciones = predict(best_model, newdata = valores_age, type = "link", se.fit = TRUE)

valores_age$probabilidad = plogis(predicciones$fit)
valores_age$CI.superior = plogis(predicciones$fit + 1.96 * predicciones$se.fit)
valores_age$CI.inferior = plogis(predicciones$fit - 1.96 * predicciones$se.fit)

ggplot(valores_age, aes(x = Age, y = probabilidad, color = Pclass)) +
  geom_line(size = 1) +
  geom_ribbon(aes(ymin = CI.inferior, ymax = CI.superior, fill = Pclass), alpha = 0.2) +
  labs(title = "Probabilidad de Supervivencia según Edad y Clase",
       x = "Edad",
       y = "P(Survived | Age, Pclass)") +
  scale_color_manual(values = c("blue", "red", "green"), name = "Clase") +
  scale_fill_manual(values = c("blue", "red", "green"), name = "Clase") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Escribe su ecuación, analiza sus coeficientes y detecta el efecto de cada predictor en la clasificación.

```
cat("\nEcuación del modelo:\n")
```

```
##
## Ecuación del modelo:
```

```
cat("logit(Survived) = 4.86 - 1.72*Pclass2 - 2.72*Pclass3 - 3.59*Sexmale - 0.04*Age - 0.30*SibSp\n")
```

```
## logit(Survived) = 4.86 - 1.72*Pclass2 - 2.72*Pclass3 - 3.59*Sexmale - 0.04*Age - 0.30*SibSp
```

4. Analiza las predicciones para los datos de entrenamiento

Elabora la matriz de confusión

```
train_predictions = predict(best_model, newdata = train_data, type = "response")
train_data$Predicted = ifelse(train_predictions > 0.5, 1, 0) # Clasificamos como '1' (so brevive) o '0' (no sobrevive) usando un umbral de 0.5

confusion_matrix = table(Predicted = train_data$Predicted, Actual = train_data$Survived)
accuracy = sum(diag(confusion_matrix)) / sum(confusion_matrix)

cat("Matriz de Confusión:\n")
```

```
## Matriz de Confusión:
```

```
print(confusion_matrix)
```

```
##          Actual
## Predicted   0   1
##         0 409  69
##         1  42 209
```

```
cat("\nPrecisión:", round(accuracy, 2))
```

```
##
## Precisión: 0.85
```

Elabora la Curva ROC

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

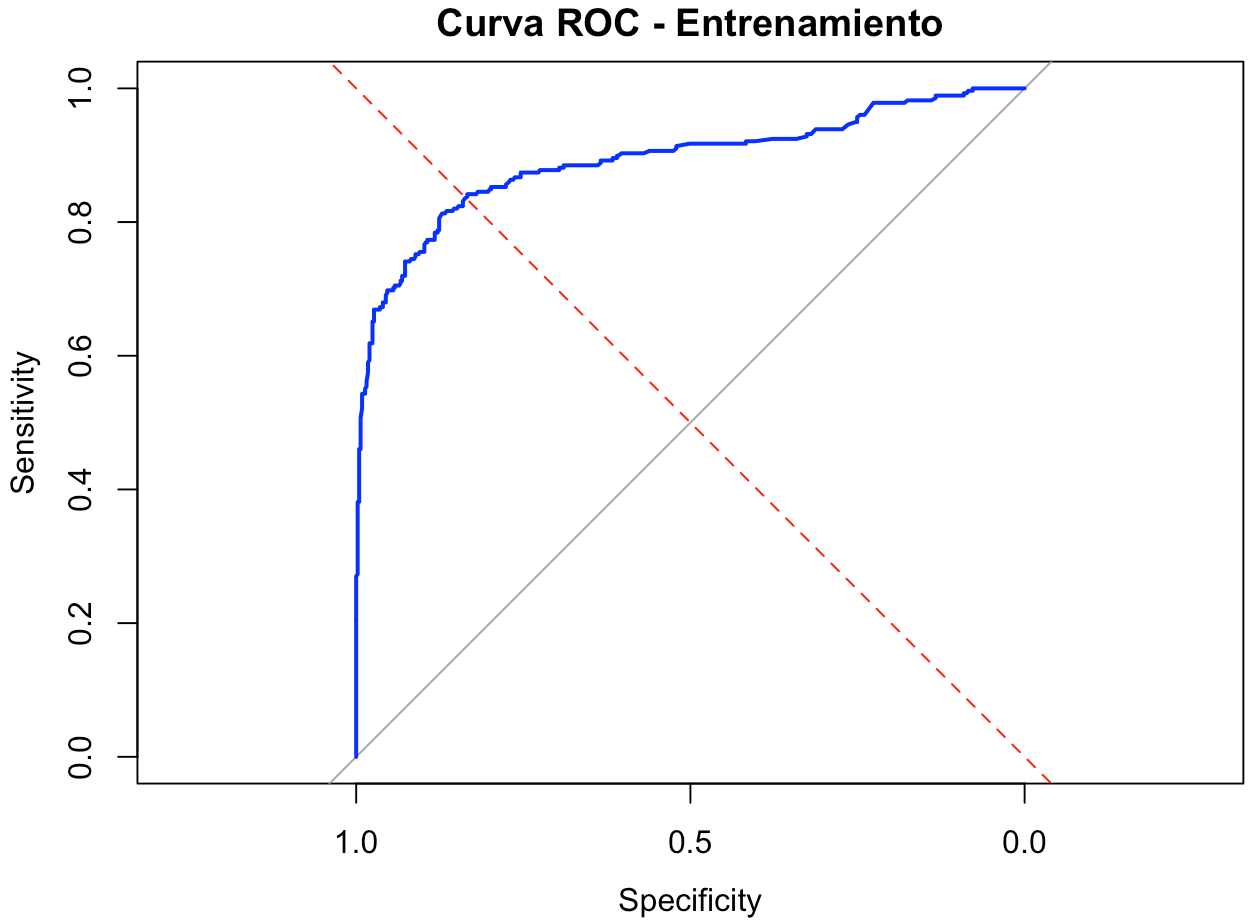
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
roc_curve = roc(train_data$Survived, train_predictions)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_curve, col = "blue", main = "Curva ROC - Entrenamiento")
abline(a = 0, b = 1, lty = 2, col = "red")
```

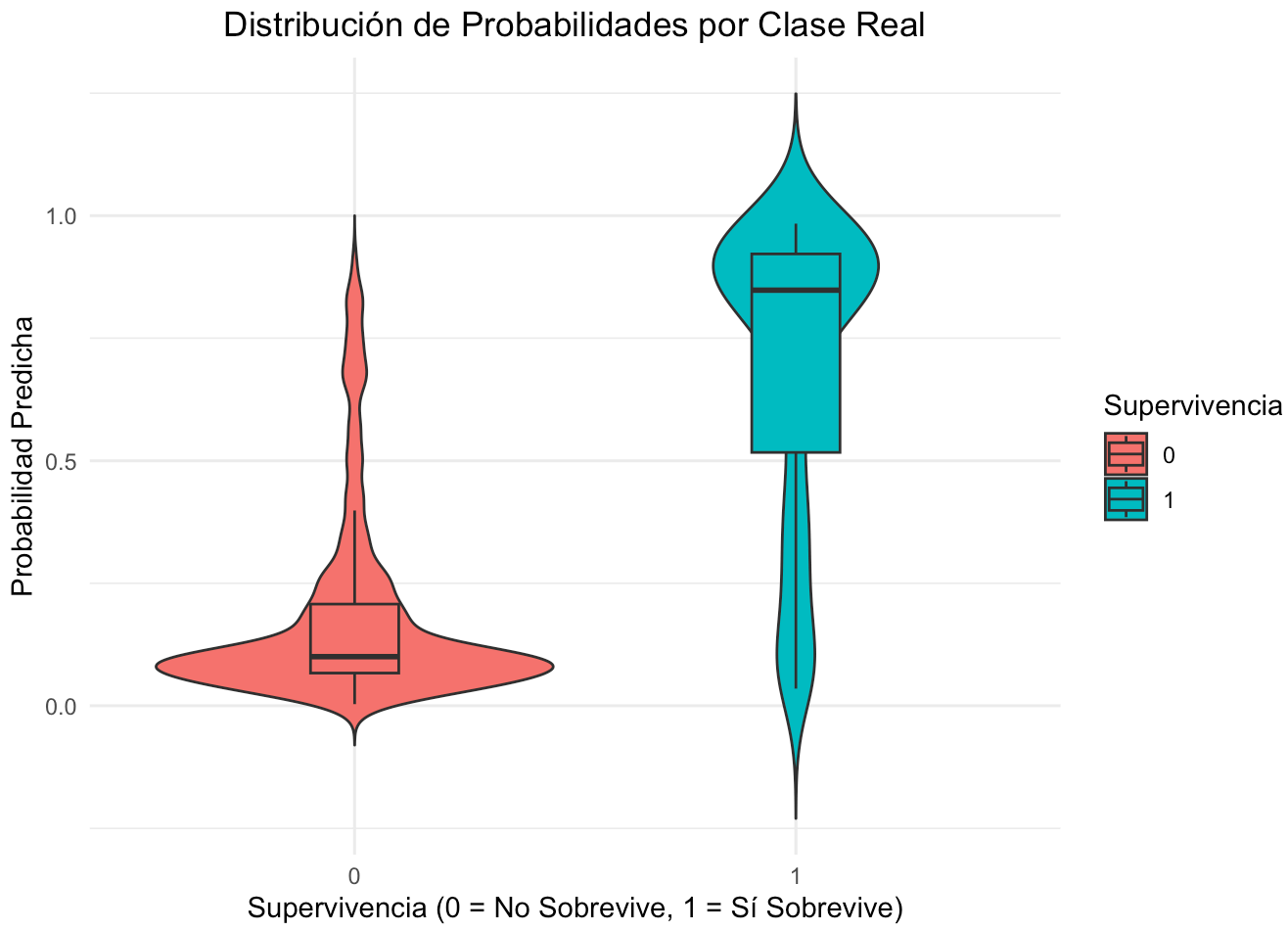



```
cat("\nÁrea bajo la curva (AUC):", round(auc(roc_curve), 3), "\n")
```

```
##
## Área bajo la curva (AUC): 0.893
```

Elabora el gráfico de violín

```
library(ggplot2)

ggplot(train_data, aes(x = as.factor(Survived), y = train_predictions, fill = as.factor
(Survived))) +
  geom_violin(trim = FALSE) +
  geom_boxplot(width = 0.2, position = position_dodge(0.9), outlier.shape = NA) +
  labs(title = "Distribución de Probabilidades por Clase Real",
       x = "Supervivencia (0 = No Sobrevive, 1 = Sí Sobrevive)",
       y = "Probabilidad Predicha",
       fill = "Supervivencia") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



5. Validación del modelo con la base de datos de validación

Elije un umbral de clasificación óptimo

```
test_data_clean = na.omit(test_data)
validation_predictions_clean = predict(best_model, newdata = test_data_clean, type = "re
sponse")
validation_roc = roc(test_data_clean$Survived, validation_predictions_clean)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
optimal_coords = coords(validation_roc, "all", ret = c("threshold", "sensitivity", "spec
ificity"), transpose = FALSE)
```

```
optimal_threshold = coords(validation_roc, "best", ret = "threshold", transpose = FALSE)
# cat("Umbral ajustado:", adjusted_threshold, "\n")
adjusted_threshold = 0.33

test_data_clean$Predicted = ifelse(validation_predictions_clean > adjusted_threshold, 1,
0)
confusion_matrix_validation = table(Predicted = test_data_clean$Predicted, Actual = test
_data_clean$Survived)
accuracy_validation = sum(diag(confusion_matrix_validation)) / sum(confusion_matrix_vali
dation)
```

Elabora la matriz de confusión con el umbral de clasificación óptimo

```
cat("\nMatriz de Confusión (Validación):\n")
```

```
##
## Matriz de Confusión (Validación):
```

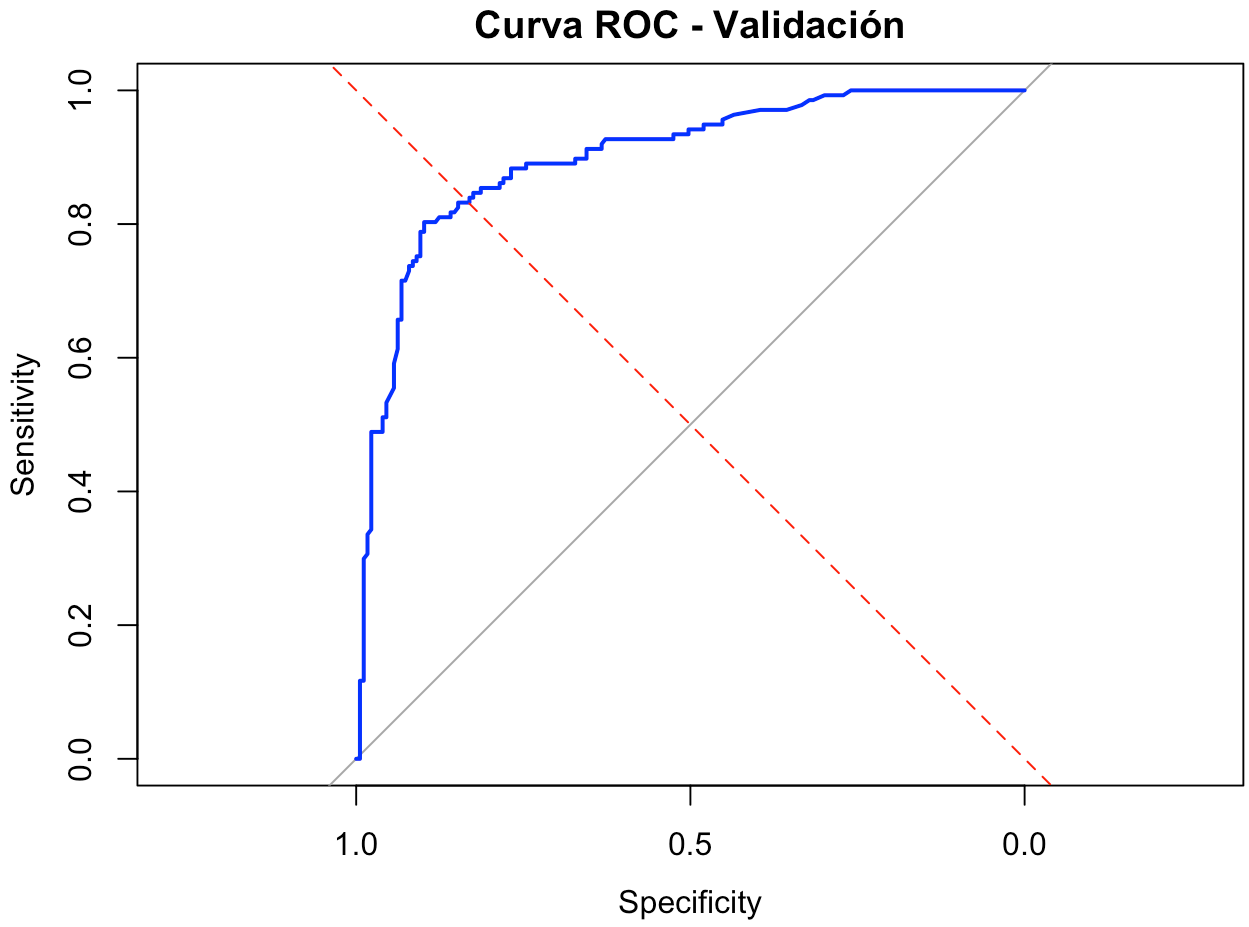
```
print(confusion_matrix_validation)
```

```
##          Actual
## Predicted   0   1
##         0 141  20
##         1  36 117
```

```
cat("\nPrecisión (Validación):", round(accuracy_validation, 2))
```

```
##
## Precisión (Validación): 0.82
```

```
plot(validation_roc, col = "blue", main = "Curva ROC - Validación")
abline(a = 0, b = 1, lty = 2, col = "red")
```




```
cat("\nÁrea bajo la curva (AUC):", round(auc(validation_roc), 3), "\n")
```

```
##
## Área bajo la curva (AUC): 0.9
```

6. Elabora el testeo con la base de datos de prueba.

```
test_data_final = read.csv("Titanic_test.csv")

cat("Estructura inicial de los datos de prueba:\n")
```

```
## Estructura inicial de los datos de prueba:
```

```
str(test_data_final)
```

```
## 'data.frame':    418 obs. of  11 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
##  $ Name       : chr  "Kelly, Mr. James" "Wilkes, Mrs. James (Ellen Needs)" "Myles, M
r. Thomas Francis" "Wirz, Mr. Albert" ...
##  $ Sex        : chr  "male" "female" "male" "male" ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Ticket     : chr  "330911" "363272" "240276" "315154" ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Cabin      : chr  "" "" "" "" ...
##  $ Embarked   : chr  "Q" "S" "Q" "S" ...
```

```
test_data_final$Pclass = as.factor(test_data_final$Pclass)
test_data_final$Sex = as.factor(test_data_final$Sex)
test_data_final$Embarked = as.factor(test_data_final$Embarked)
test_data_final_clean = na.omit(test_data_final)
```

```
final_predictions = predict(best_model, newdata = test_data_final_clean, type = "respons
e")
cat("Número de predicciones generadas:", length(final_predictions), "\n")
```

```
## Número de predicciones generadas: 331
```

```
adjusted_threshold = 0.3
test_data_final_clean$Predicted = ifelse(final_predictions > adjusted_threshold, 1, 0)

#output = data.frame(PassengerId = test_data_final_clean$PassengerId, Predicted = test_d
ata_final_clean$Predicted)
#write.csv(output, "Titanic_test_predictions.csv", row.names = FALSE)
```

7. Concluye en el contexto del problema:

Define las principales características que influyen en el modelo seleccionado e interpretalas: ¿qué características tuvieron las personas que sobrevivieron?

Después de todo este show, lo que hace el análisis de nuestro modelo logístico nos ayuda a identificar las principales características que influyen en la probabilidad de supervivencia de una persona en el Titanic.

Variables:

(Pclass): La clase socioeconómica tuvo un impacto significativo en la supervivencia, Los pasajeros de primera clase tuvieron una mayor probabilidad de sobrevivir vs los de tercera clase que tuvieron la probabilidad más baja.

(Sex): Ser mujer aumenta significativamente la probabilidad de supervivencia ya que los hombres tienen una probabilidad mucho más baja de sobrevivir.

Edad (Age): La edad también tiene efecto en la probabilidad de supervivencia.

Entonces podemos describir a las personas que sobrevivieron, de acuerdo con el modelo:

Clase Alta, Mujeres y Niños.

Interpreta los coeficientes del modelo

```
summary(best_model)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp, family = binomial,
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2283  -0.4944  -0.3428   0.4607   2.5900
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.801075   0.513306   9.353  < 2e-16 ***
## Pclass2     -1.360143   0.313370  -4.340 1.42e-05 ***
## Pclass3     -2.319298   0.308325  -7.522 5.38e-14 ***
## Sexmale     -3.788652   0.256399 -14.776  < 2e-16 ***
## Age         -0.043149   0.009025  -4.781 1.74e-06 ***
## SibSp       -0.470568   0.141493  -3.326 0.000882 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 969.16  on 728  degrees of freedom
## Residual deviance: 540.30  on 723  degrees of freedom
## AIC: 552.3
##
## Number of Fisher Scoring iterations: 5
```

Intercepto: Su valor positivo indica que la probabilidad de supervivencia en esta categoría es alta. *Pclass2 y Pclass3:*: Coeficiente negativo significa que estar en segunda clase reduce la probabilidad de supervivencia en comparación con la primera clase. Lo mismo para la tercera clase. *Sexmale:* Coeficiente negativo indica que los

hombres tienen una probabilidad mucho más baja de sobrevivir en comparación con las mujeres. *Age:* Coeficiente negativo indica que, por cada aumento en un año de edad, la probabilidad de supervivencia disminuye ligeramente.

Se obtiene que: *Positivo o de Mayor probabilidad: Ser mujer, de primera clase y menor edad.*

Que concuerda con lo anterior.

Define cuál es el mejor umbral de clasificación y por qué

```
roc_curve = roc(train_data$Survived, predict(best_model, newdata = train_data, type = "r
esponse"))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
optimal_threshold = coords(roc_curve, "best", ret = "threshold", transpose = FALSE)
print(round(optimal_threshold, 3))
```

```
##   threshold
## 1     0.366
```

```
optimal_metrics = coords(roc_curve, "best", ret = c("sensitivity", "specificity"), trans
pose = FALSE)
```

El umbral óptimo identificado es 0.366, nos indica que la si probabilidad predicha por el modelo es mayor o igual a 0.366, clasificamos la observación como 1 (sobrevive), si es menor, se clasifica como 0 (no sobrevive). Lo que maximiza el balance entre sensibilidad (detecta correctamente a los sobrevivientes) y especificidad (detecta correctamente a los no sobrevivientes) en los datos de entrenamiento o validación.