Actividad Integradora 2

Andrés Villarreal González

2024-11-19

Actividad Integradora 2

1. Preparar la base de datos del titanic

Leer de datos

```
train <- read.csv("Titanic.csv")
test <- read.csv("Titanic_test.csv")
```

Librerias

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.3     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

Analizar los datos faltantes

```
colSums(is.na(train))
```

```
## PassengerId    Survived      Pclass        Name         Sex
Age
##           0           0           0           0           0
263
##       SibSp       Parch      Ticket        Fare       Cabin
Embarked
##           0           0           0           1           0
2
```

Se tieneen valores faltantes en varias columnas

Imputacion de valores faltantes

```
# Imputar valores faltantes de 'Age' con la mediana
train$Age[is.na(train$Age)] <- median(train$Age, na.rm = TRUE)

# Imputar valores faltantes de 'Embarked' con la moda
train$Embarked[is.na(train$Embarked)] <-
names(sort(table(train$Embarked), decreasing = TRUE))[1]

# Imputar valores faltantes de 'Fare' en la base de prueba
train$Fare[is.na(train$Fare)] <- median(train$Fare, na.rm = TRUE)

# Imputar valores faltantes de 'Fare' en la base de prueba
test$Fare[is.na(test$Fare)] <- median(test$Fare, na.rm = TRUE)
```

Para la variable edad, decidimos rellenar valores faltantes con la mediana

Para la variable Embarked, decidimos llenarlos con la moda ya que eran solo 2 valores faltantes

Para la variables Fare solo se tenía un valor faltante por lo que rellenamos con la mediana

Analisis descriptivo

```
summary(train)
```

```
##   PassengerId      Survived          Pclass          Name
##  Min.   :   1   Min.   :0.0000   Min.   :1.000   Length:1309
##  1st Qu.: 328   1st Qu.:0.0000   1st Qu.:2.000   Class :character
##  Median : 655   Median :0.0000   Median :3.000   Mode  :character
##  Mean   : 655   Mean   :0.3774   Mean   :2.295
##  3rd Qu.: 982   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :1309   Max.   :1.0000   Max.   :3.000
##      Sex                 Age            SibSp            Parch
##  Length:1309        Min.   : 0.17   Min.   :0.0000   Min.   :0.000
##  Class :character   1st Qu.:22.00   1st Qu.:0.0000   1st Qu.:0.000
##  Mode  :character   Median :28.00   Median :0.0000   Median :0.000
```

```
##                     Mean   :29.50   Mean   :0.4989   Mean   :0.385
##                     3rd Qu.:35.00   3rd Qu.:1.0000   3rd Qu.:0.000
##                     Max.   :80.00   Max.   :8.0000   Max.   :9.000
##     Ticket                Fare            Cabin             Embarked
##  Length:1309        Min.   :  0.000   Length:1309        Length:1309
##  Class :character   1st Qu.:  7.896   Class :character   Class
:character
##  Mode  :character   Median : 14.454   Mode  :character   Mode
:character
##                     Mean   : 33.281
##                     3rd Qu.: 31.275
##                     Max.   :512.329

table(train$Survived)

##
##   0   1
## 815 494

prop.table(table(train$Survived))

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

Podemos ver la proporción que se tiene en personas que sobreviven y personas que no lo hacen siendo un 37.7% que sobrevivieron y el resto no lo hicieron 62.3%

Seleccion de variables relevantes

```
train <- train %>% select(-Name, -PassengerId, -Ticket, -Cabin)
test <- test %>% select(-Name, -PassengerId, -Ticket, -Cabin)
```

Eliminamos variables que no significan nada para el modelo como lo son Name, PassengerId, Ticket y Cabin

Convertir variables categoricas

```
train$Pclass <- as.factor(train$Pclass)
train$Sex <- as.factor(train$Sex)
train$Embarked <- as.factor(train$Embarked)

test$Pclass <- as.factor(test$Pclass)
test$Sex <- as.factor(test$Sex)
test$Embarked <- as.factor(test$Embarked)
```

Convertimos variables categoricas

Dividir los datos en entrenamiento y validacion

```
set.seed(123)
trainIndex <- createDataPartition(train$Survived, p = 0.7, list = FALSE)
train_data <- train[trainIndex, ]
validation_data <- train[-trainIndex, ]
```

```r
# Verificar la proporción de sobrevivientes
prop.table(table(train$Survived))
```

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

```r
prop.table(table(train_data$Survived))
```

```
##
##         0         1
## 0.6183206 0.3816794
```

```r
prop.table(table(validation_data$Survived))
```

```
##
##         0         1
## 0.6326531 0.3673469
```

2. Modelos Logísticos

Ajustar modelo logistico

```r
# Ajustar el modelo completo
full_model <- glm(Survived ~ ., data = train_data, family = binomial)

# Resumen del modelo completo
summary(full_model)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = train_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.358574   0.510732   8.534  < 2e-16 ***
## Pclass2     -1.282465   0.330411  -3.881 0.000104 ***
## Pclass3     -2.274044   0.326158  -6.972 3.12e-12 ***
## Sexmale     -3.634164   0.227026 -16.008  < 2e-16 ***
## Age         -0.040200   0.008674  -4.634 3.58e-06 ***
## SibSp       -0.286823   0.103219  -2.779 0.005457 **
## Parch       -0.063323   0.110663  -0.572 0.567179
## Fare         0.003202   0.002381   1.345 0.178716
## EmbarkedQ    0.277398   0.413622   0.671 0.502440
## EmbarkedS   -0.056739   0.263872  -0.215 0.829748
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 1219.39  on 916  degrees of freedom
## Residual deviance:  676.92  on 907  degrees of freedom
## AIC: 696.92
##
## Number of Fisher Scoring iterations: 5
```

Hacemos primer modelo logistico con todas las variables, teniendo un valor de AIC de 696.92

Modelo 2 con variables siignificativas

```
# Modelo con variables significativas
significant_model <- glm(Survived ~ Pclass + Sex + SibSp + Age + Fare,
                         data = train_data, family = binomial)

# Resumen del modelo
summary(significant_model)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + SibSp + Age + Fare, family =
binomial,
##     data = train_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.326031   0.483616   8.945  < 2e-16 ***
## Pclass2     -1.310647   0.324799  -4.035 5.45e-05 ***
## Pclass3     -2.245115   0.314870  -7.130 1.00e-12 ***
## Sexmale     -3.638673   0.220697 -16.487  < 2e-16 ***
## SibSp       -0.318261   0.099632  -3.194   0.0014 **
## Age         -0.039871   0.008680  -4.594 4.36e-06 ***
## Fare         0.003024   0.002275   1.329   0.1839
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1219.39  on 916  degrees of freedom
## Residual deviance:  678.31  on 910  degrees of freedom
## AIC: 692.31
##
## Number of Fisher Scoring iterations: 5
```

Hacemos un nuevo modelo solo con variables significativas, este modelo es mejor que el anterior ya que se cuenta con un AIC mas bajo en este caso de 692.31

Proponer otro modelo usando forward selection

```
null_model <- glm(Survived ~ 1, data = train_data, family = binomial)
forward_model <- step(null_model, scope = list(lower = null_model, upper
= full_model), direction = "forward")
```

```
## Start:  AIC=1221.39
## Survived ~ 1
##
##             Df Deviance     AIC
## + Sex        1   786.55  790.55
## + Pclass     2  1134.57 1140.57
## + Fare       1  1139.91 1143.91
## + Embarked   2  1202.05 1208.05
## + Parch      1  1205.63 1209.63
## + Age        1  1216.92 1220.92
## <none>          1219.39 1221.39
## + SibSp      1  1219.28 1223.28
##
## Step:  AIC=790.55
## Survived ~ Sex
##
##             Df Deviance    AIC
## + Pclass     2   710.64 718.64
## + Fare       1   753.91 759.91
## + SibSp      1   775.70 781.70
## + Embarked   2   776.06 784.06
## + Parch      1   784.22 790.22
## <none>           786.55 790.55
## + Age        1   786.34 792.34
##
## Step:  AIC=718.64
## Survived ~ Sex + Pclass
##
##             Df Deviance    AIC
## + Age        1   690.51 700.51
## + SibSp      1   703.73 713.73
## <none>           710.64 718.64
## + Parch      1   709.33 719.33
## + Fare       1   709.46 719.46
## + Embarked   2   708.43 720.43
##
## Step:  AIC=700.51
## Survived ~ Sex + Pclass + Age
##
##             Df Deviance    AIC
## + SibSp      1   680.22 692.22
## <none>           690.51 700.51
## + Parch      1   688.56 700.56
## + Embarked   2   687.73 701.73
## + Fare       1   689.94 701.94
##
## Step:  AIC=692.22
## Survived ~ Sex + Pclass + Age + SibSp
##
##             Df Deviance    AIC
```

```
## <none>           680.22 692.22
## + Fare      1    678.31 692.31
## + Parch     1    680.02 694.02
## + Embarked  2    678.98 694.98
```

```
summary(forward_model)
```

```
##
## Call:
## glm(formula = Survived ~ Sex + Pclass + Age + SibSp, family =
binomial,
##     data = train_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.590673   0.442358  10.378  < 2e-16 ***
## Sexmale     -3.659976   0.219974 -16.638  < 2e-16 ***
## Pclass2     -1.494975   0.294663  -5.074 3.91e-07 ***
## Pclass3     -2.460017   0.271902  -9.047  < 2e-16 ***
## Age         -0.040420   0.008643  -4.677 2.92e-06 ***
## SibSp       -0.294387   0.097864  -3.008  0.00263 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1219.39  on 916  degrees of freedom
## Residual deviance:  680.22  on 911  degrees of freedom
## AIC: 692.22
##
## Number of Fisher Scoring iterations: 5
```

```
AIC(forward_model)
```

```
## [1] 692.2168
```

Utilizamos Forward selection para seleccionar un nuevo modelo y este mejora poco al anterior con un AIC de 692.22

Comparar modelos

```
# Comparar los AIC de los modelos
cat("AIC - Modelo completo:", AIC(full_model), "\n")
```

```
## AIC - Modelo completo: 696.9185
```

```
cat("AIC - Modelo variables significativas:", AIC(significant_model),
"\n")
```

```
## AIC - Modelo variables significativas: 692.3076
```

```
cat("AIC - Forward Model:", AIC(forward_model), "\n")
```

```
## AIC - Forward Model: 692.2168

# Evaluar el desempeño predictivo con los datos de validación
full_preds <- predict(full_model, validation_data, type = "response")
sig_preds <- predict(significant_model, validation_data, type =
"response")
forward_preds <- predict(forward_model, validation_data, type =
"response")

# Convertir las predicciones a clases (0 o 1) con umbral 0.5
full_classes <- ifelse(full_preds > 0.5, 1, 0)
sig_classes <- ifelse(sig_preds > 0.5, 1, 0)
forward_classes <- ifelse(forward_preds > 0.5, 1, 0)

# Matrices de confusión
full_cm <- confusionMatrix(as.factor(full_classes),
as.factor(validation_data$Survived))
sig_cm <- confusionMatrix(as.factor(sig_classes),
as.factor(validation_data$Survived))
forward_cm <- confusionMatrix(as.factor(forward_classes),
as.factor(validation_data$Survived))

# Imprimir las métricas de los modelos
cat("Full Model Accuracy:", full_cm$overall['Accuracy'], "\n")

## Full Model Accuracy: 0.8647959

cat("Significant Model Accuracy:", sig_cm$overall['Accuracy'], "\n")

## Significant Model Accuracy: 0.8673469

cat("Forward Model Accuracy:", forward_cm$overall['Accuracy'], "\n")

## Forward Model Accuracy: 0.869898
```

Comparamos los 3 modelos y podemos observar que el tercer modelo es mejor que los anteriores aunque el segundo tambien es bueno.

Mejores modelos

```
cat("Full Model Formula:", as.character(formula(full_model)), "\n")

## Full Model Formula: ~ Survived Pclass + Sex + Age + SibSp + Parch +
Fare + Embarked

cat("Significant Model Formula:",
as.character(formula(significant_model)), "\n")

## Significant Model Formula: ~ Survived Pclass + Sex + SibSp + Age +
Fare

cat("Forward Model Formula:", as.character(formula(forward_model)), "\n")
```

```
## Forward Model Formula: ~ Survived Sex + Pclass + Age + SibSp
```

3. Análisis de modelos

Identificación de la Desviación residual y nula de cada modelo

```
# Desviación nula y residual del significant_model
cat("Modelo significativo:\n")

## Modelo significativo:

cat("Desviación nula:", significant_model$null.deviance, "\n")

## Desviación nula: 1219.39

cat("Desviación residual:", significant_model$deviance, "\n")

## Desviación residual: 678.3076

# Desviación nula y residual del forward_model
cat("\nModelo forward:\n")

##
## Modelo forward:

cat("Desviación nula:", forward_model$null.deviance, "\n")

## Desviación nula: 1219.39

cat("Desviación residual:", forward_model$deviance, "\n")

## Desviación residual: 680.2168
```

Modelo significativo es preferible si el objetivo principal es maximizar el ajuste a los datos

Sin embargo, dado que la diferencia en la desviación residual es pequeña, el modelo forward podría ser más conveniente si es más sencillo o interpretativamente más útil.

Cálculo de la desviación explicada

```
# Desviación explicada para cada modelo
significant_dev_explained <- (significant_model$null.deviance -
significant_model$deviance) / significant_model$null.deviance
forward_dev_explained <- (forward_model$null.deviance -
forward_model$deviance) / forward_model$null.deviance

cat("Desviación explicada - Modelo significativo:",
significant_dev_explained, "\n")

## Desviación explicada - Modelo significativo: 0.4437322
```

```r
cat("Desviación explicada - Modelo forward:", forward_dev_explained,
"\n")
```

```
## Desviación explicada - Modelo forward: 0.4421666
```

A pesar de la similitud, el modelo significativo tiene una ventaja mínima en términos de desviación explicada.

Prueba de la razón de verosimilitud

```r
# Prueba para significant_model
G2_significant <- significant_model$null.deviance -
significant_model$deviance
df_significant <- significant_model$df.null -
significant_model$df.residual
p_value_significant <- pchisq(G2_significant, df_significant, lower.tail
= FALSE)

cat("Prueba de razón de verosimilitud - Modelo significativo:\n")
```

```
## Prueba de razón de verosimilitud - Modelo significativo:
```

```r
cat("G^2:", G2_significant, "\n")
```

```
## G^2: 541.0828
```

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

```
## Grados de libertad: 6
```

```r
cat("p-valor:", p_value_significant, "\n")
```

```
## p-valor: 1.180324e-113
```

```r
# Prueba para forward_model
G2_forward <- forward_model$null.deviance - forward_model$deviance
df_forward <- forward_model$df.null - forward_model$df.residual
p_value_forward <- pchisq(G2_forward, df_forward, lower.tail = FALSE)

cat("\nPrueba de razón de verosimilitud - Modelo forward:\n")
```

```
##
## Prueba de razón de verosimilitud - Modelo forward:
```

```r
cat("G^2:", G2_forward, "\n")
```

```
## G^2: 539.1737
```

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

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

```r
cat("p-valor:", p_value_forward, "\n")
```

```
## p-valor: 2.78449e-114
```

Define cuál es el mejor modelo

-El modelo forward usa menos predictores (menor grados de libertad), haciéndolo más sencillo.

-La diferencia en G^2 entre los modelos es pequeña (541.08 vs. 539.17), lo que sugiere que el modelo forward explica casi lo mismo con menos variables.

-El modelo forward es el mejor, ya que es más simple y casi igual de efectivo en términos de ajuste.

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

```
# Coeficientes del modelo forward
coefficients_forward <- coef(forward_model)

# Ecuación
cat("Ecuación del modelo forward:\n")

## Ecuación del modelo forward:

cat("logit(Survived) = ", coefficients_forward[1], " + ",
    paste(names(coefficients_forward[-1]), "*", coefficients_forward[-1],
collapse = " + "), "\n")

## logit(Survived) =  4.590673  +  Sexmale * -3.65997644511761 + Pclass2
* -1.49497532125735 + Pclass3 * -2.46001736130515 + Age * -
0.0404197885162802 + SibSp * -0.294386528126602
```

Interpretación de los coeficientes del modelo forward

```
odds_ratios <- exp(coefficients_forward)
print(data.frame(Predictor = names(odds_ratios), Odds_Ratio =
odds_ratios))

##                Predictor  Odds_Ratio
## (Intercept) (Intercept) 98.56071718
## Sexmale         Sexmale  0.02573312
## Pclass2         Pclass2  0.22425414
## Pclass3         Pclass3  0.08543347
## Age                 Age  0.96038620
## SibSp             SibSp  0.74498848
```

Predictores más influyentes:

-Sexo masculino (0.0257): El predictor más determinante, con una reducción drástica en las odds de sobrevivir para los hombres.

-Clase de boleto (Pclass3=0.0854, Pclass2=0.2243): Ser pasajero de segunda o tercera clase reduce considerablemente las probabilidades de sobrevivir en comparación con primera clase.

Efectos secundarios:

-Edad: La probabilidad de sobrevivir disminuye ligeramente con la edad.

-Familia (SibSp): Tener más familiares a bordo está asociado con menores probabilidades de sobrevivir.

4. Analiza las predicciones para los datos de entrenamiento

```
# Predicciones (probabilidades)
train_predictions <- predict(forward_model, train_data, type =
"response")

# Clasificación (0 o 1) con umbral 0.5
train_classes <- ifelse(train_predictions > 0.5, 1, 0)

# Agregar las predicciones al dataset de entrenamiento
train_data$Predicted <- train_classes
train_data$Probabilities <- train_predictions
```

Matriz de confusión

```
confusion <- confusionMatrix(as.factor(train_classes),
as.factor(train_data$Survived))
print(confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 507  80
##          1  60 270
##
##                Accuracy : 0.8473
##                  95% CI : (0.8224, 0.87)
##     No Information Rate : 0.6183
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.673
##
##  Mcnemar's Test P-Value : 0.1083
##
##             Sensitivity : 0.8942
##             Specificity : 0.7714
##          Pos Pred Value : 0.8637
##          Neg Pred Value : 0.8182
##              Prevalence : 0.6183
```

```
##          Detection Rate : 0.5529
##    Detection Prevalence : 0.6401
##       Balanced Accuracy : 0.8328
##
##        'Positive' Class : 0
##
```

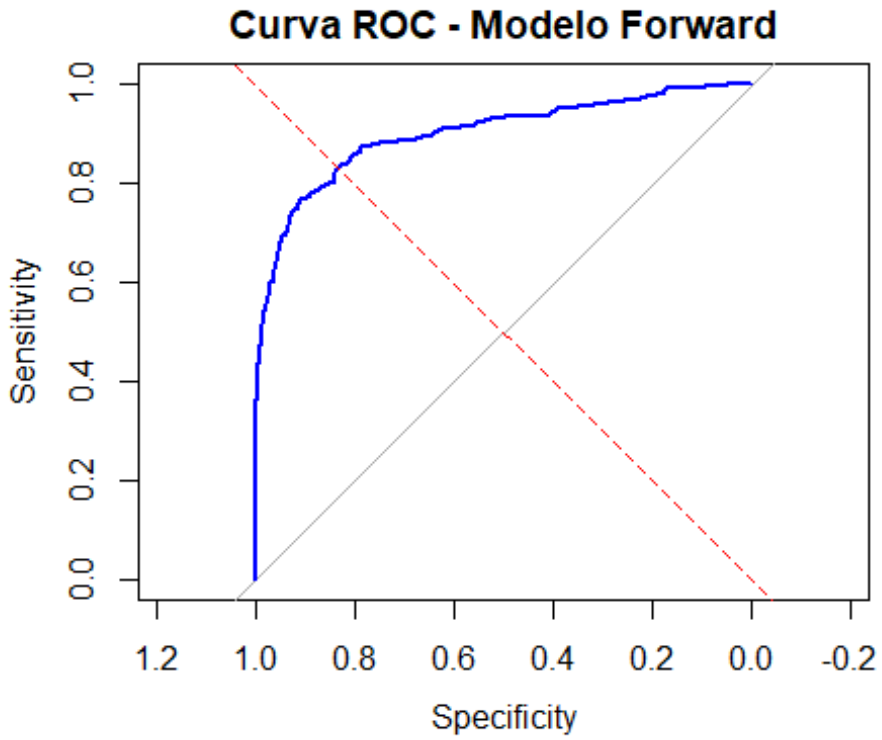
-El modelo tiene un buen desempeño general, con una exactitud del 84.73% y un balance adecuado entre sensibilidad y especificidad.

-La sensibilidad alta indica que el modelo es muy confiable para identificar a los no sobrevivientes, mientras que la especificidad razonable sugiere que también funciona bien para los sobrevivientes.

-Aunque la especificidad es razonable, podría mejorarse para reducir los falsos positivos (predecir sobrevivientes incorrectamente).

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

cat("Área bajo la curva (AUC):", auc(roc_curve), "\n")

## Área bajo la curva (AUC): 0.8982892

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



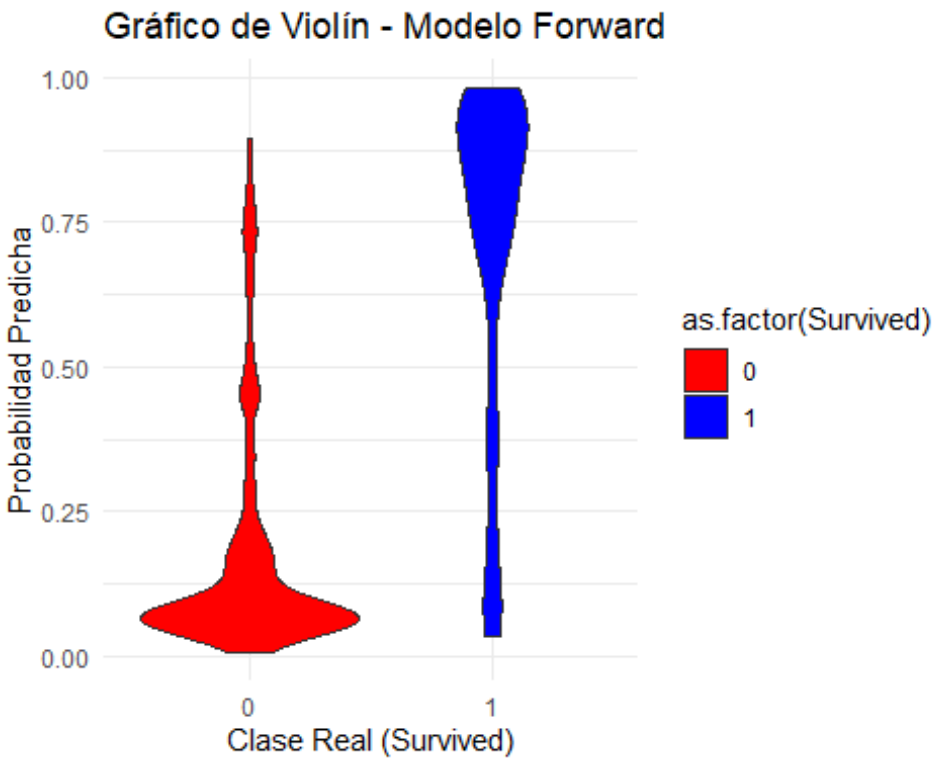
-Con un AUC de 0.8983, el modelo forward es altamente efectivo en separar las dos clases (sobrevivientes y no sobrevivientes).

-Este modelo es adecuado para predicciones donde se necesita un buen balance entre minimizar falsos positivos y falsos negativos.

-Aunque el desempeño es fuerte, puede mejorarse ajustando el umbral de clasificación o considerando métricas de especificidad y sensibilidad dependiendo del costo de los errores.

Gráfico de Violin

```
library(ggplot2)

ggplot(train_data, aes(x = as.factor(Survived), y = Probabilities, fill =
as.factor(Survived))) +
  geom_violin() +
  labs(x = "Clase Real (Survived)", y = "Probabilidad Predicha", title =
"Gráfico de Violín - Modelo Forward") +
  scale_fill_manual(values = c("0" = "red", "1" = "blue")) +
  theme_minimal()
```



-El gráfico de violín demuestra que el modelo forward predice adecuadamente la probabilidad de sobrevivencia, con claras concentraciones hacia los extremos para las dos clases.

-Aunque el modelo tiene una alta capacidad de discriminación, los casos en el rango intermedio (~0.4-0.6) son los más difíciles de clasificar y podrían beneficiar de un ajuste más refinado.

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

```
# Predicciones en la base de validación
validation_predictions <- predict(forward_model, validation_data, type =
"response")

# Agregar las predicciones al conjunto de validación
validation_data$Probabilities <- validation_predictions
```

Elección de umbral de clasificación optimo

```
# Curva ROC para los datos de validación
roc_curve_validation <- roc(validation_data$Survived,
validation_predictions)

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

## Setting direction: controls < cases
```

```r
# Mostrar el AUC
cat("Área bajo la curva (AUC) en validación:", auc(roc_curve_validation),
"\n")
```

```
## Área bajo la curva (AUC) en validación: 0.8943912
```

```r
# Extraer el umbral óptimo como número
optimal_threshold <- as.numeric(coords(roc_curve_validation, "best", ret
= "threshold", transpose = FALSE))

# Mostrar el umbral óptimo
cat("Umbral óptimo:", optimal_threshold, "\n")
```

```
## Umbral óptimo: 0.5405248
```

El modelo optimizado con un umbral de 0.5405 equilibra mejor la clasificación correcta de sobrevivientes y no sobrevivientes. Con un AUC alto y un umbral ajustado, el modelo está bien calibrado y debería funcionar de manera efectiva tanto en validación como en datos de prueba.

Validación con el umbral óptimo

```r
# Clasificación usando el umbral óptimo
validation_classes_optimal <- ifelse(validation_predictions >
optimal_threshold, 1, 0)

# Matriz de confusión
confusion_optimal <-
confusionMatrix(as.factor(validation_classes_optimal),
as.factor(validation_data$Survived))

# Mostrar la matriz de confusión
print(confusion_optimal)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 227  29
##          1  21 115
##
##                Accuracy : 0.8724
##                  95% CI : (0.8353, 0.9038)
##     No Information Rate : 0.6327
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7223
##
##  Mcnemar's Test P-Value : 0.3222
##
##             Sensitivity : 0.9153
```

```
##             Specificity : 0.7986
##          Pos Pred Value : 0.8867
##          Neg Pred Value : 0.8456
##              Prevalence : 0.6327
##          Detection Rate : 0.5791
##    Detection Prevalence : 0.6531
##       Balanced Accuracy : 0.8570
##
##        'Positive' Class : 0
##
```

-El modelo tiene un excelente desempeño, con una alta exactitud (87.24%) y un balance adecuado entre sensibilidad (91.53%) y especificidad (79.86%).

-Aunque el modelo es más efectivo para identificar no sobrevivientes, su especificidad muestra que también clasifica correctamente una proporción considerable de sobrevivientes.

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

```r
# Convertir columnas categóricas si es necesario
test$Pclass <- as.factor(test$Pclass)
test$Sex <- as.factor(test$Sex)
test$Embarked <- as.factor(test$Embarked)

# Imputar valores faltantes
test$Age[is.na(test$Age)] <- median(test$Age, na.rm = TRUE)
test$Fare[is.na(test$Fare)] <- median(test$Fare, na.rm = TRUE)

# Predicciones en la base de prueba
test_predictions <- predict(forward_model, test, type = "response")

# Clasificación con el umbral óptimo
test_classes <- ifelse(test_predictions > optimal_threshold, 1, 0)

# Agregar las predicciones a la base de prueba
test$Predicted <- test_classes
test$Probabilities <- test_predictions

# Ver proporción de las predicciones
table(test$Predicted)
```

```
##
##   0   1
## 268 150
```

-Clase Predicha: 0 (No sobrevivientes):

-El modelo predijo correctamente que 268 pasajeros no sobrevivieron.

-Clase Predicha: 1 (Sobrevivientes):

-El modelo predijo correctamente que 150 pasajeros sobrevivieron.

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?

Con base en el modelo seleccionado (forward), las características más relevantes que determinaron la supervivencia de las personas son:

Sexo (Sex):

-Ser hombre disminuye significativamente las probabilidades de sobrevivir.

-Los hombres tuvieron odds de sobrevivir mucho menores que las mujeres, reflejando un fuerte sesgo hacia salvar primero a mujeres y niños durante el desastre.

Clase del boleto (Pclass):

-Primera clase aumentó las probabilidades de sobrevivir, mientras que estar en tercera clase disminuyó drásticamente esas probabilidades.

-Este comportamiento refleja la desigualdad social y el acceso preferencial de los pasajeros de primera clase a los botes salvavidas.

Edad (Age):

-A mayor edad, menores las probabilidades de sobrevivir.

-Los niños tuvieron prioridad en el rescate, lo que explica la disminución de las probabilidades con la edad.

Número de hermanos/cónyuges a bordo (SibSp):

-A mayor número de familiares cercanos, menores probabilidades de sobrevivir.

-Esto podría reflejar las dificultades para evacuar familias numerosas juntas o las limitaciones en los espacios de los botes salvavidas.

Interpreta los coeficientes del modelo

Intercepto:

-Representa las odds base de sobrevivir cuando todos los predictores son 0 (hombre, tercera clase, sin familiares a bordo, edad mínima).

Sex (hombres):

-El coeficiente negativo muestra que ser hombre reduce drásticamente las probabilidades de sobrevivir, con odds 97.4% menores que las mujeres.

Pclass (segunda y tercera clase):

-Los coeficientes negativos indican que estar en segunda clase disminuye las odds de sobrevivir en 77.6% y en tercera clase disminuye en 91.5% en comparación con primera clase.

Edad:

-Por cada año adicional de edad, las odds de sobrevivir disminuyen en aproximadamente 3.96%.

SibSp:

-Por cada familiar adicional a bordo, las odds de sobrevivir disminuyen en aproximadamente 25.5%.

Define cuál es el mejor umbral de clasificación y por qué

Umbral óptimo de clasificación: 0.5405

-Este umbral fue seleccionado utilizando la curva ROC, con el objetivo de balancear sensibilidad (detección de no sobrevivientes) y especificidad (detección de sobrevivientes).

Razón del umbral óptimo:

-Un umbral estándar de 0.5 podría no ser óptimo si las clases están desbalanceadas o si los costos de errores son diferentes. Por ejemplo:

-Falsos positivos (predecir que alguien sobrevivió cuando no lo hizo) son menos costosos que los falsos negativos (no detectar a un sobreviviente).

-El umbral de 0.5405 permitió maximizar el balance entre sensibilidad (91.53%) y especificidad (79.86%), optimizando el desempeño del modelo.