Modelado estadístico de datos: Práctica 2

Sergio Soler Rocha
Universidad Nacional de Educación a Distancia

1. ¿Qué es la regresión logística multinomial? Se pide además ilustrar este concepto con algún ejemplo numérico. En el documento que se entregue habrá que incluir el código utilizado

La regresión multinomial es un tipo de modelo de regresión utilizado para predecir la probabilidad de ocurrencia de múltiples resultados categorizados. A diferencia de la regresión logística binaria, que se usa para predecir entre dos resultados posibles, la regresión multinomial maneja más de dos categorías o clases.

En la regresión logística multinomial, buscamos etiquetar cada observación con una clase k de un conjunto de K clases, con la condición de que solo una de estas clases sea la correcta (a veces denominada clasificación exclusiva; una observación no puede pertenecer a varias clases simultáneamente). Utilizaremos la siguiente representación: la salida $\mathbf{y}$ para cada entrada $\mathbf{x}$ será un vector de tamaño K. Supongamos que la clase c es la correcta, entonces el vector $\mathbf{y}$ será $y_c = 1$ y el resto de elementos $y_j = 0\ \forall j \neq c$. Un vector $\mathbf{y}$ con un solo valor igual a 1 y el resto 0 es conocido como vector one-hot. El trabajo del clasificador es producir un vector estimado $\hat{\mathbf{y}}$. Para cada clase k, el valor $\hat{\mathbf{y}}_k$ será la estimación de probabilidad del clasificador $p(y_k = 1|\mathbf{x})$.

El clasificador logístico multinomial utiliza una generalización del sigmoid, llamada función softmax, para calcular $p(y_k = 1|\mathbf{x})$. La función softmax toma un vector $\mathbf{z} = [z_1, z_2, ..., z_k]$ de K valores arbitrarios y los asigna a una distribución de probabilidad, con cada valor en el intervalo $(0, 1)$, sumando todos los valores 1. Al igual que la función sigmmoid, es una función exponencial.

Para un vector $\mathbf{z}$ de dimensión K, la función softmax se define como:

$$\text{softmax}(z_i) = \frac{exp(z_i)}{\sum_{j=1}^{K} exp(z_j)} \tag{1}$$

Para el vector completo $\mathbf{z}$ se define como:

$$\text{softmax}(\mathbf{z}) = \left[\frac{exp(z_1)}{\sum_{i=1}^{K} exp(z_i)}, \frac{exp(z_2)}{\sum_{i=1}^{K} exp(z_i)}, ..., \frac{exp(z_K)}{\sum_{i=1}^{K} exp(z_i)}\right] \tag{2}$$

El denominador $\sum_{i=1}^{K} exp(z_i)$ se utiliza para normalizar todos los valores en probabilidades. Por ejemplo, para un vector dado:

$$\mathbf{z} = [1{,}25, -1, 0{,}6]$$

Obtenemos:

$$\text{softmax}(\mathbf{z}) = [0{,}614, 0{,}064, 0{,}320]$$

Al igual que ocurre con la función sigmoid, al valor que obtiene la probabilidad más alta se le asigna un 1 y al resto de valores un 0.

Cuando aplicamos softmax para la regresión logística, la entrada (al igual que para la función sigmoid) será el producto escalar entre un vector de peso $\mathbf{b}$ y un vector de entrada $\mathbf{x}$ (más un sesgo b_0). Ahora necesitaremos vectores de peso separados $\mathbf{b}_k$ y sesgos b_{k0} para cada una de las K clases. La probabilidad de cada una de las clases de salida $\hat{y}_k$ se calcula de la siguiente manera:

$$p(y_k = 1|\mathbf{x}) = \frac{exp(\mathbf{b}_k \cdot \mathbf{x} + b_{k0})}{\sum_{j=1}^{K} exp(\mathbf{b}_j \cdot \mathbf{x} + b_{j0})} \tag{3}$$

Podríamos calcular así cada salida por separado utilizando la ecuación (3), sin embargo, es preferible hacerlo mediante procesamiento vectorial. Así que el conjunto de K vectores se representarán mediante una matriz de pesos $\mathbf{B}$ y un vector de sesgo $\mathbf{b}_0$. $\mathbf{B}$ tendrá una dimensión $[K \times p]$, donde K es el número de clases y p es el número de características de entrada. El vector de sego $\mathbf{b}_0$ tiene un valor para cada una de las K clases de salida. Si representamos los pesos de esta manera, podemos calcular $\hat{\mathbf{y}}$, el vector de probabilidades de salida para cada una de las K clases, mediante una sola ecuación:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{B}\mathbf{x} + \mathbf{b}_0) \tag{4}$$

¿Cómo se aprenden los parámetros del modelo, los pesos $\mathbf{B}$ y el sesgo $\mathbf{b}_0$? Esto requiere dos componentes. El primero es una métrica para medir qué tan cercana está la etiqueta actual $\hat{\mathbf{y}}$ a la verdadera etiqueta de referencia $\mathbf{y}$. En lugar de medir la similitud, generalmente hablamos sobre lo contrario: la distancia entre la salida del sistema y la salida de referencia, a lo que llamamos función de pérdida (*loss*) o función de costo (*cost*). Lo segundo que necesitamos es un algoritmo de optimización para actualizar iterativamente los pesos de manera que se minimice esta función de pérdida. El algoritmo estándar para esto es el descenso de gradiente (*gradient descent*).

La función de pérdida para la regresión logística multinomial generaliza la función de pérdida para la regresión logística binaria de 2 a K clases. Recuerdemos que la pérdida de entropía cruzada para la regresión logística binaria es:

$$L(\hat{y}, y) = -\log(p(y|x)) = -[y \log \hat{y} + (1-y)\log(1-\hat{y})] \tag{5}$$

La función de pérdida para la regresión logística multinomial generaliza los dos términos en la ecuación (5) a K términos. Como mencionamos anteriormente, para la regresión multinomial representaremos tanto $\mathbf{y}$ como $\hat{\mathbf{y}}$ como vectores. La etiqueta verdadera $\mathbf{y}$ es un vector con K elementos, cada uno correspondiente a una clase, con $y_c = 1$ si la clase correcta es c, y con todos los otros elementos de y siendo 0. El clasificador producirá un vector de estimación con K elementos $\hat{\mathbf{y}}$, donde cada elemento $\hat{y_k}$ representa la probabilidad estimada $p(y_k = 1|x)$. La función de pérdida para un solo ejemplo $\mathbf{x}$, generalizando a partir de la regresión logística binaria, es la suma de los logaritmos de las K clases de salida, cada uno ponderado por su probabilidad y_k:

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k \tag{6}$$

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\log \frac{\mathbf{b}_c \cdot \mathbf{x} + b_{c0}}{\sum_{j=1}^{K} exp(\mathbf{b}_j \cdot \mathbf{x} + b_{j0})} \quad \text{siendo c la clase correcta} \tag{7}$$

El objetivo con el descenso de gradiente es encontrar los pesos óptimos: minimizar la función de pérdida que hemos definido para el modelo., es decir, encontrar el conjunto de pesos que minimice la función de pérdida, promediando todos los ejemplos.

$$\mathbf{b} = \text{argmin}\frac{1}{m}\sum_{i=1}^{m} L \tag{8}$$

¿Cómo encontraremos el mínimo de esta (o cualquier) función de pérdida? El descenso del gradiente es un método que encuentra un mínimo de una función al determinar en qué dirección (en el espacio de los parámetros $\mathbf{b}$) la pendiente de la función está aumentando más abruptamente, y moviéndose en la dirección opuesta. La intuición es que si estás caminando en un cañón e intentas descender lo más rápido posible hacia el río en el fondo, podrías mirar a tu alrededor 360 grados, encontrar la dirección donde el suelo está inclinado más pronunciadamente y caminar cuesta abajo en esa dirección. Para la regresión logística, esta función de pérdida es convexa. Una función convexa tiene a lo sumo un mínimo; no hay mínimos locales en los que quedar atrapado, por lo que el descenso del gradiente a partir de cualquier punto está garantizado a encontrar el mínimo. La magnitud del desplazamiento en el descenso del gradiente es el valor del gradiente $\nabla L(\hat{\mathbf{y}}, \mathbf{y})$ ponderado por una tasa de aprendizaje η. El cambio que realizamos en nuestro parámetro es la tasa de aprendizaje multiplicada por el gradiente (o la pendiente,en el caso de una sola variable):

$$\mathbf{b}^{t+1} = \mathbf{b}^t - \eta \nabla L(\hat{\mathbf{y}}, \mathbf{y}) \tag{9}$$

Como ventajas en el usa de regresión logística tenemos:

- Permite trabajar con variables dependientes categóricas con más de dos categorías, lo que la hace adecuada para una variedad de escenarios de clasificación.
- La regresión logística multinomial suele considerarse un análisis atractivo porque no supone normalidad, linealidad u homocedasticidad.
- No es solo un modelo de clasificación, sino que también nos da las diferentes probabilidades de pertenencia a distintas categorías. Esta es una gran ventaja sobre los modelos que solo pueden proporcionar la clasificación final.
- Permite controlar y ajustar los efectos de múltiples variables predictoras en cada categoría de la variable dependiente.

Algunas de las desventajas que tiene son:

- Al manejar más categorías, se necesita una cantidad suficiente de datos en cada categoría para obtener estimaciones precisas y evitar problemas de sobreajuste.
- Con múltiples categorías, la interpretación de los coeficientes y su impacto en cada categoría puede volverse más complicada que en la regresión logística binomial.
- Si las variables predictoras están altamente correlacionadas, puede aumentar la multicolinealidad, lo que dificulta la interpretación de los efectos individuales.
- Si hay desequilibrio entre las categorías de la variable dependiente, el modelo puede tener dificultades para predecir las categorías minoritarias.
- puede sufrir de separación completa. Si hay una característica que separe perfectamente las dos clases, el modelo de regresión logística ya no puede ser entrenado. Esto se debe a que el peso de esa característica no convergería, porque el peso óptimo sería infinito.

Para el ejemplo numérico vamos a utilizar los datos 'mobile_train.csv' sacados del siguiente enlace de kaggle. Estos datos representan una amplia gama de características de teléfonos móviles. El objetivo es crear un modelo de regresión logística multinomial que pueda predecir la categoría de precio a la que pertenece cada teléfono móvil. Las categorías de precio son: 0 (precio bajo), 1 (precio medio), 2 (precio alto) y 3 (precio muy alto). El conjunto de datos dispone de las siguientes variables explicativas:

1. battery_power: Capacidad de la batería del teléfono en mAh (milivatios-hora).
2. blue: Indicador binario (0 o 1) que representa si el teléfono tiene capacidad Bluetooth (1 si tiene, 0 si no tiene).
3. clock_speed: Velocidad del reloj del procesador del teléfono en GHz (gigahercios).
4. dual_sim: Indica si el teléfono tiene capacidad para dos tarjetas SIM (1 si tiene, 0 si no tiene).
5. fc: Resolución de la cámara frontal en megapíxeles.
6. four_g: Indica si el teléfono es compatible con la red 4G (1 si es compatible, 0 si no lo es).
7. int_memory: Memoria interna del teléfono en GB (gigabytes).
8. m_dep: Profundidad del teléfono en cm (centímetros).
9. mobile_wt: Peso del teléfono en gramos.
10. n_cores: Número de núcleos del procesador del teléfono.

11. pc: Resolución de la cámara principal en megapíxeles.
12. px_height: Altura de resolución de la pantalla en píxeles.
13. px_width: Ancho de resolución de la pantalla en píxeles.
14. ram: Memoria RAM del teléfono en MB (megabytes).
15. sc_h: Altura de la pantalla del teléfono en cm (centímetros).
16. sc_w: Ancho de la pantalla del teléfono en cm (centímetros).
17. talk_time: Tiempo de conversación máximo estimado en horas.
18. three_g: Indica si el teléfono es compatible con la red 3G (1 si es compatible, 0 si no lo es).
19. touch_screen: Indica si el teléfono tiene pantalla táctil (1 si tiene, 0 si no tiene).
20. wifi: Indica si el teléfono es compatible con Wi-Fi (1 si es compatible, 0 si no lo es).

El procedimiento completo se encuentra en el archivo adjunto .ejercicio1.R". Decidimos dividir los datos en dos conjuntos: uno de entrenamiento y otro de prueba para crear tres modelos mediante regresión logística multinomial. El primer modelo utiliza las veinte variables independientes, el segundo solo emplea la variable 'ram', que presenta la mayor correlación con la variable dependiente 'price_range'. El tercer modelo utiliza únicamente las cinco variables independientes más correlacionadas con 'price_range' ('battery_power', 'pc', 'px_height', 'px_width' y 'ram').

Vamos a comentar diferentes medidas para evaluar los modelos:

- Residual Deviance: Es una medida de la discrepancia entre el modelo ajustado y los datos observados. Cuanto menor sea la Residual Deviance, mejor será el ajuste del modelo. Una deviance baja indica que el modelo explica bien la variabilidad en los datos. Es una forma de evaluar cuánta información no está explicada por el modelo.
- AIC (Criterio de Información de Akaike): Es un indicador de la calidad relativa de un modelo estadístico. Cuanto menor sea el valor del AIC, mejor será el modelo. El AIC tiene en cuenta la bondad del ajuste del modelo y la complejidad del mismo, penalizando la inclusión de variables innecesarias. Es una herramienta útil para comparar modelos, donde un modelo con un AIC más bajo se considera preferible.
- Precisión: mide la proporción de predicciones correctas realizadas por el modelo sobre el total de predicciones.

Modelo 1. Veinte variables independientes

- Residual Deviance: 56.64119
- AIC: 182.6412
- Precisión en el conjunto de entrenamiento: 0.99375
- Precisión en el conjunto de pruebas: 0.975

La Residual Deviance relativamente baja sugiere que este modelo explica una cantidad considerable de la variabilidad en los datos. El AIC, aunque no es el más bajo, muestra un ajuste razonable del modelo considerando la complejidad introducida por las 20 variables. Muestra un alta precisión en los datos de entrenamiento con un 99.38 % y en los datos de prueba con un 97.50 %. Esto sugiere un ajuste adecuado del modelo a los datos de entrenamiento y una capacidad considerable para generalizar con los datos de prueba.

Modelo 2. Una variable Independiente (RAM)

- Residual Deviance: 1760.789
- AIC: 1772.789

- Precisión en el conjunto de entrenamiento: 0.75375
- Precisión en el conjunto de pruebas: 0.7525

La Residual Deviance alta indica que este modelo tiene dificultades para explicar la variabilidad de los datos. El valor del AIC es significativamente mayor que en el primer modelo, lo que sugiere que este modelo con una única variable es menos adecuado para describir los datos. La precisión es mucho menor en comparación con el primer modelo. La exactitud del 75.38 % en los datos de entrenamiento y del 75.25 % en los datos de prueba sugiere una limitación en la capacidad predictiva del modelo al utilizar solo una variable independiente.

Modelo 3. Cinco variables independientes ('battery_power', 'pc', 'px_height', 'px_width' y 'ram')

- Residual Deviance: 51.04925
- AIC: 87.04925
- Precisión en el conjunto de entrenamiento: 0.95125
- Precisión en el conjunto de pruebas: 0.98

Tiene la Residual Deviance más baja que los otros modelos, lo que indica un mejor ajuste. El AIC más bajo en este modelo indica que, a pesar de tener menos variables que el primer modelo, logra explicar bien los datos, lo que lo convierte en la opción más sólida. Obtiene una precisión ligeramente menor en los datos de entrenamiento (95.13 %) en comparación con el primer modelo, pero con una precisión algo superior del 98.00 % en los datos de prueba.

Como conclusión podemos sacar que el rango de precios preestablecido de los teléfonos móviles puede ser explicado con la capacidad de la batería, con la resolución de la cámara principal, con la resolución de pantalla y con la memoria RAM del teléfono.

2. La separación completa no se puede dar para una variable respuesta nominal.

Al aplicar regresión logística al análisis de datos, un problema que a menudo puede surgir es el de la separación completa. La separación completa es el fenómeno donde existe una combinación lineal:

$$z_i = x_{i1}\beta_1 + ... + x_{ip}\beta_p \tag{10}$$

En donde si $z_i < k$ y_i siempre será igual a uno, y por el contrario si $z_i > k$ y_i siempre será igual a cero. En una situación de separación completa, las variables predictoras predicen la variable de respuesta de manera perfecta, ya que los datos pueden dividirse completamente mediante una función lineal de las variables predictoras, ubicando todos los resultados positivos en un lado y todos los negativos en el otro. En este escenario, uno o más coeficientes estimados tienden hacia el infinito (positivo o negativo). Esto resulta en que el predictor lineal en sí sea igual a infinito (positivo o negativo), lo que implica que la probabilidad de un resultado positivo sea igual a 1 (o 0) para un subconjunto de datos donde todos los valores observados son 1 (o todos son cero). En esencia, las variables predictoras se ajustan demasiado bien a los datos, permitiendo una clasificación perfecta en un subconjunto de resultados binarios.

Aunque la separación completa se describe comúnmente en la literatura en contextos donde la variable de respuesta es dicotómica, este fenómeno puede extenderse a situaciones con variables de respuesta nominales, especialmente en el marco de la regresión logística multinomial. La separación completa se manifiesta cuando un subconjunto de las variables predictoras, representadas por el subvector $\mathbf{X}_S \in \mathbf{X}$, clasifica de manera determinista cada observación en una de las categorías de $\mathbf{Y}$, siendo $\mathbf{Y} = 1, 2, ..., g$ donde g representa el número de categorías en la variable respuesta. Bajo esta situación, las estimaciones de probabilidad de las variables predictoras no pueden ser obtenidas y tienden hacia infinito (Albert y Anderson, 1984). Sin embargo, en muchos casos, los software

de análisis devuelven estimaciones de valores finitos, informando el último valor obtenido antes de detener la búsqueda de probabilidad. Estos valores no ofrecen información sobre el parámetro poblacional subyacente y, por lo tanto, se debe abstenerse de realizar inferencias basadas en dicho parámetro. Por tanto la afirmación del enunciado es **FALSA**.

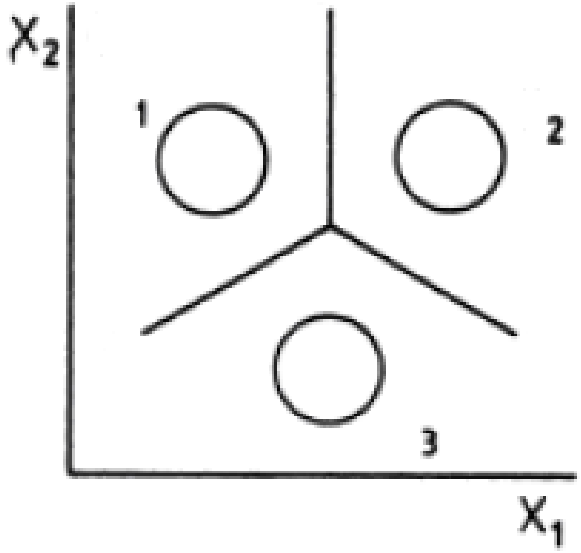

Figura 1: Ejemplo de separación completa para una variable nominal con tres categorías

3. La separación completa no se puede dar con tres variables explicativas.

Atendiendo a la definición de separación completa, si ocurre que con tres variables explicativas lo siguiente:

$$z_i = x_{i1}\beta_1 + x_{i2}\beta_2 + x_{i3}\beta_3$$

En donde si $z_i < k$ y_i siempre será igual a uno (o cero), y por el contrario si $z_i > k$ y_i siempre será igual a cero (o uno), tendremos un caso de separación completa. Por tanto la afirmación del ejercicio es **FALSA**. La separación completa puede ocurrir con cualquier número de variables explicativas en un modelo, incluso con tres o más variables. La separación completa depende de la relación entre las variables explicativas y la variable de respuesta, donde ciertos patrones en los datos pueden permitir la separación perfecta de las categorías de la variable de respuesta basada en combinaciones lineales de las variables explicativas.

Imaginemos una competición compuesta por tres modalidades distintas que representarán las variables explicativas, tales como el desempeño en salto de altura, salto de pértiga y salto de longitud. La variable de respuesta es dicotómica y se refiere a la clasificación (o no) para los Juegos Olímpicos. Supongamos que se asignan puntos de manera proporcional en cada una de las tres modalidades. Si establecemos una puntuación mínima (k puntos) como requisito para clasificarse en dicha competición y acceder a los Juegos Olímpicos, entonces:

$$z_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} > k \implies y = 1, \text{ consigue la clasificación el participante i}$$

$$z_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} < k \implies y = 0, \quad \text{no consigue la clasificación el participante i}$$

Esto sería un ejemplo de separación completa con tres variables explicativas.

4. Se pide comentar qué hace el siguiente código en R e interpretar el resultado que proporciona

```
rm(list=ls())

library(titanic)
data("titanic_train")

df <- data.frame(titanic_train)
summary(df)
str(df)
colSums(is.na(df))
colSums(df=='')
unique(df$Sex)
table(df$Sex)

df$Sexf <- factor(df$Sex)
df$Sexfn = as.numeric(df$Sexf)

df$Sexfn1 <- (df$Sexfn==1)*1
df$Sexfn2 <- (df$Sexfn==2)*1

summary(glm(Survived~Sexfn1, family=binomial, data=df))
summary(glm(Survived~Sexfn2, family=binomial, data=df))
```

`rm(list=ls())` borra todos los objetos guardados en la memoria actual del entorno de trabajo, útil al comenzar un nuevo script para así evitar confusiones y tener un espacio limpio.

`library(titanic)` carga la biblioteca 'titanic', que contiene conjuntos de datos relacionados con el Titanic. `data("titanic_train")` carga el conjunto de datos 'titanic_train' desde la biblioteca 'titanic' a la sesión de trabajo en R. Se trata pues de un conjunto de datos en el cual se describen diferentes aspectos de los pasajeros que fueron a bordo del Titanic en el momento del hundimiento del mismo en el año 1912.

`df <- data.frame(titanic_train)` crea un nuevo data frame llamado 'df' a partir del conjunto de datos 'titanic_train'. `summary(df)` muestra un resumen estadístico de las variables en el data frame 'df'. `str(df)` muestra la estructura del data frame 'df', incluyendo la cantidad de observaciones y variables, así como los tipos de datos de cada variable y los diez primeros valores de cada una de las variables. Vemos que el conjunto de datos 'df' contiene un total de 12 variables con 891 observaciones. Las variables son las siguientes:

- PassengerId: se trata de una variable de tipo *int*. Representa un identificador único asignado a cada pasajero. Este identificador no tiene relación directa con ninguna característica específica del pasajero ni influye en el análisis.
- Survived: variable de tipo *int*. Indica si el pasajero sobrevivió (1) o no (0) al hundimiento del Titanic.
- Pclass: variable de tipo *int*. Hace referencia a la clase socio-económica del pasajero (1 = Primera clase, 2 = Segunda clase, 3 = Tercera clase).
- Name: variable de tipo *chr*. Nombre del pasajero.
- Sex: variable de tipo *chr*. Indica el sexo del pasajero.
- variable de tipo *num*. Indica la edad del pasajero.
- SibSp: variable de tipo *int*. Esta variable cuantifica las relaciones familiares cercanas de un pasajero en términos de hermanos o cónyuges que estaban presentes a bordo del Titanic. Por ejemplo, si un pasajero tiene 1 en esta columna, significa que tenía un hermano/a o cónyuge a bordo. Si tiene 0, indica que viajaba solo en términos de hermanos y cónyuges.

- Parch: variable de tipo *int*: Esta variable representa la cantidad de relaciones familiares directas relacionadas con padres o hijos que un pasajero tenía en el Titanic. Por ejemplo, si un pasajero tiene 1 en esta columna, significa que tenía uno de sus padres o un hijo a bordo. Si tiene 0, indica que no viajaba con padres o hijos.
- Ticket: variable de tipo *chr*. Identifica el ticket específico utilizado por cada pasajero en el viaje del Titanic.
- Fare: variable de tipo *num*. Representa la tarifa o el costo del billete pagado por cada pasajero para viajar en el Titanic. Esta variable cuantifica el precio que cada persona pagó por su pasaje.
- Cabin: variable de tipo *chr*. Indica el número de cabina del pasajero.
- Embarked: Variable de tipo *chr*. Hace referencia al puerto de embarque (C = Cherbourg, Q = Queenstown, S = Southampton).

A partir del análisis realizado, se pueden deducir algunos aspectos interesantes con respecto a las variables. Sobrevive alrededor del 38.38 % de los pasajeros. Más de la mitad de ellos viaja en tercera clase. La media de edad es de 29.70 años. En promedio, cada pasajero tiene 0.523 hermanos o cónyuges a bordo. La media de relaciones directas (padres o hijos) entre pasajeros es de 0.3816. El precio promedio por billete es de 32.20 libras, mientras que la mediana del precio es de 14.45 libras.

`colSums(is.na(df))` calcula la cantidad de valores faltantes (NA) en cada columna del data frame 'df', vemos que la única columna con datos faltantes (NA) es 'Age' con un total de 177. `colSums(df=="")` calcula la cantidad de valores vacíos en forma de cadenas de texto vacías en cada columna del data frame 'df', la columna 'Cabin' tiene un total de 687 datos vacíos (la mayoría de ellos) y la columna 'Embarked' contiene 2 datos vacíos. `unique(df$Sex)` muestra los valores únicos en la columna 'Sex' del data frame 'df', muestra los dos únicos datos existentes, 'male' (varón) y 'female' (hembra). `table(df$Sex)` crea una tabla de frecuencia de los valores en la columna 'Sex' del data frame 'df', habiendo un total de 314 pasajeros etiquetados como 'female' y 577 como 'male'.

`df$Sexf <- factor(df$Sex)` crea una nueva columna 'Sexf' en 'df' que convierte la columna 'Sex' en un factor, es decir, se crea una representación numérica de esos valores categóricos. `df$Sexfn = as.numeric(df$Sexf)` crea una nueva columna 'Sexfn' en 'df' que convierte el factor 'Sexf' en valores numéricos. Al valor numérico asociado a 'female' es 1, mientras que el valor númerico asociado a 'male' es 2.

`df$Sexfn1 <- (df$Sexfn==1)*1` crea una nueva columna 'Sexfn1' en 'df' que tiene un valor de 1 si 'Sexfn' es igual a 1, y 0 en otro caso, es decir, asigna el valor 1 a los datos categorizados como 'female' y 0 a 'male'. `df$Sexfn2 <- (df$Sexfn==2)*1` crea una nueva columna 'Sexfn2' en 'df' que tiene un valor de 1 si 'Sexfn' es igual a 2, y 0 en otro caso, al contrario que en el caso anterior asigna el valor 1 a 'male' y 0 a 'female'.

`summary(glm(Survived Sexfn1, family=binomial, data=df))` ajusta un modelo de regresión logística donde 'Survived' es la variable dependiente y 'Sexfn1' es la variable independiente y muestra un resumen de este modelo. `summary(glm(Survived Sexfn2, family=binomial, data=df))` Ajusta un modelo de regresión logística donde 'Survived' es la variable dependiente y 'Sexfn2' es la variable independiente y muestra un resumen de este modelo.

El objetivo de este código es crear un modelo de aprendizaje supervisado de clasificación utilizando regresión logística. Se busca predecir si un pasajero logra sobrevivir o no en función de las variables proporcionadas.

Una observación inicial en el código es la redundancia al utilizar `Sexfn1` y `Sexfn2` ya que ambos contienen la misma información. En el primero, se emplea el valor 1 para categorizar 'female' y 0 para 'male', mientras que en el segundo, sucede lo contrario: 0 representa 'female' y 1 'male'. Por ende, al realizar los ajustes en `summary(glm(Survived Sexfn1, family=binomial, data=df))` y `summary(glm(Survived Sexfn2, family=binomial, data=df))` obtendremos información idéntica.

Al realizar una regresión logística simple, obtenemos una función lineal en la forma $z = b_0 + b_1 x$, que luego se transforma mediante la función sigmoide $\frac{1}{1+exp(-z)}$. Si para `Sexfn1` obtenemos $z = b_0 + b_1 x$ y para `Sexfn2` $z = b'_0 + b'_1 x$, entonces, cuando el valor de x es 1 (indicando 'female') para el primero, en el segundo caso es 0 (también 'female'). Esto nos lleva a $z = b_0 + b_1$ para el

primer caso y $z = b'_0$ para el segundo. Al pasar ambas z por la función sigmoide, obtendríamos la misma probabilidad si $b_0 + b_1 = b'_0$. Siguiendo un razonamiento similar para el caso de los hombres, tendremos $b_0 = b'_0 + b'_1$. Al realizar los ajustes, obtenemos respectivamente los siguientes términos:

$$z = -1{,}4571 + 2{,}5137x \quad \texttt{Sexfn1}$$

$$z = 1{,}0566 - 2{,}5137x \quad \texttt{Sexfn2}$$

Donde se ve que $-1{,}4571 + 2{,}5137 = 1{,}0566$ y $-1{,}4571 = 1{,}0566 - 2{,}5137$. Por lo tanto queda demostrado que ambos ajustes son en realidad el mismo.

Ahora con la función sigmoide podemos calcular las probabilidades de que una persona siendo hombre o mujer sobreviva:

$$f(x = 1) = \frac{1}{1 + exp(-1{,}0566)} = 0{,}7420$$

$$f(x = 0) = \frac{1}{1 + exp(1{,}4571)} = 0{,}1889$$

Lo que nos indica que con una probabilidad de un 74,20 % una persona sobrevive si es mujer y con una probabilidad de un 18,89 % una persona sobrevive si es hombre. Al tener esto en cuanta el modelo lo que va a hacer a la hora de hacer predicciones es que si se trata de una mujer va ha predecir que sobrevive y que si por el contrario se trata de un hombre el modelo va a predecir que no sobrevive. Al calcular la precisión del modelo, es decir, el porcentaje que acierta, obtenemos un 0.7867565.

Otra observación que podemos hacer es que no era necesario realizar un ajuste de regresión logística, habría bastado con calcular las probabilidades condicionadas de cada una de los sexos, es decir, la probalidad de que una persona sobreviva sabiendo que es mujer $P(\text{sobrevive/mujer})$ y la probabilidad de que una persona sobreviva sabiendo que es hombre $P(\text{sobrevive/mujer})$ ya que solo hay una variable predictora y ésta es una variable dummy. Al calcularlas obtenemos los mismos valores que obtuvimos previamente.

$$P(\text{sobrevive/mujer}) = 0{,}7420$$

$$P(\text{sobrevive/hombre}) = 0{,}1889$$

Una vez calculadas las probabilidades condicionadas asignaríamos un 1 (sobrevive) a las mujeres ya que que su probabilidad es mayor que 0.5 y asignaríamos un 0 (no sobrevive) a los hombres ya que su probabilidad es menor que 0.5. Para calcular la precisión del modelo basta con realizar el siguiente calculo:

$$P(\text{Mujer})P(\text{sobrevive/mujer}) + P(\text{Hombre})(1 - P(\text{sobrevive/hombre})) \tag{11}$$

Llegando al mismo resultado (0.7867565) que obtuvimos anteriormente.

Los cálculos realizados en este ejercicio pueden verse en ejercicio4.R

5. Se ha realizado el siguiente diálogo con ChatGPT. Se pide dar una nota numérica de 0 a 10 justificando dicha puntuación a través de una rúbrica.

Para hacer la evaluación de las respuestas, vamos a puntuar cada pregunta del 0 al 10. Cada respuesta tiene em mismo peso, por lo que la puntuación final será la suma de las puntuaciones individuales de cada respuesta dividido entre el número total de preguntas. Para tener una guía más o menos coherente a la hora de evaluar las respuestas vamos a seguir la siguiente rúbrica (Cuadro 1).

Puntuación	Criterios de Evaluación
[0, 2.5)	Contenido insatisfactorio: La respuesta carece completamente de información relevante. La información carece de sentido. Contiene información incorrecta o no relacionada al tema.
[2.5, 5)	Contenido incompleto: poco desarrollado, respuestas muy generales, esquivando ser específico
[5, 7.5)	Contenido adecuado: proporciona información adecuada y relevante. Presenta argumentos razonables o detalles que respaldan el tema. Puede carecer de profundidad o claridad en ciertos aspectos.
[7.5, 10]	Contenido bueno: información completa y bien desarrollada. Presenta argumentos claros y detallados, respaldados por evidencia o ejemplos. Proporciona una comprensión sólida del tema sin imprecisiones.

Cuadro 1: Rúbrica para Evaluar las respuestas

1. **Cuál es la relación entre chi-cuadrado para diferencia de proporciones y regresión logística**

   El primer párrafo lo considero correcto, se usan ambas en el contexto de análisis estadístico y modelado de datos.

   En la segunda parte menciona que la prueba chi-cuadrado se utiliza para evaluar si existe una diferencia significativa entre dos proporciones en una muestra, a lo que yo añadiría: se utiliza para determinar la existencia de una diferencia significativa entre proporciones en una muestra, ya sea entre dos o más grupos distintos. Recalcando así que se puede usar para más de dos grupos.

   En el tercer párrafo, todo lo que se expone es correcto.

   En el cuarto párrafo, nos dice que *la relación entre ellos radica en que la regresión logística puede ser utilizada para modelar la relación entre variables categóricas, similar a cómo se examina la diferencia de proporciones*, sin embargo El test de chi-cuadrado se enfoca en evaluar si existe una diferencia significativa entre las proporciones de dos o más grupos categóricos, y La regresión logística, por otro lado, modela la relación entre una variable dependiente binaria y una o más variables independientes, algunas de las cuales pueden ser categóricas. En la última parte cuando dice *el análisis de los coeficientes del modelo se relaciona con las diferencias en las proporciones entre las categorías de las variables predictoras.* puede ser confuso, yo añadiría: Los coeficientes en regresión logística representan el cambio logarítmico en las odds por cada unidad de cambio en la variable predictora. Cuando se trabaja con variables categóricas, los coeficientes reflejan cómo cada categoría en comparación con la categoría de referencia afecta esas probabilidades.

   En el último parrafo lo que se expone está bien, sin embargo, vuelve a incidir diciendo *el chi-cuadrado para diferencia de proporciones es una prueba específica para evaluar la independencia entre dos variables categóricas en una muestra*, cuando pueden ser más de dos variables categóricas.

   Dado que el contenido en general es correcto con algunas imprecisiones y falta de claridad, le asigno un 6.

2. **¿Numéricamente están relacionadas?**

   Considero que el primer y segundo párrafo lo que se expone es correcto.

   En el tercer párrafo podría ser más específico y aclarar que los coeficientes se obtienen usando el método de máxima verosimilitud. No se aclara tampoco que quiere decir con *Sin embargo, en algunos casos, el estadístico de prueba Wald de la regresión logística se relaciona con el estadístico chi-cuadrado.*, es decir, en que contexto en regresión logística se usa esa prueba y por qué está relacionada con la prueba chi-cuadrado.

   En el último párrafo comenta que no hay una relación numérica directa entre ambas, lo cual es cierto. Vuelve a insistir en que la prueba chi-cuadrado para diferencias de proporciones

se utiliza para comparar dos variables, lo que puede llevar a confusión ya que pueden ser comparadas más de dos variables categóricas.

Es algo confuso a la hora de exponer si están numéricamente relacionadas. Se podría haber dado algún ejemplo sencillo para aclararlo, por ejemplo, si comparamos dos variables categóricas y el test chi-cuadrado muestra que no existen diferencias significativas entre ambas proporciones (indicando independencia), al calcular el odds ratio utilizando el coeficiente obtenido en la regresión logística, es decir, al obtener la exponencial de dicho coeficiente, obtendríamos un valor cercano a uno.

La nota que obtiene es un 6.

3. **¿Cuál es la relación entre chi-cuadrado de razón de verosimilitudes para diferencia de proporciones y regresión logística?**

   Por lo general hace una exposición correcta sobre el test chi-cuadrado de razón de verosimilitudes y de la regresión logística. Se menciona que ambos usan la teoría de verosimilitud y además también se menciona el propósito de cada uno de los procedimientos. Quizás podría haber desarrollado un poco más en que consiste y como realizar los cálculos en el test de razón de verosimilitudes.

   La puntuación obtenida es de un 9.

La nota final de la evaluación de las tres preguntas es de un **7**

6. Se pide describir en qué consiste la fase de limpieza de datos. Se pide además aplicarlo al conjuto de datos siguiente.

```
library(titanic)
data("titanic_train")
```

La limpieza y el preprocesamiento de datos son etapas cruciales en el proceso de la ciencia de datos y, a menudo, consumen una parte significativa del tiempo de un científico de datos. ¿Por qué son tan cruciales? En esencia, los datos son inherentemente confusos. Los conjuntos de datos del mundo real, aquellos que las empresas y organizaciones recopilan a diario, suelen estar plagados de imprecisiones, inconsistencias y valores faltantes. Si se utilizan datos sucios e inexactos para entrenar modelos predictivos, su rendimiento y precisión se verán comprometidos. Más ampliamente, la limpieza y el preprocesamiento de datos son imperativos para garantizar la integridad de los mismos. Estas etapas nos permiten refinar y organizar los datos crudos en un formato más adecuado, lo que posibilita su análisis de manera efectiva y confiable. La integridad de los datos que empleamos en nuestros análisis impacta directamente la validez de nuestras conclusiones. Por tanto, dedicar tiempo a esta fase del proceso puede evitar la generación de conclusiones incorrectas, la toma de decisiones erróneas o el desarrollo de modelos ineficaces. Algunos aspectos clave de la limpieza de datos incluyen:

- **Detección y manejo de valores faltantes**: Identificar y abordar valores faltantes en el conjunto de datos, ya sea imputando valores basados en tendencias o eliminando registros con datos incompletos.
- **Eliminación de duplicados**: Identificar y eliminar entradas duplicadas en los datos para evitar distorsiones en los resultados del análisis.
- **Corrección de errores y coherencia**: Verificar la consistencia de los datos, corregir errores tipográficos o inconsistentes, y asegurar que los datos sigan un formato coherente.
- **Manejo de valores atípicos**: Identificar valores extremos o atípicos que puedan distorsionar los análisis y decidir si deben ser corregidos, eliminados o tratados de manera especial según el contexto.

- **Manejo de datos desbalanceados**: Situación en la que una o más clases en un conjunto de datos están representadas de manera significativamente menor en comparación con otras clases. Esta disparidad puede causar problemas en el entrenamiento de modelos de machine learning, ya que los algoritmos tienden a favorecer la clase mayoritaria y no aprenden de manera efectiva la clase minoritaria. Esto puede resultar en modelos con un rendimiento deficiente al predecir la clase minoritaria. Existen diferentes técnicas para tratar con datos desbalanceados como re-muestreo, ponderación de clases ogeneración sintética de muestras.
- **Normalización o estandarización**: Consiste en ajustar las características de los datos para que estén en una escala común y uniforme. Esta fase es crucial cuando se trabaja con algoritmos sensibles a las magnitudes de las variables o cuando los datos tienen diferentes escalas. La normalización puede realizarse de varias maneras, como la estandarización, que consiste en restar la media y dividir por la desviación estándar de cada característica, o la normalización min-max, que reescala los valores al intervalo deseado.
- **Codificación de variables categóricas** Las variables categóricas representan un desafío al construir modelos de aprendizaje automático porque estos modelos, en esencia, son algebraicos. Como resultado, requieren entradas numéricas. Esto implica la transformación de variables categóricas a un formato numérico adecuado.

La fase de limpieza de datos del Titanic se encuentra en el archivo ejercicio6.R.

La primera tarea que vamos a hacer a la hora de realizar la limpieza de datos del Titanic es la búsqueda de valores nulos, para eso usamos la siguiente función:

```
colSums(is.na(titanic_train))
```

Esta función nos proporciona el recuento de datos 'na' en cada columna. En la columna 'Age', obtenemos 177 valores 'na'. Existen diversas estrategias para abordar estos valores nulos. Una de ellas implica eliminar estos datos faltantes. No obstante, con 177 valores, esta cantidad es significativa, por lo que en este caso específico optaremos por la estrategia de rellenar estos datos faltantes. Una opción sería completarlos con la media (o mediana) de los valores existentes en la columna con datos faltantes. Considerando que la edad podría haber sido un factor relevante para la supervivencia, antes de completar los datos faltantes, evaluaremos la correlación entre la edad y otras variables. Para eso usamos la siguiente función:

```
cor(titanic_train_1[,c("Age","Pclass","SibSp","Parch","Fare", "Sex_numeric")])
```

Observamos que la edad muestra una correlación significativa con 'PClass' (-0.369) y con 'SibSp' (-0.308). Esto indica que, por un lado, a medida que el número de clase disminuye, la edad tiende a aumentar. Lo mismo se puede decir con respecto a 'SibSp'; si un pasajero tiene varios hermanos a bordo, es probable que su edad sea inferior a la media. Por lo tanto, en lugar de completar los datos con la media de 'Age', optaremos por llenarlos en función de los valores específicos de 'PClass' y 'SibSp' que posea cada dato. Para eso vamos a usar el paquete 'mice', una herramienta utilizada para manejar y tratar valores faltantes en conjuntos de datos. En lugar de simplemente eliminar o rellenar los valores faltantes con un solo valor (como la media o la mediana), el enfoque de múltiples imputaciones busca capturar la incertidumbre asociada con los valores faltantes mediante la generación de múltiples conjuntos de datos completos. Este proceso permite simular una estimación más precisa de los valores perdidos.

```
# Creamos un nuevo conjunto que contiene solo las columnas "Age", "Pclass" y "SibSp"
simple <- titanic_train[c("Age","Pclass","SibSp")]

# realizamos la imputación múltiple en el conjunto de datos "simple"
imp_mice <- mice(simple,seed=144,m=5,maxit=10)

# Muestra los valores imputados para la columna "Age" del conjunto de datos imputados.
imp_mice$imp$Age
```

```
imputed <- complete(imp_mice,1)
imputed

# Reemplazamos la columna "Age" original por la modificada
titanic_train$Age <- imputed$Age
```

Una vez completados los datos faltantes de la columna 'Age' pasamos a ver si existen datos duplicados, para eso usamos la función siguiente:

```
any(duplicated(titanic_train))
```

Al obtener 'FALSE' deducimos automáticamente que no existen datos duplicados.

Ahora pasamos a la fase de corrección de errores o de datos faltos de coherencia. Para eso buscamos primero si existen cadenas vacías en el caso de columnas con datos categóricos, para eso usamos la siguiente sentencia:

```
colSums(titanic_train = '')
```

Obtenemos un total de 687 valores vacíos en la columna 'Cabin' y 2 en la columna 'Embarked'. Con respecto a la columna 'Cabin' debido por un lado a la inmensa cantidad de datos vacíos junto con la poca relevancia que pudiera tener a priori el número de camarote decidimos eliminar dicha columna completamente:

```
titanic_train <- titanic_train[, -which(names(titanic_train) == 'Cabin')]
```

Para el caso de 'Embarked' al haber dos datos vacíos tenemos la opción de eliminar esos datos o por el contrario rellenarlos con por ejemplo la categoría más repetida, que en este caso es S con un total de 644. Optamos por la primera opción debido a que son solo dos datos y no podemos adivinar con certeza el puerto de procedencia.

```
# Eliminamos los datos vacíos de Embarked
titanic_train <- titanic_train[titanic_train$Embarked != '', ]
```

Ahora pasaremos a estudiar columna por columna. Eliminamos directamente 'PassengerId' ya que no aporta información alguna. Después de revisar las columnas 'Survived', 'PClass', 'SibSp' y 'Parch' las dejamos sin modificar. A pesar de que pudiera haber algo de información relevante, tales como títulos de algunos pasajeros (Capt, Don, Mayor, Sir, etc..) decidimos prescindir de la columna 'Name'. La columnna 'Sex' la transformamos a numérica:

```
titanic_train$Sex <- ifelse(titanic_train$Sex == 'female', 1, 0)
```

También decidimos no tener en cuenta la columna 'Ticket' ya que son combinaciones de letras y números, siendo probablemente un tipo de enumeración de los pasajes. La variable 'Embarked' es de tipo categórica nominal, así que aplicamos one hot encoding para transformarla a numérica:

```
# Convertimos la columna 'Embarked' en variables dummy
dummies_embarked <- model.matrix(~ Embarked - 1, data = titanic_train)

# Unimos las variables dummy al conjunto de datos
titanic_train <- cbind(titanic_train, dummies_embarked)

# Borramos la columna Embarked
titanic_train <- titanic_train[, -which(names(titanic_train) == 'Embarked')]
```

Al usar un diagrama de caja en la columna 'Fare' vemos una cantidad reseñable de outliers. Decidimos hacer un histograma y un histograma de tipo logarítmico para compararlos y vemos que en el segundo se asemeja más a una distribución normal y se suaviza la cantidad de outliers. Por lo tanto decidimos reemplazar la columna 'Fare' por su respectiva logarítmica.

```
boxplot(titanic_train$log_Fare)
# vemos la presencia de outliers

# Histograma de la columna 'Fare'
hist(titanic_train$Fare, breaks = 25, col = "skyblue")

# Histograma logarítmico de la columna 'Fare'
hist(log(titanic_train$Fare + 1), breaks = 25, col = "skyblue")

# Sustituimos la columna Fare por su equivalente logaritmico
titanic_train$log_Fare <- log1p(titanic_train$Fare)

# Borramos la columna Fare
titanic_train <- titanic_train[, -which(names(titanic_train) == 'Fare')]
```

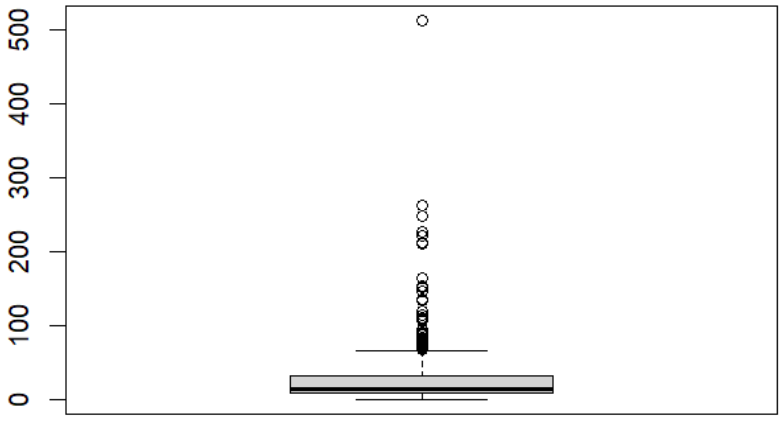

Figura 2: Diagrama de caja de la variable 'Fare'

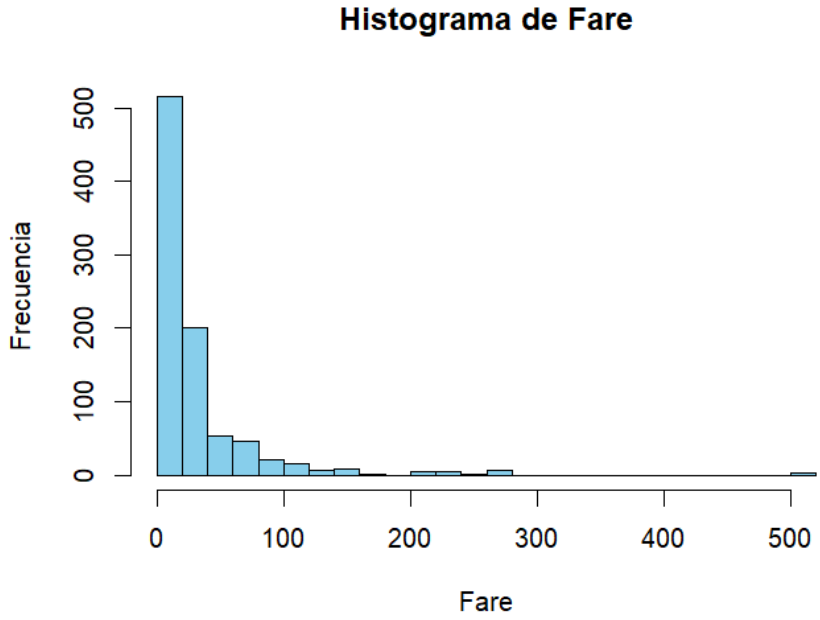

Figura 3: Histograma de la variable 'Fare'

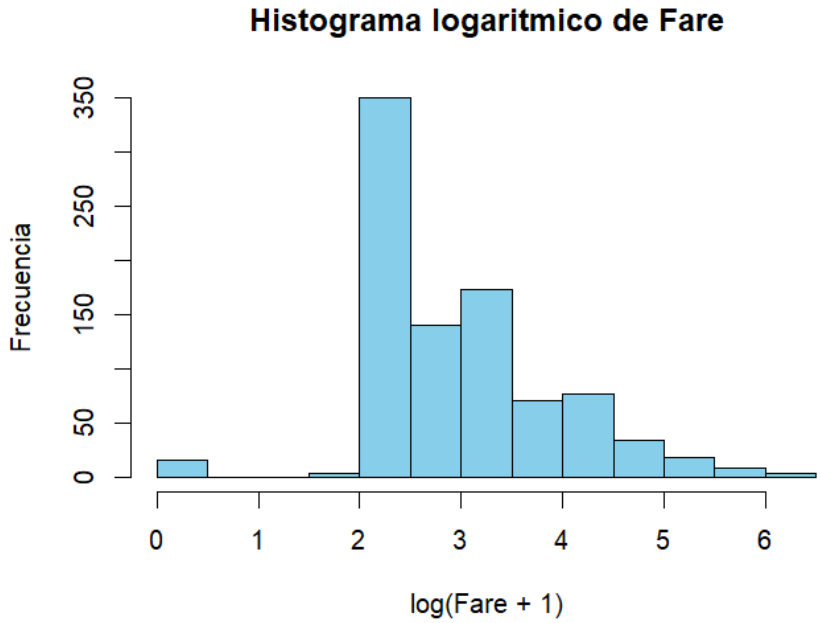

Figura 4: Histograma del logarítmo de la variable 'Fare'

Por último, aplicamos la técnica de scaling Min-Max a las variables, es decir, normalizamos todas las variables independientes para que se encuentren en la misma escala, en este caso limitando sus valores entre 0 y 1. Esto facilita la comparación y el análisis de las variables, ya que todas comparten el mismo rango de valores, lo que ayuda a evitar que características con escalas diferentes influyan de manera desproporcionada en el modelo.

```
library(caret)
```

```
# Definimos el preprocesamiento con el método de escalado Min-Max
preprocess <- preProcess(titanic_train, method = "range")

# Aplicamos el preprocesamiento a los datos
scaled_data <- predict(preprocess, titanic_train)
```

Una vez aplicados todos estos cambios, los datos están listos para poder ser modelados, en nuestro caso en el archivo ejercicio6.R optamos por regresión logística.

Referencias

[1] Gareth James, Daniela Witten, Trevor Hastie y Robert Tibshirani «An Introduction to Statistical Learning, Second Edition» , 2023.

[2] Scott J. Cook, John Niehaus and Samantha Zuhlke: A warning on separation in multinomial logistic models. Recuperado de https://journals.sagepub.com/doi/pdf/10.1177/2053168018769510

[3] Robert G. Clark, Wade Blanchard, Francis K.C, Hui, Ran Tian, Haruka Woods: Dealing with complete separation and quasi-complete separation in logistic regression for linguistic data. Recuperado de https://www.sciencedirect.com/science/article/pii/S2772766123000046

[4] Juan C. Correa, Marisol Valencia: La separación en regresión logística, una solución y aplicación. Universidad Nacional de Colombia, 2011.

[5] Carolyn J., Anderson Leslie Rutkowski: MULTINOMIAL LOGISTIC REGRESSION. Recuperado en https://cja.education.illinois.edu/docs/default-source/carolyn-anderson/edpsy589/lectures/8_multicategory_logit/anderson_and_rutkowski.pdf?sfvrsn=26d81088_0

[6] Data Science Horizons: Data Cleaning and Preprocessing. Recuperado en https://datasciencehorizons.com/wp-content/uploads/2023/06/Data_Cleaning_and_Preprocessing_for_Data_Science_Beginners_Data_Science_Horizons_2023_Data_Science_Horizons_Final_2023.pdf

[7] Análisis de Tablas de Contingencia. Recuperado en https://rua.ua.es/dspace/bitstream/10045/8139/1/CONTINGENCIA.pdf