MÁSTER EN CIENCIA DE DATOS

TIPOLOGÍA DE DATOS (PRÁCTICA #2)

Integrantes:

Fernando Cachadiña
fcachadinag@uoc.edu

Héctor Bastidas Vallejo
hbastidasv@uoc.edu

Tema: Limpieza y análisis de datos

Fecha: 09 de junio de 2020

ÍNDICE

1 OBJETIVOS DE LA ACTIVIDAD

Los objetivos concretos de esta práctica son:

- Aprender a aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios o multidisciplinares.
- Saber identificar los datos relevantes y los tratamientos necesarios (integración, limpieza y validación) para llevar a cabo un proyecto analítico.
- Aprender a analizar los datos adecuadamente para abordar la información contenida en los datos.
- Identificar la mejor representación de los resultados para aportar conclusiones sobre el problema planteado en el proceso analítico.
- Actuar con los principios éticos y legales relacionados con la manipulación de datos en función del ámbito de aplicación.
- Desarrollar las habilidades de aprendizaje que les permitan continuar estudiando de un modo que tendrá que ser en gran medida autodirigido o autónomo.
- Desarrollar la capacidad de búsqueda, gestión y uso de información y recursos en el ámbito de la ciencia de datos.

2 DESCRIPCIÓN DEL DATASET

El dataset que analizaremos en esta práctica está relacionado con el hundimiento del Titanic en 1912. El conjunto de datos es de libre acceso y está disponible en la plataforma Kaggle, el cual es utilizado para competencias de machine learning donde el principal objetivo *es predecir cuales pasajeros sobrevivieron al naufragio del Titanic.*

Este hundimiento es una de las catástrofes marítimas más infames de la historia que ocurrió el 15 de abril de 1912 durante su viaje inaugural, el ampliamente considerado "insumergible" RMS Titanic se hundió después de chocar con un iceberg. Desafortunadamente, no había suficientes botes salvavidas para todos a bordo, lo que resultó en la muerte de 1502 de 2224 pasajeros y tripulantes.

Si bien hubo algún elemento de suerte involucrado en la supervivencia, parece que algunos grupos de personas tenían más probabilidades de sobrevivir que otros. Es decir, se pretende analizar cuáles variables tuvieron mayor influencia en la supervivencia de un pasajero cuando ocurrió el naufragio.

El conjunto de datos recopila información de 891 pasajeros donde se especifica mediante una variable independiente si el pasajero sobrevivió o no. A continuación se presenta una tabla con el detalle de cada una de las variables presentes en el dataset.

Variable	Definición	Clave
PassengerId	ID de cada pasajero en el dataset	
Survived	Supervivencia	0 = No, 1 = Sí
Pclass	Clase del boleto	1 = Primera clase 2 = Segunda clase 3 = Tercera clase
Name	Nombre completo y a veces incluye el apellido de soltero	
Sex	Sexo (masculino o femenino)	
Age	Edad en años	
SibSp	Número de hermanas/cónyuges abordo en el Titanic	
Parch	Número de padres/hijos abordo en el Titanic	
Ticket	Número de boleto	
Fare	Tarifa del boleto	
Cabin	Número de la cabina	
Embarked	Puerto de embarque	C = Cherbourg, Q = Queenstown, S = Southampton

- *Pclass:*

La clase del boleto es un indicio del estatus socioeconómico del pasajero

Primera clase: alto
Segunda clase: medio
Tercera clase: bajo

- *Age:*

La edad es fraccional si es menor a 1.

- *SibSp:*

El conjunto de datos presenta las relaciones familiares en la siguiente manera:

Hermano: hermano, hermana, hermanastro o hermanastra
Esposo: marido o esposa (se ignoran novios y enamorados)

- *Parch:*

El conjunto de datos presenta las relaciones familiares en la siguiente manera:

Pariente: padre o madre
Niños: hijo, hija, hijastro o hijastra
Algunos niños viajaron solo con la niñera, para ellos parch = 0

3 LIMPIEZA DE DATOS

3.1 VALORES PERDIDOS Y CEROS

- **Solución en R:**

En primer lugar realizamos la lectura del fichero con la función *read.csv()* la cual nos devuelve un dataframe en el ambiente R. Los datos perdidos son codificados con NA.

```
# Lectura del fichero .csv
titanicTrainData <- read.csv("train.csv")
```

Examinamos el tipo de datos con los que R ha interpretado a cada variable mediante el uso de la función *str()* que muestra la estructura interna del dataframe almacenado en el paso anterior.

```
# Estructura del dataframe
str(titanicTrainData)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520 629 417 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 86 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

Para desplegar los valores resumen y perdidos de cada variable utilizamos la función *summary()*.

```
# Resumen y valores perdidos de cada variable
summary (titanicTrainData)
```

```
##   PassengerId       Survived          Pclass     
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000  
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000  
##  Median :446.0   Median :0.0000   Median :3.000  
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309  
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000  
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000  
##                                                  
##                     Name         Sex         Age       
##  Abbing, Mr. Anthony   :  1   female:314   Min.   : 0.42  
##  Abbott, Mr. Rossmore Edward  :  1   male  :577   1st Qu.:20.12  
##  Abbott, Mrs. Stanton (Rosa Hunt)  :  1                Median :28.00
```

```
##  Abelson, Mr. Samuel                 :  1                    Mean   :29.70
##  Abelson, Mrs. Samuel (Hannah Wizosky) :  1                  3rd Qu. :38.00
##  Adahl, Mr. Mauritz Nils Martin      :  1                    Max.   :80.00
##  (Other)                             :885                    NA's   :177
```

```
##      SibSp           Parch           Ticket         Fare
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00
##  Max.   :8.000    Max.   :6.0000   CA 2144 :  6   Max.   :512.33
##                                   (Other) :852
```

```
##          Cabin     Embarked
##                :687    :  2
##  B96 B98       :  4   C :168
##  C23 C25 C27   :  4   Q : 77
##  G6            :  4   S :644
##  C22 C26       :  3
##  D             :  3
##  (Other)       :186
```

Se observa que hay valores perdidos o vacíos en tres variables: “Age”, “Cabin” y “Embarked”. Para el caso de la variable “Age” que tiene 177 valores perdidos hemos optado por aplicar imputación de valores considerando que las observaciones guardan relación entre sí. Para este fin se toman en cuenta los k vecinos más similares que para nuestro caso serán los 3 más cercanos. La función que permite realizar esta imputación en R es kNN de la librería VIM.

```
# Imputación de valores con la función kNN ()
titanicTrainData$Age <- kNN(titanicTrainData, variable = c('Age'), k = 3)$Age
```

La variable “Cabin” contiene 687 valores vacíos pero por tratarse del número de cabina del pasajero hemos decido dejarla tal y como se encuentra sin realizar ninguna imputación.

Finalmente, la variable “Embarked” tiene 2 valores vacíos que han sido reemplazados con el valor más frecuente dentro del dataset que es “S” correspondiente a Southampton.

```
# Imputación de valores en la variable "Embarked" con el valor más frecuente
suppressWarnings(suppressMessages(titanicTrainData[titanicTrainData==""]<-"S"))
```

- **Solución en Python:**

Cargamos el Daset en un Dataframe de la librería Pandas:

```
import pandas as pd

titanicTraindata = pd.read_csv("train.csv")
```

Obtenemos la estructura de capturada del dataset dentro del DataFrame.

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 12 columns):

PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object

dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

Encontramos el número de variables (columnas del DataFrame) que contienen valores perdidos

```
# La cantidad por columnas es:

titanicTraindata.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
```

Fare	0
Cabin	687
Embarked	2

Vamos a imputar los valores del campo “Age” mediante KNN en base a los tres valores más próximos por distancia.

```
from sklearn.impute import KNNImputer as KNNIM

# Creamos un DF auxuliar con los campos numéricos que nos permitan hacer una imputación por distancia euclidea
aux_df = titanicTraindata[['Age','Pclass','SibSp','Parch','Fare']]

imputer = KNNIM(n_neighbors=3)
aux_df2 = imputer.fit_transform(aux_df)

nueva_age = (    (aux_df2[:,0])  )

print (nueva_age.shape)

# Redondeamos las fechas a la parte entera de la edad.
nueva_age = nueva_age.astype(int)

print (np.unique(nueva_age))
```

Como resultado vemos que no tenemos ya valores NaN en el campo “Age” con 891 valores.

```
(891,)
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 70 71 74 80]
```

Ahora imputamos los valores a DF original , en el campo “Age” con los valores de arriba.

```
ftitanicTraindata[['Age']] = (nueva_age)
```

Respecto a ‘Embarked’ asignamos el valor más frecuente = “S”

```
# Para el valor Embartked asignamos el valor más comun = "S"

titanicTraindata[['Embarked']] = titanicTraindata[['Embarked']].fillna('S')
```

4.2 VALORES EXTREMOS

- **Solución en R:**

Con el objetivo de identificar los valores extremos presentes en el dataframe hacemos uso de la función *boxplot.stats()*. Esta función arroja las estadísticas resultantes del diagrama de caja de la variable de interés de la cual seleccionaremos solo los valores atípicos ($out).

```
# Valores extremos de la variable "Age"
boxplot.stats(titanicTrainData$Age)$out
```

```
## [1] 66.0 71.0 70.5 71.0 80.0 70.0 70.0 74.0
```

```
# Valores extremos de la variable "SibSp"
boxplot.stats(titanicTrainData$SibSp)$out
```

```
##  [1] 3 4 3 3 4 5 3 4 5 3 3 4 8 4 4 3 8 4 8 3 4 4 4 4 8 3 3 5 3 5 3 4 4 3 3 5 4 3
## [39] 4 8 4 3 4 8 4 8
```

```
# Valores extremos de la variable "Parch"
boxplot.stats(titanicTrainData$Parch)$out
```

```
##   [1] 1 2 1 5 1 1 5 2 2 1 1 2 2 2 1 2 2 2 3 2 2 1 1 1 1 2 1 1 2 2 1 2 2 2 1 2 1
##  [38] 1 2 1 4 1 1 1 1 2 2 1 2 1 1 1 2 1 1 2 2 2 1 1 2 2 1 2 1 1 1 1 1 1 1 2 1 2
##  [75] 2 1 1 2 1 1 2 1 1 1 1 2 1 1 1 4 1 1 2 2 2 2 2 1 1 1 2 2 1 1 2 2 3 4 1 2 1
## [112] 1 2 1 2 1 2 1 1 2 2 1 1 1 1 2 2 2 2 2 2 1 1 2 1 4 1 1 2 1 2 1 1 2 5 2 1 1
## [149] 1 2 1 5 2 1 1 1 2 1 6 1 2 1 2 1 1 1 1 1 1 1 3 2 1 1 1 1 2 1 2 3 1 2 1 2 2
## [186] 1 1 2 1 2 1 2 1 1 1 2 1 1 2 1 2 1 1 1 1 3 2 1 1 1 1 5 2
```

```
# Valores extremos de la variable "Fare"
boxplot.stats(titanicTrainData$Fare)$out
```

```
##   [1]  71.2833 263.0000 146.5208  82.1708  76.7292  80.0000  83.4750  73.5000
##   [9] 263.0000  77.2875 247.5208  73.5000  77.2875  79.2000  66.6000  69.5500
##  [17]  69.5500 146.5208  69.5500 113.2750  76.2917  90.0000  83.4750  90.0000
##  [25]  79.2000  86.5000 512.3292  79.6500 153.4625 135.6333  77.9583  78.8500
##  [33]  91.0792 151.5500 247.5208 151.5500 110.8833 108.9000  83.1583 262.3750
##  [41] 164.8667 134.5000  69.5500 135.6333 153.4625 133.6500  66.6000 134.5000
##  [49] 263.0000  75.2500  69.3000 135.6333  82.1708 211.5000 227.5250  73.5000
##  [57] 120.0000 113.2750  90.0000 120.0000 263.0000  81.8583  89.1042  91.0792
##  [65]  90.0000  78.2667 151.5500  86.5000 108.9000  93.5000 221.7792 106.4250
##  [73]  71.0000 106.4250 110.8833 227.5250  79.6500 110.8833  79.6500  79.2000
##  [81]  78.2667 153.4625  77.9583  69.3000  76.7292  73.5000 113.2750 133.6500
##  [89]  73.5000 512.3292  76.7292 211.3375 110.8833 227.5250 151.5500 227.5250
##  [97] 211.3375 512.3292  78.8500 262.3750  71.0000  86.5000 120.0000  77.9583
## [105] 211.3375  79.2000  69.5500 120.0000  93.5000  80.0000  83.1583  69.5500
## [113]  89.1042 164.8667  69.5500  83.1583
```

Analizando la variable “Age” se valida que los valores extremos están entre 66 y 74 años que es un rango de edad que sí puede darse. De igual forma, los valores extremos de las variables “SibSp” y “Parch” también pueden ser reales porque lo que no se efectúa ninguna acción sobre las mismas. Finalmente, la variable “Fare” presenta 3 observaciones con valores alrededor de 500, sin embargo se comprueba que los registros corresponde al boleto “PC 17755” por lo que también se tratan de valores reales.

En base a lo observado, se opta por mantener los valores tal y como se presentan en el dataset sin aplicar ninguna eliminación o imputación.

- **Solución en Python:**

En Python no hay un método tan claro como en R para el cálculo de los “outliers” desde una función.
Por ello hemos creado nuestra propia función para calcular los “outliers” en base a la definición de IQR +/- Q3

```
#Cálculo de los márgenes de decisión de los valores “outliers”

# Q3 +1.5   IQR
# Q1 −1.5   IQR

def limites_outliers (x):
    Q3 = np.percentile(x, 75)
    Q1 = np.percentile(x, 75)
    IQR = np.subtract(np.percentile(x, 75),np.percentile(x, 25))
    lim_inf = Q1 - 1.5 *IQR
    lim_sup = Q3 + 1.5 *IQR
    resultado = (lim_inf,lim_sup)
    return resultado
```

En base a esta función buscamos los valores del cada campo del Dataset que cumplen que

- O son superiores al valor calculado de “lim_sup”
- O son inferiores al valor calculado de “lim_inf”

Veamos para el campo: “SibSp“

```
## Encontramos los valores outliers del dataset para la variable "SibSp"

aux = titanicTraindata['SibSp']
aux2 = (limites_outliers (aux))
aux2 = titanicTraindata.loc[ (titanicTraindata['SibSp'] < aux2[0]) | (titanicTraindata['SibSp'] > aux2[1]) ]

print (list(aux2.SibSp))
```

[3, 4, 3, 3, 4, 5, 3, 4, 5, 3, 3, 4, 8, 4, 4, 3, 8, 4, 8, 3, 4, 4, 4, 4, 8, 3, 3, 5, 3, 5, 3, 4, 4, 3, 3, 5, 4, 3, 4, 8, 4, 3, 4, 8, 4, 8]

Veamos para el campo: "Fare"

```
## Encontramos los valores outliers del dataset para la variable "Fare"

campo = 'Fare'
aux = titanicTraindata[campo]
aux2 = (limites_outliers (aux))
aux2 = titanicTraindata.loc[ (titanicTraindata[campo] < aux2[0]) | (titanicTraindata[campo] > aux2[1]) ]

print (list(aux2[campo]))
```

[71.2833, 263.0, 146.5208, 82.1708, 76.7292, 80.0, 83.475, 73.5, 263.0, 77.2875, 247.5208, 73.5, 77.2875, 79.2, 66.6, 69.55, 69.55, 146.5208, 69.55, 113.275, 76.2917, 90.0, 83.475, 90.0, 79.2, 86.5, 512.3292, 79.65, 153.4625, 135.6333, 77.9583, 78.85, 91.0792, 151.55, 247.5208, 151.55, 110.8833, 108.9, 83.1583, 262.375, 164.8667, 134.5, 69.55, 135.6333, 153.4625, 133.65, 66.6, 134.5, 263.0, 75.25, 69.3, 135.6333, 82.1708, 211.5, 227.525, 73.5, 120.0, 113.275, 90.0, 120.0, 263.0, 81.8583, 89.1042, 91.0792, 90.0, 78.2667, 151.55, 86.5, 108.9, 93.5, 221.7792, 106.425, 71.0, 106.425, 110.8833, 227.525, 79.65, 110.8833, 79.65, 79.2, 78.2667, 153.4625, 77.9583, 69.3, 76.7292, 73.5, 113.275, 133.65, 73.5, 512.3292, 76.7292, 211.3375, 110.8833, 227.525, 151.55, 227.525, 211.3375, 512.3292, 78.85, 262.375, 71.0, 86.5, 120.0, 77.9583, 211.3375, 79.2, 69.55, 120.0, 93.5, 80.0, 83.1583, 69.55, 89.1042, 164.8667, 69.55, 83.1583]

Veamos para el campo: "Age"

```
## Encontramos los valores outliers del dataset para la variable "Age"

campo = 'Fare'
aux = titanicTraindata[campo]
aux2 = (limites_outliers (aux))
aux2 = titanicTraindata.loc[ (titanicTraindata[campo] < aux2[0]) | (titanicTraindata[campo] > aux2[1]) ]

print (list(aux2[campo]))
```

[2, 14, 4, 58, 14, 2, 8, 66, 14, 3, 4, 7, 65, 5, 11, 4, 0, 59, 71, 14, 70, 2, 12, 12, 9, 1, 9, 61, 4, 1, 9, 1, 4, 3, 58, 2, 59, 5, 8, 62, 3, 58, 63, 7, 65, 2, 0, 61, 2, 3, 60, 3, 1, 1, 3, 10, 14, 64, 4, 13, 5, 65, 0, 2, 9, 63, 58, 9,

71, 2, 7, 9, 11, 64, 8, 62, 62, 60, 4, 61, 80, 9, 2, 0, 58, 70, 14, 60, 14, 4, 60, 6, 11, 70, 4, 6, 0, 5, 13, 8, 1, 11, 0, 6, 10, 2, 1, 62, 0, 4, 74, 9, 4]

Veamos para el campo: "Parch"

```
## Encontramos los valores outliers del dataset para la variable "Parch"

campo = 'Parch'
aux = titanicTraindata[campo]
aux2 = (limites_outliers (aux))
aux2 = titanicTraindata.loc[ (titanicTraindata[campo] < aux2[0]) | (titanicTraindata[campo] > aux2[1]) ]

print (list(aux2[campo]))
```

[1, 2, 1, 5, 1, 1, 5, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 3, 2, 2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 1, 2, 2, 2, 1, 2, 1, 1, 2, 1, 4, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 4, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 1, 2, 2, 3, 4, 1, 2, 1, 1, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 4, 1, 1, 2, 1, 2, 1, 1, 2, 5, 2, 1, 1, 1, 2, 1, 5, 2, 1, 1, 1, 2, 1, 6, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 2, 1, 1, 1, 1, 2, 1, 2, 3, 1, 2, 1, 2, 2, 1, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 3, 2, 1, 1, 1, 1, 5, 2]

4 ANÁLISIS DE DATOS

4.1 SELECCIÓN DE LOS GRUPOS PARA ANÁLISIS

En un principio, se han seleccionado los siguientes grupos del conjunto de datos que serán considerados para realizar diversos análisis sobre los mismos.

```
# Agrupación por clase de boleto

# Grupo de pasajeros de primera clase
primeraClase <- subset(titanicTrainData, Pclass == 1)
# Grupo de pasajeros de segunda clase
segundaClase <- subset(titanicTrainData, Pclass == 2)
# Grupo de pasajeros de tercera clase
terceraClase <- subset(titanicTrainData, Pclass == 3)
```

```
# Agrupación por sexo

# Grupo de pasajeros de sexo masculino
sexoMasculino <- subset(titanicTrainData, Sex == 'male')
# Grupo de pasajeros de sexo femenino
sexoFemenino <- subset(titanicTrainData, Sex == 'female')
```

```
# Agrupación por puerto de embarque

# Grupo de pasajeros que embarcaron en Cherbourg
puertoCherbourg <- subset(titanicTrainData, Embarked == 'C')
# Grupo de pasajeros que embarcaron en Queenstown
puertoQueenstown <- subset(titanicTrainData, Embarked == 'Q')
# Grupo de pasajeros que embarcaron en Southampton
puertoSouthampton <- subset(titanicTrainData, Embarked == 'S')
```

4.2 COMPROBACIÓN DE LA NORMALIDAD Y HOMOGENEIDAD DE LA VARIANZA

4.2.1 TEST DE NORMALIDAD

- **Solución en R:**

Para verificar la suposición de normalidad, en este caso aplicaremos el test de Shapiro-Wilk. La hipótesis nula indica que la variable sigue una distribución normal donde nuestro nivel de significancia será igual 0.05. Nuestras conclusiones serán obtenidas a partir de la comparación del p-valor con el nivel de significancia establecido

```
# Test de normalidad mediante el test de Shapiro-Wilk
shapiro.test(titanicTrainData$Age)
shapiro.test(titanicTrainData$SibSp)
shapiro.test(titanicTrainData$Parch)
shapiro.test(titanicTrainData$Fare)
```

Age:

Shapiro-Wilk normality test

data: titanicTrainData$Age

W = 0.98014, p-value = 1.181e-09 → No sigue una distribución normal

SibSp:

Shapiro-Wilk normality test

data: titanicTrainData$SibSp

W = 0.51297, p-value < 2.2e-16 → No sigue una distribución normal

Parch:

Shapiro-Wilk normality test

data: titanicTrainData$Parch

W = 0.53281, p-value < 2.2e-16 → No sigue una distribución normal

Fare:

Shapiro-Wilk normality test

data: titanicTrainData$Fare

W = 0.52189, **p-value < 2.2e-16** → No sigue una distribución normal

- **Solución en Python:**

```
' Test de normalidad mediante el test de Shapiro-Wilk'


from scipy.stats import shapiro


# Test de normalidad de la variable Pclass Resultado de (Prob,  "p" o singificancia)

x1 = shapiro(titanicTraindata.Pclass)

print (x1)

# Test de normalidad de la variable Age Resultado de (Prob,  "p" o singificancia)

x2 = shapiro(titanicTraindata.Age)
print (x2)

# Test de normalidad de la variable SibSp Resultado de (Prob,  "p" o singificancia)

x3 = shapiro(titanicTraindata.SibSp)
print (x3)

# Test de normalidad de la variable Parch Resultado de (Prob,  "p" o singificancia)

x4 = shapiro(titanicTraindata.Parch)
print (x4)

# Test de normalidad de la variable Fare Resultado de (Prob,  "p" o singificancia)

x5 = shapiro(titanicTraindata.Fare)
print (x5)
```

Los resultados de "prob" y "p" obtenidos son:

- *(0.718337893486023, 3.3958319924210316e-36)*
- *(0.98072749376297, 1.8490181608044054e-09)*
- *(0.5129655003547668, 5.74532370373175e-44)*
- *(0.5328145027160645, 2.382207389352189e-43)*
- *(0.5218914747238159, 1.0789998175301091e-43)*

Con lo cual la hipótesis nula queda desvirtuada en todos los casos: No son series normales.

4.2.2 HOMOGENEIDAD DE LA VARIANZA

- **Solución en R:**

A fin de comprobar la homocedasticidad o igualdad de varianzas entre los grupos que se van a comparar vamos a aplicar el test de Fligner-Killeen ya que nuestro conjunto de datos no cumple con la condición de normalidad. La hipótesis nula asume igualdad de varianzas en los diferentes grupos donde mantenemos el nivel de significancia de 0.05.

Se procede a analizar la igualdad de varianzas para los grupos conformados por mujeres y hombres, al igual que aquellos grupos de personas que embarcaron en cada uno de los puertos.

```
# Test de igualdad de varianzas mediante el test de Fligner-Killeen
# Supervivencia por sexo
fligner.test(Survived ~ Sex, data=titanicTrainData)
```

```
##  Fligner-Killeen test of homogeneity of variances
## data:  Survived by Sex
## Fligner-Killeen:med chi-squared = 5.7729, df = 1, p-value = 0.01627
```

```
# Supervivencia por puerto de embarque
fligner.test(Survived ~ Embarked, data=titanicTrainData)
```

```
##  Fligner-Killeen test of homogeneity of variances
## data:  Survived by Embarked
## Fligner-Killeen:med chi-squared = 6.8863, df = 2, p-value = 0.03196
```

```
# Convertimos la variable "Pclass" a tipo factor
titanicTrainData$Pclass <- as.factor(titanicTrainData$Pclass)
# Supervivencia por clase de boleto
fligner.test(Survived ~ Pclass, data=titanicTrainData)
```

```
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Survived by Pclass
## Fligner-Killeen:med chi-squared = 35.766, df = 2, p-value = 1.712e-08
```

Puesto que hemos obtenido un p-valor menor al nivel de significancia (0.05) en los tres casos, entonces se rechaza la hipótesis nula y se concluye que la variable “Survived” presenta varianzas estadísticamente diferentes para los diferentes grupos de “Sex”, “Embarked” y “Pclass”.

- **Solución en Python:**

La funcionalidad de SCIpy sólo permite valores numéricos, por ello tenemos que convertir algún campo.

- Para Sex:

```
from scipy.stats import fligner as FK

# En el meteodo encontrado en la libreria hemos podido comprobar que no se admiten campos no numéricos.
# Por ello los campos tipo STR  en 'Sex' , los convertimos en numéricos.

Sex_int = titanicTraindata.Sex

Sex_int = Sex_int.replace('female',0)
Sex_int = Sex_int.replace('male',1)

x1 = FK(Sex_int, titanicTraindata.Survived )

print (x1)
```

FlignerResult(statistic=1.8903305, p-value= 0.1691646)

No queda invalidada la hipótesis nula.

- Para Embarked

```
# En el meteodo encontrado en la libreria hemos podido comprobar que no se admiten campos no numéricos.
# Por ello los campos tipo STR  en 'Embarked' , los convertimos en numéricos.

Embarked_int = titanicTraindata.Embarked

Embarked_int = Embarked_int.replace('C',0)
Embarked_int = Embarked_int.replace('Q',1)
Embarked_int = Embarked_int.replace('S',2)

x1 = FK(Embarked_int, titanicTraindata.Survived )

print (x1)
```

FlignerResult(statistic= 4.124670954313547, p-value = 0.042262)

Queda invalidada la hipótesis NULA

- Para PClass

```
Pclass_int = titanicTraindata.Pclass

x1 = FK(Pclass_int, titanicTraindata.Survived )

print (x1)
```

FlignerResult(statistic = 89.50879252308928, p - value = 3.05278689 e-21)

Queda invalidada la hipótesis nula.

5.3 PRUEBAS ESTADÍSTICAS

5.3.1 ANÁLISIS DE CORRELACIÓN

- **Solución en R:**

En primer lugar, vamos a realizar un **análisis de correlación con las variables cuantitativas** para determinar aquellas que tienen una mayor influencia sobre la supervivencia de los pasajeros. Debido a que las variables no siguen una distribución normal aplicaremos el coeficiente de correlación de Spearman.

Hipótesis nula (H0): No existe relación entre las variables
Regla de decisión: Si p-valor < 0.05 se rechaza H0

```
# Test de correlación de Spearman de la variable "Age"
suppressWarnings(suppressMessages(spearman.test(titanicTrainData$Age,titanicTrainData$Survived)))
```

data: titanicTrainData$Age and titanicTrainData$Survived
S = 126740000, **p-value = 0.02513**
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
-0.07503034

```
# Test de correlación de Spearman de la variable "SibSp"
suppressWarnings(suppressMessages(spearman.test(titanicTrainData$SibSp,titanicTrainData$Survived)))
```

data: titanicTrainData$SibSp and titanicTrainData$Survived
S = 107410000, **p-value = 0.007959**
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
0.08887948

```
# Test de correlación de Spearman de la variable "Parch"
suppressWarnings(suppressMessages(spearman.test(titanicTrainData$Parch,titanicTrainData$Survived)))
```

data: titanicTrainData$Parch and titanicTrainData$Survived
S = 101590000, p-value = **3.513e-05**
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
0.1382656

```
# Test de correlación de Spearman de la variable "Fare"
suppressWarnings(suppressMessages(spearman.test(titanicTrainData$Fare,titanicTrainData$Survived)))
```

data: titanicTrainData$Fare and titanicTrainData$Survived
S = 79726000, **p-value < 2.2e-16**
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
0.3237361

Con los resultados obtenidos (p-valor < 0.05) se concluye que las variables cuantitativas “Age”, “SibSp”, “Parch” y “Fare” si presentan una correlación estadísticamente significativa con la variable supervivencia. La variable “Fare” es la que presenta la mayor correlación de todas.

También efectuaremos un **análisis de dependencia con las variables categóricas** mediante el test Chi Cuadrado.

Hipótesis nula (H0): Las variables son independientes.
Regla de decisión: Si p-valor < 0.05 se rechaza H0

```
# Test Chi Cuadrado con la variable "Sex"
auxTableSex <- table(titanicTrainData$Sex, titanicTrainData$Survived) # Tabla de contingencia
chisq.test(auxTableSex, correct=TRUE) #"Yates' Continuity Correction or Fisher's Exact"
```

Pearson's Chi-squared test with Yates' continuity correction
data: auxTableSex
X-squared = 260.72, df = 1, **p-value < 2.2e-16**

```
# Test Chi Cuadrado con la variable “Embarked”
auxTableEmbarked <- table(titanicTrainData$Embarked , titanicTrainData$Survived) # Tabla de contingencia
chisq.test(auxTableEmbarked, correct=TRUE) #"Yates' Continuity Correction or Fisher's Exact"
```

Pearson's Chi-squared test
data: auxTableEmbarked
X-squared = 25.964, df = 2, **p-value = 2.301e-06**

```
# Test Chi Cuadrado con la variable “Pclass”
auxTablePclass <- table(titanicTrainData$Pclass, titanicTrainData$Survived) # Tabla de contingencia
chisq.test(auxTablePclass, correct=TRUE) #"Yates’ Continuity Correction or Fisher’s Exact"
```

Pearson's Chi-squared test
data: auxTablePclass
X-squared = 102.89, df = 2, **p-value < 2.2e-16**

Con los resultados obtenidos (p-valor < 0.05) se concluye que las variables categóricas “Embarked”, “Pclass” y “Sex” si presentan una relación con la supervivencia de pasajeros.

- **Solución en Python:**

Respecto a las relaciones de dependencia o independencia de las variables NO NORMALES.

- Relación y correlación, para la variable: Age frente a Survived

```
# Aplicamos la correlación de Spearman.
# Hipótesis nula: No existe correlación significativa entre ambas variables.

from scipy.stats import spearmanr as SPR

x1 = SPR(titanicTraindata.Age , titanicTraindata.Survived )

print (x1)
```

SpearmanrResult (correlation= -0.03319443011904244, pvalue = 0.3223095243860028)

Vemos que NO existe una correlación estadísticamente significativa entre ambas variables.

- Relación y correlación, para la variable: SibSp frente a Survived

```
#
x1 = SPR(titanicTraindata.SibSp , titanicTraindata.Survived )

print (x1)
```

SpearmanrResult (correlation= 0.08887948468090501, pvalue= 0.007941431285733533)

Vemos que SÍ existe una correlación estadísticamente significativa entre ambas variables de un 8,9% en positivo.

- Relación y correlación, para la variable: Fare frente a Survived

```
x1 = SPR(titanicTraindata.Fare , titanicTraindata.Survived )

print (x1)
```

SpearmanrResult(correlation= 0.32373613944480834, pvalue= 3.471227970207005e-23)

Vemos que SÍ existe una clara y potente correlación estadísticamente significativa entre ambas variables de un 32,3% en positivo. ¡ Los ricos siempre ganan !

- Relación y correlación, para la variable: Parch frente a Survived

```
x1 = SPR(titanicTraindata.Parch , titanicTraindata.Survived )
print (x1)
```

SpearmanrResult (correlation= 0.13826563286545587, pvalue= 3.453591460380432e-05)

Vemos que SÍ existe una correlación estadísticamente significativa entre ambas variables de un 13,8% en positivo.

Respecto a la variable Sex realizamos una prueba de ChiSq

```
from scipy.stats import chi2_contingency as chi2


# Aplicamos un test de ChiSq.
# Hipótesis nula: No existe correlación significativa entre ambas variables Sex / Survived
# Para ello empleamos la lista numérica calculada anteriormente Sex_int
# Obtenemos la tabla de frecuencias

contingency_table = pd.crosstab( titanicTraindata.Survived, titanicTraindata.Sex )

print ( contingency_table )

x1 = chi2( contingency_table )

print ("--------------------")
print ("El valor del Estadístico de comprobación es")
print ( x1[0])
print ("--------------------")
print ("La 'p' calculada es:")
print ( x1[1])
```

 # Matriz de contingencia:

Sex female male

Survived

0 81 468

1 233 109

El valor del Estadístico de comprobación es

260.71702016732104

La 'p' calculada es:

1.1973570627755645e-58

Con lo cual queda anulada la hipótesis nula del test CHI2. SÏ existe una relación entre sexo y supervivencia.

Sí existe correlación entre Sex y la salvación.

5.3.2 CONTRASTE DE HIPÓTESIS

¿Las personas que sobrevivieron pagaron un precio superior por su boleto que aquellas que no sobrevivieron?

A continuación se va a realizar un contraste de hipótesis sobre dos muestras para determinar si la tarifa del boleto es superior dependiendo si la persona sobrevivió o no. Para este objetivo, dividiremos el dataset en dos muestras, una para los sobrevivientes y otra para los que no sobrevivieron junto a la tarifa del boleto.

Adicionalmente, aplicaremos la prueba de suma de rangos de Wilcoxon que se trata de una prueba no paramétrica que no requiere condiciones particulares sobre la distribución de los datos.

- **Solución en R:**

Hipótesis nula (H0): La tarifa promedio del boleto es igual para los sobrevivientes y no sobrevivientes.
Hipótesis alternativa (H1): La tarifa promedio del boleto de los sobrevivientes es superior a la tarifa promedio boleto de los no sobrevivientes (**UNILATERAL**)

Regla de decisión: Si p-valor < 0.05 se rechaza H0

```
diedGroup <- subset(titanicTrainData, Survived == 1) # Grupo de sobrevivientes
survivedGroup <- subset(titanicTrainData, Survived == 0) # Grupo de fallecidos
# Test de Wilcoxon para la variable "Fare" entre sobrevivientes y fallecidos
wilcox.test(x = diedGroup$Fare, y = survivedGroup$Fare, alternative = "greater",
            paired = FALSE, conf.int = 0.95)
```

```
##  Wilcoxon rank sum test with continuity correction
## data:  diedGroup$Fare and survivedGroup$Fare
## W = 129950, p-value < 2.2e-16
## alternative hypothesis: true location shift is greater than 0
## 95 percent confidence interval:
##  7.399983      Inf
## sample estimates:
## difference in location
##               9.499951
```

El p-valor es menor que el nivel de significación, lo que indica que el valor del estadístico de contraste que hemos observado tiene una alta probabilidad de salir bajo la hipótesis alternativa. Por lo tanto, se rechaza la hipótesis nula y sabemos que la tarifa promedio del boleto pagado por los sobrevivientes es superior a la pagada por aquellos que no sobrevivieron.

- **Solución en Python:**

La solución aplicada en R es: la de Mann-Whitey / Wilcoxon test “rank-sum” (MWW) que en R se define con la función:

wilcox.test (with *paired=FALSE*)

Traducido al entorno Pyhon Scipy, tiene una redacción que puede llevar a errores puesto que puede confundirse con el Wilcoxon signed-rank test.

La diferencia viene de las asunciones:

- En el MWW se está interesado en la diferencia entre dos poblaciones independientes ((null hypothesis: la misma, alternative: hay una diferencia) ,
- mientras que el el Wilcoxon signed-rank test, el interés es el mismo pero sobre dos muestra relacionadas (paired)

Si aplicamos :

- **Si empleamos: scipy.stats.wilcoxon**

  Estamos calculando el el Wilcoxon signed-rank test o sea PAIRED = YES

- si empleamos: **scipy.stats.ranksums** se asume el valor PAIRED = TRUE. Pero el problema es que no nos da opciones de bilateralidad o unilateralidad y parte de la hipótesis de que las muestras son normales: "... The test statistic under the large-sample approximation that the rank sum statistic is normally distributed…"

Por ello al final nos decantamos por la función **scipy.stats.mannwhitneyu:**

```
ffrom scipy.stats import mannwhitneyu as mann

# Seleccionamos el grupo de pasajeros que no sobrevivieron

titanicTraindata_died = titanicTraindata.loc[titanicTraindata['Survived'] == 0]
fare_died = titanicTraindata_died.Fare

titanicTraindata_surv = titanicTraindata.loc[titanicTraindata['Survived'] == 1]
fare_surv = titanicTraindata_surv.Fare
```

U = 57806.5 p = 2.2767385896251186e-22

Podemos anular la hipótesis base: con lo que Y > X pero la p exacta no la podemos determinar.

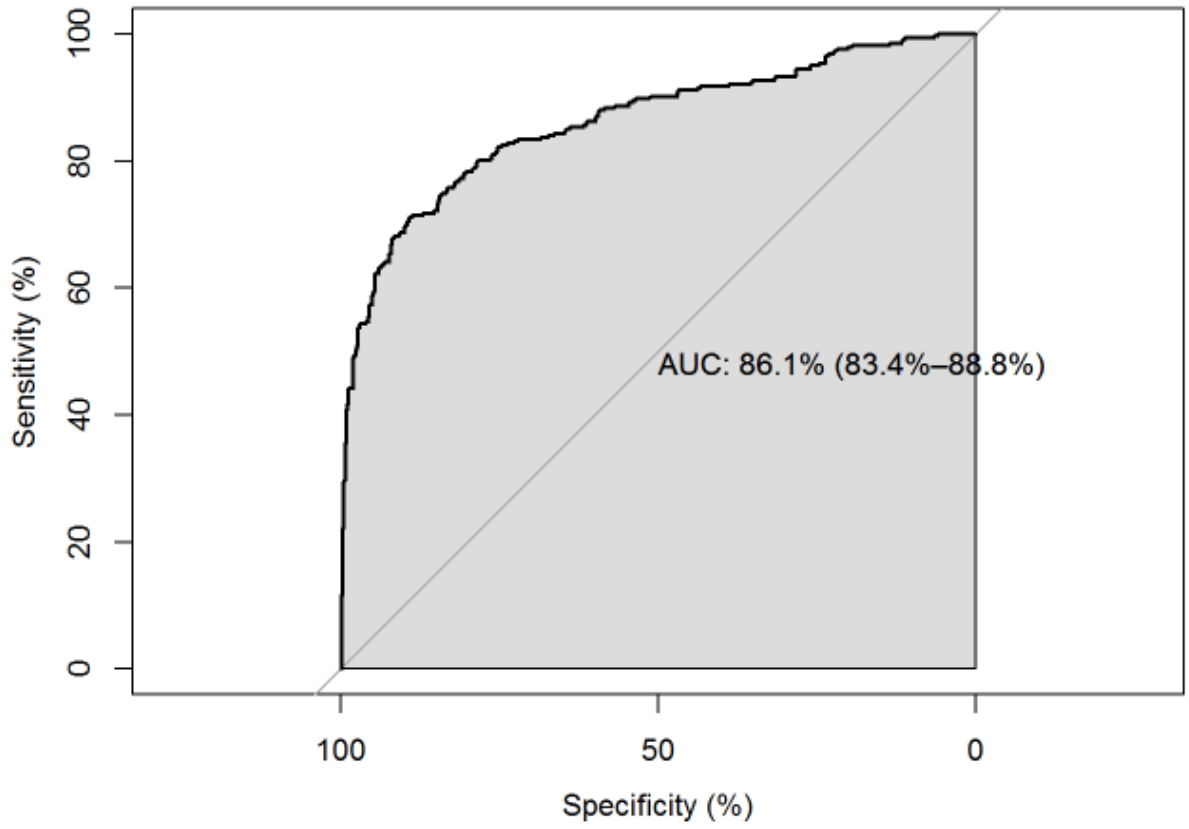
5.3.3 MODELO DE REGRESIÓN LOGÍSTICA

- **Solución en R:**

Finalmente, para dar solución al objetivo inicial de conocer a los pasajeros que sobrevivieron al naufragio se propone un modelo de regresión logística a fin de predecir la supervivencia en base a los predictores que tienen más influencia en esta variable. Después de varias combinaciones hemos seleccionado al modelo que ha mostrado los indicadores más altos de bondad de ajuste, calidad y efectividad.

```
# Modelo de regresión logística con el mejor performance
logitModel <- glm(titanicTrainData$Survived ~ titanicTrainData$Fare + titanicTrainData$Sex + titanicTrainData$Pclass + titanicTrainData$Age + titanicTrainData$SibSp + titanicTrainData$Embarked, data = titanicTrainData, family = "binomial")
```

```
# Curva ROC
test_prob1 = predict(logitModel, titanicTrainData, type = "response")
test_roc1 = roc(titanicTrainData$Survived ~ test_prob1, plot = TRUE, print.auc = TRUE, smoothed = TRUE,
show.thres=TRUE, auc.polygon=TRUE, ci=TRUE, percent=TRUE)
as.numeric(test_roc1$auc)/100 # Área bajo la curva
```



Área bajo la curva: 0.86

La curva ROC (Receiver Operating Characteristic) que hemos graficado es una representación gráfica que ilustra la relación entre la sensibilidad y la especificidad de un sistema clasificador para diferentes puntos de corte y mediante el cual se tiene una idea clara de la efectividad del modelo propuesto.

Con este preámbulo, nuestro modelo de regresión logística presenta un área bajo la curva ROC igual a 0,86. Por lo tanto, podemos interpretar que este modelo tiene un rendimiento bueno ya que está dentro del rango (0,75 - 0,9), y también significa que existe una una probabilidad de 86,10% de que la predicción realizada sobre la supervivencia sea correcta.

- **Solución en Python:**

En Python vamos a tener que convertir las variables a categóricas con varios niveles. Los métodos de las librerías no lo realizan por si mismas.

```
import statsmodels.api as sm

y = titanicTraindata[['Survived']]
X = titanicTraindata[['Fare','Sex','Pclass','Age','SibSp','Embarked']]

# Tenemos que tratar la variables categoricas y las convertimos en numericas.

X = X.replace('female',0)
X = X.replace('male',1)

X = X.replace('C',0)
X = X.replace('Q',1)
X = X.replace('S',2)


logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

En base a ello realizamos la regresión logística con la librería Stats Models que nos da un resumen similar al de las aplicaciones en R

Optimization terminated successfully.

Results: Logit

Model:	Logit	Pseudo R-squared:	0.255
Dependent Variable:	Survived	AIC:	895.8653
		BIC:	924.6193
No. Observations:	891	Log-Likelihood:	-441.93
Df Model:	5	LL-Null:	-593.33
Df Residuals:	885	LLR p-value:	2.5168e-63
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Fare	0.0172	0.0028	6.0364	0.0000	0.0116	0.0227
Sex	-2.2518	0.1751	-12.8604	0.0000	-2.5950	-1.9086
Pclass	0.0253	0.0756	0.3355	0.7373	-0.1227	0.1734
Age	0.0067	0.0055	1.2138	0.2248	-0.0041	0.0176
SibSp	-0.3121	0.0922	-3.3858	0.0007	-0.4927	-0.1314
Embarked	0.1030	0.1055	0.9760	0.3290	-0.1038	0.3097

Ahora realizamos el mismo modelo con la librería “Sklearn”.

```
ifrom sklearn.linear_model import LogisticRegression as LGR


# Realizamos el otro métido por Sklearn.
logit = LGR()
result=logit.fit(X,y)
```

Evaluamos los resultados

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Realizamos la predicción de probabilidades.
y_pred = result.predict(X)


confusion_matrix = confusion_matrix(y, y_pred)
print((confusion_matrix))


print(classification_report(y, y_pred))
```

Con ello obtenemos una matriz de confusión de:

[[470 79]
[99 243]]

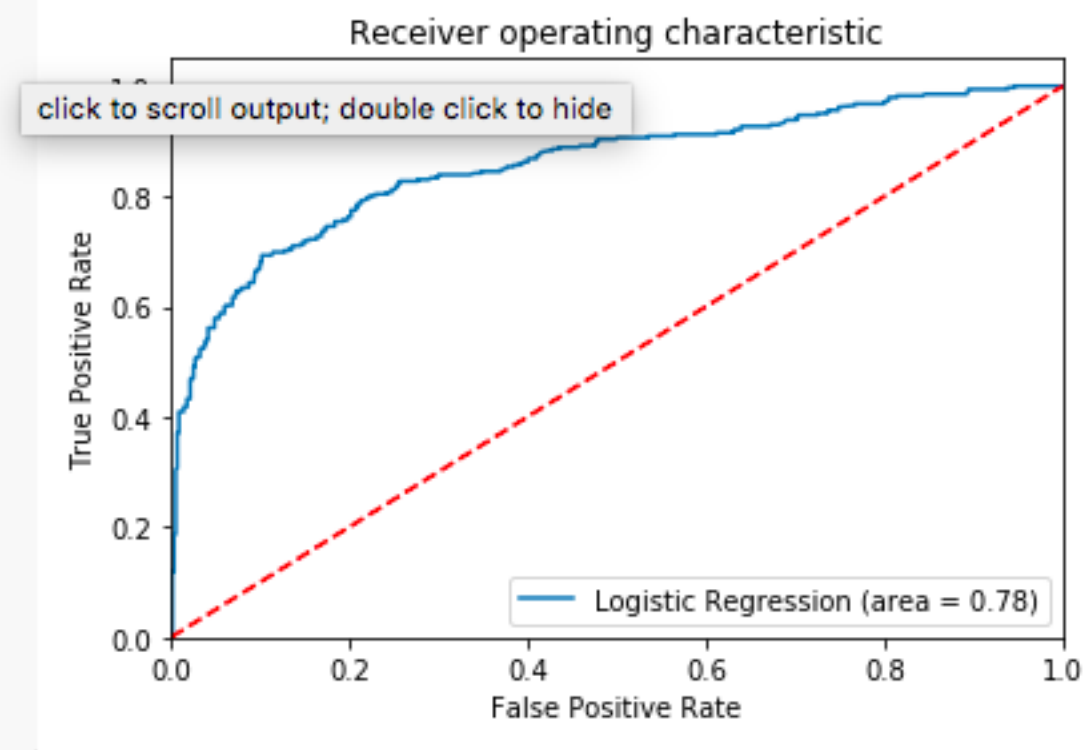
Y un informe de resultados que nos dá una “accuracy del 80%”

	precision	recall	f1-score	support
0	0.83	0.86	0.84	549
1	0.75	0.71	0.73	342
accuracy			0.80	891
macro avg	0.79	0.78	0.79	891
weighted avg	0.80	0.80	0.80	891

Sólo nos falta indicar las curvas de ROC y valor de AUC

```
fimport matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

logit_roc_auc = roc_auc_score(y, result.predict(X))
fpr, tpr, thresholds = roc_curve(y, result.predict_proba(X)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

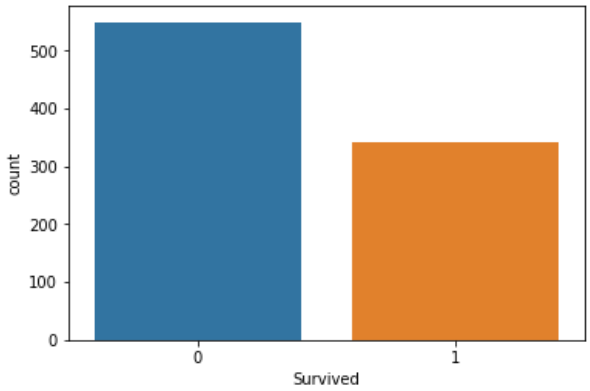


El nivel de área bajo la curva es ligeramente inferior siendo un modelo bueno. El motivo puede ser que sea necesario mejorar el tratamiento de las variables categóricas , creando más campos Dummy con valor 0,1 en vez de enteros multinivel.

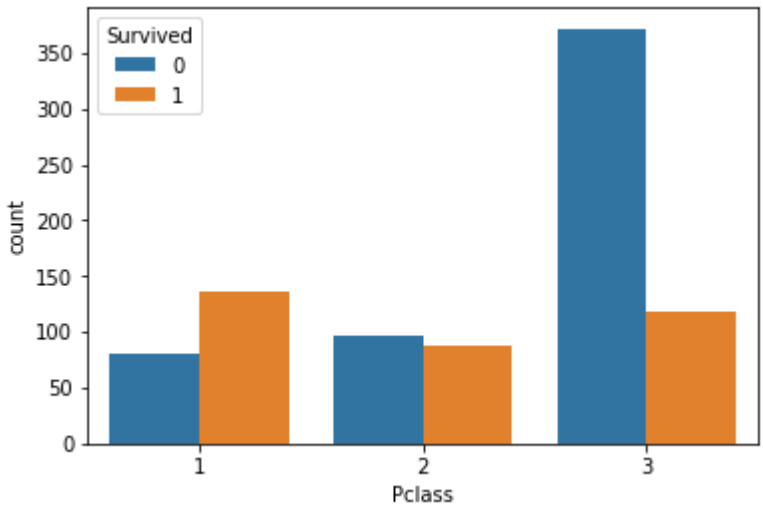
6 REPRESENTACIÓN GRÁFICA

6.1 DISTRIBUCIÓN DE VARIABLES

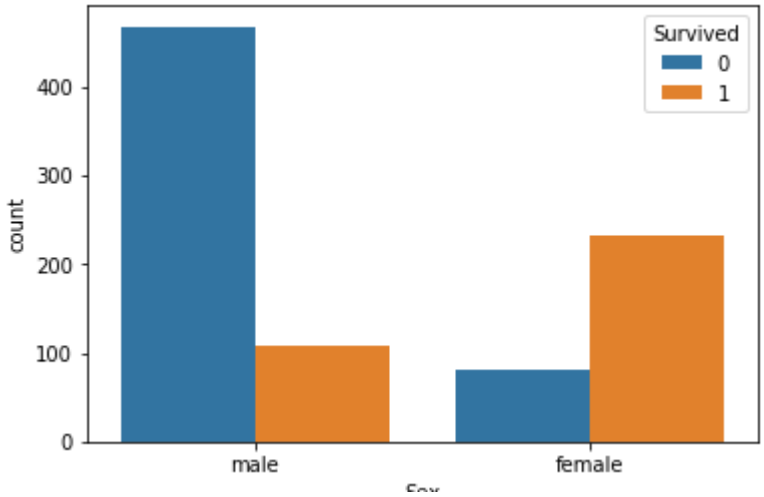
```
# Variable a predecir “Survived”
sns.countplot(titanicTrainData['Survived'])
```



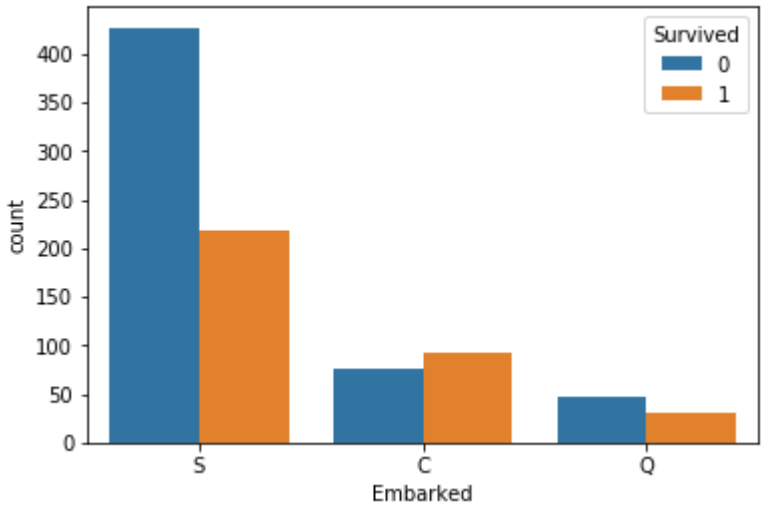

```
sns.countplot(titanicTrainData['Pclass'], hue=titanicTrainData['Survived'])
```



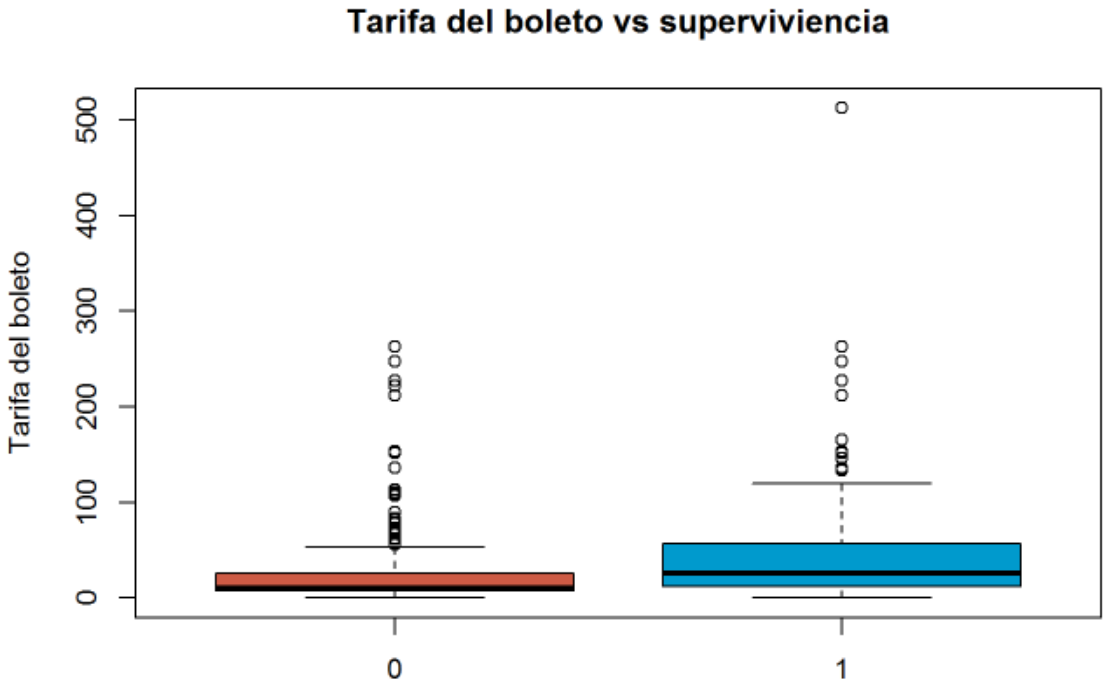

```
sns.countplot(titanicTrainData['Sex'], hue=titanicTrainData['Survived'])
```



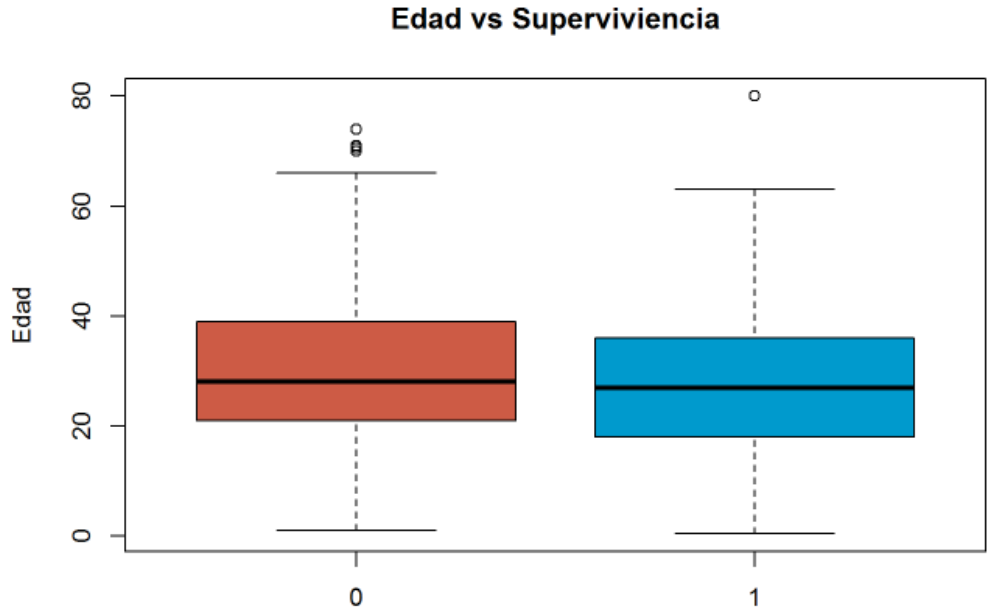

```
sns.countplot(titanicTrainData['Embarked'], hue=titanicTrainData['Survived'])
```



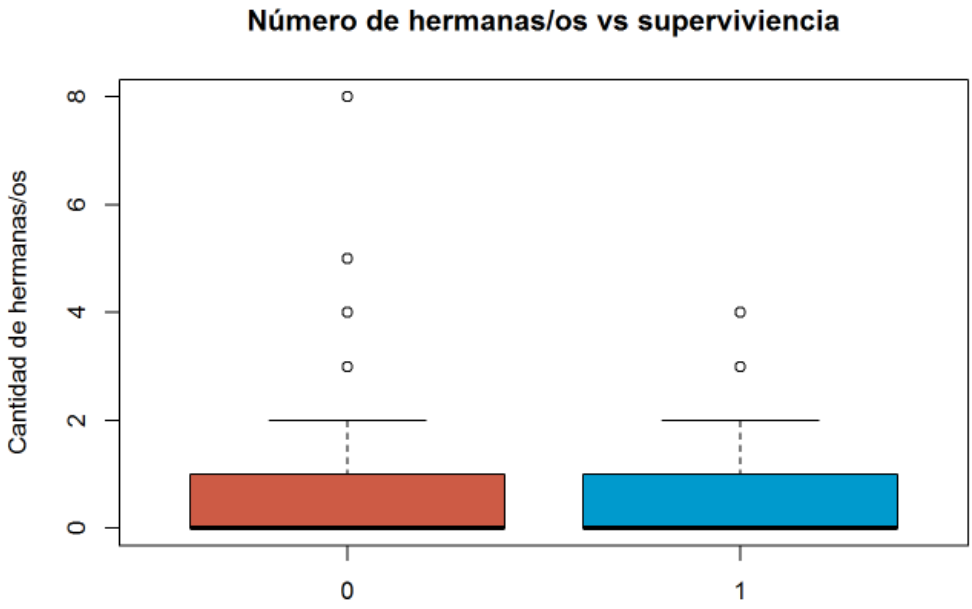

```
boxplot(titanicTrainData$Fare~Survived,data=titanicTrainData, main="Tarifa del boleto vs superviviencia", ylab="Tarifa del boleto", col=(c("coral3","deepskyblue3", "lightgoldenrod2")))
```



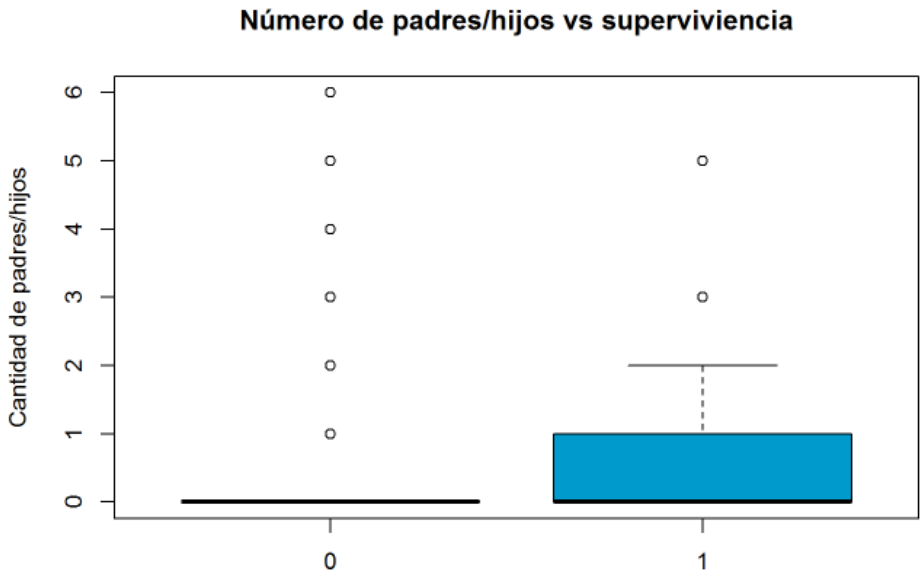

```
boxplot(titanicTrainData$Age~Survived,data=titanicTrainData, main="Edad vs Superviviencia",
ylab="Edad", col=(c("coral3","deepskyblue3", "lightgoldenrod2")))
```



```
boxplot(titanicTrainData$SibSp~Survived,data=titanicTrainData, main="Número de hermanas/os vs superviviencia", ylab="Cantidad de hermanas/os", col=(c("coral3","deepskyblue3", "lightgoldenrod2")))
```




```
boxplot(titanicTrainData$Parch~Survived,data=titanicTrainData, main="Número de padres/hijos vs superviviencia", ylab="Cantidad de padres/hijos", col=(c("coral3","deepskyblue3", "lightgoldenrod2")))
```



7 RESOLUCIÓN Y CONCLUSIONES

1. En esta práctica se han implementado diferentes técnicas de limpieza y tratamiento de datos como imputación de valores perdidos para conformar un dataset que pueda ser analizado y el mismo permita dar solución a un problema planteado como en nuestro caso es el de conocer las variables que más influyeron en la supervivencia de los pasajeros del Titanic y ser capaces este evento.

2. El tratamiento de valores extremos debe ser llevado con cuidado ya que el hecho de que un valor se encuentre muy alejado de la distribución normal de la variable no significa que se trate de un dato ilegítimo. Tal y como ha sucedido en el dataset de Titanic, se analizaron los valores extremos y se comprobó que estos podrían ser reales por tal motivo no se aplicó ninguna acción sobre los mismos.

3. El contraste de hipótesis y el análisis de correlación nos permitió conocer la medida de asociación que tienen las variables predictoras respecto a la supervivencia y de lo cual se puede concluir que el hecho de ser mujer y haber adquirido un boleto de primera clase aumentaba la probabilidad de sobrevivir.

4. La comprobación de normalidad en el conjunto de datos es importante ya que de ello depende seleccionar las mejores técnicas para análisis posteriores los cuales pueden ser test paramétricos como el t.test cuando sabemos que los datos provienen de una distribución normal o test no paramétricos cuando no cumplen esta condición y se puede hacer uso del test de Wilcoxon.

5. En Python: algunos de los campos de KNN de estimación de valores perdidos en “Age” no corresponden exactamente con los calculados en R.

6. En Python se puede añadir que de nota que es un lenguaje mucho más generalista y que algunas de las funciones de estadística avanzada no están tan bien solventadas como en R en comparación, inclusos con librerías especializadas. Esto ha llevado a situaciones de nomenclatura confusa como en el test de WILCOX para muestras aisladas y NO normales.

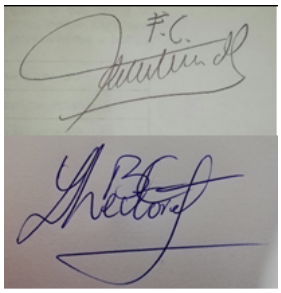
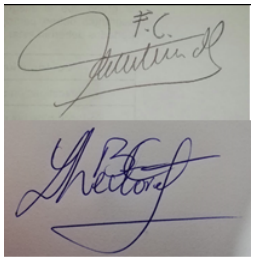
7. El tratamiento de variables categóricas en los métodos de regresión logística EN PYTHON no es tan desarrollado como en R. Se tienen que generar columnas tipo “DUMMY” con ceros y unos para su tratamiento. Esta función está integrada en el caso de de R y creemos que es el causante de la diferencia de rendimiento AUC entre ambos lenguajes.

8 REFERENCIAS

- https://stats.stackexchange.com/questions/113936/what-is-the-difference-between-the-mann-whitney-and-wilcoxon-rank-sumtest
- https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.ranksums.html
  https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ranksums.html#:~:text=Compute%20the%20Wilcoxon%20rank%2Dsum,values%20in%20the%20other%20sample.
- https://www.theanalysisfactor.com/linear-models-r-plotting-regression-lines/

- https://economipedia.com/definiciones/r-cuadrado-coeficiente-determinacion.html
- https://blog.minitab.com/es/analisis-de-regresion-como-puedo-interpretar-el-r-cuadrado-y-evaluar-la-bondad-de-ajuste
- https://www.statmethods.net/stats/regression.html
- http://rstudio-pubs-static.s3.amazonaws.com/379699_e23d0c34255841e08d2eb4c98ff899fd.html
- https://rpubs.com/Joaquin_AR/226291
- http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0378-78182006000200007
- http://www.dm.uba.ar/materias/analisis_de_datos/2008/1/teoricas/Teor5.pdf
- https://stats.idre.ucla.edu/r/dae/logit-regression/
- https://rpubs.com/Joaquin_AR/229736
- https://thestatsgeek.com/2014/02/16/the-hosmer-lemeshow-goodness-of-fit-test-for-logistic-regression/

9 TABLA DE CONTRIBUCIONES

CONTRIBUCIONES	INTEGRANTE	FIRMA
INVESTIGACIÓN PREVIA	FERNANDO CACHADIÑA HÉCTOR BASTIDAS	
PLANTEAMIENTO Y ELABORACIÓN DEL DOCUMENTO	FERNANDO CACHADIÑA HÉCTOR BASTIDAS	
DESARROLLO DEL CÓDIGO	FERNANDO CACHADIÑA HÉCTOR BASTIDAS