Tipología y Ciclo de Vida de Datos - Práctica 2

aruizplaza & rcotillas

10/12/2020

Contents

.- Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

La importancia de este dataset se puede explicar desde un punto de vista histórico (periodístico, documental, etc.), así como desde un punto de vista socioeconómico. En cuanto al primero, el suceso al que está vinculado

este conjunto de datos, fue uno de los accidentes más impactantes del siglo pasado y el estudio de los datos puede ayudar a entender un poco mejor que ocurrió, al menos en lo referente a lo que la supervivencia se refiere. En cuanto al segundo, el estudio de este dataset nos puede ayudar a encontrar patrones que pueden repetirse en situaciones similares, así como estudiar si la pertencencia a cierta clase o género otorga o no ventajas ante este tipo de situaciones.

En esta práctica trataremos de responder a las siguientes preguntas:

(1) ¿Se puede predecir la supervivencia?

(2) ¿Que tratamiento de nulos en la variable edad se comporta mejor en la prediccion de la supervivencia?

(3) ¿Que variables influyen mas en la supervivencia?

(4) ¿la supervivencia es mayor en caso de ser niño o mujer?

.- Integración y selección de los datos de interés a analizar.

Instalación y Llamada de los paquetes a utilizar durante la práctica

```
# Cargamos paquete TIDYVERSE que engloba a otros paquetes que nos permiten llevar a cabo la importación
if(!require(tidyverse)){
    install.packages('tidyverse', repos='http://cran.us.r-project.org')
}
library(tidyverse)

# Cargamos paquete PSYCH que agrupa funciones  utilizadas en psicología.
if(!require(psych)){
    install.packages('psych', repos='http://cran.us.r-project.org')
}
library(psych)

# Cargamos paquete reticulate proporciona un conjunto de  herramientas para integrar código Python y R.
if(!require(reticulate)){
    install.packages('reticulate', repos='http://cran.us.r-project.org')
}
library(reticulate)

# Cargamos paquete CORRPLOT nos permite la representación gráfica de una matriz de correlación, así com
if(!require(corrplot)){
    install.packages('corrplot', repos='http://cran.us.r-project.org')
}
library(corrplot)

# Cargamos paquete DESCTOOLS que proporciona una colección de funciones estadísticas básicas para la des
if(!require(DescTools)){
    install.packages('DescTools', repos='http://cran.us.r-project.org')
}
library(DescTools)

# Cargamos paquete ARULES que proporciona la infraestructura para representar, manipular y analizar los
if(!require(arules)){
    install.packages('arules', repos='http://cran.us.r-project.org')
```

```
}
library(arules)

# Cargamos paquete CLUSTER que nos proporciona métodos de análisis de clusters (Agrupaciones)
if(!require(cluster)){
    install.packages('cluster', repos='http://cran.us.r-project.org')
}
library(cluster)

# Cargamos paquete MCLUST que nos proporciona herraminetas para la agrupación, clasificación y estimaci
if(!require(mclust)){
    install.packages('mclust', repos='http://cran.us.r-project.org')
}
library(mclust)

#Cargamos el paquete GGALLY, que amplía 'GGPLOT2' añadiendo varias funciones para reducir la complejidad
if(!require(GGally)){
    install.packages('GGally', repos='http://cran.us.r-project.org')
}
library(GGally)

#Cargamos el paquete RPART paquete que nos permite el particionado recursivo y el uso de árboles de regr
if(!require(rpart)){
    install.packages('rpart', repos='http://cran.us.r-project.org')
}
library(rpart)

#Cargamos el paquete IPRED que Mejora de los modelos de predicción mediante la clasificación indirecta y
if(!require(ipred)){
    install.packages('ipred', repos='http://cran.us.r-project.org')
}
library(ipred)

#Cargamos el paquete VIM que introduce nuevas herramientas para la visualización de los valores perdidos
if(!require(VIM)){
    install.packages('VIM', repos='http://cran.us.r-project.org')
}
library(VIM)

#Cargamos el paquete NORTEST que agrupa 5 pruebas 'Omnibus' para probar la hipótesis compuesta de la nor
if(!require(nortest)){
    install.packages('nortest', repos='http://cran.us.r-project.org')
}
library(nortest)

#Cargamos el paquete CARET(Classification And REgression Training) es un conjunto de funciones que inter
if(!require(caret)){
    install.packages('caret', repos='http://cran.us.r-project.org')
}
library(caret)

#Cargamos el paquete RANDOMFOREST que nos permite la aplicación de Clasificación y regresión mediante el
if(!require(randomForest)){
```

```
    install.packages('randomForest', repos='http://cran.us.r-project.org')
}
library(randomForest)

#Cargamos el paquete GGRAPH, que es una extensión de la API de ggplot2 adaptada a las visualizaciones d
if(!require(ggraph)){
    install.packages('ggraph', repos='http://cran.us.r-project.org')
}
library(ggraph)

#Cargamos el paquete GGRAPH, que nos proporciona Rutinas para gráficos simples y análisis de redes.
if(!require(igraph)){
    install.packages('igraph', repos='http://cran.us.r-project.org')
}
library(igraph)
```

Como en esta práctica utilizaremos tanto código R como Python, procedemos a instalar los paquetes que utilizaremos con Python:

```
# Instalamos paquete PANDAS para ser usado en los fragmentos de código Python
if(!require(pandas)){
    py_install("pandas")
}

# Instalamos paquete MATPLOTLIB para ser usado en los fragmentos de código Python
if(!require(matplotlib)){
    py_install("matplotlib")
}

# Instalamos paquete SEABORN para ser usado en los fragmentos de código Python
if(!require(seaborn)){
    py_install("seaborn")
}
```

```
# Cargamos paquete PANDAS
import pandas as pd
# Cargamos paquete MATPLOTLIB
import matplotlib.pyplot as plt
# Cargamos paquete SEABORN
import seaborn as sns
```

```
# Carga de los datos de entrenamiento desde el archivo csv a un nuevo dataframe llamado train
train <- read.csv("./titanic/train.csv",header=T, sep=",")

# Visualizamos las 6 primeras filas, y las 6 últimas, del nuevo dataset
head(train)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
```

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

```
tail(train)
```

```
##     PassengerId Survived Pclass                                      Name    Sex
## 886         886        0      3     Rice, Mrs. William (Margaret Norton) female
## 887         887        0      2                     Montvila, Rev. Juozas   male
## 888         888        1      1              Graham, Miss. Margaret Edith female
## 889         889        0      3 Johnston, Miss. Catherine Helen "Carrie" female
## 890         890        1      1                     Behr, Mr. Karl Howell   male
## 891         891        0      3                       Dooley, Mr. Patrick   male
##     Age SibSp Parch     Ticket   Fare Cabin Embarked
## 886  39     0     5     382652 29.125              Q
## 887  27     0     0     211536 13.000              S
## 888  19     0     0     112053 30.000   B42        S
## 889  NA     1     2 W./C. 6607 23.450              S
## 890  26     0     0     111369 30.000  C148        C
## 891  32     0     0     370376  7.750              Q
```

El dataframe que utilizaremos para crear el/los modelo/s que crearemos a lo largo de la práctica, contiene 891 observaciones con 12 variables, todos datos referidos a pasajeros del Titanic. A simple vista, podemos observar que existen valores nulos en la variable Age(Edad) y valores desaparecidos(blancos) en la variable Cabin.

En nuestro caso, no nos hará falta integrar este fichero con ninguna otra fuente de datos, así pues, a continuación realizaremos un análisis descriptivo del dataset.

El conjunto de datos está compuesto por las sifguientes variables:

Variable	**Definición**	**Clave**
PassengerId	Identificador de pasajero	
Survived	Supervivencia	0 = No, 1 = Yes
Pclass	Clase de ticket	1 = 1st, 2 = 2nd, 3 = 3rd
Name	Nombre del/de la Pasajera/o	
Sex	Sexo	
Age	Edad en años	
Sibsp	# of hermanas/os / esposasa bordo del Titanic	
Parch	# of ascendientes / descendientes a bordo del Titanic	
Ticket	Número de ticket	

Variable	Definición	Clave
Fare	Tarifa de Pasaje	
Cabin	Número de camarote	
Embarked	Puerto de Embarque	C = Cherbourg, Q = Queenstown, S = Southampton

```
#Creamos un nuevo set, copia del anterior, que usaremos en adelante. De esta manera, si necesitamos vol
dfTrain <- train
```

Se utilizan las funciones summary y describe, para tener una primera idea de las características de las variables, sus tipos de datos, distribución de sus valores, maximos, minimos, etc.

```
# Se utiliza la función sumary para
summary(dfTrain)
```

```
##   PassengerId       Survived          Pclass          Name
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:891
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   Class :character
##  Median :446.0   Median :0.0000   Median :3.000   Mode  :character
##  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
##
##      Sex                 Age            SibSp           Parch
##  Length:891         Min.   : 0.42   Min.   :0.000   Min.   :0.0000
##  Class :character   1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000
##  Mode  :character   Median :28.00   Median :0.000   Median :0.0000
##                     Mean   :29.70   Mean   :0.523   Mean   :0.3816
##                     3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000
##                     Max.   :80.00   Max.   :8.000   Max.   :6.0000
##                     NA's   :177
##     Ticket               Fare           Cabin             Embarked
##  Length:891         Min.   :  0.00   Length:891         Length:891
##  Class :character   1st Qu.:  7.91   Class :character   Class :character
##  Mode  :character   Median : 14.45   Mode  :character   Mode  :character
##                     Mean   : 32.20
##                     3rd Qu.: 31.00
##                     Max.   :512.33
##
```

Sex, Cabin y Embarked son variables categóricas.

```
describe(dfTrain)
```

```
##             vars   n   mean     sd median trimmed    mad  min    max  range
## PassengerId    1 891 446.00 257.35 446.00  446.00 330.62 1.00 891.00 890.00
## Survived       2 891   0.38   0.49   0.00    0.35   0.00 0.00   1.00   1.00
## Pclass         3 891   2.31   0.84   3.00    2.39   0.00 1.00   3.00   2.00
## Name*          4 891 446.00 257.35 446.00  446.00 330.62 1.00 891.00 890.00
## Sex*           5 891   1.65   0.48   2.00    1.68   0.00 1.00   2.00   1.00
## Age            6 714  29.70  14.53  28.00   29.27  13.34 0.42  80.00  79.58
## SibSp          7 891   0.52   1.10   0.00    0.27   0.00 0.00   8.00   8.00
```

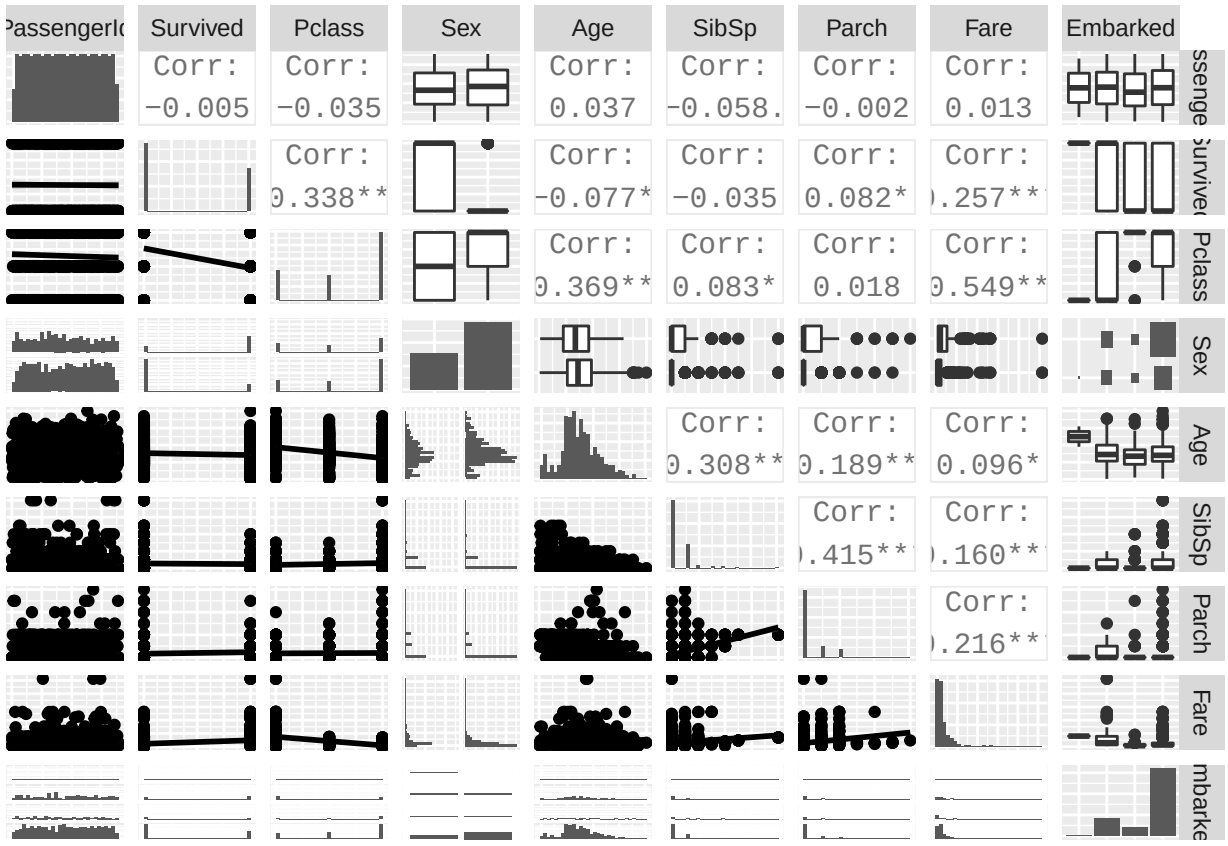
```
## Parch          8 891   0.38   0.81   0.00    0.18   0.00 0.00   6.00   6.00
## Ticket*        9 891 339.52 200.83 338.00  339.65 268.35 1.00 681.00 680.00
## Fare          10 891  32.20  49.69  14.45   21.38  10.24 0.00 512.33 512.33
## Cabin*        11 891  18.63  38.14   1.00    8.29   0.00 1.00 148.00 147.00
## Embarked*     12 891   3.53   0.80   4.00    3.66   0.00 1.00   4.00   3.00
##                skew kurtosis   se
## PassengerId  0.00    -1.20 8.62
## Survived     0.48    -1.77 0.02
## Pclass      -0.63    -1.28 0.03
## Name*        0.00    -1.20 8.62
## Sex*        -0.62    -1.62 0.02
## Age          0.39     0.16 0.54
## SibSp        3.68    17.73 0.04
## Parch        2.74     9.69 0.03
## Ticket*      0.00    -1.28 6.73
## Fare         4.77    33.12 1.66
## Cabin*       2.09     3.07 1.28
## Embarked*   -1.27    -0.16 0.03
```

Ya en este punto, sobre este conjunto de dato:

1.- Si observamos el valor de la media de la variable Survided, podemos inferir que en torno al 38% de los pasajeros contemplados en el dataset, sobrevivieron al naufrágio. 2.- Si observamos la media y la mediana de la variable Pclass, podríamos decir que la mayoría de pasajeros tenían tickets de tercera clase. 3.- Si observamos la media y la mediana de edad (Age), podríamos decir que mayoritariamente era adulta de mediana edad (según el criterio de aquella época, que hoy serían considerados jóvenes) personas entre 20 y 40 años, si bien existen 177 pasajeros/as sobre las/los que no conocemos su edad. 4.- Si observamos la variable Fare (media, mediana y cuartiles 1 y 3, mínimo y máximo), podemos inferir que, si bien el rango de tarifas era amplio (de 0 a 512), la amplia mayoría de tickets pertenecía a los precios más bajos.

A continuación, se ejecuta la función ggpairs del paquete GGally , esta función da un análisis de las variables de un data set por pares, sus correlacciones y sus distribuciones.

```
# Se elimina de este estudio las variables name, cabin y tiket ya que al ser nominales cuentan con exces
ds_aux = subset(dfTrain, select = - c(Name,Ticket,Cabin) )
ggpairs(ds_aux, lower = list(continuous = "smooth"),
        diag = list(continuous = "bar"), axisLabels = "none")
```



Aunque con la función ggpairs aporta mucha información acerca de las distribuciones y correlacciones de las variables, los gráficos aportados son excesivamente pequeños y no se entienden bien. Para visualizar un poco mejor dichos gráficos se recomienda, si está visualizando desde RStudio, usar la opción ver en nueva ventana, que permite ver la matríz de gráficos, más grande.

Queriamos aprovechar tambien la potencia y los concimientos que tenemos de python, y hemos estimado el usar ambos lenguajes en esta practica usando la libreria Reticulate.

```
# Transfomamos el dataset de R a un dataframe de Pandas (python)
pdTrain <- r_to_py(dfTrain)
```

Se observan los tipos de variables que del dataframe transformado:

```
r.pdTrain.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 891 entries, 0 to 890
## Data columns (total 12 columns):
##  #   Column       Non-Null Count  Dtype
## ---  ------       --------------  -----
##  0   PassengerId  891 non-null    int32
##  1   Survived     891 non-null    int32
##  2   Pclass       891 non-null    int32
##  3   Name         891 non-null    object
##  4   Sex          891 non-null    object
##  5   Age          714 non-null    float64
```

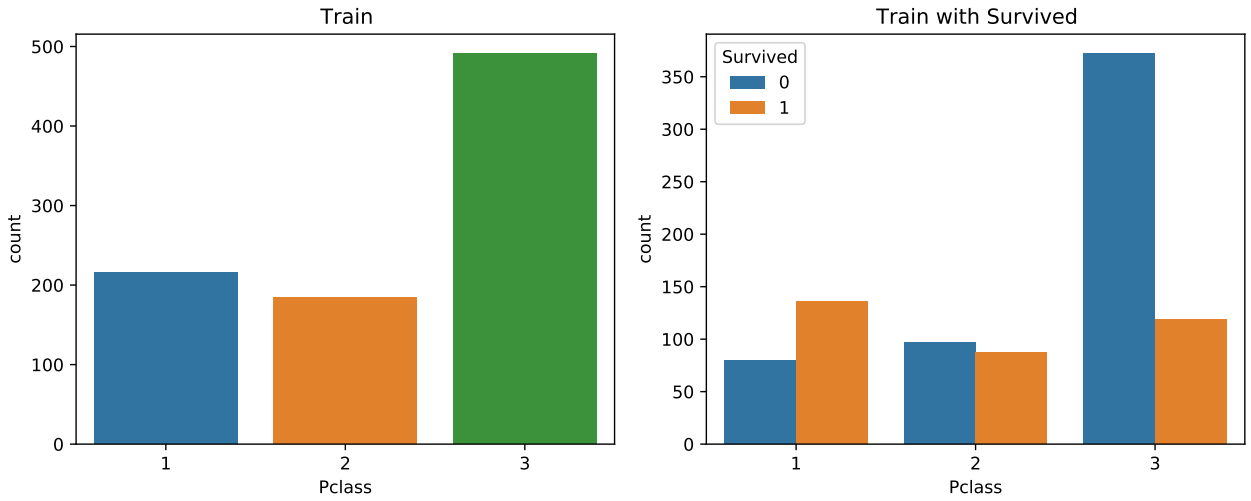
```
##  6   SibSp        891 non-null    int32
##  7   Parch        891 non-null    int32
##  8   Ticket       891 non-null    object
##  9   Fare         891 non-null    float64
##  10  Cabin        891 non-null    object
##  11  Embarked     891 non-null    object
## dtypes: float64(2), int32(5), object(5)
## memory usage: 66.3+ KB
```

A continuación definimos, en Python, dos funciones que nos permitiran generar las diferentes visualizaciones de Distribuciones de datos e Histogramas que deseamos utilizar:

```
#test["Pclass"].value_counts().sort_index().plot(kind="bar", colormap='Paired')
def viewDist(x, data, hue=None):
    fig, axs = plt.subplots(ncols=2, constrained_layout=True, figsize=(10,4))

    sns.countplot(x=x, hue=hue, data=data, ax=axs[0]).set(title='Train')
    sns.countplot(x=x, hue='Survived', data=data, ax=axs[1]).set(title='Train with Survived')
    plt.show()

def viewHist(x, data, hue=None):
    fig, axs = plt.subplots(ncols=2, constrained_layout=True,figsize=(15,4))

    sns.histplot(data[x], ax=axs[0]).set(title='Train Hist')
    sns.boxplot(x=data[x], ax=axs[1]).set(title='Train Box')
    plt.show()
```

Se observa la distribución del campo PClass, tanto de marera aislada, como relacionada con la variable que se quiere predecir (Survived).

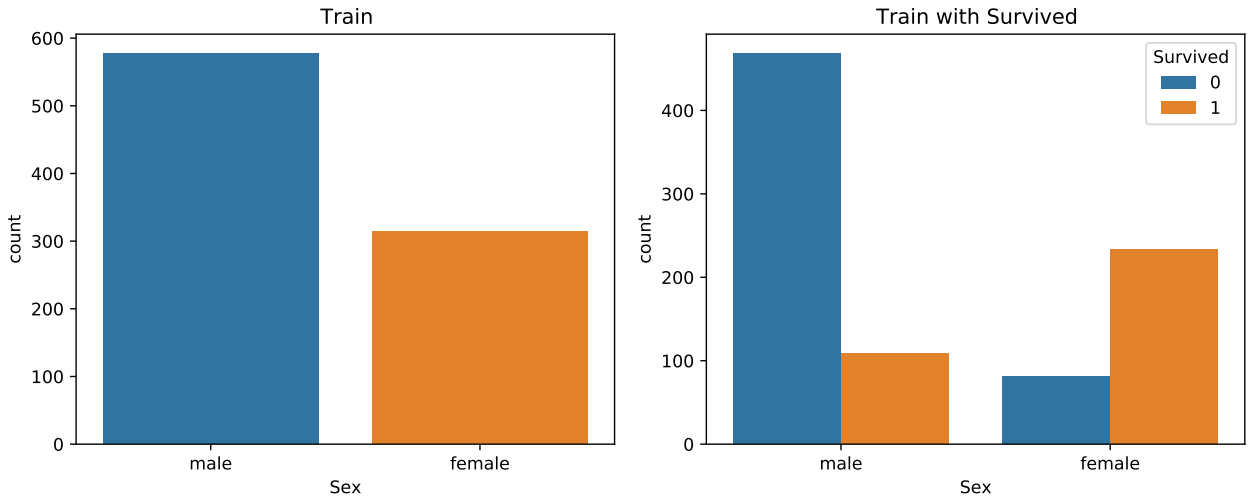
```
viewDist("Pclass", r.pdTrain)
```



Aquí podemos ver, tal y como se comentó al analizar los resultados de sumary y describe, que la mayoría de pasajeros se ubican en 3a clase. También podemos observar que el porcentaje de supervivientes en la 3a clase, es mucho menor que en primera y segunda.

Se observa la distribución del campo Sex, tanto de marera aislada, como relacionada con la variable que se quiere predecir (Survived).

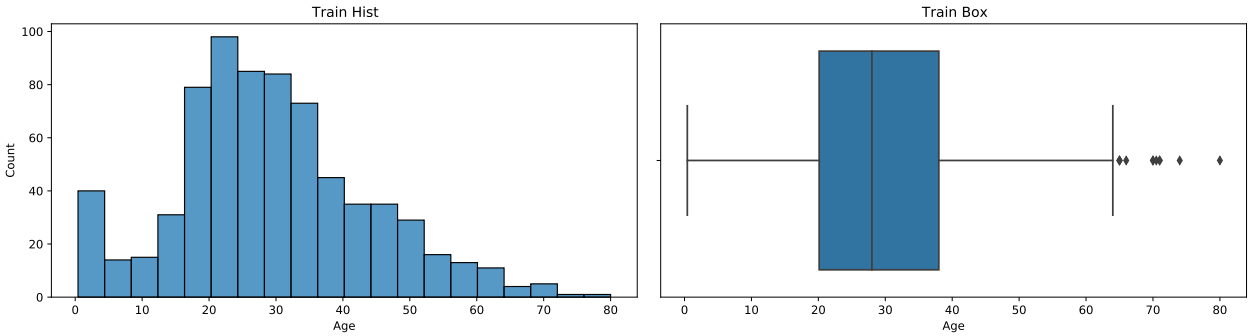
```
viewDist("Sex", r.pdTrain)
```



Si observamos los gráficos anteriores, vemos que a pesar de que embarcaron casi el doble de hombres que mujeres, el número de supervivientes masculinos fue menos de la mitad que el número de mujeres.

A continuación, se observa el histograma sobre el campo Age, y además, un boxplot que nos permite localizar posibles candidatos a outlier.

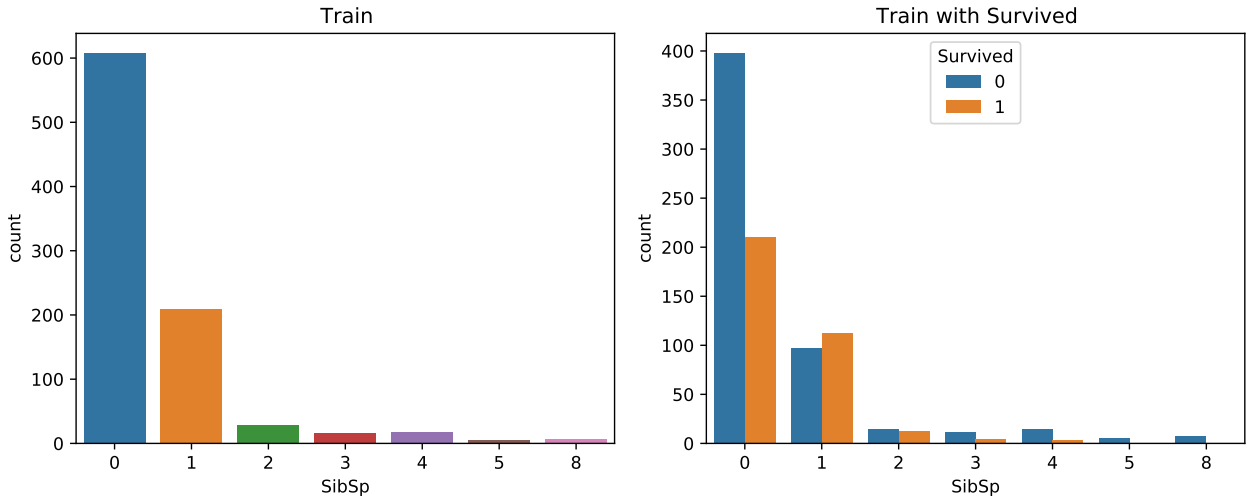
```
viewHist("Age", r.pdTrain)
```



Si nos fijamos en el boxplot basado en la variable Age, vemos que la gran mayoría del pasaje entaba en la franja entre 20 y 40 años, y no abundaban personas con más de 65 años.

A continuación, se observa la distribución del campo SibSp, tanto de marera aislada, como relacionada con la variable que se quiere predecir (Survived).

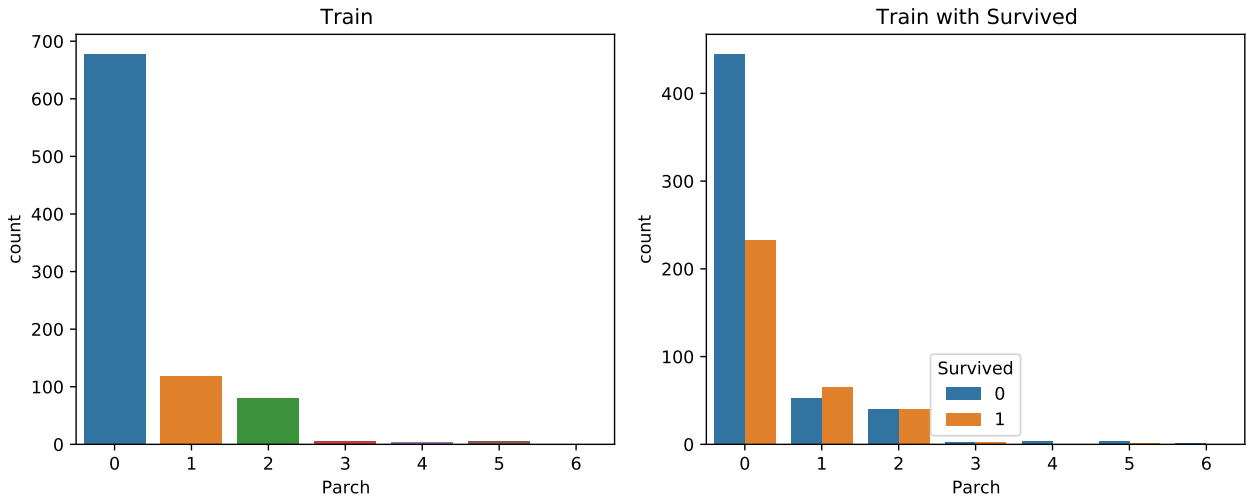
```
viewDist("SibSp", r.pdTrain)
```



Si nos atenemos a la distribución de la variable SibSp, si los datos son correctos, podríamos decir que la mayoría de pasajeros viajaban sin hermanos ni pareja.

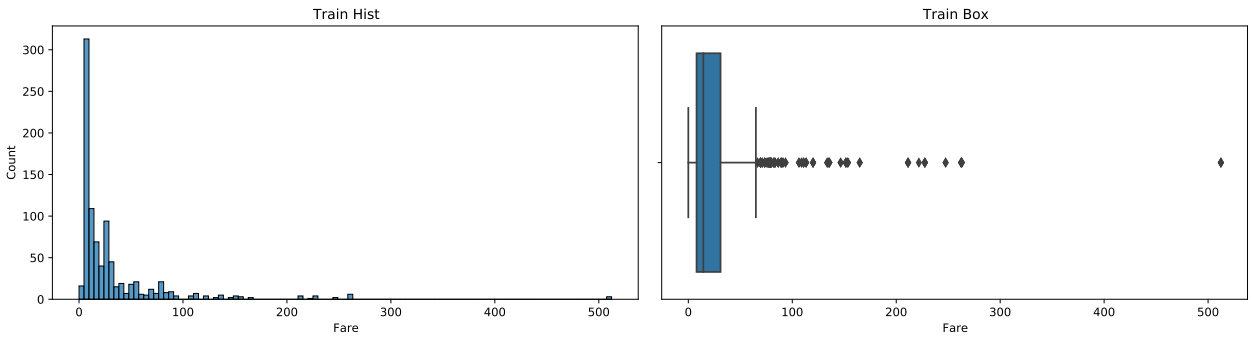
A continuación, se observa la distribución del campo Parch, tanto de marera aislada, como relacionada con la variable que se quiere predecir (Survived).

```
viewDist("Parch", r.pdTrain)
```



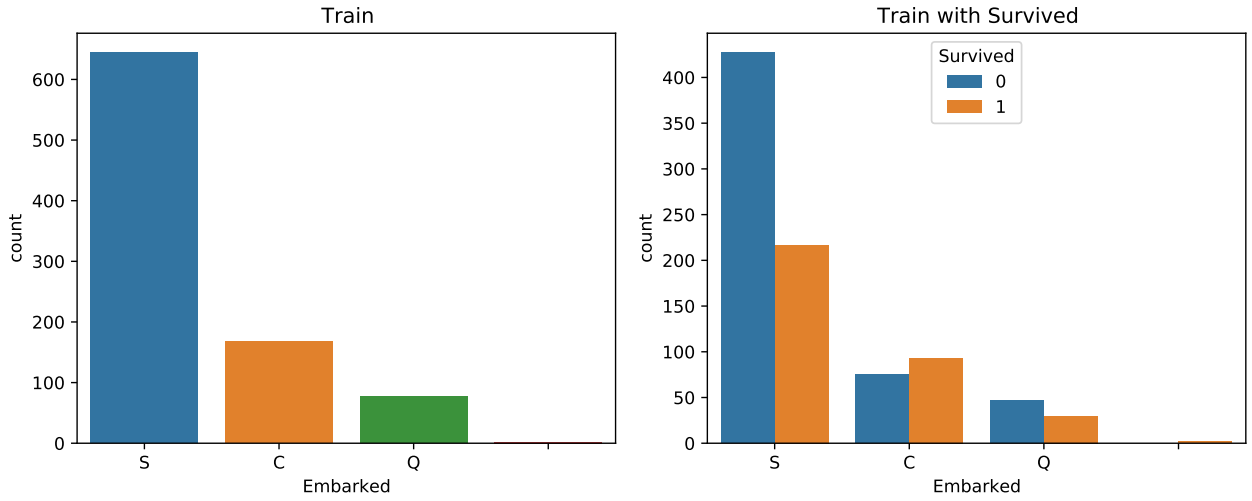
En cuanto a la variable Parch que nos indica si el pasajero viajaba con padres o hijos, si observamos la distribución de los valores, podríamos decir que la mayoría de pasajeros viajaban sin padres ni hijos.

```
viewHist("Fare", r.pdTrain)
```



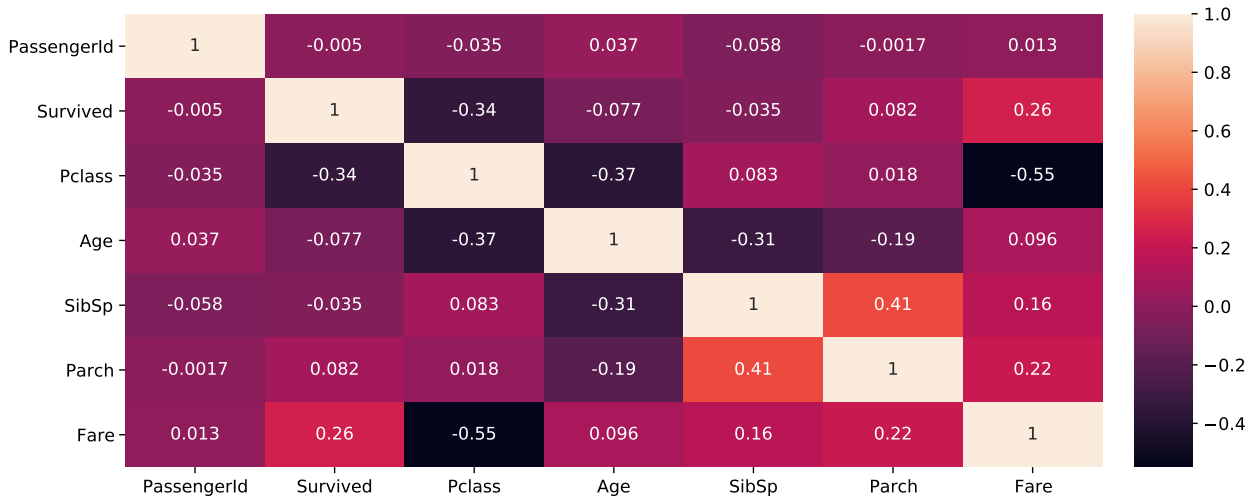
Si observamos el histograma y el boxplot realizados sobre la variable Fare, podríamos considerar como outliers aquellas tarifas que superen las 60 Libras, sin embargo, este tema se aborda en el apartado de outliers.

```
viewDist("Embarked", r.pdTrain)
```



Podemos ver que la mayoría de los pasajeros embarcaron en Southampton(S), sin embargo, el indice de supervivientes es mayor entre los que embarcaron en Cherbourg(C) o Queenstown (Q).

```
correlation_mat = r.dfTrain.corr()
sns.heatmap(correlation_mat, annot = True)
plt.show()
```



.- Limpieza de los datos.

Preparación de datos (Según las necesidades del modelo a aplicar en cada caso)

.- ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

Observamos los valores nulos de las diferentes variables:

```
# Mediante observación directa del fichero csv y las información obtenida en el estudio ya realizado, s

# Número de valores "" contenidos por la variable Cabin
cat("Numero de valores nulos en Cavin:", NROW(dfTrain$Cabin[dfTrain$Cabin == ""]))
```

```
## Numero de valores nulos en Cavin: 687
```

```
# Valores únicos para Embarked
cat("Valores únicos para Embarked:")
```

```
## Valores únicos para Embarked:
```

```
unique(dfTrain$Embarked)
```

```
## [1] "S" "C" "Q" ""
```

```
# Valores nulos en enbarked
cat("Numero de valores nulos para Embarked: ",NROW(dfTrain$Embarked[dfTrain$Embarked == ""]))
```

```
## Numero de valores nulos para Embarked:  2
```

```
#Recodificamos como valores desconocidos aquellos valores == "" en los campos mencionados anteriormente
dfTrain$Cabin[dfTrain$Cabin == ""] <- NA
dfTrain$Embarked[dfTrain$Embarked == ""] <- NA

# Estadísticas de valores vacíos
colSums(is.na(dfTrain))
```

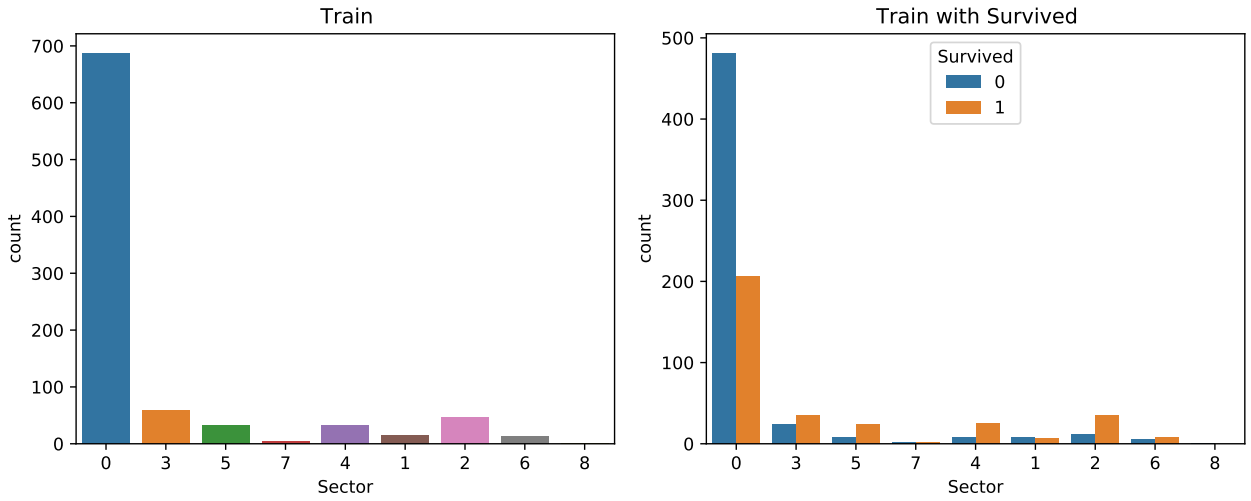
```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0           0           0           0           0         177
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0         687           2
```

A continuación se estudia la opción de reemplazar los valores nulos de la variable Cabin por algo más útil. Según parece, la letra que precede al número que identifica una cabina, indica la sección donde se ubica la misma, generamos, y factorizamos, una nueva variable "Sector" a partir de la letra inicial de la cabina. Los valores nulos se reemplazan por el valor "0".

```
dfTrain$Sector[!is.na(dfTrain$Cabin)] <- as.factor(substr(dfTrain[!is.na(dfTrain$Cabin),]$Cabin,1,1))
dfTrain$Sector[is.na(dfTrain$Cabin)] <- "0"
#dfTrain$Sector
```

A continuación, se observa la distribución de la nueva variable (Sector), tanto de marera aislada, como relacionada con la variable que se quiere predecir (Survived).

```
viewDist("Sector", r.dfTrain)
```




```
# Creamos nuevo data frame sobre el que trabajaremos para obtener todas las variables de tipo numérico,
dfTrain_cat <- dfTrain

# Verificamos la estructura del juego de datos y factorizamos las variables categóricas (Sex, Embarked y
str(dfTrain_cat)
```

```
## 'data.frame':    891 obs. of  13 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
```

```
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  NA "C85" NA "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
##  $ Sector     : chr  "0" "3" "0" "3" ...
```

```
# Discretizamos la edad creando 5 rangos de edades: Bebé(0-3], Infante(3-13], joven(13-29), Adulto(29,6
dfTrain_cat$Age <- cut(dfTrain_cat$Age, c(0,3,13,29,60,100), labels=c(1:5))


# Eliminamos del dataset las variables Ticket y Cabin
dfTrain_cat <- dfTrain_cat[-c(9,11)]

# Creamos nuevo data frame sobre el que trabajaremos para obtener todas las variables de tipo numérico,

dfTrain_cat$Sex <- as.factor(dfTrain_cat$Sex)
dfTrain_cat$Embarked <- as.factor(dfTrain_cat$Embarked)
dfTrain_cat$Age <- as.factor(dfTrain_cat$Age)
dfTrain_cat$Pclass <- as.factor(dfTrain_cat$Pclass)
dfTrain_cat$SibSp <- as.factor(dfTrain_cat$SibSp)
dfTrain_cat$Parch <- as.factor(dfTrain_cat$Parch)
dfTrain_cat$Sector<- as.factor(dfTrain_cat$Sector)

# Eliminamos del dataset la variable Name
dfTrain_cat <- dfTrain_cat[-4]

# Eliminamos del dataset la variable PassengerId
dfTrain_cat <- dfTrain_cat[-1]


str(dfTrain_cat)
```

```
## 'data.frame':    891 obs. of  9 variables:
##  $ Survived: int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Sex     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age     : Factor w/ 5 levels "1","2","3","4",..: 3 4 3 4 4 NA 4 1 3 3 ...
##  $ SibSp   : Factor w/ 7 levels "0","1","2","3",..: 2 2 1 2 1 1 1 4 1 2 ...
##  $ Parch   : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 2 3 1 ...
##  $ Fare    : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked: Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
##  $ Sector  : Factor w/ 9 levels "0","1","2","3",..: 1 4 1 4 1 1 6 1 1 1 ...
```

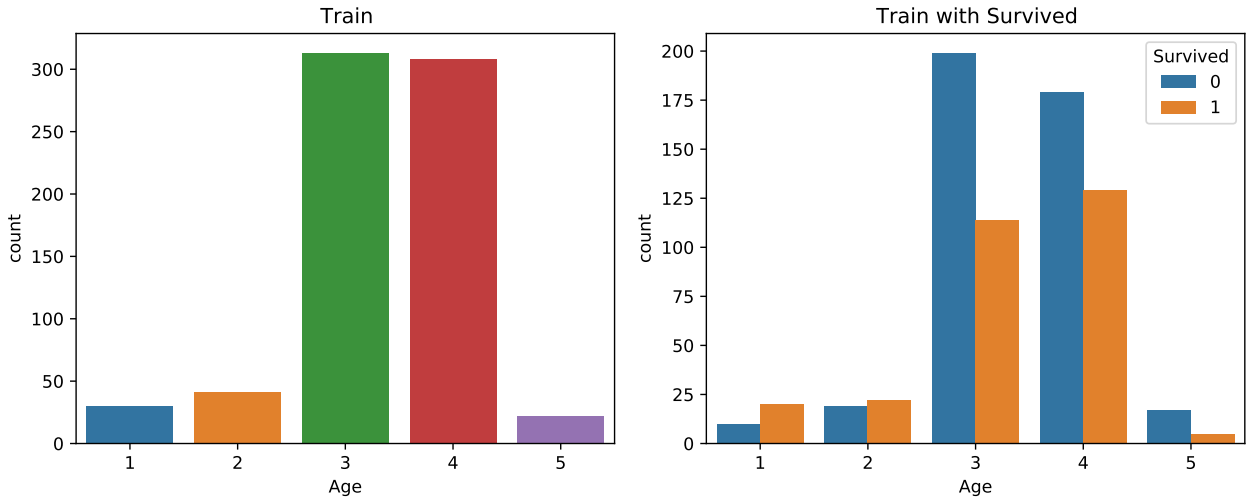
Usamos la funcion kNN (vecinos mas cercanos para la imputacion de los valores nulos en las variables. Nuestra primera opcion fue usar la media sin embargo pensamos que usar un método de imputacion basado en la similitud seria más correcto.

```
dfTrain_cat$Embarked <- kNN(dfTrain_cat)$Embarked
colSums(is.na(dfTrain_cat))
```

```
## Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
##        0        0        0      177        0        0        0        0
##   Sector
##        0
```

Una vez se tienen el campo edad categorizado, volvemos a aplicar el modelo a ver si esta vez es viable su uso. (Se ha ejecutado varias veces para ver los p-valores de las variables y descartar aquellas que no tengan significancia). Pero antes de ello visualizamos como esta distribuida la variable Age una vez categorizada y su relaccion con la variable survived.

```
viewDist("Age", r.dfTrain_cat)
```



Observando la gráfica podemos entender que si eras bebe tenias muchas probabilidades de sobrevivir y los niños un poco más del 50%. Si eras un anciano las posibilidades eran bajas y para el resto de rango de edades estaba parejo, por lo que se podría categorizar en la misma franja (quedando los siguientes rangos de edad: bebes, niños, adultos y ancianos).

```
dfTrain_cat_age <- dfTrain_cat[!is.na(dfTrain_cat$Age),]

model_age_cat <- lm(as.numeric(Age)~as.numeric(Pclass) + as.numeric(SibSp) + as.numeric(Parch), data = d
summary(model_age_cat)
```

```
##
## Call:
## lm(formula = as.numeric(Age) ~ as.numeric(Pclass) + as.numeric(SibSp) +
##     as.numeric(Parch), data = dfTrain_cat_age)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.36670 -0.33002  0.04951  0.50968  1.88055
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)         4.51930    0.09105  49.634  < 2e-16 ***
## as.numeric(Pclass) -0.26992    0.03207  -8.416  < 2e-16 ***
## as.numeric(SibSp)  -0.25590    0.03130  -8.176 1.35e-15 ***
## as.numeric(Parch)  -0.12362    0.03404  -3.632 0.000302 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7162 on 710 degrees of freedom
## Multiple R-squared:  0.2231, Adjusted R-squared:  0.2198
## F-statistic: 67.96 on 3 and 710 DF,  p-value: < 2.2e-16
```

Se observa qué el Ajusted R-squared es bastante bajo (0,2198). Hay que tener en cuenta que el dataset se ha quedado desbalancedo, ya que hay muchos registros del tipo 3 y 4, pocos del tipo 1, 2 y 5.

Probamos otro modelo, tratando de conseguir una mayor precisión.

```
## 80% of the sample size
smp_size <- floor(0.80 * nrow(dfTrain_cat_age))

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(dfTrain_cat_age)), size = smp_size)

train <- dfTrain_cat_age[train_ind, ]
test <- dfTrain_cat_age[-train_ind, ]

# Usamos Bagging nos permite combinar los resultados de varios modelos de clasificación para conseguir
fit <- bagging(Age~.-Sector, data=train)

# Predecimos los valores del test
pred <- predict(fit, test)
```

```
#t = table(pred, test$Age)
confusionMatrix(pred, test$Age)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5
##          1  1  1  0  0  0
##          2  1  2  2  1  0
##          3  1  3 39 39  0
##          4  0  1 18 28  4
##          5  0  0  1  1  0
##
## Overall Statistics
##
##                Accuracy : 0.4895
##                  95% CI : (0.4051, 0.5744)
##     No Information Rate : 0.4825
##     P-Value [Acc > NIR] : 0.4663
##
##                   Kappa : 0.1267
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity          0.333333  0.28571   0.6500   0.4058  0.00000
## Specificity          0.992857  0.97059   0.4819   0.6892  0.98561
## Pos Pred Value       0.500000  0.33333   0.4756   0.5490  0.00000
## Neg Pred Value       0.985816  0.96350   0.6557   0.5543  0.97163
## Prevalence           0.020979  0.04895   0.4196   0.4825  0.02797
## Detection Rate       0.006993  0.01399   0.2727   0.1958  0.00000
## Detection Prevalence 0.013986  0.04196   0.5734   0.3566  0.01399
## Balanced Accuracy    0.663095  0.62815   0.5660   0.5475  0.49281
```

Si nos fijamos en el accuracy tenemos un modelo muy pobre, uno de los problemas es que los datos estan poco balanceados, esto se podria ajustar usando la funcion SMOTE. Sin embargo no vamos a seguir profundizando, finalmente descartamos el usar un modelo custom para predecir la edad y lo solucionamos de la misma manera que se ha hecho con las variables fare y embarked.

```
dfTrain_cat$Age <- kNN(dfTrain_cat)$Age
colSums(is.na(dfTrain_cat))
```

```
## Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
##        0        0        0        0        0        0        0        0
##   Sector
##        0
```

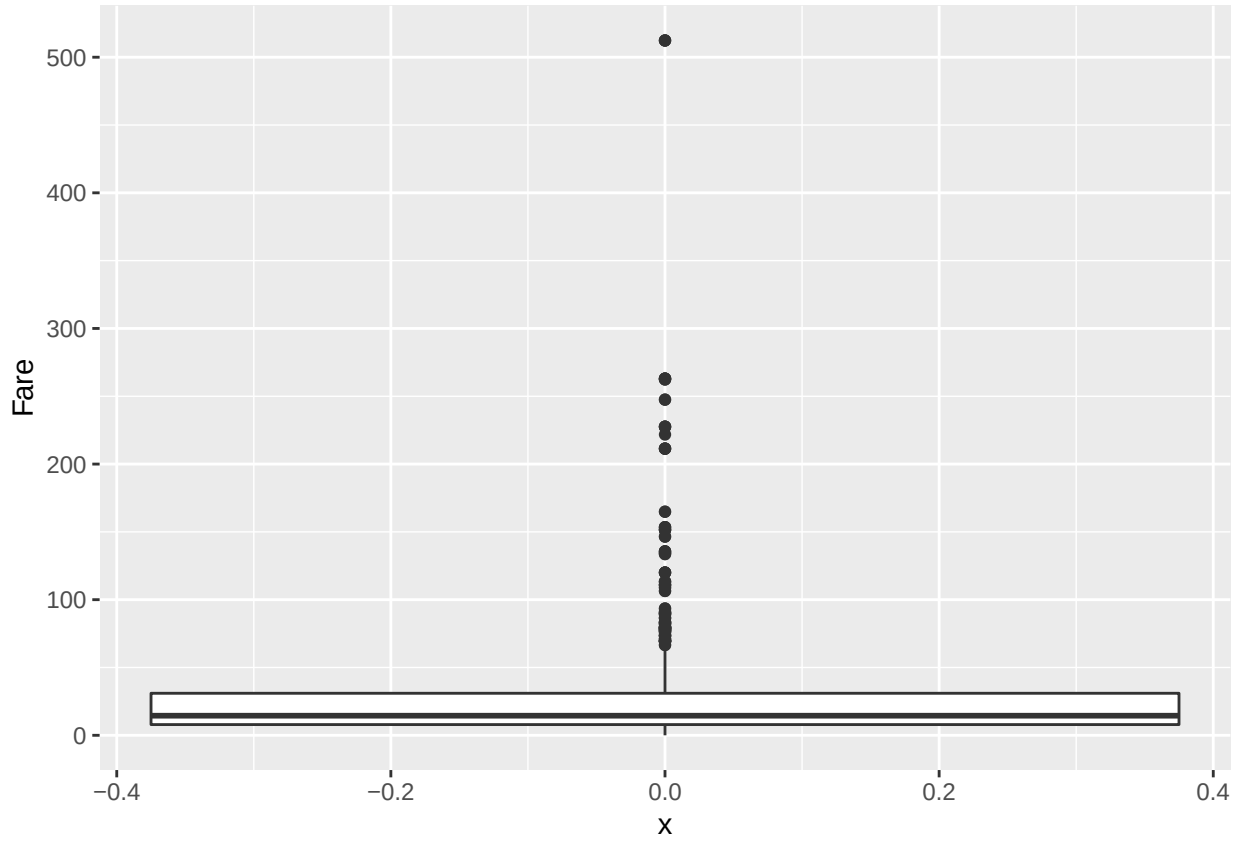
```
# Creamos nuevo data frame sobre el que trabajaremos para obtener todas las variables de tipo numérico,
dfTrain_fact <- dfTrain_cat

# Verificamos la estructura del juego de datos y factorizamos las variables categóricas (Sex, Embarked y
dfTrain_fact$Sex <- as.numeric(as.factor(dfTrain_fact$Sex))
dfTrain_fact$Embarked <- as.numeric(as.factor(dfTrain_fact$Embarked))
dfTrain_fact$Age <- as.numeric(as.factor(dfTrain_fact$Age))
dfTrain_fact$Pclass <- as.numeric(as.factor(dfTrain_fact$Pclass))
dfTrain_fact$SibSp <- as.numeric(as.factor(dfTrain_fact$SibSp))
dfTrain_fact$Parch <- as.numeric(as.factor(dfTrain_fact$Parch))
dfTrain_fact$Sector <- as.numeric(as.factor(dfTrain_fact$Sector))

str(dfTrain_fact)
```

```
## 'data.frame':    891 obs. of  9 variables:
##  $ Survived: int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass  : num  3 1 3 1 3 3 1 3 3 2 ...
##  $ Sex     : num  2 1 1 1 2 2 2 2 1 1 ...
##  $ Age     : num  3 4 3 4 4 4 4 1 3 3 ...
##  $ SibSp   : num  2 2 1 2 1 1 1 4 1 2 ...
##  $ Parch   : num  1 1 1 1 1 1 1 2 3 1 ...
##  $ Fare    : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked: num  3 1 3 3 3 2 3 3 3 1 ...
##  $ Sector  : num  1 4 1 4 1 1 6 1 1 1 ...
```

valores extremos:

En las visualizaciones de cajas que se han realizado en el primer apartado se pudo ver que la variable Fare podia tener valores extremos.

```
# Buscamos valores extremos en la variable Fare
ggplot(data = dfTrain_cat,
       mapping = aes(x = 0,
                     y = Fare)
       ) +
  geom_boxplot()
```



Si observamos el boxplot realizado sobre la variable Fare, podríamos observar que a partir de 60 Libras, se podrían considerar valores extremos. Su tratamiento podría realizarse mediante la omisión de aquellas observaciones asociadas a dichos valores extremos, la asignación del valor máximo permitido sin ser outlier o incluso, o categorizando la variable incluyendo todos los valores.

Finalmente, se opta por no considerar la existencia de valores extremos, pues se tratan de valores que, aun siendo mayores que el resto de observaciones, son posibles teniendo en cuenta que habían camarotes de autentico lujo.Así pues, no se realiza tratamiento alguno sobre estos casos. Además, usaremos un modelo robusto ante la existencia de valores extremos.

.- Análisis de los datos.

.- Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

Se preparan los datos para realizar los caontrastes de hipostesis

```
dfTrain.mujer <- dfTrain_fact[dfTrain_fact$Sex == "1",]$Survived
dfTrain.hombre <- dfTrain_fact[dfTrain_fact$Sex == "2",]$Survived
dfTrain.niños = dfTrain_fact[dfTrain_fact$Age == "1" | dfTrain_fact$Age == "2",]$Survived
dfTrain.adultos = dfTrain_fact[dfTrain_fact$Age != "1" & dfTrain_fact$Age != "2",]$Survived
```

.- Comprobación de la normalidad y homogeneidad de la varianza.

```
alpha = 0.05
col.names = colnames(dfTrain_fact)

for (i in 1:ncol(dfTrain_fact)) {
  if (i == 1)
    cat("Variables que no siguen una distribución normal:\n")
  if (is.integer(dfTrain_fact[,i]) | is.numeric(dfTrain_fact[,i])) {
    p_val = ad.test(dfTrain_fact[,i])$p.value
    if (p_val < alpha) {
      cat(col.names[i])
      # Format output
      if (i < ncol(dfTrain_fact) - 1)
        cat(", ")
      if (i %% 3 == 0)
        cat("\n")
    }
  }
}
```

```
## Variables que no siguen una distribución normal:
## Survived, Pclass, Sex,
## Age, SibSp, Parch,
## Fare, EmbarkedSector
```

Ahora Estudiamos la homogeneidad de varianzas mediante la aplicacion un test de Fligner-Killeen, en este caso estudiamos la homogeneidad de los grupos formados por los supervivientes y los no supervivientes con respecto al precio del tiket.

```
fligner.test(Fare ~ Survived, data = dfTrain_fact)
```

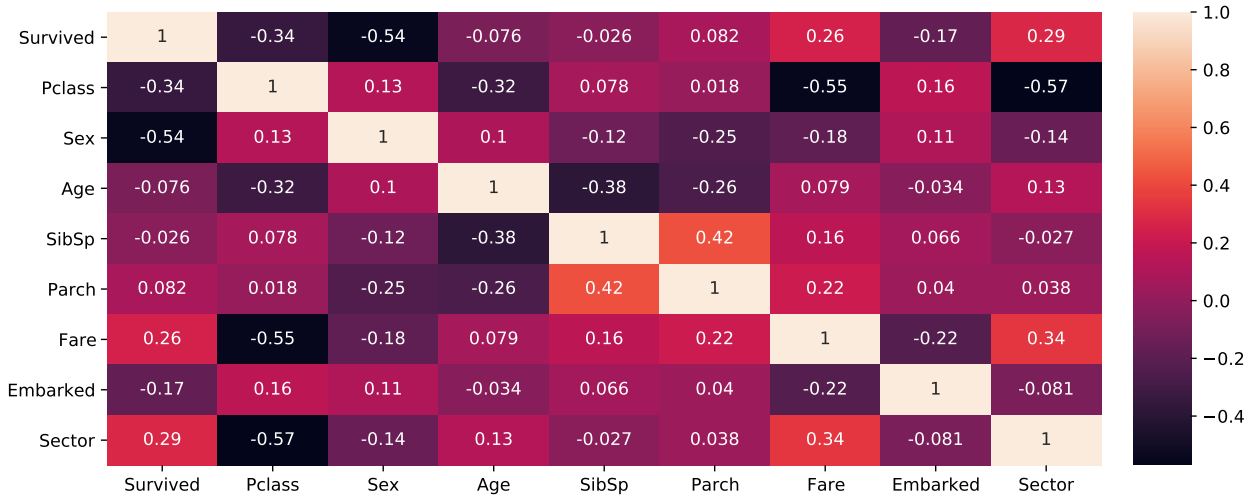
```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fare by Survived
## Fligner-Killeen:med chi-squared = 96.253, df = 1, p-value < 2.2e-16
```

como el p valor es $< 0{,}05$ rechazamos la hipotesis de que las varianzas de ambas muestras sean homogéneas

.- Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

¿Que variables influyen mas en la supervivencia?

```
correlation_mat = r.dfTrain_fact.corr()
sns.heatmap(correlation_mat, annot = True)
plt.show()
```



Usando la matriz de correlacciones usada anteriormente, pero ya con el dataset limpio, podemos concluir que las variables que más influyen en la supervivencia son el sexo, la clase, y el Fare (en ese orden).

¿la supervivencia es mayor en caso de ser niño o mujer (mujeres y niños primero)?

Se plantea el contraste de hipotesis de dos muestras sobre la diferencia de medias, el cual es unilateral atendiendo a la formulación de la hipótesis alternativa:

$H_0 : \mu_h - \mu_m = 0$

$H_1 : \mu_h - \mu_m < 0$

donde µh es la media de la población de la que se extrae la primera muestra y µm es la media de la población de la que extrae la segunda. Así, tomaremos = 0, 05.

```
t.test(dfTrain.hombre, dfTrain.mujer, alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  dfTrain.hombre and dfTrain.mujer
## t = -18.672, df = 584.43, p-value < 2.2e-16
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##        -Inf -0.5043259
## sample estimates:
## mean of x mean of y
## 0.1889081 0.7420382
```

Puesto que obtenemos un p-valor menor que el valor de significación fijado, rechazamos la hipótesis nula. Por tanto, podemos concluir que las mujeres tienen una supervivencia superior a la de los hombres.

Repetimos el procesos para los niños, en este caso consideramos niños a los grupos de edad 1 y 2:

Formulamos el contraste de hipotesis:

$H_0 : \mu_n - \mu_a = 0$

$H_1 : \mu_n - \mu_a < 0$

Y lo realizamos de la misma manera que el anterior

```
t.test(dfTrain.adultos, dfTrain.niños, alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  dfTrain.adultos and dfTrain.niños
## t = -2.8207, df = 101.07, p-value = 0.002885
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##         -Inf -0.06621461
## sample estimates:
## mean of x mean of y
## 0.3684864 0.5294118
```

Puesto que obtenemos un p-valor menor que el valor de significación fijado, rechazamos la hipótesis nula. Por tanto, podemos concluir que los niños tienen una supervivencia superior al resto, por lo tanto podemos afirmar que se cumple el dicho ¡Mujeres y niños primero!

Modelo Predictivo

A continuación se plantea un modelo predictivo para intentar predecir la supervivencia. En este caso se usará un random forest (anteriormente se han probado otros modelos de clasificacion como la regresión logistica y los arboles de decision pero con peores resultados) para evaluar el modelo de una manera eficiente y evitar perder precisión con ello (ya que no se cuenta con excesivos datos) se ha aplicado la técnica de 'cross-validation', dividiendo el dataset en 10 "trozos". No se ha especificado el número de predictores para que sea el propio modelo el que determine este dato.

```
set.seed(14)
dfTrain_fact$Survived = as.factor(dfTrain_fact$Survived)

# Definimos train control
train_control <- trainControl(method='repeatedcv',
                              number=10,
                              repeats=5)

# Entrenamos el modelo
(train.rpart <- train(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, #+ Sector,
                      data=dfTrain_fact,
                      method="rf",
                      metric='Accuracy',
                      trControl=train_control))
```

```
## Random Forest
##
## 891 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 802, 802, 802, 802, 802, 801, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8284974  0.6231060
##   4     0.8309622  0.6341047
##   7     0.8235437  0.6191953
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```
# Imprimimos las puntuaciones de cross-validation
summary(train.rpart)
```

```
##                 Length Class      Mode
## call               4   -none-     call
## type               1   -none-     character
## predicted        891   factor     numeric
## err.rate        1500   -none-     numeric
## confusion          6   -none-     numeric
## votes           1782   matrix     numeric
## oob.times        891   -none-     numeric
## classes            2   -none-     character
## importance         7   -none-     numeric
## importanceSD       0   -none-     NULL
## localImportance    0   -none-     NULL
## proximity          0   -none-     NULL
## ntree              1   -none-     numeric
## mtry               1   -none-     numeric
## forest            14   -none-     list
## y                891   factor     numeric
## test               0   -none-     NULL
## inbag              0   -none-     NULL
## xNames             7   -none-     character
## problemType        1   -none-     character
## tuneValue          1   data.frame list
## obsLevels          2   -none-     character
## param              0   -none-     list
```

```
summary(train.rpart$finalModel)
```

```
##                 Length Class      Mode
## call               4   -none-     call
## type               1   -none-     character
## predicted        891   factor     numeric
```

```
## err.rate        1500   -none-     numeric
## confusion          6   -none-     numeric
## votes           1782   matrix     numeric
## oob.times        891   -none-     numeric
## classes            2   -none-     character
## importance         7   -none-     numeric
## importanceSD       0   -none-     NULL
## localImportance    0   -none-     NULL
## proximity          0   -none-     NULL
## ntree              1   -none-     numeric
## mtry               1   -none-     numeric
## forest            14   -none-     list
## y                891   factor     numeric
## test               0   -none-     NULL
## inbag              0   -none-     NULL
## xNames             7   -none-     character
## problemType        1   -none-     character
## tuneValue          1   data.frame list
## obsLevels          2   -none-     character
## param              0   -none-     list
```

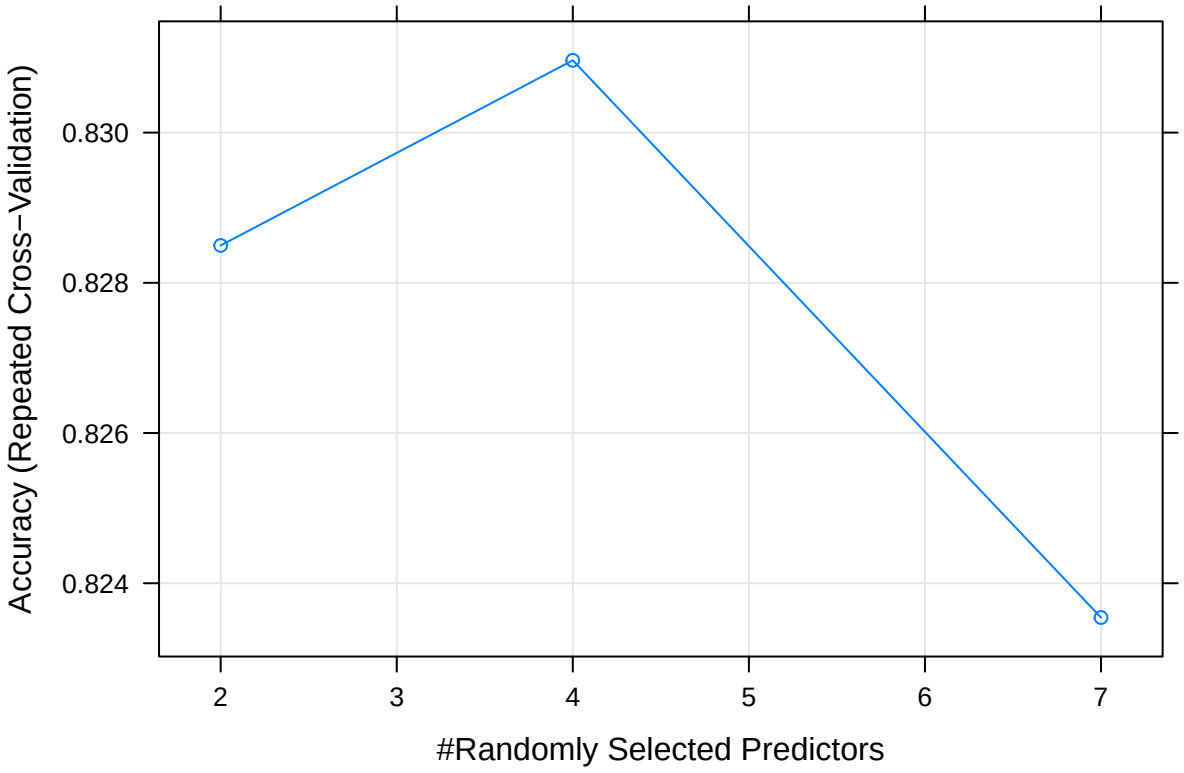
```
# Generamos la predicción
pred <- predict(train.rpart, dfTrain_fact)
```

Podemos observar que el modelo arroja unos resultados decentes con aproximadamente un 83% de precisión (Finalmente se descarto el uso del sector ya que este vajaba la precisión, esto es debido a gran numero de nulos que tiene). El mejor resultado se ha obtenido usando 4 predictores.

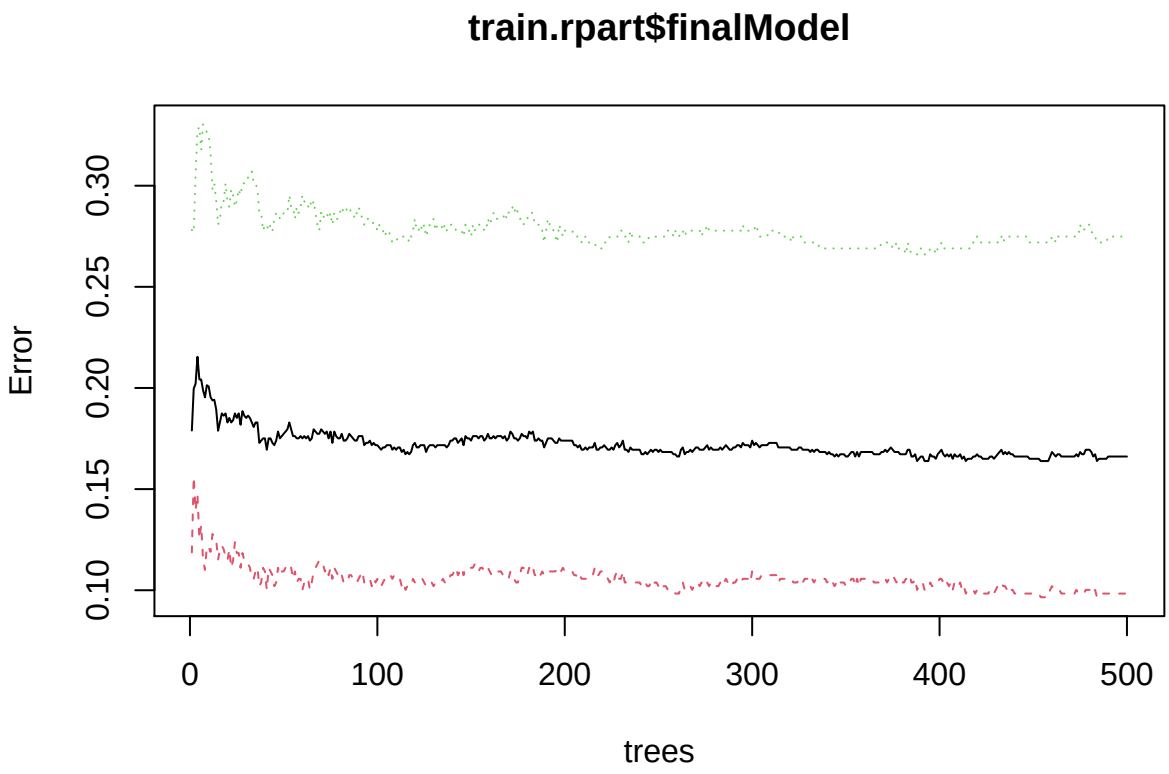
.- Representación de los resultados a partir de tablas y gráficas.

Visualizamos los datos que arroja el modelo:

```
plot(train.rpart)
```




```
plot(train.rpart$finalModel)
```



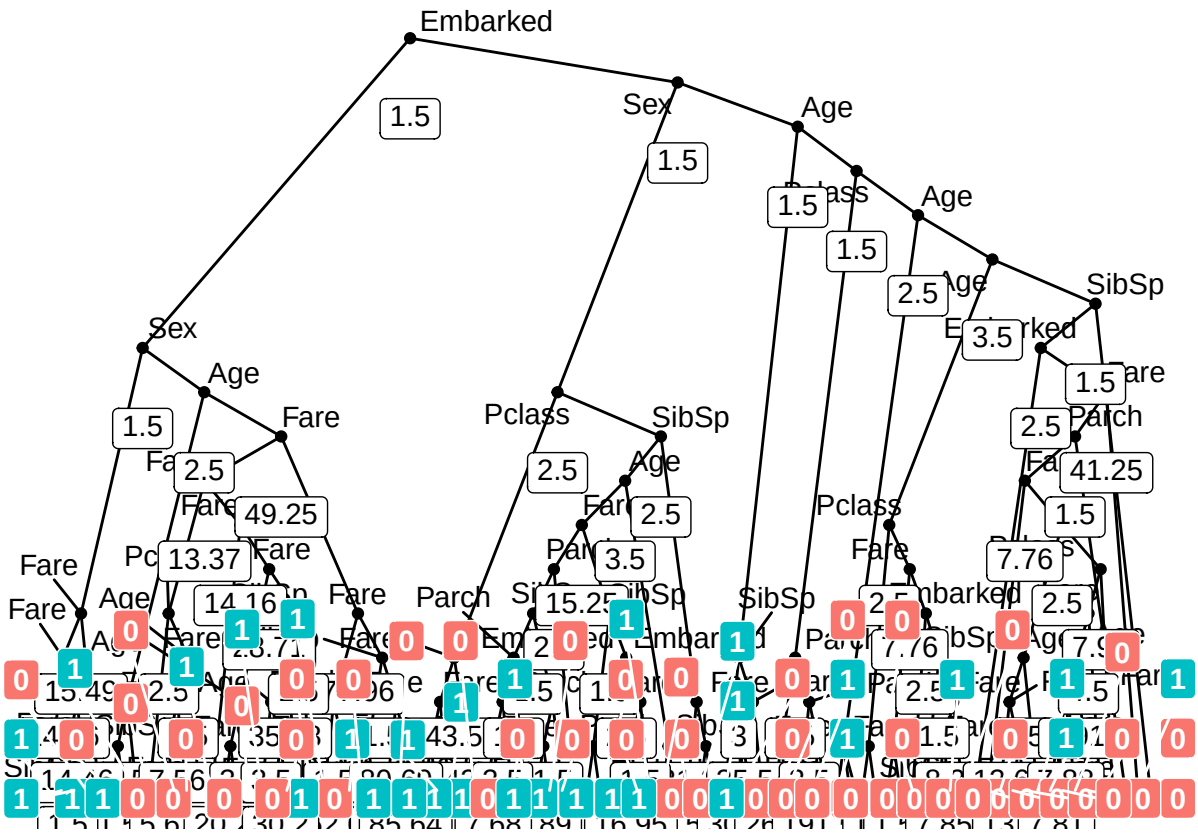
Se utiliza la siguiente función para generar el arbol de decisión final.

```
#Función optenida de https://shiring.github.io/machine_learning/2017/03/16/rf_plot_ggraph
tree_func <- function(final_model,
                      tree_num) {

  # get tree by index
  tree <- randomForest::getTree(final_model,
                                k = tree_num,
                                labelVar = TRUE) %>%
    tibble::rownames_to_column() %>%
    # make leaf split points to NA, so the 0s won't get plotted
    mutate(`split point` = ifelse(is.na(prediction), `split point`, NA))

  # prepare data frame for graph
  graph_frame <- data.frame(from = rep(tree$rowname, 2),
                            to = c(tree$`left daughter`, tree$`right daughter`))

  # convert to graph and delete the last node that we don't want to plot
  graph <- graph_from_data_frame(graph_frame) %>%
    delete_vertices("0")

  # set node labels
  V(graph)$node_label <- gsub("_", " ", as.character(tree$`split var`))
  V(graph)$leaf_label <- as.character(tree$prediction)
  V(graph)$split <- as.character(round(tree$`split point`, digits = 2))
```

```
  # plot
  plot <- ggraph(graph, 'dendrogram') +
    theme_bw() +
    geom_edge_link() +
    geom_node_point() +
    geom_node_text(aes(label = node_label), na.rm = TRUE, repel = TRUE) +
    geom_node_label(aes(label = split), vjust = 2.5, na.rm = TRUE, fill = "white") +
    geom_node_label(aes(label = leaf_label, fill = leaf_label), na.rm = TRUE,
                    repel = TRUE, colour = "white", fontface = "bold", show.legend = FALSE) +
    theme(panel.grid.minor = element_blank(),
          panel.grid.major = element_blank(),
          panel.background = element_blank(),
          plot.background = element_rect(fill = "white"),
          panel.border = element_blank(),
          axis.line = element_blank(),
          axis.text.x = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks = element_blank(),
          axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          plot.title = element_text(size = 30))

  print(plot)
}
```

Visualizamos el árbol de decisión resultante:

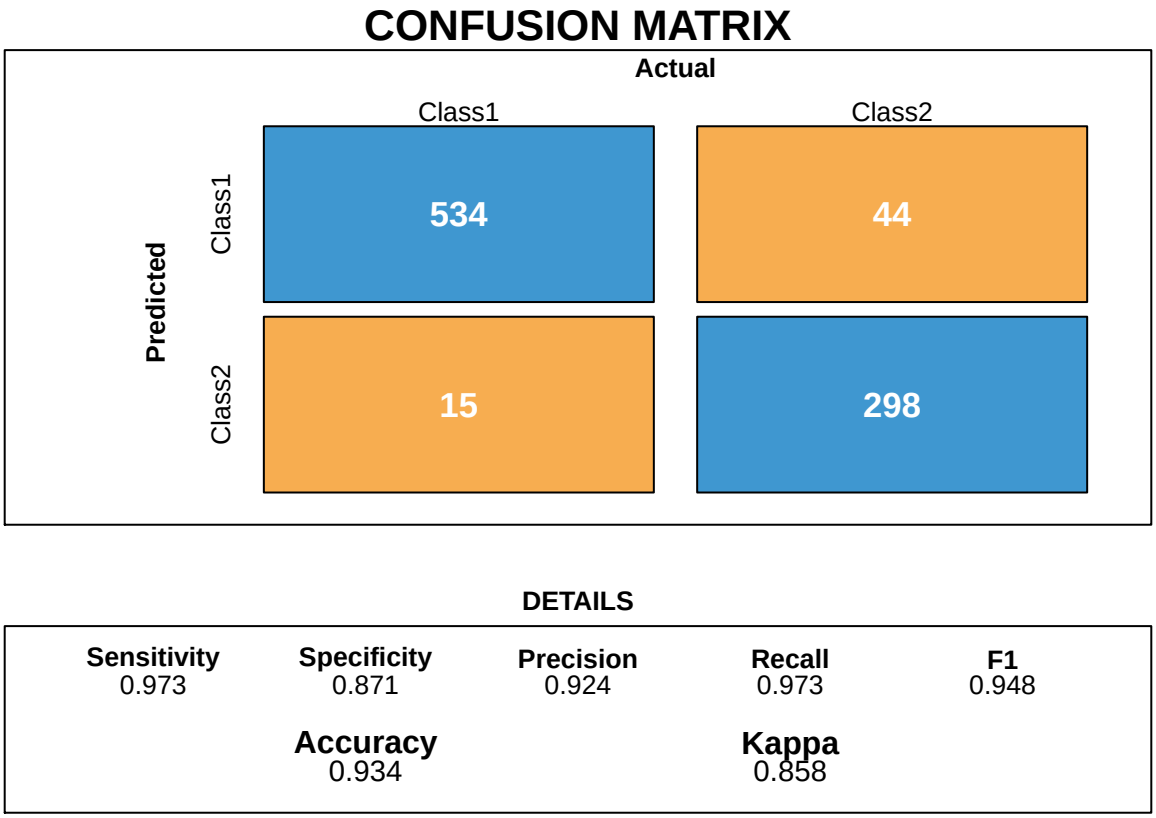
```
tree_num <- which(train.rpart$finalModel$forest$ndbigtree == min(train.rpart$finalModel$forest$ndbigtree
tree_func(final_model = train.rpart$finalModel, tree_num)
```



A continuación analizamos la matriz de confusión:

```
#Función https://stackoverrun.com/es/q/6546365
draw_confusion_matrix <- function(cm) {

    layout(matrix(c(1,1,2)))
    par(mar=c(2,2,2,2))
    plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')
    title('CONFUSION MATRIX', cex.main=2)

    # create the matrix
    rect(150, 430, 240, 370, col='#3F97D0')
    text(195, 435, 'Class1', cex=1.2)
    rect(250, 430, 340, 370, col='#F7AD50')
    text(295, 435, 'Class2', cex=1.2)
    text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
    text(245, 450, 'Actual', cex=1.3, font=2)
    rect(150, 305, 240, 365, col='#F7AD50')
    rect(250, 305, 340, 365, col='#3F97D0')
    text(140, 400, 'Class1', cex=1.2, srt=90)
    text(140, 335, 'Class2', cex=1.2, srt=90)

    # add in the cm results
    res <- as.numeric(cm$table)
    text(195, 400, res[1], cex=1.6, font=2, col='white')
    text(195, 335, res[2], cex=1.6, font=2, col='white')
```

```
    text(295, 400, res[3], cex=1.6, font=2, col='white')
    text(295, 335, res[4], cex=1.6, font=2, col='white')

    # add in the specifics
    plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')
    text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
    text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
    text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
    text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
    text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
    text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
    text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
    text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
    text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
    text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

    # add in the accuracy information
    text(30, 35, names(cm$overall[1]), cex=1.5, font=2)
    text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)
    text(70, 35, names(cm$overall[2]), cex=1.5, font=2)
    text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)
}

cm <- confusionMatrix(pred, train.rpart$trainingData$.outcome, mode = "everything")
draw_confusion_matrix(cm)
```

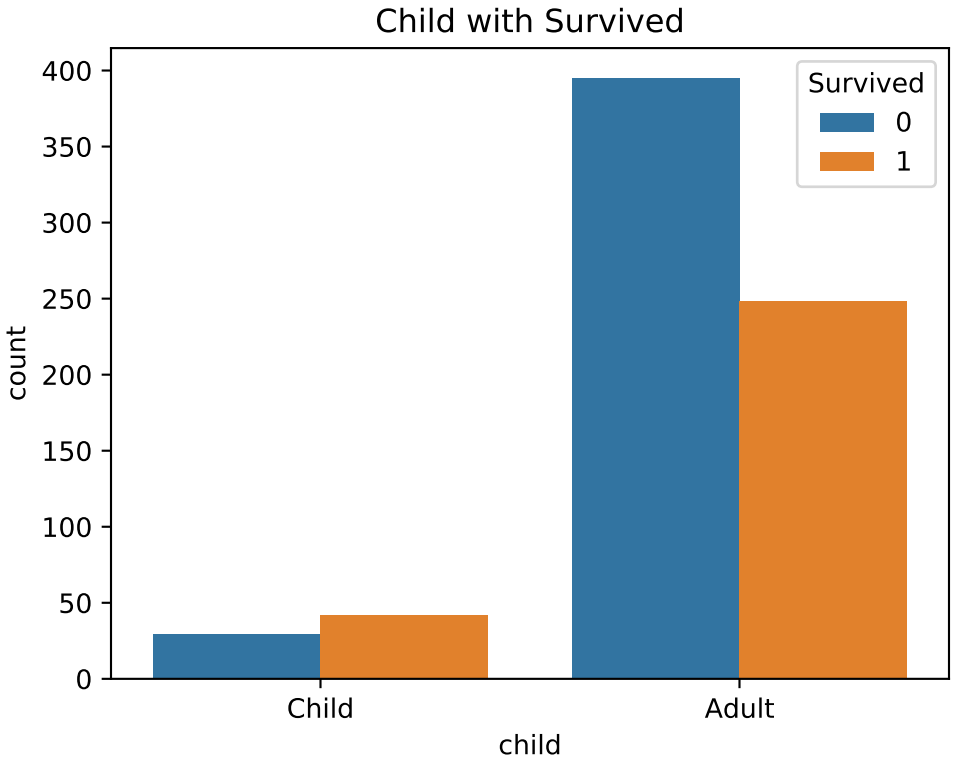


Esta matriz de confusión arroja mejores resultados a los obtenidos mediante el proceso de 'Cross-Validation'(CV). Esto es debido a que se estan usando todos los datos para confeccionarla (ya que no hemos dividido el dataset en train y test). Por lo tanto es mucho más confiable la accuracy optenida en el proceso de CV, sin embargo si que vale para darnos una idea de las otras métricas.

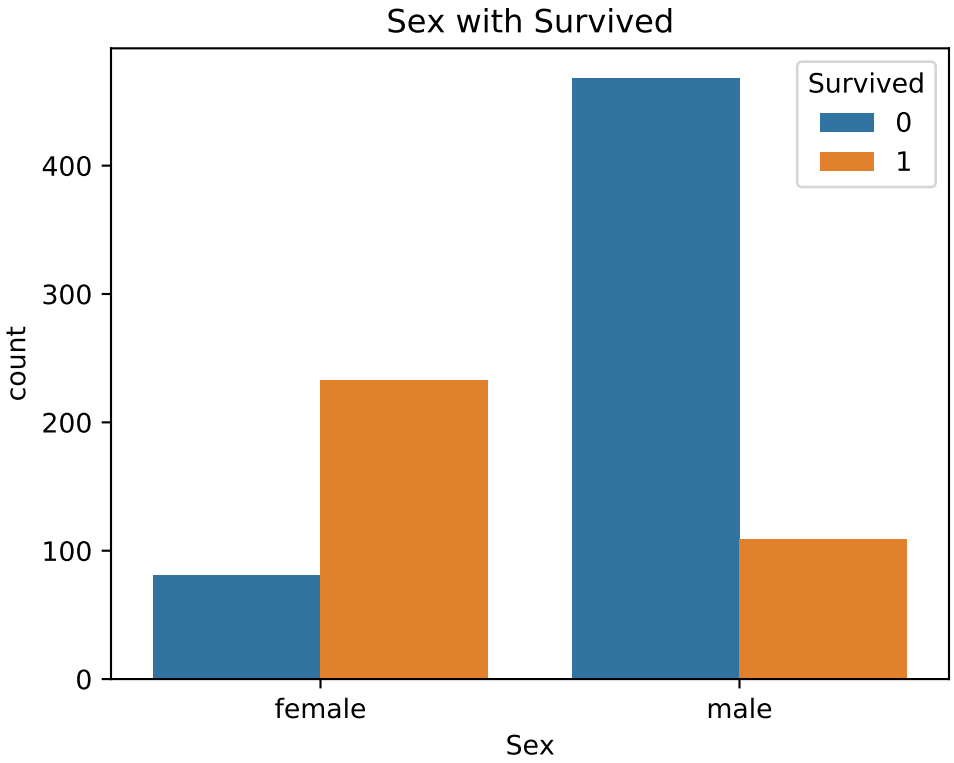
Por otro lado vamos a ver las siguientes gráficas referentes a los contrastes de hipótesis realizados con anterioridad donde se puede observar gráficamente cómo la supervivecia es mucho mayor en el caso de las mujeres y los niños.

```
dfTrain_cat$child <- cut(dfTrain$Age, c(0,13,100), labels=c("Child", "Adult"))
dfTrain_cat$child[dfTrain_cat$child == "1"] <- "Child"
```

```
fig, axs = plt.subplots(ncols=1, constrained_layout=True, figsize=(5,4))
sns.countplot(x="child", hue='Survived', data=r.dfTrain_cat, ax=axs).set(title='Child with Survived')
plt.show()
```




```
fig, axs = plt.subplots(ncols=1, constrained_layout=True, figsize=(5,4))
sns.countplot(x="Sex", hue='Survived', data=r.dfTrain_cat, ax=axs).set(title='Sex with Survived')
plt.show()
```



.- Conclusiones

Al principio de la práctica se planteaban una serie de preguntas que se han ido respondiendo en el transcurso de la misma.

A la pregunta de cuales son las variables más influyentes a la hora de predecir la supervivencia, se respondió con un una tabla de correlación, a la de si las mujeres y los niños tenían una supervivencia mayor, se respondió con contrastes de hipótesis y a la pregunta de si se podía predecir la supervivencia se aporto un modelo (random forest) para el cual se usaron técnicas de validación cruzada para aprovechar mejor el escaso dataset con el que nos encontramos.

Previo a estas pruebas estadísticas, se sometieron los datos a un preprocesamiento para manejar los casos de elementos vacíos y outliers. Para el caso de los elementos vacíos se opto por el uso de modelos de predicción sencillos para evitar tener que eliminar las filas, excepto en el caso de la columna Cabin, que si bien se optó por rellenar los valores desconocidos con un valor por defecto, en la aplicación del modelo se observo que no aportaba nada y finalmente se elimino de esté. El caso de la variable Age es un caso especial ya que intentamos predecirla usando un modelo complejo, pero nos encontramos un problema de datos desbalanceados que finalmente se decidió no abordar y usar el mismo modelo de vecinos más cercanos usado en las otras variables. Para el caso de los outliers, se estudió categorizar la variable, pero finalmente se optó por incluir los valores extremos en los análisis dado que son valores totalmente posibles en un crucero y eliminarlos seria eliminar del estudio los valores de lujo, además el modelo elegido es un modelo muy robusto ante los outliers.

Por otro lado, hemos incluido constantes visualizaciones y comentarios en cada proceso para que sea más fácil entender el dataset en un principio y los resultados obtenidos de las pruebas.

Finalmente, creemos que se ha conseguido un modelo decente para predecir la supervivencia, aunque se podría mejorar si se encontrara una manera de usar mejor la variable cabin y hubieramos podido predecir con más efectividad la edad.

.- Recursos

- Calvo M., Subirats L., Pérez D. (2019). Introducción a la limpieza y análisis de los datos. Editorial UOC.
- Megan Squire (2015). Clean Data. Packt Publishing Ltd.
- Jiawei Han, Micheine Kamber, Jian Pei (2012). Data mining: concepts and techniques.Morgan Kaufmann.
- Jason W. Osborne (2010). Data Cleaning Basics: Best Practices in Dealing with Extreme Scores. Newborn and Infant Nursing Reviews; 10 (1): pp. 1527-3369.
- Peter Dalgaard (2008). Introductory statistics with R. Springer Science & Business Media.
- Wes McKinney (2012). Python for Data Analysis. O'Reilley Media, Inc.

.- Tabla de contribuciones

Contribuciones	Firma
Investigación previa	aruizplaza, rcotillas
Redacción de las respuestas	aruizplaza, rcotillas
Desarrollo código	aruizplaza, rcotillas