Universitat Oberta de Catalunya

Estudios de Informática, Multimedia y Telecomunicaciones

Tipología y ciclo de vida de los datos: PEC 2-Limpieza y análisis de los datos

Autor: Jorge Santos Neila & Javier Cela López

Mayo 2020

1 PEC 2 - Limpieza y análisis de los datos

1.1 Integrantes

En esta segunda práctica de Tipología y ciclo de via de los datos se realiza un caso práctico orientado a aprender a identificar los datos relevantes para un proyecto analítico y uso las herramientas de integración, limpieza, validación y análisis de las mismas. Esta práctica ha sido realizada en un grupo de dos formado por las siguientes personas:

- Jorge Santos Neila
- Javier Cela López

1.2 Objetivos

Los objetivos que se persiguen mediante el desarrollo de esta actividad práctica son los siguientes:

- 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 permita 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.

1.3 Competencias

Por último, las competencias que se buscan desarrollar a lo largo de esta práctica del Máster de Data Science de la UOC son:

- Capacidad de analizar un problema en el nivel de abstracción adecuado a cada situación y aplicar las habilidades y conocimientos adquiridos para abordarlo y resolverlo.
- Capacidad para aplicar las técnicas específicas de tratamiento de datos (integración, transformación, limpieza y validación) para su posterior análisis.

1.4 Organización

A lo largo del código se dará respuesta a las preguntas planteadas por el profesor. Se empezará explicando el dataset seleccionado y los motivos de su elección. A continuación se realizará la tarea de integración y selección de los datos de interés a analizar, seguido del trabajo de limpieza de datos y análisis de datos. El grueso del trabajo estará definido bajo estos dos puntos, ya que son los que más trabajo requieren. Se seguirá con una representación visual de los resultados obtenidos y, para terminar, se expondrán las conclusiones que se han llevado a cabo a lo largo de este ejercicio.

---

2 Descripción del dataset

El Conjunto de datos, *Dataset* en inglés, objeto de análisis se ha obtenido a partir del siguiente enlace en Kaggle. No obstante, este es un ejercicio propuesto por Kaggle para hacer predicciones y utilizar algoritmos de machine learning, por lo que el dataset estaba dividido en dos con los siguientes ficheros:

- Train.csv, dataset específico para entrenar el modelo de machine learning.
- Test.csv, dataset específico para testear o probar el modelo realizado.

Para poder trabajar correctamente con todos los datos, el primer paso que se debe realizar es unificar ambos ficheros en uno sólo. Para ello se utiliza la función *bind-rows*, de la siguiente manera:

```
# Cargamos los paquetes R que vamos a usar
library(ggplot2)
library(dplyr)

# Guardamos el conjunto de datos test y train en un único dataset
test <- read.csv('../dataset/test.csv',stringsAsFactors = FALSE)
train <- read.csv('../dataset/train.csv', stringsAsFactors = FALSE)

# Unimos los dos conjuntos de datos en uno solo
titanic_file <- bind_rows(train, test)

# Verificamos la estructura del conjunto de datos
str(titanic_file)
```

```
## 'data.frame':    1309 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       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
"Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
##  $ 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  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

Tras unificar el fichero, podemos observar que está constituido por 12 características (columnas) que presentan 1309 personas (filas o registros). Este dataset esta compuesto por las siguientes variables:

- **PassengerId**: Identificador único del pasajero. Tipo de dato: **integer**.
- **Survived**: Código que identifica si la persona sobrevivió al accidente del Titanic en 1912. Los dos valores que admite esta variable son 0, si la persona no sobrevivió, o 1 en caso contrario. El tipo de dato es un **integer**. Esta es nuestra variable objetivo y la que queremos utilizar para predecir el modelo.
- **Pclass**: Código que identifica la clase en la que viajaba el pasajero. Los valores posibles son;'1', correspondiente a primera clase. '2', clase media. '3', clase baja. El tipo de dato es un **integer**.
- **Name**: Variable descriptiva que indica el nombre de la persona. Esta variable es una cadena de caracteres del tipo **varchar**.
- **Sex**: Código que identifica el sexo de la persona. Los valores posibles son 'male', si la persona era un hombre, o 'female', si la persona era una mujer. El tipo de dato es un **varchar**.
- **Age**: Número que identifica la edad del pasajero. Esta variable es de tipo **numeric**.
- **SibSp**: Número de hermanos o cónyuges que tenía el pasajero a bordo. Esta variable es de tipo **integer**.
- **Parch**: Número de padres o hijos que tenía cada pasajero a bordo. Esta variable es de tipo **integer**.
- **Ticket**: Identificador del ticket del pasajero. Esta variable es de tipo **varchar**.

- **Fare**: Tarifa del pasajero para subir a bordo. Esta variable, aunque el sentido común indica que debería ser un float, R lo lee como un **numeric**
- **Cabin**: Identificador de la cabina del pasajero. Esta variable es de tipo **varchar**.
- **Embarked**: Código de embarque del pasajero. Esta variable tiene los siguientes tres valores posibles; 'C' = Cherbourg, 'Q' = Queenstown, 'S' = Southampton. La variable es de tipo **varchar**.

2.1 Factorización de las variables

A continuación, con el fin de tener el conjunto de datos preparado para la limpieza y análisis del mismo, se analiza qué variables tendría sentido factorizar.

```
apply(titanic_file,2, function(x) length(unique(x)))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##        1309           3           3        1307           2          99
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           7           8         929         282         187           4
```

La factorización tiene sentido realizarla sobre variables que tienen pocas clases, como son en este caso las variables 'Survived', 'Pclass', 'Sex' o 'Embarked'. Podemos observar que las variables survived y embarked tienen una clase más que las mencionadas en en análisis del dataset. Esto es debido a que, seguramente, tenga algún registro vacío o nulo. Es necesario realizar un pre-análisis de limpieza del conjunto de datos antes de proceder con el modelo. Este apartado se realizará más adelante. Dicho lo cual, se procede a factorizar las variables mencionadas:

```
# Discretizamos las variables con pocas clases
cols<-c("Survived","Pclass","Sex","Embarked")
for (i in cols){
  titanic_file[,i] <- as.factor(titanic_file[,i])
}

# Después de los cambios, analizamos la nueva estructura del conjunto de datos
str(titanic_file)
```

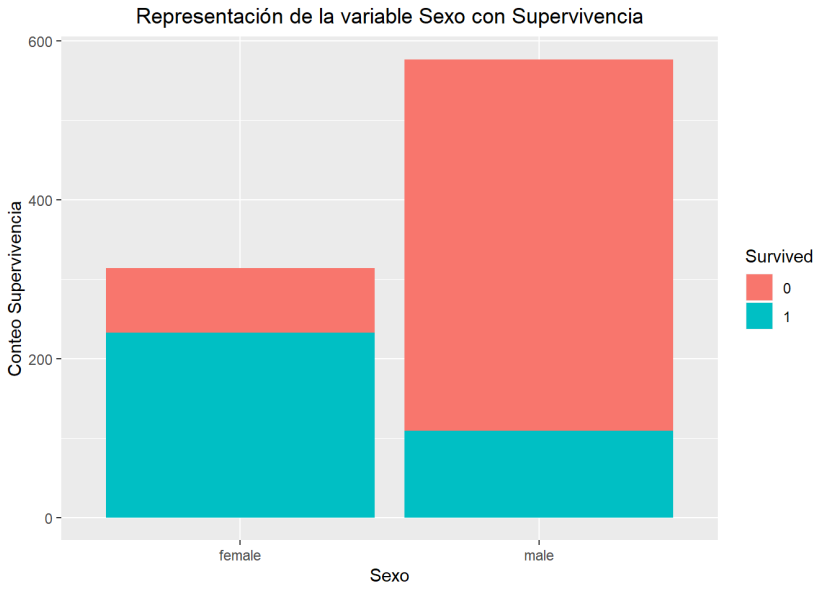
```
## 'data.frame':    1309 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
"Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
##  $ 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     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

Se observa que las cuatro variables se han factorizado correctamente.

2.2 Estudio del fichero

A continuación, con el fin de ampliar información sobre el conjunto de datos del fichero, se procede a hacer una representación de los mismos en torno a nuestra variable objetivo, la supervivencia al accidente, variable 'Survived'. No obstante, esta representación se hace sobre fichero *train.csv*, ya que el fichero test no tiene la variable objetivo representada puesto que es la variable que se quiere predecir.

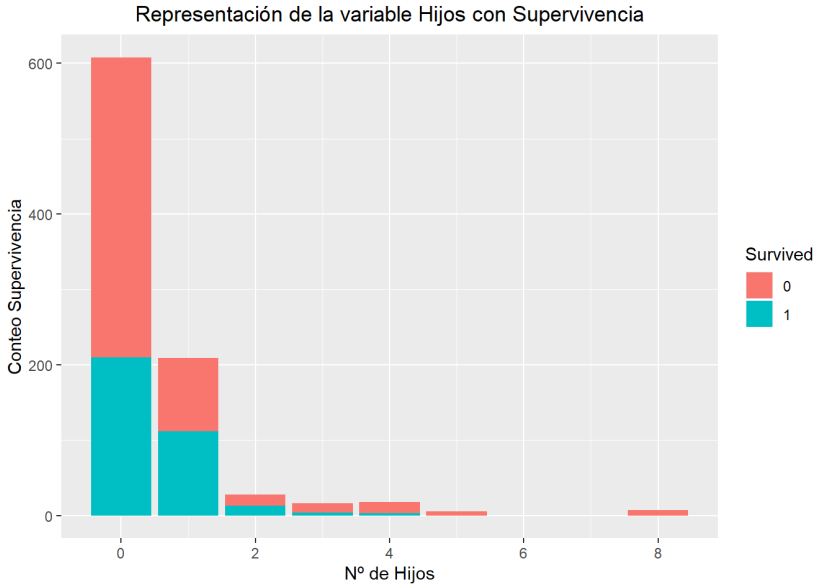
```
filas=dim(train)[1]
ggplot(data=titanic_file[1:filas,],aes(x=Sex,fill=Survived))+geom_bar() + ggtitle("Representación de la v
ariable Sexo con Supervivencia") + xlab("Sexo") + ylab("Conteo Supervivencia") + theme(plot.title =
element_text(hjust = 0.5))
```



Podemos observar que en el crucero hubo casi el doble de chicos que de chicas, aunque sobrevivieron más del doble de mujeres que de hombres.

A continuación se representa el número de hijos de cada familia en relación a la supervivencia:

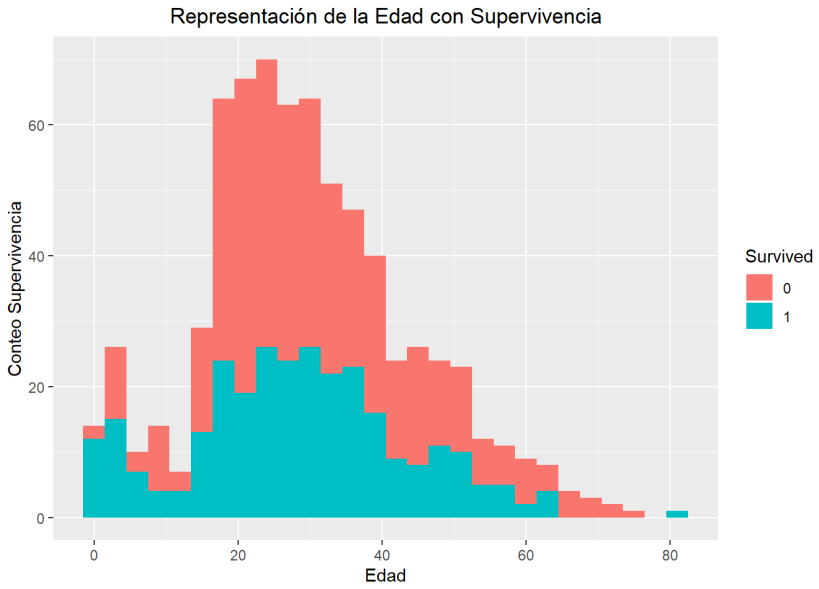
```
# Survivial como función de SibSp
ggplot(data = titanic_file[1:filas,],aes(x=SibSp,fill=Survived))+geom_bar() + ggtitle("Representación de
la variable Hijos con Supervivencia") + xlab("N° de Hijos") + ylab("Conteo Supervivencia") +
theme(plot.title = element_text(hjust = 0.5))
```



Podemos observar que la gran mayoría de familias que sobrevivieron tenían, de media, menos de dos hijos. Es interesante crear una nueva variable en la que sume el total de miembros en la familia ('SibSp' + 'Parch') para entrenar nuestro modelo. Esto se hará más adelante.

Por último, se muestra la distribucción de la edad de las personas en función de la supervivencia:

```
# Survival como función de age:
titanic_file1<-titanic_file[1:filas,]

ggplot(data = titanic_file1[!
(is.na(titanic_file[1:filas,]$Age)),],aes(x=Age,fill=Survived))+geom_histogram(binwidth =3) +
ggtitle("Representación de la Edad con Supervivencia") + xlab("Edad") + ylab("Conteo Supervivencia") +
theme(plot.title = element_text(hjust = 0.5))
```



Podemos observar que la gran mayoría de las personas que sobrevivieron al accidente eran menores de 40 años.

2.3 Importancia y objetivos de los análisis

Tras este primer análisis del dataset, la siguiente pregunta que debemos hacernos es por qué es importante este conjunto de datos, y qué problema pretende responder.

A partir de este conjunto de datos se plantea la problemática de determinar qué variables influyen más sobre la posibilidad de sobrevivir a un accidente como el Titanic, como pueden ser las variables sexo, clase o edad. Además, se podrá proceder a crear modelos de aprendizaje (regresión logarítmica o árbol de decisión) que permitan predecir si un pasajero ha sobrevivido o no en función de sus características y contrastes de hipótesis. Esto ayudará a identificar propiedades interesantes en las muestras que puedan ser inferidas con respecto a la población.

Estos análisis adquieren una gran relevancia ya que nos permite conocer cómo eran las clases sociales de hace 100 años, se puede estudiar su correlación, y evitar así posibles inconsistencias entre clases sociales para los futuros cruceros o eventos, como conseguir que todos tengan la misma probabilidad de sobrevivir independientemente de la situación del pasajero o del dinero que este tenga.

3 Integración y selección de los datos

A lo largo de este apartado se dará respuesta a las variables que son interesantes analizar para encontrar la posibilidad de que un pasajero sobreviva al accidente. Aunque la selección de datos es una tarea que se realizará con más detalle en el apartado de Análisis de datos debido a su complejidad, podemos adelantar que algunas de las variables que se pueden dejar fuera son todas aquellas que, a priori, no tiene relación con la variable objetivo (sobrevivir al accidente), como podrían ser 'PassengerId', 'Name, 'Cabin' o 'Ticket'. Estas variables, al ser identificadores, no ayudan al modelo. Dicho con otras palabras, no habrá ninguna relación entre estas variables y la variable objetivo.

Por otro lado, las variables que sí se analizarán son:

- 'Survived' para conocer si el pasajero sobrevivió o no. Esta variable es nuestra variable objetivo a predecir.
- A su vez, será interesante conocer si las variables 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare' o 'Embarked' tiene correlación con la variable 'Survived'.

4 Limpieza de los datos

Se tratará en este apartado el asunto de la limpieza del conjunto de datos, y nos enfocaremos en dos grandes puntos: la gestión de datos nulos o vacíos (o missing values) y la identificación y tratamiento de valores extremos (o outliers).

4.1 Gestión de datos nulos o vacíos

En primer lugar, identificamos aquellos atributos que tengan valores vacíos. Se seguirá una estrategia individualizada teniendo en cuenta el tipo de atributo del que se trate.

```
colSums(is.na(titanic_file))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           1           0           0
```

```
colSums(titanic_file=="")
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0          NA           0           0           0          NA
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0          NA        1014           2
```

Observamos dos tipos de missing values: valores con NA y valores con cadenas vacías. A efectos prácticos, serán tratados de igual forma unos y otros.

Antes de comenzar, analizaremos también la existencia de ceros en las variables continuas, y nos plantearemos si tiene sentido que existan o si

debemos tratarlos como valores vacíos. Estudiaremos las variables Age y Fare.

```
length(titanic_file[which(titanic_file$Fare == 0),]$Fare)
```

```
## [1] 15
```

```
length(titanic_file[which(titanic_file$Age == 0),]$Age)
```

```
## [1] 0
```

Encontramos ceros en la variable Fare. A priori, es una variable en la que no tiene sentido la existencia de ceros. Es esperable que todos los pasajeros hayan pagado un precio por su billete, por lo que consideramos que estos datos deberán tratarse como valores vacíos y serán incluidos en el tratamiento.

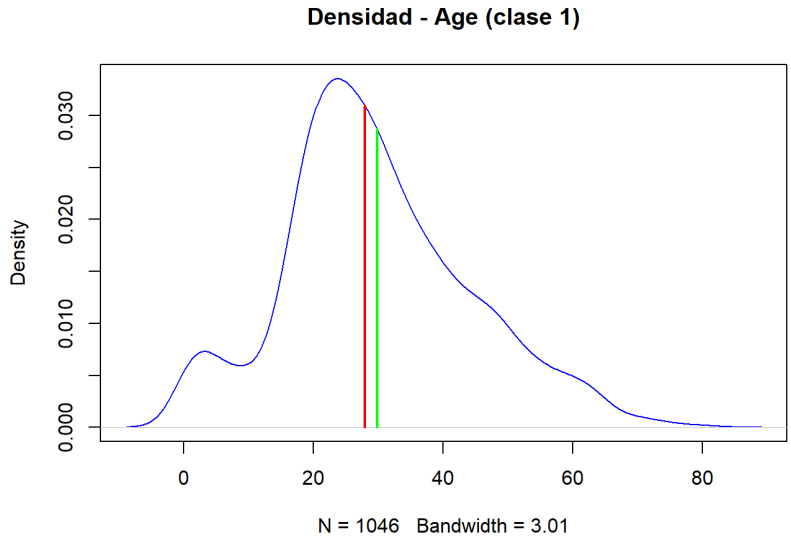
```
titanic_file[which(titanic_file$Fare == 0), "Fare"] = NA
```

Para empezar, como ya comentábamos, la variable Cabin no aporta conocimiento de las personas que sobrevivieron. Es una cadena de caracteres que ni siquiera tiene categorías o valores predeterminados, por lo que no es interesante para el análisis. Eliminamos dicha variable.

```
titanic_file$Cabin <- NULL
```

La variable de la edad contiene 263 valores vacíos. En principio, imputar los missing values de esta variable usando la media del resto de edades podría ser una buena estrategia. Observamos la distribución de esta variable para determinar si es así. Si la distribución es normal y centrada en la media, utilizaremos este método. Si la distribución estuviese sesgada hacia algún extremo, encontraremos otra manera.

```
dens <- density(titanic_file[!is.na(titanic_file$Age), "Age"])

n <- length(dens$y)                          #$
dx <- mean(diff(dens$x))                     # Typical spacing in x $
y.unit <- sum(dens$y) * dx                   # Check: this should integrate to 1 $
dx <- dx / y.unit                            # Make a minor adjustment
x.mean <- sum(dens$y * dens$x) * dx
y.mean <- dens$y[length(dens$x[dens$x < x.mean])] #$
x.mode <- dens$x[i.mode <- which.max(dens$y)]
y.mode <- dens$y[i.mode]                     #$
y.cs <- cumsum(dens$y)                       #$
x.med <- dens$x[i.med <- length(y.cs[2*y.cs <= y.cs[n]])] #$
y.med <- dens$y[i.med]

plot(dens, col = "blue", main="Densidad - Age (clase 1)")
mapply(function(x,y,c) lines(c(x,x), c(0,y), lwd=2, col=c), c(x.mean, x.med), c(y.mean, y.med), c("green",
"red"))
```



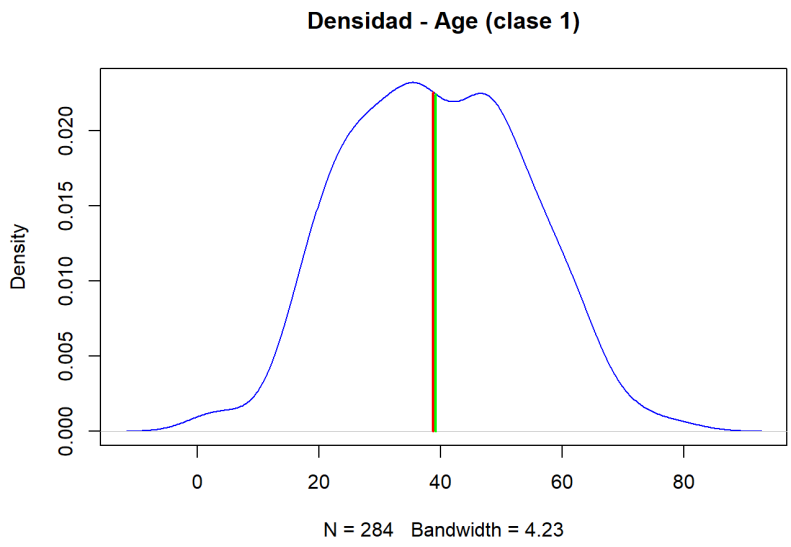

```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

Como se puede apreciar, la función de densidad está ligeramente escorada (o sesgada) hacia la izquierda. La línea roja representa la mediana, y la verde la media. Este escoramiento hace que la media se desplace hacia valores mayores. Esto puede indicar que existen valores extremos o, simplemente, que existen valores altos que encontramos en menor densidad. REcordemos que la media es una medida sensible a este tipo de valores, y por tanto se desplaza. Aunque no es demasiado acusado, quezá sea mejor utilizar la mediana para imputar los valores faltantes.

Dado que se trata de un porcentaje considerable de missing values (263 / 1309), podemos afinar algo más el análisis para imputar los valores de forma más específica. Dividiremos la muestra en las tres clases distintas en las que puede viajar un pasajero. El razonamiento detrás de esto es

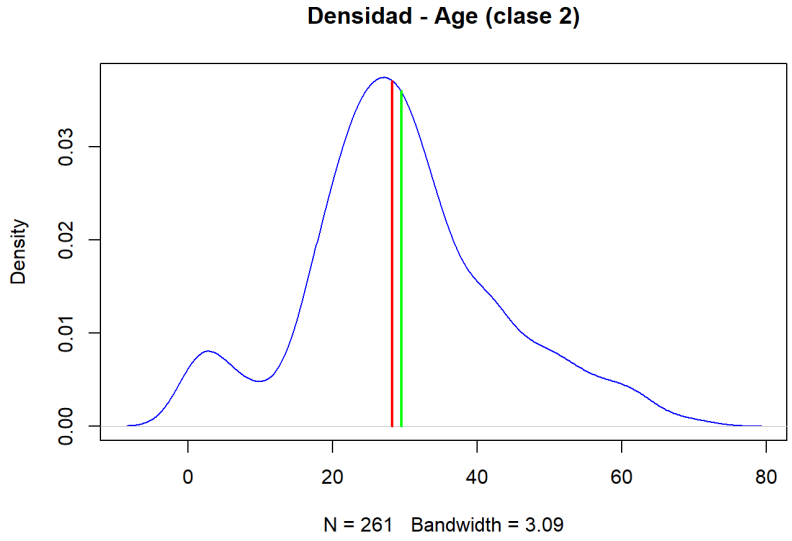
que es de esperar que la edad sea mayor en las mejores clases (la gente mayor suele viajar con más comodidades que la gente joven). Por tanto, es fácil imaginarse que el atributo de la clase puede tener que ver con la edad del viajero. Vemos a continuación el detalle.

```
# Clase 1

dens <- density(titanic_file[!is.na(titanic_file$Age) & titanic_file$Pclass == 1, "Age"])

n <- length(dens$y)                          #$
dx <- mean(diff(dens$x))                     # Typical spacing in x $
y.unit <- sum(dens$y) * dx                   # Check: this should integrate to 1 $
dx <- dx / y.unit                            # Make a minor adjustment
x.mean <- sum(dens$y * dens$x) * dx
y.mean <- dens$y[length(dens$x[dens$x < x.mean])] #$
x.mode <- dens$x[i.mode <- which.max(dens$y)]
y.mode <- dens$y[i.mode]                     #$
y.cs <- cumsum(dens$y)                       #$
x.med <- dens$x[i.med <- length(y.cs[2*y.cs <= y.cs[n]])] #$
y.med <- dens$y[i.med]

plot(dens, col = "blue", main="Densidad - Age (clase 1)")
mapply(function(x,y,c) lines(c(x,x), c(0,y), lwd=2, col=c), c(x.mean, x.med), c(y.mean, y.med), c("green",
"red"))
```



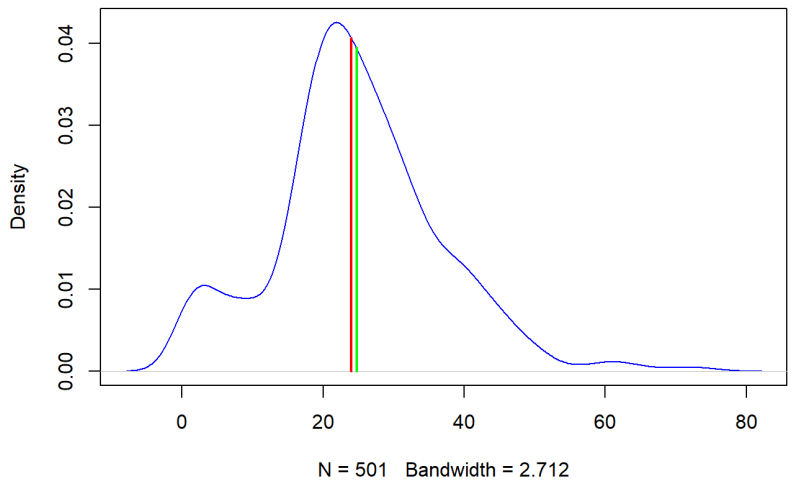

```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

```
# Clase 2

dens <- density(titanic_file[!is.na(titanic_file$Age) & titanic_file$Pclass == 2, "Age"])

n <- length(dens$y)                          #$
dx <- mean(diff(dens$x))                     # Typical spacing in x $
y.unit <- sum(dens$y) * dx                   # Check: this should integrate to 1 $
dx <- dx / y.unit                            # Make a minor adjustment
x.mean <- sum(dens$y * dens$x) * dx
y.mean <- dens$y[length(dens$x[dens$x < x.mean])] #$
x.mode <- dens$x[i.mode <- which.max(dens$y)]
y.mode <- dens$y[i.mode]                     #$
y.cs <- cumsum(dens$y)                       #$
x.med <- dens$x[i.med <- length(y.cs[2*y.cs <= y.cs[n]])] #$
y.med <- dens$y[i.med]

plot(dens, col = "blue", main="Densidad - Age (clase 2)")
mapply(function(x,y,c) lines(c(x,x), c(0,y), lwd=2, col=c), c(x.mean, x.med), c(y.mean, y.med), c("green",
"red"))
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

```
# Clase 3

dens <- density(titanic_file[!is.na(titanic_file$Age) & titanic_file$Pclass == 3, "Age"])

n <- length(dens$y)                          #$
dx <- mean(diff(dens$x))                     # Typical spacing in x $
y.unit <- sum(dens$y) * dx                   # Check: this should integrate to 1 $
dx <- dx / y.unit                            # Make a minor adjustment
x.mean <- sum(dens$y * dens$x) * dx
y.mean <- dens$y[length(dens$x[dens$x < x.mean])] #$
x.mode <- dens$x[i.mode <- which.max(dens$y)]
y.mode <- dens$y[i.mode]                     #$
y.cs <- cumsum(dens$y)                       #$
x.med <- dens$x[i.med <- length(y.cs[2*y.cs <= y.cs[n]])] #$
y.med <- dens$y[i.med]

plot(dens, col = "blue", main="Densidad - Age (clase 3)")
mapply(function(x,y,c) lines(c(x,x), c(0,y), lwd=2, col=c), c(x.mean, x.med), c(y.mean, y.med), c("green",
"red"))
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

Vemos que, para la clase 1, la distribución se centra más en la media, que es cercana a los 40 años. Esto puede ser debido a que no exista mayor concentración de edades bajas que de edades altas como pasa con la muestra total. Es decir, se cumple lo que pronosticábamos: en mejores clases, la edad suele ser más alta. Confirmamos esto viendo que la media y la mediana coinciden prácticamente.

Para el caso de la clase 2, también encontramos una distribución más centrada. La media se sitúa algo por debajo de los 30 años, y la mediana con un valor ligeramente menor. De nuevo, esta observación confirma la teoría.

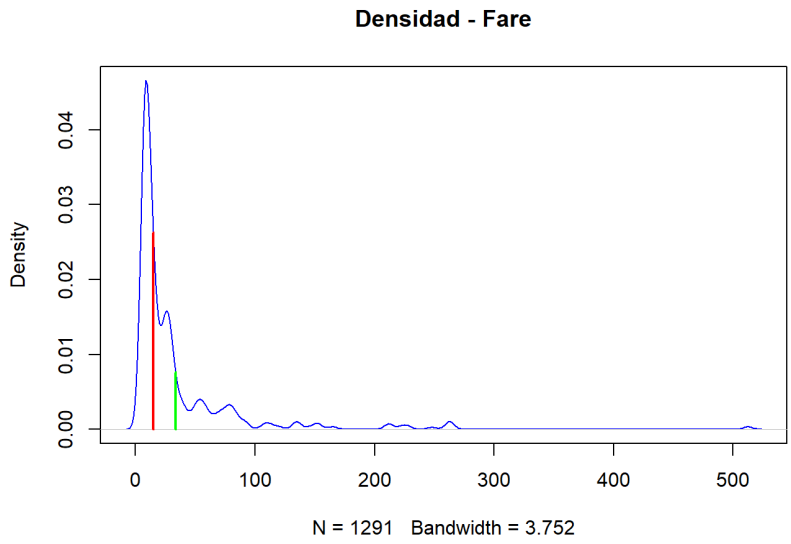
Por último, nos encontramos con una densidad para la clase 3 más escorada, y con una media y medianas por debajo de 25 años.

Como sospechábamos, la edad del viajero sí que parece guardar relación con la clase en la que viaje. Utilizaremos las medianas de cada submuestra para imputar los valores nulos de sus observaciones.

```
titanic_file[is.na(titanic_file$Age) & titanic_file$Pclass == 1, "Age"] =
median(titanic_file[!is.na(titanic_file$Age) & titanic_file$Pclass == 1, "Age"])
titanic_file[is.na(titanic_file$Age) & titanic_file$Pclass == 2, "Age"] =
median(titanic_file[!is.na(titanic_file$Age) & titanic_file$Pclass == 2, "Age"])
titanic_file[is.na(titanic_file$Age) & titanic_file$Pclass == 3, "Age"] =
median(titanic_file[!is.na(titanic_file$Age) & titanic_file$Pclass == 3, "Age"])
```

Para la variable Fare, hacemos el mismo análisis para ver su distribución y determinar si la media o la mediana pueden ser buenos valores para sustituir el valor faltante.

```
dens <- density(titanic_file[!is.na(titanic_file$Fare), "Fare"])

n <- length(dens$y)                          #$
dx <- mean(diff(dens$x))                     # Typical spacing in x $
y.unit <- sum(dens$y) * dx                   # Check: this should integrate to 1 $
dx <- dx / y.unit                            # Make a minor adjustment
x.mean <- sum(dens$y * dens$x) * dx
y.mean <- dens$y[length(dens$x[dens$x < x.mean])] #$
x.mode <- dens$x[i.mode <- which.max(dens$y)]
y.mode <- dens$y[i.mode]                     #$
y.cs <- cumsum(dens$y)                       #$
x.med <- dens$x[i.med <- length(y.cs[2*y.cs <= y.cs[n]])] #$
y.med <- dens$y[i.med]

plot(dens, col = "blue", main="Densidad - Fare")
mapply(function(x,y,c) lines(c(x,x), c(0,y), lwd=2, col=c), c(x.mean, x.med), c(y.mean, y.med), c("green",
"red"))
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

Como se puede observar, la variable Fare está lejos de distribuirse de forma normal. Usar medidas estadísticas como la media o la mediana no sería la estrategia más acertada. Es de esperar que el valor de esta variable esté relacionado con otras de las del conjunto de datos, como puede ser la clase en la que se viaja, el puerto en el que se embarca, la edad, etc. Es por eso que lo más adecuado para imputar esta variable sería utilizar el algoritmo de vecinos próximos. Seleccionaremos las 10 observaciones más “próximas” en función de la distancia entre las variables observadas y usaremos el valor de dichas observaciones (media para variables continuas) para imputar nuestros missing values. Usaremos la función kNN del paquete VIM de R con \(k = 10\).

```
library(VIM)

mv <- titanic_file[, c("Survived", "Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked")]

aux_df <- kNN(mv, variable = c("Fare"), k = 10, dist_var = c("Pclass", "Sex", "Age", "SibSp", "Parch", "Su
rvived", "Embarked"))

titanic_file$Fare <- aux_df$Fare
```

Encontramos dos valores vacíos para la variable Embarked. Al ser números muy reducidos y tratarse de una variable categórica, podemos utilizar la estrategia de imputarlos con la moda (valor que más aparece) sin preocuparnos de introducir sesgos en el conjunto de datos.

```
uv <- unique(titanic_file$Embarked)
titanic_file[is.na(titanic_file$Embarked) | titanic_file$Embarked == "", "Embarked"] =
uv[which.max(tabulate(match(titanic_file$Embarked, uv)))]
```

En el caso de la variable Survived, quizá no tenga sentido imputar estas observaciones vacías, ya que se trata de la variable objetivo. Son precisamente estos valores faltantes los que trataremos de predecir cuando construyamos un modelo sobre este conjunto de datos.

Para terminar, observamos de nuevo la cantidad de valores vacíos. Si todo ha ido bien, ya no deberíamos ver ninguno (a excepción de la variable Survived, naturalmente).

```
colSums(is.na(titanic_file))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0           0
##       SibSp       Parch      Ticket        Fare    Embarked
##           0           0           0           0           0
```

```
colSums(titanic_file=="")
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0          NA           0           0           0           0
##       SibSp       Parch      Ticket        Fare    Embarked
##           0           0           0           0           0
```

4.2 Identificación y tratamiento de valores extremos

Los valores extremos, outliers o valores fuera de rangos pueden ser identificados de distinta forma en función de la tipología de la variable analizada.

En primer lugar, identificaremos valores fuera de rango en variables categóricas. Se trata de categorías no válidas, distintas maneras de codificar una categoría, etc. Extraeremos los valores únicos de dichas variables para ver si existen valores que haya que corregir.

```
titanic_file %>% group_by(Survived) %>% count()
```

```
## # A tibble: 3 x 2
## # Groups:   Survived [3]
##   Survived     n
##   <fct>    <int>
## 1 0          549
## 2 1          342
## 3 <NA>       418
```

```
titanic_file %>% group_by(Pclass) %>% count()
```

```
## # A tibble: 3 x 2
## # Groups:   Pclass [3]
##   Pclass     n
##   <fct>  <int>
## 1 1        323
## 2 2        277
## 3 3        709
```

```
titanic_file %>% group_by(SibSp) %>% count()
```

```
## # A tibble: 7 x 2
## # Groups:   SibSp [7]
##   SibSp     n
##   <int> <int>
## 1     0   891
## 2     1   319
## 3     2    42
## 4     3    20
## 5     4    22
## 6     5     6
## 7     8     9
```

```
titanic_file %>% group_by(Parch) %>% count()
```

```
## # A tibble: 8 x 2
## # Groups:   Parch [8]
##   Parch     n
##   <int> <int>
## 1     0  1002
## 2     1   170
## 3     2   113
## 4     3     8
## 5     4     6
## 6     5     6
## 7     6     2
## 8     9     2
```

```
titanic_file %>% group_by(Embarked) %>% count()
```

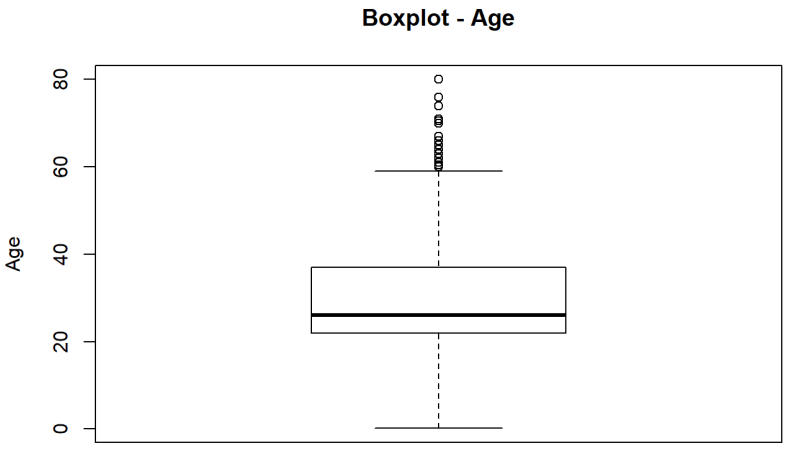
```
## # A tibble: 3 x 2
## # Groups:   Embarked [3]
##   Embarked     n
##   <fct>    <int>
## 1 C          270
## 2 Q          123
## 3 S          916
```

Como vemos, existen pocas sospechas de que algún valor esté fuera de rango. **(COMO MUCHO VER SI AGRUPAMOS LOS 3, 4, 5, 6 Y 9 DE PARCH, 5 Y 6 DE SIBSP, ETC).**

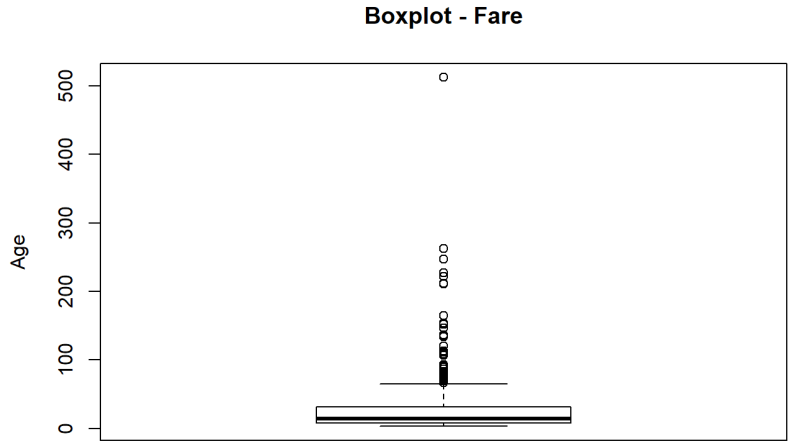
Para las variables numéricas, usaremos la convención de identificar valores extremos como aquellos que se desvíen más de 1.5 IQR (rango intercuartiles) del primer y tercer cuartil respectivamente. En caso de encontrar algún valor atípico, se aplicará un floor y un cap (valores máximos y mínimos aceptados) que serán, respectivamente, \(floor = Q1 - 1.5*IQR\) y \(cap = Q3 + 1.5*IQR\). Identificaremos estos outliers de forma visual con un diagrama de tipo boxplot, y también realizaremos el cálculo para mayor seguridad. Otra opción podría ser la de normalizar la variable de manera que se distribuya siguiendo una normal \(N(0, 1)\). De esta manera, también se eliminarían los outliers.

Sin embargo, discutiremos si los outliers encontrados son valores realmente erróneos o si por el contrario se encuentran en el rango esperable para cada variable. Por ejemplo, sería evidente que una edad de 150 años es un outlier y habría que transformarlo. Pero quizá una edad de 90 no lo sea, y aún así esté siendo identificado como outlier por nuestro método al existir pocas observaciones de personas con 90 años.

```
boxplot(titanic_file$Age, main = "Boxplot - Age", ylab = "Age")
```




```
boxplot(titanic_file$Fare, main = "Boxplot - Fare", ylab = "Age")
```



Como se puede observar, en ambas variables encontramos valores extremos por encima de la media. Vemos a continuación las listas de dichos valores. Además, si fuera procedente, realizaríamos la transformación cap y floor correspondiente en cada caso.

```
IQR_age = quantile(titanic_file$Age)[4] - quantile(titanic_file$Age)[2]
IQR_fare = quantile(titanic_file$Fare)[4] - quantile(titanic_file$Fare)[2]

floor_age = quantile(titanic_file$Age)[2] - (1.5 * IQR_age)
floor_fare = quantile(titanic_file$Fare)[2] - (1.5 * IQR_fare)

cap_age = quantile(titanic_file$Age)[4] + (1.5 * IQR_age)
cap_fare = quantile(titanic_file$Fare)[4] + (1.5 * IQR_fare)

sort(titanic_file[titanic_file$Fare < floor_fare | titanic_file$Fare > cap_fare, "Fare"])
```

```
##   [1]  66.6000  66.6000  69.3000  69.3000  69.5500  69.5500  69.5500  69.5500
##   [9]  69.5500  69.5500  69.5500  69.5500  69.5500  69.5500  69.5500  71.0000
##  [17]  71.0000  71.2833  71.2833  73.5000  73.5000  73.5000  73.5000  73.5000
##  [25]  73.5000  73.5000  75.2417  75.2417  75.2500  75.2500  76.2917  76.2917
##  [33]  76.7292  76.7292  76.7292  77.2875  77.2875  77.9583  77.9583  77.9583
##  [41]  78.2667  78.2667  78.8500  78.8500  78.8500  79.2000  79.2000  79.2000
##  [49]  79.2000  79.2000  79.2000  79.6500  79.6500  79.6500  80.0000  80.0000
##  [57]  81.8583  81.8583  81.8583  82.1708  82.1708  82.2667  82.2667  83.1583
##  [65]  83.1583  83.1583  83.1583  83.1583  83.1583  83.4750  83.4750  86.5000
##  [73]  86.5000  86.5000  89.1042  89.1042  90.0000  90.0000  90.0000  90.0000
##  [81]  90.0000  91.0792  91.0792  93.5000  93.5000  93.5000  93.5000 106.4250
##  [89] 106.4250 106.4250 108.9000 108.9000 108.9000 110.8833 110.8833 110.8833
##  [97] 110.8833 113.2750 113.2750 113.2750 120.0000 120.0000 120.0000 120.0000
## [105] 133.6500 133.6500 134.5000 134.5000 134.5000 134.5000 134.5000 135.6333
## [113] 135.6333 135.6333 135.6333 136.7792 136.7792 146.5208 146.5208 146.5208
## [121] 151.5500 151.5500 151.5500 151.5500 151.5500 151.5500 153.4625 153.4625
## [129] 153.4625 164.8667 164.8667 164.8667 164.8667 211.3375 211.3375 211.3375
## [137] 211.3375 211.5000 211.5000 211.5000 211.5000 211.5000 221.7792 221.7792
## [145] 221.7792 221.7792 227.5250 227.5250 227.5250 227.5250 227.5250 247.5208
## [153] 247.5208 247.5208 262.3750 262.3750 262.3750 262.3750 262.3750 262.3750
## [161] 262.3750 263.0000 263.0000 263.0000 263.0000 263.0000 263.0000 512.3292
## [169] 512.3292 512.3292 512.3292
```

```
sort(titanic_file[titanic_file$Age < floor_age | titanic_file$Age > cap_age, "Age"])
```

```
##  [1] 60.0 60.0 60.0 60.0 60.0 60.0 60.0 60.5 61.0 61.0 61.0 61.0 61.0 62.0 62.0
## [16] 62.0 62.0 62.0 63.0 63.0 63.0 63.0 64.0 64.0 64.0 64.0 64.0 65.0 65.0 65.0
## [31] 66.0 67.0 70.0 70.0 70.5 71.0 71.0 74.0 76.0 80.0
```

En las listas de datos proporcionadas (ordenadas de mayor a menor), se puede apreciar que el valor máximo para la edad es de 80 años, lo cual es más que posible que se trate de un valor correcto, y que por tanto no debemos transformar. Siguiendo el mismo razonamiento, el resto de valores también permanecerán intactos.

Por parte de la variable Fare, encontramos varios valores inusualmente altos. Se trata de un precio de billete por encima de los 500 dólares. Es sospechoso, pero encontramos 4 observaciones distintas con exactamente el mismo precio. Esto nos sugiere que perfectamente se puede tratar de un billete de lujo. De ser así, debería tratarse de un viajero de primera clase. Es una comprobación muy superficial, pero nos indicará si nuestra hipótesis va por buen camino o no.

```
titanic_file[titanic_file$Fare == max(titanic_file$Fare), c("Pclass", "Fare")]
```

```
##      Pclass     Fare
## 259       1 512.3292
## 680       1 512.3292
## 738       1 512.3292
## 1235      1 512.3292
```

Como sospechábamos, se trata de viajeros de primera clase. Esto sumado a que los cuatro contienen el mismo valor en la variable Fare nos lleva a pensar que es un valor correcto, por lo que no deberemos transformarlo tampoco.

5 Análisis de los datos

A lo largo de este apartado se hará un análisis del conjunto de datos con el fin de obtener un modelo para predecir si una persona puede sobrevivir o no a un accidente como el Titanic dadas unas variables. En primer lugar, se seleccionarán los grupos de interés, seguido de un análisis de normalidad y homogeneidad. Por último, se harán unas pruebas estadísticas para obtener el modelo que mejor se adapte al conjunto de datos.

No obstante, puesto que el análisis se va a realizar con el conjunto de datos cuya variable Survived está informada para obtener mejores resultados en el modelo, se procede a modificar el set:

```
titanic_file_train <- titanic_file[!is.na(titanic_file$Survived),]
```

5.1 Selección de los grupos de datos de interés

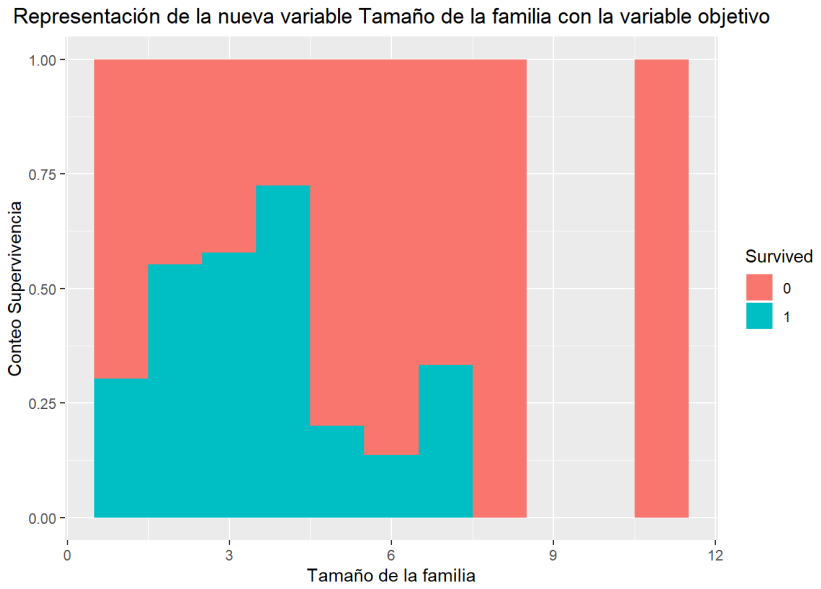
En primer lugar, se seleccionan las variables que son interesantes dejar en el modelo para conocer si una persona ha sobrevivido al accidente. Para empezar, podemos eliminar aquellas que, claramente, no son relevantes para el modelo, como son las variables 'PassengerID', 'Name', 'Ticket' o 'Cabin'. Estas variables, por su definición, no representan nada para el modelo. Es decir, aunque entrenemos al modelo con estas variables, no va a ver una mejoría, sino que seguramente empeore el modelo haciéndolo más complejo.

```
# Se eliminan las variables
titanic_file_train$PassengerId <- NULL
titanic_file_train$Name <- NULL
titanic_file_train$Ticket <- NULL
titanic_file_train$Cabin <- NULL

# Se comprueba que no aparecen en el fichero
str(titanic_file_train)
```

```
## 'data.frame':    891 obs. of  8 variables:
##  $ Survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ 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     : num  22 38 26 35 35 24 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 ...
##  $ Fare    : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked: Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

No obstante, es necesario realizar un análisis con el resto de variables para conocer si merece la pena eliminarlas o no. Sin embargo, antes de hacer este análisis, vemos conveniente crear una nueva variable que represente el tamaño de la familia haciendo la suma entre las variables 'SibSp', número de hermanos, y 'Parch', número de padres o hijos. Esta nueva variable se llamará **FamilySize**.

```
# Construimos un atributo nuevo: family size.
titanic_file_train$FamilySize <- titanic_file_train$SibSp + titanic_file_train$Parch +1

# Observamos su distribucción
titanic_file1 <- titanic_file_train[1:filas,]
ggplot(data =
titanic_file1[!is.na(titanic_file_train[1:filas,]$FamilySize),],aes(x=FamilySize,fill=Survived))+geom_histo
ram(binwidth =1,position="fill")+ylab("Frecuencia") + ggtitle("Representación de la nueva variable Tamaño
de la familia con la variable objetivo") + xlab("Tamaño de la familia") + ylab("Conteo Supervivencia") +
theme(plot.title = element_text(hjust = 0.5))
```



Podemos sacar como conclusión que aquellas familias compuestas por menos integrantes sobrevivieron más, en porcentaje, que aquellas con más de 5 miembros.

Tras la construcción de esta nueva variable, procedemos a realizar el análisis comentado para ver qué variables son representativas en el modelo. Este análisis es diferente en función del tipo de la variable, utilizando:

- **Correlación** para aquellas variables que numéricas, como 'Age', 'Fare', 'SibSp', 'Parch', o la nueva variable 'FamilySize'. Estas variables, aunque sean de tipo integer, se corresponden con variables numéricas por lo que será necesario convertirlas al tipo de dato correcto.
- **Test Chi Cuadrado** para aquellas variables que son categóricas, 'Pclass', 'Sex' o 'Embarked'

5.1.1 Correlación

Así pues, se procede a realizar un análisis de correlación entre las distintas variables para determinar cuáles de ellas ejercen una mayor influencia sobre la variable objetivo.

Tal y como se ha comentado, el primer paso es modificar las variables del número de hermanos y de hijos/padres para que sea del tipo numérico.

```
# Se convierten las variables al tipo numérico
titanic_file_train$SibSp <- as.numeric(titanic_file_train$SibSp)
titanic_file_train$Parch <- as.numeric(titanic_file_train$Parch)

# Se comprueba que el tipo de dato es correcto.
str(titanic_file_train)
```

```
## 'data.frame':    891 obs. of  9 variables:
##  $ Survived  : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ 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       : num  22 38 26 35 35 24 54 2 27 14 ...
##  $ SibSp     : num  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch     : num  0 0 0 0 0 0 0 1 2 0 ...
##  $ Fare      : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked  : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ FamilySize: num  2 2 1 2 1 1 1 5 3 2 ...
```

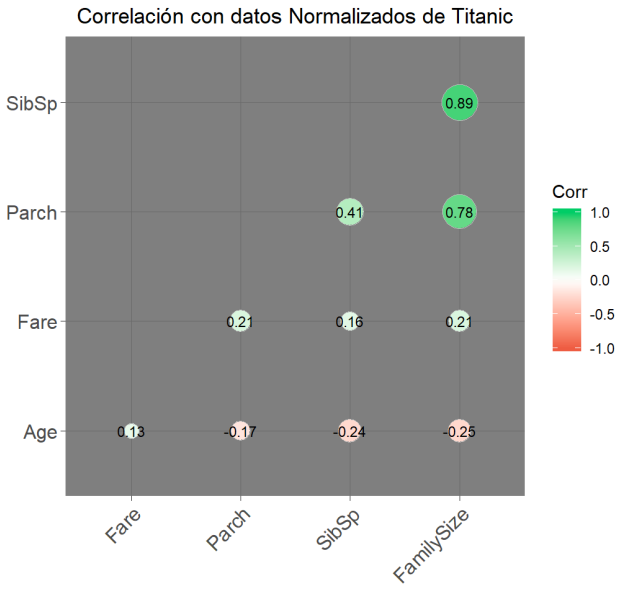
El siguiente paso será analizar dichas variables con una correlación. Para ello se obtiene solamente las variables mencionadas:

```
titanic_corr <- titanic_file_train %>% select('Age', 'Fare', 'SibSp', 'Parch', 'FamilySize')

# Pequeña comprobación de que la copia se ha hecho correctamente
head(titanic_corr, 2)
```

```
##   Age    Fare SibSp Parch FamilySize
## 1  22  7.2500     1     0          2
## 2  38 71.2833     1     0          2
```

Dicho lo cual, se procede a analizar cuál es la correlación entre dichas variables.

```
library(ggcorrplot)
corr_norm <- cor(titanic_corr)

ggcorrplot(corr_norm, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato2", "white", "springgreen3"),
           title="Correlación con datos Normalizados de Titanic",
           ggtheme=theme_dark) + theme(plot.title = element_text(hjust = 0.5))
```



Podemos observar que la correlación entre las variables suele ser bastante baja, a excepción de la nueva variable creada, FamilySize, que tiene una alta correlación con las variables 'SibSp' y 'Parch'. Así pues, con el fin de optimizar el modelo y no añadir variables con alta correlación, se eliminan las dos variables utilizadas para crear el Tamaño de la familia del fichero original.

```
# Se eliminan las variables
titanic_file_train$SibSp <- NULL
titanic_file_train$Parch <- NULL

# Se comprueba que no aparecen en el fichero
str(titanic_file_train)
```

```
## 'data.frame':    891 obs. of  7 variables:
##  $ Survived  : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ 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       : num  22 38 26 35 35 24 54 2 27 14 ...
##  $ Fare      : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked  : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ FamilySize: num  2 2 1 2 1 1 1 5 3 2 ...
```

5.1.2 Test Chi Cuadrado

Por último, se analiza si existe relación entre las variables categóricas con la variable objetivo, que son las variables 'Pclass', 'Sex' o 'Embarked'. Para ello, se utiliza el test Chi-cuadrado de Pearson. Un resultado significativo nos dirá que existe asociación entre ambas variables. Este apartado se realiza bajo la función *chisq.test*, la cual recibe por parámetro la tabla de contingencia entre la variable objetivo y la variable a estudiar.

Respecto al análisis de los resultados, se confirmará que las variables son dependientes y tienen relación entre sí siempre y cuando el valor del test sea inferior al valor significativo, el cual es de 0.05, y por tanto se rechaza la hipótesis nula de independencia. Esta hipótesis está definida de la siguiente manera:

H_{0}: La variable objetivo y la variable a analizar ('Pclass', 'Sex' o 'Embarked') son independientes
H_{1}: La variable objetivo y la variable a analizar ('Pclass', 'Sex' o 'Embarked') no son independientes

5.1.2.1 Variable objetivo con Pclass

Dicho lo cual, se genera la tabla de contingencia entre la variable objetivo y la primera variable a analizar, las clases sociales.

```
#install.packages("knitr") # Tabla [14]
library(knitr)
#install.packages("kableExtra") # Tabla [14]
library(kableExtra)

# Creamos la tabla de contingencia
tc_pclass <- table(titanic_file_train$Survived, titanic_file_train$Pclass)
# Le damos formato a la tabla de contingencia
tc_pclass %>% kable() %>% kable_styling(bootstrap_options = "striped", full_width = F)
```

	1	2	3
0	80	97	372
1	136	87	119

A simple vista, se puede confirmar que la gran mayoría de muertos al accidente del Titanic eran aquellos que pertenecían a la tercera clase. A continuación se realiza el Test con la tabla de contingencia:

```
chisq.test(tc_pclass)
```

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

Puesto que el p-valor es muy cercano a 0, se debe rechazar la hipótesis nuela y afirmar que existe relación entre las variables.

5.1.2.2 Variable objetivo con Sexo

De la misma manera, se realiza el mismo análisis entre la variable objetivo y la variable sexo:

```
# Creamos la tabla de contingencia
tc_sex <- table(titanic_file_train$Survived, titanic_file_train$Sex)
# Le damos formato a la tabla de contingencia
tc_sex %>% kable() %>% kable_styling(bootstrap_options = "striped", full_width = F)
```

	female	male
0	81	468
1	233	109

A simple vista, se puede confirmar que la gran mayoría de los muertos eran hombres, casi 4 veces más que las mujeres. A continuación se realiza el Test con la tabla de contingencia:

```
chisq.test(tc_sex)
```

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

Tal y como se ha dicho anteriormente, debido a que el valor de p es inferior al valor significativo de 0.05, debemos rechazar la hipótesis nula y afirmar que las variables son dependientes entre sí.

5.1.2.3 Variable objetivo con Embarked

Por último, toca analizar la relación entre la variable objetivo y la variable de embarcación. Primero, se realiza la tabla de contingencia:

```
# Creamos la tabla de contingencia
tc_embarked <- table(titanic_file_train$Survived, titanic_file_train$Embarked)
tc_embarked <- tc_embarked[,-1]

# Le damos formato a la tabla de contingencia
tc_embarked %>% kable() %>% kable_styling(bootstrap_options = "striped", full_width = F)
```

	C	Q	S
0	75	47	427
1	93	30	219

Una vez obtenida la tabla de contingencia, se ejecuta la función para conseguir el valor de Chi-Square, obteniendo un valor inferior al valor significativo de 0.05. Así pues, se debe rechazar la hipótesis nula y afirmar que las variables tienen relación entre sí.

```
chisq.test(tc_embarked)
```

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

5.1.2.4 Conclusiones del análisis de las variables

Tras el análisis de las variables realizado, llegamos a la conclusión de que las siguientes variables son interesantes utilizarlas para crear el modelo en base a nuestra variable objetivo, la supervivencia al accidente:

```
str(titanic_file_train)
```

```
## 'data.frame':    891 obs. of  7 variables:
##  $ Survived  : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ 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       : num  22 38 26 35 35 24 54 2 27 14 ...
##  $ Fare      : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked  : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ FamilySize: num  2 2 1 2 1 1 1 5 3 2 ...
```

5.2 Comprobación de la normalidad y homogeneidad de la varianza

A lo largo de este apartado se estudiará la normalidad y homogeneidad de las variables.

5.2.1 Normalidad

La comprobación de normalidad se realiza para aquellas variables que son numéricas, que en este caso son 'Age', 'Fare' y la nueva variable creada, 'FamilySize'.

Para la comprobación de que los valores que toman nuestras variables cuantitativas provienen de una población distribuida normalmente, utilizaremos la prueba de normalidad de Anderson-Darling. Así, se comprueba que para que cada prueba se obtiene un p-valor superior al nivel de significación prefijado \(\alpha = 0.05\). Si esto se cumple, entonces se considera que variable en cuestión sigue una distribución normal.

```
library(nortest)
alpha = 0.05
col.names = colnames(titanic_file_train)

for (i in 1:ncol(titanic_file_train)) {
  if (i == 1) cat("Variables que no siguen una distribución normal:\n")

  if (is.integer(titanic_file_train[,i]) | is.numeric(titanic_file_train[,i])) {
    p_val = ad.test(titanic_file_train[,i])$p.value
    if (p_val < alpha) {
      cat(col.names[i])

      # Format output
      if (i < ncol(titanic_file_train) - 1) cat(", ")
      if (i %% 3 == 0) cat("\n")
    }
  }
}
```

```
## Variables que no siguen una distribución normal:
## Age, Fare, FamilySize
```

Observamos que ninguna de las tres variables numéricas siguen una distribución normal.

5.2.2 Homogeneidad

Seguidamente, pasamos a estudiar la homogeneidad de varianzas mediante la aplicación de un test de Fligner-Killeen. Este test se aplica para las variables que no siguen una distritucción de normalidad, como son las tres variables numéricas que se ha analizado anteriormente. En el siguiente test, la hipótesis nula consiste en que ambas varianzas son diferentes, es decir:

\(H_{0}\): La variable objetivo y la variable a analizar tienen diferente varianza
\(H_{1}\): La variable objetivo y la variable a analizar tienen la misma varianza

Como siempre, se aceptará la hipótesis nula siempre y cuando el P-Valor sea inferior al valor significativo de 0.05.

Se ejecuta el test de Fligner-Killeen para todas las variables:

```
fligner.test(as.numeric(Survived) ~ Age, data = titanic_file_train)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  as.numeric(Survived) by Age
## Fligner-Killeen:med chi-squared = 70.86, df = 87, p-value = 0.8957
```

```
fligner.test(as.numeric(Survived) ~ Fare, data = titanic_file_train)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  as.numeric(Survived) by Fare
## Fligner-Killeen:med chi-squared = 259.98, df = 250, p-value = 0.3191
```

```
fligner.test(as.numeric(Survived) ~ FamilySize, data = titanic_file_train)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  as.numeric(Survived) by FamilySize
## Fligner-Killeen:med chi-squared = 27.095, df = 8, p-value = 0.0006806
```

Se puede observar que para las variables de Edad y Fare se obtiene un p-valor superior a 0.05 (Age ~ 0.9 y Fare ~ 0.32) por lo que se debe aceptar la hipótesis de que las varianzas de ambas muestras son homogéneas. Para el resto de variables, al ser el p-valor inferior al intervalo de confianza, no podemos decir que sean homogéneas.

5.2.3 Prueba estadística con Mann-Whitney

La segunda prueba estadística que se aplicará consistirá en un contraste de hipótesis sobre una muestra para determinar si la probabilidad de sobrevivir al accidente es mayor dependiendo de las variables.

Se debe destacar que un test paramétrico se debe utilizar cuando los datos siguen una distribucción normal, si la muestra es de tamaño inferior a 30. Como en nuestro caso esto no es así ya que la variable no sigue una distribucción normal ni de homogeneidad, se deben utilizar funciones no paramétricas, como podría ser Mann-Whitney. Así pues, se estudia la relación para las variables numéricas del modelo, 'Age' y 'Fare' (FamilySize se queda fuera puesto que sí es homocedástica).

```
wilcox.test(as.numeric(titanic_file_train$Survived), titanic_file_train$Age)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  as.numeric(titanic_file_train$Survived) and titanic_file_train$Age
## W = 12262, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(as.numeric(titanic_file_train$Survived), titanic_file_train$Fare)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  as.numeric(titanic_file_train$Survived) and titanic_file_train$Fare
## W = 0, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Puesto que para ambas variables se obtiene un p-valor menor que el valor de significación fijado, rechazamos la hipótesis nula. Por tanto, podemos concluir que, efectivamente, existe relación entre la variable objetivo y las variables de Edad y Precio.

5.3 Modelos de predicción

En el último apartado de este punto de análisis estadístico, se procede a realizar tres modelos de predicción diferentes y comparar cuál es el mejor.

- En primer lugar, se utilizará una regresión múltiple logarítmica ya que la variable objetivo es dicotómica. En caso de que no fuese una variable dicotómica, tendría más sentido realizar una regresión múltiple lineal.
- A continuación se utilizará el modelo de clasificación supervisado C50 que corresponde a un árbol de decisión.
- Por último, se utilizará otro algoritmo de árbol de decisión como es el RPART.

5.3.1 Regresión logarítmica

En primer lugar, se procede a generar una regresión logarítmica múltiple con la función **glm**. Esta función recibe tres parámetros, siendo la primera la variable objetivo, la segunda las variables independientes, y en tercer lugar el set de datos. Adicionalmente, le indicamos que la familia elegida para el modelo es la binomial puesto que la variable objetivo es dicotómica.

Tal y como se ha visto, las 6 variables analizadas, a parte de la variable objetivo, son relevantes para el modelo. No obstante, se crean diferentes modelos logarítmicos con diferentes variables para analizar cúal es el que mejor resultados obtiene.

```
# Se crean 5 modelos con diferentes variables.
modelo_glm_1 <- glm(Survived ~ Pclass + Sex + Embarked, data = titanic_file_train, family = "binomial")
modelo_glm_2 <- glm(Survived ~ Age + Fare + FamilySize, data = titanic_file_train, family = "binomial")
modelo_glm_3 <- glm(Survived ~ Pclass + Sex + Age + Fare, data = titanic_file_train, family = "binomial")
modelo_glm_4 <- glm(Survived ~ Sex + Fare + Embarked + FamilySize, data = titanic_file_train, family = "b
inomial")
modelo_glm_5 <- glm(Survived ~ Pclass + Sex + Age + Fare + Embarked + FamilySize, data =
titanic_file_train, family = "binomial")
```

Una vez se tienen los modelos, se comprueba cuál es el mejor. Para ello se obtiene el R^2 de cada modelo y se compara con los demás. Para obtener este valor se utiliza la función *rsq* de la librería RSQ. La teória indica que el mejor modelo será aquel que tenga el valor R^2 más alto.

```
library(rsq)
# Se crea la tabla de coeficientes
tabla.coeficientes <- matrix(c(1, rsq(modelo_glm_1, adj=TRUE),
                               2, rsq(modelo_glm_2, adj=TRUE),
                               3, rsq(modelo_glm_3, adj=TRUE),
                               4, rsq(modelo_glm_4, adj=TRUE),
                               5, rsq(modelo_glm_5, adj=TRUE)),
                             ncol = 2, byrow = TRUE
                     )

# Se muestra por pantalla la matriz creada
colnames(tabla.coeficientes) <- c("Modelo", "R^2")
tabla.coeficientes
```

```
##      Modelo        R^2
## [1,]      1 0.37678930
## [2,]      2 0.09534147
## [3,]      3 0.39280374
## [4,]      4 0.34294410
## [5,]      5 0.40609609
```

Como podemos observar, el modelo que mejor resultado da es el último, el modelo 5, el cual contiene las 6 variables. No obstante, cabe mencionar que el modelo 3, el cual se ha realizado en base a las variables categóricas más la variable numérica Fare, que son dos variables menos, tiene unos resultados bastantes parecidos al modelo 5.

Otra manera de obtener el mejor modelo es con el parámetro AIC, Akaike information criterion. Cuanto más bajo sea el parámetro, mejor será el modelo. Se comprueba este parámetro para los dos mejores modelos, el modelo 3 y el 5.

```
summary(modelo_glm_3$aic)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   814.1   814.1   814.1   814.1   814.1   814.1
```

```
summary(modelo_glm_5$aic)
```

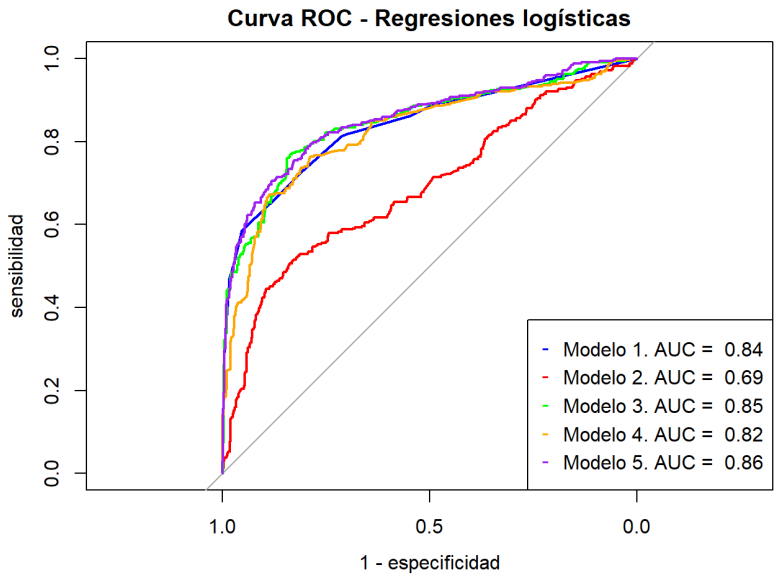
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   801.6   801.6   801.6   801.6   801.6   801.6
```

Se puede comprobar que el parámetro AIC para el Modelo 3 es de 814, mientras que en el Modelo 5 tiene un valor de 801. Así pues, entre los modelos de regresión logarítmico, el mejor modelo es el utilizado con todas las variables, el **Modelo 5**.

5.3.1.1 Curva ROC

Para ahondar más en el análisis de los modelos construidos, dibujaremos su curva ROC. Esto nos ofrecerá una visión de su sensibilidad (capacidad del modelo para identificar verdaderos positivos) y la relación con su especifidad (capacidad del modelo para identificar verdaderos negativos). El área bajo esta curva nos ofrece otra medida de la bondad del modelo, siendo 1 el valor máximo y 0.5 el mínimo. Un aspecto bueno de este análisis es que es insensible completamente a la distribución de las clases en los datos. Pongamos por ejemplo que de 1000 personas observadas, 999 hubieran sobrevivido al accidente del Titanic. Un modelo que dijera siempre que la persona observada ha sobrevivido ofrecería una precisión del 99.9 %, pero realmente este no sería un buen modelo y analizar únicamente la precisión nos llevaría a conclusiones erróneas. Este modelo tendría una sensibilidad enorme, pero su especificidad sería pésima, ya que no sería capaz de identificar verdaderos negativos. La curva ROC, por el contrario, nos revelaría la verdad acerca de este modelo.

```
library(installr)
library(pROC)

roc1 <- roc(titanic_file_train$Survived, modelo_glm_1$fitted.value)
roc2 <- roc(titanic_file_train$Survived, modelo_glm_2$fitted.value)
roc3 <- roc(titanic_file_train$Survived, modelo_glm_3$fitted.value)
roc4 <- roc(titanic_file_train$Survived, modelo_glm_4$fitted.value)
roc5 <- roc(titanic_file_train$Survived, modelo_glm_5$fitted.value)

plot(roc1, col = "blue", main = "Curva ROC - Regresiones logísticas", xlab = "1 - especificidad", ylab =
"sensibilidad")
lines(roc2, type = "l", col = "red")
lines(roc3, type = "l", col = "green")
lines(roc4, type = "l", col = "orange")
lines(roc5, type = "l", col = "purple")
legend("bottomright", legend = c(paste("Modelo 1. AUC = ", round(auc(roc1), 2)), paste("Modelo 2. AUC = "
, round(auc(roc2), 2)), paste("Modelo 3. AUC = ", round(auc(roc3), 2)), paste("Modelo 4. AUC = ",
round(auc(roc4), 2)), paste("Modelo 5. AUC = ", round(auc(roc5), 2))), col = c("blue", "red", "green", "or
ange", "purple"), pch = c("-", "-", "-", "-", "-"), ncol = 1)
```



La métrica del área bajo la curva ROC nos dice que el mejor modelo en términos de sensibilidad y especifidad es el modelo número 5, con un área bajo la curva de 0.86.

5.3.1.2 Predicción del Modelo Logarítmico

Tras confirmar que el mejor modelo logarítmico es el 5, se predice la probabilidad de supervivencia de una nueva persona.

En este caso, añadimos una chica joven perteneciente a la primera clase. Podemos comprobar que la probabilidad de sobrevivir es de un 96%, bastante alta. Si aplicamos estos mismos valores para un chico, podemos observar que la probabilidad disminuye a un 66%.

```
new_female <- data.frame(
  Pclass = "1",
  Sex = "female",
  Age = 25,
  Fare = 80,
  Embarked = "C",
  FamilySize = 1
)

new_male <- data.frame(
  Pclass = "1",
  Sex = "male",
  Age = 25,
  Fare = 80,
  Embarked = "C",
  FamilySize = 1
)

# Predecir la supervivencia con el modelo logaritmico
predict(modelo_glm_5, new_female, type = "response")
```

```
##         1
## 0.9686665
```

```
predict(modelo_glm_5, new_male, type = "response")
```

```
##         1
## 0.6643927
```

5.3.2 Árbol de decisión - C50

El siguiente modelo de predicción que se va a utilizar es el modelo C50, un modelo supervisado que replica un árbol de decisión.

5.3.2.1 Preparación de los datos

A continuación se preparan los datos para ejecutar correctamente el modelo supervisado. En primer lugar, puesto que la clase 'Embarked' tiene 4 categorías cuando en verdad son 3, se pondrá a la primera categoría el valor 'missing.

```
levels(titanic_file_train$Embarked)[1] = "missing"
```

En segundo lugar, sobre el conjunto de datos preprocesado con las 7 variables (variable objetivo más las 6 variables independientes), se procede a dividir el dataset entre la variable objetivo (el diagnóstico) y el resto de variables.

```
train_file <- titanic_file_train[!is.na(titanic_file_train$Survived),]

set.seed(666)
# Variable objetivo en Y
y <- train_file[,1]
# Resto de variables en X
X <- train_file[,2:7]
```

A continuación se procede a dividir el dataset en dos con los datos de entrenamiento y de testeo. La manera más óptima de dividir los datos son en 2/3 para el conjunto de entrenamiento y 1/3 para el conjunto de prueba con una función para no tener que hacer el análisis a mano.

```
indexes = sample(1:nrow(titanic_file_train), size=floor((2/3)*nrow(titanic_file_train)))
trainX<-X[indexes,]
trainy<-y[indexes]
testX<-X[-indexes,]
testy<-y[-indexes]
```

5.3.2.2 Implementación del algoritmo

Tras tener el conjunto de datos divididos en dos, se procede a implementar el modelo con la función C50, propia de la librería 'C50', y a observar los resultados obtenidos:

```
#install.packages("C50")
library(C50)

model <- C50::C5.0(trainX, trainy,rules=TRUE )
summary(model)
```

```
##
## Call:
## C5.0.default(x = trainX, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Jun 08 22:50:27 2020
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 594 cases (7 attributes) from undefined.data
```

```
##
## Rules:
##
## Rule 1: (20/1, lift 1.4)
##  Sex = male
##  FamilySize > 4
##  ->  class 0  [0.909]
##
## Rule 2: (159/20, lift 1.4)
##  Pclass = 3
##  Fare <= 10.5167
##  Embarked = S
##  ->  class 0  [0.870]
##
## Rule 3: (18/2, lift 1.3)
##  Pclass = 3
##  Sex = female
##  Fare > 17.4
##  Embarked = S
##  ->  class 0  [0.850]
##
## Rule 4: (377/60, lift 1.3)
##  Sex = male
##  Age > 9
##  ->  class 0  [0.839]
##
## Rule 5: (106/5, lift 2.6)
##  Pclass in {1, 2}
##  Sex = female
##  ->  class 1  [0.944]
##
## Rule 6: (13, lift 2.5)
##  Sex = male
##  Age <= 9
##  FamilySize <= 4
##  ->  class 1  [0.933]
##
## Rule 7: (25/4, lift 2.2)
##  Sex = female
##  Fare > 10.5167
##  Fare <= 17.4
##  Embarked = S
##  ->  class 1  [0.815]
##
## Rule 8: (72/13, lift 2.2)
##  Sex = female
##  Embarked in {C, Q}
##  ->  class 1  [0.811]
##
## Default class: 0
##
##
## Evaluation on training data (594 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##       8   91(15.3%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     357    19    (a): class 0
##      72   146    (b): class 1
##
##
##  Attribute usage:
##
##   95.96% Sex
##   65.66% Age
##   47.64% Pclass
##   46.13% Embarked
##   34.01% Fare
##    5.56% FamilySize
##
##
## Time: 0.0 secs
```

Para este caso, se han analizado 594 casos de los cuales podemos sacar las siguientes conclusiones:

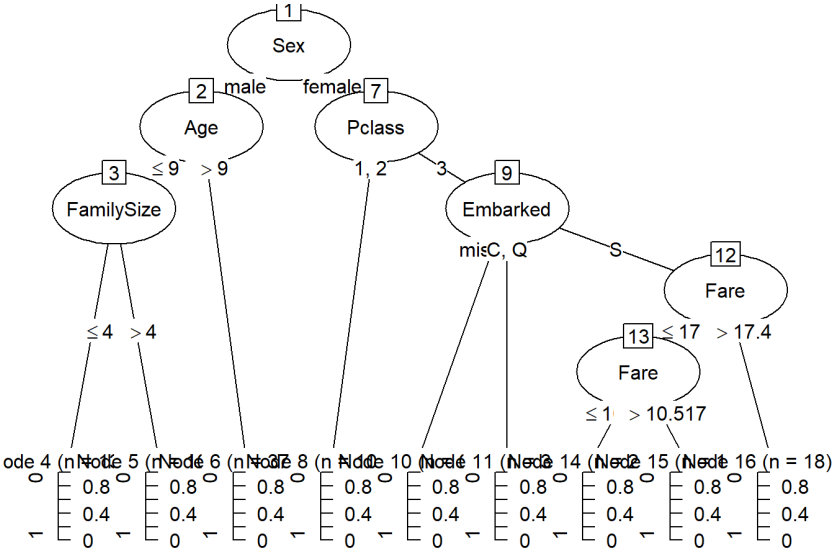
- Sólamente ha clasificado mal 91 casos, lo que hace una tasa de error del 15.3% o, mejor dicho, un acierto de aproximadamente un 85%

Adicionalmente, podemos observar que el modelo ha utilizado todas las variables, obteniendo en más de un 50% la variable Sexo y Edad. El resto de variables parecen no ser tan significativas en este modelo.

5.3.2.3 Gráfico C50

A continuación se muestra el árbol obtenido que representa las reglas de asociación anteriores:

```
model <- C50::C5.0(trainX, trainy)
plot(model)
```



Por ejemplo, si la persona es de sexo masculino, mayor de 9 años, y con más de 4 miembros en la familia, con más de un 80% de probabilidades la persona no sobrevivirá al accidente.

5.3.2.4 Predecir con muestras no usadas

Una vez tenemos el modelo, podemos comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

```
predicted_model <- predict(model, testX, type="class" )

print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model == testy) /
length(predicted_model)))
```

```
## [1] "La precisión del árbol es: 83.5017 %"
```

Nuestro modelo predice con un 83% de acierto, lo que lo hace bastante eficaz. Como en este caso se tienen pocas clases, se puede utilizar una matriz de confusión para identificar los tipos de errores cometidos:

```
mat_conf<-table(testy,Predicted=predicted_model)
mat_conf
```

```
##       Predicted
## testy   0   1
##     0 163  10
##     1  39  85
```

Podemos observar que ha clasificado bastante bien los 248 registros, y que se ha confundido en 49, 10 prediciendo supervivencia cuando no lo era, y 39 de viceversa.

Adicionalmente, se puede utilizar la función 'crosstable' de la librería 'gmodels' para obtener más detalle de los resultados obtenidos utilizando el set de prueba.

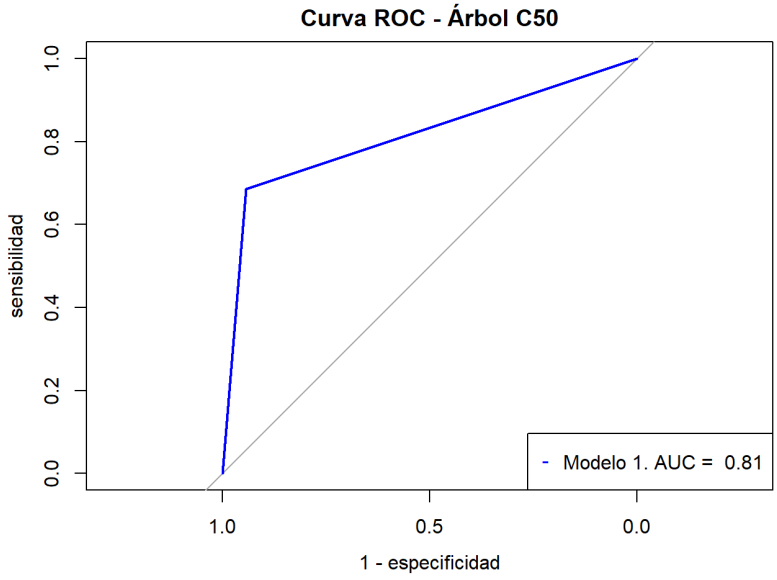
```
library(gmodels)
CrossTable(testy, predicted_model,prop.chisq  = FALSE, prop.c = FALSE, prop.r =FALSE,dnn = c('Reality', '
Prediction'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  297
##
##
##              | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       163 |        10 |       173 |
##              |     0.549 |     0.034 |           |
## -------------|-----------|-----------|-----------|
##            1 |        39 |        85 |       124 |
##              |     0.131 |     0.286 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       202 |        95 |       297 |
## -------------|-----------|-----------|-----------|
##
##
```

5.3.2.5 Curva ROC

Al igual que hemos hecho para el modelo de regresión logística, construiremos la curva ROC del modelo de árbol de clasificación a fin de establecer una comparación completa. Recordemos que la curva ROC de referencia del modelo de regresión nos dejaba un área bajo la curva de 0.86.

```
roc_c50 <- roc(as.numeric(testy), as.numeric(predicted_model))

plot(roc_c50, col = "blue", main = "Curva ROC - Árbol C50", xlab = "1 - especificidad", ylab = "sensibilid
ad")
legend("bottomright", legend = c(paste("Modelo 1. AUC = ", round(auc(roc_c50), 2))), col = c("blue"), pch
= c("--"), ncol = 1)
```



La curva ROC de este modelo nos ofrece un parámetro AUC = 0.81. Este análisis nos sugiere que el modelo de árbol es ligeramente peor al modelo de regresión, al menos en términos de sensibilidad frente a especificidad.

5.3.3 Aplicación algoritmo RPART

El último apartado de la esta práctica de algoritmos supervisados es ejecutar el dataset obtenido con otro árbol de decisión. En este caso, se ha utilizado la librería 'party' [9] y 'rpart' [10], ambos en la bibliografía.

El algoritmo RPART es una implantación de CART [Classification and Regression Tree en inglés] de aprendizaje automático supervisado. Este algoritmo se llama así por Recursive PARTitioning. Como C50, rpart usa métricas computacionales para determinar la mejor regla de división de datos entre clases. En cada nodo, rpart minimiza el coeficiente de Gini. Sin embargo, la diferencia respecto a C50 es que rpart() utiliza regresiones predefinidas como primer argumento. La forma de llamar a la función es: variable objetivo ~ variable A de entrada + [resto de variables de entrada]. Como salida se obtiene un árbol de decisión que puede ser usado para asignar o predecir una clase a los datos sin clasificar.

A continuación se ejecuta la función rpart, insertándole nuestra variable objetivo 'Survived' y el resto de datos. Se utilizará la división de datos que se hizo previamente en el apartado anterior, donde 'y' almacena la variable objetivo y 'X' almacena el resto de variables.

```r
# Librerías
library(rpart)
library(rpart.plot)
library(party)

# Algoritmo Rpart
rpart_data = rpart(y ~ ., data=X, method="class")
summary(rpart_data)
```

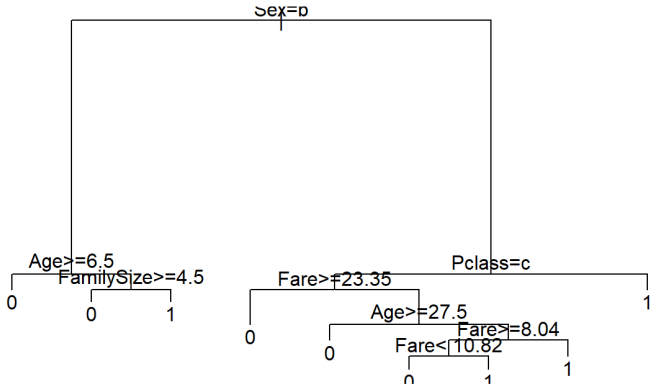
```
## Call:
## rpart(formula = y ~ ., data = X, method = "class")
##   n= 891
##
##           CP nsplit rel error    xerror       xstd
## 1 0.44444444      0 1.0000000 1.0000000 0.04244576
## 2 0.03070175      1 0.5555556 0.5555556 0.03574957
## 3 0.02339181      3 0.4941520 0.5029240 0.03444798
## 4 0.02046784      4 0.4707602 0.4912281 0.03413963
## 5 0.01169591      6 0.4298246 0.5000000 0.03437157
## 6 0.01000000      8 0.4064327 0.5087719 0.03459945
##
## Variable importance
##        Sex       Fare     Pclass FamilySize        Age   Embarked
##         45         17         13         12         10          3
##
## Node number 1: 891 observations,    complexity param=0.4444444
##   predicted class=0  expected loss=0.3838384  P(node) =1
##     class counts:   549   342
##    probabilities: 0.616 0.384
##   left son=2 (577 obs) right son=3 (314 obs)
##   Primary splits:
##       Sex        splits as  RL,           improve=124.42630, (0 missing)
##       Pclass     splits as  RRL,          improve= 43.78183, (0 missing)
##       Fare       < 52.2771  to the left,  improve= 35.31500, (0 missing)
##       FamilySize < 1.5      to the left,  improve= 17.43059, (0 missing)
##       Embarked   splits as  -RLL,         improve= 11.92920, (0 missing)
##   Surrogate splits:
##       Fare       < 77.6229  to the left,  agree=0.679, adj=0.089, (0 split)
##       FamilySize < 1.5      to the left,  agree=0.672, adj=0.070, (0 split)
##       Age        < 15.5     to the right, agree=0.651, adj=0.010, (0 split)
##
## Node number 2: 577 observations,    complexity param=0.02339181
##   predicted class=0  expected loss=0.1889081  P(node) =0.647587
##     class counts:   468   109
##    probabilities: 0.811 0.189
##   left son=4 (553 obs) right son=5 (24 obs)
##   Primary splits:
##       Age        < 6.5      to the right, improve=11.431650, (0 missing)
##       Pclass     splits as  RLL,          improve=10.019140, (0 missing)
##       Fare       < 26.26875 to the left,  improve= 9.269959, (0 missing)
##       FamilySize < 1.5      to the left,  improve= 3.147488, (0 missing)
##       Embarked   splits as  -RLL,         improve= 3.079304, (0 missing)
##
## Node number 3: 314 observations,    complexity param=0.03070175
##   predicted class=1  expected loss=0.2579618  P(node) =0.352413
##     class counts:    81   233
##    probabilities: 0.258 0.742
##   left son=6 (144 obs) right son=7 (170 obs)
##   Primary splits:
##       Pclass     splits as  RRL,          improve=31.163130, (0 missing)
##       FamilySize < 4.5      to the right, improve=16.243840, (0 missing)
##       Fare       < 48.2     to the left,  improve=10.114210, (0 missing)
##       Embarked   splits as  -RLL,         improve= 3.450116, (0 missing)
##       Age        < 32.25    to the left,  improve= 2.950860, (0 missing)
##   Surrogate splits:
##       Fare       < 25.69795 to the left,  agree=0.799, adj=0.563, (0 split)
##       Age        < 24.5     to the left,  agree=0.704, adj=0.354, (0 split)
##       Embarked   splits as  -RLR,         agree=0.637, adj=0.208, (0 split)
##       FamilySize < 4.5      to the right, agree=0.608, adj=0.146, (0 split)
##
## Node number 4: 553 observations
##   predicted class=0  expected loss=0.1681736  P(node) =0.620651
##     class counts:   460    93
##    probabilities: 0.832 0.168
##
## Node number 5: 24 observations,    complexity param=0.02046784
##   predicted class=1  expected loss=0.3333333  P(node) =0.02693603
##     class counts:     8    16
##    probabilities: 0.333 0.667
##   left son=10 (9 obs) right son=11 (15 obs)
##   Primary splits:
##       FamilySize < 4.5      to the right, improve=8.8888890, (0 missing)
##       Pclass     splits as  RRL,          improve=3.8095240, (0 missing)
```

```
##       Fare       < 20.825   to the right, improve=2.666670, (0 missing)
##       Age        < 1.5      to the right, improve=0.6095238, (0 missing)
##   Surrogate splits:
##       Pclass   splits as  RRL,          agree=0.792, adj=0.444, (0 split)
##       Fare     < 26.95    to the right, agree=0.750, adj=0.333, (0 split)
##       Embarked splits as  -RLR,         agree=0.708, adj=0.222, (0 split)
##
## Node number 6: 144 observations,    complexity param=0.03070175
##   predicted class=0  expected loss=0.5  P(node) =0.1616162
##     class counts:    72    72
##    probabilities: 0.500 0.500
##   left son=12 (27 obs) right son=13 (117 obs)
##   Primary splits:
##       Fare       < 23.35    to the right, improve=10.051280, (0 missing)
##       FamilySize < 4.5      to the right, improve=10.051280, (0 missing)
##       Embarked   splits as  -RRL,         improve= 7.071429, (0 missing)
##       Age        < 38.5     to the right, improve= 4.545455, (0 missing)
##   Surrogate splits:
##       FamilySize < 4.5      to the right, agree=0.944, adj=0.704, (0 split)
##       Age        < 37.5     to the right, agree=0.819, adj=0.037, (0 split)
##
## Node number 7: 170 observations
##   predicted class=1  expected loss=0.05294118  P(node) =0.1907969
##     class counts:     9   161
##    probabilities: 0.053 0.947
##
## Node number 10: 9 observations
##   predicted class=0  expected loss=0.1111111  P(node) =0.01010101
##     class counts:     8     1
##    probabilities: 0.889 0.111
##
## Node number 11: 15 observations
##   predicted class=1  expected loss=0  P(node) =0.01683502
##     class counts:     0    15
##    probabilities: 0.000 1.000
##
## Node number 12: 27 observations
##   predicted class=0  expected loss=0.1111111  P(node) =0.03030303
##     class counts:    24     3
##    probabilities: 0.889 0.111
##
## Node number 13: 117 observations,    complexity param=0.02046784
##   predicted class=1  expected loss=0.4102564  P(node) =0.1313131
##     class counts:    48    69
##    probabilities: 0.410 0.590
##   left son=26 (23 obs) right son=27 (94 obs)
##   Primary splits:
##       Age        < 27.5     to the right, improve=3.3508150, (0 missing)
##       Embarked   splits as  -RRL,         improve=2.6048030, (0 missing)
##       Fare       < 7.8875   to the right, improve=2.0325270, (0 missing)
##       FamilySize < 1.5      to the right, improve=0.1785425, (0 missing)
##
## Node number 26: 23 observations
##   predicted class=0  expected loss=0.3478261  P(node) =0.02581369
##     class counts:    15     8
##    probabilities: 0.652 0.348
##
## Node number 27: 94 observations,    complexity param=0.01169591
##   predicted class=1  expected loss=0.3510638  P(node) =0.1054994
##     class counts:    33    61
##    probabilities: 0.351 0.649
##   left son=54 (49 obs) right son=55 (45 obs)
##   Primary splits:
##       Fare       < 8.0396   to the right, improve=1.9626670, (0 missing)
##       Embarked   splits as  -RRL,         improve=1.7716050, (0 missing)
##       Age        < 6.5      to the right, improve=0.9354783, (0 missing)
##       FamilySize < 2.5      to the left,  improve=0.1324899, (0 missing)
##   Surrogate splits:
##       FamilySize < 1.5      to the right, agree=0.851, adj=0.689, (0 split)
##       Embarked   splits as  -LRL,         agree=0.723, adj=0.422, (0 split)
##       Age        < 11       to the left,  agree=0.628, adj=0.222, (0 split)
##
## Node number 54: 49 observations,    complexity param=0.01169591
##   predicted class=1  expected loss=0.4489796  P(node) =0.05499439
##     class counts:    22    27
##    probabilities: 0.449 0.551
##   left son=108 (12 obs) right son=109 (37 obs)
##   Primary splits:
##       Fare       < 10.825   to the left,  improve=4.6953480, (0 missing)
##       FamilySize < 1.5      to the left,  improve=3.0961800, (0 missing)
##       Age        < 6.5      to the right, improve=2.5331860, (0 missing)
##       Embarked   splits as  -RRL,         improve=0.5815983, (0 missing)
##   Surrogate splits:
##       FamilySize < 1.5      to the left,  agree=0.878, adj=0.5, (0 split)
##
```

```
## Node number 55: 45 observations
##   predicted class=1  expected loss=0.2444444  P(node) =0.05050505
##     class counts:    11    34
##    probabilities: 0.244 0.756
##
## Node number 108: 12 observations
##   predicted class=0  expected loss=0.1666667  P(node) =0.01346801
##     class counts:    10     2
##    probabilities: 0.833 0.167
##
## Node number 109: 37 observations
##   predicted class=1  expected loss=0.3243243  P(node) =0.04152637
##     class counts:    12    25
##    probabilities: 0.324 0.676
```

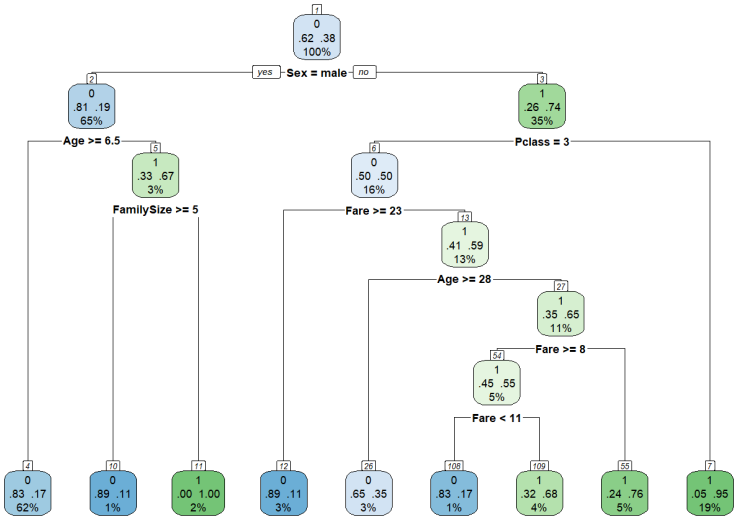
La gran diferencia de este algoritmo con C50 es cómo representa el árbol de decisión. Para ello se debe primero crear el esqueleto con la función plot() y luego añadir el texto de decisión con la función text().

```
plot(rpart_data)
text(rpart_data)
```



Debido a que puede ser un poco dificil de entender la gráfica anterior, se puede utilizar la función 'rpart.plot()' para obtener una mejor visualización. Los nodos interiores están coloreados en un tono claro de la clase objetivo. El nodo muestra la proporción de cada clase en ese nodo y el porcentaje del conjunto de datos en ese nodo. Los nodos hoja se colorean en función de la clase de nodo. El nodo muestra la proporción de cada clase en ese nodo y el porcentaje de la clase correcta del conjunto de datos en ese nodo.

```
rpart.plot(rpart_data, extra = 104, nn = TRUE)
```



Las principales reglas que se obtienen son:

- Si el sexo de la persona es femenino, se asignará con un 62% de precisión a una persona de supervivencia.
- Si el sexo de la persona es masculino y es mayor de 6.5 años, entonces con un 83% de precisión se asignará como que no sobrevive.
- Si el sexo de la persona es femenino, y pertenece a una tercera clase, entonces con un 20% de precisión se asignará a una persona de supervivencia.

Por último, se utiliza el modelo de predicción para los datos de prueba que no se han usado en el aprendizaje. Podemos observar que la precisión es de un 84%, ligeramente inferior al algoritmo C50.

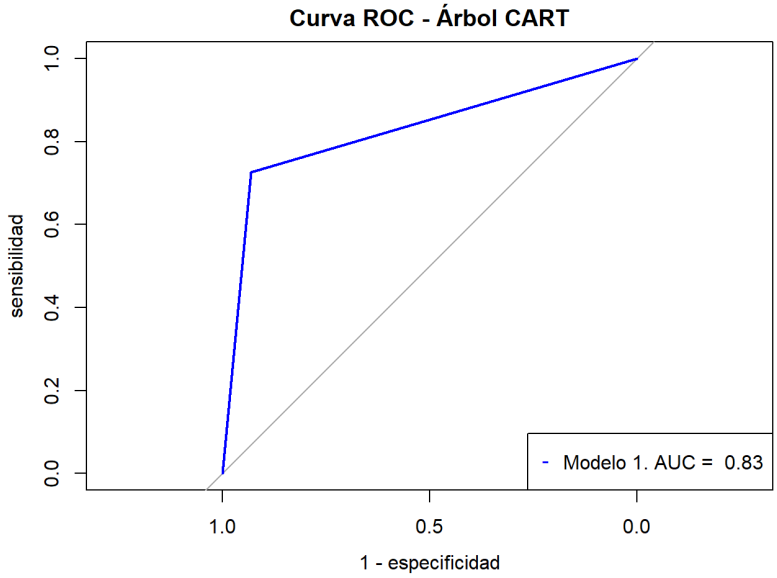
```
predicted_model <- predict(rpart_data, testX, type="class")
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model == testy) /
length(predicted_model)))
```

```
## [1] "La precisión del árbol es: 84.5118 %"
```

5.3.3.1 Curva ROC

Como ya hemos hecho con los modelos anteriores, además de calcular la precisión, dibujaremos la curva ROC de este último árbol para ofrece una medida más de la bondad del modelo.

```
roc_rpart <- roc(as.numeric(testy), as.numeric(predicted_model))

plot(roc_rpart, col = "blue", main = "Curva ROC - Árbol CART", xlab = "1 - especificidad", ylab = "sensibi
lidad")
legend("bottomright", legend = c(paste("Modelo 1. AUC = ", round(auc(roc_rpart), 2))), col = c("blue"), p
ch = c("--"), ncol = 1)
```



Por su parte, este último modelo nos ofrece un parámetro AUC = 0.83. En el aspecto de la curva ROC, el modelo de regresión logística sigue ofreciendo mejores resultados.

6 Exportación del fichero

Por último, tras terminar la práctica, se exporta el fichero limpio utilizado para la generación de los modelos predictivos a un formato CSV:

```
write.csv(titanic_file_train, "../dataset/Titanic_Final.csv")
```

7 Conclusiones del análisis

A lo largo de esta práctica se ha aprendido a realizar un análisis de datos de principio a fin. En primer lugar, se hizo un estudio exhaustivo del conjunto de datos más óptimo que se debía utilizar para dar respuesta a las preguntas formuladas. Finalmente, se optó por utilizar el dataset con la información de los pasajeros del Titanic proporcionado por Kaggle.

En segundo lugar, se han sometido los datos a un preprocesamiento para manejar los casos de ceros o elementos vacíos y valores extremos (outliers). En primer lugar, se han identificado los valores que debemos considerar como vacíos. Aquellas observaciones que contienen nulos o cadenas vacías son consideradas en primer lugar. Pero para las variables continuas, hemos analizado si los ceros existentes pueden tener sentido o si éstos son también valores perdidos. En el caso de la variable del precio del billete, los ceros han sido descartados y tratados como missing values. Se han aplicado varias técnicas para imputarlos. Aquellas variables numéricas cuya distribución lo permitía, han sido imputadas utilizando su mediana. La variable Fare, lejos de distribuirse normalmente, ha requerido utilizar el método de los k vecinos más próximos para imputar sus valores perdidos (incluyendo los ceros). Por último, para las variables categóricas que presentan datos faltantes, hemos utilizado la moda (al ser muy reducido el número de missing values). Para identificar los valores extremos, se ha estudiado la posibilidad de que existen valores no esperados en las variables categóricas. Para las variables numéricas, hemos utilizado diagramas de caja y la convención de considerar como outlier todo aquel valor que se aleje más de 1.5 veces el rango intercuartiles por encima y por debajo del tercer y primer cuartil, respectivamente. Los valores identificados han sido tomados como buenos, y no se les ha aplicado transformación alguna.

En tercer lugar, se ha realizado el análisis estadístico de los datos. En este apartado se ha elegido el conjunto de datos que mejor ayudaban a crear el modelo para predecir la variable objetivo, la variable de supervivencia. Para ello se han utilizado técnicas de correlación, para las variables numéricas, y el Test de Chi-Square para las variables categóricas basándonos en las hipótesis. Adicionalmente, se ha realizado un

estudio de normalidad y homogeneidad entre las variables numéricas. Puesto que las variables no seguían una normal ni homocedasticidad, se ha obtenido la relación de las variables numéricas respecto a la variable objetivo con un test de Mann Whitney. A continuación, se han creado tres modelos de predicción, los cuales son el modelo de regresión múltiple logarítmico, y dos modelos supervisados de clasificación basados en árboles de decisión, utilizando el algoritmo de C50 y Rpart. Hemos comprobado que, de estos dos últimos modelos, el C50 daba mejores resultados y conseguía predecir con más precisión. No obstante, el mejor modelo de los tres ha sido el modelo de regresión logarítmico. Durante este apartado se ha llegado a la conclusión de que había más probabilidades de sobrevivir si la persona del crucero era una chica, perteneciente a la primera clase social, y joven. El resto de variables, aunque ayudan a crear el modelo, no son tan representativas como estas tres.

Adicionalmente, tanto la tarea de preanálisis como de análisis de datos ha venido acompañada en todo momento de gráficas y tablas que ayudan a aclarar y confirmar las conclusiones obtenidas.

8 Tabla de participantes

A continuación se muestra la tabla de participantes firmada por ambos.

```
##                                 Firma Jorge Santos Firma Javier Cela
## Contribuciones                  JSN                JCL
## Investigación previa            JSN                JCL
## Limpieza de los datos           JSN                JCL
## Análisis de los datos           JSN                JCL
## Representación de los resultados JSN               JCL
```

9 Bibliografía

Dataset

- [1] Selección del dataset utilizado: https://www.kaggle.com/c/titanic/overview

Libros

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

URLs

- [8] Análisis del R^2 dado un modelo de regresión logarítmico.
- [9] Librería Party: https://cran.r-project.org/web/packages/party/index.html
- [10] Algoritmo Rpart: https://www.rdocumentation.org/packages/rpart/versions/4.1-15/topics/rpart
- [11] Ejemplo de código utilizando rpart y C50: http://mercury.webster.edu/aleshunas/R_learning_infrastructure/Classification%20of%20data%20using%20decision%20tree%20and%20regression%20tree%