Tipología y ciclo de vida de los datos - Práctica 2 (35% nota final)

Presentación

En esta práctica se elabora un caso práctico orientado a aprender a identificar los datos relevantes para un proyecto analítico y usar las herramientas de integración, limpieza, validación y análisis de las mismas. Para hacer esta práctica tendréis que trabajar en grupos de 2 personas. Tendréis que entregar un solo archivo con el enlace Github (https://github.com) donde se encuentren las soluciones incluyendo los nombres de los componentes del equipo. Podéis utilizar la Wiki de Github para describir vuestro equipo y los diferentes archivos que corresponden a vuestra entrega. Cada miembro del equipo tendrá que contribuir con su usuario Github. Aunque no se trata del mismo enunciado, los siguientes ejemplos de ediciones anteriores os pueden servir como guía:

- Ejemplo: https://github.com/Bengis/nba-gap-cleaning
- Ejemplo complejo (archivo adjunto).

Competencias

En esta práctica se desarrollan las siguientes competencias del Máster de Data Science:

- 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.

Objetivos

Los objetivos concretos de esta práctica son:

- Aprender a aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios o multidisciplinares.
- Saber identificar los datos relevantes y los tratamientos necesarios (integración, limpieza y validación) para llevar a cabo un proyecto analítico.

- Aprender a analizar los datos adecuadamente para abordar la información contenida en los datos.
- Identificar la mejor representación de los resultados para aportar conclusiones sobre el problema planteado en el proceso analítico.
- Actuar con los principios éticos y legales relacionados con la manipulación de datos en función del ámbito de aplicación.
- Desarrollar las habilidades de aprendizaje que les permitan continuar estudiando de un modo que tendrá que ser en gran medida autodirigido o autónomo.
- Desarrollar la capacidad de búsqueda, gestión y uso de información y recursos en el ámbito de la ciencia de datos.

Descripción de la Práctica a realizar

El objetivo de esta actividad será el tratamiento de un dataset, que puede ser el creado en la práctica 1 o bien cualquier dataset libre disponible en Kaggle (https://www.kaggle.com). Algunos ejemplos de dataset con los que podéis trabajar son:

- Red Wine Quality (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009)
- Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic)

El último ejemplo corresponde a una competición activa de Kaggle de manera que, opcionalmente, podéis aprovechar el trabajo realizado durante la práctica para entrar en esta competición.

Siguiendo las principales etapas de un proyecto analítico, las diferentes tareas a realizar (y justificar) son las siguientes:

1.Descripción del dataset.

¿Por qué es importante y qué pregunta/problema pretende responder?

El conjunto de datos objeto de análisis es el dataset disponible en Kaggle Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic). He elegido este dataset por la posibilidad que ofrece de entrar en la competición y Kaggle, y además me parece uno de los conjuntos de datos más claros para poder asentar los conocimientos adquiridos en la asignatura.

A partir de este conjunto de datos, el objetivo será construir un modelo que permita predecir la supervivencia al hundimiento del Titanic. Pese a que evidentemente ha habido elementos relacionados con la suerte, si que parece que determinados grupos

de personas tenían más posibilidades de sobrevivir que otros. El modelo debería de responder a la pregunta utilizando la información del dataset.¿Qué clases de personas tenían más posibilidades de sobrevivir?

Descripción del dataset:

En Kaggle se proporcionan dos conjuntos de datos: • Training set: Conjunto de entrenamiento que proporciona el resultado final. • Test set: Conjunto de prueba para testear el modelo.

Variable	Definición	Clave
PassengerId	Identificador numérico de un pasajero	
Survived	Solo aplica al dataset de entrenamiento. Indica si ese pasajero ha sobrevivido o no.	0 = No, 1 = Sí
Pclass	Es la clase del pasaje, permite inferir la situación económica del pasajero.	1 - Primera clase = Superior 2 - Segunda clase = Medio 3 - Tercera clase = Bajo
Name	Apellidos y nombre de los pasajeros	
Sex	Sexo del pasajero male/female	
Age	Edad en años	
Sibsp	Número de hermanos y esposos	
Parch	Número de padres e hijos	
Ticket	Número de ticket	
Fare	Tarifa del pasajero	
Cabin	Cabina del pasajero	
Embarked	Puerto origen en el que ha embarcado	C = Cherbourg, Q = Queenstown, S = Southampton

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

Revisamos la estructura de los dos dataset y vemos que hay determinada información que no nos va a aportar valor en el análisis a realizar con lo que la eliminamos tanto del conjunto de datos de entrenamiento como del de test. Los campos a eliminar son los siguientes:

- PasssengerID
- Name
- Ticket

- Cabin

Todos ellos son valores que no nos aportan valor para un análisis de machine learning ya que identifican individualmente a cada registro, o se trata de códigos numéricos. Realmente con el nombre sí se podría obtener información relevante, pero implica la aplicación de funciones de text analytics que no se van a revisar en este análisis.

En este punto hacemos los siguientes pasos:

- Cargamos los ficheros de test y training en data frames
- Revisamos la estructura de los ficheros
- Eliminamos aquellas columnas que no nos interesan
- De cara a aplicar las mismas reglas de transformación unificamos en un único dataset los dos subconjuntos.

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

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyverse)

## -- Attaching packages --------------------------------------
tidyverse 1.3.1 --

## v tibble  3.1.1     v purrr   0.3.4
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts -----------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(nortest)
library(class)
library(VIM)

## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.
```

```
## Suggestions and bug-reports can be submitted at:
https://github.com/statistikat/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep

library(knitr)
library(VIM)
library(car)

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:dplyr':
##
##     recode

library(C50)
library(e1071)
library(rpart)
library(rpart.plot)
library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

#Cargamos los dos conjuntos de datos a utilizar el de entrenamiento y el
de testing.
test_raw<-read.csv("D:/OneDrive/03. Formación/UOC. Máster Data
Science/M2.851 - Tipología y ciclo de vida de los datos/Práctica 2 -
Limpieza/input/test.csv", header=TRUE)
train_raw<-read.csv("D:/OneDrive/03. Formación/UOC. Máster Data
Science/M2.851 - Tipología y ciclo de vida de los datos/Práctica 2 -
Limpieza/input/train.csv", header=TRUE)

#Revisamos la estructura de ambos ficheros
str(test_raw)
```

```
## 'data.frame':    418 obs. of  11 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
##  $ Name       : chr  "Kelly, Mr. James" "Wilkes, Mrs. James (Ellen
Needs)" "Myles, Mr. Thomas Francis" "Wirz, Mr. Albert" ...
##  $ Sex        : chr  "male" "female" "male" "male" ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Ticket     : chr  "330911" "363272" "240276" "315154" ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Cabin      : chr  "" "" "" "" ...
##  $ Embarked   : chr  "Q" "S" "Q" "S" ...

str(train_raw)

## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : 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" ...

#Eliminamos de los data frames la información que no nos va a ser útil
para el análisis
train=subset(train_raw,select= c(-1,-4,-9,-11))
test=subset(test_raw,select = c(-1,-3,-8,-10))

#De cara a realizar el preprocesado y el feature enginneirng vamos a
combinar los dos datasets.
#Después lo volveremos a dividir para aplicar los algoriatmos de ML
titanic=bind_rows(train,test)
LT=dim(train)[1]
```

3.Limpieza de los datos.

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

Revisamos si los datos contienen valores cero o vacíos

```
#############################################################################
#############
#TRATAMIENTO DE VALORES NULOS O VACÍOS
#############################################################################
#############
#Revisamos los valores vacíos
colSums(is.na(titanic))

## Survived   Pclass      Sex      Age    SibSp    Parch     Fare
Embarked
##      418        0        0      263        0        0        1
0

colSums(titanic=="")

## Survived   Pclass      Sex      Age    SibSp    Parch     Fare
Embarked
##       NA        0        0       NA        0        0       NA
2
```

Como podemos ver tenemos los siguientes casos, en los que vamos a aplicar las siguientes acciones:

- Age: Presenta 263 registros con valor 0. En este caso vamos a sustituir esos valores aplicando el método de los k vecinos más cercanos. Hemos elegido este método ya que las características de los pasajeros pueden tener relación entre sí.
- Fare: Presenta 1 registro con valor 0.En este caso vamos a sustituir esos valores por la media de las tarifas.
- Embarked: Presenta 2 registros con valor vacío.Vamos a asignarle el valor más frecuente en los datos que es el de S- Southampton.

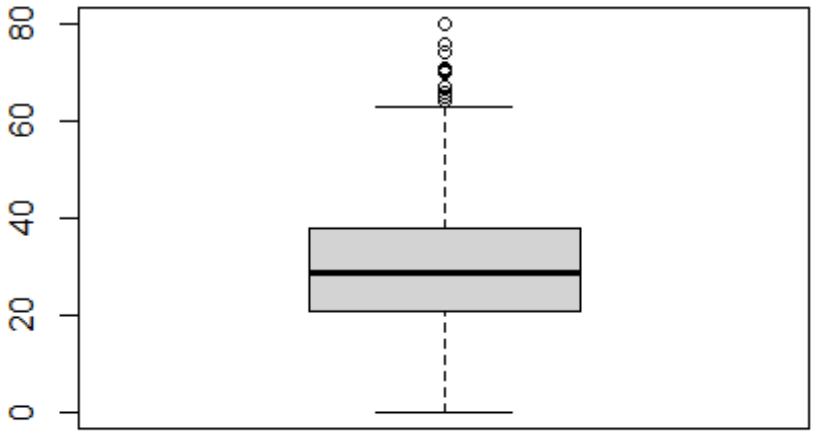
```
#Realizamos el tratamiento de los datos de las columnas con valores
vacíos o cero.
#Edad/Age
#titanic1<-knn(titanic,variable=c("Age"),k=6)
#summary(titanic1)
titanic$Age<-kNN(titanic)$Age

#Fare/Tarifa
media_tarifa=median(titanic$Fare,na.rm = TRUE)
titanic$Fare <- replace_na(titanic$Fare, media_tarifa)

#Embarked/Origen
#En este caso tenemos solo dos valores, con lo que los reemplazamos por
el más usado
titanic$Embarked[titanic$Embarked==""] <-'S'

#Revisamos si ahora aparecen esos valores cero o vacíos
colSums(is.na(titanic))
```

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

colSums(titanic=="")

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

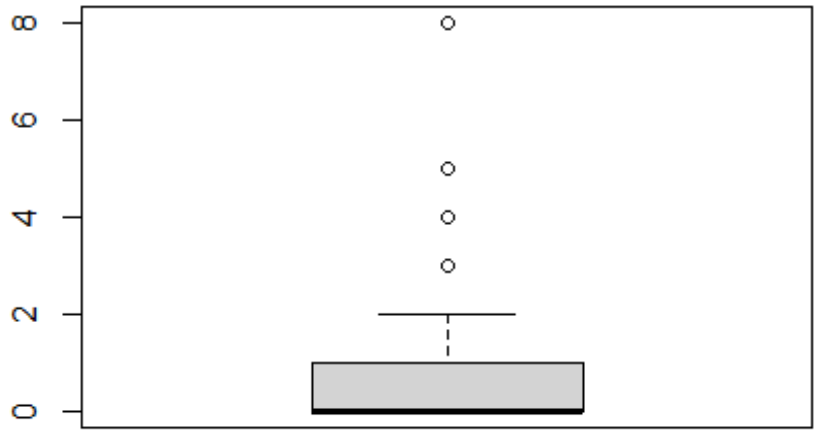
3.2. Identificación y tratamiento de valores extremos.

Los valores extremos o outliers son aquellos que parecen no ser congruentes si los comparamos con el resto de los datos. Para identificarlos, vamos a hacer uso de la función boxplots.stats() de R, la cual se emplea a continuación.

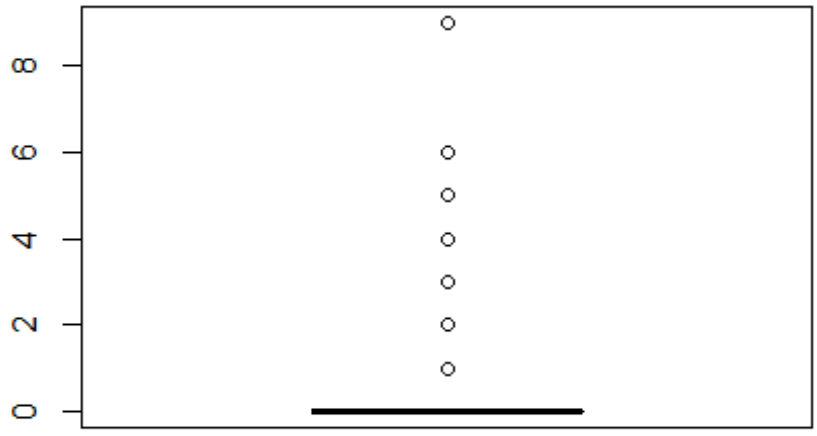
```
# Diagrama de cajas para cada variable de tipo numeric o integer
boxplot(titanic$Age)
```



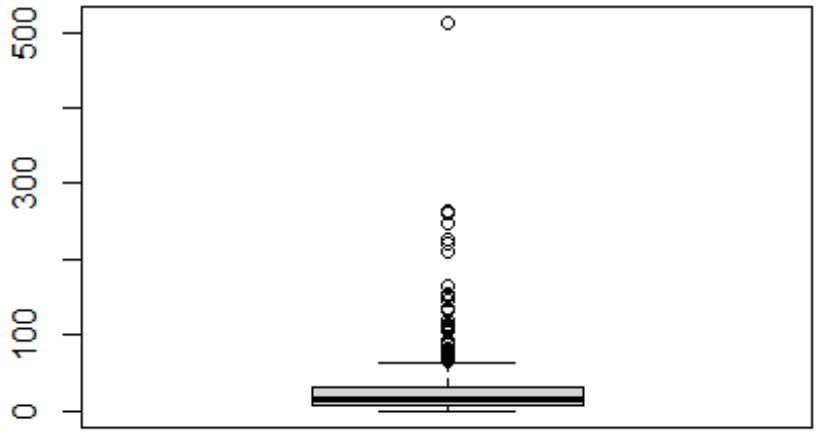

```
boxplot(titanic$SibSp)
```




```
boxplot(titanic$Parch)
```




```
boxplot(titanic$Fare)
```



Revisando cada una de las variables vemos que, aunque hay valores marcados como valores extremos, son perfectamente plausibles dentro del conjunto de datos, con lo que no vamos a hacer ningún tratamiento específico.

4. Análisis de los datos.

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

Los grupos de datos que se van a analizar son las siguientes variables:

- Pclass
- Sex
- Age
- SibSp
- Parch
- Fare
- Embarked

El objetivo de comparar dichas variables, es determinar, que valor que presenta la variable Survived, con el objetivo de saber si el méetodo de entrenamiento es el correcto.

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

4.2.1 Comprobación de la normalidad

Para determinar la normalidad, se va a llevar a cabo el test de Shapiro-Wilk. El test parte de la suposición de la hipótesis nula en donde la población está distribuida normalmente, por lo tanto, si el p-valor es menor al nivel de significancia (en este caso 0.05), la hipótesis nula es rechazada y se concluye que los datos no cuentan con una distribución normal.

```
shapiro.test (titanic$Age)

##
##  Shapiro-Wilk normality test
##
## data:  titanic$Age
## W = 0.9818, p-value = 8.839e-12

shapiro.test (titanic$SibSp)

##
##  Shapiro-Wilk normality test
##
## data:  titanic$SibSp
## W = 0.51108, p-value < 2.2e-16

shapiro.test (titanic$Parch)

##
##  Shapiro-Wilk normality test
##
## data:  titanic$Parch
## W = 0.49797, p-value < 2.2e-16

shapiro.test (titanic$Fare)

##
##  Shapiro-Wilk normality test
##
## data:  titanic$Fare
## W = 0.52766, p-value < 2.2e-16
```

Como se puede observar el p-valor es siempre inferior a 0.05 en todas las variables, por lo que se puede rechazar la hiótesis nula y entender que no hay normalidad.

4.2.2 Homogeneidad de la varianza.

Para realizar la comprobación de la homogeneidad de varianza se va a realizar el test de Leven. Vamos a aplicar el test sobre el conjunto de datos de entrenamiento, dado que la variable que nos interesa analizar es la supervivencia.

```r
#Vamos a convertir en factores las características que apliquen
cols<-c("Survived","Pclass","Sex","Embarked")
for (i in cols){
  titanic[,i] <- as.factor(titanic[,i])
}

#Extraemos del dataset los datos de entrenamiento
entrenamiento<-titanic[1:891,]
test<-titanic[892:1309,]

# Test de Levene para la homogeneidad de varianza
leveneTest(y = entrenamiento$Age , group =entrenamiento$Survived , center
= "median")

## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value Pr(>F)
## group   1  3.2652 0.0711 .
##       889
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

leveneTest (y = entrenamiento$SibSp , group =entrenamiento$Survived ,
center = "median")

## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value Pr(>F)
## group   1  1.1106 0.2922
##       889

leveneTest (y = entrenamiento$Parch , group =entrenamiento$Survived ,
center = "median")

## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value Pr(>F)
## group   1  5.9635 0.0148 *
##       889
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

leveneTest (y = entrenamiento$Fare , group =entrenamiento$Survived ,
center = "median")

## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value    Pr(>F)
## group   1    45.1 3.337e-11 ***
##       889
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tomando como referencia el p-valor igual a 0.05, se puede determinar que no encuentra diferencias significativas entre las varianzas de Age y SibSp, en cambio si las encuentra sobre Parch y Fare.

Debido a los datos obtenidos en los puntos anteriores, vamos a normalizar los datos.

```
# Funcion para la normalizacion de los datos
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}
#Normalizacion de los datos
# Para el conjunto de datos de entrenamiento
titanic$Age <- normalize(titanic$Age)
titanic$SibSp <- normalize(titanic$SibSp)
titanic$Parch <- normalize(titanic$Parch)
titanic$Fare <- normalize(titanic$Fare)
```

4.3. 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.

Vamos a aplicar diferentes algoritmos con el objeivo de predecir la supervivencia de los pasajeros. De cara validar la calidad de la predicción vamos a utilizar un dataset que contiene la información real de supervivencia.

```
validacion<-read.csv("D:/OneDrive/03. Formación/UOC. Máster Data
Science/M2.851 - Tipología y ciclo de vida de los datos/Práctica 2 -
Limpieza/input/validation.csv", header=TRUE)
```

4.3.1. Regresión Logística

La Regresión Logística es un tipo de modelo de regresión. Es empleado para predecir variables categóricas. Calcula las probabilidades de ocurrencia de alguna de las clases del modelo:

```
#Creamos el modelo
modeloRL <- glm ( Survived ~ ., family = binomial , data=entrenamiento)

#Prediccion de la clasificacion en el conjunto de test
prediccionRL <- predict(modeloRL, type = 'response', newdata = test)
prediccionRL <- ifelse (prediccionRL> 0.5 , 1, 0)
prediccionRL <- as.factor(prediccionRL)
```

Analizamos ahora la validez del modelo respecto al conjunto de datos real

```
exitoPrediccionRL <- sum(prediccionRL ==
validacion$Survived)/length(prediccionRL)
exitoPrediccionRL

## [1] 0.8923445
```

```
tablePrediccionRL <- table(prediccionRL, validacion$Survived)
tablePrediccionRL

##
## prediccionRL   0   1
##            0 246  25
##            1  20 127
```

Como podemos ver obtenemos un porcentaje de acierto del 89,2%

4.3.2. Clasificador Bayesiado Ingenuo

Este modelo es un clasificador probabilístico y basado en el Teorema de Bayes. Se denomina ingenuo porque parte de la presunción de que todas las variables predictoras del modelo tienen total independencia lineal.

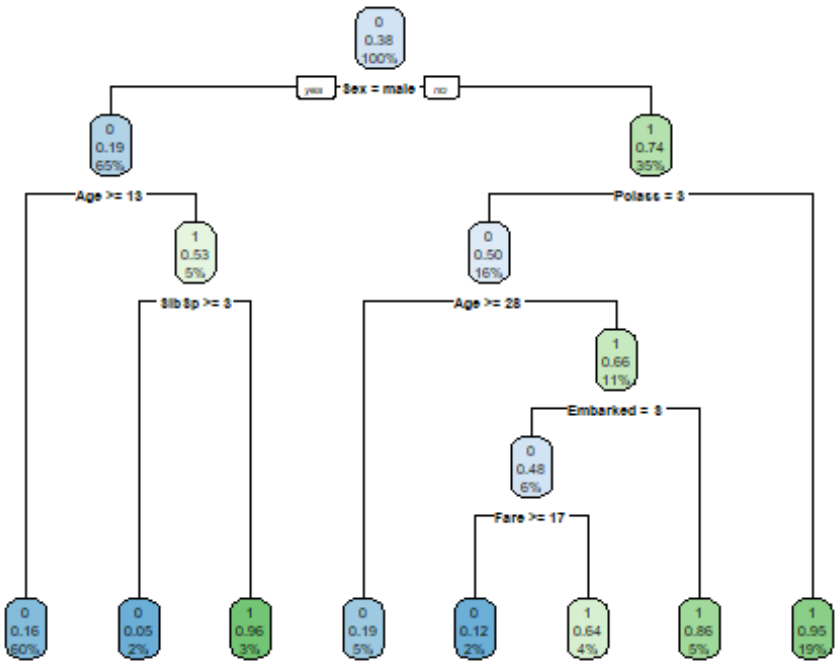
```
# Se establece una semilla para la generacion de numeros
set.seed (1234)
# Modelo
clasificadorBayes <- naiveBayes(Survived ~ ., data =entrenamiento)

# Prediccion de la clasificacion en el conjunto de test
prediccionBayes <- predict ( clasificadorBayes , newdata =test)
# Analisis de la clasificacion
exitoPrediccionBayes <- sum ( prediccionBayes == validacion $
Survived )/ length ( prediccionBayes )
exitoPrediccionBayes

## [1] 0.8325359

tablaPrediccionBayes <- table ( prediccionBayes , validacion $
Survived )
tablaPrediccionBayes

##
## prediccionBayes   0   1
##               0 240  44
##               1  26 108
```

Como podemos ver obtenemos un porcentaje de acierto del 83,2%

4.3.3. Clasificación con Árbol de Decisión

```
# Se establece una semilla para la generacion de numeros aleatorios
set.seed(1234)
# Construccion del modelo
clasifcadorArbol <-rpart(Survived ~ ., data = entrenamiento)
# Prediccion de la clasificacion en el conjunto de test
prediccionArbol <- predict(clasifcadorArbol, newdata = test, type =
'class')
# Analisis de la clasificacion
exitoPrediccionAD <- sum(prediccionArbol ==
```

```
validacion$Survived)/length(prediccionArbol)
exitoPrediccionAD

## [1] 0.8899522

tablePrediccionAD <- table(prediccionArbol, validacion$Survived)
tablePrediccionAD

##
## prediccionArbol   0   1
##               0 255  35
##               1  11 117

# Grafico arbol decision
rpart.plot(clasifcadorArbol)
```



En este caso obtenemos un porcentaje de acierto del 88,99%

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

Representaciones ya realizadas en el apartado anterior.

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Las conclusiones que se pueden obtener del análisis son las siguientes:

- La limpieza y preparación de los datos es clave para los resultados finales de los modelos aplicados.
- Es necesario conocer bien cada tipo de técnica y para que problemas está indicado.
- Los resultados obtenidos permiten responder a la pregunta de predecir que pasajeros sobreviven.
- El mejor modelo en nuestro caso ha sido la regresión con un porcentaje de 89.23.

7. Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.

En referente a la entrega final, hay que entregar un único fichero que contenga el enlace Github, el cual no se podrá modificar posteriormente a la fecha de entrega, donde haya:

- 1. Una Wiki con los nombres de los componentes del grupo y una descripción de los ficheros.
- 2. Un documento PDF con las respuestas a las preguntas y los nombres de los componentes del grupo. Además, al final del documento, deberá aparecer la siguiente tabla de contribuciones al trabajo, la cual debe firmar cada integrante del grupo con sus iniciales. Las iniciales representan la confirmación de que el integrante ha participado en dicho apartado. Todos los integrantes deben participar en cada apartado, por lo que, idealmente, los apartados deberían estar firmados por todos los integrantes.

Contribuciones	Firma
Investigación previa	Jose Dosil
Redacción de las respuestas	Jose Dosil
Desarrollo código	Jose Dosil

- 3. Una carpeta con el código generado para analizar los datos.
- 4. El fichero CSV con los datos originales.
- 5. El fichero CSV con los datos finales analizados.