MASTER UNIVERSITARIO DE CIENCIA DE DATOS

UNIVERSITAT OBERTA DE CATALUNYA

Tipología y ciclo de vida de los datos

Práctica (II)

Autor: ***Miguel Pérez Rego***

A Coruña, Junio 2020

Índice

1. Descripción del dataset

El dataset que se va a analizar es el siguiente conjunto de datos disponible en Kaggle y que se propone como ejemplo de uso en el enunciado de la práctica:

- Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic)

En este primer apartado se debe dar respuesta a la siguiente pregunta planteada: *¿Por qué es importante y qué pregunta/problema pretende responder?*

El conjunto de datos pretende predecir mediante el uso del aprendizaje automático qué pasajeros sobrevivieron al naufragio del *Titanic*. Asimismo, los datos también son importantes para poder conocer qué tipo de personas en base a sus características físicas y a las de su viaje a bordo del barco han sobrevivido.

Para el desarrollo de esta práctica se parte de los dos conjuntos de datos que se describen a continuación:

- Conjunto de entrenamiento: Es el que debe usarse para construir los modelos de aprendizaje automático. Se proporciona el resultado de la variable que indica la supervivencia de cada pasajero que es sobre la que se va a llevar a cabo el estudio.
- Conjunto de test: Es el que debe usarse para ver lo bien funciona el modelo con datos en base al aprendizaje automático llevado a cabo. No incluye la variable sobre la supervivencia de cada pasajero.
- Conjunto de datos para validar el porcentaje de los datos de test que han sido clasificados correctamente.

A continuación, se va a llevar a cabo una descripción de cada uno de os atributos de los que forma parte cada dataset:

- PassengerId: Identificador del pasajero a bordo.
- Survived: Indica si el pasajero ha sobrevivido. El valor *0* indica que no ha sobrevivido y el *1* que sí. Es la variable que se trata de predecir en el conjunto de test.

- Pclass: Es la clase en la que ha navegado el pasajero. Existen los siguiente valores: *1* es igual a primera clase, *2* es igual a segunda clase y *3* es igual a tercera clase.
- Name: Nombre completo del pasajero.
- Sex: Sexo del pasajero.
- Age: Edad en años del pasajero.
- SibSp: Número de hermanos o cónyuges a bordo del barco.
- Parch: Número de padres o hijos a bordo del barco.
- Ticket: El identificador del ticket.
- Fare: Tarifa del pasajero.
- Cabin: Número de cabina.
- Embarked: Puerto en el que ha embarcado el pasajero. C significa Cherbourg, Q significa Queenstown y S significa Southampton.

2. Integración y selección

En este apartado se va a llevar a cabo la integración y la selección de los datos que van a ser de interés para llevar a cabo el análisis.

Como se ha comentado en el apartado anterior, hay tres conjuntos de datos: entrenamiento, test y validación. Cada conjunto de datos tiene un identificador único que es el *PassengerId* que va a permitir relacionar el conjunto de test con el de validación para poder comprobar qué pasajeros fueron clasificados correctamente.

La carga de los datos se lleva a cabo con el siguiente código en R:

```
# Se cargan los datos del conjunto de entrenamiento
entrenamiento <- read.csv("/Users/miguelpr93/Desktop/titanic/
    train.csv", header = TRUE)

# Se cargan los datos del conjunto de test
test <- read.csv("/Users/miguelpr93/Desktop/titanic/test.csv",
    header = TRUE)

# Se cargan los datos del conjunto de test
validacion <- read.csv("/Users/miguelpr93/Desktop/titanic/gender_
    submission.csv", header = TRUE)
```

Se comprueba la correcta carga de los datos con la función *head* que muestra la cabecera y los primeros seis registros que forman parte de cada dataset. A continuación, se muestra el código usado:

```
# Con los siguientes registros se comprueba la correcta carga de
    los datos mostrando un ejemplo de los primeros seis registros
head(entrenamiento)
head(test)
head(validacion)
```

El resultado obtenido es el siguiente:

```
> head(entrenamiento)
  PassengerId Survived Pclass                                                 Name    Sex Age SibSp Parch           Ticket    Fare Cabin Embarked
1           1        0      3                              Braund, Mr. Owen Harris   male  22     1     0        A/5 21171  7.2500              S
2           2        1      1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0         PC 17599 71.2833   C85        C
3           3        1      3                               Heikkinen, Miss. Laina female  26     0     0 STON/O2. 3101282  7.9250              S
4           4        1      1       Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0           113803 53.1000  C123        S
5           5        0      3                             Allen, Mr. William Henry   male  35     0     0           373450  8.0500              S
6           6        0      3                                     Moran, Mr. James   male  NA     0     0           330877  8.4583              Q
> head(test)
  PassengerId Pclass                                     Name    Sex  Age SibSp Parch  Ticket    Fare Cabin Embarked
1         892      3                         Kelly, Mr. James   male 34.5     0     0  330911  7.8292              Q
2         893      3             Wilkes, Mrs. James (Ellen Needs) female 47.0     1     0  363272  7.0000              S
3         894      2                    Myles, Mr. Thomas Francis   male 62.0     0     0  240276  9.6875              Q
4         895      3                             Wirz, Mr. Albert   male 27.0     0     0  315154  8.6625              S
5         896      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female 22.0     1     1 3101298 12.2875              S
6         897      3                     Svensson, Mr. Johan Cervin   male 14.0     0     0    7538  9.2250              S
> head(validacion)
  PassengerId Survived
1         892        0
2         893        1
3         894        0
4         895        0
5         896        1
6         897        0
```

Se seleccionan los atributos que van a ser usados y que aportarán información a la hora de hacer el análisis en cada uno de los conjuntos de datos. Para ello, se ha usado el siguiente código:

```
# Observar el tipo de dato que asigna a cada variable R por
   defecto.
tipoEntrenamiento <- sapply(entrenamientoAnalisis,class)
kable(data.frame(variables=names(tipoEntrenamiento),clase=as.
   vector(tipoEntrenamiento)))

tipoTest <- sapply(testAnalisis,class)
kable(data.frame(variables=names(tipoTest),clase=as.vector(
   tipoTest)))

tipoValidacion <- sapply(validacion,class)
kable(data.frame(variables=names(tipoValidacion),clase=as.vector(
   tipoValidacion)))
```

Se han eliminado los siguientes campos dado que no aportan información a la hora de determinar qué personas han fallecido en la travesía:

- PassengerId: el identificador del pasajero no aporta información para determinar si un pasajero ha sobrevivido.
- Name: el nombre del pasajero no aporta información para determinar si un pasajero ha sobrevivido.
- Ticket: el número del ticket que posee el pasajero no aporta información para determinar si un pasajero ha sobrevivido.
- Cabina: la cabina en la que viaja el pasajero no aporta información para determinar si un pasajero ha sobrevivido.

3. Limpieza de los datos

3.1. Análisis tipos de datos

Como primer paso, se analiza el tipo de dato que asigna R automáticamente a cada variable, por si es necesario llevar a cabo alguna modificación de manera manual. El código usado para detectar el tipo de dato es el siguiente:

```
# Observar el tipo de dato que asigna a cada variable R por
   defecto.
tipoEntrenamiento <- sapply(entrenamientoAnalisis,class)
kable(data.frame(variables=names(tipoEntrenamiento),clase=as.
   vector(tipoEntrenamiento)))

tipoTest <- sapply(testAnalisis,class)
kable(data.frame(variables=names(tipoTest),clase=as.vector(
   tipoTest)))

tipoValidacion <- sapply(validacion,class)
kable(data.frame(variables=names(tipoValidacion),clase=as.vector(
   tipoValidacion)))
```

A continuación, se muestra el resultado de cada uno de los tipos de datos que forman parte de los datasets con los que se va a trabajar.

Conjunto de datos de entrenamiento:

```
> tipoEntrenamiento <- sapply(entrenamientoAnalisis,class)
> kable(data.frame(variables=names(tipoEntrenamiento),clase=as.vector(tipoEntrenamiento)))
```

variables	clase
Survived	integer
Pclass	integer
Sex	factor
Age	numeric
SibSp	integer
Parch	integer
Fare	numeric
Embarked	factor

Conjunto de datos de test:

```
> tipoTest <- sapply(testAnalisis,class)
> kable(data.frame(variables=names(tipoTest),clase=as.vector(tipoTest)))
```

```
|variables |clase   |
|:---------|:-------|
|Pclass    |integer |
|Sex       |factor  |
|Age       |numeric |
|SibSp     |integer |
|Parch     |integer |
|Fare      |numeric |
|Embarked  |factor  |
```

Conjunto de datos de validación:

```
> tipoValidacion <- sapply(validacion,class)
> kable(data.frame(variables=names(tipoValidacion),clase=as.vector(tipoValidacion)))
```

```
|variables   |clase   |
|:-----------|:-------|
|PassengerId |integer |
|Survived    |integer |
```

Se puede observar que las variables *Survived* y *Pclass* son variables cualitativas, por lo que es necesario cambiar el tipo de dato para que no se interpreten como valores enteros. El cambio de este tipo de dato se lleva a cabo con la función "as.factor" en R tal y como se observa en el siguiente fragmento de código:

```
# Se cambia el tipo de dato para que se interprete como variable
    categorica
entrenamientoAnalisis$Survived <- as.factor(entrenamientoAnalisis
    $Survived)
entrenamientoAnalisis$Pclass <- as.factor(entrenamientoAnalisis$
    Pclass)
testAnalisis$Pclass <- as.factor(testAnalisis$Pclass)
validacion$Survived <- as.factor(validacion$Survived)
```

Finalmente, para terminar este paso, se puede ejecutar la comprobación anterior sobre el tipo de dato para observar que se ha llevado a cabo correctamente el cambio en los tipos.

Conjunto de datos de entrenamiento:

```
> tipoEntrenamiento <- sapply(entrenamientoAnalisis,class)
> kable(data.frame(variables=names(tipoEntrenamiento),clase=as.vector(tipoEntrenamiento)))
```

variables	clase
Survived	factor
Pclass	factor
Sex	factor
Age	numeric
SibSp	integer
Parch	integer
Fare	numeric
Embarked	factor

Conjunto de datos de test:

```
> tipoTest <- sapply(testAnalisis,class)
> kable(data.frame(variables=names(tipoTest),clase=as.vector(tipoTest)))
```

variables	clase
Pclass	factor
Sex	factor
Age	numeric
SibSp	integer
Parch	integer
Fare	numeric
Embarked	factor

Conjunto de datos de validación:

```
> tipoValidacion <- sapply(validacion,class)
> kable(data.frame(variables=names(tipoValidacion),clase=as.vector(tipoValidacion)))
```

variables	clase
PassengerId	integer
Survived	factor

3.2. Ceros o elementos vacíos

En este apartado se va a dar respuestas a las preguntas planteadas en el enunciado de la práctica: *¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?*

Para responder a estas preguntas, para cada conjunto de datos se va a analizar, si cada variable tiene:

- Valores no informados.
- Valores informados con una cadena de texto vacía.
- Valores 0 sobre aquellos conjuntos de datos que aplique.

El código usado para llevar a cabo el análisis comentado ha sido el siguiente:

```
# Se comprueba los valores que tiene para conjunto de datos como
   0 o vacios
sapply(entrenamientoAnalisis, function(x) sum(is.na(x)))
sapply(entrenamientoAnalisis, function(x) sum(trimws(x) == ""))
sapply(entrenamientoAnalisis, function(x) sum(x == 0))

sapply(testAnalisis, function(x) sum(is.na(x)))
sapply(testAnalisis, function(x) sum(trimws(x) == ""))
sapply(testAnalisis, function(x) sum(x == 0))

sapply(validacion, function(x) sum(is.na(x)))
sapply(validacion, function(x) sum(trimws(x) == ""))
sapply(validacion, function(x) sum(x == 0))
```

El resultado obtenido es el siguiente. Para el análisis de los valores 0 se asume lo siguiente:

- La variable “Survived” puede presentar valores 0 dado que significa que el pasajero no ha sobrevivido.
- La variable “Age” puede presentar valor 0 dado que puede haber algún pasajero registrado que no haya cumplido un año de edad.
- La variable “SibSp” puede presentar valor 0 dado que puede haber algún pasajero que ese sea en número de hermanos o cónyuges a bordo del barco.
- La variable “Parch” puede presentar valor 0 dado que puede haber algún pasajero que ese sea en número de padres o hijos a bordo del barco.
- La variable “Fare” puede presentar valor 0 dado que puede haber algún pasajero que no pague por navegar a bordo del Titanic.

Análisis conjunto de datos de entrenamiento:

```
> # Se comprueba los valores que tiene para conjunto de datos como 0 o vacios
> sapply(entrenamientoAnalisis, function(x) sum(is.na(x)))
Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
       0        0        0      177        0        0        0        0
> sapply(entrenamientoAnalisis, function(x) sum(trimws(x) == ""))
Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
       0        0        0       NA        0        0        0        2
> sapply(entrenamientoAnalisis, function(x) sum(x == 0))
Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
     549        0        0       NA      608      678       15        0
```

Análisis conjunto de datos de test:

```
> sapply(testAnalisis, function(x) sum(is.na(x)))
  Pclass      Sex      Age    SibSp    Parch     Fare Embarked
       0        0       86        0        0        1        0
> sapply(testAnalisis, function(x) sum(trimws(x) == ""))
  Pclass      Sex      Age    SibSp    Parch     Fare Embarked
       0        0       NA        0        0       NA        0
> sapply(testAnalisis, function(x) sum(x == 0))
  Pclass      Sex      Age    SibSp    Parch     Fare Embarked
       0        0       NA      283      324       NA        0
```

Análisis conjunto de datos de validación:

```
> sapply(validacion, function(x) sum(is.na(x)))
PassengerId    Survived
          0           0
> sapply(validacion, function(x) sum(trimws(x) == ""))
PassengerId    Survived
          0           0
> sapply(validacion, function(x) sum(x == 0))
PassengerId    Survived
          0         266
```

Por tanto, las variables que son necesarias tratar son las siguientes:

- La variable “Age” en los conjuntos de datos de validación y test dado que tienen valores no informados.
- La variable “Embarked” en los conjuntos de datos de validación dado que presenta valores con texto vacío.

- La variable “Fare” en los conjuntos de datos de test dado que presenta un valor no informado.

Para tratar a las variable *Age* y *Fare* al tratarse de valores numéricos se ha seleccionado el método de los *k vecinos más cercanos* para tratar aquellos campos que no veían informados. Se ha seleccionado este método debido a que se entiende que las características de los pasajeros pueden tener relación entre sí.

Por otro lado, el valor de *Embarked* al ser de un literal, se ha sustituido por el valor *NA* para tener claro que no aplica. Para ello ha sido necesario crear un valor nuevo al tipo factor de la variable y posteriormente su asignación.

El tratamiento de lo comentado en los dos párrafos anteriores se puede ver en el siguiente fragmento de código:

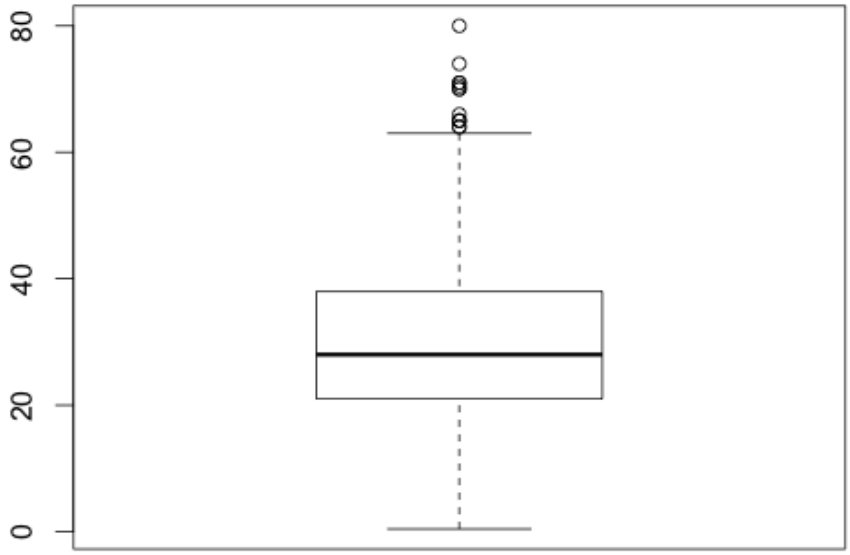
```
# Correccion de los ceros un elementos vacios
entrenamientoAnalisis$Age <- kNN(entrenamientoAnalisis)$Age
testAnalisis$Age <- kNN(testAnalisis)$Age
testAnalisis$Fare <- kNN(testAnalisis)$Fare

# Creacion de una valor nuevo para el valor para Embarked
levels(entrenamientoAnalisis$Embarked) <- c(levels(
   entrenamientoAnalisis$Embarked), "NA")
levels(testAnalisis$Embarked) <- c(levels(testAnalisis$Embarked),
    "NA")

# Asignacion del valor NA en vez del valor vacio
entrenamientoAnalisis$Embarked[entrenamientoAnalisis$Embarked ==
   ""] <- 'NA'
```

3.3. Valores extremos

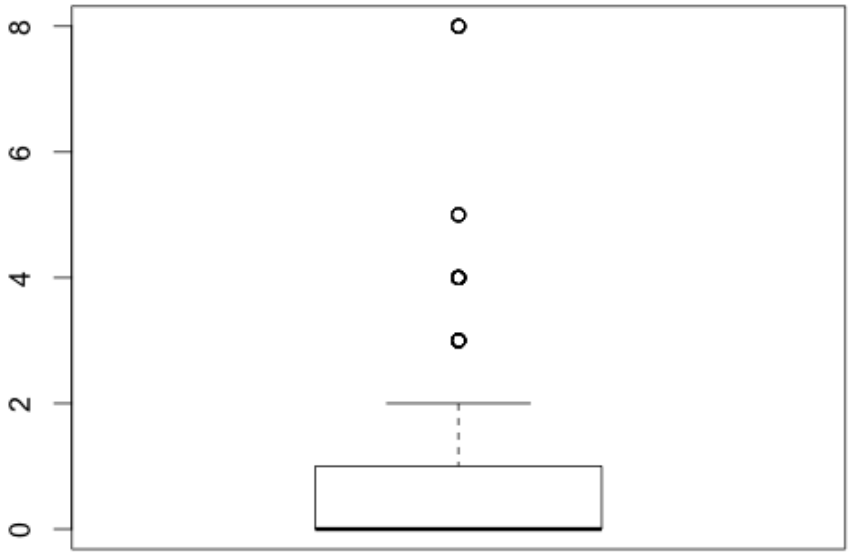
En este apartado se va a realizar un estudio de los valores extremos. Para ello, se van a analizar aquellas variables que tienen valores de tipo *numeric* o *integer*.

El procedimiento para identificar estos valores extremos en R se puede llevar a cabo con la representación de un diagrama de caja para cada variable. Por lo tanto, se va a obtener una representación gráfica de cada tipo para cada una de las variables.

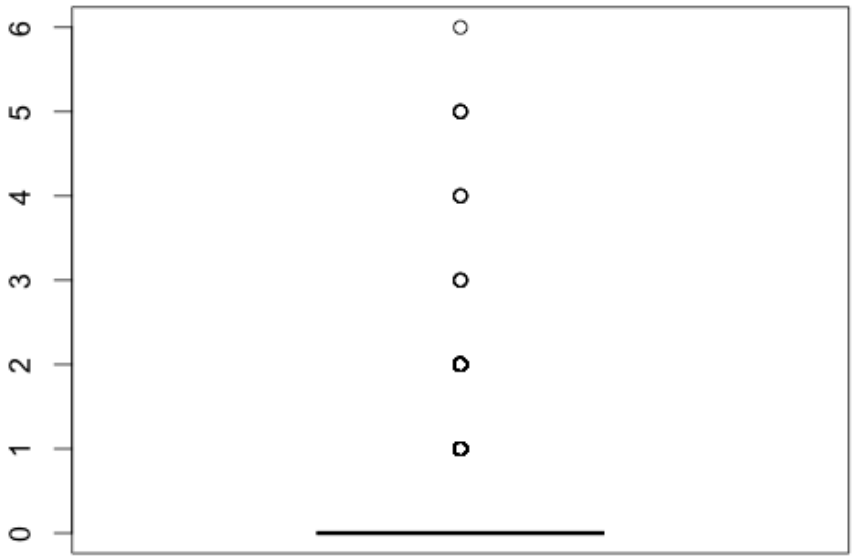
Variable "Age" del conjunto de datos de entrenamiento:



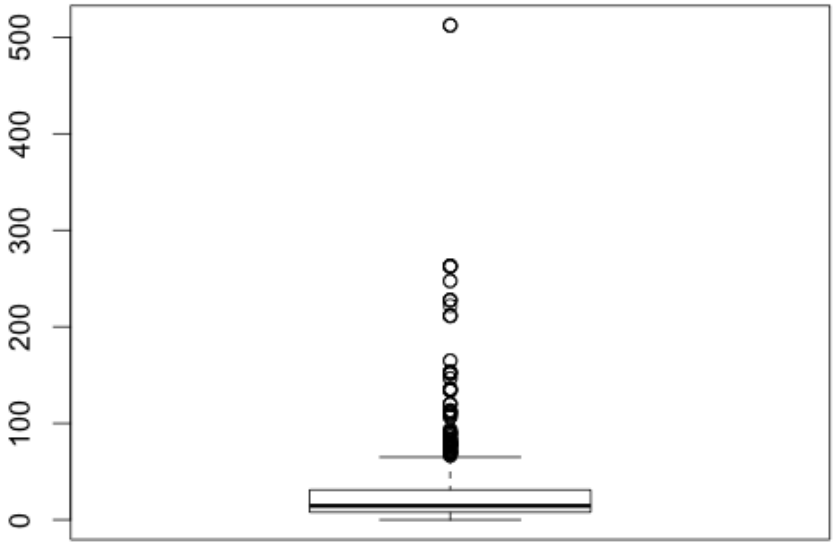
Variable "SibSp" del conjunto de datos de entrenamiento:



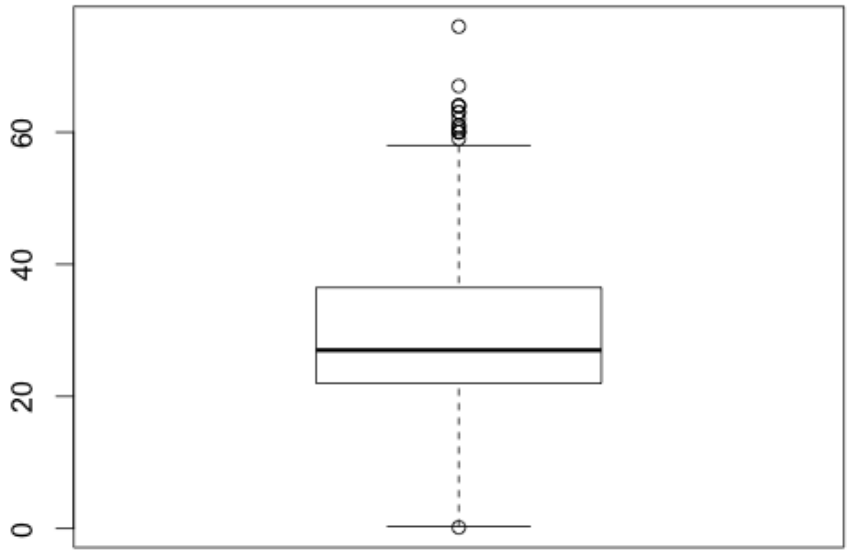
Variable “Parch” del conjunto de datos de entrenamiento:



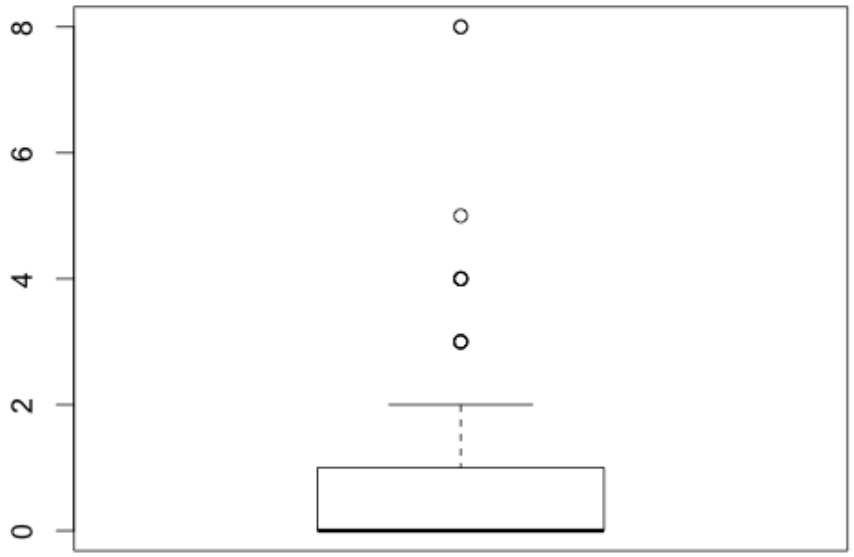
Variable “Fare” del conjunto de datos de entrenamiento:



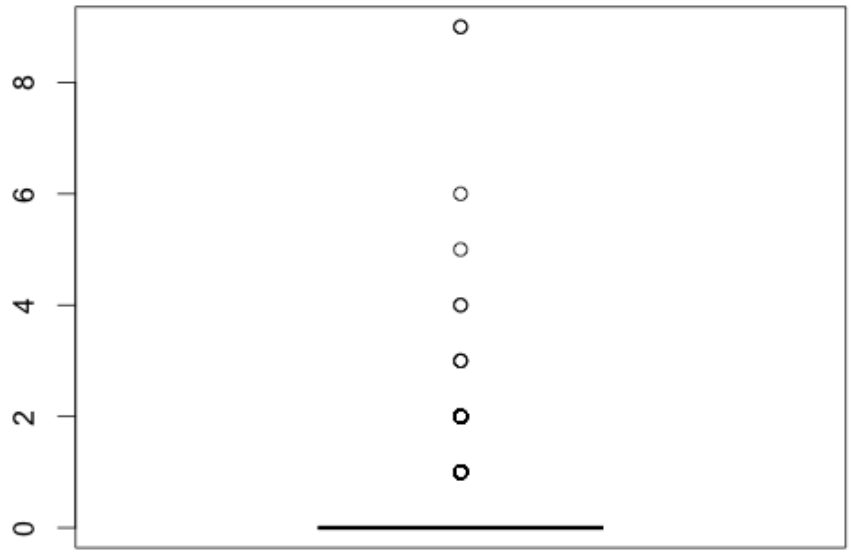
Variable “Age” del conjunto de datos de test:



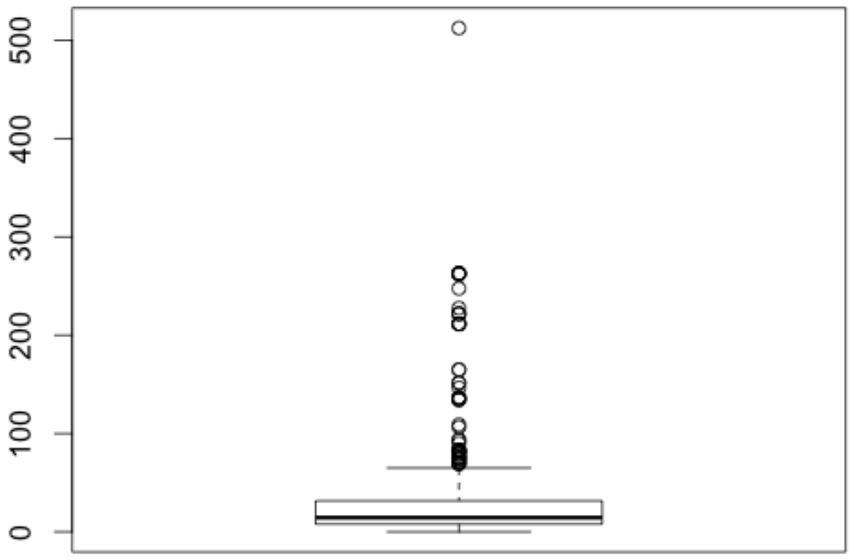
Variable “SibSp” del conjunto de datos de test:



Variable “Parch” del conjunto de datos de test:



Variable “Fare” del conjunto de datos de test:



Analizando estos gráficos lo que se puede observar es que a pesar de tener valores extremos, estos valores tienen sentido ser analizados dado que representan el mismo rango de valores en el conjunto de entrenamiento y de test.

4. Análisis de los datos

4.1. Selección de los grupos de datos

En este apartado, se solicita la *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 con los que se ha estado trabajando hasta el momento, con sus variables ya seleccionadas y tratadas para poder trabajar con ellos de manera eficiente. Por lo tanto, se va a hacer uso de los tres conjuntos de datos en los que se van aplicar tres métodos distintos de clasificación siguiendo estos pasos:

- Se creará un clasificador haciendo uso del conjunto de datos de entrenamiento.
- Este clasificador se aplicará sobre el conjunto de datos de test para clasificar cada registro, es decir, para conocer si pasajero ha sobrevivido o no.
- Se cruzará el resultado del valor de clasificación del conjunto de datos de test indicado en el punto anterior con el valor real contenido en el conjunto de datos de validación, para saber si ha clasificado correctamente y así determinar el porcentaje de acierto.

Para la planificación que se va a llevar a cabo, va a consistir en la aplicación de los tres siguientes métodos de clasificación:

- Clasificación con Árbol de Decisión
- Regresión Logística
- Clasificador Bayesiado Ingenuo

4.2. Comprobación de la normalidad

En el siguiente apartado, uno de los temas que nos pide el enunciado es la comprobación de la normalidad. Para determinar esta normalidad, se va a llevar a cabo el test de Shapiro-Wilk. El test parte de la suposición e 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.

Para esta comprobación se lleva a cabo el siguiente fragmento de código en R:

```
# Comprobacion de la normalidad de los datos
# Para el conjunto de datos de entrenamiento
shapiro.test(entrenamientoAnalisis$Age)
shapiro.test(entrenamientoAnalisis$SibSp)
shapiro.test(entrenamientoAnalisis$Parch)
shapiro.test(entrenamientoAnalisis$Fare)
# Para el conjunto de datos de test
shapiro.test(testAnalisis$Age)
shapiro.test(testAnalisis$SibSp)
shapiro.test(testAnalisis$Parch)
shapiro.test(testAnalisis$Fare)
```

El resultado obtenido para el conjunto de datos de entrenamiento es el siguiente:

```
> shapiro.test(entrenamientoAnalisis$Age)

        Shapiro-Wilk normality test

data:  entrenamientoAnalisis$Age
W = 0.98059, p-value = 1.665e-09

> shapiro.test(entrenamientoAnalisis$SibSp)

        Shapiro-Wilk normality test

data:  entrenamientoAnalisis$SibSp
W = 0.51297, p-value < 2.2e-16

> shapiro.test(entrenamientoAnalisis$Parch)

        Shapiro-Wilk normality test

data:  entrenamientoAnalisis$Parch
W = 0.53281, p-value < 2.2e-16

> shapiro.test(entrenamientoAnalisis$Fare)

        Shapiro-Wilk normality test

data:  entrenamientoAnalisis$Fare
W = 0.52189, p-value < 2.2e-16
```

El resultado obtenido para el conjunto de datos de test es el siguiente:

```
> shapiro.test(testAnalisis$Age)

        Shapiro-Wilk normality test

data:  testAnalisis$Age
W = 0.95456, p-value = 4.737e-10

> shapiro.test(testAnalisis$SibSp)

        Shapiro-Wilk normality test

data:  testAnalisis$SibSp
W = 0.51508, p-value < 2.2e-16

> shapiro.test(testAnalisis$Parch)

        Shapiro-Wilk normality test

data:  testAnalisis$Parch
W = 0.44252, p-value < 2.2e-16

> shapiro.test(testAnalisis$Fare)

        Shapiro-Wilk normality test

data:  testAnalisis$Fare
W = 0.53916, 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 hipótesis nula y entender que no hay normalidad.

4.3. Comprobación de la homogeneidad de varianza

Como se ha llevado ha cabo el estudio de la normalización de los datos y se ha determinado la no existencia de ella, se va a llevar a cabo la comprobación de la homogeneidad de varianza con el test de Leven. Otro test que de podría llevar a cabo en este caso por el mismo motivo es el test no paramétrico Fligner-Killeen. Sin embargo, si las variables estuviesen normalizadas los test aconsejables serían el F-test y el test de Bartlet.

Únicamente la aplicación del test se va a llevar a cabo sobre el conjunto de entrenamiento, dado que la variable que nos interesa analizar es “Survived”, es decir, si afecta a la variable a la supervivencia del pasajero. El fragmento de código usado para la aplicación del test de leven es el siguiente:

```
# Test de Levene para la homogeneidad de varianza
leveneTest(y = entrenamientoAnalisis$Age, group =
   entrenamientoAnalisis$Survived, center = "median")
leveneTest(y = entrenamientoAnalisis$SibSp, group =
   entrenamientoAnalisis$Survived, center = "median")
leveneTest(y = entrenamientoAnalisis$Parch, group =
   entrenamientoAnalisis$Survived, center = "median")
leveneTest(y = entrenamientoAnalisis$Fare, group =
   entrenamientoAnalisis$Survived, center = "median")
```

El resultado obtenido sobre la aplicación de las variables es el siguiente:

```
> leveneTest(y = entrenamientoAnalisis$Age, group = entrenamientoAnalisis$Survived, center = "median")
Levene's Test for Homogeneity of Variance (center = "median")
       Df F value Pr(>F)
group   1  0.3276 0.5672
      889
> leveneTest(y = entrenamientoAnalisis$SibSp, group = entrenamientoAnalisis$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 = entrenamientoAnalisis$Parch, group = entrenamientoAnalisis$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 = entrenamientoAnalisis$Fare, group = entrenamientoAnalisis$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 los dos grupos que son “Age” y “Sipn”, en cambio si las encuentra sobre “Parch” y “Fare”.

4.4. Normalización de los datos

Debido a los estudios llevados a cabo en los puntos anterior, es necesario aplicar una normalización en los datos. Para ello, se crea la siguiente función en R, donde los valores de cada variable a aplicar serán ahora entre 0 y 1, donde el 0 es el valor más bajo y 1 el valor más alta. El código es el siguiente:

```
# Funcion para la normalizacion de los datos
normalize <- function(x) {
        return ((x - min(x)) / (max(x) - min(x)))
}
```

Finalmente, se aplica la función sobre cada una de las variables de los conjuntos de datos:

```
#Normalizacion de los datos
# Para el conjunto de datos de entrenamiento
entrenamientoAnalisis$Age <- normalize(entrenamientoAnalisis$Age)
entrenamientoAnalisis$SibSp <- normalize(entrenamientoAnalisis$
   SibSp)
entrenamientoAnalisis$Parch <- normalize(entrenamientoAnalisis$
   Parch)
entrenamientoAnalisis$Fare <- normalize(entrenamientoAnalisis$
   Fare)
# Para el conjunto de datos de test
testAnalisis$Age <- normalize(testAnalisis$Age)
testAnalisis$SibSp <- normalize(testAnalisis$SibSp)
testAnalisis$Parch <- normalize(testAnalisis$Parch)
testAnalisis$Fare <- normalize(testAnalisis$Fare)
```

4.5. Aplicación pruebas estadísticas

Para la aplicación de las pruebas estadísticas, tal y como se ha comentado, se va a aplicar tres métodos de análisis diferentes:

- Clasificación con Árbol de Decisión
- Regresión Logística
- Clasificador Bayesiado Ingenuo

4.5.1. Clasificación con Árbol de Decisión

Este primer método, es la clasificación mediante árbol de decisión. Se trata de modelos que pueden ser usados para predicción y cuyo funcionamiento se basa en la construcción de reglas lógicas (divisiones de los datos entre rangos o condiciones) a partir de los datos de entrada.

El entrenamiento de los árboles de decisión se centra principalmente en la maximización de la ganancia de información al momento de realizar las reglas lógicas que forman el árbol. Para construir el modelo se usará la función *rpart* del paquete *rpart* para crear el árbol de decisión:

```
# Arbol de decision

# Se establece una semilla para la generacion de numeros
   aleatorios
set.seed(1234)

# Construccion del modelo
clasifcadorAD <-rpart(Survived ~ ., data = entrenamientoAnalisis)
```

4.5.2. Regresión Logística

La Regresión Logística es un tipo de modelo de regresión. Es empleado para predecir variables categóricas. En términos generales, calcula las probabilidades de ocurrencia de alguna de las clases del modelo a partir del uso de la función logística.

Para crear y entrenar el modelo de regresión logística, se va a hacer uso de la función *glm* del paquete *stats*:

```
# Regresion logistica

# Construccion del modelo
clasificadorRL <- glm(Survived ~ ., family = binomial, data =
```

```
    entrenamientoAnalisis)
```

4.5.3. Clasificador Bayesiado Ingenuo

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

Para crear y entrenar el clasificador Bayesiano se hace uso de la función *naiveBayes* del paquete *e1071*:

```
# Clasificador Bayesiano Ingenuo

# Se establece una semilla para la generacion de numeros
    aleatorios
set.seed(1234)

# Construccion del modelo
clasificadorBayes <- naiveBayes(Survived ~ ., data =
    entrenamientoAnalisis)
```

5. Aplicación resultados

En este apartado se solicita la *representación de los resultados a partir de tablas y gráficas*. Para llevar a cabo este análisis, se van a detallar los resultados de cada uno de los métodos aplicados en el apartado anterior.

Para el análisis se van a realizar los siguientes pasos:

- Llevar a cabo la predicción sobre el conjunto de datos de test haciendo uso de cada uno de los clasificadores con la función *predict* de R.
- Se calcula el éxito de la predicción. Para ello, se realiza un cálculo sobre la predicción obtenida para elemento en el paso anterior comparándolo con el resultado real que

se tiene en la variable *Survived* del conjunto de datos de validación. Dividiendo los casos coincidentes sobre el total de casos, se obtiene el porcentaje de acierto en la clasificación.

- Se crea la matriz de confusión en formato de tabla y de manera gráfica. Esta matriz es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Su objetivo es que facilitan ver si el sistema está confundiendo dos clases. Para llevar a cabo este análisis, se construye una taba con el resultado obtenido de la predicción y el valor real y se hace uso de la función de R *fourfoldplot* para la construcción de la representación gráfica.

5.1. Clasificación con Árbol de Decisión

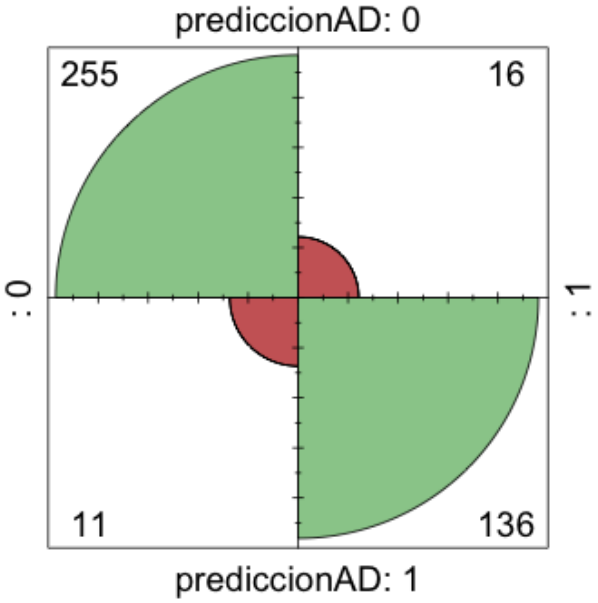
El código usado para el análisis de la predicción sobre la clasificación del árbol de decisión es la siguiente:

```
# Prediccion de la clasificacion en el conjunto de test
prediccionAD <- predict(clasifcadorAD, newdata = testAnalisis,
   type = 'class')

# Analisis de la clasificacion
exitoPrediccionAD <- sum(prediccionAD == validacion$Survived)/
   length(prediccionAD)
exitoPrediccionAD
tablePrediccionAD <- table(prediccionAD, validacion$Survived)
tablePrediccionAD
fourfoldplot(tablePrediccionAD, color = c("#CC6666", "#99CC99"),
   conf.level = 0, margin = 1, main = "Matriz de confusion")

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

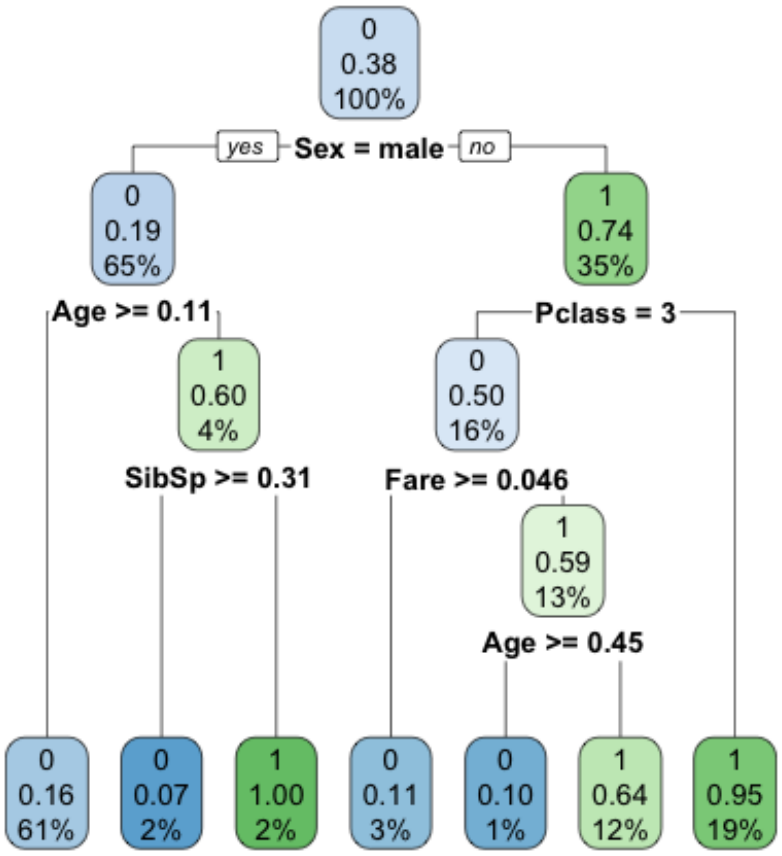
El resultado obtenido es el que se aprecia en la siguiente imagen:

```
> exitoPrediccionAD
[1] 0.9354067
> tablePrediccionAD <- table(prediccionAD, validacion$Survived)
> tablePrediccionAD

prediccionAD   0   1
           0 255  16
           1  11 136
```

Se puede observar un éxito en la predicción de 93,54 %. La representación gráfica de la matriz de confusión es la siguiente:



En el siguiente gráfico se puede observar, el árbol que se ha construido para la toma de decisiones, teniendo en cuenta cual es la variable que más afecta a la decisión y como se llega a la clasificación final.



5.2. Regresión Logística

El código usado para el análisis de la predicción sobre la regresión logística es la siguiente:

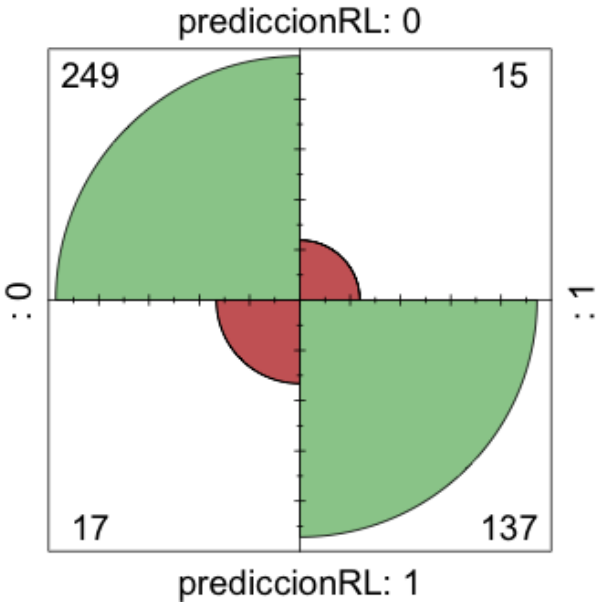
```
# Prediccion de la clasificacion en el conjunto de test
prediccionRL <- predict(clasificadorRL, type = 'response',
   newdata = testAnalisis)
prediccionRL <- ifelse(prediccionRL > 0.5, 1, 0)
prediccionRL <- as.factor(prediccionRL)

# Analisis de la clasificacion
exitoPrediccionRL <- sum(prediccionRL == validacion$Survived)/
   length(prediccionRL)
exitoPrediccionRL
tablePrediccionRL <- table(prediccionRL, validacion$Survived)
```

```
tablePrediccionRL
fourfoldplot(tablePrediccionRL, color = c("#CC6666", "#99CC99"),
    conf.level = 0, margin = 1, main = "Matriz de confusion")
```

El resultado obtenido es el que se aprecia en la siguiente imagen:

```
> exitoPrediccionRL
[1] 0.923445
> tablePrediccionRL <- table(prediccionRL, validacion$Survived)
> tablePrediccionRL

prediccionRL   0   1
           0 249  15
           1  17 137
```

Se puede observar un éxito en la predicción de 92,34 %. La representación gráfica de la matriz de confusión es la siguiente:



5.3. Clasificador Bayesiado Ingenuo

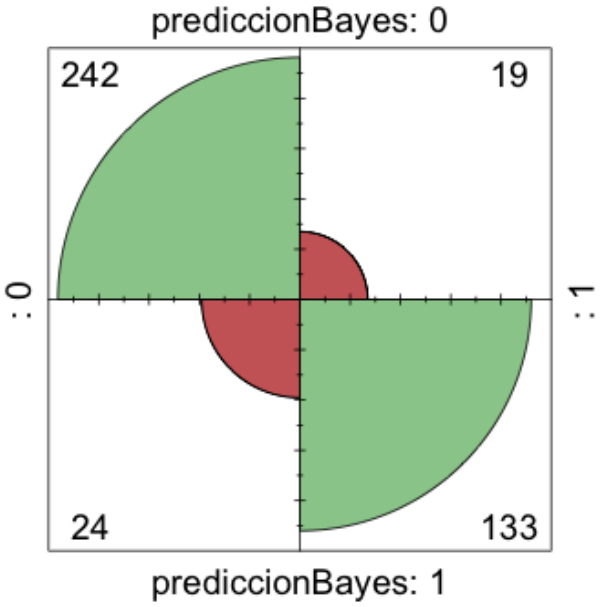
El código usado para el análisis de la predicción sobre el clasificador Bayesiano Ingenuo es la siguiente:

```
# Prediccion de la clasificacion en el conjunto de test
prediccionBayes <- predict(clasificadorBayes, newdata =
    testAnalisis)

# Analisis de la clasificacion
exitoPrediccionBayes <- sum(prediccionBayes == validacion$
    Survived)/length(prediccionBayes)
exitoPrediccionBayes
tablePrediccionBayes <- table(prediccionBayes, validacion$
    Survived)
tablePrediccionBayes
fourfoldplot(tablePrediccionBayes, color = c("#CC6666", "#99CC99"
    ), conf.level = 0, margin = 1, main = "Matriz de confusion")
```

El resultado obtenido es el que se aprecia en la siguiente imagen:

```
> exitoPrediccionBayes
[1] 0.8971292
> tablePrediccionBayes <- table(prediccionBayes, validacion$Survived)
> tablePrediccionBayes

prediccionBayes   0   1
              0 242  19
              1  24 133
```

Se puede observar un éxito en la predicción de 89,71 %. La representación gráfica de la matriz de confusión es la siguiente:



5.4. Exportación

Finalmente, tal y como se pide en la práctica, se exportan los datos a ficheros en formato *csv*. Como el árbol de decisión fue en el que mejor porcentaje de predicción obtuvo, se añade el valor de los resultados obtenidos a una nueva columna en conjunto de datos de entrenamiento. El código generado es el siguiente:

```
# Exportacion de los datos
# Se incluye la prediccion generada por el arbol de decision
testAnalisis$Survived <- prediccionAD
# Se almacenan los datos en csv
write.csv(entrenamientoAnalisis, "/Users/miguelpr93/Desktop/
   titanic/trainExport.csv", row.names = FALSE)
write.csv(testAnalisis, "/Users/miguelpr93/Desktop/titanic/
   testExport.csv", row.names = FALSE)
write.csv(validacion, "/Users/miguelpr93/Desktop/titanic/
   validacionExport.csv", row.names = FALSE)
```

6. Resolución del problema

En este apartado, se plantea como pregunta final la siguiente cuestión: *"A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?"*

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

- Todas las técnicas de preparación de los datos previas a la construcción de los modelos son necesarias para que las técnicas de construcción de los modelos que se apliquen sean correctas y efectivas y no se desvirtúen los resultados finales.
- Una vez aplicadas las técnicas, la construcción de los modelos aplicando las librerías y funciones de R dan los resultados que son necesarios analizar. Cada técnica tiene sus ventajas y limitaciones y es esto lo que hay que analizar para obtener el modelo más óptimo para dar respuesta a la pregunta planteada.
- Lo que se quería lograr es la construcción de un modelo que ayude a predecir aquellos pasajeros que había o no sobrevivido al hundimiento del Titanic en base a algunas de sus características tanto individuales como de travesía en el crucero.
- Finalmente, se puede determinar que el modelo que mejor ha clasificado de los tres analizados en el de clasificación con árbol de decisión, dado que es el que tiene un porcentaje de predicción más elevado con 93,54 % de los casos.

7. Firma integrantes

En este apartado se recoge la firma de los integrantes que han hecho cada uno de las tareas de la práctica.

Contribuciones	Firma
Investigación previa	miguelpre
Redacción de las respuestas	miguelpre
Desarrollo código	miguelpre