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

David Poggio Moro

6/12/2020

1 Descripción del Dataset.

Para esta práctica utilizaremos el dataset titulado “Titanic: Machine Learning from Disaster” de Kaggle, que podemos encontrar en este enlace: https://www.kaggle.com/c/titanic/data (https://www.kaggle.com/c/titanic/data)

Los campos de los que disponemos en este juego de datos son los siguientes:

- PassengerId: Identificador único por pasajero.
- Survived: Indica si el pasajero sobrevivió (1) o no (0).
- Pclass: Clase del billete del pasajero. Primera (1), segunda (2) o tercera (3).
- Name: Nombre del pasajero.
- Sex: Género del pasajero.
- Age: Edad del pasajero.
- SibSp: Número de hermanos o cónyujes a bordo (movimiento horizontal en el árbol genealógico).
- Parch: Número de padres o hijos a bordo (movimiento vertical en el árbol genealógico).
- Ticket: Número de billete.
- Fare: Tarifa del billete.
- Cabin: Número de camarote.
- Embarked: Lugar de embarque. Cherbourg (C) , Queenstown (Q) o Southampton (S).

Los datos está repartidos en dos ficheros:

- Train.csv: Fichero de entrenamiento que incluye todos los atributos mencionados antes.
- Test.csv: Fichero de prueba que incluye todos los atributos menos “Survived”, ya que es el que queremos estimar.

El objetivo es llegar a la creación de un conjunto de reglas que permitan estimar, en base a los atributos de los que disponemos, si un individuo sobreviviría o no.

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

Empezaremos cargando los datos de los ficheros en unos dataframes:

```
df_train <- read.csv(".\\Titanic\\train.csv")
df_test <- read.csv(".\\Titanic\\test.csv")
```

Una vez cargados los ficheros, hacemos una vista preliminar descriptiva de los datos:

```
summary(df_train)
```

```
##   PassengerId       Survived          Pclass
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000
##  Median :446.0   Median :0.0000   Median :3.000
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000
##
##                                     Name         Sex           Age
##  Abbing, Mr. Anthony                   :  1   female:314   Min.   : 0.42
##  Abbott, Mr. Rossmore Edward           :  1   male  :577   1st Qu.:20.12
##  Abbott, Mrs. Stanton (Rosa Hunt)      :  1                Median :28.00
##  Abelson, Mr. Samuel                   :  1                Mean   :29.70
##  Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00
##  Adahl, Mr. Mauritz Nils Martin        :  1                Max.   :80.00
##  (Other)                               :885                NA's   :177
##      SibSp           Parch             Ticket         Fare
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33
##                                   (Other) :852
##          Cabin     Embarked
##             :687    :  2
##  B96 B98    :  4   C:168
##  C23 C25 C27:  4   Q: 77
##  G6         :  4   S:644
##  C22 C26    :  3
##  D          :  3
##  (Other)    :186
```

Como paso previo a la limpieza tratremos de reducir la dimensionalidad de nuestros datos. Para ello proponemos las siguientes acciones:

- Combinar los atributos “SibSp” y “Parch” en un único atributo “Familiares” que nos muestre si la persona viajaba con familiares o no.
- Descartar el atributo del “Name” puesto que no nos aportará nada porque podemos identificar a los pasajeros con su “PassengerID”.
- Por último, eliminaremos también los atributos de “Ticket” y “Cabin”, ya que podrían contribuir a un posible overfitting del modelo.

```
require(dplyr)
df_train$Familiares <- ifelse(df_train$SibSp > 0 | df_train$Parch>0,"Si","No")
df_test$Familiares <- ifelse(df_test$SibSp > 0 | df_test$Parch>0,"Si","No")
df_train_clean <- select(df_train,c("PassengerId","Age","Sex","Pclass","Survived","Fare","Emb
arked","Familiares"))
```

También convertiremos la columna “Survived” en una variable categórica:

```
df_train_clean$Survived <- factor(df_train_clean$Survived)
levels(df_train_clean$Survived)=c("No","Si")
```

3 Limpieza de los datos

3.1 Gestión de valores perdidos y vacíos

Buscamos los valores perdidos de nuestro dataset por columnas:

```
sapply(df_train_clean, function(x) sum(is.na(x)))
```

```
## PassengerId         Age         Sex      Pclass    Survived        Fare
##           0         177           0           0           0           0
##    Embarked  Familiares
##           0           0
```

Vemos que tenemos valores perdidos en la edad. Este dato nos es relevante para la creación de reglas en nuestro modelo por lo que no debemos eliminar los 177 registros porque es un alto porcentaje de las muestras de nuestro set de entrenamiento.En vez de esto, para no introducir ruido, reemplazaremos estos valores perdidos con la mediana ya que es un estimador robuesto para el ajuste de la edad.

```
df_train_clean$Age[is.na(df_train_clean$Age)] <- median(df_train_clean$Age, na.rm=TRUE)
```

Aprovechando que estamos operando con la columna edad, añadiremos una nueva que nos distinga entre mayores y menores de edad.

```
df_train_clean$Adulto <- ifelse(df_train_clean$Age < 18, "Menor", "Adulto")
df_test$Adulto <- ifelse(df_test$Age < 18, "Menor", "Adulto")
```

Buscamos ahora valores vacíos:

```
colSums(df_train_clean=="")
```

```
## PassengerId         Age         Sex      Pclass    Survived        Fare
##           0           0           0           0           0           0
##    Embarked  Familiares      Adulto
##           2           0           0
```

Podemos ver que tenemos dos valores vacíos en el campo “Embarked”. En este caso es un porcentaje ínfimo de los registros que tenemos por lo que podemos descartarlos directamente.

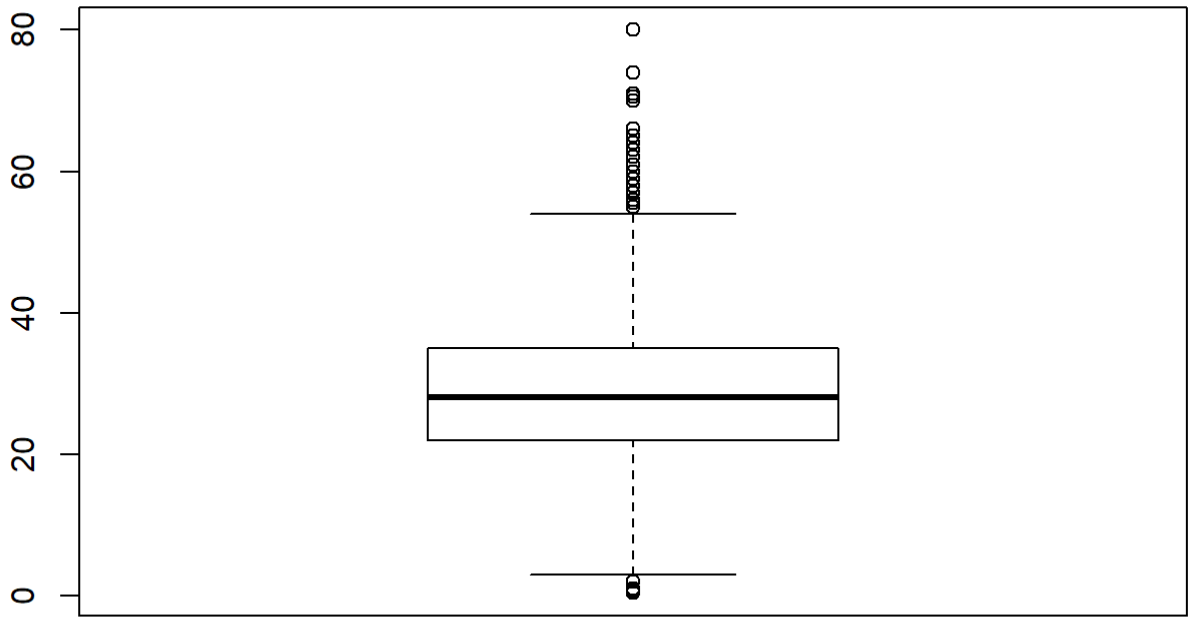
```
df_train_clean <- subset(df_train_clean,Embarked!="")
colSums(df_train_clean=="")
```

```
## PassengerId         Age         Sex      Pclass    Survived        Fare
##           0           0           0           0           0           0
##    Embarked  Familiares      Adulto
##           0           0           0
```

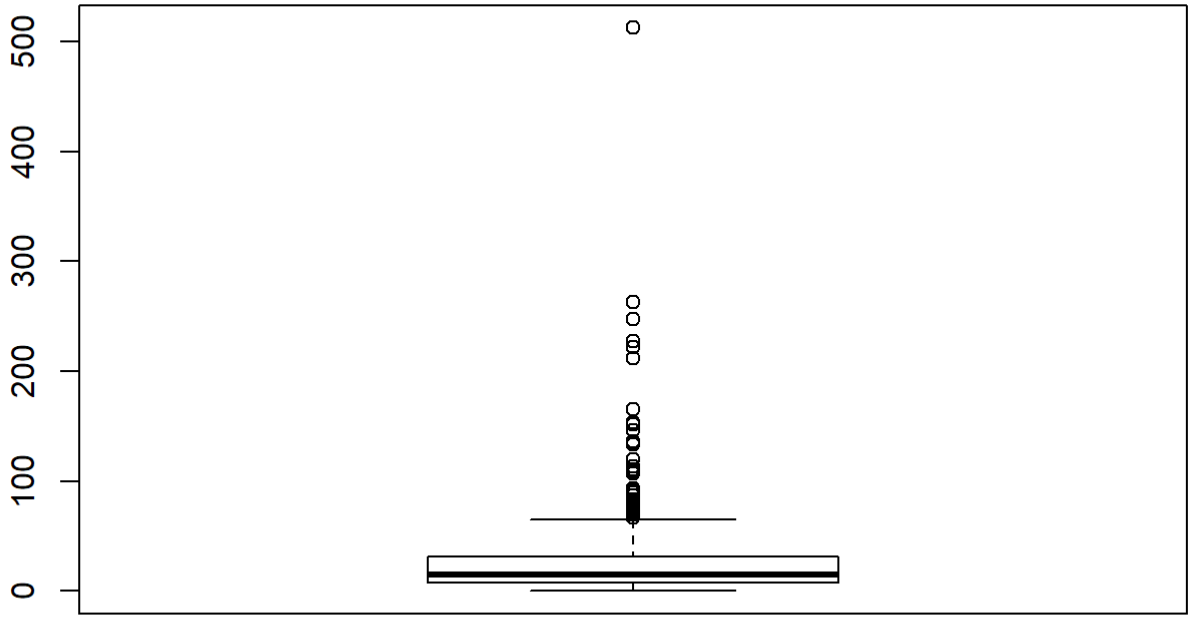
3.2 Identificación y tratamiento de valores extremos

Representaremos mediante un boxplot las variables cuantitativas

```
boxplot(df_train_clean$Age)
```



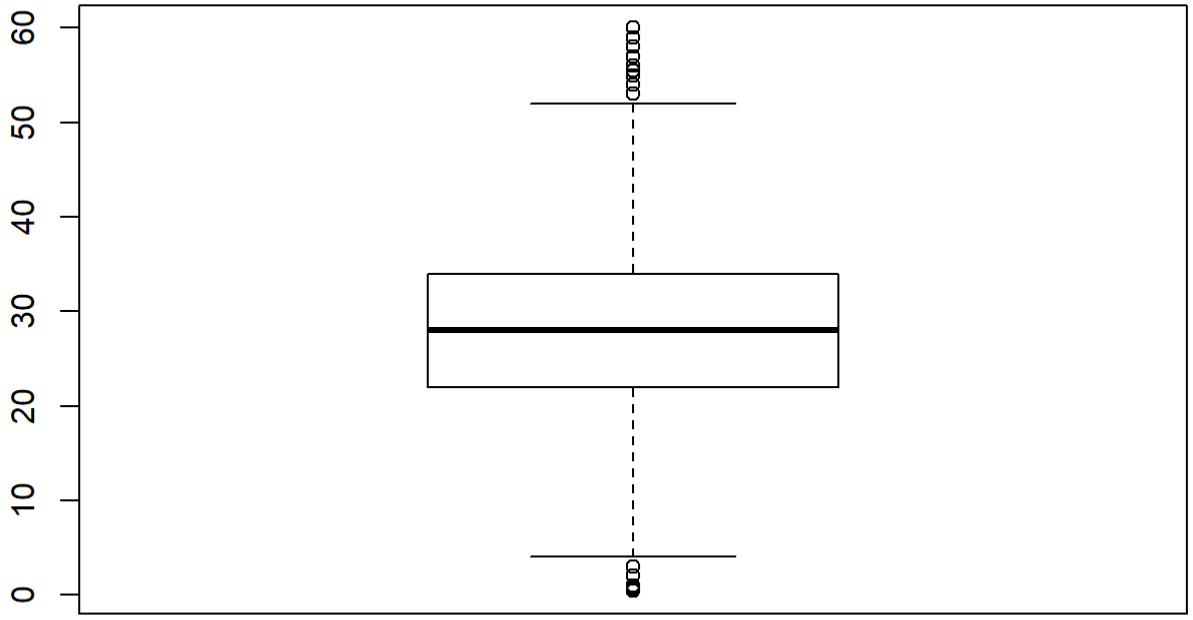

```
boxplot(df_train_clean$Fare)
```



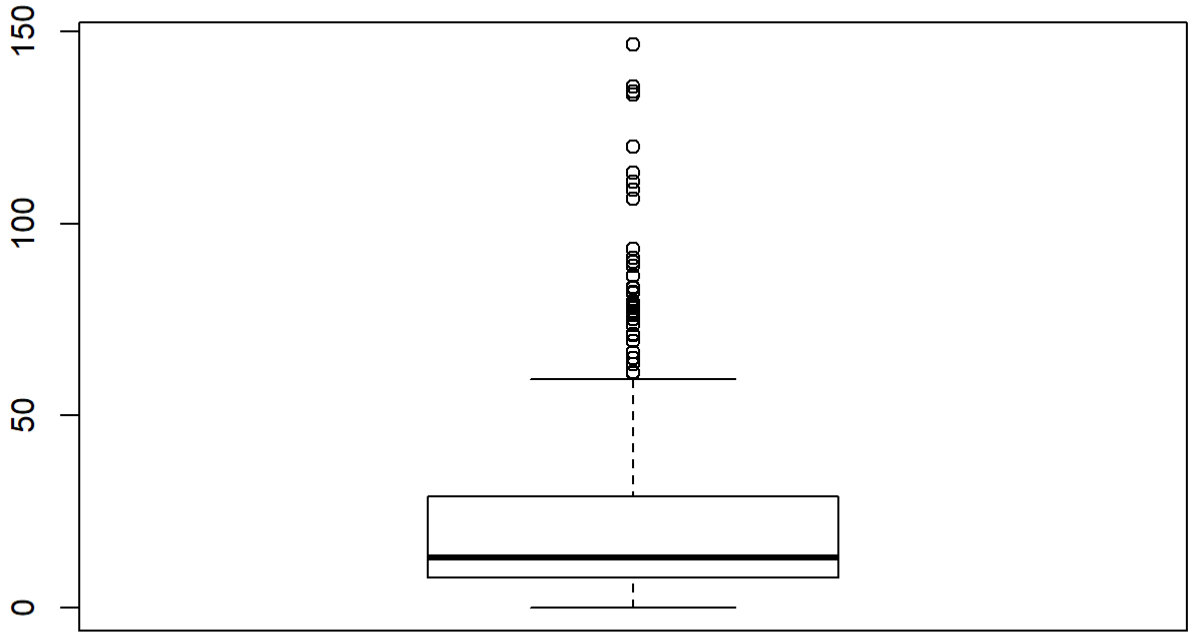
Viendo los diagramas de caja, estableceremos unos criterios para evitar que los outliers puedan afectar a las reglas del modelo:

- La edad podrá ser cómo máximo 60 años.
- El precio del billete será como máximo de 150 dólares.

```
df_train_clean <- subset(df_train_clean, Age <=60 & Fare <= 150)
boxplot(df_train_clean$Age)
```




```
boxplot(df_train_clean$Fare)
```



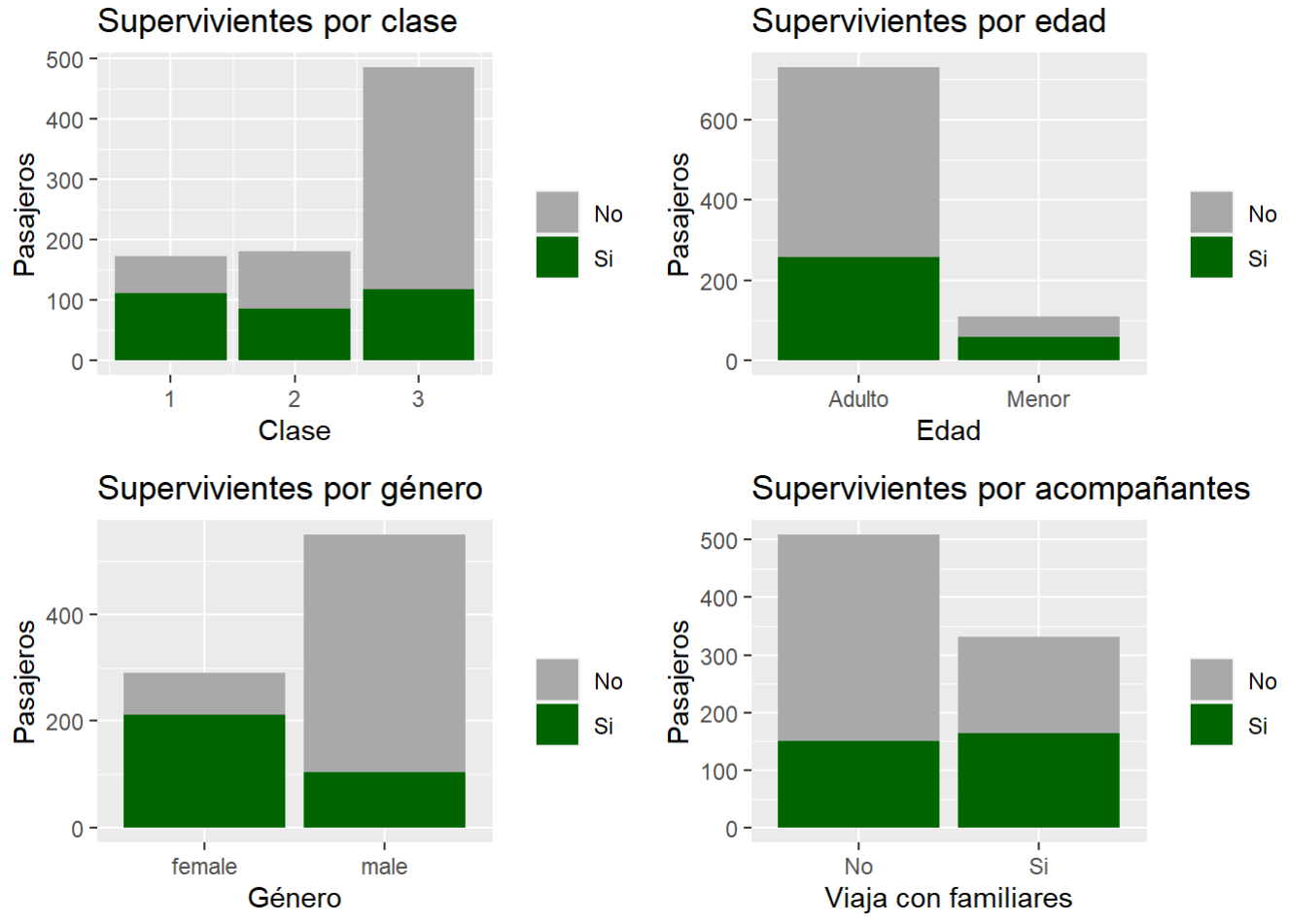
4 Análisis de los datos

4.1 Planificación del análisis

Vamos a buscar una relación entre la supervivencia de los pasajeros y el resto de atributos que disponemos. Para ello representaremos gráficamente algunos valores para intentar identificar estas correlaciones de forma intuitiva.

```
require(ggplot2)
require(grid)
require(gridExtra)

grid.newpage()
graficaClase<-ggplot(df_train_clean,aes(Pclass,fill=Survived))+geom_bar() +labs(x="Clase", y=
"Pasajeros")+ guides(fill=guide_legend(title=""))+ scale_fill_manual(values=c("darkgrey","dar
kgreen"))+ggtitle("Supervivientes por clase")
graficaEdad<-ggplot(df_train_clean,aes(Adulto,fill=Survived))+geom_bar() +labs(x="Edad", y="P
asajeros")+ guides(fill=guide_legend(title=""))+ scale_fill_manual(values=c("darkgrey","darkg
reen"))+ggtitle("Supervivientes por edad")
graficaGenero<-ggplot(df_train_clean,aes(Sex,fill=Survived))+geom_bar() +labs(x="Género", y=
"Pasajeros")+ guides(fill=guide_legend(title=""))+ scale_fill_manual(values=c("darkgrey","dar
kgreen"))+ggtitle("Supervivientes por género")
graficaFamilia<-ggplot(df_train_clean,aes(Familiares,fill=Survived))+geom_bar() +labs(x="Viaj
a con familiares", y="Pasajeros")+ guides(fill=guide_legend(title=""))+ scale_fill_manual(val
ues=c("darkgrey","darkgreen"))+ggtitle("Supervivientes por acompañantes")
grid.arrange(graficaClase,graficaEdad,graficaGenero,graficaFamilia,ncol=2)
```



A la luz de estas gráficas podemos intuir las siguientes posibles relaciones:

- La clase y la proporción de supervivientes.
- El sexo y la proporción de supervivientes.
- La edad y la proporción de supervivientes.
- Viajar con familiares y la proporción de supervivientes.

Para aterrizar estas visualizaciones, obtenemos las tablas de contingencia ver las proporciones exactas.

```
tabla_Genero <- table(df_train_clean$Sex, df_train_clean$Survived)
prop.table(tabla_Genero, margin = 1)
```

```
##
##                 No        Si
##   female 0.2714777 0.7285223
##   male   0.8105647 0.1894353
```

```
tabla_Clase <- table(df_train_clean$Pclass, df_train_clean$Survived)
prop.table(tabla_Clase, margin = 1)
```

```
##
##            No        Si
##   1 0.3526012 0.6473988
##   2 0.5248619 0.4751381
##   3 0.7572016 0.2427984
```

```
tabla_Adulto <- table(df_train_clean$Adulto, df_train_clean$Survived)
prop.table(tabla_Adulto, margin = 1)
```

```
##
##                 No        Si
##   Adulto 0.6479452 0.3520548
##   Menor  0.4636364 0.5363636
```

```
tabla_Familiares <- table(df_train_clean$Familiares, df_train_clean$Survived)
prop.table(tabla_Familiares, margin = 1)
```

```
##
##             No        Si
##   No 0.7033399 0.2966601
##   Si 0.5015106 0.4984894
```

4.2 Comprobación de la normalidad y homogeneidad de la varianza

Utilizaremos el test de Shapiro, donde planteamos un contraste de hipotesis para discernir si la distribución a analizar sigue una distribución normal. La hipotesis nula será que nos encontramos ante una distribución normal y la hipotesis alternativa por tanto será que no seguimos una distribución normal.

Para esta prueba usaremos un nivel de significancia (α) de 0.01.

```
shapiro.test(df_train_clean$Age)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_train_clean$Age
## W = 0.96248, p-value = 7.42e-14
```

```
shapiro.test(df_train_clean$Fare)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df_train_clean$Fare
## W = 0.71843, p-value < 2.2e-16
```

Se puede observar que el p-valor que obtenemos en el resultado del test para la edad y el precio del billete es mucho menor que el nivel de significancia de 0.01, por lo que rechazamos la hipotesis nula en favor de la hipotesis alternativa: no se sigue una distribución normal, por lo que se deberán usar tests no paramétricos en el futuro.

Respecto a la varianza, aplicaremois el test de Fligner:

```
fligner.test(Age ~ Survived, data=df_train_clean)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Age by Survived
## Fligner-Killeen:med chi-squared = 9.8871, df = 1, p-value = 0.001664
```

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

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fare by Survived
## Fligner-Killeen:med chi-squared = 81.399, df = 1, p-value < 2.2e-16
```

Vemos que, por poco, podemos rechazar la hipóteis nula y asumir que nos encontramos ante un conjunto con varianza homogénea.

4.3 Aplicación de pruebas estadísticas para comparar grupos de datos

A continuación utilizaremos el test de Wilcox para estudiar las posibles diferencias significativas en la varianza según la supervivencia para los atributos de los que disponemos. Esta será nuestra hipotesis nula.

```
wilcox.test(Age ~ Survived, data = df_train_clean)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Age by Survived
## W = 86423, p-value = 0.2832
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(Fare ~ Survived, data = df_train_clean)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Fare by Survived
## W = 51833, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Vemos que si mantenemos el nivel de significancia de 0.01 rechazamos la hipotesis nula. Por tanto, no existen diferencias significativas con respecto a la varianza en funcion de la edad de cara a analizar la superviencia del pasajero.

Sin embargo, si parece existir una diferencia de varianza significativa para la supervivencia en función del precio del billete, por lo que aceptamos la hipotesis nula y asumimos una relación entre ambas.

Aplicaremos también el test de Kruskal, que es similar al test de Wilcox pero admite más de dos grupos.

```
kruskal.test(Age ~ Survived, data = df_train_clean)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Age by Survived
## Kruskal-Wallis chi-squared = 1.152, df = 1, p-value = 0.2831
```

```
kruskal.test(Fare ~ Survived, data = df_train_clean)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Fare by Survived
## Kruskal-Wallis chi-squared = 82.636, df = 1, p-value < 2.2e-16
```

Tal y como esperabamos, los resultados coinciden con los proporcionados por el test de Wilcox.

Por último, podemos realizar un test χ^2 para encontrar diferencias significativas entre grupos de variables categóricas. En este caso, la hipotesis nula es que hay independencia entre los atributos, y la hipotesis alternativa es que existe una dependencia entre los atributos.

```
chisq.test(table(df_train_clean$Adulto,df_train_clean$Survived))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(df_train_clean$Adulto, df_train_clean$Survived)
## X-squared = 13.064, df = 1, p-value = 0.0003011
```

```
chisq.test(table(df_train_clean$Familiares,df_train_clean$Survived))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(df_train_clean$Familiares, df_train_clean$Survived)
## X-squared = 33.961, df = 1, p-value = 5.623e-09
```

Tenemos un p-valor menor que 0.01 en ambos casos por lo que rechazamos la hipotesis nula y por lo tanto tenemos evidencia de que existe una relación entre la supervivencia y ser menor/mayor de edad o viajar con familia.

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

En apartados anteriores se realizó este análisis de tablas y gráficas. Por comodidad, volveremos a mostrarlas aquí:

```
prop.table(tabla_Genero, margin = 1)
```

```
##
##                No        Si
##   female 0.2714777 0.7285223
##   male   0.8105647 0.1894353
```

```
prop.table(tabla_Clase, margin = 1)
```

```
##
##           No        Si
##   1 0.3526012 0.6473988
##   2 0.5248619 0.4751381
##   3 0.7572016 0.2427984
```

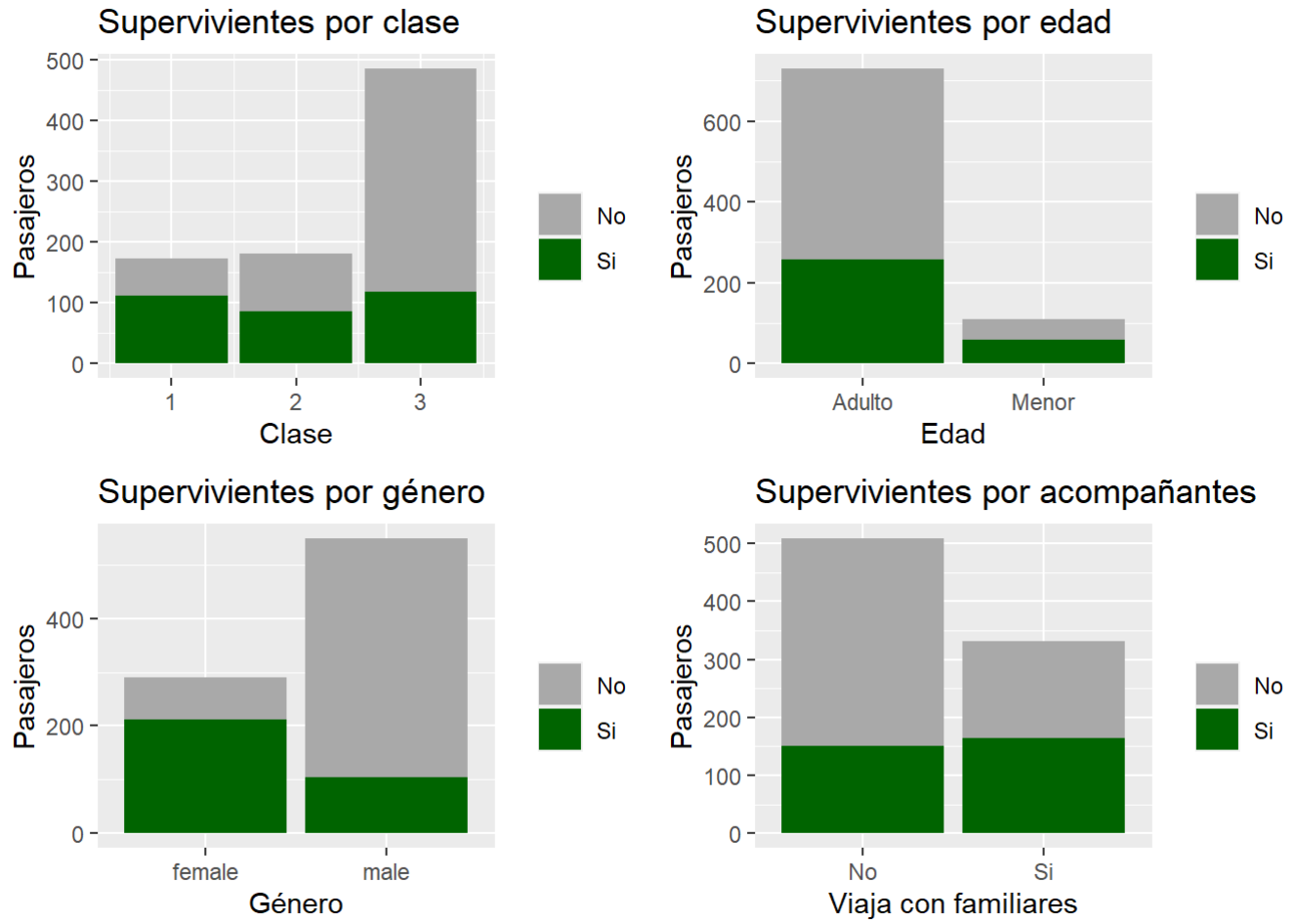
```
prop.table(tabla_Adulto, margin = 1)
```

```
##
##                No        Si
##   Adulto 0.6479452 0.3520548
##   Menor  0.4636364 0.5363636
```

```
prop.table(tabla_Familiares, margin = 1)
```

```
##
##             No        Si
##   No 0.7033399 0.2966601
##   Si 0.5015106 0.4984894
```

```
grid.arrange(graficaClase,graficaEdad,graficaGenero,graficaFamilia,ncol=2)
```



6 Resolución del problema

Vistas estas correlaciones entre atributos y la supervivencia de pasajeros parece razonable buscar reglas que nos ayuden a modelar esta supervivencia.

6.1 Creación de los sets de entrenamiento y test

Randomizamos los datos disponibles en el dataframe y utilizamos el conjunto de entrenamiento.

```
train_random <- df_train_clean[sample(nrow(df_train_clean)),]
y <- train_random[,5]
X <- train_random[,c(3,4,9)]
```

```
trainX<-X
trainy<-y
testX<-df_test[,c(2,4,13)]
```

6.2 Creación del arbol de decisión

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

```
##
## Call:
## C5.0.default(x = trainX, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Sun Dec 06 15:06:22 2020
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 840 cases (4 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (343/47, lift 1.4)
##  Sex = male
##  Pclass > 2
##  ->  class No  [0.861]
##
## Rule 2: (492/82, lift 1.3)
##  Sex = male
##  Adulto = Adulto
##  ->  class No  [0.832]
##
## Rule 3: (32/2, lift 2.4)
##  Pclass <= 2
##  Adulto = Menor
##  ->  class Si  [0.912]
##
## Rule 4: (291/79, lift 1.9)
##  Sex = female
##  ->  class Si  [0.727]
##
## Default class: No
##
##
## Evaluation on training data (840 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##       4  173(20.6%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     443    81    (a): class No
##      92   224    (b): class Si
##
##
##  Attribute usage:
##
##   98.33% Sex
##   62.38% Adulto
##   44.64% Pclass
##
```

```
##
## Time: 0.0 secs
```

Tras el análisis del resumen del modelo vemos que solo con el género podríamos estimar correctamente el 98% de los resultados.

6.3 Validación del modelo

En este caso no podemos validar los resultados con el conjunto de prueba ya que en teoría Kaggle ofrece estos datos como una competición. Suponiendo que tuviéramos este conjunto real de test para validar los datos, podríamos hacerlo como se muestra a continuación:

```
require(gmodels)
modelo_predicho <- predict(modelo_supervivientes, testX, type="class")
#CrossTable(testy, predicted_model,prop.chisq  = FALSE, prop.c = FALSE, prop.r =FALSE,dnn = c
('Reality', 'Prediction'))

# Extraemos nuestra predicción

df_prediccion <- cbind(df_test,modelo_predicho)
write.csv2(df_prediccion,"./resultado.csv")
```