Practica2

Jiaqian Lin

2021/01/01

Índice

---

1 Detalles de la actividad

1.1 Descripción

En esta actividad se elabora un caso práctico, consiste en el tratamiento de un conjunto de datos, orientad a aprender a identificar los datos relevante para un proyecto analítico y usar las herramientas de integración, limpieza, validación y análisis de las mismas.

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 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.

1.3 Competencias

En esta práctica se desarrollan las siguientes competencias del Master 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.

2 Resolución

2.1 Descripción

El conjunto de datos Titanic esta dividido en un subconjunto de entrenamiento y otro de test con 891 y 418 registros respectivamente. En el subconjunto de train tiene 12 variables:

- PassengerId: variable numerico de tipo int que representa el Id de cada pasajero.
- Survived: variable de salida que representa si el pasajero sobrevive o no. Es una variable categorica, donde 0 es negativo y 1 es positivo.
- Pclass: variable categorico de tipo int que representa la clase de los pasarero. 1 es primera clase, 2 es segunda y 3 es tercera.
- Name: variable de tipo string que determina el nombre de cada pasajero.
- Sex: variable categorico de 2 niveles donde representa el sexo de los pasajeros.
- Age: variable numerico de tipo int que representa la edad de cada pasajero.
- Sibsp: variable numerico de tipo int que representa numero de hermanos / conyuges embarcados.
- parch: variable numerico de tipo int que representa numero de padres / hijos embarcados.
- ticket: variable de tipo string que determina el numero de ticket de cada pasajero.
- Fare: variablen numerico de tipo double que representa el precio del ticket.
- Cabin: variable de tipo string que representa el numero de la cabina.
- Embarked: varianle categorico que representa el lugar de embarcamiento de los pasajeros. C-Cherbourg, Q-Queenstown y S-Southampton.

En el subconjunto de test tiene las mismas variables menos la variable de salida survided.

Nuestro objetivo es construir un modelo donde determinara si un pasajero sobrevive o no en el accidente de Titanic segun las variables de entrada.

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

```
# Importar dataset
train_df <- read.csv("./train.csv")
test_df <- read.csv("./test.csv")
head(train_df)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
```

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

Para realizar esta practica, eliminaremos las variables Name, Ticket y Cabin ya que no son relevantes para el analisis posterior.

```
train_df <- subset(train_df, select=-c(Ticket, Name, Cabin))
test_df <- subset(test_df, select=-c(Ticket, Name, Cabin))
```

2.3 Limpieza de los datos.

2.3.1 Valores nulos y en blanco

Encontrar valores nulos

```
colSums(is.na(train_df))
```

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

```
colSums(is.na(test_df))
```

```
## PassengerId      Pclass         Sex         Age       SibSp       Parch
##           0           0           0          86           0           0
##        Fare    Embarked
##           1           0
```

Encontramos 177 valores nulos en la variable age del subconjunto train y 86 valores nulos en el subconjunto test. Tambien encontramos un valor nulo en la variable Fare del subconjunto test. Para tratarlo, sustituiremos los valores nulos por la media de cada variable.

```
# imputación de la variable Age
train_df[is.na(train_df$Age),'Age'] <- round(mean(train_df$Age, na.rm = TRUE))
test_df[is.na(test_df$Age),'Age'] <- round(mean(test_df$Age, na.rm = TRUE))

# imputación de la variable fare
test_df[is.na(test_df$Fare),'Fare'] <- round(mean(test_df$Fare, na.rm = TRUE))
```

Una vez imputamos los valos nulos, buscaremos los valores vacios.

```
colSums(train_df == '')
```

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

```
colSums(test_df == '')
```

```
## PassengerId      Pclass         Sex         Age       SibSp       Parch
##           0           0           0           0           0           0
##        Fare    Embarked
##           0           0
```

Encontramos 2 casos de valores vacios en la variable Embarked del subconjunto train. Al ser una variable categorico, imputaremos estos valores mediante el metodo de knn.

```
# convertir los dos valores vacios en NA para poder imputar despues
train_df$Embarked[train_df$Embarked == ''] <- NA

# Imputación de valores mediante la función kNN() del paquete VIM
suppressWarnings(suppressMessages(library(VIM)))
train_df$Embarked <- kNN(train_df)$Embarked
```

2.3.2 Valores extremos

Los valores extremos o outliers son aquellos que parecen no ser congruentes sin los comparamos con el resto de los datos. No buscaremos valores extremos para la variable PassergerId ya que solo es un numero identificativo. Tampoco buscaremos valores extremos para las variables Survived, Pclass, Sex y Embarked porque son variables categoricos.

```
# usar la funcion boxplot.stats() para encontrar los valores extremos
boxplot.stats(train_df$Age)$out
```

```
##  [1]  2.00 58.00 55.00  2.00 66.00 65.00  0.83 59.00 71.00 70.50  2.00 55.50
## [13]  1.00 61.00  1.00 56.00  1.00 58.00  2.00 59.00 62.00 58.00 63.00 65.00
## [25]  2.00  0.92 61.00  2.00 60.00  1.00  1.00 64.00 65.00 56.00  0.75  2.00
## [37] 63.00 58.00 55.00 71.00  2.00 64.00 62.00 62.00 60.00 61.00 57.00 80.00
## [49]  2.00  0.75 56.00 58.00 70.00 60.00 60.00 70.00  0.67 57.00  1.00  0.42
## [61]  2.00  1.00 62.00  0.83 74.00 56.00
```

```
boxplot.stats(train_df$SibSp)$out
```

```
##  [1] 3 4 3 3 4 5 3 4 5 3 3 4 8 4 4 3 8 4 8 3 4 4 4 4 8 3 3 5 3 5 3 4 4 3 3 5 4 3
## [39] 4 8 4 3 4 8 4 8
```

```
boxplot.stats(train_df$Parch)$out
```

```
##   [1] 1 2 1 5 1 1 5 2 2 1 1 2 2 2 1 2 2 2 3 2 2 1 1 1 1 2 1 1 2 2 1 2 2 2 1 2 1
##  [38] 1 2 1 4 1 1 1 1 2 2 1 2 1 1 1 2 1 1 2 2 2 1 1 2 2 1 2 1 1 1 1 1 1 1 2 1 2
##  [75] 2 1 1 2 1 1 2 1 1 1 1 2 1 1 1 4 1 1 2 2 2 2 2 1 1 1 2 2 1 1 2 2 3 4 1 2 1
## [112] 1 2 1 2 1 2 1 1 2 2 1 1 1 1 2 2 2 2 2 2 1 1 2 1 4 1 1 2 1 2 1 1 2 5 2 1 1
## [149] 1 2 1 5 2 1 1 1 2 1 6 1 2 1 2 1 1 1 1 1 1 1 3 2 1 1 1 1 2 1 2 3 1 2 1 2 2
## [186] 1 1 2 1 2 1 2 1 1 1 2 1 1 2 1 2 1 1 1 1 3 2 1 1 1 1 5 2
```

```
boxplot.stats(train_df$Fare)$out
```

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

```
boxplot.stats(test_df$Age)$out
```

```
##  [1] 62.00 63.00 60.00 60.00 67.00  2.00 76.00 63.00  1.00 61.00 60.50 64.00
## [13] 61.00  0.33 60.00 57.00 64.00  0.92  1.00  0.75  2.00  1.00 64.00  0.83
## [25] 57.00 58.00  0.17 59.00 57.00  3.00
```

```
boxplot.stats(test_df$SibSp)$out
```

```
##  [1] 3 4 5 3 4 8 4 8 4 3 3
```

```
boxplot.stats(test_df$Parch)$out
```

```
##  [1] 1 1 1 1 3 1 2 2 1 2 1 2 1 2 4 1 1 2 1 1 1 4 6 2 3 1 1 2 2 2 1 1 2 5 2 3 2 1
## [39] 1 1 2 1 2 2 2 1 2 1 1 2 1 2 1 2 1 2 2 1 1 1 1 1 2 1 1 2 1 1 1 2 1 2 9 1 1 1
## [77] 2 2 2 1 9 1 1 2 2 1 1 2 1 1 1 1 1 1
```

```
boxplot.stats(test_df$Fare)$out
```

```
##  [1]  82.2667 262.3750  76.2917 263.0000 262.3750 262.3750 263.0000 211.5000
##  [9] 211.5000 221.7792  78.8500 221.7792  75.2417 151.5500 262.3750  83.1583
## [17] 221.7792  83.1583  83.1583 247.5208  69.5500 134.5000 227.5250  73.5000
## [25] 164.8667 211.5000  71.2833  75.2500 106.4250 134.5000 136.7792  75.2417
## [33] 136.7792  82.2667  81.8583 151.5500  93.5000 135.6333 146.5208 211.3375
## [41]  79.2000  69.5500 512.3292  73.5000  69.5500  69.5500 134.5000  81.8583
## [49] 262.3750  93.5000  79.2000 164.8667 211.5000  90.0000 108.9000
```

En cada variable se ha obtenido elevados numeros de valores extremos. Sin embargo, si revisamos los valores obtenidos, son valores que puede darse el caso aunque son valores atipicos (muy altos o muy bajos) y no son errores a la hora del registro.

2.4 Analisis de los datos

Antes de comenzar a analizar, inspeccionaremos los datos.

```
str(train_df)
```

```
## 'data.frame':    891 obs. of  9 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 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 30 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 ...
```

```
str(test_df)
```

```
## 'data.frame':    418 obs. of  8 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 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 2 2 1 2 1 2 1 2 ...
##  $ 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 ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 2 3 2 3 3 3 2 3 1 3 ...
```

Observamos que las variables Survived y Pclass son numericos del tipo int en lugar de categorico que tenemos de convertirlos en factor y en la variable Embaked existe un nivel mas que tenemos que eliminar.

```
# convertir en factor
train_df$Survived <- as.factor(train_df$Survived)
train_df$Pclass <- as.factor(train_df$Pclass)
test_df$Pclass <- as.factor(test_df$Pclass)

# eliminar el nivel "" de la variable Embarked
train_df$Embarked <- factor(train_df$Embarked)

# volver a mostrar la estructura de los dos subconjuntos
str(train_df)
```

```
## 'data.frame':    891 obs. of  9 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 ...
```

```
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 30 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/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
```

```
str(test_df)
```

```
## 'data.frame':    418 obs. of  8 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 3 3 2 3 3 3 3 2 3 3 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 2 2 1 2 1 2 1 2 ...
##  $ 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 ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 2 3 2 3 3 3 2 3 1 3 ...
```

2.4.1 Seleccion de los grupos de datos a analizar

Seleccionaremos los pasajeros por la clase, sexo y lugar de embarcamiento para analizar y comparar que efectos tiene estas variables sobre la supervivencia.

```
# agrupacion por sexo
pasajeros.mujer <- train_df[train_df$Sex == 'female',]
pasajeros.hombre <- train_df[train_df$Sex == 'male',]

# agrupacion por clase
pasajeros.1ra <- train_df[train_df$Pclass == 1,]
pasajeros.2da <- train_df[train_df$Pclass == 2,]
pasajeros.3ra <- train_df[train_df$Pclass == 3,]

# agrupacion por lugar de embarcamiento
pasajero.C <- train_df[train_df$Embarked == 'C',]
pasajero.Q <- train_df[train_df$Embarked == 'Q',]
pasajero.S <- train_df[train_df$Embarked == 'S',]
```

2.4.2 Comprobación de la normalidad y homogeneidad de la varianza

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.

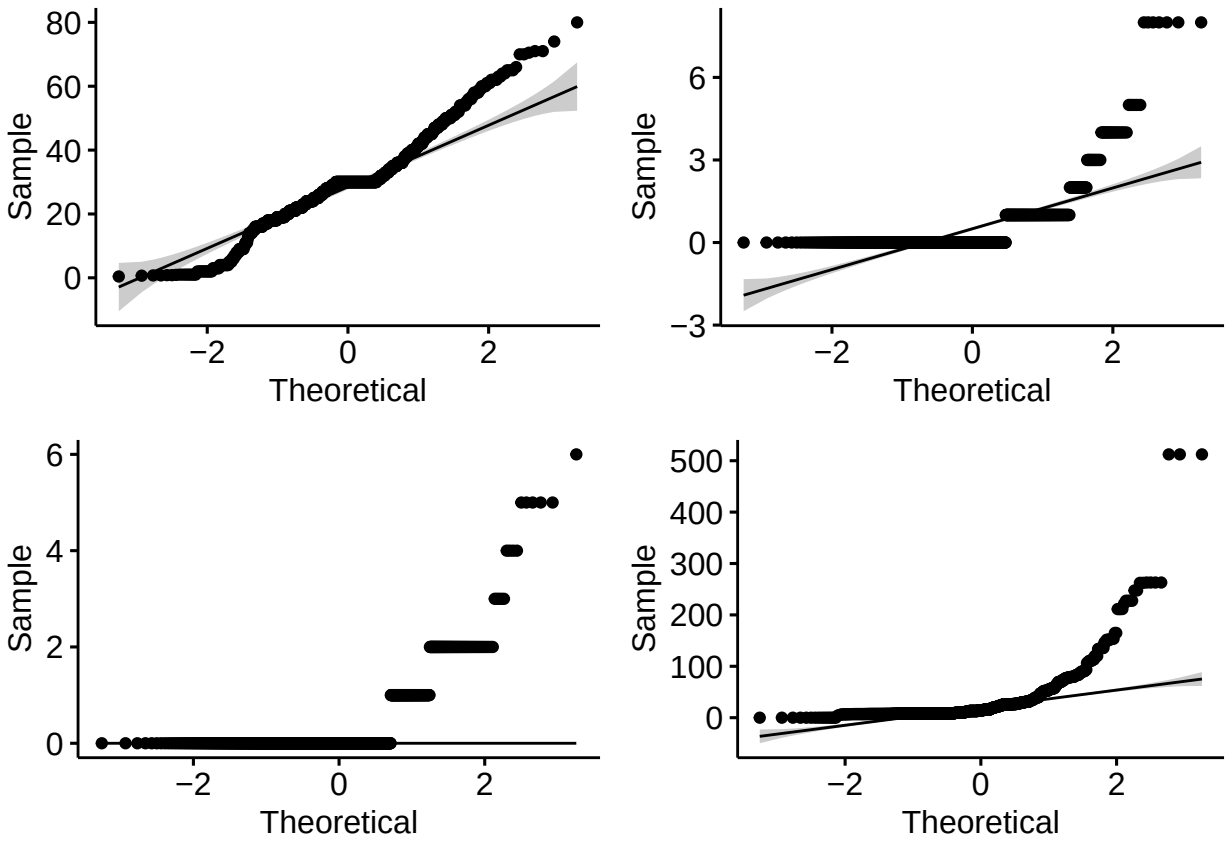
```
library(nortest)

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

```r
for (i in 1:ncol(train_df)) {
  if (i == 1) cat("Variables que no siguen una distribución normal:\n")
  if (is.integer(train_df[,i]) | is.numeric(train_df[,i])) {
    p_val = ad.test(train_df[,i])$p.value
    if (p_val < alpha) {
      cat(col.names[i])

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

```
## Variables que no siguen una distribución normal:
## PassengerId, Age, SibSp,
## Parch, Fare
```

El resultado nos dice que ninguna de las variables continuas sigue una distribución normal. En el caso de PassegerId ya se podia decir ya que es simplemente una información para identificar los pasajeros.Comprobaremos la normalidad de las otras variables mediante metodos visuales.

```r
suppressWarnings(suppressMessages(library(ggpubr)))
suppressWarnings(suppressMessages(library("gridExtra")))
g1 <- ggqqplot(train_df$Age)
g2 <- ggqqplot(train_df$SibSp)
g3 <- ggqqplot(train_df$Parch)
g4 <- ggqqplot(train_df$Fare)
grid.arrange(g1,g2,g3,g4, nrow=2)
```



2.4.3 Pruebas estadisticas

Vamos a hacer el test de chi-cuadrada entre la variable survived con otras variables para determinar cuales son mas importantes para nuestro modelo final de prediccion. Las variables que compararemos seran: Pclass, Sex y Embarked.

```
chisq.test(train_df$Survived, train_df$Pclass)
```

```
##
##  Pearson's Chi-squared test
##
## data:  train_df$Survived and train_df$Pclass
## X-squared = 102.89, df = 2, p-value < 2.2e-16
```

```
chisq.test(train_df$Survived, train_df$Sex)
```

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

```r
chisq.test(train_df$Survived, train_df$Embarked)
```

```
##
##  Pearson's Chi-squared test
##
## data:  train_df$Survived and train_df$Embarked
## X-squared = 26.977, df = 2, p-value = 1.387e-06
```

El valor p de los tres tests es inferior al 0.05, por lo cual quiere decir que estas tres variables son significantes y seran incluidos en el modelo final.

Tambien vamos a analizar las correlaciones entre las variables para saber que variables podemos dejar en la hora del modelado.

```r
library("corrplot")
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

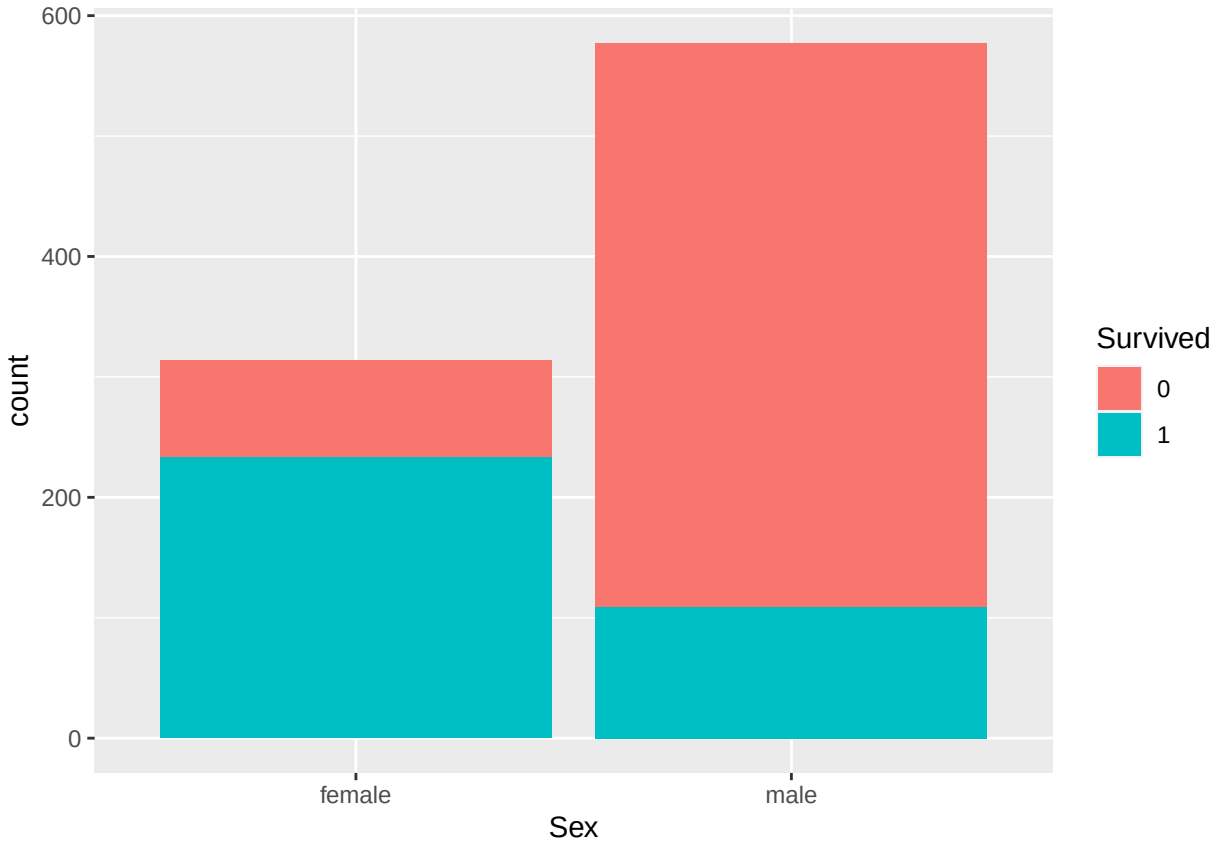
```r
res <- cor(train_df[,c(5,6,7,8)])
res
```

```
##               Age      SibSp      Parch       Fare
## Age    1.00000000 -0.2324395 -0.1803297 0.09063187
## SibSp -0.23243953  1.0000000  0.4148377 0.15965104
## Parch -0.18032972  0.4148377  1.0000000 0.21622494
## Fare   0.09063187  0.1596510  0.2162249 1.00000000
```

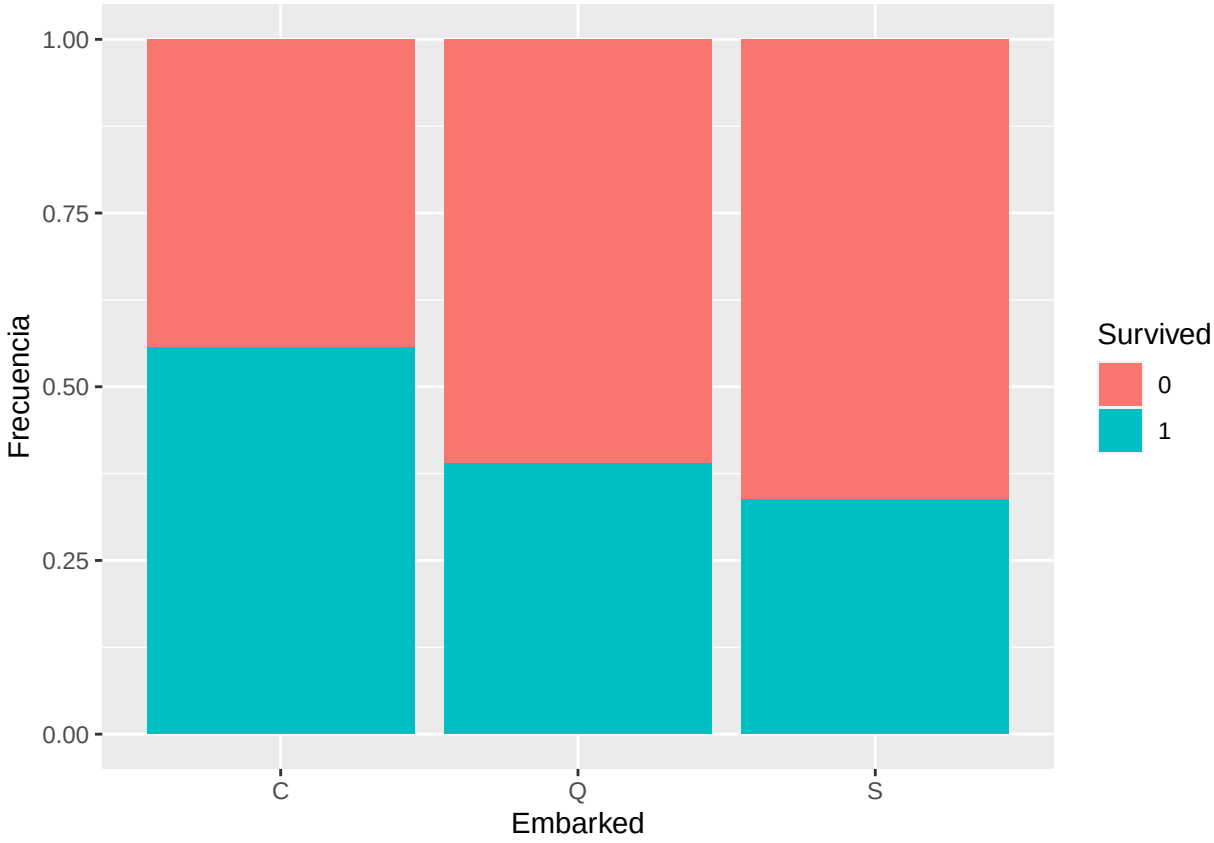
Podemos ver que no hay ninguna pareja de variable con una fuerte correlacion. Por lo tanto, incluiremos todas estas variables en el modelado final.

A continuacion miraremos como cuales son los grupos que es mas probable de sobrevivir.

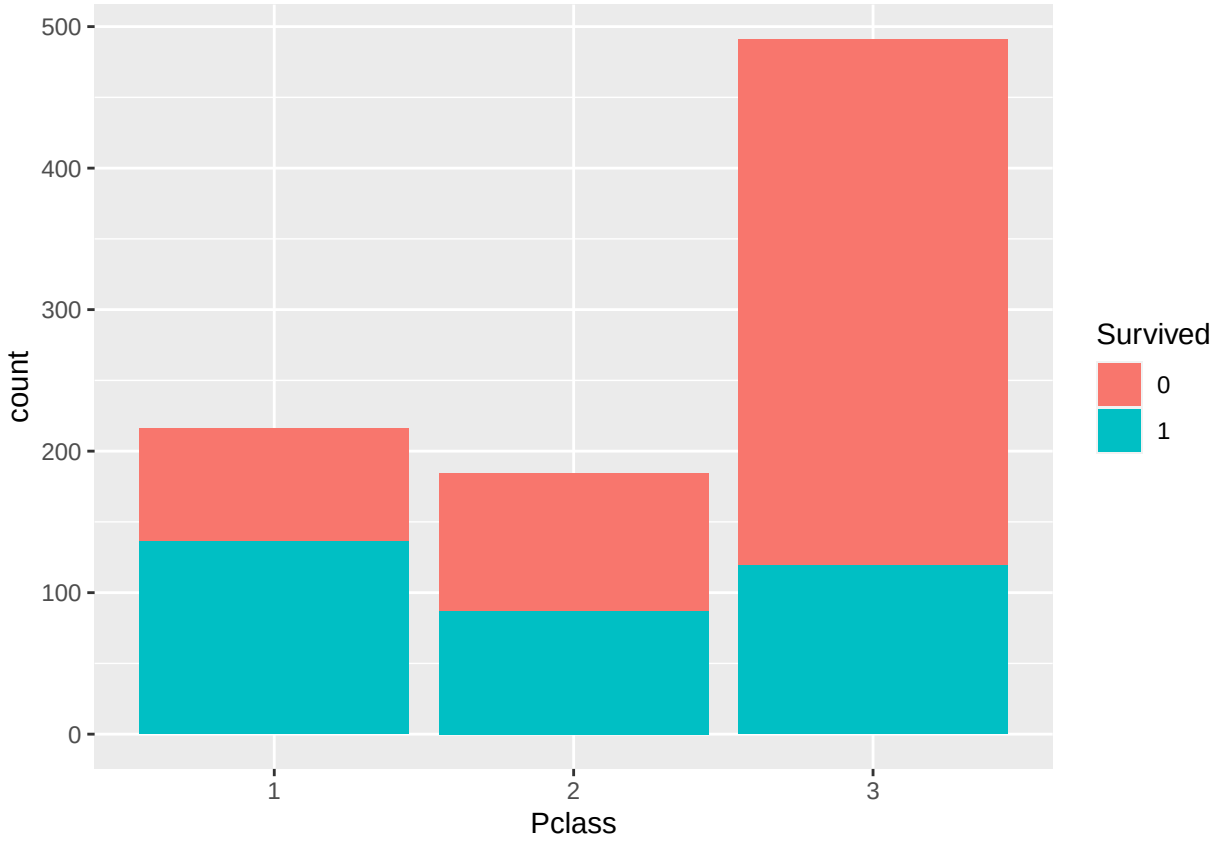
```r
library(ggplot2)
# Visualizamos la relación entre las variables "sex" y "survived":
ggplot(data=train_df,aes(x=Sex,fill=Survived))+geom_bar()
```




```
# Otro punto de vista. Survived como función de Embarked:
ggplot(data = train_df,aes(x=Embarked,fill=Survived)) +
  geom_bar(position="fill")+ylab("Frecuencia")
```




```
# Y por lo ultimo, Survived como funcion de clase
ggplot(data=train_df,aes(x=Pclass,fill=Survived))+geom_bar()
```



Podemos ver que la proporción de mujeres que sobrevive es mucho mas alto que la proporcion de hombres. En la segunda grafica, observamos que los embarcados en Cherbourg es mas probable de sobrevivir que los embargados en Queenstown o Southamptom. Y en la ultima grafica observamos que los de la tercera clase solo sobrevive una pequeña proprción. Mientras los de la primera clase es mucho mas posible de sobrevivir.

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:ggpubr':
##
##     mutate
```

```
# calcular numericamente las proporciones
sobrevive.mujer <- count(pasajeros.mujer,
                         "Survived")$freq[2]/nrow(pasajeros.mujer)*100
sobrevive.hombre <- count(pasajeros.hombre,
                         "Survived")$freq[2]/nrow(pasajeros.hombre)*100

sobrevive.C <- count(pasajero.C,
                     "Survived")$freq[2]/nrow(pasajero.C)*100
sobrevive.Q <- count(pasajero.Q,
```

```
                    "Survived")$freq[2]/nrow(pasajero.Q)*100
sobrevive.S <- count(pasajero.S,
                    "Survived")$freq[2]/nrow(pasajero.S)*100

sobrevive.1ra <- count(pasajeros.1ra,
                    "Survived")$freq[2]/nrow(pasajeros.1ra)*100
sobrevive.2da <- count(pasajeros.2da,
                    "Survived")$freq[2]/nrow(pasajeros.2da)*100
sobrevive.3ra <- count(pasajeros.3ra,
                    "Survived")$freq[2]/nrow(pasajeros.3ra)*100
```

```
# mostrar los resultados en un dataframe

grupo <- c('mujer', 'hombre',
           'Cherbourg', 'Queenstown', 'Southampton',
           '1ra', '2da', '3ra')
proporciones <- c(sobrevive.mujer, sobrevive.hombre,
                  sobrevive.C, sobrevive.Q, sobrevive.S,
                  sobrevive.1ra, sobrevive.2da, sobrevive.3ra)
prop_sv <- data.frame(grupo, proporciones)
prop_sv
```

```
##         grupo proporciones
## 1       mujer     74.20382
## 2      hombre     18.89081
## 3   Cherbourg     55.62130
## 4  Queenstown     38.96104
## 5 Southampton     33.79845
## 6         1ra     62.96296
## 7         2da     47.28261
## 8         3ra     24.23625
```

2.4.4 Modelos de prediccion

En este apartado construiremos diferentes modelos para ver la precision de cadauno de ellos. Utilizaremos el modelo KNN, regresion logistica, Support Vector Machines y CART. Para evealuar las precisiones de las diferentes algoritmos, usaremos la validacion cruzada y la funcion train del paquete caret.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
# establecer el controlador de entrenamiento trControl
# methodo cross validation de 5 fold
trControl <- trainControl(method = 'cv', number = 5)

# kNN
set.seed(111)
fit.knn <- train(Survived~., data=train_df,
```

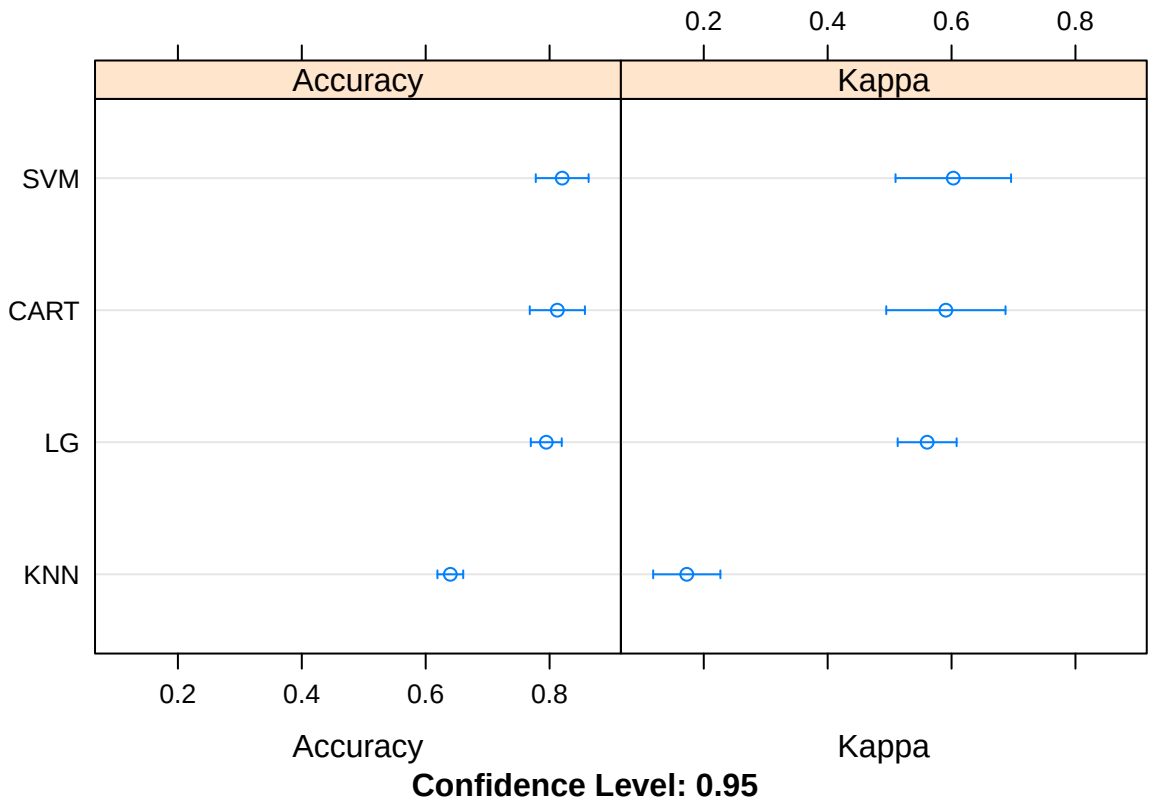
```
                method='knn', metric='Accuracy', trControl=trControl)

# regresion logistica
set.seed(222)
fit.glm <- train(Survived~., data=train_df,
                method='glm', metric='Accuracy', trControl=trControl)

# SVM
set.seed(333)
fit.svm <- train(Survived~., data=train_df, method='svmRadial',
                metric='Accuracy', trControl=trControl)

# CART
set.seed(444)
fit.cart <- train(Survived~., data=train_df,
                method='rpart', metric='Accuracy', trControl=trControl)
```

Una vez tenemos los diferentes modelos, compararemos con la funcion resamples.

```
# comparar modelos
resultado <- resamples(list(KNN=fit.knn, LG=fit.glm,
                            SVM=fit.svm, CART=fit.cart))
summary(resultado)
```

```
##
## Call:
## summary.resamples(object = resultado)
##
## Models: KNN, LG, SVM, CART
## Number of resamples: 5
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## KNN  0.6201117 0.6235955 0.6480447 0.6397546 0.6497175 0.6573034    0
## LG   0.7709497 0.7752809 0.8033708 0.7946394 0.8089888 0.8146067    0
## SVM  0.7683616 0.8146067 0.8156425 0.8203411 0.8491620 0.8539326    0
## CART 0.7528090 0.8146067 0.8212291 0.8125331 0.8248588 0.8491620    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## KNN  0.1214086 0.1523677 0.1571429 0.1725796 0.1986783 0.2333004    0
## LG   0.5152256 0.5240642 0.5756709 0.5605990 0.5908599 0.5971746    0
## SVM  0.4892674 0.5826916 0.6017627 0.6028220 0.6605324 0.6798561    0
## CART 0.4644420 0.5941618 0.5971746 0.5908267 0.6246665 0.6736885    0
```

```
dotplot(resultado)
```



Podemos ver que el algoritmo de SVM es el tiene la mejor precision, con un maximo de 85,39%. El segundo es el algoritmo CART donde solo tiene 0.48 menos, con una precision del 84.91%. Por lotanto, utilizaremos el algoritmo SVM.

Una vez tenemos decidido el algoritmo a utilizar, tunearemos para encontrar los parametros optimos. Utilizaremos la funcion tune.svm()

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```
tuned <- tune.svm(Survived~., data=train_df,
                  gamma=10^(-6:-1), cost = 10^(0:2))
summary(tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##    0.1    1
##
## - best performance: 0.1773034
```

```
##
## - Detailed performance results:
##    gamma cost     error dispersion
## 1  1e-06    1 0.3837828 0.02987088
## 2  1e-05    1 0.3837828 0.02987088
## 3  1e-04    1 0.3837828 0.02987088
## 4  1e-03    1 0.3736704 0.03027204
## 5  1e-02    1 0.2087266 0.03207204
## 6  1e-01    1 0.1773034 0.04046955
## 7  1e-06   10 0.3837828 0.02987088
## 8  1e-05   10 0.3837828 0.02987088
## 9  1e-04   10 0.3725593 0.02795401
## 10 1e-03   10 0.2132210 0.03288971
## 11 1e-02   10 0.1986267 0.03113849
## 12 1e-01   10 0.1896754 0.04317596
## 13 1e-06  100 0.3837828 0.02987088
## 14 1e-05  100 0.3725593 0.02795401
## 15 1e-04  100 0.2132085 0.03564691
## 16 1e-03  100 0.2087266 0.03250647
## 17 1e-02  100 0.1795506 0.03768393
## 18 1e-01  100 0.2121723 0.03786756
```

Vemos que se obtiene mejor resultado con gamma = 0.1 y cost=1.

2.4.5 Construccion del modelo y prediccion

```
# usaremos el algoritmo con mejor precision: SVM
set.seed(333)
modelo <- svm(Survived ~ ., data=train_df, kernel='radial',
              gamma=0.1, cost=1)
prediccion <- predict(modelo, test_df)
```

Una vez tenemos la prediccion, añadiremos al subconjunto test_df y compararemos las proporciones con las del subconjunto train_df

```
# Añadir la prediccion al subconjunto de datos test_df
test_df$SurvivedPred <- prediccion

# Agrupar los pasajeros segun el sexo, lugar de embarcamiento y clase
# agrupacion por sexo
pasajerosTs.mujer <- test_df[test_df$Sex == 'female',]
pasajerosTs.hombre <- test_df[test_df$Sex == 'male',]

# agrupacion por clase
pasajerosTs.1ra <- test_df[test_df$Pclass == 1,]
pasajerosTs.2da <- test_df[test_df$Pclass == 2,]
pasajerosTs.3ra <- test_df[test_df$Pclass == 3,]

# agrupacion por lugar de embarcamiento
pasajeroTs.C <- test_df[test_df$Embarked == 'C',]
pasajeroTs.Q <- test_df[test_df$Embarked == 'Q',]
pasajeroTs.S <- test_df[test_df$Embarked == 'S',]
```

```r
# calcular numericamente las proporciones
Psobrevive.mujer <- count(pasajerosTs.mujer,
                      "SurvivedPred")$freq[2]/nrow(pasajerosTs.mujer)*100
Psobrevive.hombre <- count(pasajerosTs.hombre,
                      "SurvivedPred")$freq[2]/nrow(pasajerosTs.hombre)*100

Psobrevive.C <- count(pasajeroTs.C,
                     "SurvivedPred")$freq[2]/nrow(pasajeroTs.C)*100
Psobrevive.Q <- count(pasajeroTs.Q,
                     "SurvivedPred")$freq[2]/nrow(pasajeroTs.Q)*100
Psobrevive.S <- count(pasajeroTs.S,
                     "SurvivedPred")$freq[2]/nrow(pasajeroTs.S)*100

Psobrevive.1ra <- count(pasajerosTs.1ra,
                     "SurvivedPred")$freq[2]/nrow(pasajerosTs.1ra)*100
Psobrevive.2da <- count(pasajerosTs.2da,
                     "SurvivedPred")$freq[2]/nrow(pasajerosTs.2da)*100
Psobrevive.3ra <- count(pasajerosTs.3ra,
                     "SurvivedPred")$freq[2]/nrow(pasajerosTs.3ra)*100

# Añadir los resultados al dataframe de proporciones para comparar
proporcionesPrd <- c(Psobrevive.mujer, Psobrevive.hombre,
                     Psobrevive.C, Psobrevive.Q, Psobrevive.S,
                     Psobrevive.1ra, Psobrevive.2da, Psobrevive.3ra)
prop_sv$prop_predic = proporcionesPrd

prop_sv
```

```
##         grupo proporciones prop_predic
## 1       mujer     74.20382   93.421053
## 2      hombre     18.89081    7.894737
## 3   Cherbourg     55.62130   50.000000
## 4  Queenstown     38.96104   52.173913
## 5 Southampton     33.79845   32.592593
## 6         1ra     62.96296   57.009346
## 7         2da     47.28261   34.408602
## 8         3ra     24.23625   32.110092
```

```r
# Distribucion de supervivencia en el subconjunto de entremaniento
table(train_df$Survived)
```

```
##
##   0   1
## 549 342
```

```r
# Distribucion de la prediccion en el subconjunto de test
table(test_df$SurvivedPred)
```

```
##
##   0   1
## 255 163
```

```r
# porporcion por sexo de pasajeros embarcado en Queenstown
# subconjunto de test
count(pasajeroTs.Q, 'Sex')
```

```
##      Sex freq
## 1 female   24
## 2   male   22
```

```r
# subconjunto de entrenamiento
count(pasajero.Q, 'Sex')
```

```
##      Sex freq
## 1 female   36
## 2   male   41
```

```r
# Salida del fichero csv
write.csv(test_df, "./output.csv")
```

2.5 Conclusión

- En el subconjunto de entremaniento ha sobrevivido 342 pasajeros de los 891, un 38.38%. Mientras en la predicción es de un 38.99%, muy similar al de los casos reales.

- En la tabla anterior de proporciones de supervivencia por diferentes grupos, vemos que en ambos casos(realidad y prediccion), sobrevive mas mujeres que hombres. Sin embargo, la porporcion de la realidad es de 74.2% y 18.9% respectativamente y la proporcion de la prediccion es de 93.4% y 7.8%. Posiblemente es debido a que nuestro modelo ha dado mas importancia la variable sexo a la hora de predecir.

- En cuanto a la proporcion por lugar de embarcamiento, el que ha sobrevivido mas es de Cherbourg (55.62%) y nuestro modelo de sobrevivencia es de 50%, un valor similar. Mientras los valores de Queenstown ha pasado de ser 38.9% al 52.2%. Podemos ver en las dos tablas anteriores sobre el sexo de los pasajeros embarcados en queenstown que en el subconjunto de test hay mas mujeres que hombres mientras en el subconjunto de entrenamiento hay mas hombres que mujeres.

- La distribución de porporciones por clase de pasajeros es similar en ambos casos. Los pasajeros de primera clase hay mas posibilidad de sobrevivir, mientras los de tercera clase solo ha sobrevivido un 24% en subconjunto de entrenamiento y 32% en prediccion.

- El subconjunto de test al no tener la variable Survived con información real, no hemos podido comprobar directamente nuestra predicción. Pero, mediante analisis de las proporciones podemos concluir que nustro modelo predice correctamente la variable de salida Survived.