Spaceship-Titanic

Juan Luis González Rodríguez & Rocío González Martínez

2022-06-04

Índice


```
# Package names
packages <- c("tidyr", "dplyr","ggplot2", "keras","reshape2","tidyverse",
              "caret","ROCR", "knitr", 'nortest')
# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
install.packages(packages[!installed_packages])
}
# Packages loading
invisible(lapply(packages, library, character.only = TRUE))
```

```
## Warning: package 'keras' was built under R version 4.1.3
```

```
## Warning: package 'ROCR' was built under R version 4.1.3
```

```
set.seed(15463)
```

1 Contexto

1.1 Descripción del Dataset

El dataset *Spaceship Titanic* [1]. Este dataset es parte de la competición homónima y tiene por objetivo crear un algoritmo para predecir qué pasajeros han desaparecido al colisionar una nave espacial denominada Titanic con una anomalía espaciotemporal. Con el conjunto de datos, se pretende predecir si el pasajero ha desaparecido o no, para enviar a un equipo a rescatarle. Para ello, se facilitan 2 ficheros (separados por entrenamiento y test), Se usará el fichero *train.csv* en uno para limpiar todos los registros y posteriormente se usará este para entrenar al modelo. Con el fichero test podremos probar el modelo (no incluye la variable objetivo).

Descripción de **Train.csv**: Conjunto de datos con información de unos 8 700 pasajeros. Este consta de los campos que se especifican más abajo.

Nombre	Tipo	Descripción
PassengerId	chr	Identificador de cada pasajero. El formato es gggg_pp (gggg hace referencia al grupo de pasajeros y pp al número dentro del grupo). Normalmente los miembros del grupo son familia.
HomePlanet	factor	Platena de origen del pasajero.
CryoSleep	logical	Indica si el pasajero está en animación suspendida durante el viaje o no.
Cabin	chr	Indican la cabina del pasajero. El formato es "plataforma/numero/lado". Lado será P o S
Destination	factor	Indica el nombre del planeta de destino del pasajero.
Age	integer	Indica la edad biológica del pasajero en años en el momento del viaje.
VIP	logical	Indica si el pasajero ha pagado por un servicio VIP o no
RoomService, FoodCouert, ShopingMall, Spa, VRDeck	numeric	Indica la cantidad de dinero que el pasajero ha gastado en cada uno de los servicios
Name	chr	Indica el nombre y apellido del pasajero
Transported	logical	Variable objetivo, indica si el pasajero ha sido transportado a otra dimensión o no (es decir si ha desaparecido).

La estructura del dataset es la siguiente:

```
df <- read.csv("~/MASTER CIENCIA DE DATOS/Tipologia y ciclo de vida de los datos/Practicas/Práctica2/Ej
               colClasses=c("HomePlanet"="factor",
                            "CryoSleep"="logical",
                            "Destination"="factor",
                            "VIP"="logical",
                            "Transported"="logical"))
df$Age <- as.integer(df$Age)

str(df)
```

```
## 'data.frame':    8693 obs. of  14 variables:
##  $ PassengerId : chr  "0001_01" "0002_01" "0003_01" "0003_02" ...
##  $ HomePlanet  : Factor w/ 4 levels "","Earth","Europa",..: 3 2 3 3 2 2 2 2 2 3 ...
##  $ CryoSleep   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Cabin       : chr  "B/0/P" "F/0/S" "A/0/S" "A/0/S" ...
##  $ Destination : Factor w/ 4 levels "","55 Cancri e",..: 4 4 4 4 4 3 4 4 4 2 ...
##  $ Age         : int  39 24 58 33 16 44 26 28 35 14 ...
##  $ VIP         : logi  FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ RoomService : num  0 109 43 0 303 0 42 0 0 0 ...
##  $ FoodCourt   : num  0 9 3576 1283 70 ...
##  $ ShoppingMall: num  0 25 0 371 151 0 3 0 17 0 ...
##  $ Spa         : num  0 549 6715 3329 565 ...
##  $ VRDeck      : num  0 44 49 193 2 0 0 NA 0 0 ...
##  $ Name        : chr  "Maham Ofracculy" "Juanna Vines" "Altark Susent" "Solam Susent" ...
##  $ Transported : logi  FALSE TRUE FALSE FALSE TRUE TRUE ...
```

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

El objetivo que se persigue con el proyecto es el de, partiendo del conjunto de datos anteriormente comentado, desarrollar un modelo supervisado que permita responder a la pregunta: **¿Ha desaparecido el pasajero que se indica?**

Con ello, la tripulación podrá dirigir los esfuerzos de una manera más eficiente y maximizar las vidas salvadas.

2 Integración y selección de los datos de interes.

Solo hay 1 fichero de origen, por lo que no hay que combinar los datos de diferentes fuentes.

Como ya se tiene a los usuarios identificados a los usuarios en base a los identificadores, no es necesario almacenar sus nombres de cara al análisis. Por otro lado, de los campos *Passenger_id* y *Cabin* se pueden extraer aún más campos como el grupo y número dentro del grupo en el primer caso y la plataforma, número de cabina y lado en el segundo.

Se elimina la variable *Name* y se crean las nuevas variables derivadas.

```
df <- select(df, -Name)
```

```
df <- df %>%
  mutate(PassengerGroup=
           as.character(sapply(strsplit(PassengerId,"_"), `[`, 1))) %>%
  mutate(PassengerNumInGroup=
           as.factor(sapply(strsplit(PassengerId,"_"), `[`, 2))) %>%
```

```
  mutate(CabinPlatform =
           as.factor(sapply(strsplit(Cabin,"/"), `[`, 1))) %>%
    mutate(CabinNumber =
           as.integer(sapply(strsplit(Cabin,"/"), `[`, 2))) %>%
    mutate(CabinSide =
           as.factor(sapply(strsplit(Cabin,"/"), `[`, 3)))

df <- select(df, -Cabin)
```

Tras crear las nuevas variables derivadas se elimina *Cabin* porque ya tenemos su información separada. PassengerId no se eliminará porque sirve para identificar los registros. Se muestra un resumen de los campos con la función *summary*.

```
summary(df)
```

```
##  PassengerId         HomePlanet   CryoSleep              Destination
##  Length:8693              : 201   Mode :logical                : 182
##  Class :character   Earth :4602   FALSE:5439      55 Cancri e  :1800
##  Mode  :character   Europa:2131   TRUE :3037      PSO J318.5-22: 796
##                     Mars  :1759   NA's :217       TRAPPIST-1e  :5915
##
##
##
##       Age            VIP           RoomService        FoodCourt
##  Min.   : 0.00   Mode :logical   Min.   :    0.0   Min.   :    0.0
##  1st Qu.:19.00   FALSE:8291      1st Qu.:    0.0   1st Qu.:    0.0
##  Median :27.00   TRUE :199       Median :    0.0   Median :    0.0
##  Mean   :28.83   NA's :203       Mean   :  224.7   Mean   :  458.1
##  3rd Qu.:38.00                   3rd Qu.:   47.0   3rd Qu.:   76.0
##  Max.   :79.00                   Max.   :14327.0   Max.   :29813.0
##  NA's   :179                     NA's   :181       NA's   :183
##   ShoppingMall          Spa              VRDeck        Transported
##  Min.   :    0.0   Min.   :    0.0   Min.   :    0.0   Mode :logical
##  1st Qu.:    0.0   1st Qu.:    0.0   1st Qu.:    0.0   FALSE:4315
##  Median :    0.0   Median :    0.0   Median :    0.0   TRUE :4378
##  Mean   :  173.7   Mean   :  311.1   Mean   :  304.9
##  3rd Qu.:   27.0   3rd Qu.:   59.0   3rd Qu.:   46.0
##  Max.   :23492.0   Max.   :22408.0   Max.   :24133.0
##  NA's   :208       NA's   :183       NA's   :188
##  PassengerGroup     PassengerNumInGroup CabinPlatform   CabinNumber
##  Length:8693        01     :6217        F      :2794   Min.   :   0.0
##  Class :character   02     :1412        G      :2559   1st Qu.: 167.2
##  Mode  :character   03     : 571        E      : 876   Median : 427.0
##                     04     : 231        B      : 779   Mean   : 600.4
##                     05     : 128        C      : 747   3rd Qu.: 999.0
##                     06     :  75        (Other): 739   Max.   :1894.0
##                     (Other):  59        NA's   : 199   NA's   :199
##  CabinSide
##  P   :4206
##  S   :4288
##  NA's: 199
##
##
```

```
##
##
```

Cabe destacar que en *HomePlanet* y en *Destination* hay campos con valores vacíos que no se han considerado como NA's. Por otro lado, Hay algunos campos que presenta NA's que podrán tratarse o desestimarse. También se observan valores extremos en algunos campos.

3 Limpieza de los Datos.

En este apartado se tratará de mejorar la calidad de los datos presentes en base a la falta de calidad. Por límite de extensión del proyecto, nos centraremos en el tratamiento de outliers y de valores nulos.

3.1 Tratamiento valores nulos.

Se remapean los campos en blanco de los campos *HomePlanet* y *Destination* por el valor *Unknown*. Con esto, no perdemos información y evitamos confundir a las personas que interpreten los resultados.

```
levels(df$HomePlanet) <- c("Unknown", "Earth", "Europa", "Mars")
levels(df$Destination) <- c("Unknown", "55 Cancri e", "PSO J318.5-22",
                            "TRAPPIST-1e")
```

Se muestran la cantidad de valores nulos que tiene cada campo.

```
sapply(df, function(x) sum(length(which(is.na(x)))))
```

```
##         PassengerId          HomePlanet           CryoSleep         Destination
##                   0                   0                 217                   0
##                 Age                 VIP         RoomService           FoodCourt
##                 179                 203                 181                 183
##        ShoppingMall                 Spa              VRDeck         Transported
##                 208                 183                 188                   0
##      PassengerGroup PassengerNumInGroup       CabinPlatform         CabinNumber
##                   0                   0                 199                 199
##           CabinSide
##                 199
```

Son relativamente pocos registros en comparación con el total que constan en el dataset. Por lo que se decide con contar con estos registros para entrenar al modelo predictivo.

```
nrow(df)
```

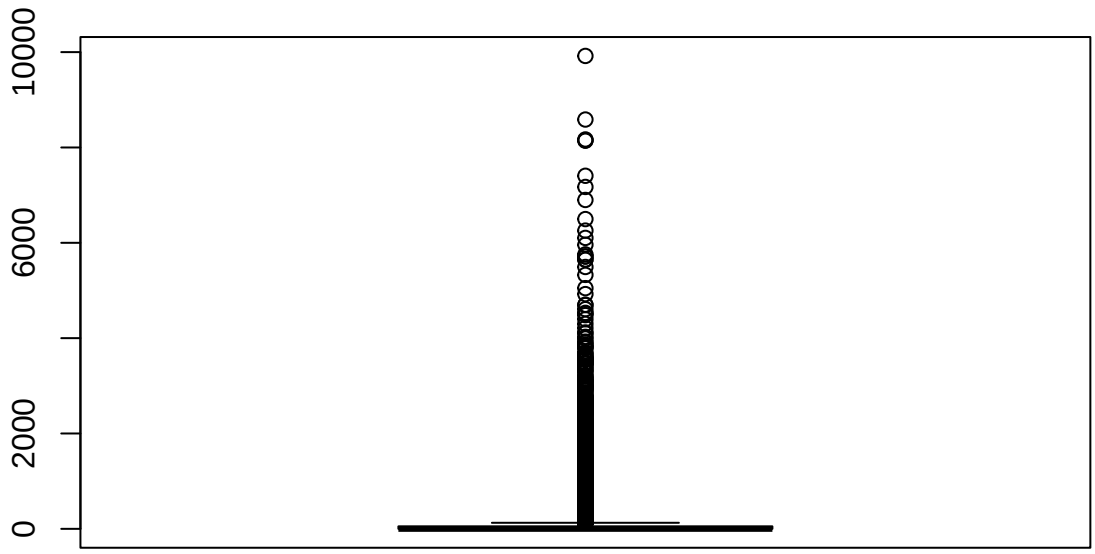
```
## [1] 8693
```

```
df <- na.omit(df)
nrow(df)
```

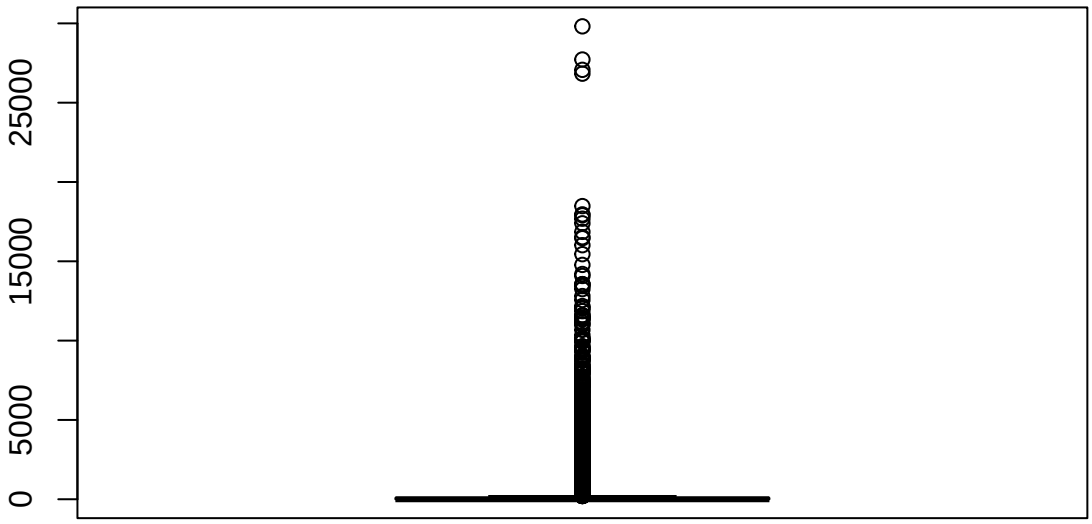
```
## [1] 7084
```

3.2 Tratamiento valores extremos.

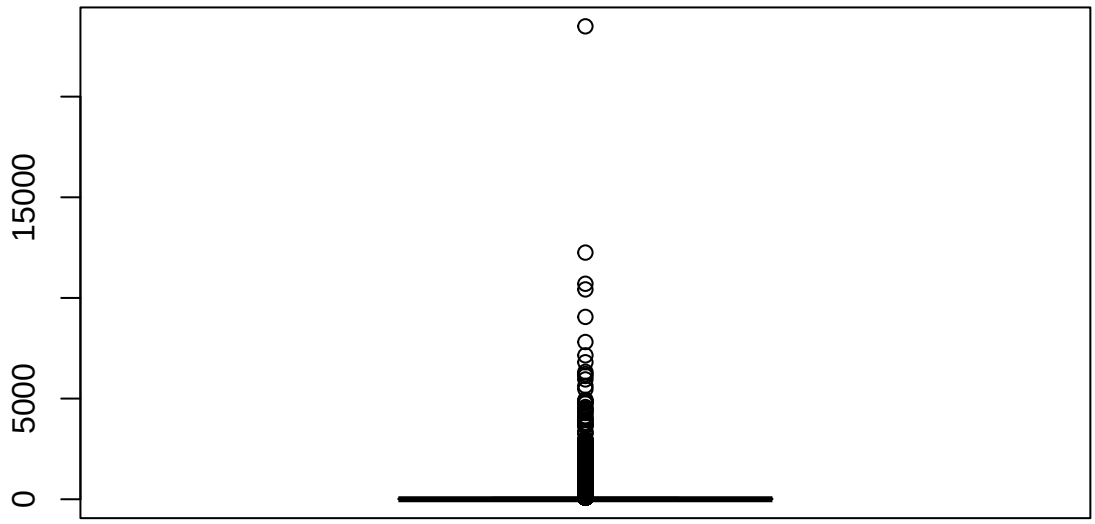
```
attach(df)
boxplot(RoomService)
```



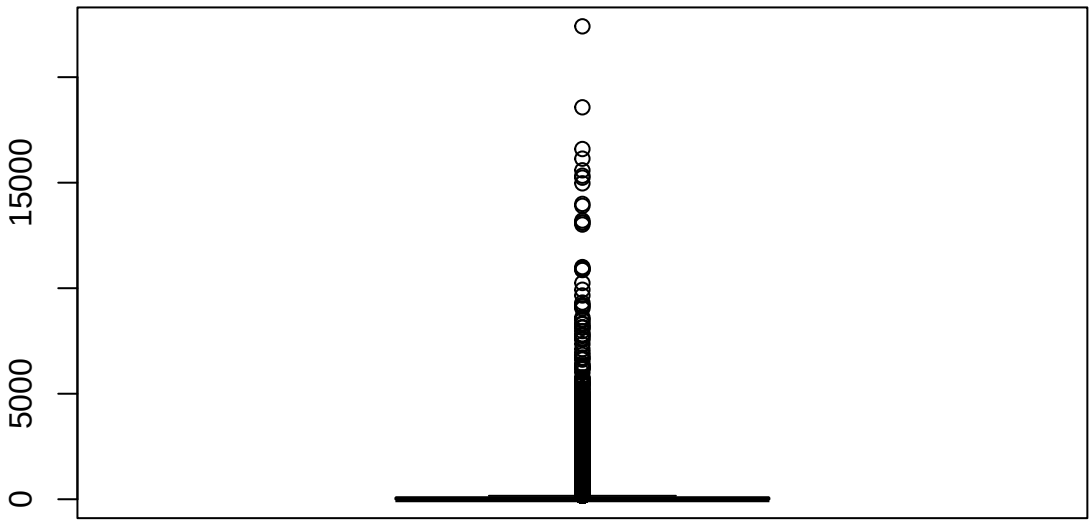

```
boxplot(FoodCourt)
```



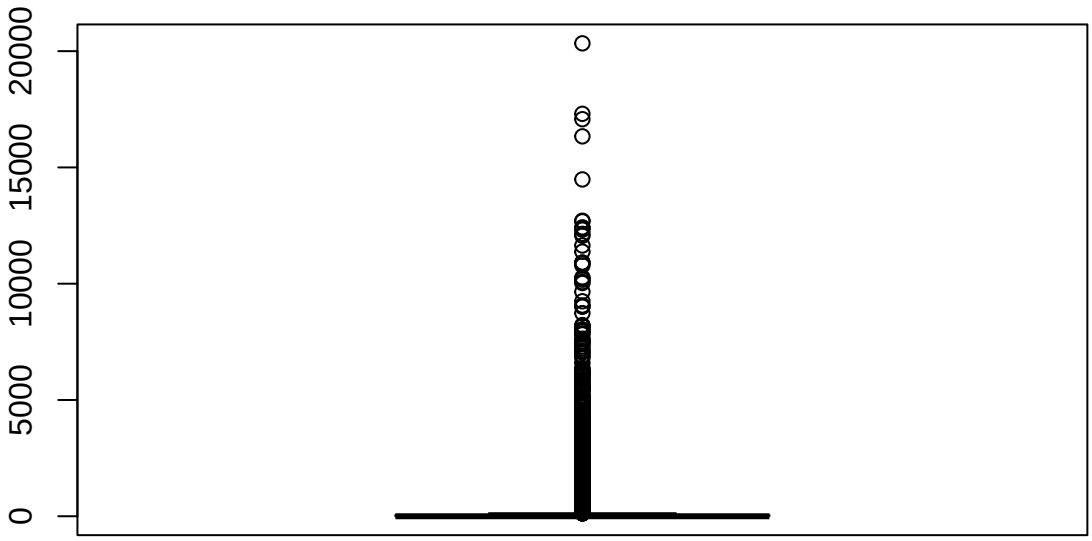

```
boxplot(ShoppingMall)
```




```
boxplot(Spa)
```




```
boxplot(VRDeck)
```




```
detach(df)
```

Aunque encontramos valores muy alejados de los valores centrales. No se consideraran como valores extremos. Se considerarán valores atípicos pero que son representativos de la variedad de nuestra muestra y por tanto formarán parte de los datos para entrenar al modelo. No se eliminará ningún valor extremo.

4 Análisis de los datos

4.1

4.2 Comprobación de la normalidad y homogeneidad de la varianza

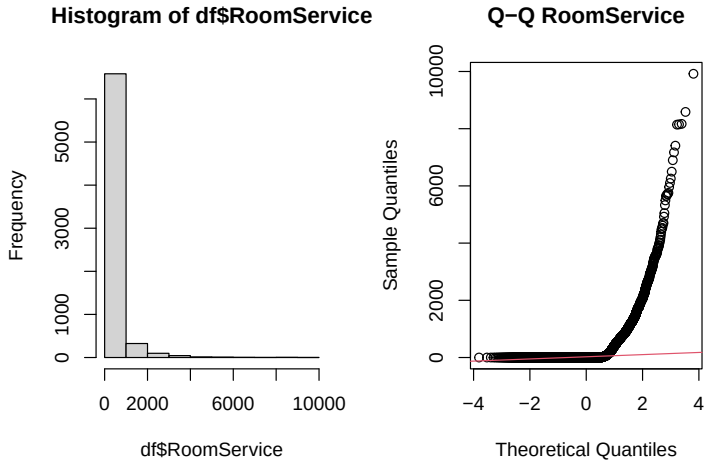
Debido a la naturaleza de las variables, se deberá estudiar la normalidad y varianza para las variables numéricas. No tiene sentido estudiar si se distribuye normalmente o con que varianza lo hace una variable categórica. Es decir, se estudiará sobre las variables *Age*,*RoomService*, *FoodCourt*, *ShoppingMall*, *Spa* y *VRDeck*.

A continuación se detalla el **estudio de la normalidad**. Debido a que, la cantidad de registros es superior a 5 000, se usará el test de normalidad de *Anderson-Darling*, ya que el test de *Shapiro-Wilk* tiene como limitación un valor máx. de 5 000 registros.

```
par(mfrow=c(1,2))

hist(df$RoomService)
```

```
qqnorm(df$RoomService, main="Q-Q RoomService")
qqline(df$RoomService,col=2)
```



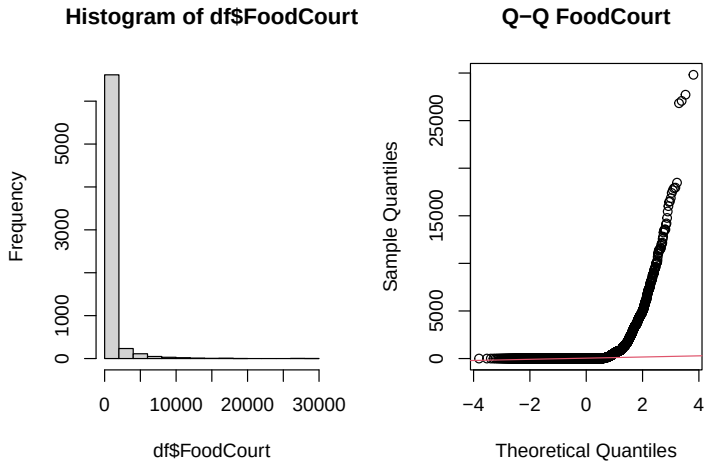

```
ad.test(df$RoomService)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$RoomService
## A = 1546.6, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$FoodCourt)
qqnorm(df$FoodCourt, main="Q-Q FoodCourt")
qqline(df$FoodCourt,col=2)
```



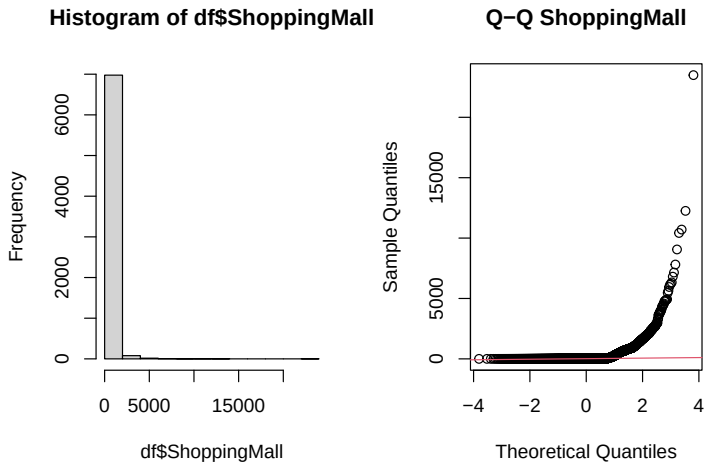

```
ad.test(df$FoodCourt)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$FoodCourt
## A = 1761, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$ShoppingMall)
qqnorm(df$ShoppingMall, main="Q-Q ShoppingMall")
qqline(df$ShoppingMall,col=2)
```



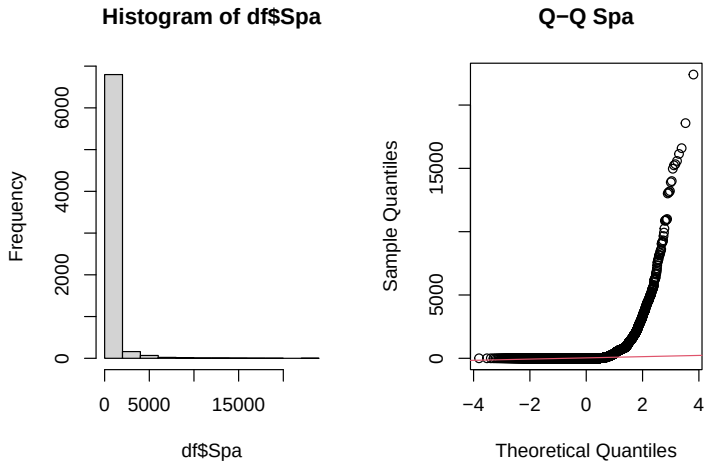
```
ad.test(df$ShoppingMall)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$ShoppingMall
## A = 1632.3, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$Spa)
qqnorm(df$Spa, main="Q-Q Spa")
qqline(df$Spa,col=2)
```



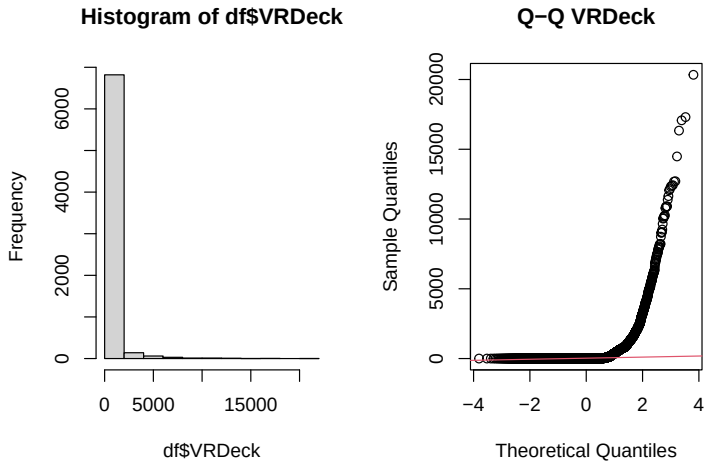

```
ad.test(df$Spa)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$Spa
## A = 1751.3, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$VRDeck)
qqnorm(df$VRDeck, main="Q-Q VRDeck")
qqline(df$VRDeck,col=2)
```



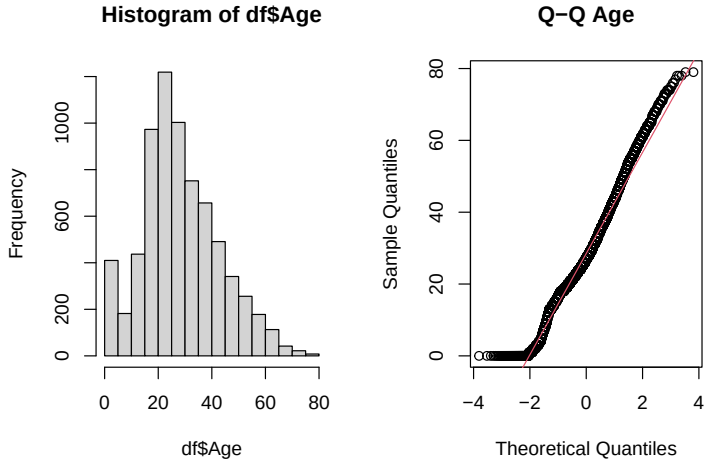

```
ad.test(df$VRDeck)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$VRDeck
## A = 1772.1, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$Age)
qqnorm(df$Age, main="Q-Q Age")
qqline(df$Age,col=2)
```



```
ad.test(df$Age)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$Age
## A = 40.901, p-value < 2.2e-16
```

Gráficamente se observa que no sigue del todo una distribución normal, hay más datos de los esperados en la parte izquierda de la distribución, por lo que aunque se aproxima no aporta infomación segura. Si realizamos un test formal de normalidad, como el de *Anderson-Daling*, indica un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

4.3

MODELO DE PRUEBA

Se usará un modelo de regresivo logística que permitirá identificar a los pasajeros entre transportados o no transportados. Para ello hará uso de las variables explicativas que determinarán en la medida de lo posible si estos pasajeros toman un valor u otro.

Se parten los datos en *df* para entrenar al modelo con un 70% de los casos y en *df_validation* para medir la bondad del modelo con el 30% restante.

```
df_raw <- df

Index <- createDataPartition(df_raw$Transported, p=0.7, list=FALSE,times=1)

df <- df_raw[Index,]
df_validation <- df_raw[-Index,]
```

Aunque no se incluye en la práctica por motivos de extensión. Se han realizado análisis previos y se ha determinado que hay variables que empeoran el modelo, por lo que estas se eliminan en el entrenamiento y en la validación.

```
df1 <- select(df, -c(PassengerId, PassengerGroup, HomePlanet, Destination,VIP,
                     PassengerNumInGroup, CabinPlatform, Age))
```

Se entrena el modelo con el dataset previamente creado. Se presupone que lasvariables no son bariables de confusión y no presentan interacción entre ellas.

```
model_log <- glm(Transported ~ ., data = df1, family = binomial(link='logit'))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_log)
```

```
##
## Call:
```

```
## glm(formula = Transported ~ ., family = binomial(link = "logit"),
##     data = df1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4313  -0.7377   0.0190   0.6776   3.3592
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.865e-01  9.125e-02  -4.236 2.28e-05 ***
## CryoSleepTRUE  1.800e+00  9.122e-02  19.734  < 2e-16 ***
## RoomService   -1.294e-03  1.119e-04 -11.555  < 2e-16 ***
## FoodCourt      7.013e-04  4.862e-05  14.423  < 2e-16 ***
## ShoppingMall   6.226e-04  7.628e-05   8.163 3.28e-16 ***
## Spa           -1.644e-03  1.348e-04 -12.194  < 2e-16 ***
## VRDeck        -1.322e-03  1.135e-04 -11.649  < 2e-16 ***
## CabinNumber   -1.810e-04  7.156e-05  -2.529   0.0114 *
## CabinSideS     5.469e-01  7.286e-02   7.505 6.13e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6874.5  on 4958  degrees of freedom
## Residual deviance: 4665.5  on 4950  degrees of freedom
## AIC: 4683.5
##
## Number of Fisher Scoring iterations: 7
```

A continuación se evalúa el mismo. Para ello se predicen los valores del conjunto *df_validation* y se comparan con los valres reales del mismo.

```
fitted.results <- predict(model_log,
                          newdata = select(df_validation, c(CryoSleep,
                                                            RoomService,
                                                            FoodCourt,
                                                            ShoppingMall,
                                                            Spa,VRDeck,
                                                            CabinNumber,
                                                            CabinSide))
                          ,type = "response")
```

```
fitted.results_FACTOR <- ifelse(fitted.results > 0.5,TRUE,FALSE)

resultado_comprobacion <- cbind("ID"=df_validation$PassengerId,
                                "Transported"=df_validation$Transported,
                                "Prediction"=fitted.results_FACTOR)

df_validation_info <- as.data.frame(resultado_comprobacion)
kable(head(df_validation_info),align='c', row.names=FALSE)
```

ID	Transported	Prediction
0004_01	TRUE	FALSE
0009_01	TRUE	TRUE
0016_01	TRUE	TRUE
0020_02	FALSE	TRUE
0020_04	TRUE	TRUE
0044_02	TRUE	TRUE

Se convierten las variables *Transported* (valor real) y *Prediction* (valor predicho) y se mide la bondad del modelo con la matriz de confusión mediante la función *confusionMatrix).

```
df_validation_info$Transported <- as.factor(df_validation_info$Transported)
df_validation_info$Prediction <- as.factor(df_validation_info$Prediction)

confusionMatrix(df_validation_info$Transported, df_validation_info$Prediction)
```

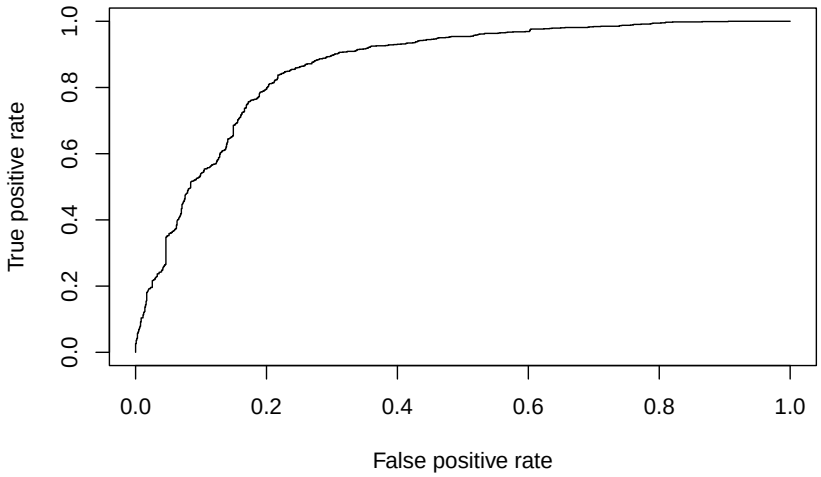
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   833  225
##      TRUE    192  875
##
##                Accuracy : 0.8038
##                  95% CI : (0.7862, 0.8205)
##     No Information Rate : 0.5176
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6075
##
##  Mcnemar's Test P-Value : 0.1171
##
##             Sensitivity : 0.8127
##             Specificity : 0.7955
##          Pos Pred Value : 0.7873
##          Neg Pred Value : 0.8201
##              Prevalence : 0.4824
##          Detection Rate : 0.3920
##    Detection Prevalence : 0.4979
##       Balanced Accuracy : 0.8041
##
##        'Positive' Class : FALSE
##
```

La clase positiva se categoriza como FALSE (no desaparecido).

Se observa una buena predicción de las clases. Con una exactitud del 80,38%. La tasa de positivos que se han asignado como positivos es del 81,27%. mientras que los negativos identificado como auténticos negativos es del 79,55%. De los verdaderos.

La precisión, es decir, los datos clasificados como positivos y que realmente lo son es del 78,73%.

```
pr <- prediction(fitted.results, df_validation$Transported)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8637542
```

Vemos como la curva ROC nos da un valor próxima a la esquina superior derecha y un área del 0.85, muy próximo a 1.

Se observa que el modelo es bueno prediciendo si los pasajeros han desaparecido o no.

[1] Kaggle, "Spaceship titanic." kaggle.com; Kaggle, 2022.Available: https://www.kaggle.com/competitions/spaceship-titanic/overview