# Práctica 2: Limpieza y validación de los datos

Ismael Rosende Rey

26/12/2020

## Índice

## 1. Descripción del dataset

Se utiliza el dataset sugerido en el enunciado, que se puede encontrar en la web de Kaggle en la siguiente dirección:

https://github.com/isrosrey/PRAC2/tree/main/data

Se trata de una competición para principiantes en Kaggle, que trata sobre el hundimiento del titanic. Los datos se dividen en dos csv, uno con 891 filas que contienen la información completa de los pasajeros, indicando si han sobrevivido o no, y otro con 418 filas que contiene la misma información execpto el dato de supervivencia. También hay otro fichero adicional pero es a modo de ejemplo para ver el formato con el que habría que subir los resultados obtenidos.

Esta competición consiste en preparar un modelo que predice si un pasajero ha sobrevivido o ha muerto en base a la información disponible. El fichero "train.csv" contiene 12 columnas con los siguientes campos acerca de los pasajeros, mientras que el fichero "test.csv" contiene 11 columnas al no disponer de la columna de supervivencia:

- **PassengerId**: Identificador único asociado a cada pasajero.
- **Survival (0=No, 1=Yes)**: Indica si ha sobrevivido.
- **Pclass (1=1st, 2=2nd, 3=3rd)**: Clase en la que viajaba.
- **Name**: Nombre completo del pasajero.
- **Sex**: Si se trataba de un hombre o mujer.
- **Age**: Edad en años.
- **Sibsp**: Número de hermanos / esposas abordo del barco.
- **Parch**: Número de padres / niños abordo del barco.
- **Ticket**: Número del ticket.
- **Fare**: Tarifa que ha pagado por el ticket.
- **Cabin**: Número de cabina.
- **Embarked (C=Cherbourg, Q=Queenstown, S=Southampton)**: Puerto de embarcación.

Este dataset está creado con fines educativos en la plataforma Kaggle, y mediante su análisis se pretende encontrar qué factores fueron determinantes en el momento del accidente para que ciertas personas sobreviviesen o muriesen. De esta forma, se podrían detectar posibles mejoras en el diseño del barco que aumenten las posibilidades de supervivencia de los pasajeros en caso de que la tragedia se repita. También podría ser útil para recomendar a los pasajeros que tipo de ticket (según clase o zona del barco) sería el más recomendable para ellos.

## 2. Integración y selección de datos

Comenzamos leyendo los ficheros de datos:

```
train_data <- read.csv("../data/train.csv")
test_data <- read.csv("../data/test.csv", stringsAsFactors = TRUE)
```

Integramos los datos en un único dataset para tener en consideración toda la información durante la limpieza de los datos:

```
# Unimos los datos de entrenamiento y prueba
merged_data  <- bind_rows(train_data, test_data)
```

Mostramos los datos de diferentes formas para ver información de interés que nos ayudará a determinar qué campos necesitan ser tratados, por ejemplo, porque tienen muchos NAs, o elementos vacíos, y también nos ayudará a decidir que datos utilizar.

```
str(merged_data)
```

```
## 'data.frame':    1309 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
```

```
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

```
summary(merged_data)
```

```
##   PassengerId      Survived          Pclass          Name
##  Min.   :   1   Min.   :0.0000   Min.   :1.000   Length:1309
##  1st Qu.: 328   1st Qu.:0.0000   1st Qu.:2.000   Class :character
##  Median : 655   Median :0.0000   Median :3.000   Mode  :character
##  Mean   : 655   Mean   :0.3838   Mean   :2.295
##  3rd Qu.: 982   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :1309   Max.   :1.0000   Max.   :3.000
##                 NA's   :418
##      Sex                 Age            SibSp            Parch
##  Length:1309        Min.   : 0.17   Min.   :0.0000   Min.   :0.000
##  Class :character   1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000
##  Mode  :character   Median :28.00   Median :0.0000   Median :0.000
##                     Mean   :29.88   Mean   :0.4989   Mean   :0.385
##                     3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:0.000
##                     Max.   :80.00   Max.   :8.0000   Max.   :9.000
##                     NA's   :263
##     Ticket               Fare            Cabin             Embarked
##  Length:1309        Min.   :  0.000   Length:1309        Length:1309
##  Class :character   1st Qu.:  7.896   Class :character   Class :character
##  Mode  :character   Median : 14.454   Mode  :character   Mode  :character
##                     Mean   : 33.295
##                     3rd Qu.: 31.275
##                     Max.   :512.329
##                     NA's   :1
```

```
colSums(merged_data == "" | merged_data == " ")
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0          NA           0           0           0          NA
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0          NA        1014           2
```

A primera vista se considera que todas las columnas pueden aportar información de interés. Podríamos eliminar el campo "PassengerId" porque no será útil para determinar si un pasajero vive o muere, pero se usará para recomponer los bloques de entrenamiento y prueba originales al finalizar la limpieza y análisis de los datos.

Podría ser útil crear algún campo adicional, por ejemplo, una columna que agrupe los miembros de una misma familia, si un pasajero es adulto o no, etc, pero se opta por comenzar sin generar campos adicionales.

## 3. Limpieza de datos

Comprobamos qué campos podrían convertirse a factor. Para ello contamos las diferencias en los valores que contienen sus filas:

```
sapply(merged_data, function(x) length(unique(x)))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##        1309           3           3        1307           2          99
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           7           8         929         282         187           4
```

Vemos que los campos "Survived", "Pclass", "Sex", "Embarked" son susceptibles de conversión al tener pocos valores asociados (menos de 4 en estos casos). Por tanto los convertimos todos a tipo factor, excepto "Survived", que lo usaremos en formato numérico para analizar la correlación con otras variables:

```
merged_data$Pclass <- as.factor(merged_data$Pclass)
merged_data$Sex <- as.factor(merged_data$Sex)
merged_data$Embarked <- as.factor(merged_data$Embarked)
```

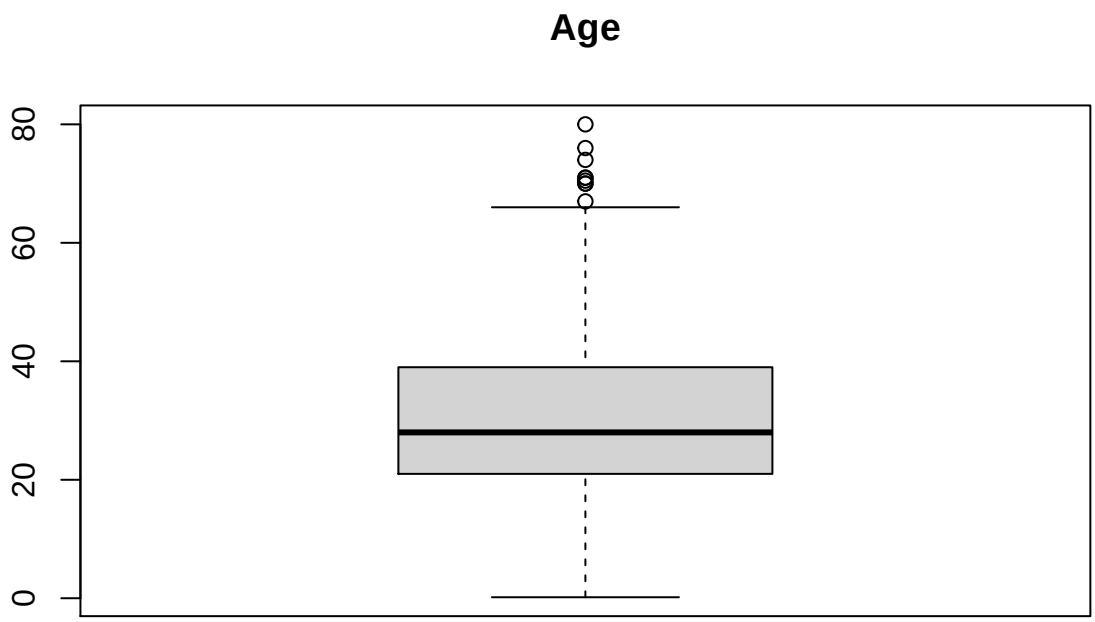
Una vez tenemos los campos con el tipo deseado, analizamos qué hacer con los elementos perdidos. Como hemos visto en el resumen de los datos del apartado anterior, existen NAs en las variables "Age" y "Fare" (además de la variable "Survived", pero ya sabemos que los datos de test omitían este dato), y tenemos elementos vacíos en las variables "Cabin" y "Embarked".

Observando los datos ordenados por el Cabin, haciendo uso del visor de RStudio, nos damos cuenta de que los pasajeros con Cabin B19 al B26 han embarcado por la puerta de Southampton, por tanto, suponemos que a los 2 valores vacíos que tenían la cabina B28 asiganda, les correspondería entrar por la puerta S también.

```
merged_data$Embarked[merged_data$Embarked == ""] <- "S"
```

En la variable "Age" habíamos detectado muchos elementos vacíos, por tanto se considera necesario imputar los valores. Haciendo uso de un gráfico boxplot, se comprueba que no hay valores extremos, por ejemplo, edades de más de 100 años.

```
# Comprobamos que no hay valores extremos sin sentido (como gente de 150 años
# por ejemplo)
boxplot(merged_data$Age, names="Edad", main="Age")
```




Vemos que sí existen valores extremos en esta variable, pero son valores que podrían darse realmente, se opta por no tratarlos y rellenar los elementos vacíos asignanado la media de edad, considerando una media diferente para las mujeres y otra para los hombres:

```
# Hallamos la media de la edad de las mujeres y de los hombres
femaleMeanAge <- mean(merged_data$Age[merged_data$Sex == "female"], na.rm=T)
maleMeanAge <- mean(merged_data$Age[merged_data$Sex == "male"], na.rm=T)
# Se asigna la media correspondiente a cada grupo
merged_data$Age[is.na(merged_data$Age) & merged_data$Sex == "female"] <- femaleMeanAge
merged_data$Age[is.na(merged_data$Age) & merged_data$Sex == "male"] <- maleMeanAge
# Se comprueba que no quedan valores sin asignar
sum(is.na(merged_data$Age))
```

```
## [1] 0
```

En el caso de la variable "Fare", también tendremos en cuenta aquellos registros con valor 0, ya que se supone que todos los pasajeros tendrían que haber pagado un determinado precio superior a 0 para embarcar. Una vez localizados esos registros, se les imputa valor NA para que no afecten al hallar la media y luego se les imputa la misma. Para ser más rigurosos se podría tener en cuenta la clase en la que viajaban e imputar la media de cada tipo de clase, ya que se considera que viajar en primera clase es más caro que segunda clase, y que segunda es más cara que tercera.

```
# Asignamos NA a los valores 0 de Fare y les imputamos la media igual que al
# registro vacío.
merged_data$Fare[merged_data$Fare == 0] <- NA
merged_data$Fare[is.na(merged_data$Fare)] <- mean(merged_data$Fare, na.rm=T)
# Se comprueba que no quedan valores nulos
sum(is.na(merged_data$Fare))
```

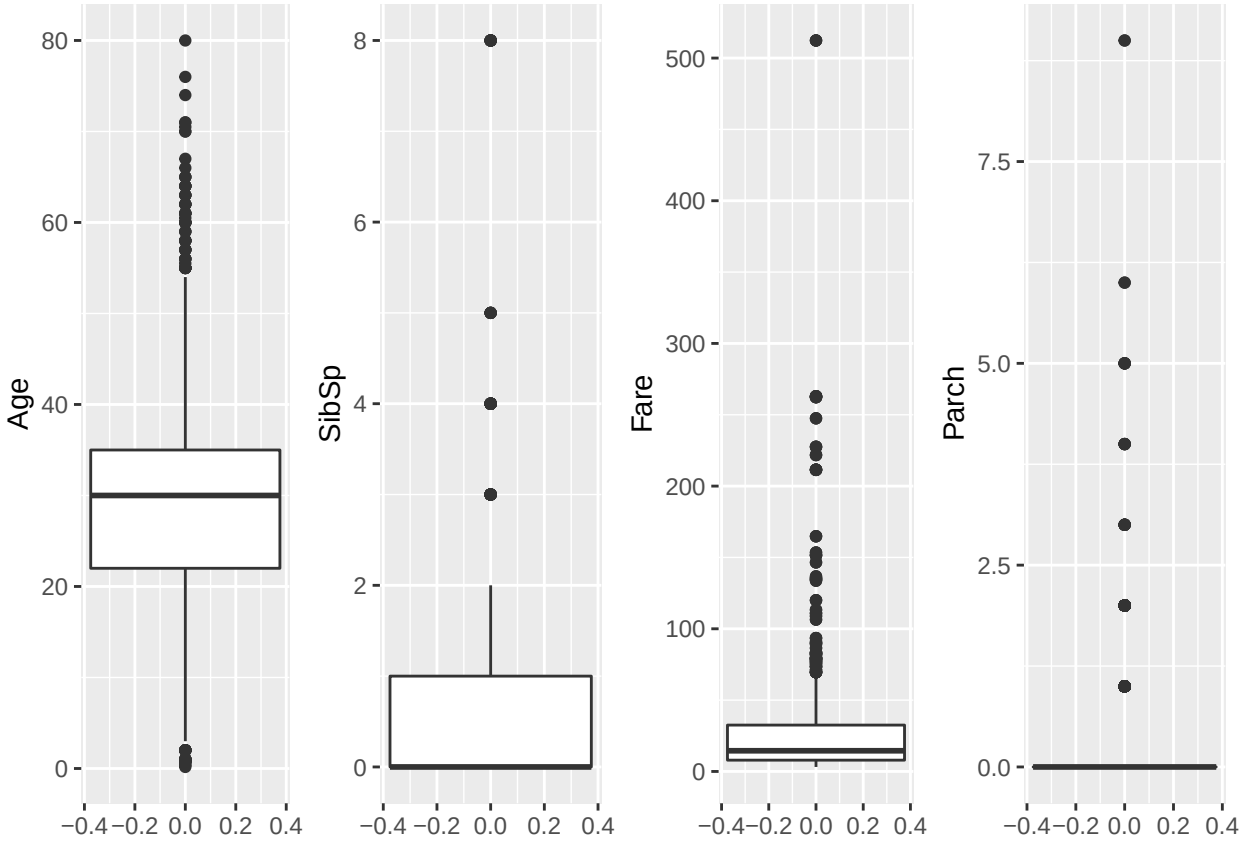
```
## [1] 0
```

En el caso de "Cabin", al tratarse de un valor categórico, no se puede asignar de la misma forma que las variables anteriores. Habría que analizar si existe una relación fuerte con alguna de las demás variables que nos permita "adivinar" cuál sería su valor. Como en este caso tenemos un 77.46 % de valores perdidos, optamos por eliminar esta columna una vez la hemos aprovechado en lo que nos ha parecido útil (asignar los 2 valores de Embarked perdidos en este caso). También se podría asignar un valor común, por ejemplo "Desconocido", pero se cree que no aportará muchos beneficios y se descarta esta opción.

```
# Eliminamos la columna "Cabin" del dataset
merged_data <- subset(merged_data, select=-Cabin)
```

Se utilizan gráficos boxplot para detectar posibles valores extremos en las variables cuantitativas:

```
g1 <- ggplot(merged_data, aes(y=Age)) + geom_boxplot()
g2 <- ggplot(merged_data, aes(y=SibSp)) + geom_boxplot()
g3 <- ggplot(merged_data, aes(y=Fare)) + geom_boxplot()
g4 <- ggplot(merged_data, aes(y=Parch)) + geom_boxplot()
grid.arrange(g1,g2, g3, g4, nrow=1)
```




Observamos que se detectan outliers en todos los casos, pero en ninguno de ellos son outliers realmente, sino valores que ocurren con poca frecuencia y por eso se muestran como outliers.
En el caso de Age, la mayor parte de pasajeros tienen alrededo de 30 años, pero hay algunos casos de personas con hasta 80 años y algunos recién nacidos.
En el caso de la variable SibSp vemos que la mayor parte tienen 0 hermanos/esposas a bordo, pero hay casos en los que tienen hasta 8 hermanos/esposas. En la época del titanic era muy común ver familias numerosas con muchos hermanos.
La variable Fare muestra también muchos outliers, pero teniendo en cuenta que en el barco hay 3 categorías

(primera, segunda y tercera clase) y que habría zonas del barco privilegiadas, los valores podrían ser normales. Vemos que la mayor parte están alrededor de 14.5.
Por último, "Parch" muestra culquier valor diferente de 0 como un outlier, esto implica que la inmensa mayoría de los registros tienen valor 0. Sin embargo, esta variable representa el número de padres/niños a bordo, y si nos fijamos, incluso el valor más extremo de valor 9, coincide con el valor extremo del número de hermanos/esposas, es decir, este valor extremo corresponde a la familia que tiene 9 hijos, los cuales tienen 8 hermanos.

Al finalizar el análisis de datos usando el dataset completo, restauramos los bloques de entrenamiento y test con los datos ya limpios usando el passengerId, y los guardamos como csv en el directorio de datos:

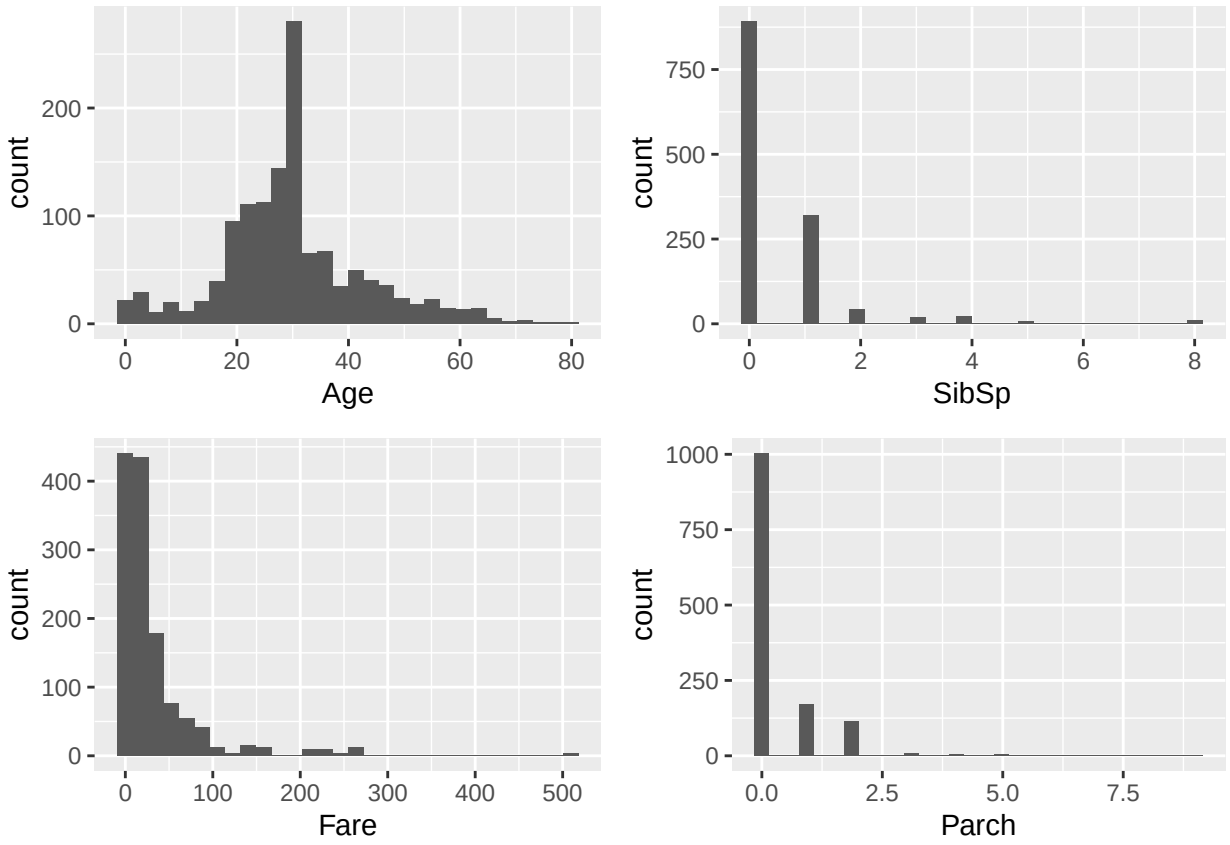
```
train_clean <- merged_data[merged_data$PassengerId %in% c(1:891), ]
test_clean <- merged_data[merged_data$PassengerId %in% c(892:1309), ]
write.csv(train_clean, "../data/train_clean.csv")
write.csv(test_clean, "../data/test_clean.csv")
```

## 4. Análisis de los datos

Comprobamos la normalidad de las variables cuantitativas utilizando histogramas para visualizar los datos y el test de Shapiro-Wilk:

```
g5 <- ggplot(merged_data, aes(x=Age)) + geom_histogram()
g6 <- ggplot(merged_data, aes(x=SibSp)) + geom_histogram()
g7 <- ggplot(merged_data, aes(x=Fare)) + geom_histogram()
g8 <- ggplot(merged_data, aes(x=Parch)) + geom_histogram()
grid.arrange(g5,g6, g7, g8, nrow=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




Visualmente vemos que no parece haber una distribución normal, la variable que más se aproxima sería "Age", que, aunque tiene el valor de 30 disparado debido a la asignación de valores medios que hemos hecho anteriormente, no muestra normalidad. Realizamos la comprobación también usando el test de Shapiro-Wilk:

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

```
##
##  Shapiro-Wilk normality test
##
## data:  merged_data$Age
## W = 0.9613, p-value < 2.2e-16
```

```
shapiro.test(merged_data$SibSp)
```

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

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

```
##
##  Shapiro-Wilk normality test
##
## data:  merged_data$Fare
## W = 0.52584, p-value < 2.2e-16
```

```r
shapiro.test(merged_data$Parch)
```

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

Como sospechábamos, ninguna de las variables sigue una distribución normal, ya que tienen valores de p-value muy por debajo del nivel de significancia(0.05).

Como los datos no tienen una distribución normal, comprobaremos la homocedasticidad utilizando el test de Fligner-Killeen:

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

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Age by Survived
## Fligner-Killeen:med chi-squared = 4.5372, df = 1, p-value = 0.03317
```

```r
fligner.test(SibSp ~ Survived, data = merged_data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  SibSp by Survived
## Fligner-Killeen:med chi-squared = 1.2514, df = 1, p-value = 0.2633
```

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

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fare by Survived
## Fligner-Killeen:med chi-squared = 87.002, df = 1, p-value < 2.2e-16
```

```r
fligner.test(Parch ~ Survived, data = merged_data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Parch by Survived
## Fligner-Killeen:med chi-squared = 11.253, df = 1, p-value = 0.0007948
```

En la mayor parte de los casos no se cumple la homocedasticidad, excepto en la variable SibSp, que sí presenta varianzas estadísticamente similares para los diferentes grupos de supervivencia.

Como nuestros datos no siguen una distribución normal, usaremos el coeficiente de correlación de Spearman para analizar cuál de las diferentes variables cuantitativas influye más en las probabilidades de supervivencia

```r
cor.test(merged_data$Survived, merged_data$Age, method="spearman")
```

```
## Warning in cor.test.default(merged_data$Survived, merged_data$Age, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
```

```
## data:  merged_data$Survived and merged_data$Age
## S = 125575645, p-value = 0.05177
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##         rho
## -0.06518269
```

```
cor.test(merged_data$Survived, merged_data$Fare, method="spearman")
```

```
## Warning in cor.test.default(merged_data$Survived, merged_data$Fare, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  merged_data$Survived and merged_data$Fare
## S = 82475354, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.3004112
```

```
cor.test(merged_data$Survived, merged_data$SibSp, method="spearman")
```

```
## Warning in cor.test.default(merged_data$Survived, merged_data$SibSp, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  merged_data$Survived and merged_data$SibSp
## S = 107413073, p-value = 0.007941
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##        rho
## 0.08887948
```

```
cor.test(merged_data$Survived, merged_data$Parch, method="spearman")
```

```
## Warning in cor.test.default(merged_data$Survived, merged_data$Parch, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  merged_data$Survived and merged_data$Parch
## S = 101590881, p-value = 3.454e-05
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.1382656
```

Excepto la variable "Age", las demás no muestran relación con la supervivencia, e incluso en el caso de "Age" esta relación no es muy fuerte. Por tanto no podemos confiar mucho en estas variables para determinar si un pasajero vive o muere.

Comprobamos el porcentaje de supervivencia de los pasajeros que contiene el set de entrenamiento:

```
summarize(train_clean, Survived=mean(Survived), .groups="drop")
```

```
##    Survived
## 1 0.3838384
```

```
summarize(group_by(train_clean, Sex), Survived=mean(Survived), .groups="drop")
```

```
## # A tibble: 2 x 2
##   Sex    Survived
##   <fct>     <dbl>
## 1 female    0.742
## 2 male      0.189
```

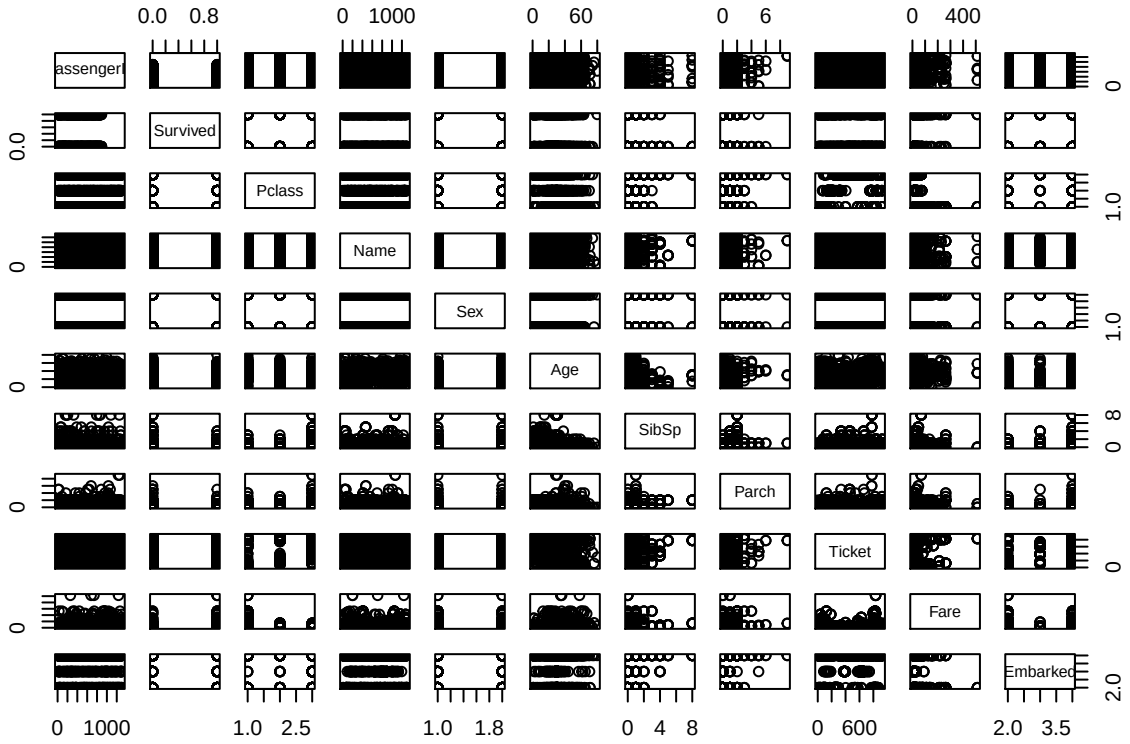
```
summarize(group_by(train_clean, Pclass), Survived=mean(Survived), .groups="drop")
```

```
## # A tibble: 3 x 2
##   Pclass Survived
##   <fct>     <dbl>
## 1 1         0.630
## 2 2         0.473
## 3 3         0.242
```

Y vemos que de los 891 pasajeros, sólo han sobrevivido el 38%. De los cuales un 74,2% fueron mujeres y el 18,8% restante fueron hombres. Clasificándolos por la clase del pasaje, vemos que el 62,9% de los pasajeros de primera clase han sobrevivido, frente al 47,3% de los de segunda clase y el 24,2% de los de tercera.

Por otro lado, comprobamos gráficamente si alguna de las variables tiene una relación lineal que permita aplicar un modelo de regresión lineal, pero como vemos en la siguiente figura, no parece haber ninguna relación lineal en el dataset:

```
plot(merged_data)
```




Por este motivo optaremos por otros tipos de modelo, como el de regresión logística, el randomForest o el treeBased. Pero antes comprobamos si el dataset está equilibrado para evitar sesgos durante el entrenamiento. Para ello analizamos la cantidad de supervivientes y muertos que contiene el dataset de entrenamiento:

```
table(train_clean$Survived)
```

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

Vemos que el número de muertos casi duplica el de supervivientes, por lo que optamos por crear unos nuevos set de datos para entrenar los diferentes modelos:

```
# Comenzamos separando los supervivientes y los muertos en dos grupos
supervivientes <- train_clean[which(train_clean$Survived == 1), ]
muertos <- train_clean[which(train_clean$Survived == 0), ]

#Datos que usaremos para el entrenamiento
set.seed(300)
filas_supervivientes <- sample(1:nrow(supervivientes), 0.8*nrow(supervivientes))
filas_muertos <- sample(1:nrow(muertos), 0.8*nrow(supervivientes))
train_supervivientes <- supervivientes[filas_supervivientes, ]
train_muertos <- muertos[filas_muertos, ]
entrenamiento <- rbind(train_supervivientes, train_muertos)
# table(entrenamiento$Survived)

#Datos de prueba
prueba_supervivientes <- supervivientes[-filas_supervivientes, ]
```

```
prueba_muertos <- muertos[-filas_muertos, ]
pruebas <- rbind(prueba_supervivientes, prueba_muertos)
# table(pruebas$Survived)
```

A continuación creamos un modelo de regresión logística, un modelo de árbol de decisión y un randomForest, y analizaremos cuál de ellos es más eficiente:

```
# Modelos de regresión logística considerando diferentes variables
regModel <- glm(Survived ~ Age + Fare + Pclass + Embarked + Sex + SibSp + Parch,
                family=binomial(link=logit),
                data=entrenamiento)

# Árbol de decisión
dTree = rpart(formula = Survived ~ Age + Fare + Pclass + Embarked + Sex +
                        SibSp + Parch,
                        data = entrenamiento,
                        method = 'class')

# Construimos el modelo con todas las variables posibles
rf_model <- randomForest(factor(Survived) ~ Age + Fare + Pclass + Embarked +
                                            Sex + SibSp + Parch,
                                            data = entrenamiento)
```

## 5. Resultados

Hacemos predicciones con los 3 modelos generados:

```
# Predicción del modelo de regresión logística
reg_prob_pred = predict(regModel,
                        type = 'response',
                        newdata = pruebas[-2])
# El modelo de regresión logística devuelve probabilidades. Por este motivo
# buscamos el punto de decisión óptimo para considerar 0 o 1 cada resultado.
optCutOff <- optimalCutoff(pruebas$Survived, reg_prob_pred)[1]
reg_pred = ifelse(reg_prob_pred > optCutOff, 1, 0)

# Predicción del modelo de árbol de decisión
tree_pred = predict(dTree,
                    type = 'class',
                    newdata = pruebas[-2])

# Predicción del modelo randomForest
rf_pred = predict(rf_model,
                  type = 'class',
                  newdata = pruebas[-2])
```

```
reg_matrix = table(pruebas[, 2], reg_pred)
tree_matrix = table(pruebas[, 2], tree_pred)
rf_matrix = table(pruebas[, 2], rf_pred)
reg_matrix
```

```
##    reg_pred
##       0   1
##   0 266  10
##   1  28  41
```

```
tree_matrix
```

```
##    tree_pred
##       0   1
##   0 211  65
##   1  14  55
```

```
rf_matrix
```

```
##    rf_pred
##       0   1
##   0 221  55
##   1  10  59
```

Observando las matrices de confusión generadas vemos que el modelo de regresión logística ha acertado 307 predicciones, frente a los 266 aciertos del árbol de decisión y los 280 del modelo de randomForest, mientras que ha fallado 38 frente a los 79 del árbol y 65 del randomForest.

Comprobamos la exactitud aplicando la fórmula definida en los apuntes:

```
# Exactitud modelo de regresión logística
(reg_matrix[1,1] + reg_matrix[2,2]) / count(pruebas)$n
```

```
## [1] 0.8898551
```

```
# Exactitud modelo de árbol de decisión
(tree_matrix[1,1] + tree_matrix[2,2]) / count(pruebas)$n
```

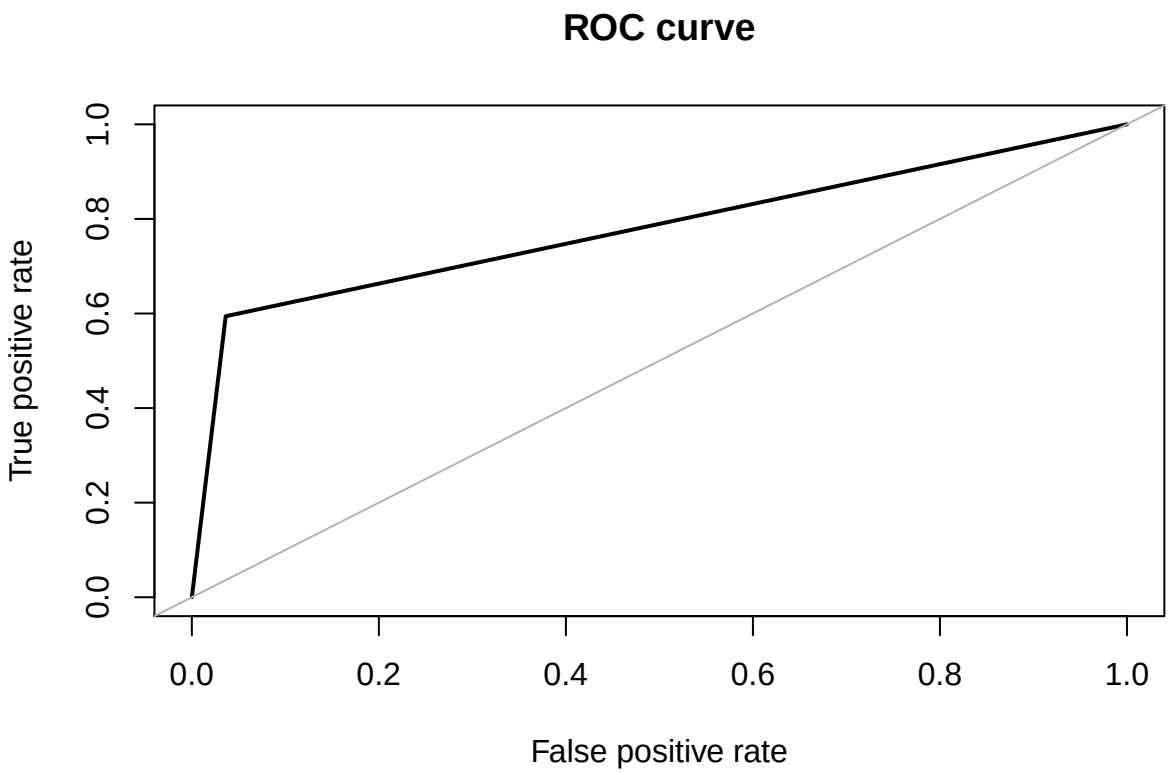
```
## [1] 0.7710145
```

```
# Exactitud modelo de randomForest
(rf_matrix[1,1] + rf_matrix[2,2]) / count(pruebas)$n
```

```
## [1] 0.8115942
```

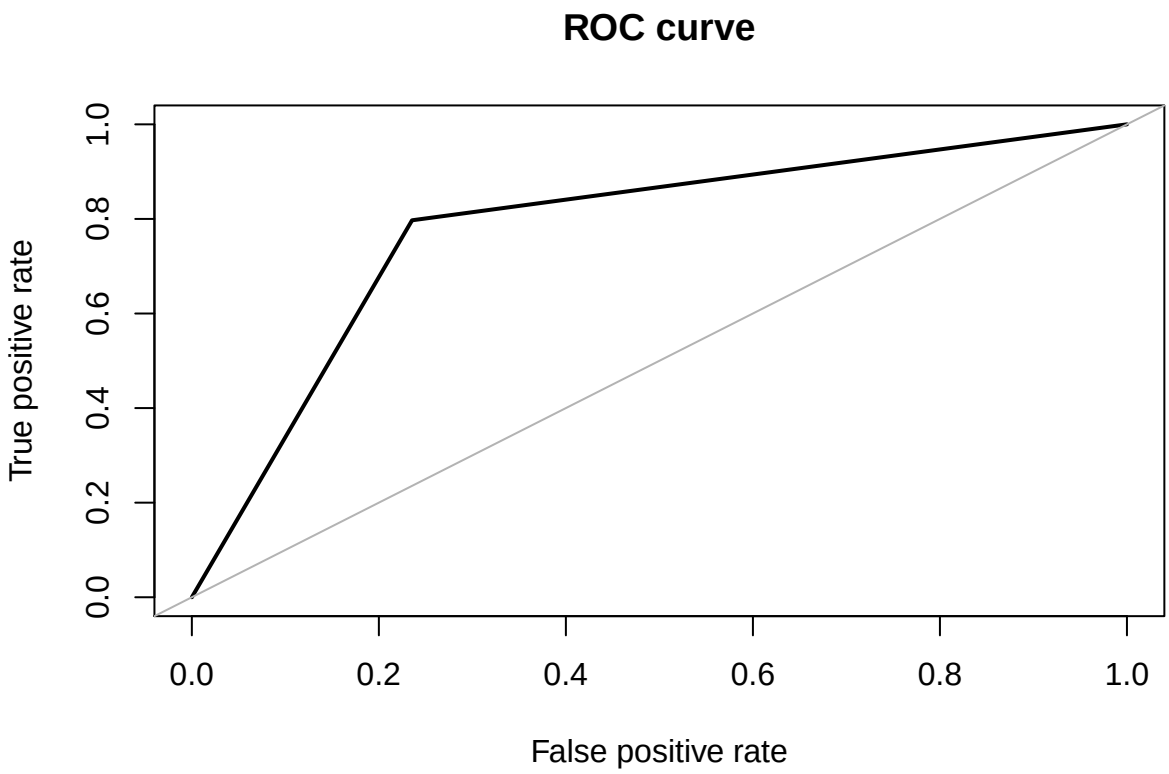
El modelo más exacto es el de regresión logística, mientras que el segundo más preciso es el de randomForest, siendo el del árbol de decisión el menos preciso de los 3. Además, vamos a desarrollar la curva ROC para ver la capacidad de los diferentes modelos para distinguir los posibles resultados.

```
# Curva ROC del modelo de regresión logística
par(mfrow = c(1, 1))
roc.curve(pruebas$Survived, reg_pred)
```



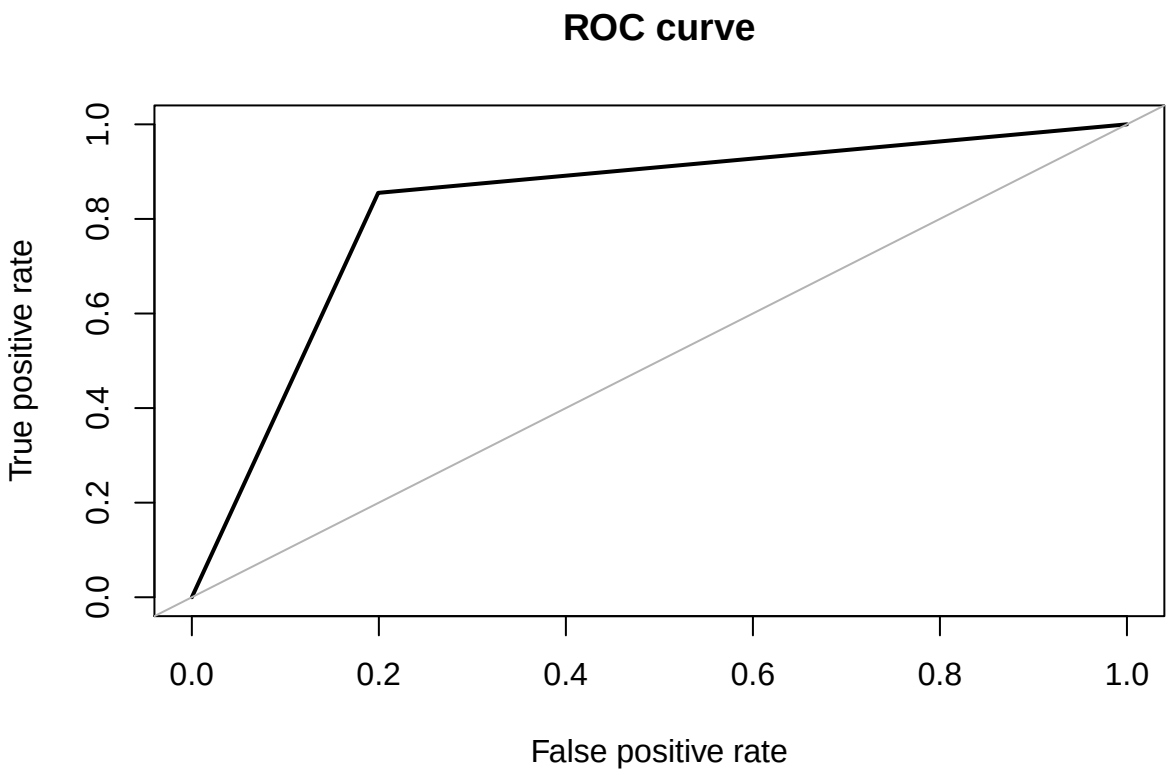

```
## Area under the curve (AUC): 0.779
```

```
roc.curve(pruebas$Survived, tree_pred)
```




```
## Area under the curve (AUC): 0.781
```

```
roc.curve(pruebas$Survived, rf_pred)
```




```
## Area under the curve (AUC): 0.828
```

vemos que los tres modelos tienen un área muy similar de 77,9%, 78,1% y 82,8% para los modelos de regresión, árbol de decisión y randomForest respectivamente.

También es interesante observar la importancia de cada variable tiene en los diferentes modelos entrenados. Para ver las diferentes importancias hacemos uso de la librería "caret":

```
varImp(regModel)
```

```
##              Overall
## Age        4.4438377
## Fare       0.7488688
## Pclass2    3.4868881
## Pclass3    6.4317988
## EmbarkedQ  0.3150451
## EmbarkedS  0.1271561
## Sexmale   10.3003655
## SibSp      2.7053609
## Parch      0.3748357
```

```
varImp(dTree)
```

```
##            Overall
## Age      32.194168
## Embarked  4.205793
## Fare     44.125780
## Parch    17.200516
## Pclass   62.695595
```

```
## Sex      75.399573
## SibSp     7.504584
```

```r
varImp(rf_model)
```

```
##            Overall
## Age      38.309152
## Fare     42.013741
## Pclass   26.537975
## Embarked  8.406589
## Sex      59.162439
## SibSp    10.946586
## Parch     9.818697
```

Vemos que en los 3 modelos la variable que más peso tiene es el sexo del pasajero, mientras que la segunda variable de más peso ya varía entre modelos. Por ejemplo, el modelo de regresión logística divide la variable Pclass en 2 (Pclass2 y Pclass3), y considera Pclass3 como la segunda de más peso para decidir. Por otro lado, el modelo del árbol de decisión también considera Pclass como la segunda variable de mayor peso pero sin dividirla, pero el modelo de randomForest considera "Fare" como el segundo factor para determinar el resultado, dejando a Pclass en cuarta posición, tras la variable Age.

## 6. Conclusiones

Tras analizar los resultados del apartado anterior, se considera que el modelo con más exactitud es el modelo de regresión logística con una exactitud del 88,9%, esto significa que el error de clasificación del modelo es de un 11,1%.

Sin embargo, analizando la curva ROC vemos que la precisión del modelo no es tan precisa como parece con el análisis de exactitud, ya que el modelo de randomForest presenta un área mayor bajo la curva ROC, lo que significa que se desarrollará mejor que el modelo de regresión logística.

Por este motivo, se usará el modelo randomForest para hacer la predicción de los supervivientes sobre el set de datos de test y lo guardaremos en un fichero csv que se podrá subir a Kaggle:

```r
# Usamos los datos del test_clean para la predicción
prediccion <- predict(rf_model, test_clean)

# Guardamos un dataframe con 2 columnas, el Id del pasajero y el resultado de
# la predicción.
solution <- data.frame(PassengerID = test_clean$PassengerId,
                       Survived = prediccion)

# Creamos un fichero con la solución para poder subirlo a Kaggle
write.csv(solution, file = '../data/randForestSolution.csv', row.names = F)
```

| Contribuciones | Firma |
|---|---|
| Investigación previa | isrosrey |
| Redacción de las respuestas | isrosrey |
| Desarrollo código | isrosrey |