Tipología y ciclo de vida de los datos - Practica 2: Limpieza y validación de los datos

Jover Erreyes Pilozo

January 2022

1. Descripción de la Práctica a realizar

El objetivo de esta actividad será el tratamiento de un dataset, que puede ser el creado en la práctica 1 o bien cualquier dataset libre disponible en Kaggle (https://www.kaggle.com). Algunos ejemplos de dataset con los que podéis trabajar son:

- Red Wine Quality (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009)
- Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic)

El último ejemplo corresponde a una competición activa de Kaggle de manera que, opcionalmente, podéis aprovechar el trabajo realizado durante la práctica para entrar en esta competición.

Siguiendo las principales etapas de un proyecto analítico, las diferentes tareas a realizar (y justificar) son las siguientes:

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?
2. Integración y selección de los datos de interés a analizar.
3. Limpieza de los datos.
   3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?
   3.2. Identificación y tratamiento de valores extremos.
4. Análisis de los datos.
   4.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).
   4.2. Comprobación de la normalidad y homogeneidad de la varianza.
   4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.
5. Representación de los resultados a partir de tablas y gráficas.
6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?
7. Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.

2. Resolución

2.1 Descripción del dataset escogido

A través del siguiente enlace de Kaggle (https://www.kaggle.com/adityakadiwal/water-potability) se ha obtenido el dataset que se trabajará a lo largo de la práctica con el objetivo de llevar a cabo tareas de

limpieza y validación de datos. Este dataset cuenta con 10 variables o columnas y 3276 registros, estas 10 variables són las siguientes:

1. pH value: Ofrece información sobre el ph, que a su vez indica la condición ácida o alcalina del estado del agua, según la OMS el límite máximo permitido es entre 6.5 y 8.5.

2. Hardness: La dureza es la capacidad del agua para diluir el jabón causado por el calcio y el magnesio.

3. Solids (Total dissolved solids - TDS): El agua tiene la capacidad de disolver una amplia gama de minerales o sales inorgánicos y algunos orgánicos. Estos minerales producen un sabor no deseado y un color diluido en apariencia de agua. Este es el parámetro importante para el uso del agua. El agua con alto valor de TDS indica que el agua está altamente mineralizada. El límite deseable de TDS es de 500 mg/l y el límite máximo es de 1000 mg/l que se prescribe para beber.

4. Chloramines: El cloro y la cloramina son los principales desinfectantes que se utilizan en los sistemas públicos de agua. Las cloraminas se forman con mayor frecuencia cuando se agrega amoníaco al cloro para tratar el agua potable. Los niveles de cloro de hasta 4 miligramos por litro (mg/L o 4 partes por millón (ppm)) se consideran seguros en el agua potable.

5. Sulfate: Los sulfatos son sustancias naturales que se encuentran en minerales, suelo y rocas. Están presentes en el aire ambiente, el agua subterránea, las plantas y los alimentos. El principal uso comercial del sulfato es la industria química. La concentración de sulfato en el agua de mar es de aproximadamente 2700 miligramos por litro (mg/L). Varía de 3 a 30 mg/L en la mayoría de los suministros de agua dulce, aunque se encuentran concentraciones mucho más altas (1000 mg/L) en algunas ubicaciones geográficas.

6. Conductivity: El agua pura no es un buen conductor de corriente eléctrica, más bien es un buen aislante. El aumento de la concentración de iones mejora la conductividad eléctrica del agua. Generalmente, la cantidad de sólidos disueltos en el agua determina la conductividad eléctrica. La conductividad eléctrica (EC) en realidad mide el proceso iónico de una solución que le permite transmitir corriente. Según los estándares de la OMS, el valor de CE no debe exceder los 400 μS/cm.

7. Organic_carbon: El carbono orgánico total (TOC) en las fuentes de agua proviene de la materia orgánica natural en descomposición (NOM), así como de fuentes sintéticas. TOC es una medida de la cantidad total de carbono en compuestos orgánicos en agua pura. Según la EPA de EE.UU. el nivel de carbono orgánico permitido es de $<$2 mg/L en agua potable y $<$4 mg/L en el agua (de origen) para ser tratada.

8. Trihalomethanes: Los THM son sustancias químicas que se pueden encontrar en el agua tratada con cloro. La concentración de THM en el agua potable varía según el nivel de material orgánico en el agua, la cantidad de cloro necesaria para tratar el agua y la temperatura del agua que se está tratando. Los niveles de THM de hasta 80 ppm se consideran seguros en el agua potable.

9. Turbidity: La turbidez del agua depende de la cantidad de materia sólida presente en estado suspendido. Es una medida de las propiedades emisoras de luz del agua y la prueba se utiliza para indicar la calidad de la descarga de desechos con respecto a la materia coloidal. El valor medio recomendado por la OMS es de 5,00 NTU.

10. Potability: Indica si el agua es segura para el consumo humano, donde 1 significa potable y 0 significa no potable.

El objetivo de escoger este dataset es el de aplicar un algoritmo de regresión logística o un árbol de decisión que permita clasificar a partir de ciertas características si el agua de un determinado lugar es agua potable o no.

2.2 Selección de datos

A partir de las variables descritas en el apartado anterior se advierte que cada una de ellas es importante para determinar si el agua es potable o no, la única que en primera instancia puede parecer prescindible en el análisis es la variable '*Hardness*' pero será necesario realizar un análisis estadístico para observar la

correlación que dicha variable guarda con el resto para poder determinar si es posible obviarla o no. Por otro lado, al observar los datos con detenimiento se advierten valores vacíos o NA's, por tanto será necesario llevar a cabo un proceso de limpieza de datos y estudiar si existen datos outliers o extremos.

```
# Se establece work directory
#setwd("
#C:/Users/Willy/Desktop/Master UOC/1r Semestre/Tipología y ciclo de vida de los datos/Practica 2 -")

# Se cargará el set de datos y se almacenará en la variable 'data'
data <- read.csv("water_potability.csv", sep = ",", stringsAsFactors = TRUE)
head(data)
```

```
##         ph Hardness   Solids Chloramines  Sulfate Conductivity Organic_carbon
## 1       NA 204.8905 20791.32    7.300212 368.5164     564.3087      10.379783
## 2 3.716080 129.4229 18630.06    6.635246       NA     592.8854      15.180013
## 3 8.099124 224.2363 19909.54    9.275884       NA     418.6062      16.868637
## 4 8.316766 214.3734 22018.42    8.059332 356.8861     363.2665      18.436524
## 5 9.092223 181.1015 17978.99    6.546600 310.1357     398.4108      11.558279
## 6 5.584087 188.3133 28748.69    7.544869 326.6784     280.4679       8.399735
##   Trihalomethanes Turbidity Potability
## 1        86.99097  2.963135          0
## 2        56.32908  4.500656          0
## 3        66.42009  3.055934          0
## 4       100.34167  4.628771          0
## 5        31.99799  4.075075          0
## 6        54.91786  2.559708          0
```

2.3 Limpieza de datos

Como se ha comentado anteriormente tras una primera observación de los datos se advierten diversos valores vacíos o Na's por tanto será necesario tratar dichos valores, en primera instancia se pensaba aplicar el método kNN ya que es uno de los métodos más populares para tratar datos vacíos pero se ha optado finalmente por aplicar el método missforest() ya que este es más robusto al no ser tan sensible a los cambios en el valor k.

```
# Se calcula la estructura del dataset, se observan las
# variables y el número de registros
str(data)
```

```
## 'data.frame':    3276 obs. of  10 variables:
##  $ ph             : num  NA 3.72 8.1 8.32 9.09 ...
##  $ Hardness       : num  205 129 224 214 181 ...
##  $ Solids         : num  20791 18630 19910 22018 17979 ...
##  $ Chloramines    : num  7.3 6.64 9.28 8.06 6.55 ...
##  $ Sulfate        : num  369 NA NA 357 310 ...
##  $ Conductivity   : num  564 593 419 363 398 ...
##  $ Organic_carbon : num  10.4 15.2 16.9 18.4 11.6 ...
##  $ Trihalomethanes: num  87 56.3 66.4 100.3 32 ...
##  $ Turbidity      : num  2.96 4.5 3.06 4.63 4.08 ...
##  $ Potability     : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
# También se puede aplicar dim para observar las dimensiones del set
dim(data)
```

```
## [1] 3276   10
```

```
# Se calcula si existen valores vacíos
sum(is.na(data))
```

```
## [1] 1434
```

```
#Cargamos missForest para imputar los valores perdidos
if( !require('missForest')) install.packages('missForest')
```

```
## Loading required package: missForest

## Warning: package 'missForest' was built under R version 4.1.2

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 4.1.2

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.1.2

## Loading required package: itertools

## Warning: package 'itertools' was built under R version 4.1.2

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 4.1.2
```

```
library('missForest')
if( !require('VIM')) install.packages("VIM")
```

```
## Loading required package: VIM

## Warning: package 'VIM' was built under R version 4.1.2

## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
```

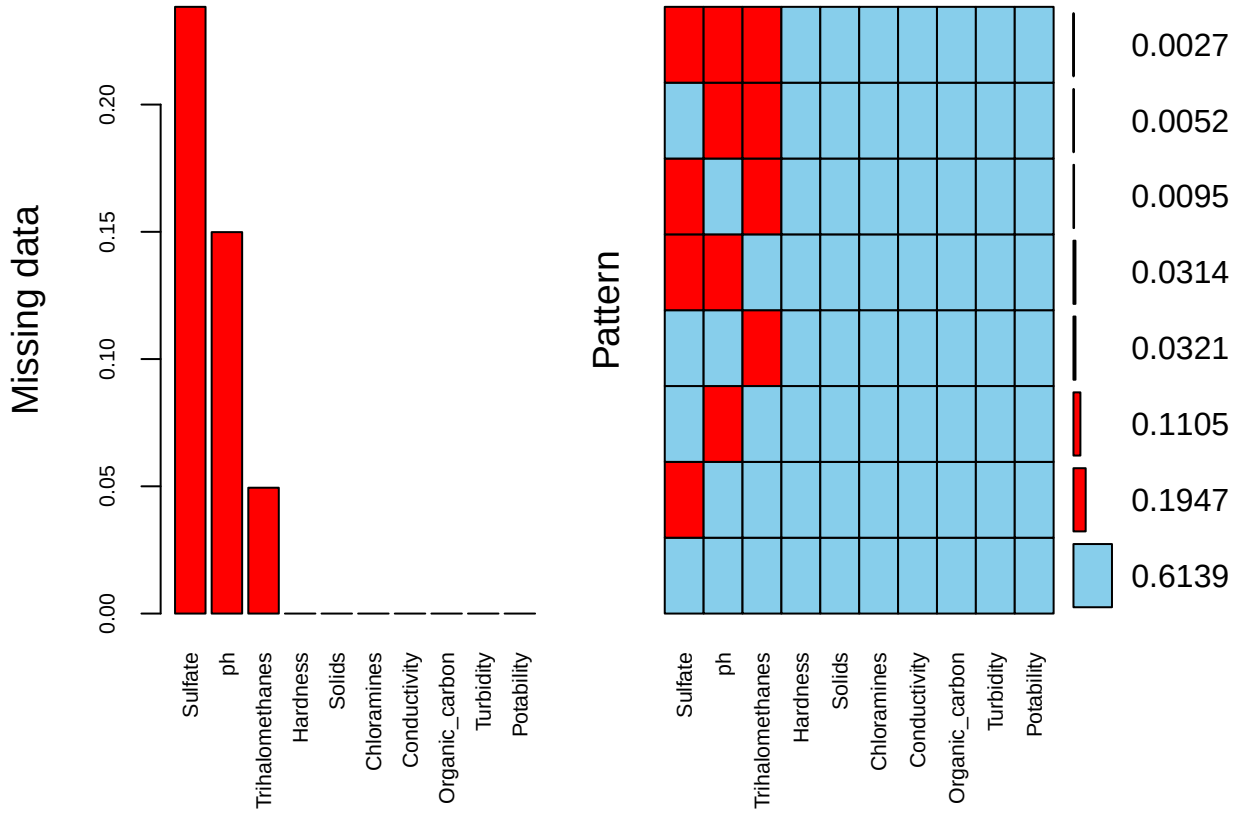
```
##
## Attaching package: 'VIM'
```

```
## The following object is masked from 'package:missForest':
##
##     nrmse
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
library('VIM')

#Se Representa los valores perdidos para cada variable
aggr(data, numbers=TRUE, sortVars=TRUE, labels=names(data),
     cex.axis=.7, gap=3, ylab=c("Missing data","Pattern"))
```




```
##
##  Variables sorted by number of missings:
##          Variable      Count
##           Sulfate 0.23840049
##                ph 0.14987790
##  Trihalomethanes 0.04945055
##          Hardness 0.00000000
##            Solids 0.00000000
```

```
##      Chloramines 0.00000000
##     Conductivity 0.00000000
##   Organic_carbon 0.00000000
##        Turbidity 0.00000000
##       Potability 0.00000000
```

Se puede observar gráficamente como los valores perdidos se concentrar en general en la variable 'Sulfate', 'ph' y 'Trihalomethanes' teniendo esto en cuenta se procurará aplicar missforest a dichas columnas para tratar estos valores y de esta manera poder trabajar con el dataset mas adelante.

```
# Se aplica metodo de imputación missForest()
data_missforest<-missForest(data, variablewise = TRUE)
```

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
```

```
# Se observa el resultado
head(data_missforest$ximp)
```

```
##         ph Hardness   Solids Chloramines  Sulfate Conductivity Organic_carbon
## 1 7.482528 204.8905 20791.32    7.300212 368.5164     564.3087      10.379783
## 2 3.716080 129.4229 18630.06    6.635246 339.9528     592.8854      15.180013
## 3 8.099124 224.2363 19909.54    9.275884 318.1358     418.6062      16.868637
## 4 8.316766 214.3734 22018.42    8.059332 356.8861     363.2665      18.436524
## 5 9.092223 181.1015 17978.99    6.546600 310.1357     398.4108      11.558279
## 6 5.584087 188.3133 28748.69    7.544869 326.6784     280.4679       8.399735
##   Trihalomethanes Turbidity Potability
## 1        86.99097  2.963135          0
## 2        56.32908  4.500656          0
## 3        66.42009  3.055934          0
## 4       100.34167  4.628771          0
## 5        31.99799  4.075075          0
## 6        54.91786  2.559708          0
```

```
class(data_missforest)
```

```
## [1] "missForest"
```

Se advierte tras una primera observación que no existen o parecen no existir valores vacíos o perdidos, por tanto, el método missForest parece haber tenido éxito, por otro lado, se advierte que el resultado obtenido no es un dataframe que es lo que nos interesa ya que se trabajará a partir de él, por ello se debe realizar las siguientes transformaciones para volver a obtenerlo:

```
# Obtenemos nuevo dataset sin valores vacíos
data_mf <- data_missforest$ximp

# Se confirma que no tenemos valores vacíos
which(is.na(data_mf))
```

```
## integer(0)
```

```r
# Obtenemos una visualización del resultado
head(data_mf)
```

```
##         ph Hardness   Solids Chloramines  Sulfate Conductivity Organic_carbon
## 1 7.482528 204.8905 20791.32    7.300212 368.5164     564.3087      10.379783
## 2 3.716080 129.4229 18630.06    6.635246 339.9528     592.8854      15.180013
## 3 8.099124 224.2363 19909.54    9.275884 318.1358     418.6062      16.868637
## 4 8.316766 214.3734 22018.42    8.059332 356.8861     363.2665      18.436524
## 5 9.092223 181.1015 17978.99    6.546600 310.1357     398.4108      11.558279
## 6 5.584087 188.3133 28748.69    7.544869 326.6784     280.4679       8.399735
##   Trihalomethanes Turbidity Potability
## 1        86.99097  2.963135          0
## 2        56.32908  4.500656          0
## 3        66.42009  3.055934          0
## 4       100.34167  4.628771          0
## 5        31.99799  4.075075          0
## 6        54.91786  2.559708          0
```

```r
str(data_mf)
```

```
## 'data.frame':    3276 obs. of  10 variables:
##  $ ph             : num  7.48 3.72 8.1 8.32 9.09 ...
##  $ Hardness       : num  205 129 224 214 181 ...
##  $ Solids         : num  20791 18630 19910 22018 17979 ...
##  $ Chloramines    : num  7.3 6.64 9.28 8.06 6.55 ...
##  $ Sulfate        : num  369 340 318 357 310 ...
##  $ Conductivity   : num  564 593 419 363 398 ...
##  $ Organic_carbon : num  10.4 15.2 16.9 18.4 11.6 ...
##  $ Trihalomethanes: num  87 56.3 66.4 100.3 32 ...
##  $ Turbidity      : num  2.96 4.5 3.06 4.63 4.08 ...
##  $ Potability     : int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
dim(data_mf)
```

```
## [1] 3276   10
```

Tras finalizar el proceso de imputación se confirma que no existen valores vacios en el nuevo set de datos, que la estructura del mismo continua intacta, por tanto el proceso se ha llevado a cabo con éxito. A continuación, a lo largo del proceso de limpieza de datos otra de las características que se deben analizar es si existen ceros en el set de datos.

```r
# Comprobación de si existen valores ceros en el set de datos
length(which(data_mf=="0"))
```

```
## [1] 1999
```

Se advierte que existen valores igual a 0, pero es necesario evaluar si estos tienen sentido dentro del set de datos o si será necesario tratarlos. Tras observar el dataset a lo largo de todo el proceso seguido, se advierte que lo más probable es que estos '0' se encuentren dentro de la variable *Potability* ya que esta indica con '0' cuando el agua no es potable y con '1' cuando sí lo es.

```
# Se comprueba el número de 0 de la variable potability
length(data_mf$Potability[data_mf$Potability=="0"])
```

```
## [1] 1998
```

```
# Se comprueba el número de 1 de la variable potability
length(data_mf$Potability[data_mf$Potability=="1"])
```

```
## [1] 1278
```

```
# Se confirma que el número total cuadra con el total de registros
length(data_mf$Potability[data_mf$Potability=="1"])+
  length(data_mf$Potability[data_mf$Potability=="0"])
```

```
## [1] 3276
```

Se observa que la gran mayoría de '0' se encuentra en esta variable y no es necesario tratarlos, por tanto, se puede avanzar a comprobar si existen valores *outliers* o extremos. Para ello se aplicarán gráficos boxplot y funciones boxplot.stats para cada una de las variables y a partir de ellos decidir que acciones llevar a cabo.

```
# Se obtienen los valores outliers de la variable ph
boxplot.stats(data_mf$ph)$out
```

```
##   [1]  3.7160801 11.1802845  3.4450619 10.4332910  3.6416298 10.6829664
##   [7] 10.5607446  3.5145455  3.7225005  1.8445384  2.6120359  3.3880906
##  [13] 11.1806947  3.7197916 11.2678284  3.6647106 13.1754017 10.7618978
##  [19] 11.3017940  3.6571231  3.5515792  2.7985491 10.4645025  3.4103597
##  [25] 10.5096686  3.6231658  1.7570371  3.6810763  3.6331628  0.2274991
##  [31]  3.4264504 10.5380984 11.8980780 10.9050764  0.9899122 10.6955325
##  [37] 10.6329096 10.6780563 10.8112895 10.5813860 11.0278799  3.3445885
##  [43] 10.7743181 10.4150440 11.2445071 10.4868084  3.6371706 10.5502733
##  [49]  2.6908312  3.4338741 12.2469281  2.5692436 11.5348805  3.5908224
##  [55]  3.4348558 10.8481304 14.0000000 11.5687680  3.7177039  3.7151714
##  [61] 10.5047602  3.6919327  2.8035631  2.5581028 10.5779305 10.3919424
##  [67] 11.2354260 10.6288209 10.5246020  2.9744294 10.4907440  2.5381158
##  [73]  2.9454691 11.0694563 10.8873050  3.1487123 11.2191347  3.2309731
##  [79] 11.9077398  3.4223942 10.6005900  2.3767681  3.6768450  3.2728201
##  [85] 13.5412402 10.9474726  3.7620976 13.3498886  1.4317816 10.8518603
##  [91]  3.2616698 11.5631691  0.9755780  2.9251743 10.8934847 11.4967025
##  [97] 10.9331107  3.7301280  3.6784318  3.1020756  0.0000000 11.4968589
## [103] 11.6211401 11.3905431 10.8178311  2.1285314  1.9853834  3.3376290
## [109] 11.4497393 10.5037865 10.4856036 10.6673639 10.8081569  3.6299221
## [115] 11.4910109
```

Se advierten valores atípicos en cuanto a la media pero no rompen de manera notoria la tendencia, si estos fueran valores más extremos como por ejemplo '999' o '5000' o cualquier valor de ese estilo, se procedería a marcarlos como NA y realizar el mismo proceso anterior de missforest() para asignarles un valor más adecuado pero en este caso no será necesario, este será el razonamiento que se siga con los siguientes análisis de valores extremos. Por otro lado, se ha detectado el valor '0' que faltaba del análisis anterior, al ser un único valor no debería tener un efecto significativo en el análisis y al existir valores próximos a '0' no se sabe a ciencia cierta si se trata de un error al introducir dicho valor o un valor correcto, de todas formas se procederá a sustituir dicho valor por la media de la variable 'ph'.

```
# Sustituimos el valor '0' por la media de la variable
data_mf$ph[data_mf$ph == 0]
```

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

```
data_mf$ph[data_mf$ph == 0] <- mean(data_mf$ph)

# Se comprueba que ya existe ningún valor igual a 0
data_mf$ph[data_mf$ph == 0]
```

```
## numeric(0)
```

Se continua con la variable 'Hardness' y la siguiente a esta:

```
boxplot.stats(data_mf$Hardness)$out
```

```
##  [1] 279.35717 304.23591 100.45762 103.46476 116.29933 300.29248 104.75242
##  [8] 281.26867 278.05632 284.09835 112.29949 105.85926 276.73357 112.82025
## [15] 103.17359  98.77164 280.08241 278.58511  47.43200 280.08965  81.71090
## [22] 113.83111  77.45959 307.70602  94.09131 282.73902 278.14752 323.12400
## [29] 311.38396 291.46190  73.49223 308.25383 281.58216 286.20176 306.62748
## [36] 108.69908 287.37021 106.38011  97.28091 283.99728  98.45293 116.06195
## [43] 279.23242 107.38333 277.11695 116.72512 113.50470 115.39298 117.05731
## [50] 110.86579 278.34036 281.59423 108.91663 278.03636 283.40957 100.80652
## [57] 107.34198 111.24641 116.90548 278.08145 283.89586 116.33828 276.69976
## [64] 286.56799 298.09868 113.02447 278.23175 111.47858 113.17596 114.46390
## [71] 111.99403 114.73354 278.61945 317.33812 114.37145  94.90898 110.90360
## [78]  98.36791 287.97554 303.70263 114.80758 277.06571  94.81255
```

```
boxplot.stats(data_mf$Solids)$out
```

```
##  [1] 46140.13 45222.51 48621.56 46113.96 52318.92 45249.45 46931.88 45510.58
##  [9] 49074.73 44868.46 52060.23 48002.08 56867.86 55334.70 48410.47 45141.69
## [17] 56351.40 44896.98 45166.91 48204.17 45166.64 45939.69 46718.56 56488.67
## [25] 45243.03 49125.36 50279.26 45041.15 46077.36 49009.92 47022.75 47852.89
## [33] 49341.42 61227.20 48175.85 47591.28 45050.00 45148.81 49456.59 44982.73
## [41] 50793.90 56320.59 53735.90 50166.53 51731.82 48007.87 47580.99
```

```
boxplot.stats(data_mf$Chloramines)$out
```

```
##  [1] 12.5800265 13.0438061  0.5303513 12.9121866 12.3632848 11.1707886
##  [7] 13.1270000  2.4843800 12.0625362  2.9813790  2.9937441 11.5861511
## [13]  1.6839926 11.5431905 11.2086883 11.2515073  2.7508373  2.5775553
## [19] 11.5235975 12.2793742  2.8660730  2.8625354 11.3028312  2.7417121
## [25] 11.1431096 11.1291537  2.4560136  3.1395527  2.6212676 11.2643858
## [31] 11.2400004  2.4586092  2.7857184 11.7539037  2.4985967  0.3520000
## [37]  3.1174410 12.6268997 11.1016281  2.8557898  1.3908709  2.3979850
## [43] 11.9942902 11.4484693  1.9202714 12.2463941 12.2271753 11.9304480
## [49]  2.3866535  2.1026910  3.1248326  2.7267656 11.2993902  2.6544910
## [55]  3.0743161 11.2243946  2.5622555 12.6533620 11.9960151  2.6483899
## [61]  3.0160326
```

```
boxplot.stats(data_mf$Sulfate)$out
```

```
##   [1] 247.2008 426.5436 241.6075 433.9522 240.9367 424.7880 234.8527 416.0835
##   [9] 187.1707 432.5564 247.3354 192.0336 180.2067 243.4859 255.0432 233.7926
##  [17] 412.6901 256.6301 444.9706 414.6367 182.3974 217.0006 187.4241 253.8306
##  [25] 209.4711 256.4738 422.9904 435.6728 223.2358 429.0223 224.2125 245.9543
##  [33] 415.2871 445.9384 229.5756 230.5559 211.8516 422.4457 252.2313 246.9426
##  [41] 424.6890 444.3757 244.2851 421.3432 251.0624 409.9885 240.1985 253.1585
##  [49] 244.7952 219.1489 419.8812 413.5604 255.9767 248.0948 257.2766 219.5534
##  [57] 227.3485 421.7220 234.2856 455.4512 415.9279 239.9119 203.4445 245.7289
##  [65] 418.3094 442.7614 433.4482 423.9520 417.9118 429.8143 247.9349 411.1007
##  [73] 429.0456 418.4706 206.2472 252.4665 441.5877 475.7375 129.0000 247.1800
##  [81] 462.4742 248.3044 421.4861 423.8763 421.3940 436.2941 241.4469 476.5397
##  [89] 254.7299 256.5905 417.6024 425.7103 449.2677 423.1875 234.6098 225.5166
##  [97] 247.6457 255.5447 411.5808 460.1071 427.1845 445.3595 458.4411 415.4508
## [105] 214.4608 256.0895 248.7120 433.6339 417.2454 418.4942 416.6984 251.3839
## [113] 437.5923 414.8558 207.8905 439.7879 251.4435 252.1089 240.8976 417.3403
## [121] 257.4595 410.3587 423.0460 235.9955 416.5305 416.9492 232.5488 447.4180
## [129] 237.5175 205.9351 427.3778 416.4952 253.4335 238.4466 233.8703 481.0306
## [137] 433.0215 410.4593 418.0000 416.8885 437.6472 419.7164 418.5593 412.0371
## [145] 418.2472 253.5802 241.1263 433.6943 435.1152 231.0537 450.9145 412.4073
## [153] 418.2409 422.7691 418.9550 227.6656 254.3588 253.4812 413.1022 426.1575
## [161] 235.7710 441.8268 424.3021 440.6355 411.2853 257.1487 244.0863 409.9187
## [169] 231.7242 446.7240 252.0673 426.5000 238.8440 419.2163 416.2628 413.9140
## [177] 254.0410
```

```
boxplot.stats(data_mf$Conductivity)$out
```

```
##  [1] 669.7251 672.5570 695.3695 656.9241 660.2549 666.6906 181.4838 708.2264
##  [9] 753.3426 657.5704 674.4435
```

```
boxplot.stats(data_mf$Organic_carbon)$out
```

```
##  [1] 23.917601 23.399516  5.315287 23.373265 23.514774 23.569645  2.200000
##  [8]  4.966862 23.952450  4.371899  4.861631  5.218233  4.473092  5.051695
## [15] 28.300000 23.317699 24.755392  5.188466  4.902888  4.466772 27.006707
## [22]  5.196717 23.667667  5.159380 23.604298
```

```
boxplot.stats(data_mf$Trihalomethanes)$out
```

```
##  [1]  17.915723 110.739299  23.817020  25.525267  18.400012 107.754043
##  [7]   8.175876 112.622733 107.189584  20.337753  26.505484  23.792950
## [13] 120.030077  25.057375  19.175175  21.355275  18.101222 107.900842
## [19]  23.136611 107.282329  17.000683 108.849568   8.577013 116.161622
## [25] 118.357275 114.208671  26.140863  16.291505  17.527765  22.749735
## [31]  22.219327 108.589414 110.431080 107.610806 113.048886  24.532773
## [37] 107.585967 112.061027  15.684877 112.412210  18.015272 124.000000
## [43]  24.914971  25.061904  24.734227 108.265227  23.075806  14.343161
## [49] 111.115310   0.738000 107.306343 111.595448 108.213981 114.034946
```

```
boxplot.stats(data_mf$Turbidity)$out
```

```
##  [1] 6.204846 6.494249 6.739000 1.680554 1.812529 6.357439 6.389161 1.496101
##  [9] 1.687625 1.492207 1.659799 6.226580 6.099632 1.641515 6.307678 1.844372
## [17] 6.494749 1.450000 1.801327
```

```
boxplot.stats(data_mf$Potability)$out
```

```
## integer(0)
```

Tras observar los valores outliers obtenidos se ha tomado la decisión de no tratarlos ya que se considera que entran dentro de las posibles medidas pese a que se alejen de la media, ya que muchos de estos valores son muy parecidos entre si y se considera que se obtienen en contextos que favorecen a estas mediciones, si fueran valores mucho más extraños o que no siguieran una cierta pauta si que se procedería a marcarlos como NA's y luego aplicar el método missForest() para rellenar esos valores, además la proporción de estos valores atípicos en comparación con el número total de registros es muy pequeña por tanto no deberían tener un impacto significativo en el análisis, en caso de estar cometiendo un error al permitirlos.

2.4 Análisis de los datos

En este punto sería necesario escoger las variables con las cuales se pretende realizar el futuro modelo pero en primer lugar se realizarán pruebas sobre la normalidad, la homogeneidad de la varianza y la correlación entre las diferentes variables que conforman el dataset para más delante poder descartar aquellas que no se consideren necesarias. En primer lugar, se procede a realizar un test sobre la normalidad del set de datos, para ello se parte de la siguiente hipótesis para el caso del test de *Shapiro*.

$$H_o\text{: p-value} > 0.05 = \text{Distribución normal}$$

$$H_1\text{: p-value} < 0.05 = \text{No hay Distribución normal}$$

A partir de esta hipótesis se procede a aplicar el test de Shapiro a las diferentes variables que conforman el dataset para comprobar su distribución.

```
# Se aplica test de Shapiro
shapiro.test(data_mf$ph)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$ph
## W = 0.98958, p-value = 8.953e-15
```

```
shapiro.test(data_mf$Hardness)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Hardness
## W = 0.99597, p-value = 9.61e-08
```

```
shapiro.test(data_mf$Solids)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Solids
## W = 0.97773, p-value < 2.2e-16
```

```
shapiro.test(data_mf$Chloramines)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Chloramines
## W = 0.99677, p-value = 1.818e-06
```

```
shapiro.test(data_mf$Sulfate)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Sulfate
## W = 0.98209, p-value < 2.2e-16
```

```
shapiro.test(data_mf$Conductivity)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Conductivity
## W = 0.99297, p-value = 1.494e-11
```

```
shapiro.test(data_mf$Organic_carbon)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Organic_carbon
## W = 0.99952, p-value = 0.6251
```

```
shapiro.test(data_mf$Trihalomethanes)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Trihalomethanes
## W = 0.99757, p-value = 4.829e-05
```

```
shapiro.test(data_mf$Turbidity)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Turbidity
## W = 0.9997, p-value = 0.9336
```

```
shapiro.test(data_mf$Potability)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data_mf$Potability
## W = 0.61882, p-value < 2.2e-16
```

Se puede observar como en algunas de las variables no se cumple la hipótesis nula, por tanto, siendo rigurosos con el test de Shapiro se podría afirmar que no existe distribución normal en ciertas variables del dataset. Por otro, lado, teniendo en cuenta el teorema central del limite y siendo el dataset de 3276 registros, lo cual se considera lo suficientemente grande, se podría afirmar que si se sigue una distribución normal. De todas formas se ha de tener en cuenta que la variable 'Potability' es una variable binaria o incluso podría ser categórica que indica si el agua es potable o no y en este caso es lógico que no siga una distribución normal.

A continuación se procede a evaluar la homogeneidad de la varianza, se aplicará el test de *Fligner-Killieen* ya que se parte del test de Shapiro anterior en el cual se ha visto que no existe normalidad, se procurará ser rigurosos por ello se parte de la conclusión de dicho test.

```
fligner.test(Potability ~ Organic_carbon, data = data_mf)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Potability by Organic_carbon
## Fligner-Killeen:med chi-squared = NaN, df = 3275, p-value = NA
```

```
fligner.test(Potability ~ ph, data = data_mf)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Potability by ph
## Fligner-Killeen:med chi-squared = NaN, df = 3275, p-value = NA
```

```
fligner.test(ph ~ Organic_carbon, data = data_mf)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  ph by Organic_carbon
## Fligner-Killeen:med chi-squared = NaN, df = 3275, p-value = NA
```

Observando los resultados se advierte que no existe homogeneidad en la varianza ya que no se obtienen resultados, por tanto no se puede comprobar si se cumple la hipótesis nula de que la varianza es la misma para las dos muestras evaluadas. Por otro lado, nos encontramos ante un set de datos que no contiene una distribución normal, según el test de Shapiro y de la misma forma no es homogéneo en cuanto a varianza, por tanto si se desea calcular la correlación entre variables, se debe realizar por el método de *Spearman* y no Pearson.

Teniendo esto en cuenta para realizar los análisis estadísticos del set de datos, será necesario recurrir a funciones o test no paramétricos como *Wilcoxon* o *Mann-Whitney*.

```
# Se procede a calcular la matriz de correlación del set de datos
if (!require("corrplot")) install.packages("corrplot")
```

```
## Loading required package: corrplot
```

```
## Warning: package 'corrplot' was built under R version 4.1.2
```

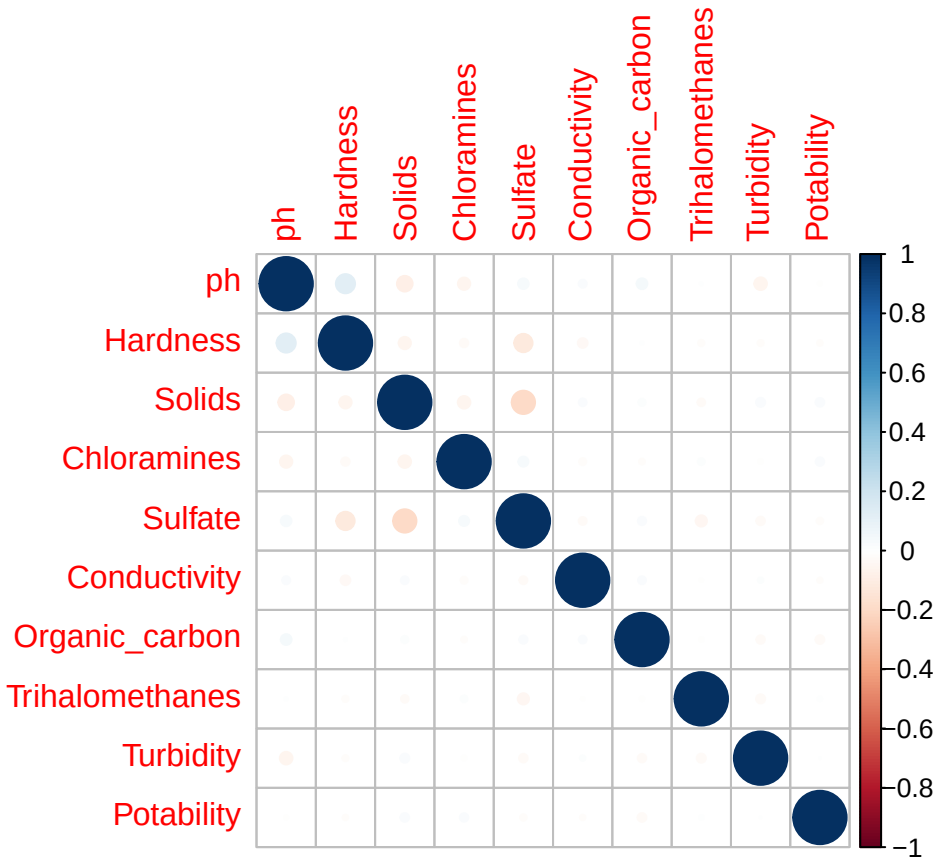
```
## corrplot 0.92 loaded
```

```
library(corrplot)
cor(data_mf, method = "spearman")
```

```
##                             ph     Hardness      Solids  Chloramines     Sulfate
## ph                1.000000000  0.128471615 -0.08565836 -0.052971913  0.03898714
## Hardness          0.128471615  1.000000000 -0.05258439 -0.024555937 -0.11731247
## Solids           -0.085658364 -0.052584392  1.00000000 -0.055037271 -0.19025914
## Chloramines      -0.052971913 -0.024555937 -0.05503727  1.000000000  0.03506650
## Sulfate           0.038987141 -0.117312465 -0.19025914  0.035066498  1.00000000
## Conductivity      0.022128674 -0.032883588  0.02108929 -0.016927957 -0.02159441
## Organic_carbon    0.040503547  0.003339523  0.01788116 -0.011910224  0.02361956
## Trihalomethanes   0.003310543 -0.013547497 -0.02028285  0.016172265 -0.04403958
## Turbidity        -0.055622062 -0.012855190  0.02847029 -0.007908762 -0.02536594
## Potability       -0.005976122 -0.010605614  0.02623379  0.024979804 -0.01478050
##                  Conductivity Organic_carbon Trihalomethanes    Turbidity
## ph                0.022128674    0.040503547     0.003310543 -0.055622062
## Hardness         -0.032883588    0.003339523    -0.013547497 -0.012855190
## Solids            0.021089292    0.017881164    -0.020282845  0.028470293
## Chloramines      -0.016927957   -0.011910224     0.016172265 -0.007908762
## Sulfate          -0.021594414    0.023619562    -0.044039579 -0.025365944
## Conductivity      1.000000000    0.021311009    -0.004346653  0.010341699
## Organic_carbon    0.021311009    1.000000000    -0.006746600 -0.024729911
## Trihalomethanes  -0.004346653   -0.006746600     1.000000000 -0.028413691
## Turbidity         0.010341699   -0.024729911    -0.028413691  1.000000000
## Potability       -0.010385918   -0.026773766     0.006842993  0.001087891
##                   Potability
## ph               -0.005976122
## Hardness         -0.010605614
## Solids            0.026233791
## Chloramines       0.024979804
## Sulfate          -0.014780495
## Conductivity     -0.010385918
## Organic_carbon   -0.026773766
## Trihalomethanes   0.006842993
```

```
## Turbidity        0.001087891
## Potability       1.000000000
```

```r
corrplot(cor(data_mf, method = "spearman"))
```



Se puede observar a partir del gráfico de correlaciones como no se advierten relaciones muy fuertes entre variables tanto positivas como negativas, no parece que haya una relación muy fuerte en cuanto al hecho de que al crecer una variable en valor, otra crezca de la misma manera o al revés, por otro lado, todas aportan información.

A continuación se realizará un análisis estadístico de los datos que consistirá en aplicar un contraste de hipótesis sobre dos muestras del set de datos, concretamente la variable potability y hardeness, se realizará dicho análisis con el objetivo de saber un poco más acerca de la incidencia de esta variable en la determinación de si el agua es potable o no, ya que a partir de la información del dataset no queda claro la influencia de la misma, en otras variables se establece un limite que se debe superar si se desea considerar el agua potable o no pero en esta variable, hardeness, no. Además tras el análisis de correlaciones no se ha observado ninguna variable con una correlación muy fuerte como para que sea de interés inmediato, por ello, se realizará este análisis sobre esta variable.

Se parte de la siguiente pregunta: ¿El hecho de que el agua sea potable convella un nivel superior de 'Hardeness' en el agua?

```r
# Se obtienen los diferentes casos para poder aplicar el contraste
Hardeness_Si_Potable <- data_mf[data_mf$Potability==1,]$Hardness
Hardeness_No_Potable <- data_mf[data_mf$Potability==0,]$Hardness
```

En caso de que la media sea igual significará que la variable hardness no determina según un valor mas grande

o mas pequeño si el agua es potable o no, si por el contraria resulta diferente nos aportará información de que si tiene cierta influencia. Por tanto, la hipótesis queda de la siguiente manera:

$$H_o\colon\ \mu_1 = \mu_2$$

$$H_1\colon\ \mu_1 \neq \mu_2$$

Se supondrá un valor de $\alpha = 0.05$ y se aplicará una función willcox.

```
wilcox.test(Hardeness_Si_Potable , Hardeness_No_Potable, alternative = "less")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Hardeness_Si_Potable and Hardeness_No_Potable
## W = 1260695, p-value = 0.272
## alternative hypothesis: true location shift is less than 0
```

Se advierte un resultado mayor a α por tanto se acepta la hipótesis nula y se considera que el nivel medio de hardness de agua potable es el mismo nivel que el agua no potable, por tanto, se puede concluir que no es una variable que marque la diferencia según su valor para determinar si el agua es potable o no, pero se incluirá en los análisis para corroborar efectivamente que no influye.

```
write.csv(data_mf, "water_potability_clean.csv", row.names = FALSE)
```

A continuación se realizará un modelo predictivo de regresión logística con el objetivo de que nos permita predecir, a partir de ciertos datos, si el agua de dicha zona analizada es potable o no.

2.4.1 Modelo Regresión Logística

En primer lugar almacenaremos las variables de interés del set de datos en diversas variables

```
# Guardamos variables
ph <- data_mf$ph
Hardness <- data_mf$Hardness
Solids <- data_mf$Solids
Chloramines <- data_mf$Chloramines
Sulfate <- data_mf$Sulfate
Conductivity <- data_mf$Conductivity
Organic_carbon <- data_mf$Organic_carbon
Trihalomethanes <- data_mf$Trihalomethanes
Turbidity <- data_mf$Turbidity
Potability <- data_mf$Potability

#Aplicamos regresión logistica

model_1 <- glm(Potability ~ ph + Solids + Chloramines + Sulfate + Conductivity + Organic_carbon + Trihal
model_2 <- glm(Potability ~ ph + Solids + Chloramines + Sulfate, data = data_mf ,family = "binomial")

summary(model_1)
```

```
##
## Call:
## glm(formula = Potability ~ ph + Solids + Chloramines + Sulfate +
##     Conductivity + Organic_carbon + Trihalomethanes + Turbidity,
##     family = "binomial", data = data_mf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1594  -1.0003  -0.9533   1.3542   1.5473
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.524e-01  5.593e-01  -0.451   0.6518
## ph              -1.658e-03  2.445e-02  -0.068   0.9460
## Solids           7.646e-06  4.183e-06   1.828   0.0676 .
## Chloramines      3.346e-02  2.276e-02   1.470   0.1414
## Sulfate         -9.295e-04  9.954e-04  -0.934   0.3504
## Conductivity    -1.962e-04  4.447e-04  -0.441   0.6591
## Organic_carbon  -1.813e-02  1.088e-02  -1.666   0.0957 .
## Trihalomethanes  9.350e-04  2.277e-03   0.411   0.6814
## Turbidity        1.715e-06  4.604e-02   0.000   1.0000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4382.0  on 3275  degrees of freedom
## Residual deviance: 4371.8  on 3267  degrees of freedom
## AIC: 4389.8
##
## Number of Fisher Scoring iterations: 4
```

```
summary(model_2)
```

```
##
## Call:
## glm(formula = Potability ~ ph + Solids + Chloramines + Sulfate,
##     family = "binomial", data = data_mf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1170  -0.9989  -0.9611   1.3577   1.5208
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.972e-01  4.390e-01  -1.133   0.2573
## ph          -3.509e-03  2.440e-02  -0.144   0.8856
## Solids       7.452e-06  4.179e-06   1.783   0.0746 .
## Chloramines  3.418e-02  2.275e-02   1.502   0.1330
## Sulfate     -9.981e-04  9.936e-04  -1.005   0.3151
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 4382  on 3275  degrees of freedom
## Residual deviance: 4375  on 3271  degrees of freedom
## AIC: 4385
##
## Number of Fisher Scoring iterations: 4
```

```
# Valor AIC obtenido
summary(model_1)$aic
```

```
## [1] 4389.761
```

```
summary(model_2)$aic
```

```
## [1] 4384.952
```

En modelos de regresión logística la medida que indica una buena bondad del modelo es el valor *AIC*, por tanto de entre los diferentes modelos se escoge aquel que dicho valor sea inferior, en este caso, el modelo 1. Por otra parte, no parecen ser resultados muy prometedores ya que se trata de un valor de AIC bastante elevado, por ello se procede a elaborar un modelo más complejo basado en un árbol de decisión.

2.4.2 Modelo árbol de decisión

Previamente a la aplicación del modelo de árbol de decisión se realizará la transformación de la variable 'Potability' para convertirla en una variable categórica ya que de esta manera se conseguirá un resultado gráficamente más fácil de entender.

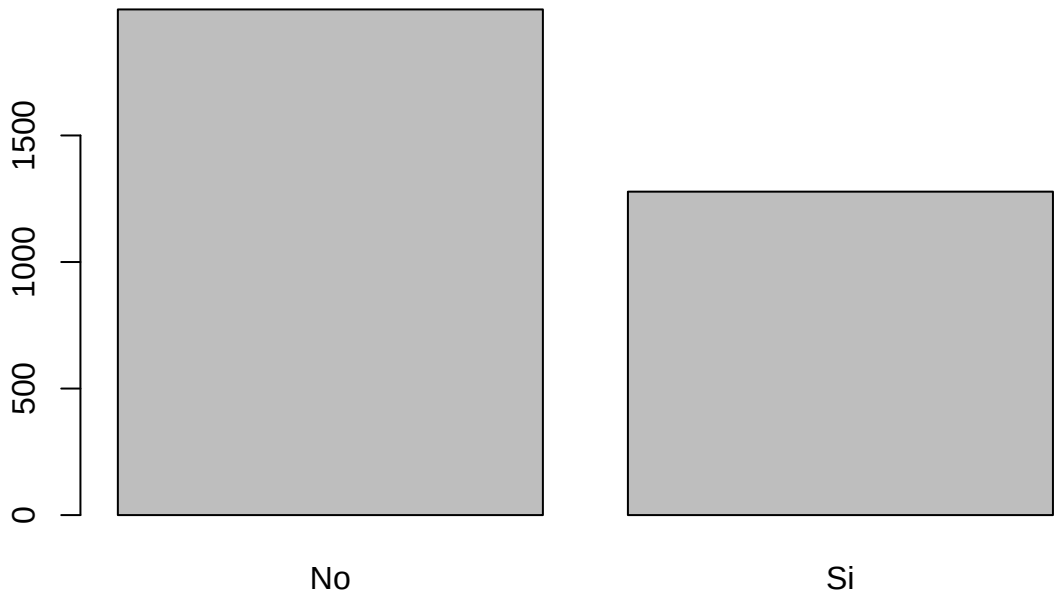
```
data_mf$Potability[data_mf$Potability == 1] <- "Si"
data_mf$Potability[data_mf$Potability == 0] <- "No"
data_mf$Potability<-as.factor(data_mf$Potability)
str(data_mf$Potability)
```

```
##  Factor w/ 2 levels "No","Si": 1 1 1 1 1 1 1 1 1 1 ...
```

```
levels(data_mf$Potability)
```

```
## [1] "No" "Si"
```

```
plot(data_mf$Potability)
```



Una vez realizado este cambio se continua con el proceso.

```
# Se genera dataset con los datos cambiados aleatoriamente para evitar overfitting
set.seed(1)
data_random <- data_mf[sample(nrow(data_mf)),]

# Se establecerá un corte de 2/3 para el training y 1/3 para el test
set.seed(2184)
summary(data_random[10])
```

```
##  Potability
##  No:1998
##  Si:1278
```

```
# Se escoge la variable de interés según su posición en el dataframe
str(data_random[10])
```

```
## 'data.frame':    3276 obs. of  1 variable:
##  $ Potability: Factor w/ 2 levels "No","Si": 1 2 1 1 2 1 2 2 2 1 ...
```

```
y <- data_random[,10]
x <- data_random[, -10]

split_prop <- 3
max_split <- floor(nrow(x)/split_prop)
```

```
indexes = sample(1:nrow(data_mf), size=floor(((split_prop-1)/split_prop)*nrow(data_mf)))

# Se generan los sets de training y test
trainX<-x[indexes,]
trainy<-y[indexes]
testX<-x[-indexes,]
testy<-y[-indexes]


# Se comprueba que se cumplen los cortes
dim(trainX)
```

```
## [1] 2184    9
```

```
dim(testX)
```

```
## [1] 1092    9
```

Una vez los diferentes sets de datos han sido creados, se crea el modelo de training que realizará la clasificación y aprenderá cuando clasificar el agua como potable(1) o no potable(0).

```
# Se crea modelo de training y se representan sus resultados
trainy = as.factor(trainy)
model <- C50::C5.0(trainX, trainy,rules=TRUE )
summary(model)
```

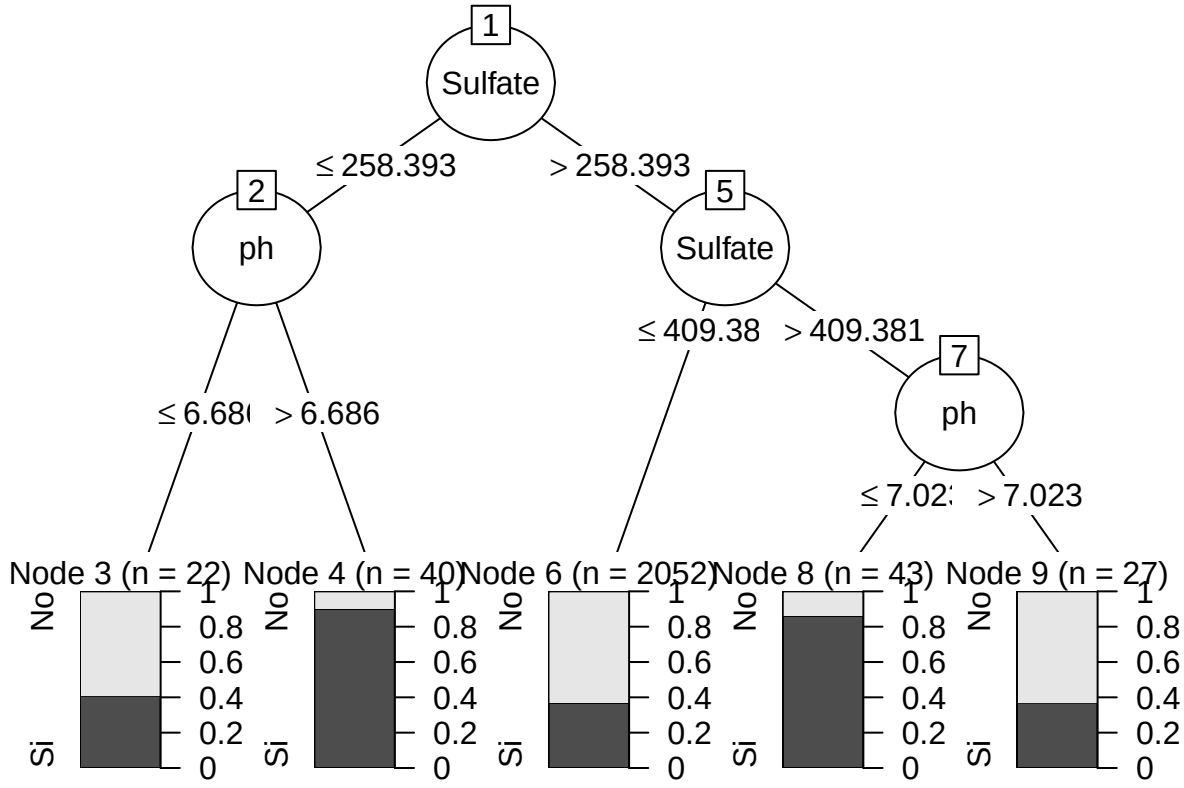
```
##
## Call:
## C5.0.default(x = trainX, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Tue Jan 04 20:58:49 2022
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 2184 cases (10 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (2122/803, lift 1.0)
##  Sulfate > 258.3927
##  ->  class No  [0.621]
##
## Rule 2: (43/6, lift 2.2)
##  ph <= 7.022952
##  Sulfate > 409.3809
##  ->  class Si  [0.844]
##
## Rule 3: (62/17, lift 1.9)
##  Sulfate <= 258.3927
##  ->  class Si  [0.719]
```

```
##
## Default class: No
##
##
## Evaluation on training data (2184 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##       3  789(36.1%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    1313    23    (a): class No
##     766    82    (b): class Si
##
##
##  Attribute usage:
##
##  100.00% Sulfate
##    1.97% ph
##
##
## Time: 0.0 secs
```

Observando los resultados se advierte un error en la precisión de 36.1%, lo cual quiere decir que se han predicho correctamente el 63.9% de los casos de entrenamiento, no es un valor pequeño de acurracy pero no es el más alentador, de todos modos habrá que observar los resultados de las predicciones con datos que el modelo no haya visto, es decir, habrá que evaluar los resultados del modelo de test.

En la siguiente imagen se muestra el modelo de árbol de decisión creado:

```
model <- C50::C5.0(trainX, trainy)
plot(model)
```



Se puede observar las variables que el algoritmo ha considerado más oportunas para realizar la clasificación, los diferentes valores que considera y a partir de los cuales realiza la clasificación idónea, por otro lado, se advierte que no existen variables fuertemente determinantes para realizar la clasificación del agua, ya que de las 9 variables que formaban parte del análisis, el algoritmo ha considerado oportunas únicamente 2, esto guardaría cierta relación con la imagen obtenida anteriormente de la correlación, ya se advertía entonces poco relación entre las variables y la variable de interés. A continuación se evaluará el modelo de test.

```
# Creamos el modelo predictivo del set de test y se obtiene su accuracy
predicted_model <- predict( model, testX, type="class")
print(sprintf("La precisión del árbol de decisión es: %.4f %%",
              100*sum(predicted_model==testy)/length(predicted_model)))
```

```
## [1] "La precisión del árbol de decisión es: 61.7216 %"
```

```
# Obtenemos confusion matrix
mat_conf<-table(testy,Predicted=predicted_model)
mat_conf
```

```
##      Predicted
## testy  No  Si
##    No 652  10
##    Si 408  22
```

En cuanto a los resultados obtenidos por le modelo con el set de test, la accuracy es muy similar al modelo del set de training, como se ha comentado anteriormente no es una precisión baja pero tampoco es idónea,

lo ideal sería tener una precisión por encima del 75%, es posible que se haya cometido algún error en la elaboración del modelo o este directamente relacionado con lo comentado en cuanto a la correlación entre variables.

2.5 Conclusión

A lo largo del proyecto se ha trabajo en un problema sobre potabilidad del agua, un tema muy interesante y con enfoques sostenibles, se han realizado diversos ajustes para conseguir el set de datos idóneo sobre el que trabajar, se ha aplicado la función missforest() para impitar de manera correcta aquellos valores nulos, se han realizado análisis para evaluar los valores outliers, se han llevado a cabo estudios estadísticos relacionados con la correlación, la normalidad, la homogeneidad de varianza, pse a realizar estas pruebas no ha sido posible establecer la variable más significativa para predecir la variable de interés que era el objetivo del proyecto.

Sobre el objetivo del proyecto, se puede afirmar que no se cumple, ya que como se ha visto anteriormente la accuracy obtenida en el modelo no es la idónea como para darlo por válido ya que esta es inferior a la precisión deseada, de todas formas el estudio se ha realizado correctamente pero los resultados no acompañan y se valora la posibilidad de que el dataset necesite de más variables, significativas, para poder determinar con mayor exactitud si el agua es potable o no, esto esta relacionado con lo comentado en apartados anteriores sobre la correlación y el valor tan pequeño que presentaban en algunas variables.