# Práctica 2: Tipología y ciclo de vida de los datos

Kyrylo Morozov

5 de enero, 2021

# Índice



# Introducción

En esta práctica nos centrarémos en el análisis íntegro de un dataset poniendo en práctica todas las técnicas aprendidas en el transcurso de la asignatura.

## 0.1. Descripción de Variables

Nuestra base de datos, dispone de una colección de propiedades físico-químicas de una gran variedad de vinos tintos con origen en el norte de Portugal. Por motivos de privacidad y protección de marcas no disponemos de datos sobre el tipo de uva, marca o brand del propio vino ni tampoco del precio. Así las variables que se nos proporcionan son las siguientes:

- fixed acidity: (g / dmˆ3) Concentración de ácidos no volatiles.
- volatile acidity: (g / dmˆ3) Ácido acético, responsable del sabor agrio, avinagrado.
- citric acid: (g / dmˆ3) Ácido cítrico, encontrado en cantidades pequeñas.
- residual sugar: (g / dmˆ3) Cantidad de azucar remanente despeés de la fermentación.
- chlorides: (g / dmˆ3) generalmente sal.

- free sulfur dioxide: (mg / dmˆ3) Dióxido de azufre, previene crecimiento de microbiomas y la oxidación del vino.
- total sulfur dioxide: (mg / dmˆ3) Total de dióxido de azufre contenido.
- density: (g / cmˆ3) Densidad del vino.
- pH: Escala pH, la mayoría de los vinos están entre 3-4.
- sulphates: (g / dmˆ3) Aditivo que contribuye a generar dioxido de azufre, gas de efecto invernadero.
- alcohol:( %Volumen). Concentración de alcohol en el vino.
- quality: Calidad del vino puntuada de 0 a 10.

**En total disponemos de 1599 observaciones**

### 0.2. Objetivos

Por el tipo de datos y ámbito del dataset, éste es adecuado para realizar técnicas de regresión o clasificación sobre él. Como objetivo principal podemos establecer la búsqueda de las propiedades físico-químicas que hacen que el vino tenga mayor o menor calidad. Así podemos agrupar nuestras variables de entrada (11) y la variable de salida, siendo ésta "quality". Además sería interesante disponer de herramientas, que mediante la entrada de las propiedades de un vino, puedan clasificarlo en bueno o malo.

Por oto lado podemos hablar de objetivos y referidos a las competecias correspondientes a los conocimientos pertenecientes al **Máster de Ciencia de datos**.

1. Analizar y establecer el problema que compete al dataset.
2. Seleccionar las variables adecuadas para la resolución del problema con su previa puesta a punto con procedimientos de limpieza, normalización y validación
3. Representar los resultados de forma adecuada y visualmente atractiva, además de hacer que ésta visualización sea intuitiva para interpretar los resultados de una forma correcta.
4. Discernir cuál de todas las técnicas disponibles para el anális de datos, es la mejor opción para cada caso en concreto.
5. Abordar el problema desde una perspectiva crítica y analítica, esto es, con visión innovadora y posiblemente disruptiva para con las técnicas ya existentes.

### 0.3. Licencia y reconomiento

El dataset con el que vamos a trabajar, es parte de un trabajo Realizado por P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Así pues, se debe reconocer la propiedad intelectual con la pertinente referencia al trabajo:

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009. Enlace Dataset

---

## 1. Limpieza del dataset

### 1.1. Carga del fichero y primer vistazo

Cargamos el fichero con el siguiente código:

```
archivo <- "winequality-red_raw.csv"
df <- read.csv(archivo)
summary(df)
```

```
##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
```

```
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
##        pH          sulphates         alcohol         quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
```

Forma de los datos y el nombre de las columnas.

```
dim(df)
```

```
## [1] 1599   12
```

```
colnames(df)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

Echamos un vistazo al resumen de la estructura de nuestra base de datos y todas las variables que intervienen en ella.

```
str(df)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

## 1.2. Valores Nulos NA's

```
which(is.na(df$fixed.acidity))
```

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

```
which(is.na(df$volatile.acidity))
```

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

```
which(is.na(df$citric.acid))
```

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

```
which(is.na(df$residual.sugar))
```

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

```
which(is.na(df$chlorides))
```

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

```
which(is.na(df$free.sulfur.dioxide))
```

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

```
which(is.na(df$total.sulfur.dioxide))
```

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

```
which(is.na(df$density))
```

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

```
which(is.na(df$pH))
```

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

```
which(is.na(df$sulphates))
```

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

```
which(is.na(df$alcohol))
```

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

```
which(is.na(df$quality))
```

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

Observamos como no existen valores nulos ni perdidos dentro de nuestro dataset por lo tanto no debemos preocuparnos por ellos.

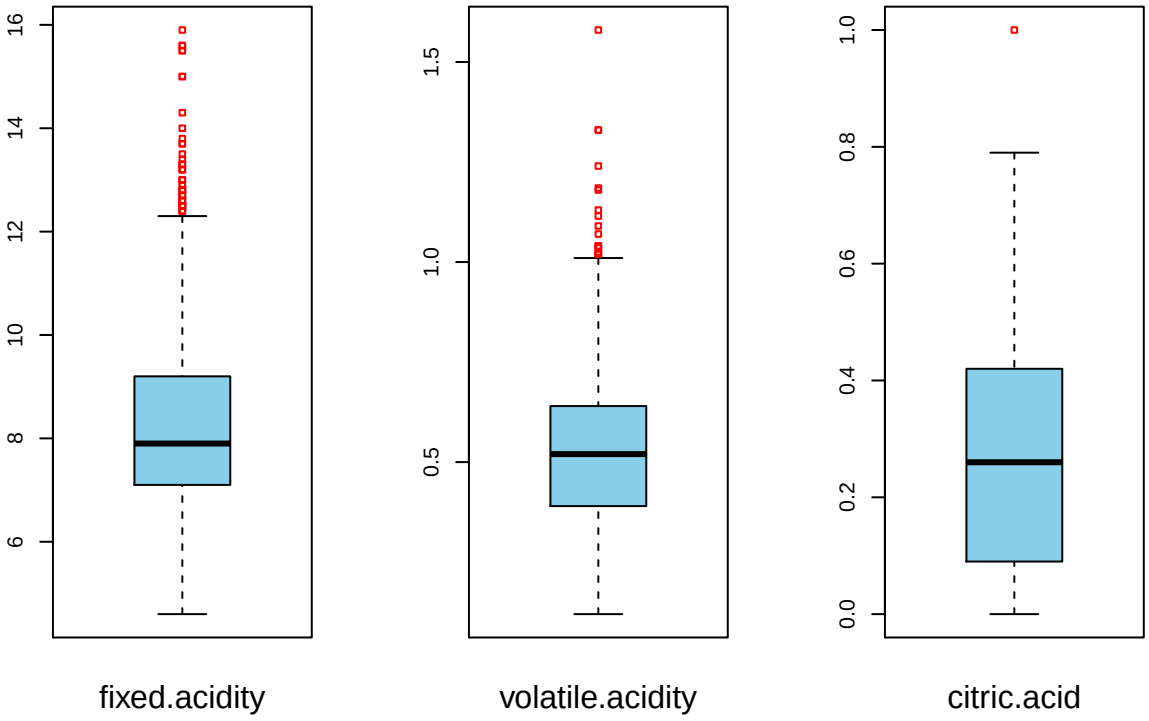
Creamos una nueva variable, por si puede sernos útiles en las predicciones, que divide los vinos en buenos, regulares y malos.

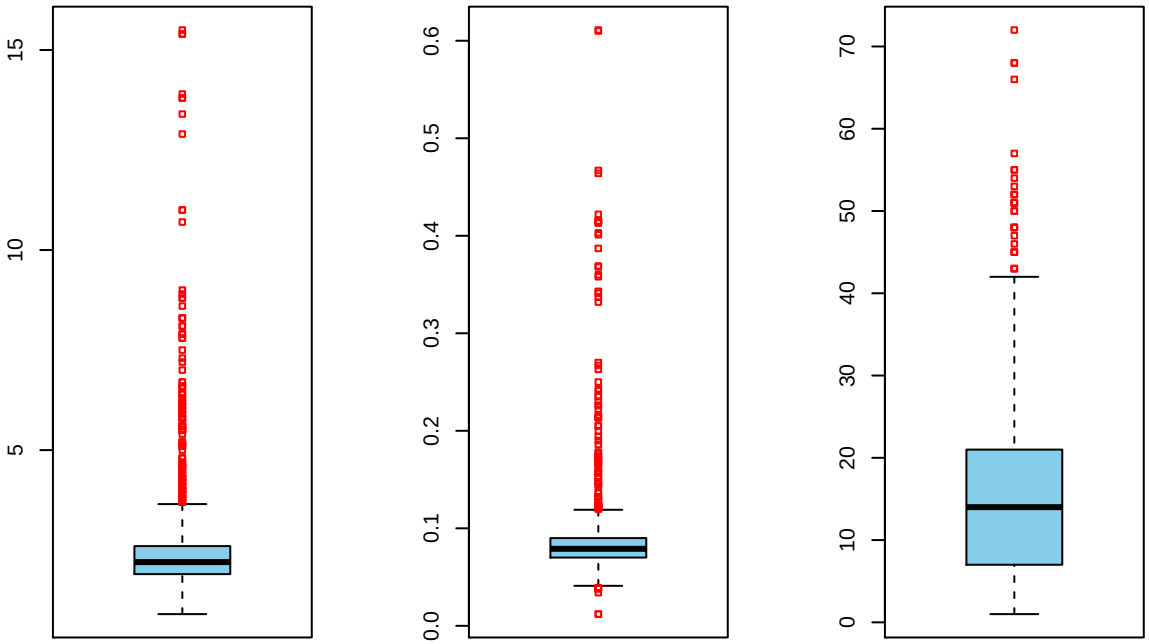
```
df$rating <- ifelse(df$quality < 5, 'bad', ifelse(
  df$quality < 7, 'average', 'good'))

df$rating <- ordered(df$rating,
                     levels = c('bad', 'average', 'good'))
```

## 1.3. Outliers

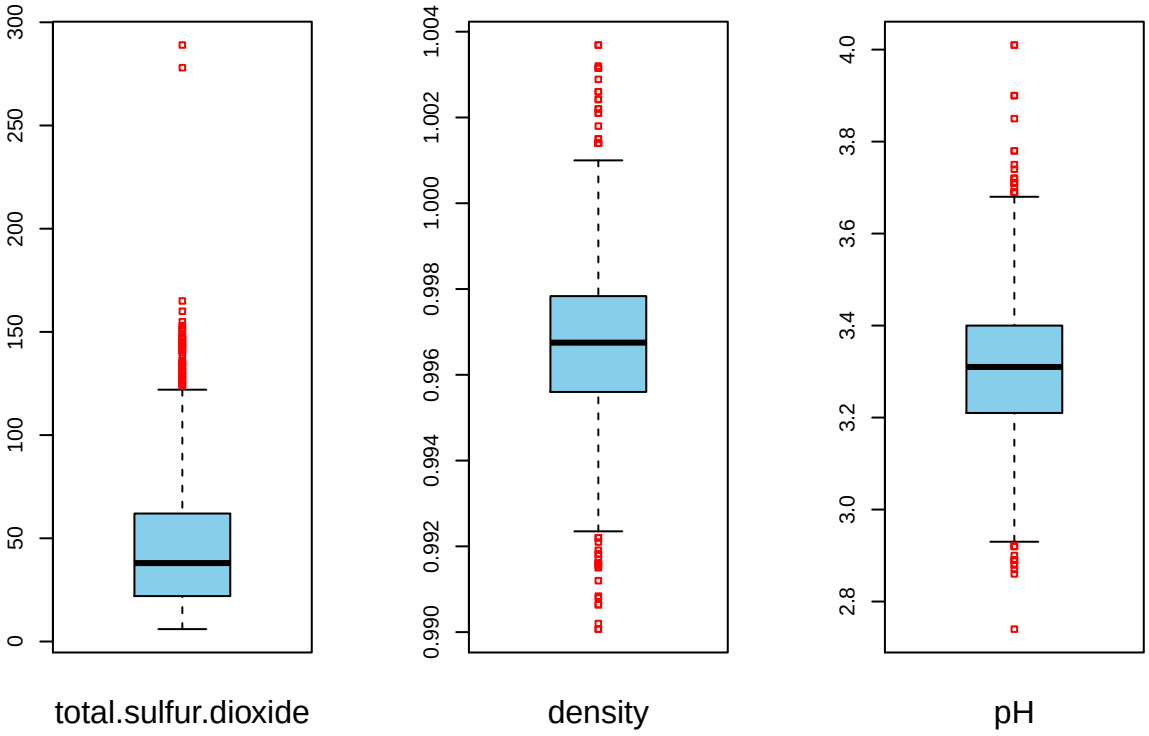
Para poder analizar mejor los outliers, vamos a hacer un boxplor para cada una de las variables disponibles en el dataset.

```
boxplots1 = par(mfrow = c(1,3))
for ( i in 1:11) {
  boxplot(df[[i]], col="skyblue",outcol='red',pch=22, method='jitter')
  mtext(names(df)[i], side = 1, line = 2)
}
```

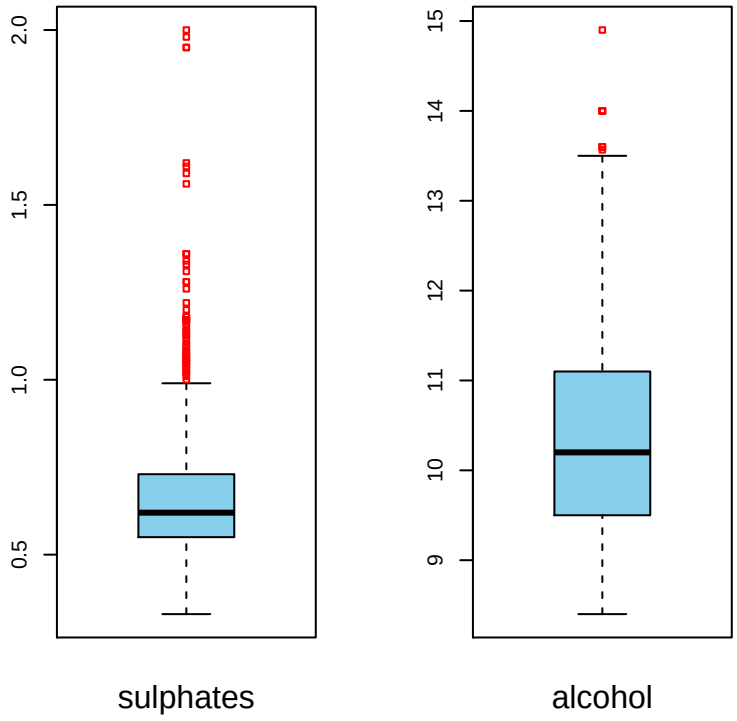

16
14
12
10
8
6
fixed.acidity
1.5
1.0
0.5
volatile.acidity
1.0
0.8
0.6
0.4
0.2
0.0
citric.acid




residual.sugar chlorides free.sulfur.dioxide




```
par(boxplots1)
```




Podemos comprobar los valores numéricos de dichos outliers con el siguiente código.

```
boxplot.stats(df$fixed.acidity)$out
```

```
##  [1] 12.8 12.8 15.0 15.0 12.5 13.3 13.4 12.4 12.5 13.8 13.5 12.6 12.5 12.8 12.8
## [16] 14.0 13.7 13.7 12.7 12.5 12.8 12.6 15.6 12.5 13.0 12.5 13.3 12.4 12.5 12.9
## [31] 14.3 12.4 15.5 15.5 15.6 13.0 12.7 13.0 12.7 12.4 12.7 13.2 13.2 13.2 15.9
## [46] 13.3 12.9 12.6 12.6
```

```
boxplot.stats(df$volatile.acidity)$out
```

```
##  [1] 1.130 1.020 1.070 1.330 1.330 1.040 1.090 1.040 1.240 1.185 1.020 1.035
## [13] 1.025 1.115 1.020 1.020 1.580 1.180 1.040
```

```
boxplot.stats(df$citric.acid)$out
```

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

```
boxplot.stats(df$residual.sugar)$out
```

```
##   [1]  6.10  6.10  3.80  3.90  4.40 10.70  5.50  5.90  5.90  3.80  5.10  4.65
##  [13]  4.65  5.50  5.50  5.50  5.50  7.30  7.20  3.80  5.60  4.00  4.00  4.00
##  [25]  4.00  7.00  4.00  4.00  6.40  5.60  5.60 11.00 11.00  4.50  4.80  5.80
##  [37]  5.80  3.80  4.40  6.20  4.20  7.90  7.90  3.70  4.50  6.70  6.60  3.70
##  [49]  5.20 15.50  4.10  8.30  6.55  6.55  4.60  6.10  4.30  5.80  5.15  6.30
##  [61]  4.20  4.20  4.60  4.20  4.60  4.30  4.30  7.90  4.60  5.10  5.60  5.60
##  [73]  6.00  8.60  7.50  4.40  4.25  6.00  3.90  4.20  4.00  4.00  4.00  6.60
##  [85]  6.00  6.00  3.80  9.00  4.60  8.80  8.80  5.00  3.80  4.10  5.90  4.10
##  [97]  6.20  8.90  4.00  3.90  4.00  8.10  8.10  6.40  6.40  8.30  8.30  4.70
## [109]  5.50  5.50  4.30  5.50  3.70  6.20  5.60  7.80  4.60  5.80  4.10 12.90
```

```
## [121]  4.30 13.40  4.80  6.30  4.50  4.50  4.30  4.30  3.90  3.80  5.40  3.80
## [133]  6.10  3.90  5.10  5.10  3.90 15.40 15.40  4.80  5.20  5.20  3.75 13.80
## [145] 13.80  5.70  4.30  4.10  4.10  4.40  3.70  6.70 13.90  5.10  7.80
```

```
boxplot.stats(df$chlorides)$out
```

```
##   [1] 0.176 0.170 0.368 0.341 0.172 0.332 0.464 0.401 0.467 0.122 0.178 0.146
##  [13] 0.236 0.610 0.360 0.270 0.039 0.337 0.263 0.611 0.358 0.343 0.186 0.213
##  [25] 0.214 0.121 0.122 0.122 0.128 0.120 0.159 0.124 0.122 0.122 0.174 0.121
##  [37] 0.127 0.413 0.152 0.152 0.125 0.122 0.200 0.171 0.226 0.226 0.250 0.148
##  [49] 0.122 0.124 0.124 0.143 0.222 0.039 0.157 0.422 0.034 0.387 0.415 0.157
##  [61] 0.157 0.243 0.241 0.190 0.132 0.126 0.038 0.165 0.145 0.147 0.012 0.012
##  [73] 0.039 0.194 0.132 0.161 0.120 0.120 0.123 0.123 0.414 0.216 0.171 0.178
##  [85] 0.369 0.166 0.166 0.136 0.132 0.132 0.123 0.123 0.123 0.403 0.137 0.414
##  [97] 0.166 0.168 0.415 0.153 0.415 0.267 0.123 0.214 0.214 0.169 0.205 0.205
## [109] 0.039 0.235 0.230 0.038
```

```
boxplot.stats(df$free.sulfur.dioxide)$out
```

```
##  [1] 52 51 50 68 68 43 47 54 46 45 53 52 51 45 57 50 45 48 43 48 72 43 51 51 52
## [26] 55 55 48 48 66
```

```
boxplot.stats(df$total.sulfur.dioxide)$out
```

```
##  [1] 145 148 136 125 140 136 133 153 134 141 129 128 129 128 143 144 127 126 145
## [20] 144 135 165 124 124 134 124 129 151 133 142 149 147 145 148 155 151 152 125
## [39] 127 139 143 144 130 278 289 135 160 141 141 133 147 147 131 131 131
```

```
boxplot.stats(df$density)$out
```

```
##  [1] 0.99160 0.99160 1.00140 1.00150 1.00150 1.00180 0.99120 1.00220 1.00220
## [10] 1.00140 1.00140 1.00140 1.00140 1.00320 1.00260 1.00140 1.00315 1.00315
## [19] 1.00315 1.00210 1.00210 0.99170 0.99220 1.00260 0.99210 0.99154 0.99064
## [28] 0.99064 1.00289 0.99162 0.99007 0.99007 0.99020 0.99220 0.99150 0.99157
## [37] 0.99080 0.99084 0.99191 1.00369 1.00369 1.00242 0.99182 1.00242 0.99182
```

```
boxplot.stats(df$pH)$out
```

```
##  [1] 3.90 3.75 3.85 2.74 3.69 3.69 2.88 2.86 3.74 2.92 2.92 2.92 3.72 2.87 2.89
## [16] 2.89 2.92 3.90 3.71 3.69 3.69 3.71 3.71 2.89 2.89 3.78 3.70 3.78 4.01 2.90
## [31] 4.01 3.71 2.88 3.72 3.72
```

```
boxplot.stats(df$sulphates)$out
```

```
##  [1] 1.56 1.28 1.08 1.20 1.12 1.28 1.14 1.95 1.22 1.95 1.98 1.31 2.00 1.08 1.59
## [16] 1.02 1.03 1.61 1.09 1.26 1.08 1.00 1.36 1.18 1.13 1.04 1.11 1.13 1.07 1.06
## [31] 1.06 1.05 1.06 1.04 1.05 1.02 1.14 1.02 1.36 1.36 1.05 1.17 1.62 1.06 1.18
## [46] 1.07 1.34 1.16 1.10 1.15 1.17 1.17 1.33 1.18 1.17 1.03 1.17 1.10 1.01
```

```
boxplot.stats(df$alcohol)$out
```

```
##  [1] 14.00000 14.00000 14.00000 14.00000 14.90000 14.00000 13.60000 13.60000
##  [9] 13.60000 14.00000 14.00000 13.56667 13.60000
```

```
boxplot.stats(df$quality)$out
```

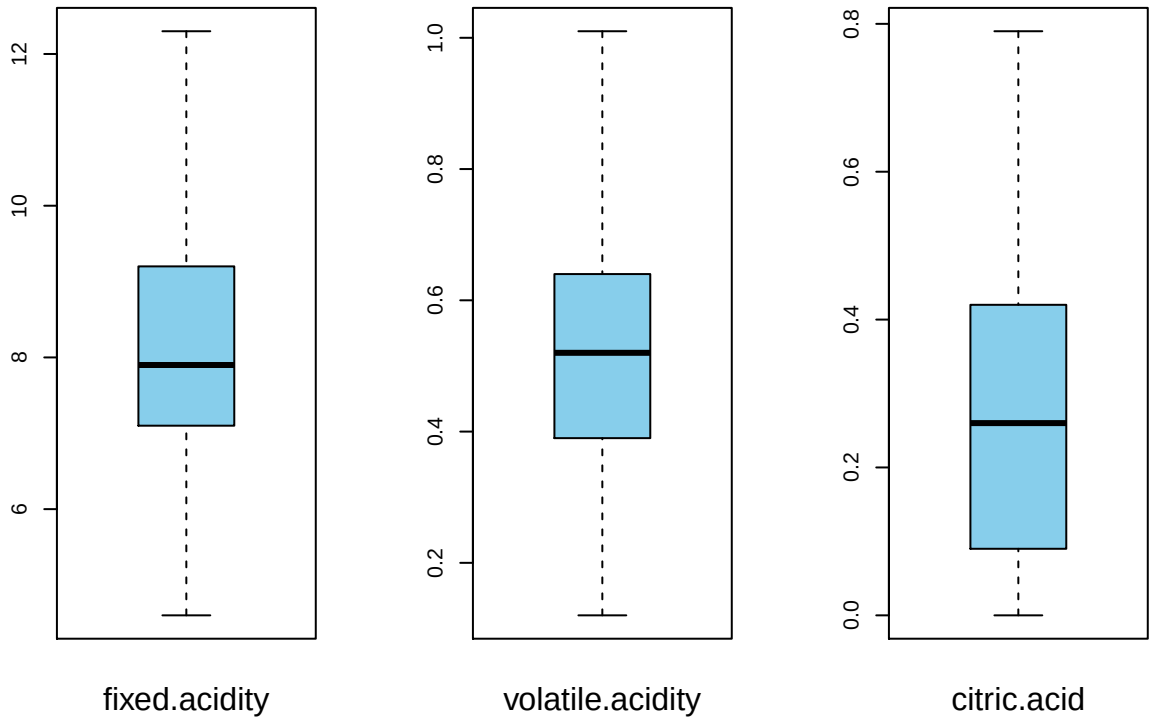
```
##  [1] 8 8 8 8 8 3 8 8 8 3 8 3 8 3 3 8 8 8 8 8 3 3 8 8 8 3 3 3 8
```

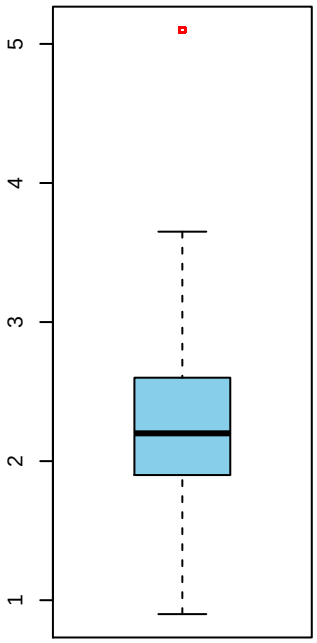
Podemos ver perfectamente como en todas las variables existen outliers. Para imputarlos, vamos a usar una estrategía bastante expandida, como la de imputación de outliers "leves", esto es, aquellos que se encuentran

1.5 pasos por encima y por debajo de los cuartiles Q1 y Q3.

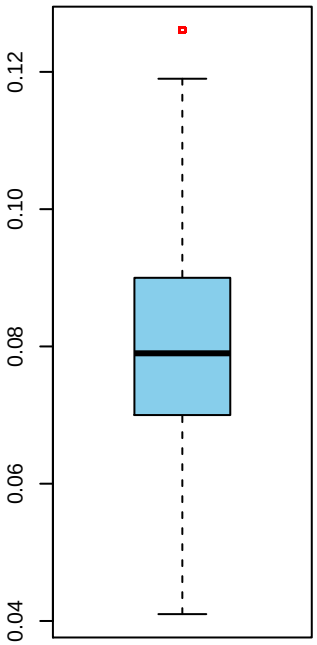
```
outnorm <- function(x){
   qntile <- quantile(x, probs=c(.25, .75))
   caps <- quantile(x, probs=c(.05, .95))
   H <- 1.5 * IQR(x, na.rm = T)
   x[x < (qntile[1] - H)] <- caps[1]
   x[x > (qntile[2] + H)] <- caps[2]
   return(x)
}

dfi <- df

for (i in 1:11){
  dfi[[i]]=outnorm(dfi[[i]])
}
```

```
boxplots2 = par(mfrow = c(1,3))
for ( i in 1:11) {
  boxplot(dfi[[i]], col="skyblue",outcol='red',pch=22, method='jitter')
  mtext(names(df)[i], side = 1, line = 2)
}
```

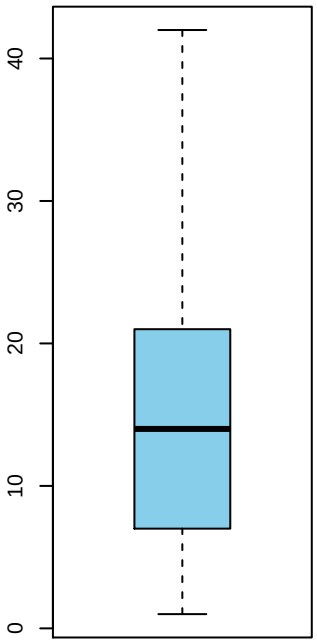




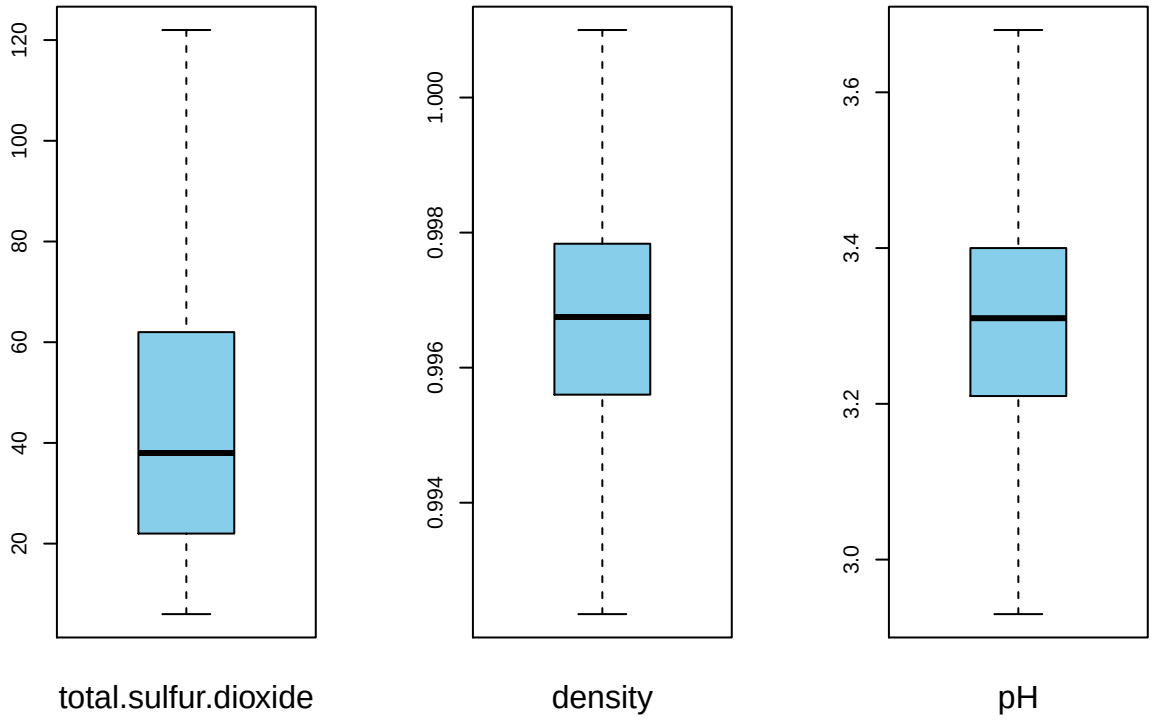

residual.sugar



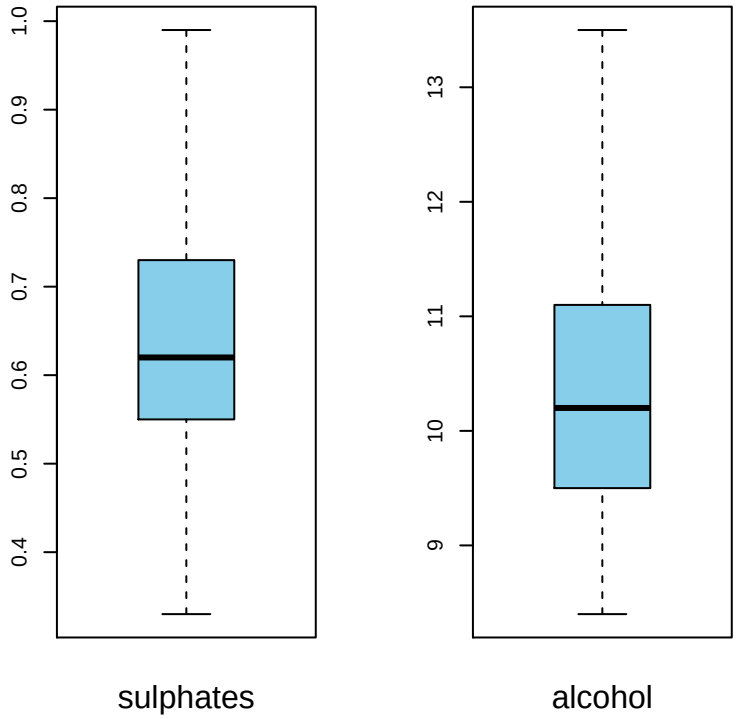

chlorides




free.sulfur.dioxide




```
par(boxplots2)
```




Podemos comprobar definitivamente que nos hemos deshecho de los outliers. Aunque esto es una técnica que consigue que nuestros datos sean más manejables de cara a hacer predicciones, puede ocurrir, que este tipo de tratamientos de datos induzcan un error en las predicciones. Puesto que estamos modificando la realidad que representan los datos. Por lo tanto, la decisión final ha sido la de proseguir con los outliers originales, para no perder exactitud en los datos. Aunque también se guarda tanto la versión imputada como sin imputar para poder hacer en un futuro comparativa entre ámbos enfoques.

```
write.csv(dfi, file = "winequality-red_clean_imputedoutliers.csv")
write.csv(df, file = "winequality-red_clean.csv")
```

# 2. Análisis de datos

## 2.1. Test Normalidad

Para comprobar la normalidad de los datos podemos usar el test de Shapiro-Wilk. Si pvalue<0.05 la distribución no es normal, en caso contrario, si lo es.

```
shapiro.test(df$pH)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$pH
## W = 0.99349, p-value = 1.712e-06
```

```
shapiro.test(df$fixed.acidity)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  df$fixed.acidity
## W = 0.94203, p-value < 2.2e-16
```

```
shapiro.test(df$volatile.acidity)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$volatile.acidity
## W = 0.97434, p-value = 2.693e-16
```

```
shapiro.test(df$citric.acid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$citric.acid
## W = 0.95529, p-value < 2.2e-16
```

```
shapiro.test(df$residual.sugar)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$residual.sugar
## W = 0.56608, p-value < 2.2e-16
```

```
shapiro.test(df$chlorides)$out
```

```
## NULL
```

```
shapiro.test(df$free.sulfur.dioxide)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$free.sulfur.dioxide
## W = 0.90184, p-value < 2.2e-16
```

```
shapiro.test(df$total.sulfur.dioxide)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$total.sulfur.dioxide
## W = 0.87322, p-value < 2.2e-16
```

```
shapiro.test(df$density)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$density
## W = 0.99087, p-value = 1.936e-08
```

```
shapiro.test(df$sulphates)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  df$sulphates
## W = 0.83304, p-value < 2.2e-16
```

```
shapiro.test(df$alcohol)
```

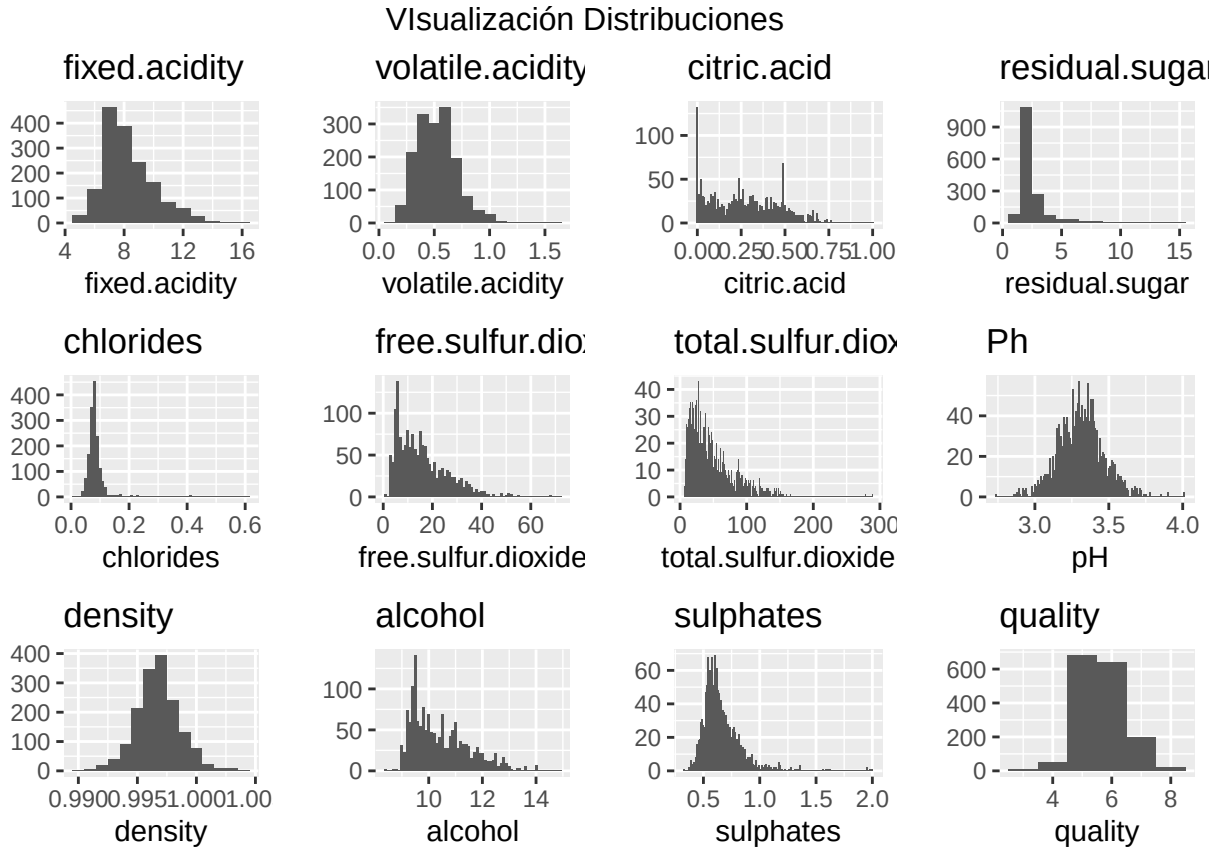
```
##
##  Shapiro-Wilk normality test
##
## data:  df$alcohol
## W = 0.92884, p-value < 2.2e-16
```

```
shapiro.test(df$quality)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$quality
## W = 0.85759, p-value < 2.2e-16
```

Se puede afirmar con total seguridad que ninguna de las variables sigue una distribución normal, estricatamente. Esto no implica en nigún momento que los datos no sean modelables, ni tampoco hay un imperativo en normalizar variables para poder realizar predicciones satisfactoriamente. De hecho para esta práctica se ha tomado la decisión de no normalizar las variables para no perder una visión numérica real de los factores físico-químicos de cara a una posible lectura del trabajo por parte de profesionales del sector.

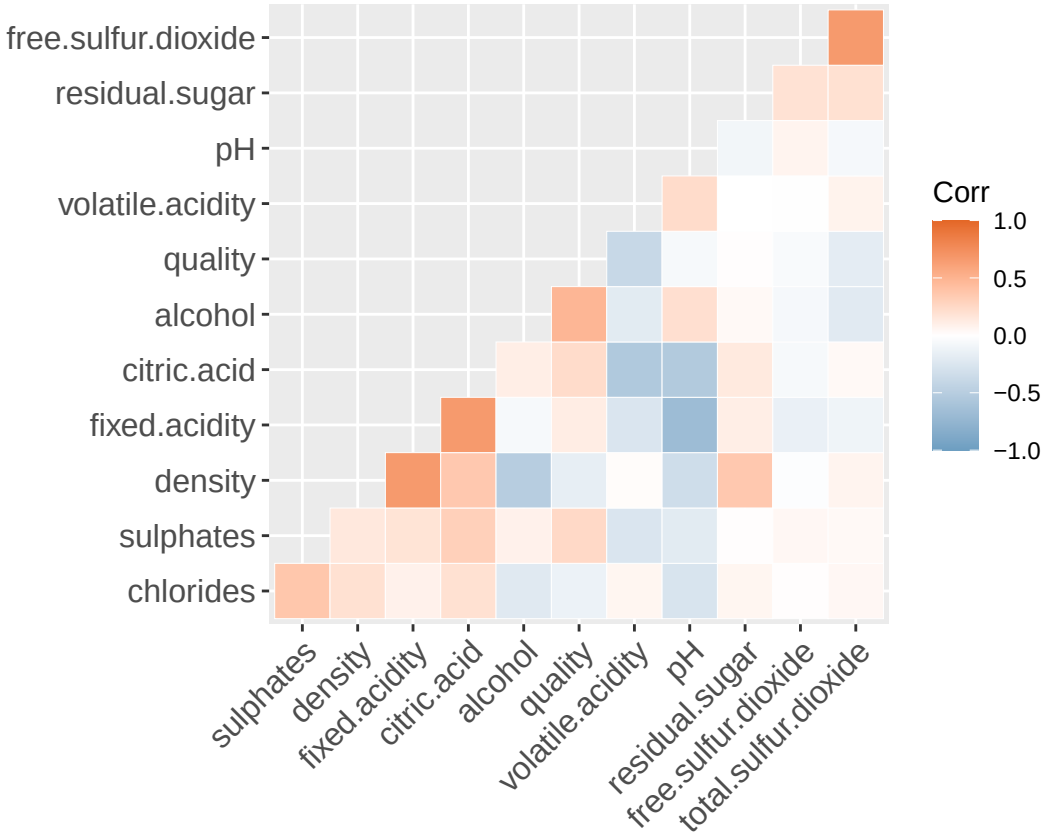
```
p1 <- qplot(fixed.acidity, data = df,binwidth=1)+ ggtitle("fixed.acidity")
p2 <- qplot(volatile.acidity, data = df,binwidth=0.1) + ggtitle("volatile.acidity")
p3 <- qplot(citric.acid, data = df,binwidth=0.01) + ggtitle("citric.acid")
p4 <- qplot(residual.sugar, data = df,binwidth=1) + ggtitle("residual.sugar")
p5 <- qplot(chlorides, data = df,binwidth=0.01) + ggtitle("chlorides")
p6 <- qplot(free.sulfur.dioxide, data = df,binwidth=1) + ggtitle("free.sulfur.dioxide")
p7 <- qplot(total.sulfur.dioxide, data = df,binwidth=1) + ggtitle("total.sulfur.dioxide")
p8 <- qplot(pH, data = df,binwidth=0.01)+ggtitle("Ph")
p9 <- qplot(density, data = df,binwidth=0.001)+ggtitle("density")
p10 <- qplot(alcohol, data = df,binwidth=0.1) + ggtitle("alcohol")
p11<- qplot(sulphates, data = df,binwidth=0.01) + ggtitle("sulphates")
p12 <- qplot(quality, data = df,binwidth=1) + ggtitle("quality")
grid.arrange(p1, p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12, nrow = 3,
top = "VIsualización Distribuciones")
```




Este último test de Shapiro es bastante sensible a pequeñas desviaciones de la normalidad, por lo tanto, aún sin ser las distribuiones normales y sin normalizar las variables, podemos asumir en algunas ocasiones analizando las distribuciones con la gráfica, que en agunos casos las variables se comportan de forma normal.

## 2.2. Correlaciones entre variables

```
dfcor <- cor(select_if(df, is.numeric))
ggcorrplot(dfcor,hc.order = TRUE,
  type = "lower",
  outline.color = "white",
  ggtheme = ggplot2::theme_gray,colors = c("#6D9EC1", "white", "#E46726"))
```




Podemos ver, en las correlaciones, que las variables que más influyen en la calidad, son el alcohol, ácidos volatiles, sulfatos, ácido cítrico, total dioxido azufre, densidad, sal,ácidos no volatiles, ph,SO2, azucar, en orden descendiente. Una vez encontradas las correlaciones más altas podemos comprobar si realmente las variables que menos correlación presentan, tienen poca significancia para la predicción de la calidad de vino. Lo harémos con T-test

## 2.3. Test estadístico

Vistas las correlaciones podemos aplicar un test estadístico para comprobar si existe por ejemplo una correlación alta y podemos decir que a más alcohol, mejor es la calidad del vino.

Con esto Hipótesis nula y alternativa mos queda de la siguiente forma.

$H_0 : \mu_1 = \mu_2$

$H_1 : \mu_1 < \mu_2$

```
t.test(df$alcohol,df$quality)
```

```
##
##  Welch Two Sample t-test
##
## data:  df$alcohol and df$quality
## t = 143.16, df = 2978.2, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  4.721397 4.852524
## sample estimates:
## mean of x mean of y
```

```
## 10.422983  5.636023
```

Como el pvalue<alpha(nivel de significancia) podemos rechazar la hipótesis nula y por consiguiente aformar que a mayor cantidad alcohol mejor será la calidad del vino.

Podemos realizar la misma prueba con otras dos variables con mayor correlación.

```
t.test(df$volatile.acidity,df$quality)
```

```
##
##  Welch Two Sample t-test
##
## data:  df$volatile.acidity and df$quality
## t = -246.94, df = 1754.7, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -5.148774 -5.067630
## sample estimates:
## mean of x mean of y
## 0.5278205 5.6360225
```

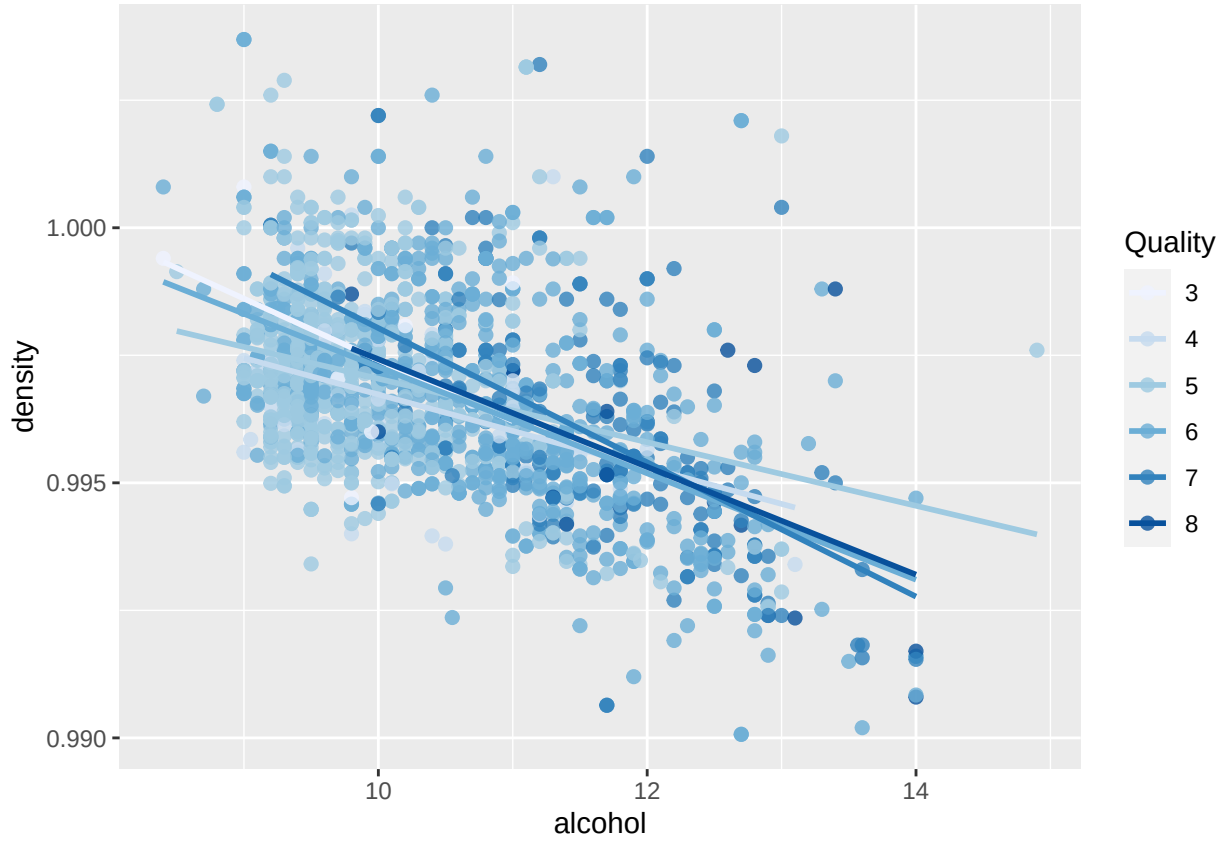
```
t.test(df$sulphates,df$quality)
```

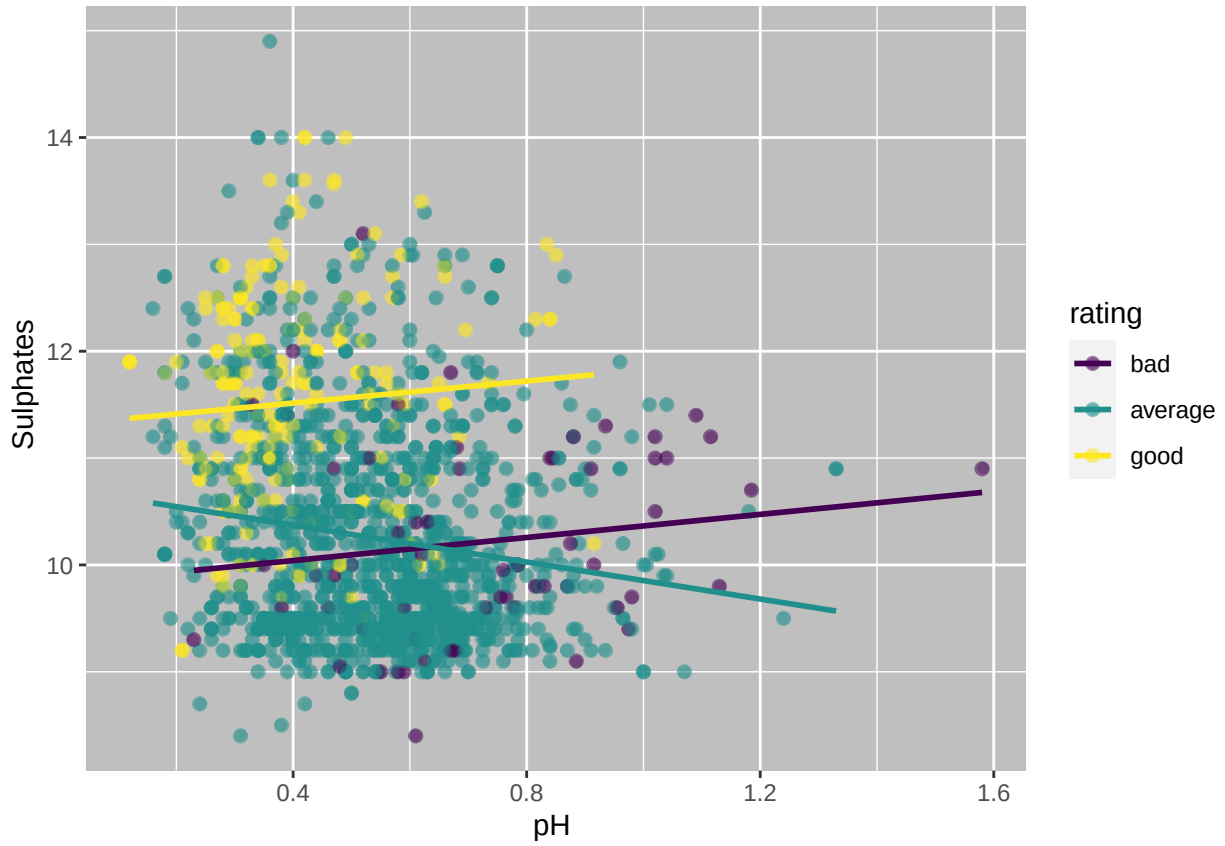
```
##
##  Welch Two Sample t-test
##
## data:  df$sulphates and df$quality
## t = -241.23, df = 1738.5, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -5.018347 -4.937400
## sample estimates:
## mean of x mean of y
## 0.6581488 5.6360225
```

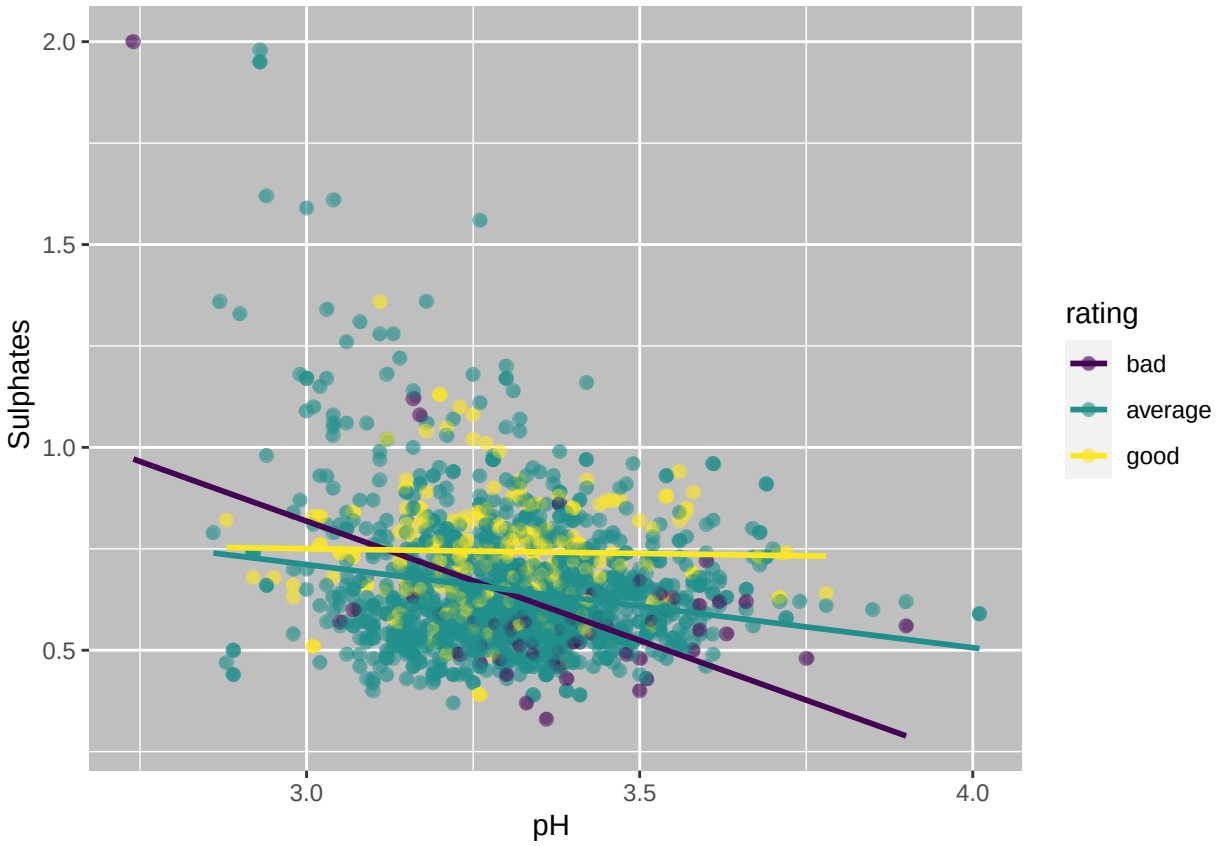
Llegando exactamente a las mismas conclusiones.

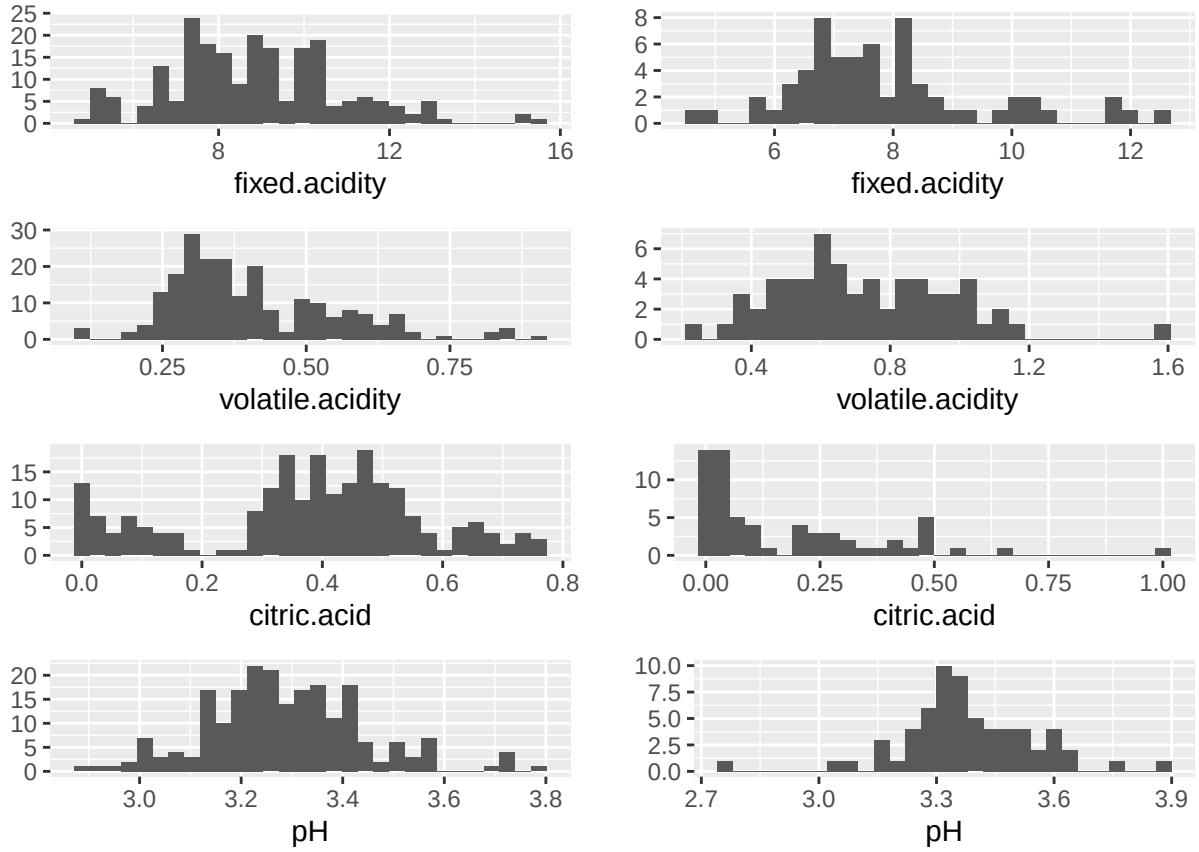
## 2.4. Análisis Gráficos

Otro método de análisis de los datos es la visualización de las relaciones que tienen las variables entre si.




Sulphates
14
12
10
0.4
0.8
1.2
1.6
pH
rating
bad
average
good


2.0
1.5
1.0
0.5
Sulphates
3.0
3.5
4.0
pH
rating
bad
average
good




Esta última gráfica permite ver las distribuciones de nuestras variables divididas en dos grupos para vinos buenos y vinos malos.

# 3. Modelos

## 3.1. Regresión lineal

El primer modelo propuesto es una regresión lineal. Empezaremos con una única variables predictora e irémos introduciendo de una en una para los siguientes modelos en el orden de la correlación creciente.

```
options(width = 60)
set.seed(1221)
training_data <- sample_frac(df, .6)
test_data <- df[ !df$X %in% training_data$X, ]
m1 <- lm(as.numeric(quality) ~ alcohol, data = training_data)
m2 <- update(m1, ~ . + volatile.acidity)
m3 <- update(m2, ~ . + sulphates)
m4 <- update(m3, ~ . + citric.acid)
m5 <- update(m4, ~ . + total.sulfur.dioxide)
m6 <- update(m5, ~ . + density)
m7 <- update(m6, ~ . + chlorides)
m8 <- update(m7, ~ . + fixed.acidity)
m9 <- update(m8, ~ . + pH)
m10 <- update(m9, ~ . + free.sulfur.dioxide)
m11 <- update(m10, ~ . + residual.sugar)
mtable(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11)
```

```
##
## Calls:
## m1: lm(formula = as.numeric(quality) ~ alcohol, data = training_data)
## m2: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity,
##     data = training_data)
## m3: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates, data = training_data)
## m4: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates + citric.acid, data = training_data)
## m5: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates + citric.acid + total.sulfur.dioxide, data = training_data)
## m6: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates + citric.acid + total.sulfur.dioxide + density,
##     data = training_data)
## m7: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates + citric.acid + total.sulfur.dioxide + density +
##     chlorides, data = training_data)
## m8: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates + citric.acid + total.sulfur.dioxide + density +
##     chlorides + fixed.acidity, data = training_data)
## m9: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates + citric.acid + total.sulfur.dioxide + density +
##     chlorides + fixed.acidity + pH, data = training_data)
## m10: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates + citric.acid + total.sulfur.dioxide + density +
##     chlorides + fixed.acidity + pH + free.sulfur.dioxide, data = training_data)
## m11: lm(formula = as.numeric(quality) ~ alcohol + volatile.acidity +
##     sulphates + citric.acid + total.sulfur.dioxide + density +
##     chlorides + fixed.acidity + pH + free.sulfur.dioxide + residual.sugar,
##     data = training_data)
##
## ================================================================================================
##                             m1          m2          m3          m4          m5          m6
## ------------------------------------------------------------------------------------------------
##   (Intercept)             2.104***    3.369***    2.966***    3.067***    3.412***   -6.852      -2.
##                          (0.225)     (0.233)     (0.250)     (0.258)     (0.265)    (15.204)    (15.
##   alcohol                 0.336***    0.291***    0.288***    0.288***    0.263***    0.273***    0.
##                          (0.021)     (0.020)     (0.020)     (0.020)     (0.021)     (0.025)     (0.
##   volatile.acidity                   -1.516***   -1.387***   -1.516***   -1.470***   -1.499***   -1.
##                                      (0.124)     (0.126)     (0.149)     (0.147)     (0.154)     (0.
##   sulphates                                       0.564***    0.607***    0.637***    0.625***    0.
##                                                  (0.134)     (0.136)     (0.135)     (0.136)     (0.
##   citric.acid                                                 -0.218      -0.180      -0.234      -0.
##                                                               (0.134)     (0.132)     (0.155)     (0.
##   total.sulfur.dioxide                                                    -0.003***   -0.003***   -0.
##                                                                           (0.001)     (0.001)     (0.
##   density                                                                             10.238       5.
##                                                                                      (15.163)    (15.
##   chlorides                                                                                       -1.
##                                                                                                   (0.
##   fixed.acidity
##
##   pH
##
```

```
##   free.sulfur.dioxide
##
##   residual.sugar
##
## ---------------------------------------------------------------------------------------------------
##   R-squared               0.203       0.311       0.324       0.326       0.341       0.341       0.3
##   N                     959         959         959         959         959         959         959
## ===================================================================================================
##   Significance: *** = p < 0.001; ** = p < 0.01; * = p < 0.05
```

Con esta información llegamos a la conclusión de que el mejor modelo es m11, m8,m9. Aunque también se puede observar que añadir más variables predictoras a costa del rendimiento no siempre mejora el nivel de acierto.

## 3.2. Regresión logística

Hemos preparado una variable nueva llamada rating, con anterioridad para poder hacer predicciones sobre variables categóricas y así poder clasificar el vino en bueno, regular y malo. Además podemos aádir una mejora a nuestro modelo, con la técnica de cross-validatión con el fin de mejorar el resultado.

```
set.seed(123)
train.control <- trainControl(method = "cv", number = 10)
model <- train(quality ~ alcohol + volatile.acidity +  sulphates + total.sulfur.dioxide,
               data = df, method = "glm", trControl = train.control)
summary(model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.72716  -0.38486  -0.06503   0.44980   2.13257
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.8258128  0.2006892  14.081  < 2e-16
## alcohol               0.2953105  0.0160331  18.419  < 2e-16
## volatile.acidity     -1.1985632  0.0966011 -12.407  < 2e-16
## sulphates             0.7121396  0.1005146   7.085 2.08e-12
## total.sulfur.dioxide -0.0022354  0.0005108  -4.376 1.28e-05
##
## (Intercept)          ***
## alcohol              ***
## volatile.acidity     ***
## sulphates            ***
## total.sulfur.dioxide ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.4290383)
##
##     Null deviance: 1042.17  on 1598  degrees of freedom
## Residual deviance:  683.89  on 1594  degrees of freedom
```

```
## AIC: 3191.7
##
## Number of Fisher Scoring iterations: 2
```

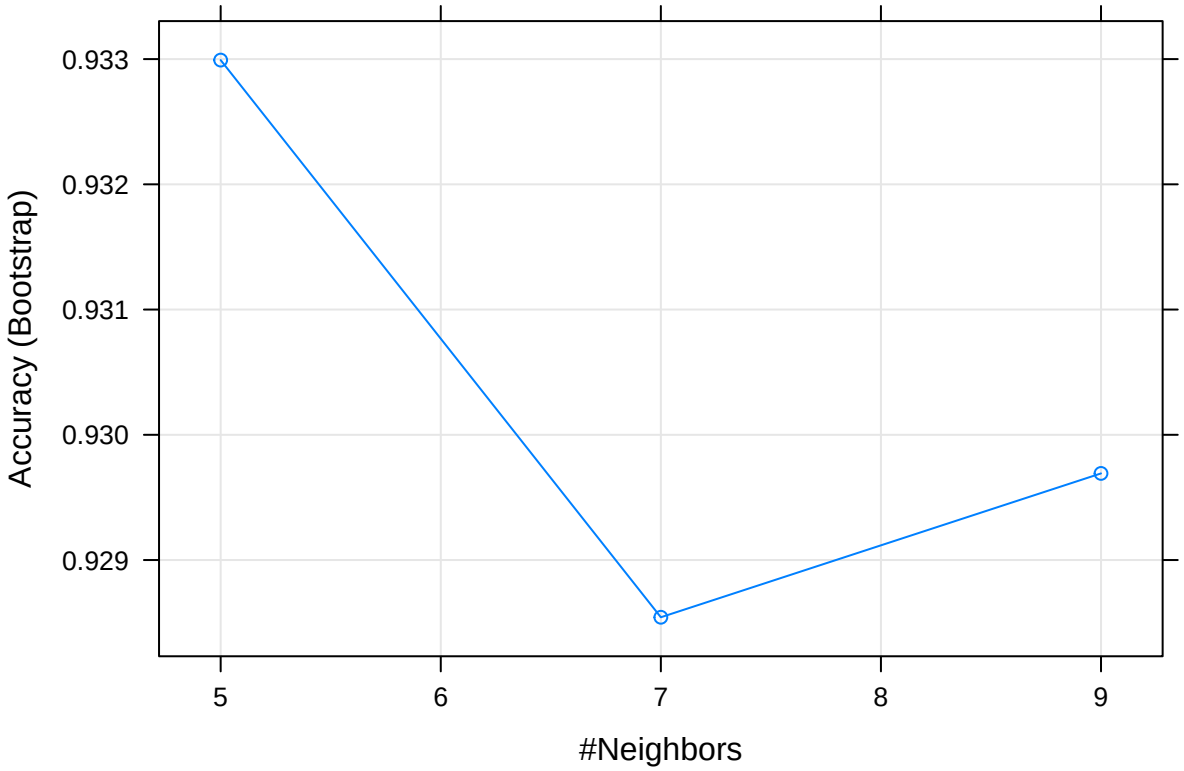
```
print(model)
```

```
## Generalized Linear Model
##
## 1599 samples
##    4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1439, 1439, 1438, 1439, 1439, 1440, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.6560876  0.3429191  0.5099489
```

Se puede ver que una regresión logística no mejora demasiado el nivel de predicción

### 3.3. KNN

```
## k-Nearest Neighbors
##
## 1121 samples
##   12 predictor
##    3 classes: 'bad', 'average', 'good'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1121, 1121, 1121, 1121, 1121, 1121, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9329919  0.7432479
##   7  0.9285431  0.7202512
##   9  0.9296917  0.7202933
##
## Accuracy was used to select the optimal model using
##  the largest value.
## The final value used for the model was k = 5.
```




```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction  bad average good
##    bad      1.5     0.1  0.0
##    average  2.3    81.7  3.1
##    good     0.0     1.2 10.1
##
##  Accuracy (average) : 0.9331
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad average good
##    bad       8       0    0
##    average  10     391   11
##    good      0       4   54
##
## Overall Statistics
##
##                Accuracy : 0.9477
##                  95% CI : (0.9238, 0.9659)
##     No Information Rate : 0.8264
##     P-Value [Acc > NIR] : 1.307e-15
##
```

```
##                  Kappa : 0.8067
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: bad Class: average Class: good
## Sensitivity             0.44444         0.9899      0.8308
## Specificity             1.00000         0.7470      0.9903
## Pos Pred Value          1.00000         0.9490      0.9310
## Neg Pred Value          0.97872         0.9394      0.9738
## Prevalence              0.03766         0.8264      0.1360
## Detection Rate          0.01674         0.8180      0.1130
## Detection Prevalence    0.01674         0.8619      0.1213
## Balanced Accuracy       0.72222         0.8684      0.9105
```

Podemos observar por la regla del codo automatizada en el algoritmo KNN como la mejor opción es k=7, es la que arroja los mejores resultados.

# 4. Conclusiones

En Definitiva, después de haber probado varios modelos de predicción y clasificación podemos afirmar que el nivel de bondad alcanzado en todos los modelos gira en torno a R2=0.34-0.35. Unas posibles mejoras y un trabajo futuro podría centrarse en hacer los modelos con las variables normalizadas. Además podemos intentar hacer modelos polinómicos o incluso logarítmicos. Esto compactaría de alguna forma la distribución y podría influir de una forma positiva en los resultados.

Hemos podido corroborar y extraer información de las correlaciones que tienen las variables y ello podría ser útil para optimizar la fabricación del vino en la industría.

Por último se puede afirmar que se han alcanzado los objetivos propuestos de modelado y clasificación de los datos.