

# Práctica 2 - Análisis y Limpieza de datos

Daniel Laureano Cerviño Cortínez

Diciembre 2020

## Contents

<b>1. Descripción de la Actividad</b>	<b>2</b>
1.1. Introducción . . . . .	2
1.2. Objetivos . . . . .	2
1.3. Competencias . . . . .	2
<b>2. Selección del juego de datos o dataset.</b>	<b>3</b>
<b>3. Importancia y objetivos del análisis</b>	<b>3</b>
<b>4. Análisis exploratorio del juego de datos.</b>	<b>4</b>
4.1 Carga de los datos . . . . .	4
4.2 Estadística descriptiva . . . . .	5
<b>5. Limpieza de datos</b>	<b>8</b>
<b>6. Análisis de cada uno de los atributos disponibles</b>	<b>9</b>
6.1. Análisis del comportamiento de las variables y detección de valores atípicos u outliers . . . . .	9
6.2. Correlación entre variables . . . . .	21
6.3. Discretización . . . . .	24
<b>7. Selección de los grupos de datos a analizar a través de PCA</b>	<b>25</b>
<b>8. Comprobación de normalidad y homogeneidad de la varianza</b>	<b>33</b>
8.1. Comprobación de normalidad . . . . .	33
8.2. Comprobación de la homocedasticidad de la varianza . . . . .	36

<b>9. Aplicación de pruebas estadísticas</b>	<b>38</b>
9.1. Correlación entre los distintos atributos ya reducidos. . . . .	38
9.2. Clasificación a través de algoritmo basado en árboles de decisión . . . . .	40
9.3. Clasificación a través de modelos de regresión . . . . .	48
9.3.1. Regresión lineal . . . . .	48
9.3.2. Regresión logística . . . . .	52
<b>Conclusiones</b>	<b>57</b>
<b>Bibliografía</b>	<b>58</b>

## 1. Descripción de la Actividad

### 1.1. Introducción

Esta práctica se centra en un caso práctica que consiste en el tratamiento de un dataset con el fin de conocer aquellos datos que son relevantes para su estudio, y emplear las distintas técnicas de integración, limpieza, validación y análisis de las mismas.

### 1.2. Objetivos

Los objetivos que se pretenden cumplir con la elaboración de esta práctica, son:

- Aprender a aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios o multidisciplinares.
- Saber identificar los datos relevantes y los tratamientos necesarios (integración, limpieza y validación) para llevar a cabo un proyecto analítico.
- Aprender a analizar los datos adecuadamente para abordar la información contenida en los datos.
- Identificar la mejor representación de los resultados para aportar conclusiones sobre el problema planteado en el proceso analítico.
- Actuar con los principios éticos y legales relacionados con la manipulación de datos en función del ámbito de aplicación.
- Desarrollar las habilidades de aprendizaje que les permitan continuar estudiando de un modo que tendrá que ser en gran medida autodirigido o autónomo.
- Desarrollar la capacidad de búsqueda, gestión y uso de información y recursos en el ámbito de la ciencia de datos.

### 1.3. Competencias

En esta práctica se desarrollan las siguientes competencias del Máster de Data Science:

- Capacidad de analizar un problema en el nivel de abstracción adecuado a cada situación y aplicar las habilidades y conocimientos adquiridos para abordarlo y resolverlo.
- Capacidad para aplicar las técnicas específicas de tratamiento de datos (integración, transformación, limpieza y validación) para su posterior análisis.

## 2. Selección del juego de datos o dataset.

Tras revisar distintos portales abiertos de datos como:

- kaggle
- UCI Machine Learning Repository
- DATA.GOV
- Datasets Wikipedia
- London Datastore

Me he decantado por utilizar el conjunto de datos proveniente del repositorio de datos sobre Machine Learning de UCI basado en la calidad de los vinos

Teniendo en cuenta los objetivos de un proyecto de análisis de datos, he seleccionado este dataset por las siguientes razones:

- Se trata de un juego de datos que puede ser emplear en términos reales para conocer cuáles son las propiedades características de cada uno de los tipos de vinos disponibles. Esta acción es muy importante para las empresas que elaboran vinos y buscan mejorar sus productos.

Teniendo en cuenta la elaboración de pruebas y métodos, he seleccionado este juego de datos por las siguientes razones:

- Todos los tipos de variables del dataset son de tipo real. Lo cual es muy importante a la hora de trabajar con algoritmos no supervisados.
- Se puede utilizar para labores de clasificación y regresión. Esto es debido a los atributos del dataset y a la existencia del atributo quality, que se trata de una variable cuyos valores muestran el grado de calidad del vino. Con esto podemos determinar que este juego de datos es válido para la práctica con algoritmos de aprendizaje supervisado y no supervisado.
- Este juego de datos se puede utilizar para la aplicación de algoritmos basados en reglas de asociación con el fin de conocer las características de los atributos que llegan a darse para cada grado de calidad del vino.
- El juego de datos se encuentra en una plataforma de alta fiabilidad.
- Existen suficientes observaciones (6497) y atributos (12+1) para la aplicación de todo tipo de algoritmos.

## 3. Importancia y objetivos del análisis

A partir del conjunto de datos comentado anteriormente, se pretende conocer las características de cada uno de los tipos de vinos. Así como, cuáles son aquellas características más relevantes para determinar cuando un vino es considerado rojo o blanco. Para ello, comprobaremos la capacidad que tiene el juego de datos para someterse a una reducción de dimensiones con el fin de generar un clasificador entre vinos rojos y blancos mediante la aplicación de una serie de algoritmos.

Como mencioné anteriormente, esta problemática es sumamente importante para aquellas empresas que se encargan de elaborar vinos con el fin de valorar y tener en cuenta aquellas propiedades o atributos más relevantes. Además, se podrían crear nuevos productos modificando las características de ellos que mejorasen la gama de productos de la empresa fabricante de vinos.

## 4. Análisis exploratorio del juego de datos.

El juego de datos está compuesto por 2 conjuntos, uno relacionado con vinos rojos y otro con vinos blancos.

Los atributos que tenemos disponibles para cada uno de ellos son los siguientes:

- **fixed.acidity.** Acidez fija.
- **volatile.acidity.** Acidez volátil.
- **citric.acid.** Ácido crítico.
- **residual.sugar.** Azúcar residual
- **chlorides.** Cloruros.
- **free.sulfur.dioxide.** Dióxido de azufre libre.
- **total.sulfur.dioxide.** Dióxido de azufre total.
- **density.** Densidad.
- **pH.** Potencial de Hidrógeno.
- **sulphates** Sulfatos.
- **alcohol** Alcohol.
- **quality** Grado de calidad.

Además, para la realización de este proyecto vamos a incluir el atributo Type con el fin de conocer si existen diferencias sustanciales entre vinos blancos y rojos.

- **type.** Donde el valor 2 se trata de vinos blancos y el valor 1 se trata de un vino rojo.

Cabe destacar que las clases de calidad de los vinos se encuentran ordenadas, aunque no se encuentran equilibradas.

### 4.1 Carga de los datos

En primer lugar, vamos a realizar la carga del juego de datos

```
data_red = read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv", encoding = "UTF-8", sep = ";")
data_white= read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv", encoding = "UTF-8", sep = ";")
```

Vamos a comprobar el número de vinos que tenemos de cada tipo

```
nrow(data_red)
```

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

```
nrow(data_white)
```

```
## [1] 4898
```

Como podemos comprobar, en nuestro juego de datos existen 1599 vinos rojos y 4898 vinos blancos.

A continuación, vamos a crear el atributo *type*, para cada uno de los vinos siendo su valor 2 para vinos blancos y su valor 1 para vinos rojos.

```
data_white$type = rep(2,nrow(data_white))
```

```
data_red$type = rep(1,nrow(data_red))
```

```
head(data_white$type)
```

```
## [1] 2 2 2 2 2 2
```

```
head(data_red$type)
```

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

Una vez ya creada esta variable, vamos a generar un conjunto de datos que integre ambos juegos de datos.

```
Dataset = rbind(data_white, data_red)
```

```
nrow(Dataset)
```

```
## [1] 6497
```

## 4.2 Estadística descriptiva

A continuación, vamos a conocer el comportamiento de los distintos atributos encontrados:

```
summary(Dataset)
```

```
## fixed.acidity  volatile.acidity  citric.acid  residual.sugar
## Min.   : 3.800  Min.   :0.0800  Min.   :0.0000  Min.   : 0.600
## 1st Qu.: 6.400  1st Qu.:0.2300  1st Qu.:0.2500  1st Qu.: 1.800
## Median : 7.000  Median :0.2900  Median :0.3100  Median : 3.000
## Mean   : 7.215  Mean   :0.3397  Mean   :0.3186  Mean   : 5.443
## 3rd Qu.: 7.700  3rd Qu.:0.4000  3rd Qu.:0.3900  3rd Qu.: 8.100
## Max.   :15.900  Max.   :1.5800  Max.   :1.6600  Max.   :65.800
## chlorides    free.sulfur.dioxide total.sulfur.dioxide  density
## Min.   :0.00900  Min.   : 1.00   Min.   : 6.0      Min.   :0.9871
## 1st Qu.:0.03800  1st Qu.:17.00   1st Qu.: 77.0     1st Qu.:0.9923
## Median :0.04700  Median :29.00   Median :118.0    Median :0.9949
## Mean   :0.05603  Mean   :30.53   Mean   :115.7    Mean   :0.9947
## 3rd Qu.:0.06500  3rd Qu.:41.00   3rd Qu.:156.0    3rd Qu.:0.9970
## Max.   :0.61100  Max.   :289.00  Max.   :440.0    Max.   :1.0390
## pH          sulphates        alcohol       quality
## Min.   :2.720  Min.   :0.2200  Min.   : 8.00   Min.   :3.000
```

```

## 1st Qu.:3.110   1st Qu.:0.4300   1st Qu.: 9.50   1st Qu.:5.000
## Median :3.210   Median :0.5100   Median :10.30   Median :6.000
## Mean   :3.219   Mean   :0.5313   Mean   :10.49   Mean   :5.818
## 3rd Qu.:3.320   3rd Qu.:0.6000   3rd Qu.:11.30   3rd Qu.:6.000
## Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :9.000
## type
## Min.   :1.000
## 1st Qu.:2.000
## Median :2.000
## Mean   :1.754
## 3rd Qu.:2.000
## Max.   :2.000

```

- El número de observaciones disponibles en el dataset es de 6497.
- La media de la variable *fixed.acidity* es de 7.215 unidades. El valor mínimo es de 3.8 unidades, mientras que el valor máximo es de 15.900 unidades. La mitad de los valores de la variable *fixed.acidity* son inferiores o iguales a 7.000 y la otra mitad superiores. El 25% de los valores del atributo *fixed.acidity* son menores o iguales a 6.400 unidades. El 75% de los valores de la variable *fixed.acidity* son menores o iguales a 7.700 unidades.
- La media de la variable *volatile.acidity* es de 0.3397 unidades. El valor mínimo es de 0.0800 unidades, mientras que el valor máximo es de 1.5800 unidades. La mitad de los valores de la variable *volatile.acidity* son inferiores o iguales a 0.2900 y la otra mitad superiores. El 25% de los valores del atributo *volatile.acidity* son menores o iguales a 0.2300 unidades. El 75% de los valores de la variable *volatile.acidity* son menores o iguales a 0.4000 unidades.
- La media de la variable *citric.acid* es de 0.3186 unidades. El valor mínimo es de 0.0000 unidades, mientras que el valor máximo es de 1.66 unidades. La mitad de los valores de la variable *citric.acid* son inferiores o iguales a 0.3100 y la otra mitad superiores. El 25% de los valores del atributo *citric.acid* son menores o iguales a 0.2500 unidades. El 75% de los valores de la variable *citric.acid* son menores o iguales a 0.3900 unidades.
- La media de la variable *residual.sugar* es de 5.443 unidades. El valor mínimo es de 0.600 unidades, mientras que el valor máximo es de 65.800 unidades. La mitad de los valores de la variable *residual.sugar* son inferiores o iguales a 3.000 y la otra mitad superiores. El 25% de los valores del atributo *residual.sugar* son menores o iguales a 1.800 unidades. El 75% de los valores de la variable *residual.sugar* son menores o iguales a 8.100 unidades.
- La media de la variable *chlorides* es de 0.05603 unidades. El valor mínimo es de 0.00900 unidades, mientras que el valor máximo es de 0.61100 unidades. La mitad de los valores de la variable *chlorides* son inferiores o iguales a 0.04700 y la otra mitad superiores. El 25% de los valores del atributo *chlorides* son menores o iguales a 0.03800 unidades. El 75% de los valores de la variable *chlorides* son menores o iguales a 0.06500 unidades.
- La media de la variable *free.sulfur.oxide* es de 30.53 unidades. El valor mínimo es de 1.00 unidades, mientras que el valor máximo es de 289.00 unidades. La mitad de los valores de la variable *free.sulfur.oxide* son inferiores o iguales a 29.00 y la otra mitad superiores. El 25% de los valores del atributo *free.sulfur.oxide* son menores o iguales a 17.00 unidades. El 75% de los valores de la variable *free.sulfur.oxide* son menores o iguales a 41.00 unidades.
- La media de la variable *total.sulfur.oxide* es de 115.7 unidades. El valor mínimo es de 6.0 unidades, mientras que el valor máximo es de 77.0 unidades. La mitad de los valores de la variable *total.sulfur.oxide* son inferiores o iguales a 118.0 y la otra mitad superiores. El 25% de los valores del atributo *total.sulfur.oxide* son menores o iguales a 77.0 unidades. El 75% de los valores de la variable *total.sulfur.oxide* son menores o iguales a 440.0 unidades.

- La media de la variable *density* es de 0.9947 unidades. El valor mínimo es de 0.9871 unidades, mientras que el valor máximo es de 1.0390 unidades. La mitad de los valores de la variable *density* son inferiores o iguales a 0.9949 y la otra mitad superiores. El 25% de los valores del atributo *density* son menores o iguales a 0.9923 unidades. El 75% de los valores de la variable *density* son menores o iguales a 0.9970 unidades.
- La media de la variable *pH* es de 3.219 unidades. El valor mínimo es de 2.720 unidades, mientras que el valor máximo es de 4.010 unidades. La mitad de los valores de la variable *pH* son inferiores o iguales a 3.210 y la otra mitad superiores. El 25% de los valores del atributo *pH* son menores o iguales a 3.110 unidades. El 75% de los valores de la variable *pH* son menores o iguales a 3.320 unidades.
- La media de la variable *sulphates* es de 0.5313 unidades. El valor mínimo es de 0.2200 unidades, mientras que el valor máximo es de 2.0000 unidades. La mitad de los valores de la variable *sulphates* son inferiores o iguales a 0.5100 y la otra mitad superiores. El 25% de los valores del atributo *sulphates* son menores o iguales a 0.4300 unidades. El 75% de los valores de la variable *sulphates* son menores o iguales a 0.6000 unidades.
- La media de la variable *alcohol* es de 10.49 unidades. El valor mínimo es de 8.00 unidades, mientras que el valor máximo es de 14.90 unidades. La mitad de los valores de la variable *alcohol* son inferiores o iguales a 10.30 y la otra mitad superiores. El 25% de los valores del atributo *alcohol* son menores o iguales a 9.50 unidades. El 75% de los valores de la variable *alcohol* son menores o iguales a 11.30 unidades.
- La media de la variable *quality* es de 5.818 unidades. El valor mínimo es de 3.000 unidades, mientras que el valor máximo es de 9.000 unidades. La mitad de los valores de la variable *quality* son inferiores o iguales a 6.000 y la otra mitad superiores. El 25% de los valores del atributo *quality* son menores o iguales a 5.000 unidades. El 75% de los valores de la variable *quality* son menores o iguales a 6.000 unidades.
- Como podemos comprobar, en nuestro juego de datos existen 1599 vinos rojos y 4898 vinos blancos. Por lo que *type=2* se encontrará 4898 veces y *type=1* se encontrará 1599 veces.

```
sapply(Dataset, function(x) class(x))
```

```
##      fixed.acidity    volatile.acidity    citric.acid
##      "numeric"        "numeric"          "numeric"
##      residual.sugar   chlorides         free.sulfur.dioxide
##      "numeric"        "numeric"          "numeric"
##      total.sulfur.dioxide density          pH
##      "numeric"        "numeric"          "numeric"
##      sulphates        alcohol           quality
##      "numeric"        "numeric"          "integer"
##      type
##      "numeric"
```

Podemos observar que el tipo de cada una de las variables que tenemos disponibles en el dataset es de tipo numeric o integer.

Observación de las primeras observaciones del dataset

```
head(Dataset)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1      7.0          0.27       0.36        20.7     0.045
## 2      6.3          0.30       0.34        1.6      0.049
```

```

## 3      8.1      0.28      0.40      6.9      0.050
## 4      7.2      0.23      0.32      8.5      0.058
## 5      7.2      0.23      0.32      8.5      0.058
## 6      8.1      0.28      0.40      6.9      0.050
##   free.sulfur.dioxide total.sulfur.dioxide density pH sulphates alcohol
## 1          45           170 1.0010 3.00      0.45      8.8
## 2          14           132 0.9940 3.30      0.49      9.5
## 3          30            97 0.9951 3.26      0.44     10.1
## 4          47           186 0.9956 3.19      0.40      9.9
## 5          47           186 0.9956 3.19      0.40      9.9
## 6          30            97 0.9951 3.26      0.44     10.1
##   quality type
## 1      6      2
## 2      6      2
## 3      6      2
## 4      6      2
## 5      6      2
## 6      6      2

```

## 5. Limpieza de datos

En primer lugar, vamos comprobar si existen campos con valores con NA

```
colSums(is.na(Dataset))
```

```

##   fixed.acidity    volatile.acidity    citric.acid
##             0                  0                  0
##   residual.sugar    chlorides    free.sulfur.dioxide
##             0                  0                  0
##   total.sulfur.dioxide    density        pH
##             0                  0                  0
##   sulphates    alcohol        quality
##             0                  0                  0
##   type
##             0

```

En segundo lugar, vamos a comprobar si existen atributos con valores vacíos.

```
colSums(Dataset=="")
```

```

##   fixed.acidity    volatile.acidity    citric.acid
##             0                  0                  0
##   residual.sugar    chlorides    free.sulfur.dioxide
##             0                  0                  0
##   total.sulfur.dioxide    density        pH
##             0                  0                  0
##   sulphates    alcohol        quality
##             0                  0                  0
##   type
##             0

```

Gracias a la información mostrada, podemos determinar que no existen valores vacíos ni nulos dentro de los atributos que componen nuestro juego de datos.

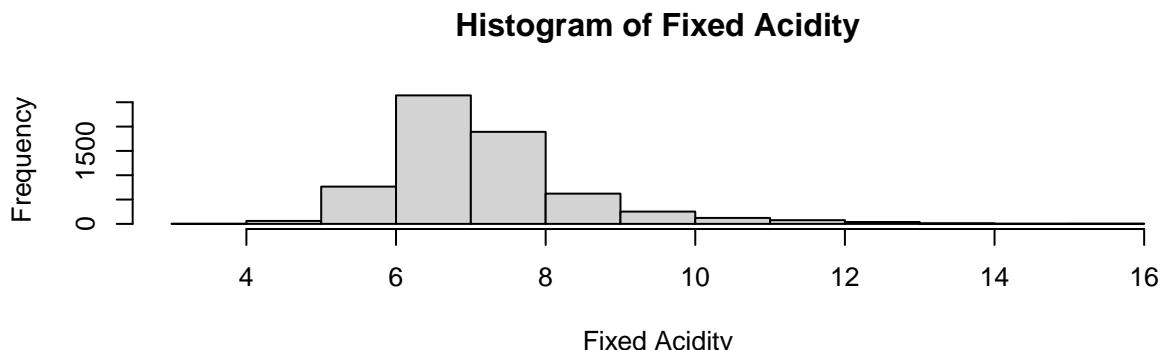
## 6. Análisis de cada uno de los atributos disponibles

### 6.1. Análisis del comportamiento de las variables y detección de valores atípicos u outliers

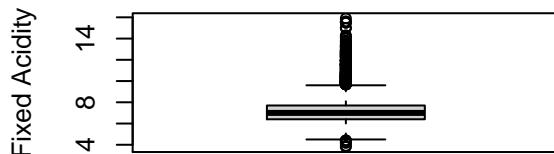
A continuación vamos a mostrar cómo se distribuye cada una de las variables que componen nuestro juego de datos.

- *fixed.acidity*

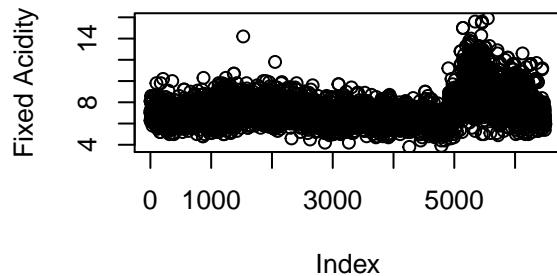
```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$fixed.acidity, main="Histogram of Fixed Acidity",
      xlab ="Fixed Acidity")
boxplot(Dataset$fixed.acidity, main="Boxplot of Fixed Acidity",
        ylab="Fixed Acidity")
plot(Dataset$fixed.acidity, main="ScatterPlot of Fixed Acidity",
      ylab="Fixed Acidity")
```



**Boxplot of Fixed Acidity**



**ScatterPlot of Fixed Acidity**



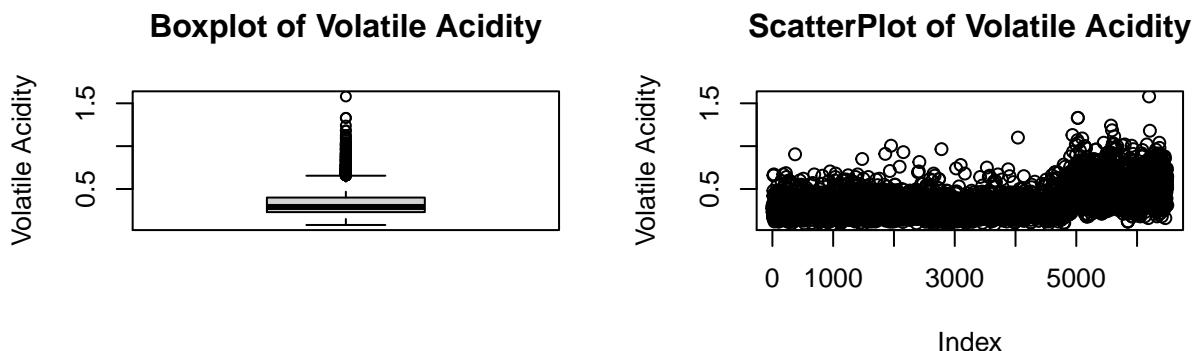
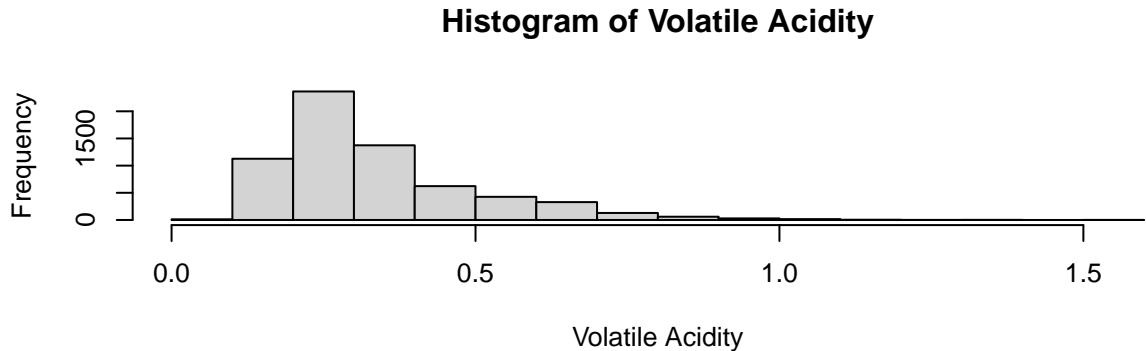
```
which.max(table(Dataset$fixed.acidity))
```

```
## 6.8
## 30
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *fixed.acidity*. Además, el valor modal de la variable *fixed.acidity* es de 6.8 unidades.

- *volatile.acidity*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$volatile.acidity, main="Histogram of Volatile Acidity",
      xlab ="Volatile Acidity")
boxplot(Dataset$volatile.acidity, main="Boxplot of Volatile Acidity",
        ylab="Volatile Acidity")
plot(Dataset$volatile.acidity, main="ScatterPlot of Volatile Acidity",
      ylab="Volatile Acidity")
```



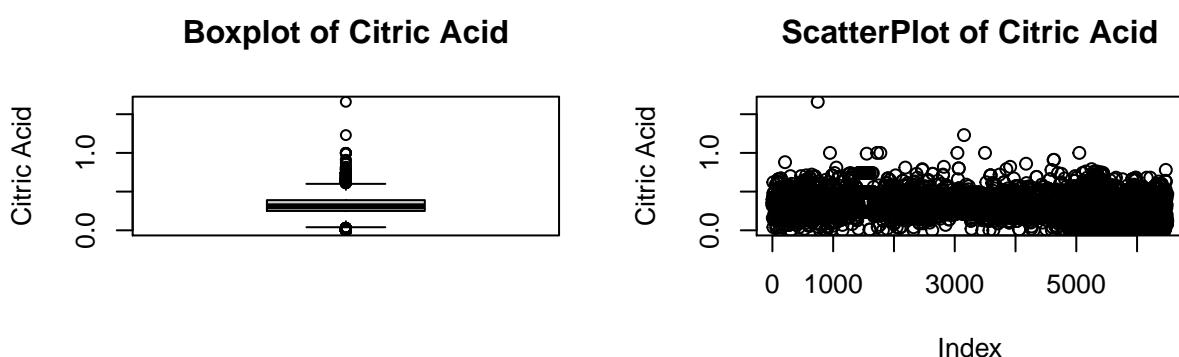
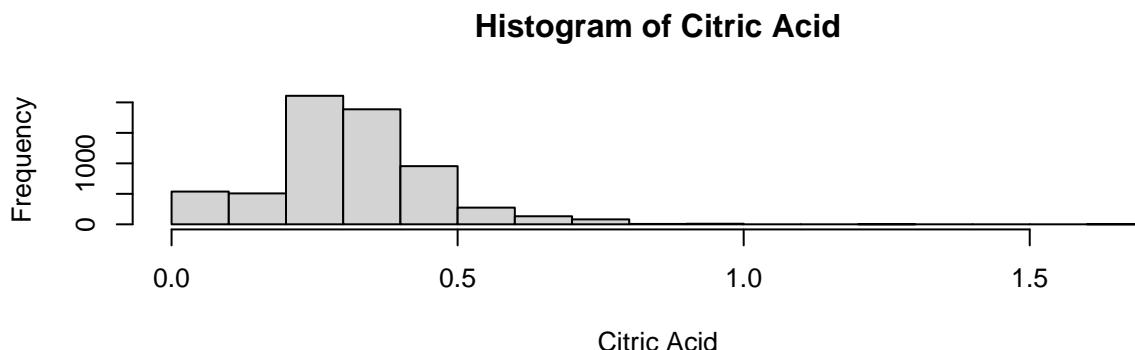
```
which.max(table(Dataset$volatile.acidity))
```

```
## 0.28
## 39
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *volatile.acidity*. Además, el valor modal de la variable *volatile.acidity* es de 0.28 unidades.

- *citric.acid*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$citric.acid, main="Histogram of Citric Acid", xlab ="Citric Acid")
boxplot(Dataset$citric.acid, main="Boxplot of Citric Acid", ylab="Citric Acid")
plot(Dataset$citric.acid, main="ScatterPlot of Citric Acid", ylab="Citric Acid")
```



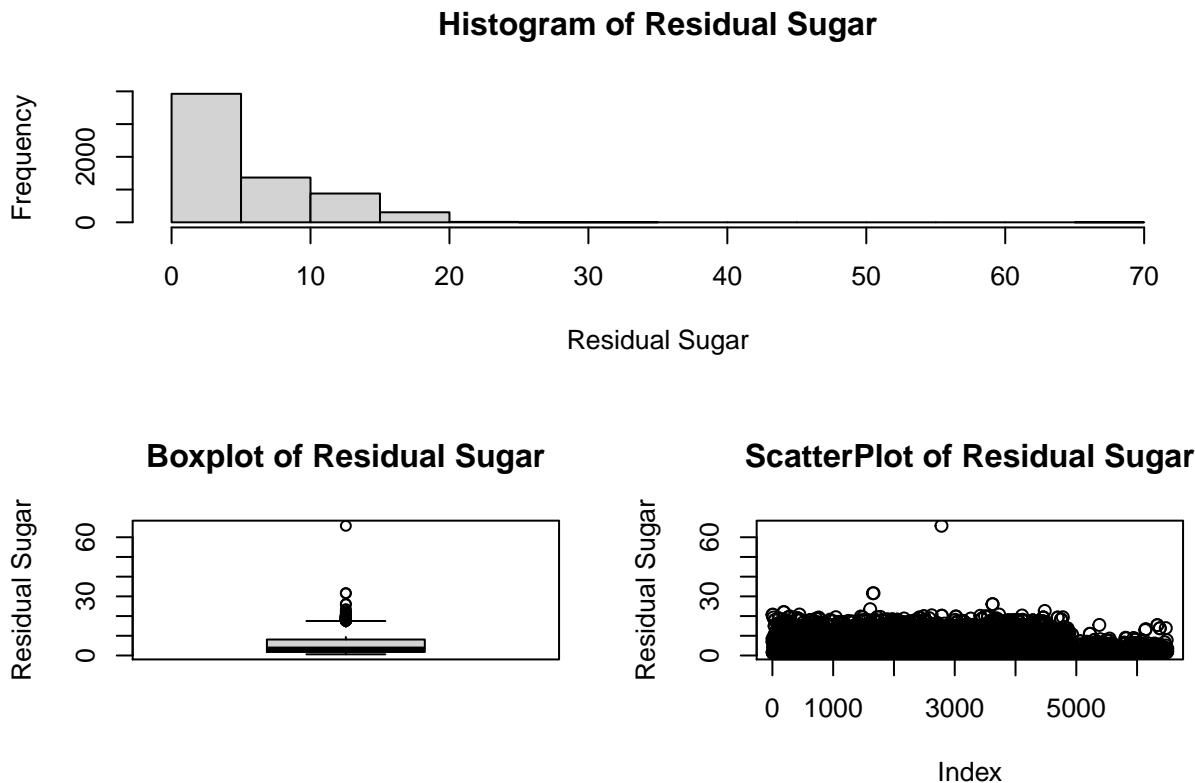
```
which.max(table(Dataset$citric.acid))
```

```
## 0.3
## 31
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *citric.acid*. Además, el valor modal de la variable *citric.acid* es de 0.3 unidades.

- *residual.sugar*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$residual.sugar, main="Histogram of Residual Sugar",
      xlab ="Residual Sugar")
boxplot(Dataset$residual.sugar, main="Boxplot of Residual Sugar",
        ylab="Residual Sugar")
plot(Dataset$residual.sugar, main="ScatterPlot of Residual Sugar",
      ylab="Residual Sugar")
```



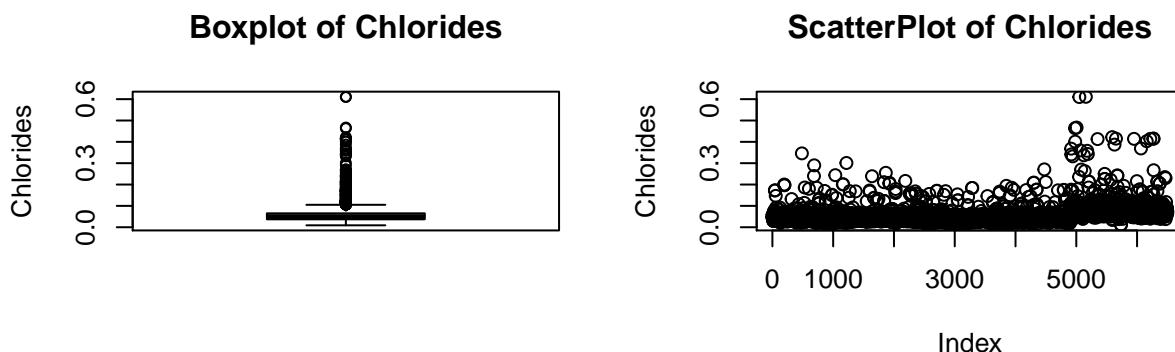
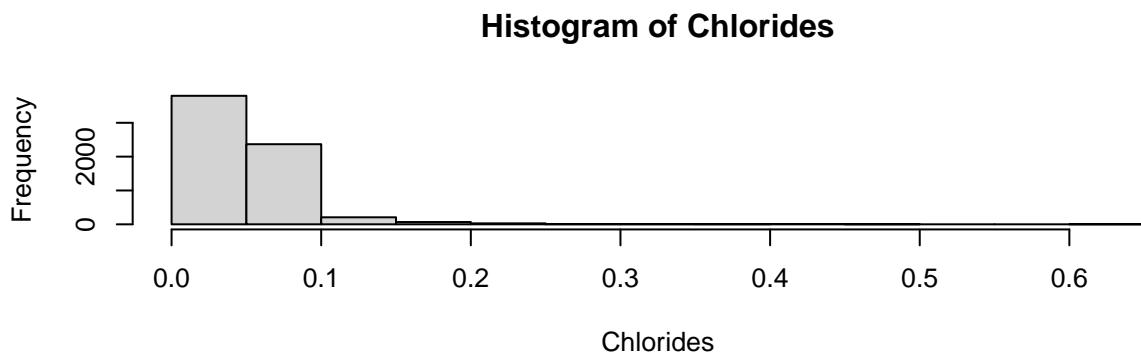
```
which.max(table(Dataset$residual.sugar))
```

```
##   2
## 26
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *residual.sugar*. Además, el valor modal de la variable *residual.sugar* es de 2.0 unidades.

- *chlorides*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$chlorides, main="Histogram of Chlorides", xlab ="Chlorides")
boxplot(Dataset$chlorides, main="Boxplot of Chlorides", ylab="Chlorides")
plot(Dataset$chlorides, main="ScatterPlot of Chlorides", ylab="Chlorides")
```



```
which.max(table(Dataset$chlorides))
```

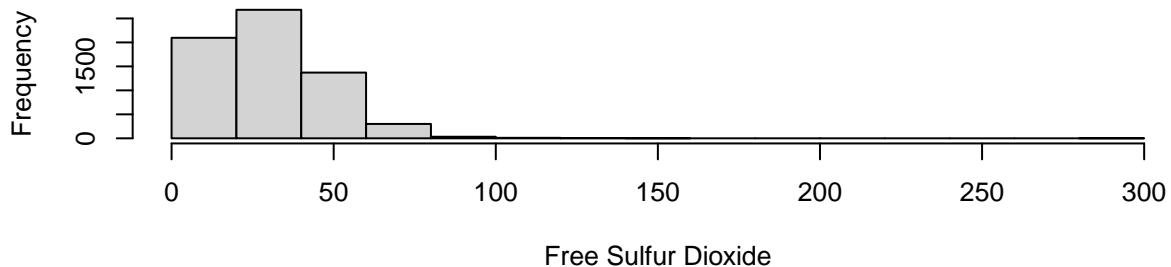
```
## 0.044
##      34
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *chlorides*. Además, el valor modal de la variable *chlorides* es de 0.044 unidades.

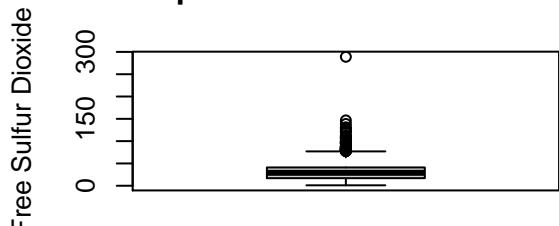
- *free.sulfur.dioxide*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$free.sulfur.dioxide, main="Histogram of Free Sulfur Dioxide",
     xlab ="Free Sulfur Dioxide")
boxplot(Dataset$free.sulfur.dioxide, main="Boxplot of Free Sulfur Dioxide",
        ylab="Free Sulfur Dioxide")
plot(Dataset$free.sulfur.dioxide, main="ScatterPlot of Free Sulfur Dioxide",
      ylab="Free Sulfur Dioxide")
```

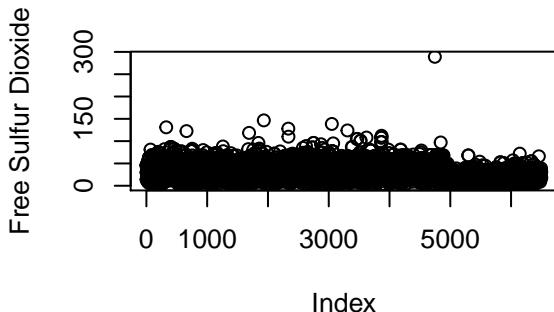
### Histogram of Free Sulfur Dioxide



### Boxplot of Free Sulfur Dioxide



### ScatterPlot of Free Sulfur Dioxide



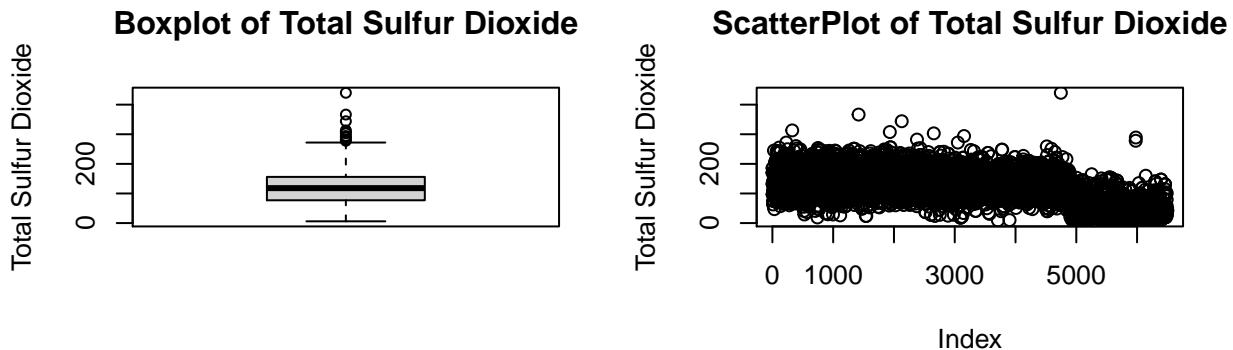
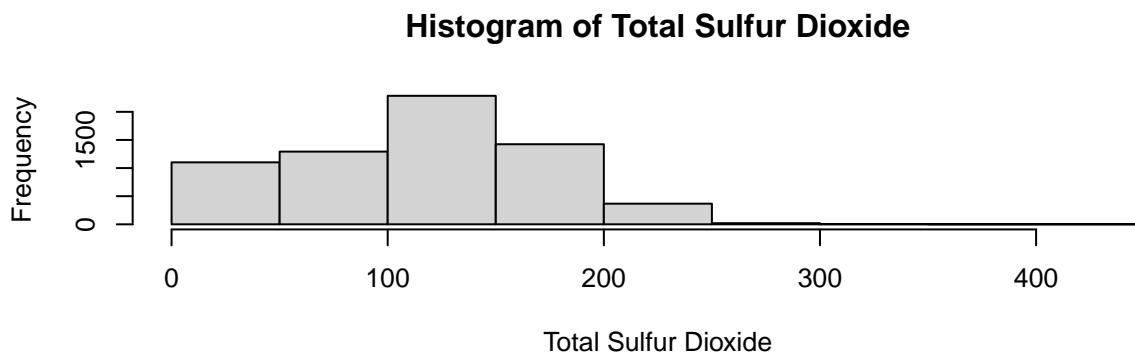
```
which.max(table(Dataset$free.sulfur.dioxide))
```

```
## 29  
## 35
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *free.sulfur.dioxide*. Además, el valor modal de la variable *free.sulfur.dioxide* es de 29.00 unidades.

- *total.sulfur.dioxide*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))  
hist(Dataset$total.sulfur.dioxide, main="Histogram of Total Sulfur Dioxide",  
     xlab ="Total Sulfur Dioxide")  
boxplot(Dataset$total.sulfur.dioxide, main="Boxplot of Total Sulfur Dioxide",  
        ylab="Total Sulfur Dioxide")  
plot(Dataset$total.sulfur.dioxide, main="ScatterPlot of Total Sulfur Dioxide",  
      ylab="Total Sulfur Dioxide")
```



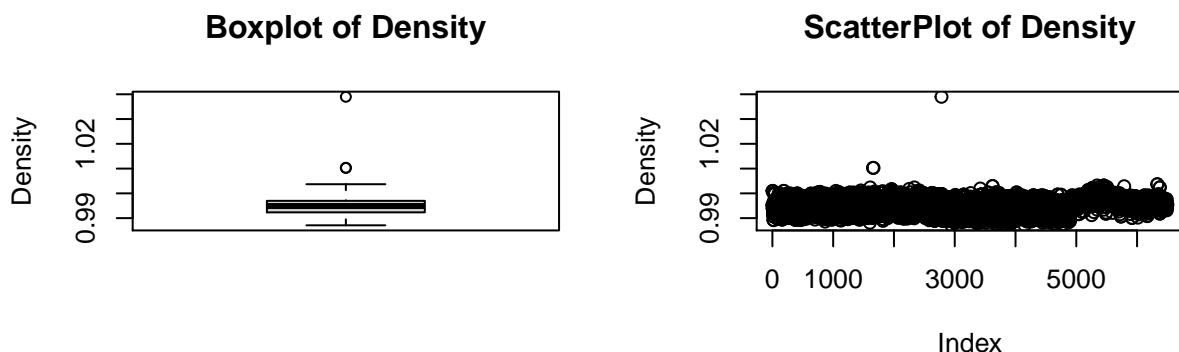
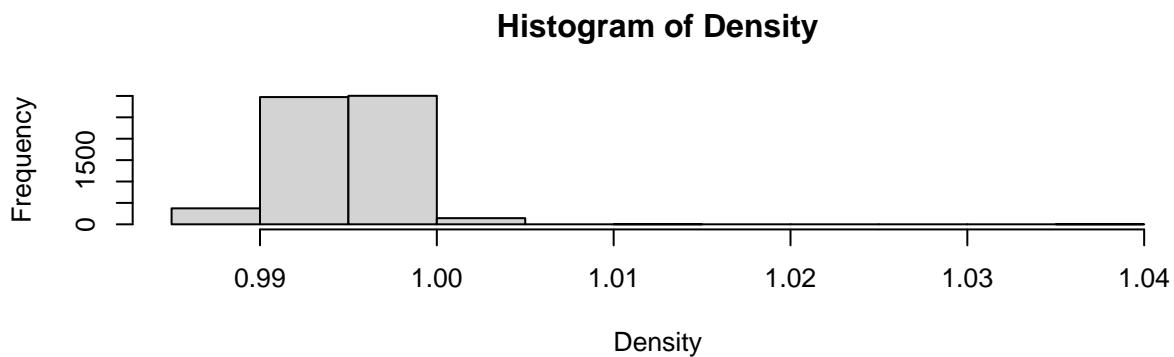
```
which.max(table(Dataset$total.sulfur.dioxide))
```

```
## 111
## 107
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *total.sulfur.dioxide*. Además, el valor modal de la variable *total.sulfur.dioxide* es de 111.00 unidades.

- *density*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$density, main="Histogram of Density", xlab ="Density")
boxplot(Dataset$density, main="Boxplot of Density", ylab="Density")
plot(Dataset$density, main="ScatterPlot of Density", ylab="Density")
```



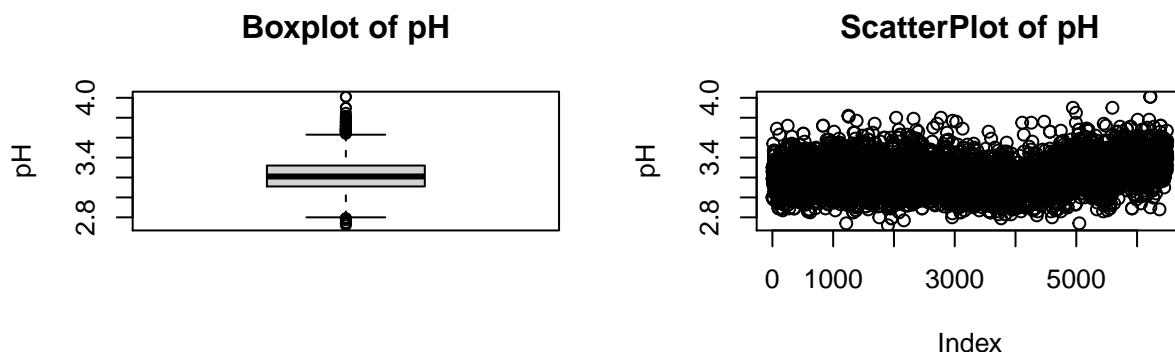
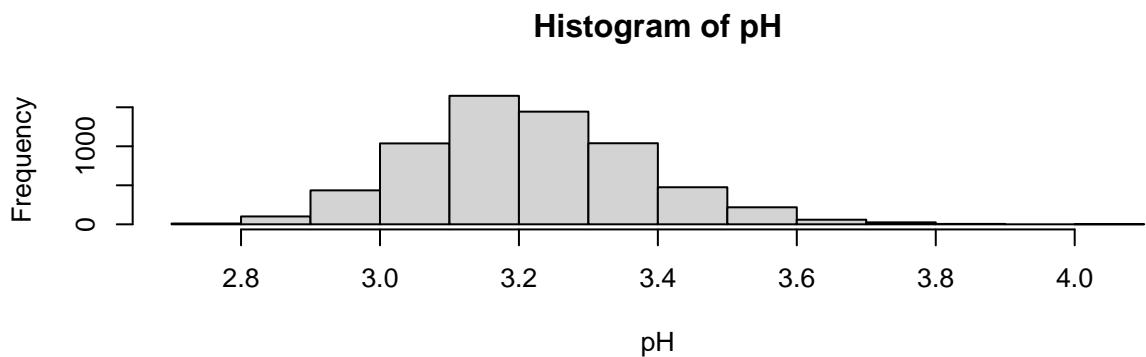
```
which.max(table(Dataset$density))
```

```
## 0.9972
##      774
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *density*. Además, el valor modal de la variable *density* es de 0.9972 unidades.

- pH

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$pH, main="Histogram of pH", xlab ="pH")
boxplot(Dataset$pH, main="Boxplot of pH", ylab="pH")
plot(Dataset$pH, main="ScatterPlot of pH", ylab="pH")
```



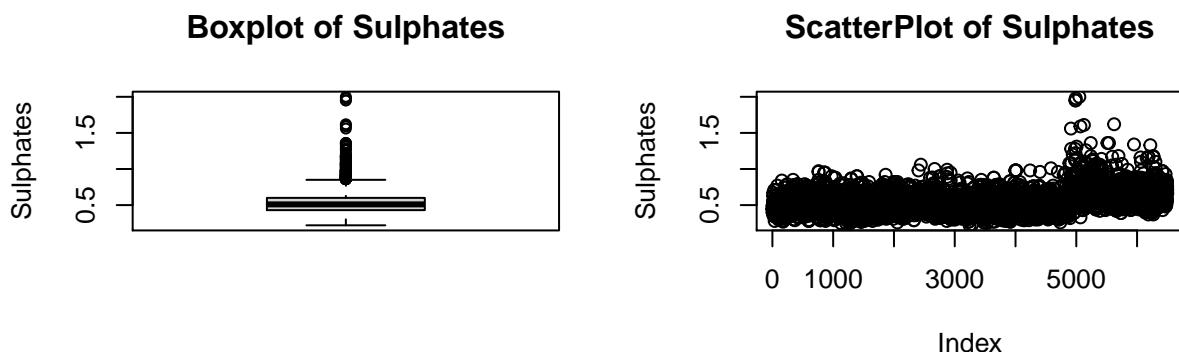
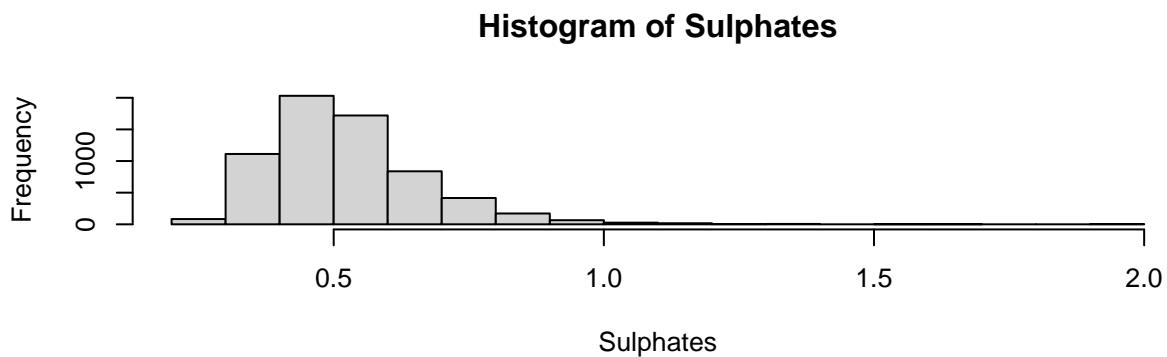
```
which.max(table(Dataset$pH))
```

```
## 3.16
##    40
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *pH*. Además, el valor modal de la variable *pH* es de 3.16 unidades.

- *sulphates*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$sulphates, main="Histogram of Sulphates", xlab ="Sulphates")
boxplot(Dataset$sulphates, main="Boxplot of Sulphates", ylab="Sulphates")
plot(Dataset$sulphates, main="ScatterPlot of Sulphates", ylab="Sulphates")
```



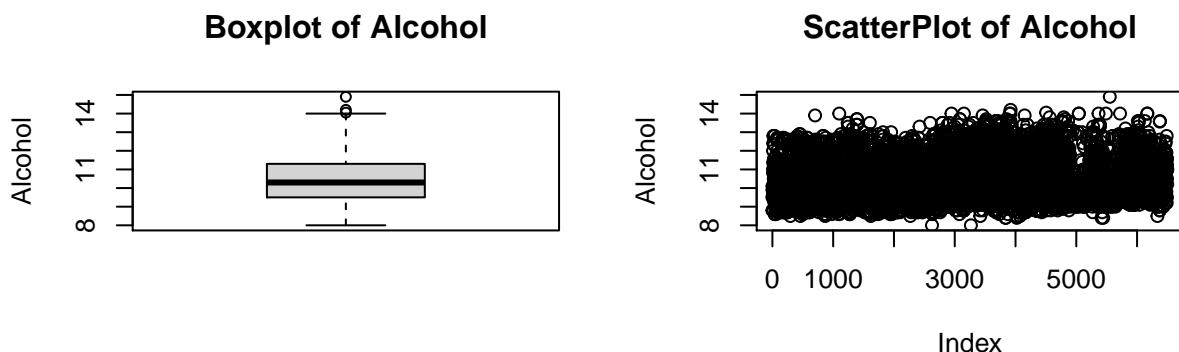
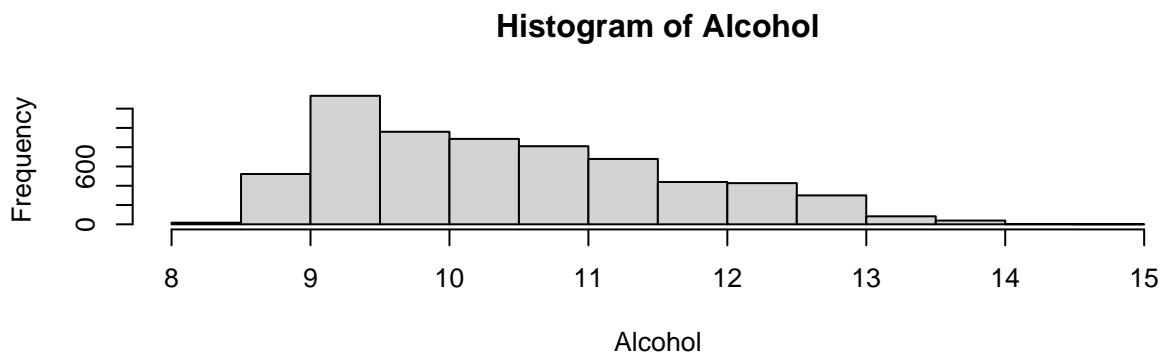
```
which.max(table(Dataset$sulphates))
```

```
## 0.5
## 28
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *sulphates*. Además, el valor modal de la variable *sulphates* es de 0.5 unidades.

- *alcohol*

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(Dataset$alcohol, main="Histogram of Alcohol", xlab ="Alcohol")
boxplot(Dataset$alcohol, main="Boxplot of Alcohol", ylab="Alcohol")
plot(Dataset$alcohol, main="ScatterPlot of Alcohol", ylab="Alcohol")
```



```
which.max(table(Dataset$alcohol))
```

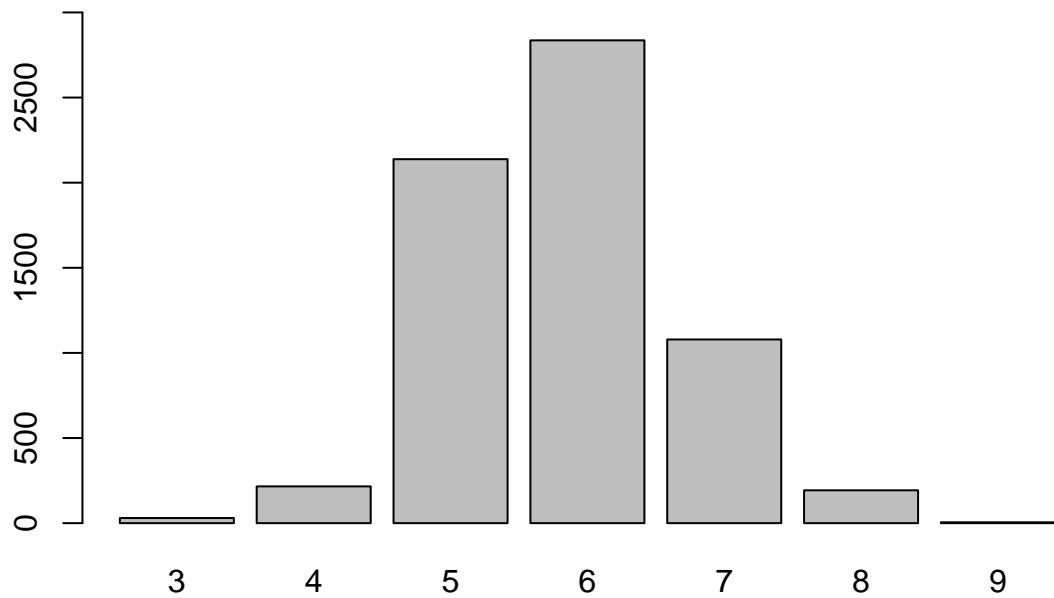
```
## 9.5
## 16
```

Observando las distintas visualizaciones podemos contemplar que existen una serie de outliers o valores atípicos para la variable *alcohol*. Además, el valor modal de la variable *alcohol* es de 9.5 unidades.

- *quality*

```
barplot(table(Dataset$quality), main="Distribución de Quality", ylim=c(0, 3000))
```

## Distribución de Quality



```
which.max(table(Dataset$quality))
```

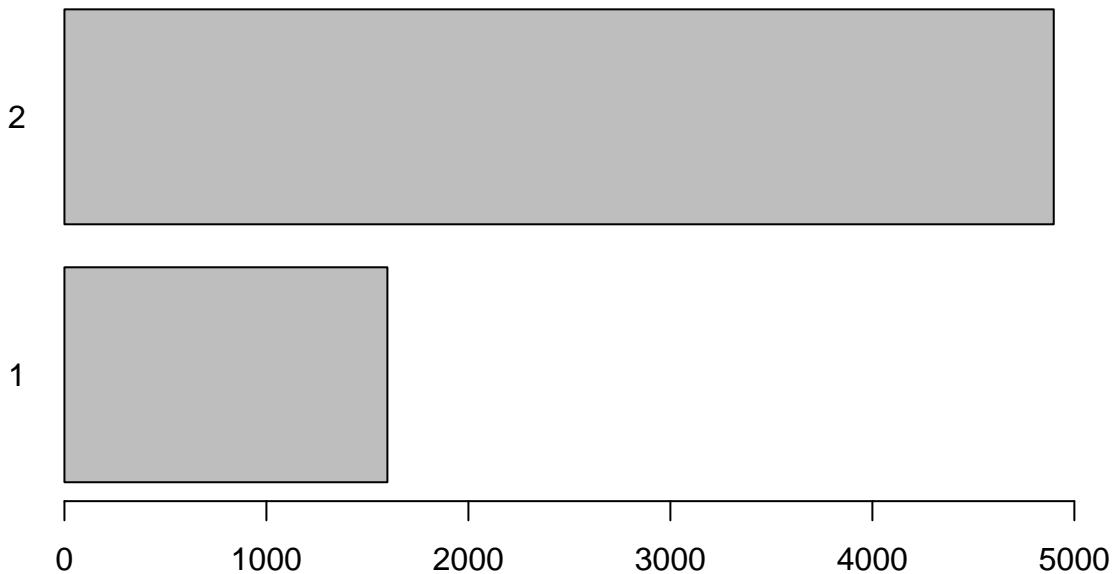
```
## 6  
## 4
```

Podemos observar que el valor modal de la variable *quality* es de 6 unidades.

- *type*

```
barplot(table(Dataset$type), main="Distribución de Type", xlim=c(0, 5000),  
       horiz=TRUE, las=1)
```

## Distribución de Type



Como podemos comprobar, en nuestro juego de datos existen 1599 *vinos rojos* y 4898 *vinos blancos*. Por lo que *type=2* se encontrará 4898 veces y *type=1* se encontrará 1599 veces.

Como se puede observar, en cada uno de los atributos que componen el juego de datos se encuentran valores atípicos u outliers. Esto puede ser consecuencia de que los datos se encuentren desequilibrados en relación a los distintos tipos de calidad de vinos. Por lo cual, no sería necesario eliminar estas observaciones atípicas debido a que forman parte del dominio de datos de cada una de las variables y las consideramos como legítimas para su estudio. Esto se puede observar en la propia descripción del dataset. Si se eliminaran estas observaciones, se produciría un sesgo debido a que sólo obtendríamos vinos de una calidad, o baja o alta.

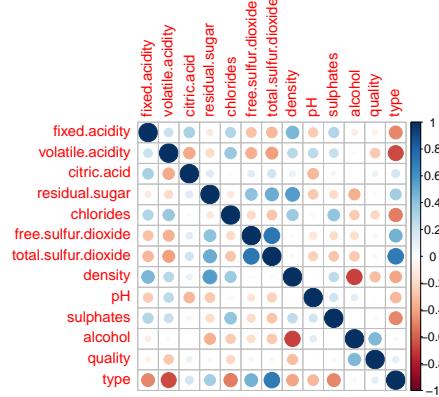
### 6.2. Correlación entre variables

A continuación, estudiaremos la correlación entre las distintas variables de las que se compone el juego de datos mediante el paquete corrplot.

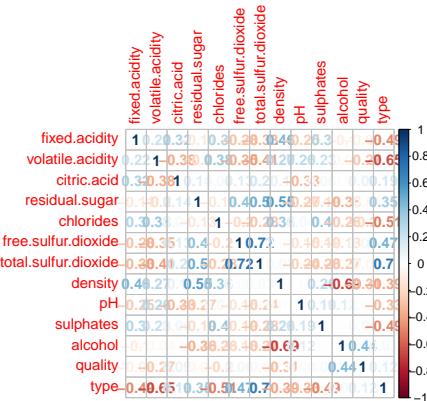
```
library(corrplot)

## corrplot 0.84 loaded

matriz = cor(Dataset, method = "pearson")
corrplot(matriz, method="circle")
```



```
corrplot(matz, method="number")
```



```
matriz
```

##	fixed.acidity	volatile.acidity	citric.acid	residual.sugar
## fixed.acidity	1.0000000	0.21900826	0.32443573	-0.11198128
## volatile.acidity	0.21900826	1.0000000	-0.37798132	-0.19601117
## citric.acid	0.32443573	-0.37798132	1.0000000	0.14245123
## residual.sugar	-0.11198128	-0.19601117	0.14245123	1.0000000
## chlorides	0.29819477	0.37712428	0.03899801	-0.12894050
## free.sulfur.dioxide	-0.28273543	-0.35255731	0.13312581	0.40287064
## total.sulfur.dioxide	-0.32905390	-0.41447619	0.19524198	0.49548159
## density	0.45890998	0.27129565	0.09615393	0.55251695
## pH	-0.25270047	0.26145440	-0.32980819	-0.26731984
## sulphates	0.29956774	0.22598368	0.05619730	-0.18592741
## alcohol	-0.09545152	-0.03764039	-0.01049349	-0.35941477
## quality	-0.07674321	-0.26569948	0.08553172	-0.03698048
## type	-0.48673983	-0.65303559	0.18739650	0.34882101
	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	
## fixed.acidity	0.29819477	-0.28273543	-0.32905390	
## volatile.acidity	0.37712428	-0.35255731	-0.41447619	
## citric.acid	0.03899801	0.13312581	0.19524198	
## residual.sugar	-0.12894050	0.40287064	0.49548159	
## chlorides	1.0000000	-0.19504479	-0.27963045	
## free.sulfur.dioxide	-0.19504479	1.0000000	0.72093408	
## total.sulfur.dioxide	-0.27963045	0.72093408	1.0000000	

```

## density          0.36261466      0.02571684      0.03239451
## pH              0.04470798     -0.14585390     -0.23841310
## sulphates       0.39559331     -0.18845725     -0.27572682
## alcohol         -0.25691558     -0.17983843     -0.26573964
## quality         -0.20066550      0.05546306     -0.04138545
## type            -0.51267825      0.47164366      0.70035716
##                  density      pH      sulphates      alcohol
## fixed.acidity   0.45890998  -0.25270047  0.299567744 -0.095451523
## volatile.acidity 0.27129565   0.26145440  0.225983680 -0.037640386
## citric.acid    0.09615393  -0.32980819  0.056197300 -0.010493492
## residual.sugar  0.55251695  -0.26731984  -0.185927405 -0.359414771
## chlorides        0.36261466   0.04470798  0.395593307 -0.256915580
## free.sulfur.dioxide 0.02571684  -0.14585390  -0.188457249 -0.179838435
## total.sulfur.dioxide 0.03239451  -0.23841310  -0.275726820 -0.265739639
## density         1.00000000   0.01168608  0.259478495 -0.686745422
## pH              0.01168608   1.00000000  0.192123407 0.121248467
## sulphates       0.25947850   0.19212341  1.000000000 -0.003029195
## alcohol         -0.68674542   0.12124847  -0.003029195 1.000000000
## quality         -0.30585791   0.01950570  0.038485446 0.444318520
## type            -0.39064532  -0.32912865  -0.487217970 0.032969551
##                  quality      type
## fixed.acidity   -0.07674321 -0.48673983
## volatile.acidity -0.26569948 -0.65303559
## citric.acid    0.08553172  0.18739650
## residual.sugar -0.03698048  0.34882101
## chlorides        0.20066550  -0.51267825
## free.sulfur.dioxide 0.05546306  0.47164366
## total.sulfur.dioxide -0.04138545  0.70035716
## density         -0.30585791 -0.39064532
## pH              0.01950570  -0.32912865
## sulphates       0.03848545  -0.48721797
## alcohol         0.44431852  0.03296955
## quality         1.00000000  0.11932328
## type            0.11932328  1.00000000

```

De esta matriz de correlación de Pearson podemos observar las relaciones entre las distintas variables disponibles en el conjunto de datos. Atendiendo a ella, podemos destacar:

- La relación entre la variable entre *free.sulfur.oxide* y *total.free.oxide* tiene un valor de 0.72. Por lo cual, se trataría de una relación fuerte y positiva. Esto es debido a que ambas variables tratan aspectos similares.
- La relación entre la variable *type* y *total.sulfur.dioxide* es fuerte y negativa con un valor de -0.7. Con esto podemos llegar a la conclusión de que los vinos blancos tengan un valor para la variable *total.sulfur.dioxide* alto, y viceversa con los vinos rojos.
- La relación entre la variable *type* y *volatile.acidity* es fuerte y positiva con un valor de 0.65. Con esto podemos llegar a la conclusión de que los vinos blancos tengan un valor para la variable *volatile.acidity* bajo, y viceversa con los vinos rojos.
- La relación entre la variable *alcohol* y *density* es fuerte y negativa con un valor de -0.69. Con esto podemos llegar a la conclusión de que los vinos que tengan un alto valor de *alcohol*, tengan un valor de densidad bajo.

La matriz de correlación de Pearson indica la relación existente entre variables. Propone valores entre -1 y 1. Mientras más cercano a límite inferior, la relación será negativa, y viceversa. Mientras más se acerque el valor a 0, nos encontraremos a una relación débil, mientras que si nos acercamos a los límites (sobre 1 y -1), la relación entre atributos será fuerte.

Cabe recordar que una relación positiva indica que si se aumenta el valor de una variable, aumenta la de otra. Mientras que, si la relación es negativa, al aumentar el valor de una variable, la otra disminuiría.

Los ejemplos comentados anteriormente hacen referencia a las relaciones más destacables dentro del juego de datos.

### 6.3. Discretización

A continuación, vamos a comprobar si tiene sentido aplicar cualquier método de discretización a alguna variable

```
apply(Dataset, 2, function(x) length(unique(x)))
```

```
##      fixed.acidity    volatile.acidity      citric.acid
##             106                  187                   89
##      residual.sugar      chlorides free.sulfur.dioxide
##             316                  214                  135
##      total.sulfur.dioxide      density          pH
##                276                  998                  108
##      sulphates      alcohol        quality
##                 111                  111                   7
##      type
##                 2
```

Como podemos observar, para las únicas variables a las que tendría sentido aplicar métodos de discretización serían las variables *quality* y *type*.

```
Dataset$quality = as.factor(Dataset[,12])
Dataset$type = as.factor(Dataset[,13])
summary(Dataset)
```

```
##   fixed.acidity    volatile.acidity      citric.acid      residual.sugar
##   Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##   1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2500   1st Qu.: 1.800
##   Median : 7.000   Median :0.2900   Median :0.3100   Median : 3.000
##   Mean   : 7.215   Mean   :0.3397   Mean   :0.3186   Mean   : 5.443
##   3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.3900   3rd Qu.: 8.100
##   Max.   :15.900   Max.   :1.5800   Max.   :1.6600   Max.   :65.800
##
##      chlorides      free.sulfur.dioxide total.sulfur.dioxide      density
##      Min.   :0.00900   Min.   : 1.00      Min.   : 6.0      Min.   :0.9871
##      1st Qu.:0.03800   1st Qu.:17.00     1st Qu.:77.0     1st Qu.:0.9923
##      Median :0.04700   Median :29.00     Median :118.0    Median :0.9949
##      Mean   :0.05603   Mean   :30.53     Mean   :115.7    Mean   :0.9947
##      3rd Qu.:0.06500   3rd Qu.:41.00     3rd Qu.:156.0   3rd Qu.:0.9970
##      Max.   :0.61100   Max.   :289.00    Max.   :440.0    Max.   :1.0390
##
##      pH      sulphates      alcohol        quality      type
##
```

```

## Min. :2.720  Min. :0.2200  Min. : 8.00  3: 30  1:1599
## 1st Qu.:3.110 1st Qu.:0.4300  1st Qu.: 9.50  4: 216 2:4898
## Median :3.210 Median :0.5100  Median :10.30  5:2138
## Mean   :3.219 Mean  :0.5313  Mean  :10.49  6:2836
## 3rd Qu.:3.320 3rd Qu.:0.6000  3rd Qu.:11.30  7:1079
## Max.   :4.010  Max. :2.0000  Max. :14.90  8: 193
##                                         9:    5

```

## 7. Selección de los grupos de datos a analizar a través de PCA

Una vez ya se hayan realizado las acciones de análisis, modelado y limpieza de los datos, se procede a investigar si se puede reducir el número de dimensiones que posee el dataset a través de las distintas características que ofrece el comportamiento de cada uno de los atributos. Cabe destacar que los atributos que intentaremos reducir son aquellos excepto las variables `quality` y `type`. Esto es debido a que las utilizaremos para una futura clasificación. Para ello, se emplearán los siguientes métodos:

```

Dataset.pca = prcomp(Dataset[,c(1:11)], center = TRUE, scale=TRUE)
Dataset.pca

```

```

## Standard deviations (1, .., p=11):
## [1] 1.7406518 1.5791852 1.2475364 0.9851660 0.8484544 0.7793021 0.7232971
## [8] 0.7081739 0.5805377 0.4771748 0.1811927
##
## Rotation (n x k) = (11 x 11):
##                               PC1          PC2          PC3          PC4          PC5
## fixed.acidity      0.23879890 -0.33635454  0.43430130 -0.16434621  0.1474804
## volatile.acidity   0.38075750 -0.11754972 -0.30725942 -0.21278489 -0.1514560
## citric.acid        -0.15238844 -0.18329940  0.59056967  0.26430031  0.1553487
## residual.sugar     -0.34591993 -0.32991418 -0.16468843 -0.16744301  0.3533619
## chlorides           0.29011259 -0.31525799 -0.01667910  0.24474386 -0.6143911
## free.sulfur.dioxide -0.43091401 -0.07193260 -0.13422395  0.35727894 -0.2235323
## total.sulfur.dioxide -0.48741806 -0.08726628 -0.10746230  0.20842014 -0.1581336
## density              0.04493664 -0.58403734 -0.17560555 -0.07272496  0.3065613
## pH                   0.21868644  0.15586900 -0.45532412  0.41455110  0.4533764
## sulphates            0.29413517 -0.19171577  0.07004248  0.64053571  0.1365769
## alcohol               0.10643712  0.46505769  0.26110053  0.10680270  0.1888920
##                               PC6          PC7          PC8          PC9
## fixed.acidity      0.20455371  0.28307944 -0.401235645 -0.3440567
## volatile.acidity   0.49214307  0.38915976  0.087435088  0.4969327
## citric.acid        -0.22763380  0.38128504  0.293412336  0.4026887
## residual.sugar     0.23347775 -0.21797554  0.524872935 -0.1080032
## chlorides           -0.16097639  0.04606816  0.471516850 -0.2964437
## free.sulfur.dioxide 0.34005140  0.29936325 -0.207807585 -0.3666563
## total.sulfur.dioxide 0.15127722  0.13891032 -0.128621319  0.3206955
## density              -0.01874307  0.04675897 -0.004831136 -0.1128800
## pH                   -0.29657890  0.41890702  0.028643277 -0.1278367
## sulphates            0.29692579 -0.52534311 -0.165818022  0.2077642
## alcohol               0.51837780  0.10410343  0.399233887 -0.2518903
##                               PC10         PC11
## fixed.acidity      0.281267685  0.3346792663
## volatile.acidity   -0.152176731  0.0847718098
## citric.acid        -0.234463340 -0.0011089514

```

```

## residual.sugar      0.001372773  0.4497650778
## chlorides          0.196630217  0.0434375867
## free.sulfur.dioxide -0.480243340 -0.0002125351
## total.sulfur.dioxide  0.713663486 -0.0626848131
## density            0.003908289 -0.7151620723
## pH                 0.141310977  0.2063605036
## sulphates          -0.045959499  0.0772024671
## alcohol             0.205053085 -0.3357018784

```

En primer lugar, se muestran las desviaciones estándar de cada una de las componentes a destacar. Podemos observar que su valor desciende a medida que avanzamos a lo largo de las componentes principales.

En segundo lugar, se observa la matriz Rotation, que muestra los valores propios relacionados a los componentes principales. Estos son los coeficientes que asociados con cada variable da lugar a cada componente a través de una combinación lineal.

Por ejemplo, el cálculo de la componente PC1, tenemos:

$PC1 = fixed.acidity * 0.2388 + volatile.acidity * 0.3808 + \dots + alcohol * 0.1064$

Mientras mayor sea en términos absolutos el coeficiente, mayor peso tendrá en el cálculo de la componente.

```
summary(Dataset.pca)
```

```

## Importance of components:
##                               PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation    1.7407 1.5792 1.2475 0.98517 0.84845 0.77930 0.72330
## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521 0.04756
## Cumulative Proportion  0.2754 0.5021 0.6436 0.73187 0.79732 0.85253 0.90009
##                               PC8     PC9     PC10    PC11
## Standard deviation     0.70817 0.58054 0.4772 0.18119
## Proportion of Variance 0.04559 0.03064 0.0207 0.00298
## Cumulative Proportion  0.94568 0.97632 0.9970 1.00000

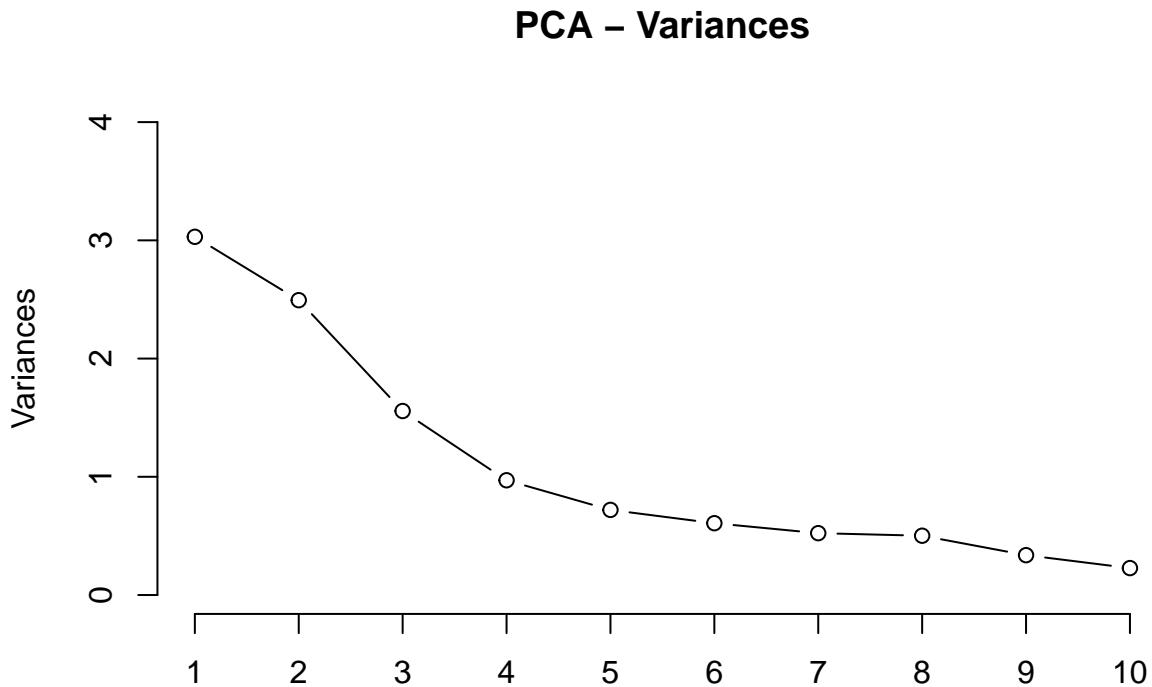
```

De este resumen podemos destacar las siguientes observaciones:

- El primer componente principal es capaz de explicar el 27.54% de la varianza, el segundo componente el 22.67% de esta, y así sucesivamente.
- Las proporciones de varianzas mostradas no son muy altas (la máxima es 27.54%), esto indica la baja dependencia entre atributos.
- Podemos destacar que los tres primeros componentes son aquellos que explican la mayor parte de la varianza, en comparación con el resto.
- Ateniendo a la proporción acumulativa de varianzas, podemos terminar que con las dos primeras componentes se puede explicar el 50.21% de la variabilidad de la muestra, mientras que si seleccionamos las 4 primeras podemos explicar el 73.19% de la muestra. Si seleccionamos todas las componentes explicaríamos el 100% de la totalidad de la muestra, lo que no podríamos reducir el número de variables dentro del juego de datos.

A continuación, vamos a mostrar una serie de visualizaciones con la finalidad de obtener información para determinar el número de componentes que podríamos utilizar para nuestro estudio.

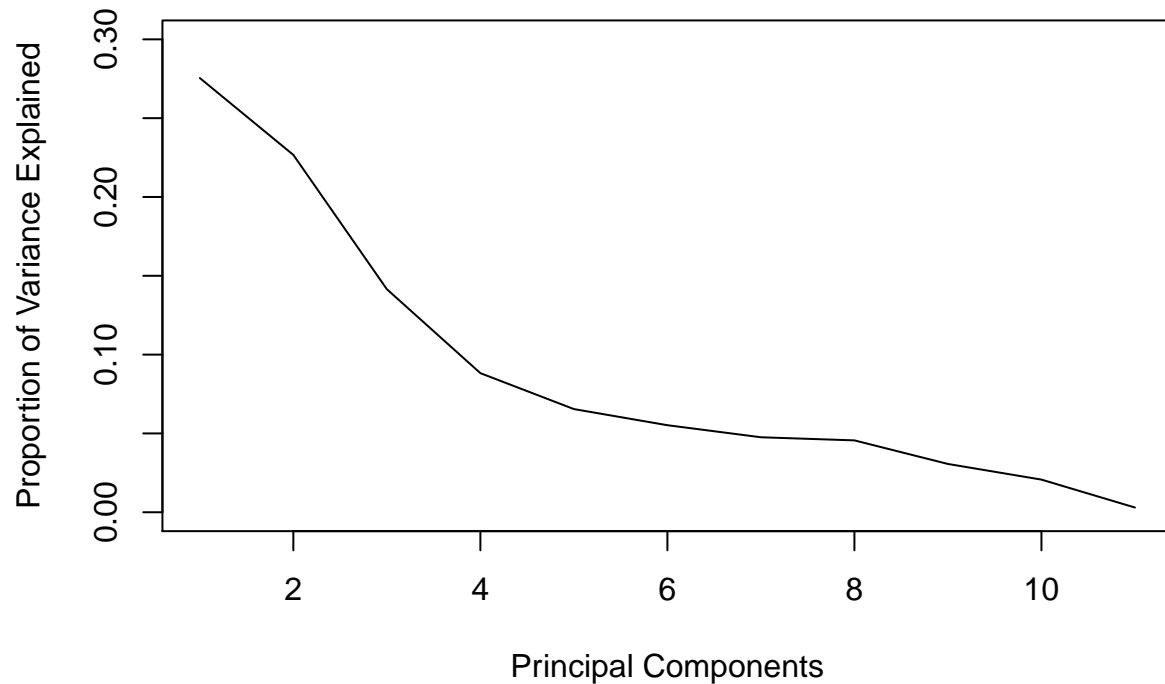
```
plot(Dataset.pca, type="l", ylim=c(0.0,4.0), main="PCA - Variances")
```



Como se puede observar, mientras más avanzamos en los distintos componentes, el valor de la varianza se reduce.

```
plot((Dataset.pca$sdev^2 / sum(Dataset.pca$sdev^2)), type="l", ylim=c(0.0,0.3),
      main="PCA – Proportion of Variance Explained",
      ylab="Proportion of Variance Explained", xlab="Principal Components")
```

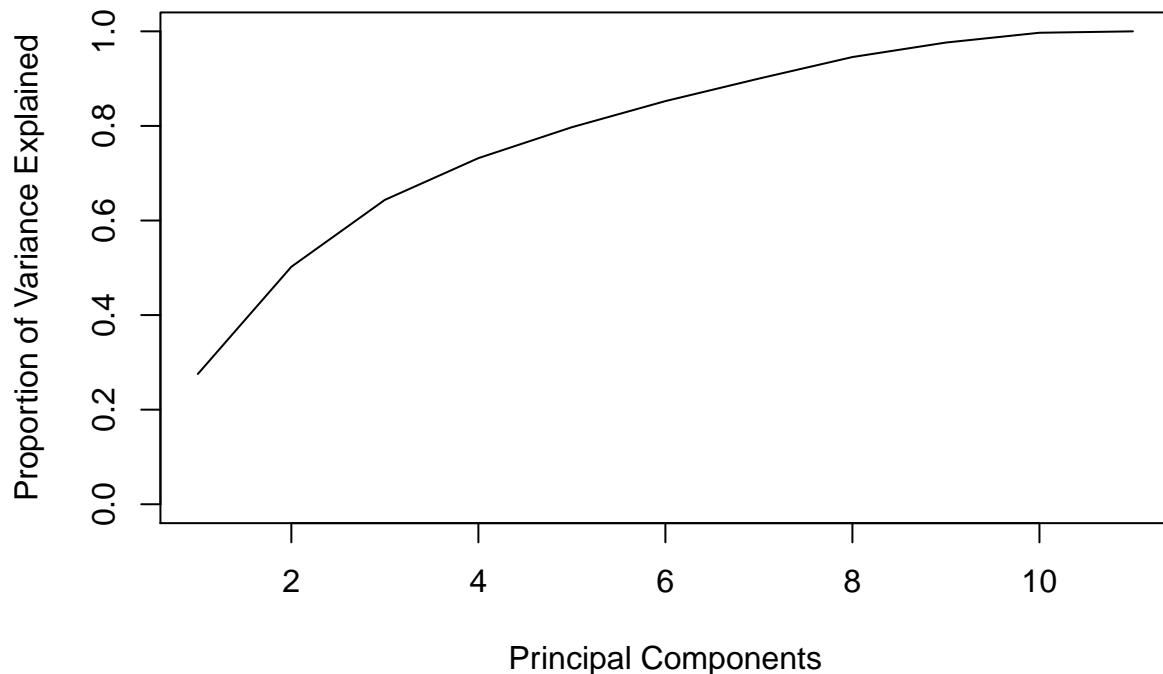
## PCA – Proportion of Variance Explained



Como se puede observar, a medida que avanzamos en las componentes, se reduce la proporción de varianza explicada por cada una de ellas.

```
plot(cumsum(Dataset.pca$sdev^2 / sum(Dataset.pca$sdev^2)), type="l",
      ylim=c(0.0,1.0), main="PCA - Accumulative Proportion of Variance Explained",
      ylab="Proportion of Variance Explained", xlab="Principal Components")
```

## PCA – Accumulative Proportion of Variance Explained



Como se puede observar, a medida que avanzamos en las componentes, aumenta la proporción acumulada de varianza explicada por cada una de ellas hasta llegar al 1.0, que será el 100%.

Atendiendo a las visualizaciones, podemos determinar lo siguiente:

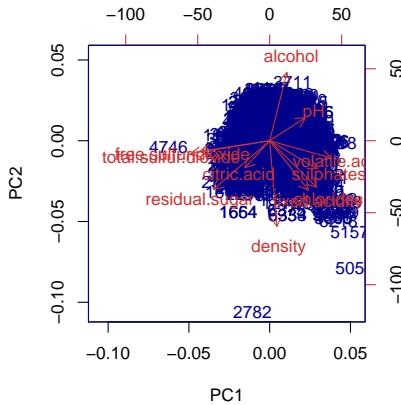
A través del *elbow method* podemos determinar que el *número de componentes* suficientes para explicar la variabilidad de la muestra sería 4, en un 73.187%, lo cual es suficiente para nuestro estudio. Además, es suficiente debido a que como se muestra en este capítulo del libro “A step-by-step approach to using the SAS system for univariate and multivariate statistics”

A continuación, vamos a representar cada una de las puntuaciones de los pares de componentes con la finalidad de encontrar relaciones entre estos.

En estas visualizaciones hay que tener en cuenta lo siguiente:

- A medida que es mayor el tamaño de la flecha, mayor será la influencia del atributo en el componente.
- La dirección de la flecha muestra hacia qué componente representa en mayor medida dicho atributo. Si se mueve en el eje vertical hace referencia a que representa más el componente del eje izquierdo al atributo, y viceversa.
- La separación entre las flechas de las variables muestran su correlación. Si las flechas se encuentran formando un ángulo de 90° no existiría relación entre los atributos involucrados. Y si forman un ángulo de 180° la relación entre ellos es negativa.
- PC1 y PC2

```
biplot(x = Dataset.pca, col= c("blue4", "brown3"), choices = c(1,2))
```



En esta visualización podemos observar la alta correlación positiva entre *total.sulfur.dioxide* y *free.sulfur.dioxide*.

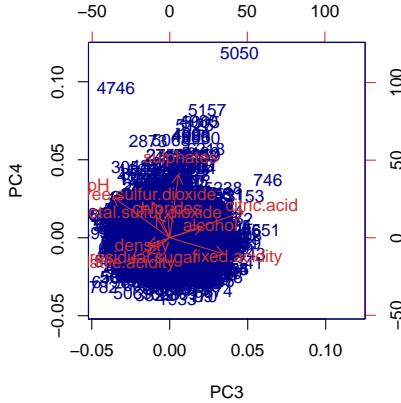
Las variables *density* y *alcohol* son las que tienen una alta influencia. Y además, la relación entre estos atributos es negativa.

Podemos observar que el componente PC2 representa en mayor medida a las variables *density* y *alcohol*.

Podemos observar que

- PC3 y PC4

```
biplot(x = Dataset.pca, col= c("blue4", "brown3"), choices = c(3,4))
```



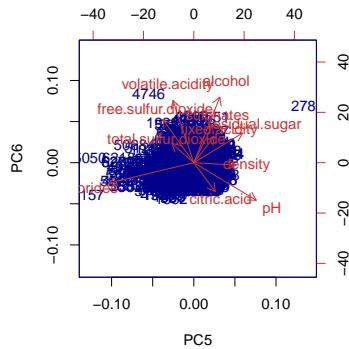
En esta visualización podemos observar las bajas correlaciones entre atributos, en términos generales.

Las variables *sulphates* y *citric.acid* son las que tienen una alta influencia.

Podemos observar que el componente PC3 representa en mayor medida a la variable *sulphates*.

- PC5 y PC6

```
biplot(x = Dataset.pca, col= c("blue4", "brown3"), choices = c(5,6))
```



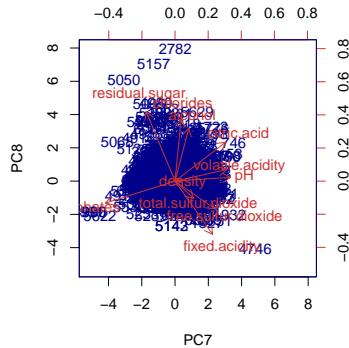
En esta visualización podemos observar la correlación negativa entre *chlorides* y *residual.sugar*.

Las variables *alcohol*, *chlorides* y *pH* son las que tienen una alta influencia.

Podemos observar que el componente PC5 representa en mayor medida a la variable *density*.

- PC7 y PC8

```
biplot(x = Dataset.pca, scale=0, col= c("blue4", "brown3"), choices = c(7,8))
```



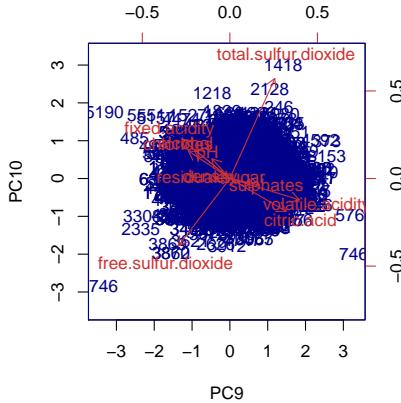
En esta visualización podemos observar la correlación negativa entre *sulphates* y *volatile.acidity*.

Las variables *fixed.acidity*, *sulphates* y *residual.sugar* son las que tienen una alta influencia.

Podemos observar que el componente PC8 representa en mayor medida a la variable *pH*.

- PC9 y PC10

```
biplot(x = Dataset.pca, scale=0, col= c("blue4", "brown3"), choices = c(9,10))
```



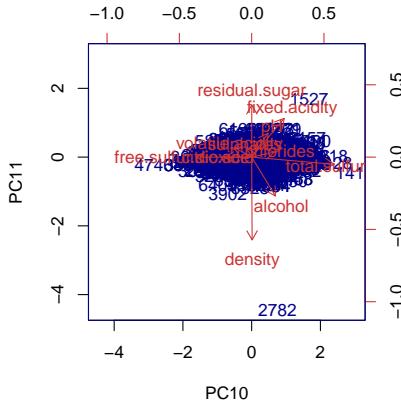
En esta visualización podemos observar la alta correlación negativa entre *free.sulfur.dioxide* y *total.sulfur.dioxide*.

Las variables *free.sulfur.dioxide* y *total.sulfur.dioxide* son las que tienen una alta influencia.

Podemos observar que el componente PC9 representa en mayor medida a la variable *residual.sugar*.

- PC10 y PC11

```
biplot(x = Dataset.pca, scale=0,col= c("blue4", "brown3"), choices = c(10,11))
```



En esta visualización podemos observar la alta correlación negativa entre *density* y *residual.sugar*.

Las variables *residual.sugar* y *density* son las que tienen una alta influencia.

Podemos observar que el componente PC11 representa en mayor medida a las variables *density* y *residual.sugar*.

*Creación del nuevo Dataset* A continuación, vamos a crear el nuevo Dataset a partir de los componentes más representativos. Recordemos que serán los 4 primeros componentes y los dos últimos atributos del juego de datos original.

```
pc1 <- apply(Dataset.pca$rotation[,1]*Dataset[,c(1:11)], 1, sum)
pc2 <- apply(Dataset.pca$rotation[,2]*Dataset[,c(1:11)], 1, sum)
pc3 <- apply(Dataset.pca$rotation[,3]*Dataset[,c(1:11)], 1, sum)
```

```

pc4 <- apply(Dataset.pca$rotation[,4]*Dataset[,c(1:11)], 1, sum)

data.new.pca = cbind(pc1,pc2,pc3,pc4)
data.new.pca = as.data.frame(data.new.pca)
colnames(data.new.pca) = c("PC1","PC2","PC3","PC4")
head(data.new.pca)

##          PC1        PC2        PC3        PC4
## 1  50.248335 -36.97711  45.40840 123.834016
## 2   7.217185  53.93211  31.36916 15.141643
## 3  27.221605 -46.78807  42.55142 -6.049857
## 4  48.351153 -35.87759 -61.66627 -21.208762
## 5 -50.793406 -42.43087  98.34508  65.035997
## 6 -32.068087 -54.10680 -21.81492 -6.371574

data.new.pca$quality = as.numeric(Dataset$quality)
data.new.pca$type = as.numeric(Dataset$type)

```

## 8. Comprobación de normalidad y homogeneidad de la varianza

### 8.1. Comprobación de normalidad

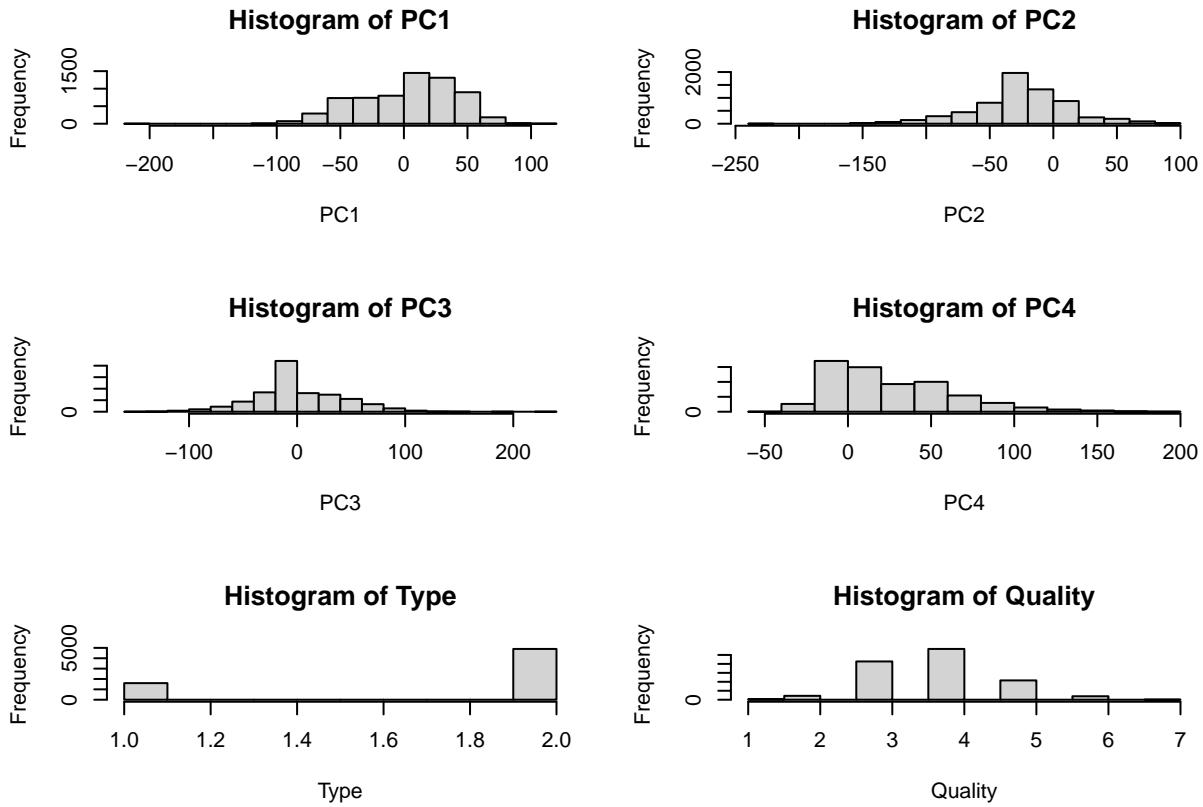
A continuación, vamos a comprobar que los valores que toman nuestras variables cuantitativas provienen de una población distribuida normalmente.

En primer lugar, vamos a representar a través de Histogramas las distintas variables para conocer su distribución de datos a priori.

```

par(mfrow=c(3,2))
hist(data.new.pca$PC1, main="Histogram of PC1", xlab ="PC1")
hist(data.new.pca$PC2, main="Histogram of PC2", xlab ="PC2")
hist(data.new.pca$PC3, main="Histogram of PC3", xlab ="PC3")
hist(data.new.pca$PC4, main="Histogram of PC4", xlab ="PC4")
hist(data.new.pca$type, main="Histogram of Type", xlab ="Type")
hist(data.new.pca$quality, main="Histogram of Quality", xlab ="Quality")

```



Visualizando los distintos histogramas, podemos contemplar que las variables PC1, PC2, PC3 y PC4 sí siguen una distribución normal debido a su forma similar a la campana de Gauss. Aunque en la variable PC4 no se ve de forma tan clara.

Debido a que esto es una medida subjetiva. Procederemos a realizar dos tipos de tests estadísticos para comprobar si las distintas variables siguen una distribución normal, como son el test de Anderson-Darling y el test de Lilliefors que se basa en el test de Kolmogorov-Smirnov.

Cabe destacar que si el valor del estadístico p es superior al nivel de significación 0.05, pues se considera que la variable sigue una distribución normal, y viceversa.

Esta serie de test estadísticos se encuentra disponible en el paquete nortest

```
library(nortest)
```

```
## Warning: package 'nortest' was built under R version 4.0.3
```

```
require(nortest)
```

```
# Test de Anderson-Darling
ad.test(data.new.pca$PC1)
```

```
##
## Anderson-Darling normality test
##
```

```

## data: data.new.pca$PC1
## A = 62.444, p-value < 2.2e-16

ad.test(data.new.pca$PC2)

##
## Anderson-Darling normality test
##
## data: data.new.pca$PC2
## A = 46.969, p-value < 2.2e-16

ad.test(data.new.pca$PC3)

##
## Anderson-Darling normality test
##
## data: data.new.pca$PC3
## A = 62.242, p-value < 2.2e-16

ad.test(data.new.pca$PC4)

##
## Anderson-Darling normality test
##
## data: data.new.pca$PC4
## A = 113.2, p-value < 2.2e-16

ad.test(data.new.pca$type)

##
## Anderson-Darling normality test
##
## data: data.new.pca$type
## A = 1571.5, p-value < 2.2e-16

ad.test(data.new.pca$quality)

##
## Anderson-Darling normality test
##
## data: data.new.pca$quality
## A = 367.17, p-value < 2.2e-16

# Test de Lilliefors

lillie.test(data.new.pca$PC1)

##
## Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: data.new.pca$PC1
## D = 0.073096, p-value < 2.2e-16

```

```

lillie.test(data.new.pca$PC2)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: data.new.pca$PC2
## D = 0.072447, p-value < 2.2e-16

lillie.test(data.new.pca$PC3)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: data.new.pca$PC3
## D = 0.096693, p-value < 2.2e-16

lillie.test(data.new.pca$PC4)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: data.new.pca$PC4
## D = 0.11052, p-value < 2.2e-16

lillie.test(data.new.pca$type)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: data.new.pca$type
## D = 0.47, p-value < 2.2e-16

lillie.test(data.new.pca$quality)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: data.new.pca$quality
## D = 0.22107, p-value < 2.2e-16

```

Teniendo en cuenta los valores de los tests realizados. Podemos observar que ninguna de las variables que tenemos disponibles siguen una distribución normal.

## 8.2. Comprobación de la homocedasticidad de la varianza

Para ello, utilizaremos el test de Fligner-Killen debido a que las variables que tenemos disponibles no cumplen con la condición de normalidad.

La hipótesis nula de este tipo de test asume la igualdad de varianzas. Cabe destacar que si el valor del estadístico p es superior al nivel de significación 0.05, se indica homocedasticidad, y viceversa.

```

fligner.test(PC1 ~ type, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: PC1 by type
## Fligner-Killeen:med chi-squared = 1159.8, df = 1, p-value < 2.2e-16

fligner.test(PC2 ~ type, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: PC2 by type
## Fligner-Killeen:med chi-squared = 541.52, df = 1, p-value < 2.2e-16

fligner.test(PC3 ~ type, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: PC3 by type
## Fligner-Killeen:med chi-squared = 632.91, df = 1, p-value < 2.2e-16

fligner.test(PC4 ~ type, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: PC4 by type
## Fligner-Killeen:med chi-squared = 1246.4, df = 1, p-value < 2.2e-16

fligner.test(PC1 ~ quality, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: PC1 by quality
## Fligner-Killeen:med chi-squared = 30.151, df = 6, p-value = 3.68e-05

fligner.test(PC2 ~ quality, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: PC2 by quality
## Fligner-Killeen:med chi-squared = 20.578, df = 6, p-value = 0.002184

```

```

fligner.test(PC3 ~ quality, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: PC3 by quality
## Fligner-Killeen:med chi-squared = 10.347, df = 6, p-value = 0.1108

fligner.test(PC4 ~ quality, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: PC4 by quality
## Fligner-Killeen:med chi-squared = 20.192, df = 6, p-value = 0.00256

fligner.test(type ~ quality, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: type by quality
## Fligner-Killeen:med chi-squared = 116.53, df = 6, p-value < 2.2e-16

fligner.test(quality ~ type, data=data.new.pca)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data: quality by type
## Fligner-Killeen:med chi-squared = 0.61775, df = 1, p-value = 0.4319

```

Teniendo en cuenta la información presentada gracias al test de Fligner-Killen, podemos destacar que las variables PC4 y quality y quality y type presentan homocedasticidad o igualdad de varianzas. Esto es debido a que el valor del estadístico p, resulta superior a 0.05.

El resto de variables se comportan de forma que sus valores presentan heterocedasticidad.

## 9. Aplicación de pruebas estadísticas

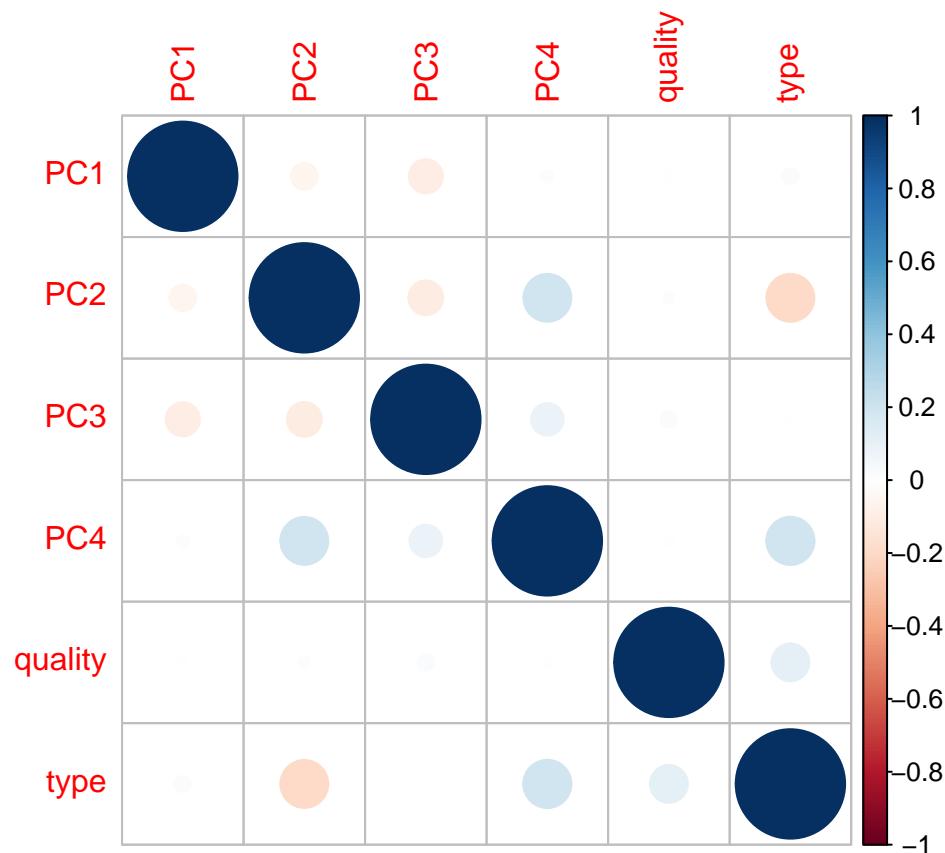
### 9.1. Correlación entre los distintos atributos ya reducidos.

A continuación, vamos a mostrar la matriz de correlación de Pearson de aquellos atributos que conforma el juego de datos ya reducido. Esta acción se realizará a través del paquete corrplot.

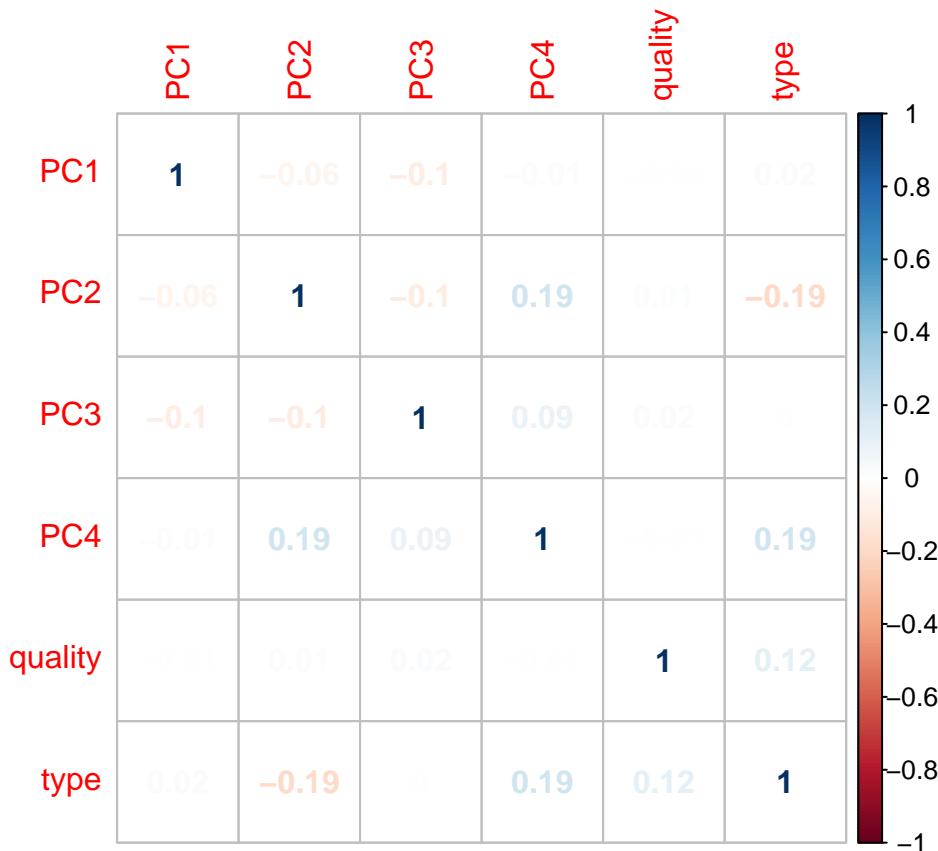
```

library(corrplot)
mat = cor(data.new.pca, method = "pearson")
corrplot(mat, method="circle")

```



```
corrplot(mat, method="number")
```



mat

```
##          PC1        PC2        PC3        PC4      quality
## PC1 1.000000000 -0.05978436 -0.097793427 -0.011896162 -0.007864048
## PC2 -0.059784357 1.000000000 -0.101166286 0.191895011 0.010939404
## PC3 -0.097793427 -0.10116629 1.000000000 0.088752386 0.022404033
## PC4 -0.011896162 0.19189501 0.088752386 1.000000000 -0.009716331
## quality -0.007864048 0.01093940 0.022404033 -0.009716331 1.000000000
## type    0.021806813 -0.19175720 -0.004058503 0.194306894 0.119323285
##          type
## PC1    0.021806813
## PC2   -0.191757205
## PC3   -0.004058503
## PC4    0.194306894
## quality 0.119323285
## type    1.000000000
```

Como podemos observar, las relaciones existentes entre los distintos atributos es baja.

## 9.2. Clasificación a través de algoritmo basado en árboles de decisión

En este apartado vamos a comprobar la capacidad de nuestros datos para distinguir entre un tipo de vino u otro a través de un algoritmo de clasificación basado en árboles de decisión.

En primer lugar, vamos a preparar los datos entre datos de entrenamiento y de prueba.

Para implementar un algoritmo basado en árboles de decisión, vamos a reordenar los atributos de forma aleatoria. Y vamos a distribuir el juego de datos tanto en datos de entrenamiento (2/3) y en datos de prueba o test (1/3).

Recordamos que tenemos disponibles 6497 observaciones, lo cuál para el conjunto de entrenamiento tendríamos 4332 observaciones y para el conjunto de prueba tendríamos 2165 observaciones.

```
set.seed(12345)
Dataset_order = data.new.pca[sample(nrow(data.new.pca)),]

Y = Dataset_order[,6]
X = Dataset_order[,1:5]

trainX = X[1:4332,]
trainY = Y[1:4332]
testX = X[4333:6497,]
testY = Y[4333:6497]
```

Tras preparar el conjunto de datos, aplicaremos el algoritmo basado en la función C5.0

```
model = C50::C5.0(trainX, as.factor(trainY), rules=TRUE)
summary(model)

##
## Call:
## C5.0.default(x = trainX, y = as.factor(trainY), rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Sat Jan 02 18:08:25 2021
## -----
##
## Class specified by attribute 'outcome'
##
## Read 4332 cases (6 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (38, lift 4.0)
##   PC2 > -38.95816
##   PC2 <= -23.74493
##   PC3 > 2.711789
##   PC4 > -2.549877
##   PC4 <= 32.58266
##   -> class 1 [0.975]
##
## Rule 2: (55/2, lift 3.9)
##   PC2 <= -23.74493
##   PC3 > 2.711789
##   PC4 > -2.549877
##   PC4 <= 32.58266
##   quality <= 4
##   -> class 1 [0.947]
##
## Rule 3: (735/40, lift 3.9)
```

```

##  PC1 > -22.95306
##  PC1 <= 17.37921
##  PC2 > -23.74493
##  PC2 <= 20.8844
##  PC3 <= 26.99381
##  PC4 <= 28.6102
##  ->  class 1  [0.944]
##
## Rule 4: (341/20, lift 3.8)
##  PC1 > -22.95306
##  PC2 > -14.16021
##  PC2 <= 20.8844
##  PC3 > -37.64994
##  PC4 <= 28.6102
##  quality > 3
##  ->  class 1  [0.939]
##
## Rule 5: (24/1, lift 3.8)
##  PC1 > -22.07363
##  PC1 <= -10.96624
##  PC2 <= -23.74493
##  PC4 <= 32.58266
##  ->  class 1  [0.923]
##
## Rule 6: (324/40, lift 3.6)
##  PC2 > -23.74493
##  PC2 <= 20.8844
##  PC4 <= 28.6102
##  quality > 2
##  quality <= 3
##  ->  class 1  [0.874]
##
## Rule 7: (12/1, lift 3.5)
##  PC1 > -30.41101
##  PC1 <= 33.06821
##  PC2 > -25.21
##  PC2 <= -23.74493
##  PC4 > -2.549877
##  quality <= 3
##  ->  class 1  [0.857]
##
## Rule 8: (12/2, lift 3.2)
##  PC1 <= -22.95306
##  PC3 <= -7.258634
##  PC4 > 21.0079
##  PC4 <= 27.23364
##  quality > 2
##  quality <= 3
##  ->  class 1  [0.786]
##
## Rule 9: (786/209, lift 3.0)
##  PC1 > -37.21238
##  PC2 > -23.74493
##  PC3 > -6.774142

```

```

##  PC3 <= 29.48739
##  PC4 <= 56.81179
##  ->  class 1  [0.734]
##
## Rule 10: (134/1, lift 1.3)
##  PC1 > -37.21238
##  PC2 > -23.74493
##  PC4 > 56.81179
##  ->  class 2  [0.985]
##
## Rule 11: (352/5, lift 1.3)
##  PC3 <= -37.64994
##  quality > 3
##  ->  class 2  [0.983]
##
## Rule 12: (110/1, lift 1.3)
##  PC2 > -23.74493
##  PC3 > 26.99381
##  PC4 <= 28.6102
##  quality > 3
##  ->  class 2  [0.982]
##
## Rule 13: (158/2, lift 1.3)
##  PC1 > -30.41101
##  PC2 <= -38.95816
##  quality > 4
##  ->  class 2  [0.981]
##
## Rule 14: (826/17, lift 1.3)
##  PC1 > 33.06821
##  PC2 <= -23.74493
##  ->  class 2  [0.978]
##
## Rule 15: (407/11, lift 1.3)
##  PC1 <= -22.07363
##  PC2 <= -23.74493
##  PC4 > -2.549877
##  ->  class 2  [0.971]
##
## Rule 16: (468/14, lift 1.3)
##  PC1 > 17.37921
##  PC2 <= -14.16021
##  PC4 <= 28.6102
##  quality > 3
##  ->  class 2  [0.968]
##
## Rule 17: (894/30, lift 1.3)
##  PC2 <= -25.21
##  PC4 > 32.58266
##  ->  class 2  [0.965]
##
## Rule 18: (822/32, lift 1.3)
##  PC3 <= 20.22863
##  PC4 > 41.97989

```

```

##  -> class 2 [0.960]
##
## Rule 19: (760/31, lift 1.3)
## PC1 > -10.96624
## PC2 <= -26.17937
## PC3 <= 2.711789
##  -> class 2 [0.958]
##
## Rule 20: (1103/55, lift 1.3)
## PC2 <= -23.74493
## PC4 <= -2.549877
##  -> class 2 [0.949]
##
## Rule 21: (1092/56, lift 1.3)
## PC3 <= -6.774142
## PC4 > 28.6102
##  -> class 2 [0.948]
##
## Rule 22: (1164/65, lift 1.2)
## PC1 <= -22.95306
##  -> class 2 [0.943]
##
## Rule 23: (966/57, lift 1.2)
## PC3 > 29.48739
##  -> class 2 [0.940]
##
## Rule 24: (360/24, lift 1.2)
## PC2 > 20.8844
##  -> class 2 [0.931]
##
## Default class: 2
##
##
## Evaluation on training data (4332 cases):
##
##          Rules
##          -----
##          No      Errors
##          --
##          24    286( 6.6%)   <<
##          --
##          (a)    (b)    <-classified as
##          --  --
##          853    208    (a): class 1
##          78     3193   (b): class 2
##          --
##          --
##          Attribute usage:
##          --
##          93.47% PC4
##          85.96% PC3
##          84.90% PC1
##          83.73% PC2

```

```

##    36.59% quality
##
## Time: 0.1 secs

```

Atendiendo a los errores, nuestro modelo tiene una tasa de error del 6.6% que se trata de 286 observaciones. Cabe destacar que estas observaciones son mayoritarias en las clases de vinos blancos que en la de rojos.

A la hora de entrenar el modelo para la futura realización de predicciones se puede encontrar que la contribución de la variable PC4 ha sido de un 93.47%, la de PC3 un 85.96%, la de buying un 84.90%, etc.

Se han generado 24 reglas de decisión, donde podemos destacar las siguientes:

- Regla 1. Los vinos cuyos valores del componente 2 se encuentre entre -38.95816 (no incluido) y -23.74493 (incluido), el valor del componente 3 es superior a 2.711789 y el valor de la componente 4 se encuentre entre -2.549877(no incluido) y 32.58266(incluido) , son considerados como vinos rojos con una validez del 97.5%.
- Regla 10. Los vinos cuyos valores del componente 1 es superior a -37.21238, el valor del componente 2 es superior a -23.744493 y el valor de la componente 4 es superior a 56.811179, son considerados como vinos blancos con una validez del 98.5%.
- Regla 11. Los vinos cuyos valores del componente 3 es inferior o igual a -37.64994 y el valor del grado de calidad sea superior a 5, son considerados como vinos blancos con una validez del 98.3%.
- Regla 12. Los vinos cuyos valores del componente 2 es superior a -23.74493, el valor del componente 3 es superior a 26.99381, el valor de la componente 4 es inferior o igual a 28.6102 y el grado de calidad es superior a 5, son considerados como vinos blancos con una validez del 98.2%.
- Regla 13. Los vinos cuyos valores del componente 1 es superior a -30.41101, el valor del componente 2 sea inferior o igual a -38.95816 y el valor del grado de calidad sea superior a 6, son considerados como vinos blancos con una validez del 98.1%.

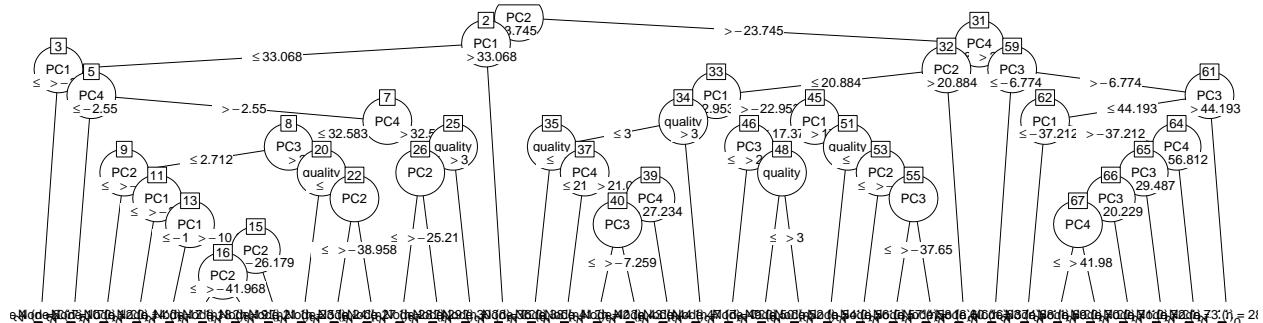
Estos son algunos ejemplos de interpretación de las reglas que ha generado nuestro árbol de decisión.

A continuación mostraremos el árbol de decisión generado y lo almacenaremos en PDF para su mejor visualización.

```

model = C50::C5.0(trainX, as.factor(trainY))
plot(model)

```



```

pdf("modelC50.pdf", width=30, height = 30)
plot(model)
dev.off()

```

```

## pdf
## 2

A continuación, vamos a probar nuestro modelo entrenado con el conjunto de datos de prueba o test. Para ello, cargaremos la librería caret con el fin de generar la matriz de confusión.

library(caret)

## Warning: package 'caret' was built under R version 4.0.3

## Loading required package: lattice

## Loading required package: ggplot2

predict_model = predict(model, testX, type="class")
confusionMatrix(predict_model,as.factor(testY),positive="2")

## Confusion Matrix and Statistics
##
##          Reference
## Prediction    1     2
##           1 419   53
##           2 119 1574
##
##          Accuracy : 0.9206
##                 95% CI : (0.9084, 0.9316)
##     No Information Rate : 0.7515
##     P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.7782
##
##  Mcnemar's Test P-Value : 7.188e-07
##
##          Sensitivity : 0.9674
##          Specificity : 0.7788
##     Pos Pred Value : 0.9297
##     Neg Pred Value : 0.8877
##          Prevalence : 0.7515
##     Detection Rate : 0.7270
## Detection Prevalence : 0.7820
##     Balanced Accuracy : 0.8731
##
##     'Positive' Class : 2
##

```

De esta matriz de confusión se pueden obtener las siguientes conclusiones:

Nuestro modelo es capaz de clasificar correctamente 1993 vinos del total que ha sido 2165. Por lo cual, clasifica adecuadamente un 92.06% de los vinos, tratándose del valor de la precisión del modelo. Y tiene una tasa de errores del 7.94%, donde se encuentran los 172 vinos restantes. Podemos determinar que el ajuste proporcionado por el modelo es bastante bueno debido a que su tasa de acierto es bastante elevada y su tasa de error es bastante baja. Cabe destacar que esto es teniendo en cuenta que hemos empleado los distintos

componentes resultantes del PCA que ofrecen un peor resultado en términos generales que el uso de todos los atributos del conjunto de datos.

De los vinos que se consideran rojos realmente, se han clasificado incorrectamente 119 vinos considerándolos blancos, tratándose del 22.11%.

De los vinos que se consideran blancos realmente, se han clasificado incorrectamente 53 vinos considerándolos rojos, tratándose del 1.41%.

Podemos observar que el mayor porcentaje de error aparece en la clasificación de vehículos que realmente son realmente rojos y considerados como blancos. Esto puede ser debido a que el conjunto de datos no se encuentra equilibrado. Esto es debido a que existen un mayor número de vinos blancos que rojos.

A continuación, vamos a mostrar el desempeño de nuestro modelo a través de la curva ROC mediante la librería pROC.

```
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

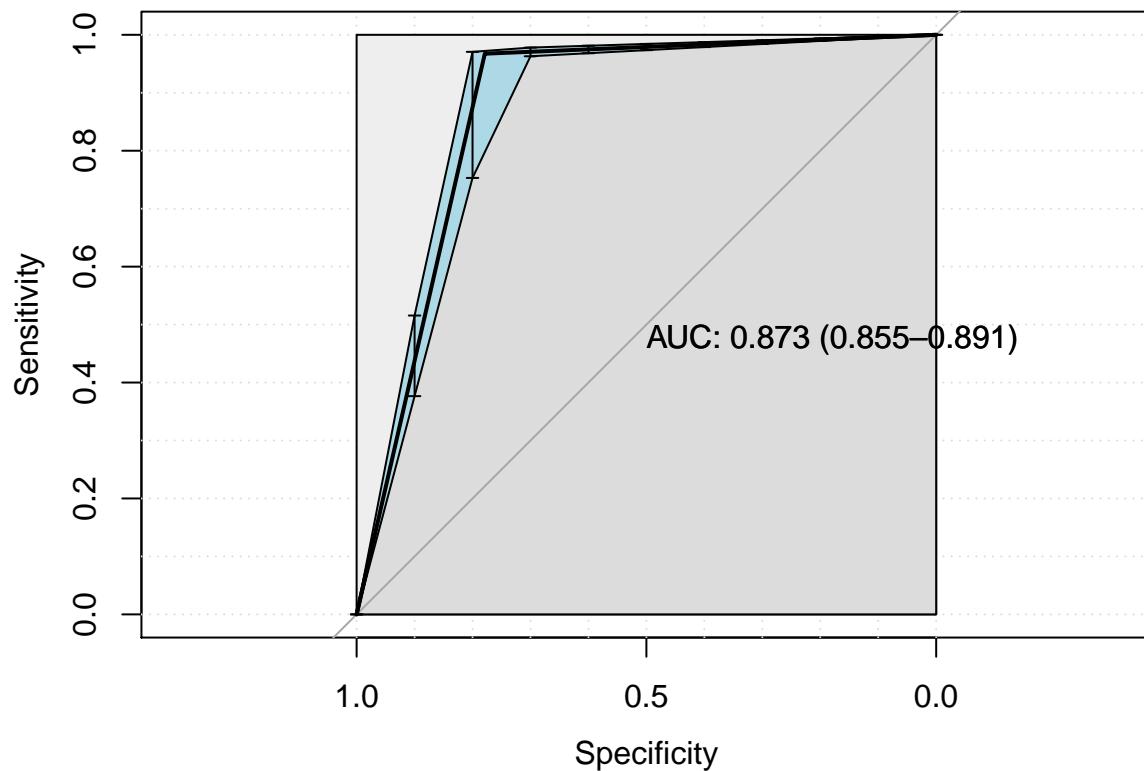
## The following objects are masked from 'package:stats':
##      cov, smooth, var

curve <- roc(testY,as.numeric(predict_model),
             smoothed = TRUE,
             # arguments for ci
             ci=TRUE, ci.alpha=0.95, stratified=FALSE,
             # arguments for plot
             plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
             print.auc=TRUE, show.thres=TRUE)

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

curve2 <- ci.se(curve)
plot(curve2, type="shape", col="lightblue")
plot(curve2, type="bars")
```



Teniendo en cuenta la información mostrada, podemos destacar que nuestro modelo ofrece un rendimiento aceptable debido a que la forma de la curva ROC es similar a la que tiene que producirse en el caso de clasificación perfecto. Además, el valor del área bajo la curva ROC es cercano a 1, por lo que manifiesta el gran desempeño de nuestro modelo para la clasificación.

### 9.3. Clasificación a través de modelos de regresión

#### 9.3.1. Regresión lineal

A continuación, vamos a mostrar el desempeño de nuestro juego de datos reducido a través de algoritmos basados en la regresión lineal.

```

modelo1 = lm(type ~ PC1 ,data=data.new.pca)
modelo2 = lm(type ~ PC2 ,data=data.new.pca)
modelo3 = lm(type ~ PC3 ,data=data.new.pca)
modelo4 = lm(type ~ PC4 ,data=data.new.pca)
modelo5 = lm(type ~ quality ,data=data.new.pca)
modelo6 = lm(type ~ PC1+PC2 ,data=data.new.pca)
modelo7 = lm(type ~ PC1+PC2+PC3 ,data=data.new.pca)
modelo8 = lm(type ~ PC1+PC2+PC3+PC4 ,data=data.new.pca)
modelo9 = lm(type ~ PC1+PC2+PC3+PC4+quality ,data=data.new.pca)

summary(modelo1)

```

```
##
```

```

## Call:
## lm(formula = type ~ PC1, data = data.new.pca)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.7749  0.2280  0.2389  0.2531  0.2985
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.7532004  0.0053578 327.226 <2e-16 ***
## PC1         0.0002513  0.0001429   1.758  0.0788 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4307 on 6495 degrees of freedom
## Multiple R-squared:  0.0004755, Adjusted R-squared:  0.0003216
## F-statistic:  3.09 on 1 and 6495 DF, p-value: 0.07882

```

```
summary(modelo2)
```

```

##
## Call:
## lm(formula = type ~ PC2, data = data.new.pca)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91578 -0.00974  0.21209  0.25507  0.51506
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.6972867  0.0063590 266.91 <2e-16 ***
## PC2        -0.0022335  0.0001418  -15.75 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4228 on 6495 degrees of freedom
## Multiple R-squared:  0.03677, Adjusted R-squared:  0.03662
## F-statistic: 247.9 on 1 and 6495 DF, p-value: < 2.2e-16

```

```
summary(modelo3)
```

```

##
## Call:
## lm(formula = type ~ PC3, data = data.new.pca)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.7570  0.2413  0.2453  0.2470  0.2559
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.754e+00  5.345e-03 328.151 <2e-16 ***
## PC3        -4.399e-05  1.345e-04  -0.327    0.744

```

```

## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4308 on 6495 degrees of freedom
## Multiple R-squared: 1.647e-05, Adjusted R-squared: -0.0001375
## F-statistic: 0.107 on 1 and 6495 DF, p-value: 0.7436

```

```
summary(modelo4)
```

```

##
## Call:
## lm(formula = type ~ PC4, data = data.new.pca)
##
## Residuals:
##       Min     1Q   Median     3Q    Max
## -0.93087 -0.03864  0.19760  0.31054  0.40927
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.6956279  0.0063880 265.44 <2e-16 ***
## PC4         0.0022788  0.0001427  15.96 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4226 on 6495 degrees of freedom
## Multiple R-squared: 0.03776, Adjusted R-squared: 0.03761
## F-statistic: 254.8 on 1 and 6495 DF, p-value: < 2.2e-16

```

```
summary(modelo5)
```

```

##
## Call:
## lm(formula = type ~ quality, data = data.new.pca)
##
## Residuals:
##       Min     1Q   Median     3Q    Max
## -0.8823  0.1177  0.2354  0.2943  0.4120
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.529127  0.023804 64.237 <2e-16 ***
## quality     0.058862  0.006077  9.686 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4277 on 6495 degrees of freedom
## Multiple R-squared: 0.01424, Adjusted R-squared: 0.01409
## F-statistic: 93.81 on 1 and 6495 DF, p-value: < 2.2e-16

```

```
summary(modelo6)
```

```
##
```

```

## Call:
## lm(formula = type ~ PC1 + PC2, data = data.new.pca)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91697 -0.01049  0.21023  0.25767  0.51280
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.6971433  0.0063614 266.789 <2e-16 ***
## PC1         0.0001196  0.0001406   0.851   0.395    
## PC2        -0.0022263  0.0001421 -15.667 <2e-16 ***  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4228 on 6494 degrees of freedom
## Multiple R-squared:  0.03688,    Adjusted R-squared:  0.03658 
## F-statistic: 124.3 on 2 and 6494 DF,  p-value: < 2.2e-16

```

```
summary(modelo7)
```

```

## 
## Call:
## lm(formula = type ~ PC1 + PC2 + PC3, data = data.new.pca)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91940 -0.01703  0.21101  0.25995  0.53277
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.697e+00  6.369e-03 266.361 <2e-16 ***
## PC1         9.215e-05  1.413e-04   0.652   0.5144    
## PC2        -2.255e-03  1.429e-04 -15.779 <2e-16 ***  
## PC3        -2.478e-04  1.334e-04  -1.858   0.0633 .  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4227 on 6493 degrees of freedom
## Multiple R-squared:  0.03739,    Adjusted R-squared:  0.03695 
## F-statistic: 84.07 on 3 and 6493 DF,  p-value: < 2.2e-16

```

```
summary(modelo8)
```

```

## 
## Call:
## lm(formula = type ~ PC1 + PC2 + PC3 + PC4, data = data.new.pca)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.1266 -0.1676  0.1752  0.2768  0.6390
##
## Coefficients:

```

```

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.608e+00 7.557e-03 212.822 < 2e-16 ***
## PC1         6.078e-05 1.371e-04   0.443    0.657
## PC2        -2.838e-03 1.415e-04 -20.048 < 2e-16 ***
## PC3        -5.417e-04 1.302e-04  -4.162  3.2e-05 ***
## PC4         2.880e-03 1.420e-04   20.281 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.41 on 6492 degrees of freedom
## Multiple R-squared:  0.09475, Adjusted R-squared:  0.09419
## F-statistic: 169.9 on 4 and 6492 DF, p-value: < 2.2e-16

```

```
summary(modelo9)
```

```

##
## Call:
## lm(formula = type ~ PC1 + PC2 + PC3 + PC4 + quality, data = data.new.pca)
##
## Residuals:
##      Min       1Q     Median       3Q      Max
## -1.2039 -0.1716  0.1599  0.2738  0.6949
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.370e+00 2.340e-02 58.560 < 2e-16 ***
## PC1         6.751e-05 1.359e-04   0.497    0.619
## PC2        -2.861e-03 1.403e-04 -20.388 < 2e-16 ***
## PC3        -5.757e-04 1.291e-04  -4.460 8.34e-06 ***
## PC4         2.902e-03 1.408e-04   20.614 < 2e-16 ***
## quality     6.198e-02 5.777e-03  10.729 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4064 on 6491 degrees of freedom
## Multiple R-squared:  0.1105, Adjusted R-squared:  0.1098
## F-statistic: 161.3 on 5 and 6491 DF, p-value: < 2.2e-16

```

Observando los valores para la métrica R cuadrado ajustado, podemos contemplar que ninguno de los modelos elaborados son suficientes para realizar una clasificación aceptable debido a que el valor de dicha métrica es muy bajo.

### 9.3.2. Regresión logística

A continuación, vamos a mostrar el desempeño de nuestro juego de datos reducido a través de algoritmos basados en la regresión logística.

```

data.new.pca$type = replace(data.new.pca$type, data.new.pca$type==2,0)
m1 = glm(type ~ PC1,data=data.new.pca, family="binomial")
m2 = glm(type ~ PC2,data=data.new.pca, family="binomial")
m3 = glm(type ~ PC3,data=data.new.pca, family="binomial")
m4 = glm(type ~ PC4,data=data.new.pca, family="binomial")

```

```

m5 = glm(type ~ quality, data=data.new.pca, family="binomial")
m6 = glm(type ~ PC1+PC2, data=data.new.pca, family="binomial")
m7 = glm(type ~ PC1+PC2+PC3, data=data.new.pca, family="binomial")
m8 = glm(type ~ PC1+PC2+PC3+PC4, data=data.new.pca, family="binomial")
m9 = glm(type ~ PC1+PC2+PC3+PC4+quality, data=data.new.pca, family="binomial")
summary(m1)

```

```

##
## Call:
## glm(formula = type ~ PC1, family = "binomial", data = data.new.pca)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.8475 -0.7639 -0.7389 -0.7202  1.7254
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.116412  0.028841 -38.709 <2e-16 ***
## PC1        -0.001348  0.000767 -1.757  0.0789 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 7251.0 on 6496 degrees of freedom
## Residual deviance: 7247.9 on 6495 degrees of freedom
## AIC: 7251.9
##
## Number of Fisher Scoring iterations: 4

```

```
summary(m2)
```

```

##
## Call:
## glm(formula = type ~ PC2, family = "binomial", data = data.new.pca)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.3410 -0.7507 -0.6728 -0.3696  2.1083
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.8477534  0.0328404 -25.81 <2e-16 ***
## PC2         0.0128802  0.0008505  15.14 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 7251.0 on 6496 degrees of freedom
## Residual deviance: 7002.9 on 6495 degrees of freedom
## AIC: 7006.9

```

```

## 
## Number of Fisher Scoring iterations: 4

summary(m3)

## 
## Call:
## glm(formula = type ~ PC3, family = "binomial", data = data.new.pca)
## 
## Deviance Residuals:
##      Min     1Q Median     3Q    Max 
## -0.7691 -0.7533 -0.7503 -0.7432  1.6820 
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -1.1194890  0.0288029 -38.867 <2e-16 ***
## PC3          0.0002369  0.0007243   0.327   0.744  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
## Null deviance: 7251.0  on 6496  degrees of freedom
## Residual deviance: 7250.9  on 6495  degrees of freedom
## AIC: 7254.9
## 
## Number of Fisher Scoring iterations: 4

```

```

summary(m4)

## 
## Call:
## glm(formula = type ~ PC4, family = "binomial", data = data.new.pca)
## 
## Deviance Residuals:
##      Min     1Q Median     3Q    Max 
## -1.1139 -0.8688 -0.6350 -0.3113  2.1959 
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -0.8191245  0.0331258 -24.73 <2e-16 ***
## PC4         -0.0145084  0.0009474  -15.31 <2e-16 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
## Null deviance: 7251.0  on 6496  degrees of freedom
## Residual deviance: 6979.6  on 6495  degrees of freedom
## AIC: 6983.6
## 
## Number of Fisher Scoring iterations: 4

```

```
summary(m5)
```

```
##  
## Call:  
## glm(formula = type ~ quality, family = "binomial", data = data.new.pca)  
##  
## Deviance Residuals:  
##      Min       1Q   Median       3Q      Max  
## -1.0857  -0.8356  -0.7259  -0.5399   1.9990  
##  
## Coefficients:  
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept)  0.10695   0.12971   0.825    0.41  
## quality     -0.32652   0.03421  -9.544  <2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## (Dispersion parameter for binomial family taken to be 1)  
##  
## Null deviance: 7251.0 on 6496 degrees of freedom  
## Residual deviance: 7156.9 on 6495 degrees of freedom  
## AIC: 7160.9  
##  
## Number of Fisher Scoring iterations: 4
```

```
summary(m6)
```

```
##  
## Call:  
## glm(formula = type ~ PC1 + PC2, family = "binomial", data = data.new.pca)  
##  
## Deviance Residuals:  
##      Min       1Q   Median       3Q      Max  
## -1.3350  -0.7565  -0.6686  -0.3679   2.1127  
##  
## Coefficients:  
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -0.8467014  0.0328461 -25.778  <2e-16 ***  
## PC1         -0.0008408  0.0008017  -1.049    0.294  
## PC2          0.0128513  0.0008515  15.093  <2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## (Dispersion parameter for binomial family taken to be 1)  
##  
## Null deviance: 7251.0 on 6496 degrees of freedom  
## Residual deviance: 7001.8 on 6494 degrees of freedom  
## AIC: 7007.8  
##  
## Number of Fisher Scoring iterations: 4
```

```
summary(m7)
```

```
##  
## Call:  
## glm(formula = type ~ PC1 + PC2 + PC3, family = "binomial", data = data.new.pca)  
##  
## Deviance Residuals:  
##      Min       1Q   Median       3Q      Max  
## -1.3809  -0.7592  -0.6705  -0.3622   2.1171  
##  
## Coefficients:  
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -0.8437575  0.0329440 -25.612 <2e-16 ***  
## PC1        -0.0005894  0.0008177  -0.721  0.4711  
## PC2         0.0129928  0.0008575  15.153 <2e-16 ***  
## PC3         0.0012767  0.0007584   1.683  0.0923 .  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## (Dispersion parameter for binomial family taken to be 1)  
##  
## Null deviance: 7251  on 6496  degrees of freedom  
## Residual deviance: 6999  on 6493  degrees of freedom  
## AIC: 7007  
##  
## Number of Fisher Scoring iterations: 4
```

```
summary(m8)
```

```
##  
## Call:  
## glm(formula = type ~ PC1 + PC2 + PC3 + PC4, family = "binomial",  
##       data = data.new.pca)  
##  
## Deviance Residuals:  
##      Min       1Q   Median       3Q      Max  
## -1.5897  -0.7783  -0.5597  -0.1401   2.8435  
##  
## Coefficients:  
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -0.3071234  0.0417761 -7.352 1.96e-13 ***  
## PC1        -0.0024896  0.0009249  -2.692  0.00711 **  
## PC2         0.0169158  0.0008760  19.311 < 2e-16 ***  
## PC3         0.0016651  0.0008194   2.032  0.04215 *  
## PC4        -0.0223329  0.0012111 -18.440 < 2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## (Dispersion parameter for binomial family taken to be 1)  
##  
## Null deviance: 7251.0  on 6496  degrees of freedom  
## Residual deviance: 6550.6  on 6492  degrees of freedom
```

```

## AIC: 6560.6
##
## Number of Fisher Scoring iterations: 5

summary(m9)

##
## Call:
## glm(formula = type ~ PC1 + PC2 + PC3 + PC4 + quality, family = "binomial",
##      data = data.new.pca)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.7341 -0.7666 -0.5332 -0.1351  3.0106
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 1.0844149 0.1427733 7.595 3.07e-14 ***
## PC1        -0.0025725 0.0009269 -2.775 0.00552 **
## PC2         0.0171841 0.0008835 19.450 < 2e-16 ***
## PC3         0.0018326 0.0008226  2.228 0.02589 *
## PC4        -0.0225146 0.0012156 -18.521 < 2e-16 ***
## quality     -0.3675564 0.0362833 -10.130 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 7251.0 on 6496 degrees of freedom
## Residual deviance: 6444.1 on 6491 degrees of freedom
## AIC: 6456.1
##
## Number of Fisher Scoring iterations: 5

```

Visualizando los distintos AIC (Criterio de Información de Akaike), podemos contemplar que el modelo 9 es aquel que presenta un mejor rendimiento.

## Conclusiones

Las tareas que se han realizado en esta práctica se han centrado en el análisis de los datos que conforman el juego de datos basado en los distintos tipos de vino. Esto conlleva la realización de un análisis descriptivo del juego de datos, la detección y tratamiento de valores vacíos y outliers, la discretización de atributos, un análisis de correlación entre las distintas variables y la reducción de dimensionalidad a través del método PCA.

Tras obtener el nuevo conjunto de datos ya reducido, hemos propuesto realizar una serie de acciones:

- En primer lugar, realizamos un análisis de correlación entre las distintas variables.
- En segundo lugar, elaboramos la clasificación de las observaciones en vinos rojos y blancos a través del algoritmo C5.0 basado en árboles de decisión.

- En tercer lugar, procedimos a comprobar la capacidad predictiva de nuestro juego de datos a través de regresiones lineales y mediante regresiones logísticas.

El objetivo principal del proyecto se ha cumplido, que se trató de crear un clasificador entre vinos rojos y blancos ateniendo a sus características. Se ha podido responder al problema debido a que gracias al algoritmo basado en árboles de decisión se ha obtenido un clasificador con un 92.06% de exactitud en sus predicciones. Cabe destacar que el rendimiento del algoritmo se ha basado en el dataset con los atributos reducidos, que a priori, ofrecen un peor desempeño que el conjunto de datos total.

La realización de esta práctica me ha supuesto un gran reto debido a que se trata de la aplicación de la mayoría de los conocimientos de la asignatura en una práctica. Cabe destacar que se podrían realizar mayores mejoras para futuras versiones del proyecto, como son:

- Aplicación de algoritmos de aprendizaje no supervisado.
- Aplicación de otros algoritmos basados en aprendizaje supervisado como por ejemplo, redes neuronales o randomforest.
- Hacer uso de otras pruebas estadísticas para comprobar el rendimiento de los distintos atributos y modelos.
- Comparar el rendimiento de los modelos utilizando el dataset con los componentes ya reducidos y el juego de datos original.
- Orientar la práctica de otra forma, intentando predecir el grado de calidad.

## Bibliografía

- Joaquín Amat Rodrigo, 2017. Recuperado de [https://rpubs.com/Joaquin\\_AR/287787](https://rpubs.com/Joaquin_AR/287787)
- Analytics Vidhya, 2016. Recuperado de <https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python/>
- Jake VanderPlas, 2016. Recuperado de <https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html>
- Linh Ngo, 2018. Recuperado de <https://blog.biöturing.com/2018/06/18/how-to-read-pca-biplots-and-scree-plots/>
- Hatcher & Stepansky, 1994. Recuperado de [https://www.researchgate.net/profile/Ehsan\\_Khedive/post/How\\_many\\_components\\_can\\_I\\_retrieve\\_in\\_principal\\_component\\_analysis/attachment/59d626f2c49f478072e9b1be/AS%3A272185124425729%401441905398541/download/Principal+Component+Analysis+SAS.pdf](https://www.researchgate.net/profile/Ehsan_Khedive/post/How_many_components_can_I_retrieve_in_principal_component_analysis/attachment/59d626f2c49f478072e9b1be/AS%3A272185124425729%401441905398541/download/Principal+Component+Analysis+SAS.pdf)
- Laia Subirats Maté, Diego Oswaldo Pérez Trenar & Mireia Calvo González, 2019. Recuperado de [http://materials.cv.uoc.edu/daisy/Materials/PID\\_00265704/pdf/PID\\_00265704.pdf](http://materials.cv.uoc.edu/daisy/Materials/PID_00265704/pdf/PID_00265704.pdf)
- Joseph Rickert, 2019. Recuperado de <https://rvviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/>
- Paulo Cortez, A.Cerdeira, F.Almeida,T.Matos & J.Reis, 2009. Recuperado de <https://archive.ics.uci.edu/ml/datasets/Wine+Quality>