Wine Data Analysis

Dr. Awala Fortune

2020-06-20

Contents

/pagebreak

0.0.1 I. Introduction

The Capstone project is part of the requirement for Data Science Professional Certification program hosted by HarvardX. The scope of this project covers application of basic understanding of R language, data visualization, probability and statistics, inference and modeling, application of productivity tools, data wrangling, linear regression and the application of machine learning in solving real life problems.

The University of California Irvine (UCI) machine learning repository is a center for machine learning and intelligent systems, it currently maintains 504 data sets as a service to the machine learning community for analysis of algorithms.

##1.1. General description of Dataset:##

This project uses the wine quality data set of white and red wines of vinho verde wine samples, from the north of Portugal. The aim of the research is to model wine quality based on physicochemical properties such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol using classification or regression model. The input is derived from the physicochemical test while the output is viewed as a sensory data such as quality.This outcomes are grouped in classes or categories for instance, quality has 10 classes, numbered from 0 to 9, and wine type has two classes, red and white.

0.0.2 II. Methods and Analysis

The research will apply several data science and machine learning classification techniques such as data cleaning, data exploration, visualization and modeling approach in Predicting wine quality and identifying the wine type, red or white in solving classification problems. The research therefore aimed at modeling wine quality based on physicochemical properties using machine learning classification techniques.

Loading of Libraries and dataset

```r
#load libraries
library("ggplot2")
library("dplyr")
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library("gridExtra")
```

```
## Warning: package 'gridExtra' was built under R version 3.6.3
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(Simpsons)
```

```
## Warning: package 'Simpsons' was built under R version 3.6.3
```

```
## Loading required package: mclust
```

```
## Warning: package 'mclust' was built under R version 3.6.3
```

```
## Package 'mclust' version 5.4.6
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 3.6.3

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(memisc)
```

```
## Warning: package 'memisc' was built under R version 3.6.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.3

## Loading required package: MASS

## Warning: package 'MASS' was built under R version 3.6.3

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

##
## Attaching package: 'memisc'

## The following objects are masked from 'package:dplyr':
##
##     collect, recode, rename, syms

## The following object is masked from 'package:ggplot2':
##
##     syms

## The following objects are masked from 'package:stats':
##
##     contr.sum, contr.treatment, contrasts

## The following object is masked from 'package:base':
##
##     as.array
```

```r
library(pander)
```

```
## Warning: package 'pander' was built under R version 3.6.3
```

```
##
## Attaching package: 'pander'
```

```
## The following object is masked from 'package:GGally':
##
##     wrap
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```r
library(RCurl)
```

```
## Warning: package 'RCurl' was built under R version 3.6.3
```

2.1. Data Preparation:

In this section, three files are downloaded from the UCI repository consisting of one csv file for each wine type and one file with the dataset description, the three files would be imported into Rstudio.

The column is added to each dataset to define the type of wine we want to predict with the factor variable type. This variable is used as the outcome that will be predicted.

Then, both datasets are combined in the wine dataset. ##2.2. Model Evaluation:##

The evaluation of machine learning algorithms has to do with comparing the predicted value with the actual outcome. The confusion matrix displays the predicted and observed values in a table and provides additional statistics summary.

##2.3. Statistical Terminology:##

#a. Confusion Matrix:# The confusion matrix is a simple table with the cross tabulation of the predicted values with the actual observed values. Statistical metrics can be calculated from the confusion matrix.

#b. Accuracy:# This the proportion of correct predictions for both positive and negative outcomes. High accuracy with a large difference in the number of positives and negatives becomes less meaningful, due to the fact that the algorithm loses the ability to predict the less common class. In this case, other metrics complements the analysis.

#c. Sensitivity:# This is the proportion of positive values when they are actually positive.

#d. Specificity:# The probability of a predicted negative value conditioned to a negative outcome.

#f. Prevalence:# This term shows how often the positive value appears in the sample. Low prevalence may lead to statistically incorrect conclusions. For the purpose of this research would use interchangeably with ubiquity.

#e. Precision:# This is the probability of an actual positive occurrence conditioned to a predicted positive result.Precision can be written as the proportion of positive values that are actually positive.

#g. Recall:# This is the same as sensitivity and is the probability of a predicted positive value conditioned to an actual positive value.

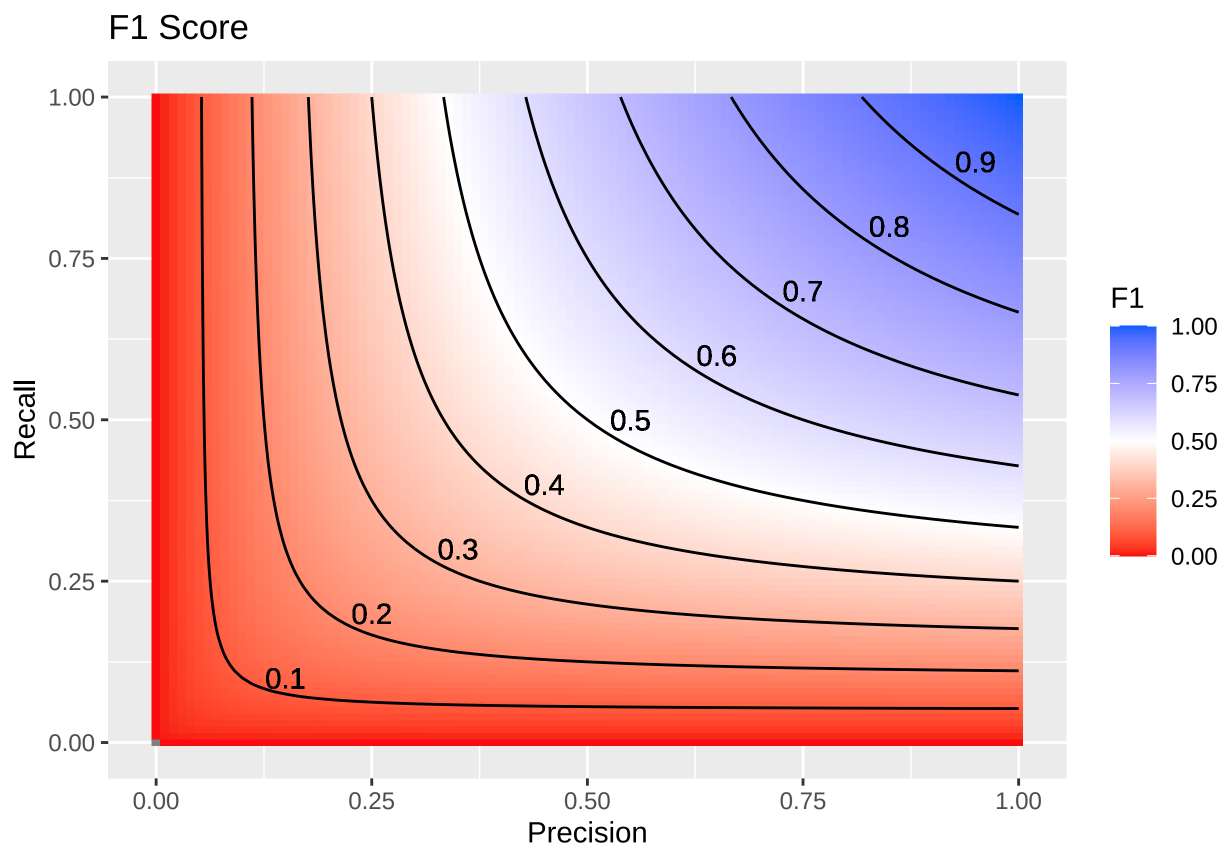
#h. ROC and AUC:# Receiver Operating Characteristic (ROC) curves are a popular way to visualize the tradeoffs between sensitivitiy and specificity in a binary classifier. Computing the area under the curve is one way to summarize it in a single value; this metric is so common that if data scientists say â€œarea under the curveâ€ or â€œAUCâ€, you can generally assume they mean an ROC curve unless otherwise specified. The area under the ROC curve (AUC) measures the probability that a model will rank a positive value higher than a negative one.AUC ranges from 0 to 1. A model with 100% wrong predictions has an AUC of 0, a model with 100% right predictions has AUC of 1 and a model with random prediction has an AUC of 0.5.

A guide for classifying the accuracy of a diagnostic test adopted from the traditional academic point system:

AUC Classification 0.90 - 1 excellent (A) 0.80 - 0.90 good (B) 0.70 - 0.80 fair (C) 0.60 - 0.70 poor (D) 0.50 - 0.60 fail (F)

#i. The F1 Score:# The F1 score is a measure of accuracy for classification problems with binary outcome. It is computed as the harmonic mean between precision and recall.

It is observed that precision and recall vary from 0 to 1, therefore a plot of both values with the F1 score variation in color gradient is ploted. The black lines indicate the same F1 score values for different precision and recall combinations.



```
## $tidyverse
## NULL
##
## $icesTAF
## NULL
##
## $readr
## NULL
##
## $lubridate
## NULL
##
## $caret
## NULL
```

The dataset is split into two parts, one for training and one for testing. The model building and tuning is done in the training set, and then we use the test set to predict new values and evaluate the results.

We create a pair of training and testing sets for each prediction problem. train_set and test_set will be used for identifying wine type, and train_set_r and test_set_r will be used for red wine quality.

###III.Results and Discussion###

##3.1. Data Exploration and Visualization:##

This section checks the dataset structure, look for possible problems in the data and provides a clear understanding of the data. The information acquired in this section may be used during the modeling phase.

Although UCI provides the dataset ready for analysis, itâ€™s good practice to check it for possible issues we may encounter in the future.

The number of empty values are counted to ascertain the NAs in the entire dataset in the training and testing sets. Empty values may cause calculation errors that can be avoided if we identify them in advance.

```r
#==========================================
# Data Explorations
#==========================================
# After importing the dataset, it's good practice to check the data.
# Here, we make some basic data checking.


# Check for empty values (NAs) in the dataset
sum(is.na(wine))
```

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

There are no empty values in the dataset.

```r
# Identification of near zero variance predictors
nearZeroVar(train_set[, xcol], saveMetrics = TRUE)
```

	freqRatio	percentUnique	zeroVar	nzv
fixed_acidity	1.11	1.78	FALSE	FALSE
volatile_acidity	1.06	3.15	FALSE	FALSE
citric_acid	1.11	1.52	FALSE	FALSE
residual_sugar	1.07	5.29	FALSE	FALSE
chlorides	1.05	3.57	FALSE	FALSE
free_sulfur_dioxide	1.03	2.24	FALSE	FALSE
total_sulfur_dioxide	1.16	4.72	FALSE	FALSE
density	1.02	16.64	FALSE	FALSE
pH	1.06	1.85	FALSE	FALSE
sulphates	1.13	1.88	FALSE	FALSE
alcohol	1.10	1.81	FALSE	FALSE
The dataset structure c	onfirms the	information prov	ided by UC	I and provides additional inform

```r
# Compactly Display the Structure of an Arbitrary R Object
str(train_set)
```

```
## tibble [5,847 x 13] (S3: tbl_df/tbl/data.frame)
##  $ fixed_acidity       : num [1:5847] 7.4 7.8 7.4 7.9 7.3 7.8 7.5 6.7 7.5 5.6 ...
##  $ volatile_acidity    : num [1:5847] 0.7 0.76 0.66 0.6 0.65 0.58 0.5 0.58 0.5 0.615
##  $ citric_acid         : num [1:5847] 0 0.04 0 0.06 0 0.02 0.36 0.08 0.36 0 ...
##  $ residual_sugar      : num [1:5847] 1.9 2.3 1.8 1.6 1.2 2 6.1 1.8 6.1 1.6 ...
##  $ chlorides           : num [1:5847] 0.076 0.092 0.075 0.069 0.065 0.073 0.071 0.097
##  $ free_sulfur_dioxide : num [1:5847] 11 15 13 15 15 9 17 15 17 16 ...
##  $ total_sulfur_dioxide: num [1:5847] 34 54 40 59 21 18 102 65 102 59 ...
##  $ density             : num [1:5847] 0.998 0.997 0.998 0.996 0.995 ...
##  $ pH                  : num [1:5847] 3.51 3.26 3.51 3.3 3.39 3.36 3.35 3.28 3.35 3.5
##  $ sulphates           : num [1:5847] 0.56 0.65 0.56 0.46 0.47 0.57 0.8 0.54 0.8 0.52
##  $ alcohol             : num [1:5847] 9.4 9.8 9.4 9.4 10 9.5 10.5 9.2 10.5 9.9 ...
##  $ quality             : Factor w/ 7 levels "Q3","Q4","Q5",..: 3 3 3 3 5 5 3 3 3 3 ..
##  $ type                : Factor w/ 2 levels "red","white": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
head(train_set)
```

fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sul
7.4	0.70	0.00	1.9	0.076	11	
7.8	0.76	0.04	2.3	0.092	15	
7.4	0.66	0.00	1.8	0.075	13	
7.9	0.60	0.06	1.6	0.069	15	
7.3	0.65	0.00	1.2	0.065	15	
7.8	0.58	0.02	2.0	0.073	9	

```r
tail(train_set)
```

fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sul
5.7	0.21	0.32	0.9	0.038	38	
6.5	0.23	0.38	1.3	0.032	29	
6.2	0.21	0.29	1.6	0.039	24	
6.6	0.32	0.36	8.0	0.047	57	
6.5	0.24	0.19	1.2	0.041	30	
6.0	0.21	0.38	0.8	0.020	22	

The statistical summary provides a clue of each variable distribution.The observation of median close to the mean may be an indication of normal distribution. The variables free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sulphates and alcohol seems to follow this rule that we will confirm later.

The relative numbers of red as 1439 and white as 4408 in the type variable give an idea of prevalence or ubiquity.

```
# Statistics summary
summary(train_set)
```

```
##  fixed_acidity   volatile_acidity  citric_acid    residual_sugar
##  Min.   : 3.80   Min.   :0.08     Min.   :0.000   Min.   : 0.6
##  1st Qu.: 6.40   1st Qu.:0.23     1st Qu.:0.250   1st Qu.: 1.8
##  Median : 7.00   Median :0.29     Median :0.310   Median : 3.0
##  Mean   : 7.21   Mean   :0.34     Mean   :0.319   Mean   : 5.5
##  3rd Qu.: 7.70   3rd Qu.:0.40     3rd Qu.:0.390   3rd Qu.: 8.1
##  Max.   :15.90   Max.   :1.58     Max.   :1.660   Max.   :65.8
##
##    chlorides     free_sulfur_dioxide total_sulfur_dioxide    density
##  Min.   :0.009   Min.   :  1.0       Min.   :  6          Min.   :0.987
##  1st Qu.:0.038   1st Qu.: 17.0       1st Qu.: 78          1st Qu.:0.992
##  Median :0.047   Median : 29.0       Median :118          Median :0.995
##  Mean   :0.056   Mean   : 30.5       Mean   :116          Mean   :0.995
##  3rd Qu.:0.065   3rd Qu.: 41.0       3rd Qu.:156          3rd Qu.:0.997
##  Max.   :0.611   Max.   :289.0       Max.   :440          Max.   :1.039
##
##        pH          sulphates        alcohol      quality      type
##  Min.   :2.72   Min.   :0.220   Min.   : 8.0   Q3:  28   red  :1439
##  1st Qu.:3.11   1st Qu.:0.430   1st Qu.: 9.5   Q4: 189   white:4408
##  Median :3.21   Median :0.510   Median :10.3   Q5:1932
##  Mean   :3.22   Mean   :0.531   Mean   :10.5   Q6:2564
##  3rd Qu.:3.32   3rd Qu.:0.600   3rd Qu.:11.3   Q7: 968
##  Max.   :4.01   Max.   :2.000   Max.   :14.9   Q8: 163
##                                                Q9:   3
```

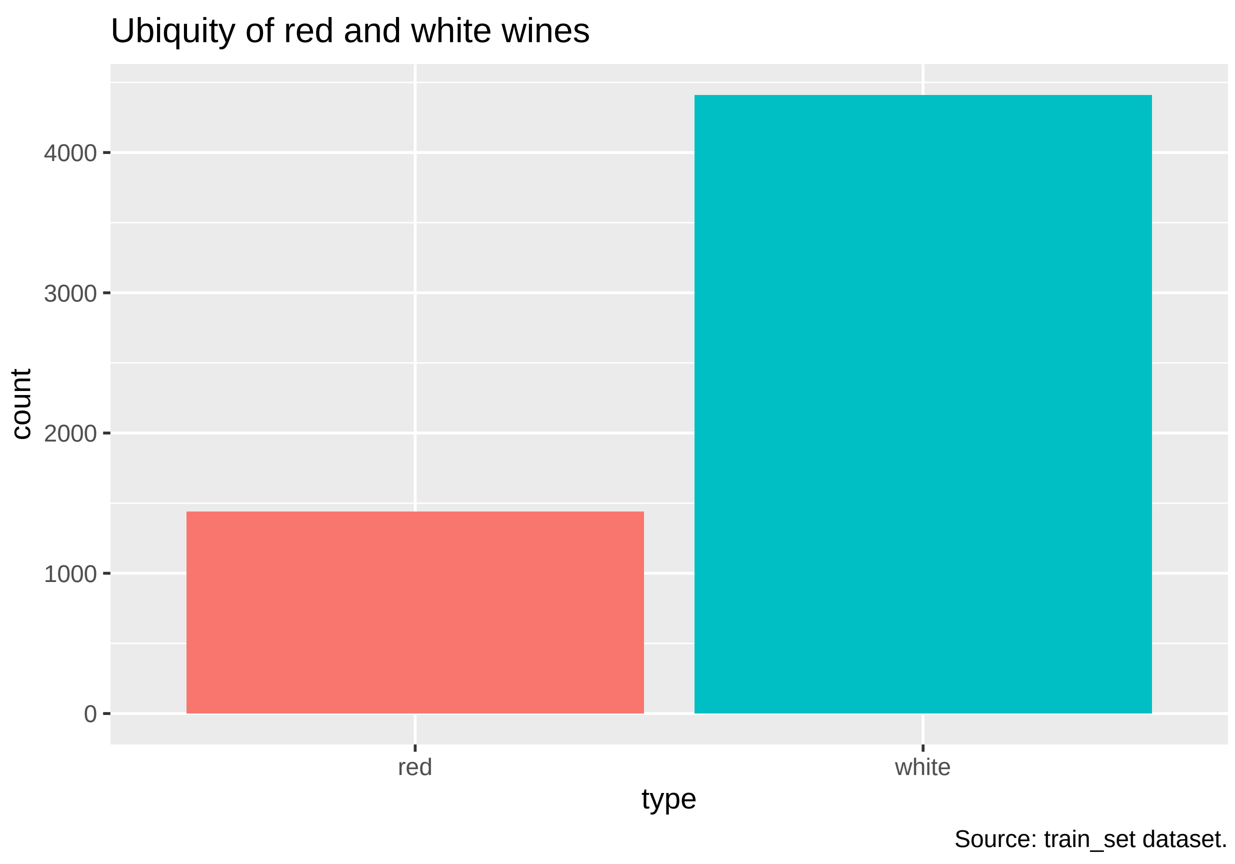
```
glimpse(train_set)
```

```
## Rows: 5,847
## Columns: 13
## $ fixed_acidity        <dbl> 7.4, 7.8, 7.4, 7.9, 7.3, 7.8, 7.5, 6.7, 7.5, 5...
## $ volatile_acidity     <dbl> 0.700, 0.760, 0.660, 0.600, 0.650, 0.580, 0.50...
## $ citric_acid          <dbl> 0.00, 0.04, 0.00, 0.06, 0.00, 0.02, 0.36, 0.08...
## $ residual_sugar       <dbl> 1.9, 2.3, 1.8, 1.6, 1.2, 2.0, 6.1, 1.8, 6.1, 1...
## $ chlorides            <dbl> 0.076, 0.092, 0.075, 0.069, 0.065, 0.073, 0.07...
## $ free_sulfur_dioxide  <dbl> 11, 15, 13, 15, 15, 9, 17, 15, 17, 16, 9, 52, ...
## $ total_sulfur_dioxide <dbl> 34, 54, 40, 59, 21, 18, 102, 65, 102, 59, 29, ...
## $ density              <dbl> 0.998, 0.997, 0.998, 0.996, 0.995, 0.997, 0.99...
## $ pH                   <dbl> 3.51, 3.26, 3.51, 3.30, 3.39, 3.36, 3.35, 3.28...
## $ sulphates            <dbl> 0.56, 0.65, 0.56, 0.46, 0.47, 0.57, 0.80, 0.54...
## $ alcohol              <dbl> 9.4, 9.8, 9.4, 9.4, 10.0, 9.5, 10.5, 9.2, 10.5...
## $ quality              <fct> Q5, Q5, Q5, Q5, Q7, Q7, Q5, Q5, Q5, Q5, Q5, Q5...
## $ type                 <fct> red, red, red, red, red, red, red, red, red, r...
```

##3.2. Red and White Wine Ubiquity or Prevalence:##

The aim is to identify and classify red and white wines, it is imperative to understand the distribution pattern of both types by reviewing the relative amount.

```r
# Distribution of red and white wines
ggplot(data = train_set) +
  geom_bar(aes(type, fill = type)) +
  labs(title = "Ubiquity of red and white wines",
       caption = "Source: train_set dataset.") +
  theme(legend.position = 'none')
```



The percentage of the prevalence of red wine in this dataset is 24.6% and that of the white wine is 75.4%.

In comparing the prevalence against the total production of red and white wines. The commission of vinho verde producers provides annual statistics of production6 for each wine type. The next step is to import the statistical data and calculate the prevalence of red wine production.

```
## $gridExtra
## NULL
##
## $ggridges
## NULL
##
## $ggplot2
## NULL
##
## $gtable
## NULL
##
## $grid
## NULL
##
## $egg
## NULL
```

```
## $readxl
## NULL
##
## $huxtable
## NULL
##
## $viridis
## NULL
##
## $ggthemes
## NULL
```

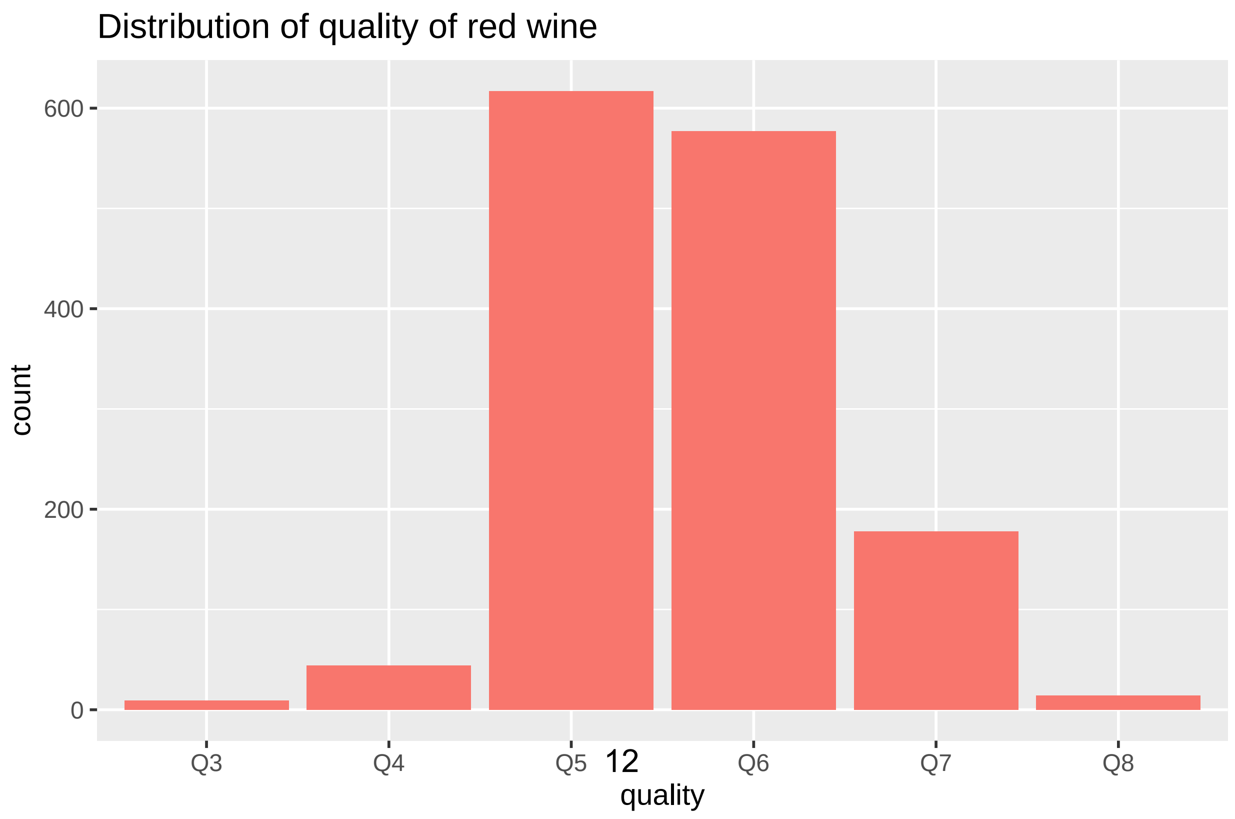
There is no information available about production by brand or grape variety but from the studies it is observed that the prevalence of the production of red wine is higher than that observed in the dataset.

##3.3. Quality distribution:##

For a better prediction of quality distribution of red wines there is the need to understand the distributional pattern. It is observed from the result that the quality distribution is highly disproportional and there are many more wines with qualities 5, 6 and 7 than those with qualities 3,4 and 8 as seen in the barplot below.

Table 4: Vinho Verde Annual Production 1999-2008

Year	**White**	**Red**	**Red Preva-lence (%)**
Year	White	Red	Red Preva-lence (%)
1999/2000	75,322,378	42,430,203	56.33
2000/2001	55,347,997	25,735,143	46.50
2001/2002	85,708,215	40,763,148	47.56
2002/2003	51,552,334	24,586,975	47.69
2003/2004	54,115,085	25,927,194	47.91
2004/2005	61,684,640	29,303,980	47.51
2005/2006	58,653,274	27,813,744	47.42
2006/2007	53,631,404	30,199,897	56.31
2007/2008	45,409,386	18,261,915	40.22



Source: train_set_r dataset

##3.4. Important Variables:##

In prediction of the wine types there are 11 variables or features that may be used as predictors.The R function filterVarImp is used to estimate the importance of each variable. For the two class outcomes, the area under the ROC curve is used as a measure of variable importance.

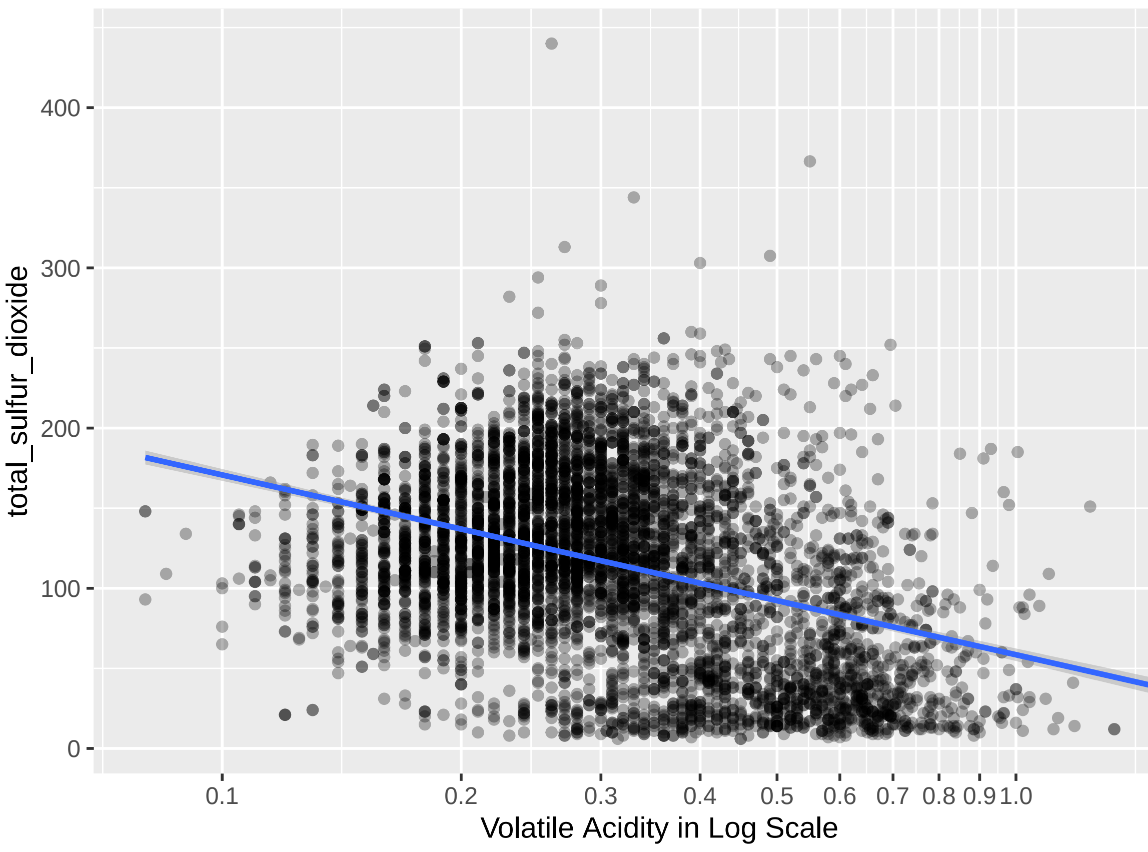
From the result obtained it is observed that the total sulfur dioxide, chlorides and volatile acidity are the most important variable for predicting wine type, while alcohol, citric acid and residual sugar are the least important.

```r
#---------------------------------------
# Important Variables
#---------------------------------------
# The important variable gives an estimate of the predictive power
# of each feature.
# Check the help file for 'filterVarImp' for more information.

# Variable importance for wine type
hux(Feature = rownames(filterVarImp(x = train_set[,xcol],
                                    y = train_set$type)),
    Red   = filterVarImp(x = train_set[,xcol],
                         y = train_set$type)$red,
    White = filterVarImp(x = train_set[,xcol],
                         y = train_set$type)$white,
    add_colnames = TRUE) %>%
  arrange(desc(Red)) %>%
  set_bold(row = 1, everywhere, value = TRUE)          %>%
  set_top_border(row = 1, everywhere, value = 1)    %>%
  set_bottom_border(row = c(1,12), everywhere, value = 1)    %>%
  set_align(row = everywhere, col = 2:3, value = 'right') %>%
  set_caption('Important Variables of Wine Type') %>%
  set_position(value = "center")
```

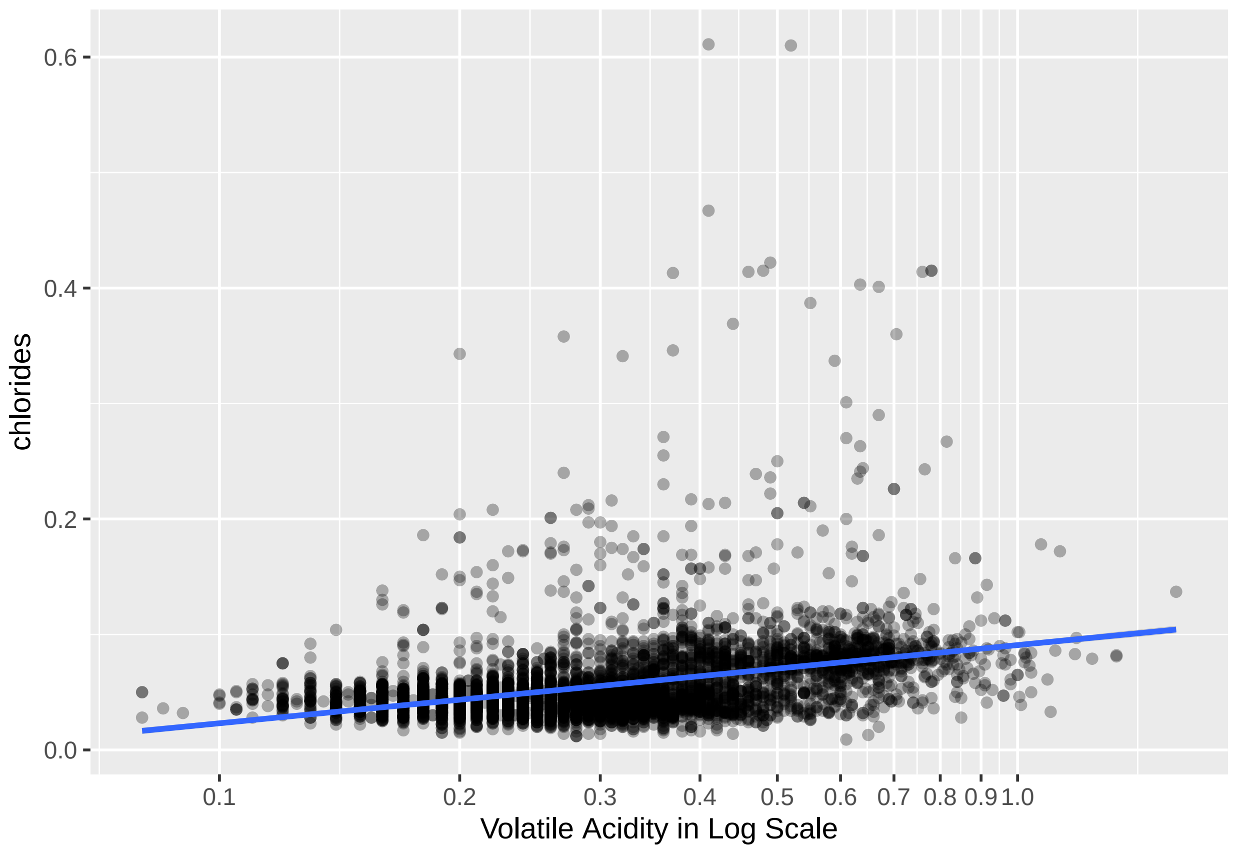
The highly important features are used to construct a geompoint plot against itself to ascer-

Table 5: Important Variables of Wine Type

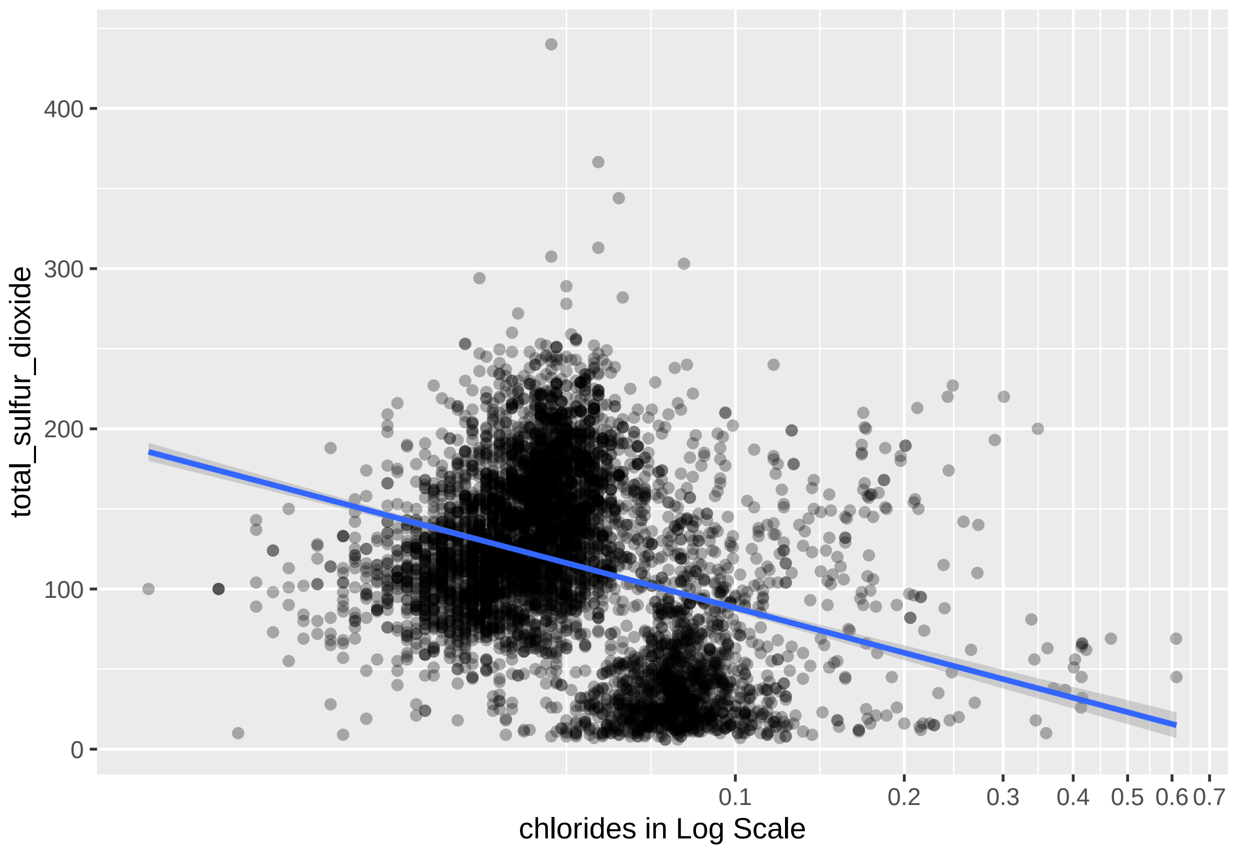
Feature	Red	White
total_sulfur_dioxide	0.953	0.953
chlorides	0.944	0.944
volatile_acidity	0.902	0.902
free_sulfur_dioxide	0.848	0.848
sulphates	0.83	0.83
fixed_acidity	0.782	0.782
density	0.772	0.772
pH	0.728	0.728
residual_sugar	0.674	0.674
citric_acid	0.608	0.608
alcohol	0.513	0.513



tain the association.



chlorides
0.6
0.4
0.2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Volatile Acidity in Log Scale



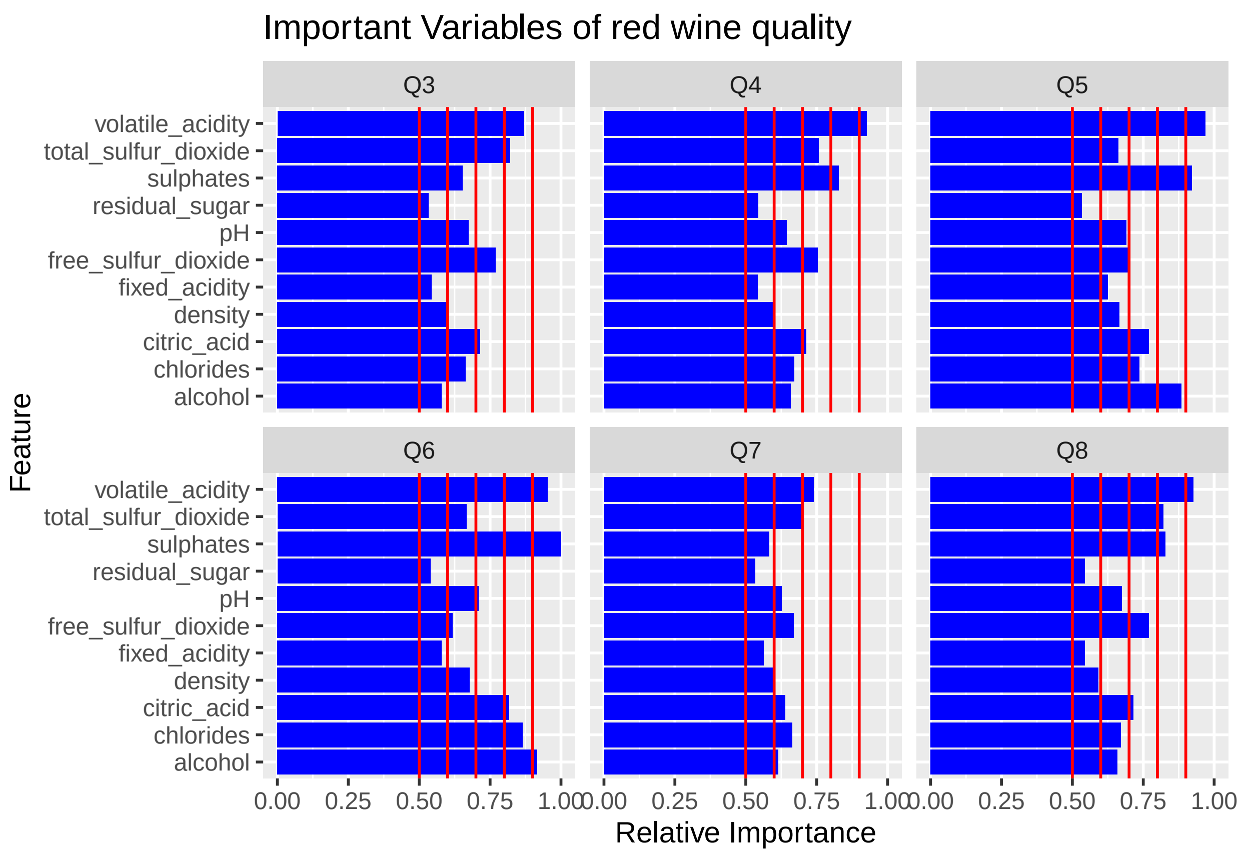
In predicting the wine quality in a scale of 1 to 9 in order to obtain a multi-class outcome for a specific class, the problem is decomposed into a pair-wise problem and the area under the curve is calculated for each class, hence the maximum area under the curve across the relevant pair-wise AUCâ€™s is used as an important variable measure. From the result the volatile acidity, sulphates and alcohol have very high AUC values for red wine quality

```
#-------------------
#Important Variables  for red wine quality
#-------------------
x <- train_set_r[,xcol]
y <- train_set_r$quality

hux(Feature = rownames(filterVarImp(x = x, y = y)),
    filterVarImp(x = x, y = y),
    add_colnames = TRUE) %>%
  # Format header row
  set_bold(row = 1, everywhere, value = TRUE)           %>%
  set_top_border(row = 1, everywhere, value = 1)        %>%
  set_bottom_border(row = c(1,12), everywhere, value = 1)     %>%
  # Format numbers
  set_number_format(row = 2:12, col = 2:7, value = 3)  %>%
```

```r
map_text_color(row = everywhere, col = 2:7,
               by_ranges(seq(0.6, 0.9, 0.1), colorblind_pal()(5))) %>%
# Format alignment
set_align(row = everywhere, col = 1,   value = 'left')  %>%
set_align(row = everywhere, col = 2:7, value = 'right') %>%
# Title
set_caption('Important variables of red Wine quality') %>%
set_position(value = "center")
```

Table 6: Important variables of red Wine quality

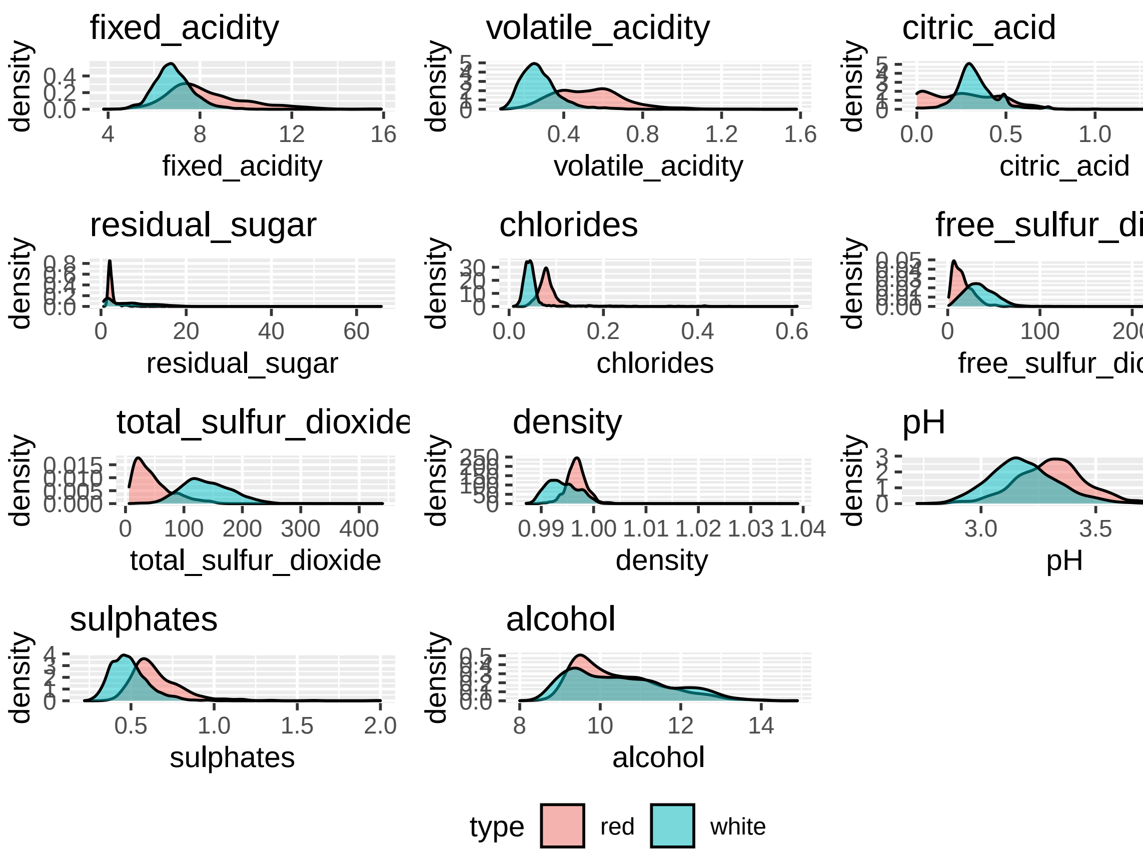
Feature	Q3	Q4	Q5	Q6	Q7	Q8
fixed_acidity	0.544	0.543	0.626	0.579	0.564	0.544
volatile_acidity	0.869	0.928	0.969	0.952	0.740	0.928
citric_acid	0.715	0.713	0.771	0.817	0.639	0.715
residual_sugar	0.534	0.544	0.534	0.540	0.534	0.544
chlorides	0.664	0.671	0.737	0.865	0.664	0.671
free_sulfur_dioxide	0.769	0.754	0.702	0.617	0.670	0.769
total_sulfur_dioxide	0.822	0.758	0.662	0.667	0.701	0.822
density	0.598	0.598	0.665	0.679	0.598	0.591
pH	0.674	0.645	0.691	0.710	0.626	0.674
sulphates	0.653	0.827	0.922	1.000	0.584	0.827
alcohol	0.579	0.660	0.885	0.917	0.614	0.660



From the relative importance of the features there is an overlapping for different quality outcome, it is therefore important to utilize machine learning models that have the capacity to use two or more features for prediction, thereby solving the problem of low predictive power as a result of highly correlated features.

A look at each feature distribution for each wine type and quality of red wines would be studied and the correlation among the features presented.

```r
#=========================================
# Data visualization
#=========================================
# In this section we create several stats plots
# to check the distribution of variables.

# Install and load the libraries used for visualization
# The 'load_lib' function was defined earlier.
load_lib(c("gridExtra", "ggridges", "ggplot2",
           "gtable", "grid", "egg"))


# The 'grid_arrange_shared_legend' function creates a grid of
# plots with one legend for all plots.
# There's no commentaries because I use the code from the source below.
# Reference: Baptiste AuguiÃ© - 2019
# https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html
grid_arrange_shared_legend <-
  function(...,
           ncol = length(list(...)),
           nrow = 1,
           position = c("bottom", "right")) {

    plots <- list(...)
    position <- match.arg(position)
    g <-
      ggplotGrob(plots[[1]] + theme(legend.position = position))$grobs
    legend <- g[[which(sapply(g, function(x)
      x$name) == "guide-box")]]
    lheight <- sum(legend$height)
    lwidth <- sum(legend$width)
    gl <- lapply(plots, function(x)
      x + theme(legend.position = "none"))
    gl <- c(gl, ncol = ncol, nrow = nrow)

    combined <- switch(
      position,
      "bottom" = arrangeGrob(
        do.call(arrangeGrob, gl),
        legend,
        ncol = 1,
        heights = unit.c(unit(1, "npc") - lheight, lheight)
      ),
      "right" = arrangeGrob(
        do.call(arrangeGrob, gl),
        legend,
        ncol = 2,
        widths = unit.c(unit(1, "npc") - lwidth, lwidth)
      )
    )

    grid.newpage()
    grid.draw(combined)

    # return gtable invisibly
    invisible(combined)

  }
```



##3.5 Density Plot##

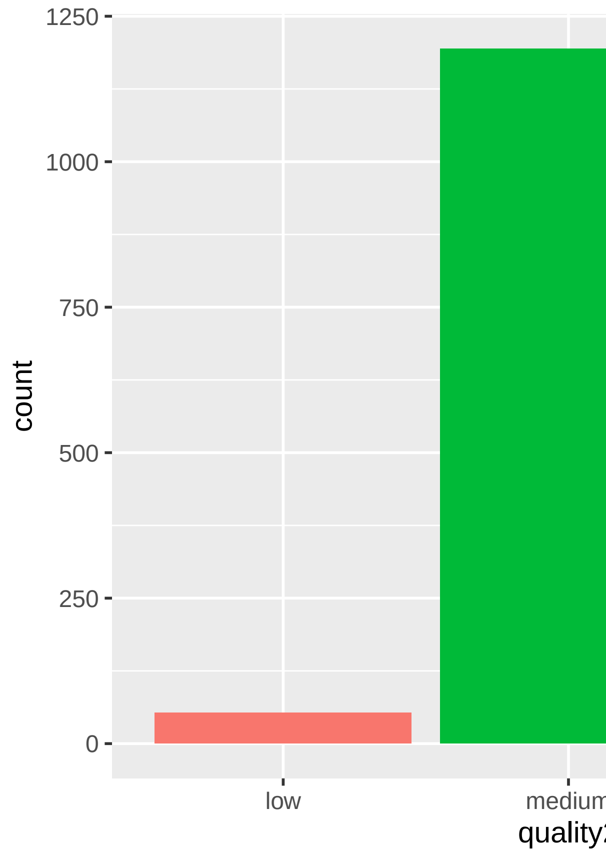
The general distribution of the variables are presented in the geomplot above. The different variables are arranged in threes to obtain a larger plot as presented below. The difference in the peaks for red and white wine suggest that the predictors would be useful in the distinguishing of both wine types. The shifting to the left side of the x-axis indicates possible outliers in the right side.

The comparative studies of the two types of wine, the red and white using three variables such as alcohol, PH and citric_acid displayed shows that the alcohol level is higher in the red wine having a peak density of 0.5 while the white wine peaked at 0.35. The citric_acid level is higher in the white wine at density of 5 while the red wine has a lower citric_acid. The PH level is equal in the two wines.

The white wine contains higher density of volatile_acidity, higher chlorides and lower total_sulfur_dioxide while the red wine possess higher total_sulfur_dioxide, lower volatile_acidity and lower chlorides.

In distiquishing the two types of wine based on the variables, the sulphates content and the fixed_acidity in the white wine is relatively higher comparing to that of red wine, while the red wine contains hiher free_sulphur_dioxide comparing to the red wine.

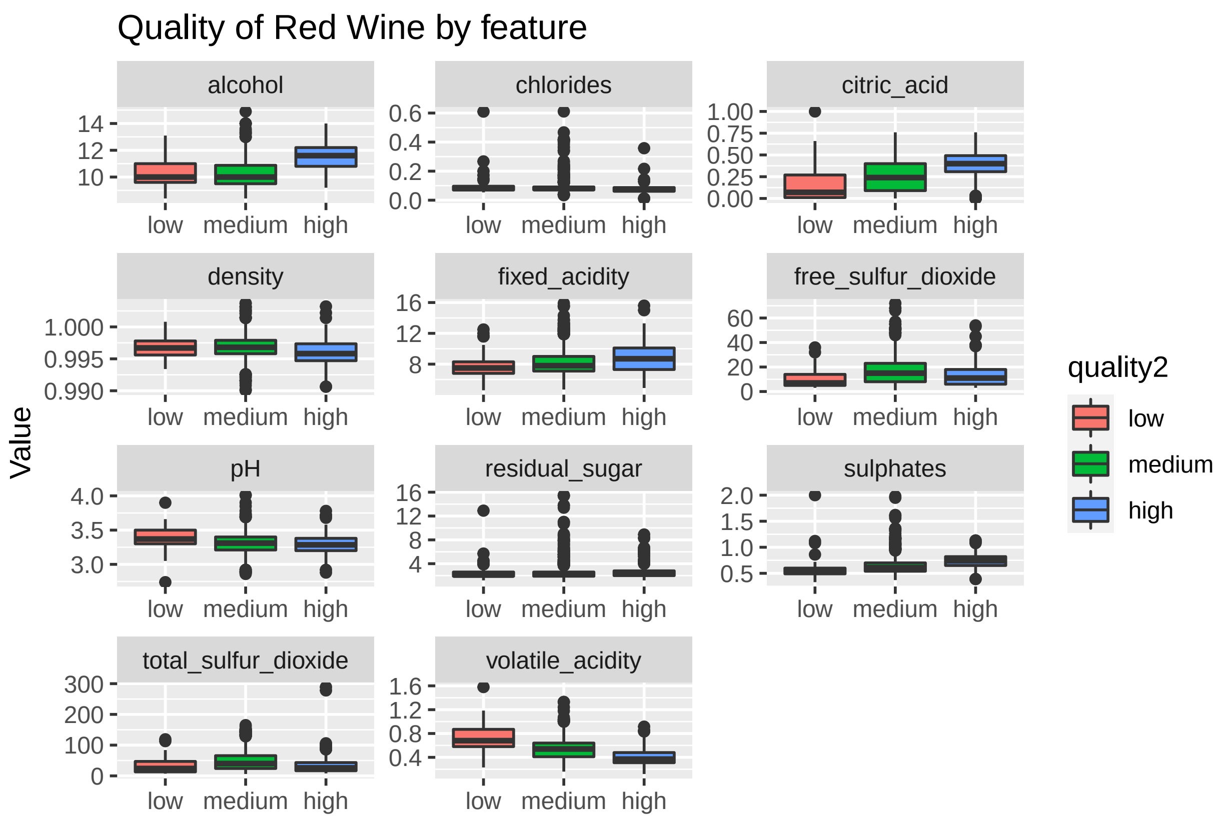
The residual_sugar and density peaked higher in red wine compared to the white wine



while the white wine is relatively higher in sulphates content.
##3.6 Box Plot##

Box plots show a summary of the main values in the distribution of red wine that is the lower and upper limits, the first and third quantiles, the mean and outliers. The distribution overlaps among all quality levels for all predictors. The mean also is flat for all quality levels and predictors, except for citric acid and sulphates that increases with quality levels and volatile acidity that decreases.This further butresses the fact that the distribution is higher in Q5 and Q6 which categorized together as medium.

All features have outliers.

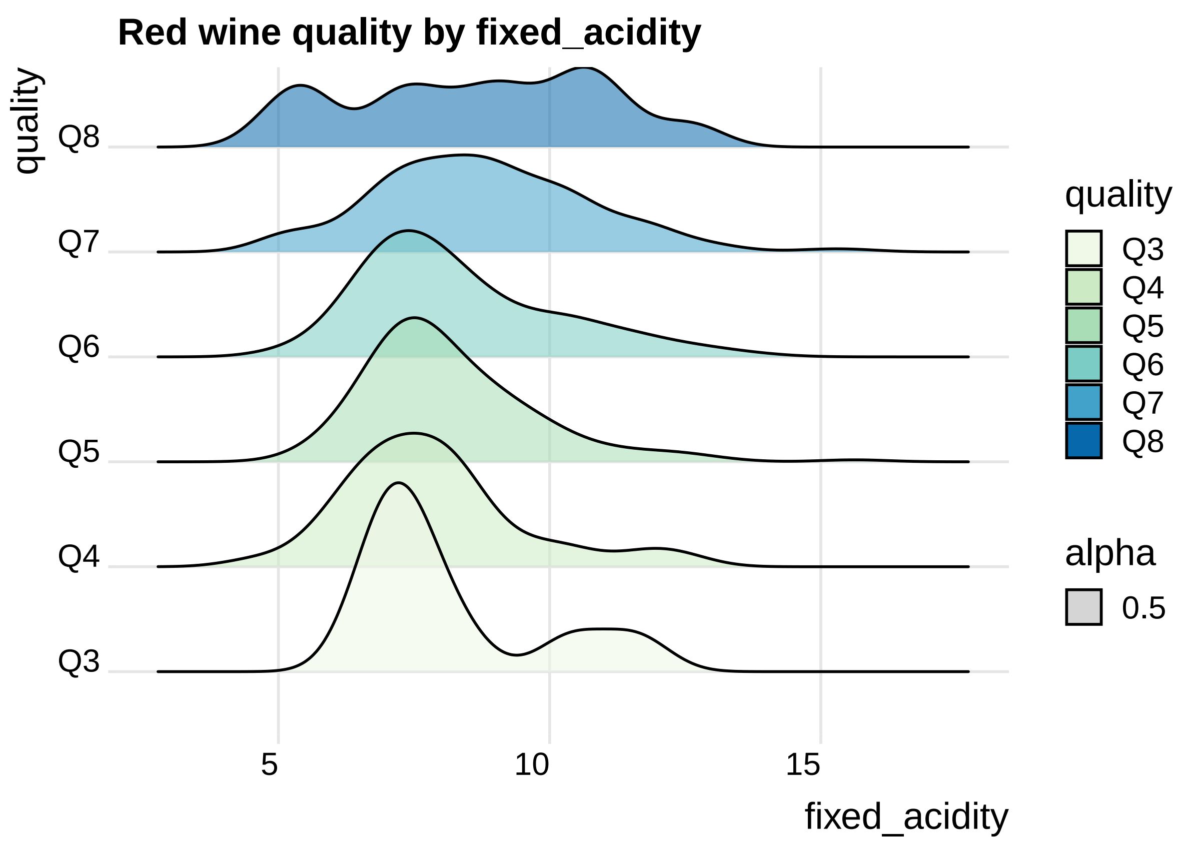


From the result earlier obtained there is no clear distinction of quality levels among the predictors. Maybe grouping the quality values may help. We would denote low quality for levels 3 and 4, medium for levels 5 and 6, and high for levels 7 and 8.

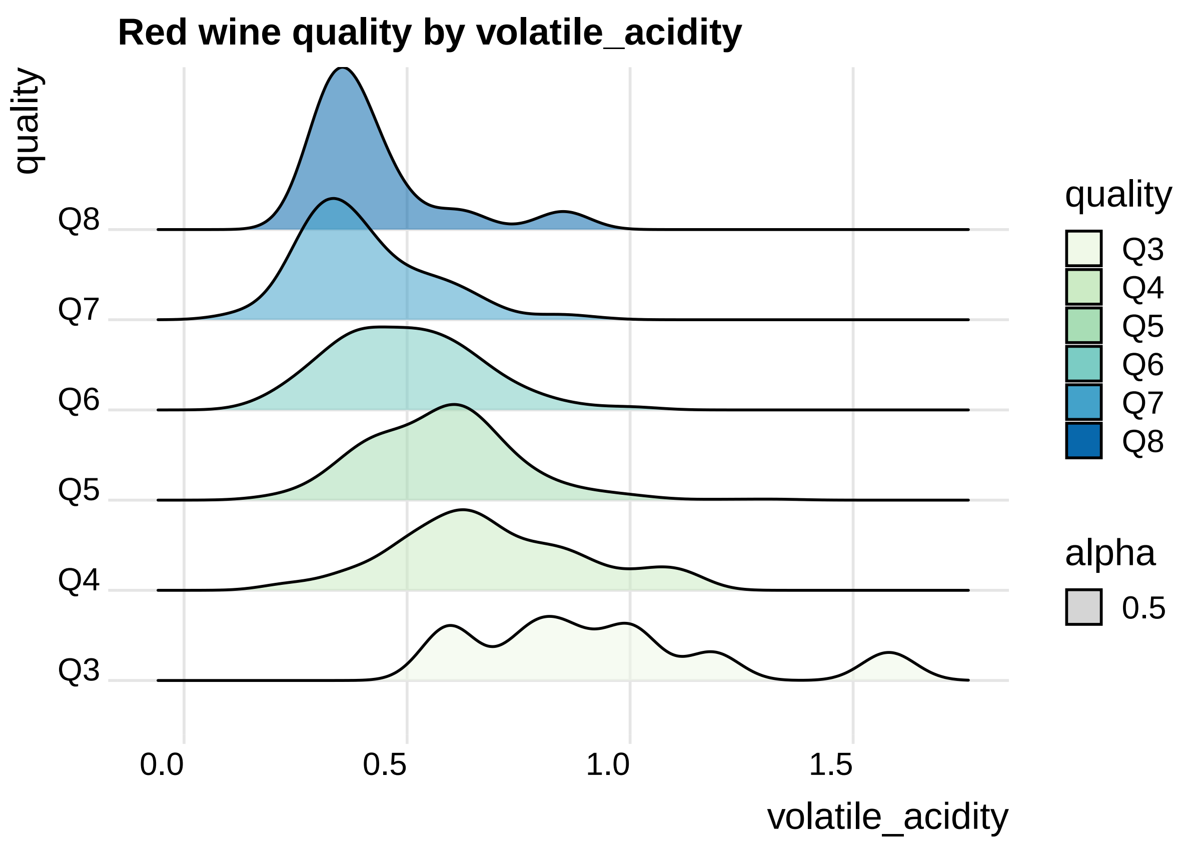
##3.7. Quantile-Quantile plot:## A Q-Q plot is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a line that is roughly straight.

The QQ plots compare the feature distribution with the normal distribution. All variables are close to the normal distribution within two standard deviations of the mean. Some machine learning algorithms, such as Linear Discriminant Analysis (LDA) and Quadractic Discriminant Analysis(QDA), assume the predictors are normally distributed.

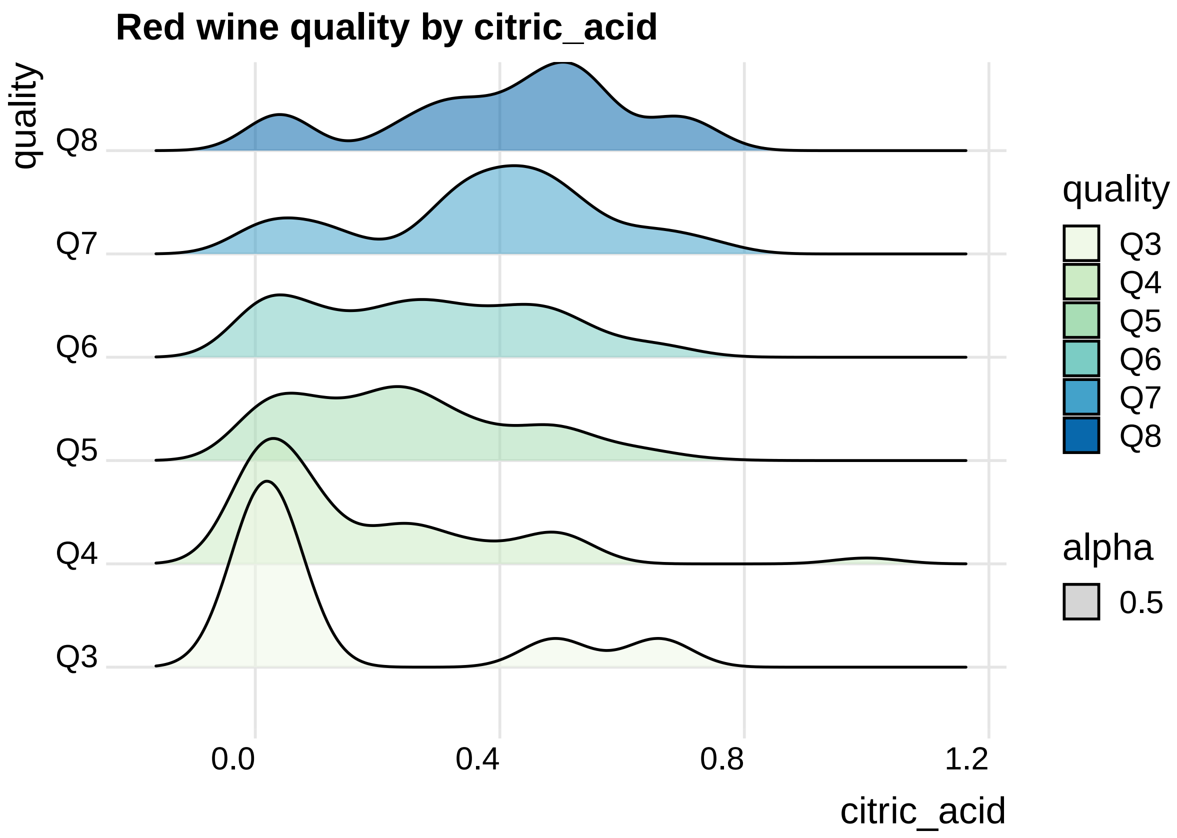
```
## [[1]]
```


Red wine quality by fixed_acidity
quality
Q8
Q7
Q6
Q5
Q4
Q3
5
10
15
fixed_acidity
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

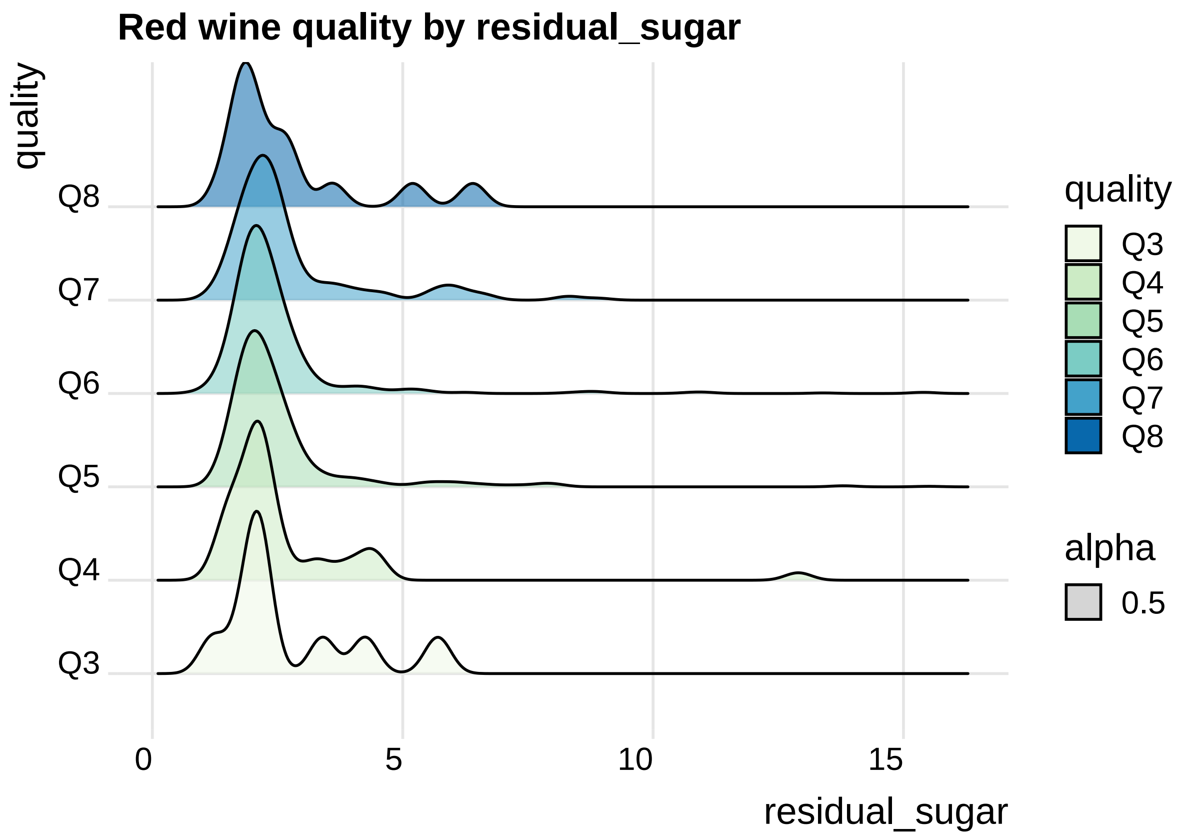
```
##
## [[2]]
```


Red wine quality by volatile_acidity
quality
Q8
Q7
Q6
Q5
Q4
Q3
0.0
0.5
1.0
1.5
volatile_acidity
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

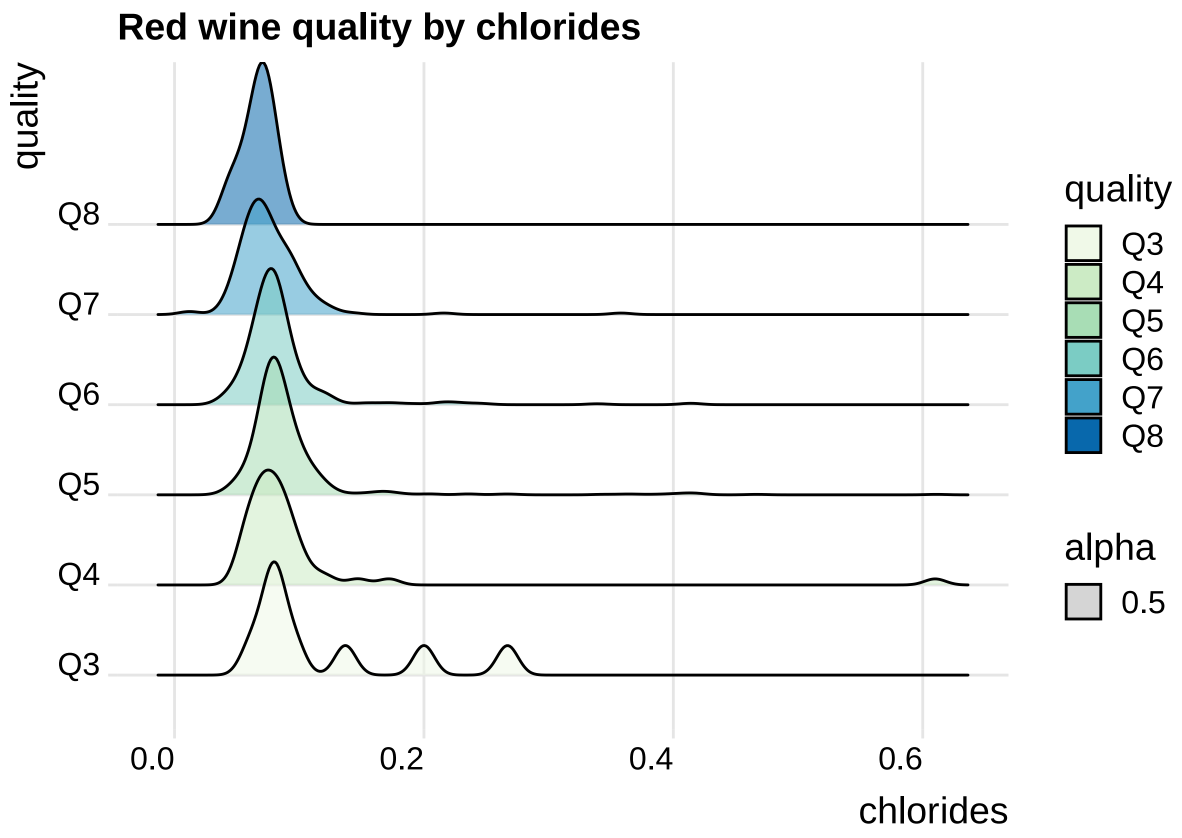
```
##
## [[3]]
```


Red wine quality by citric_acid
quality
Q8
Q7
Q6
Q5
Q4
Q3
0.0
0.4
0.8
1.2
citric_acid
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

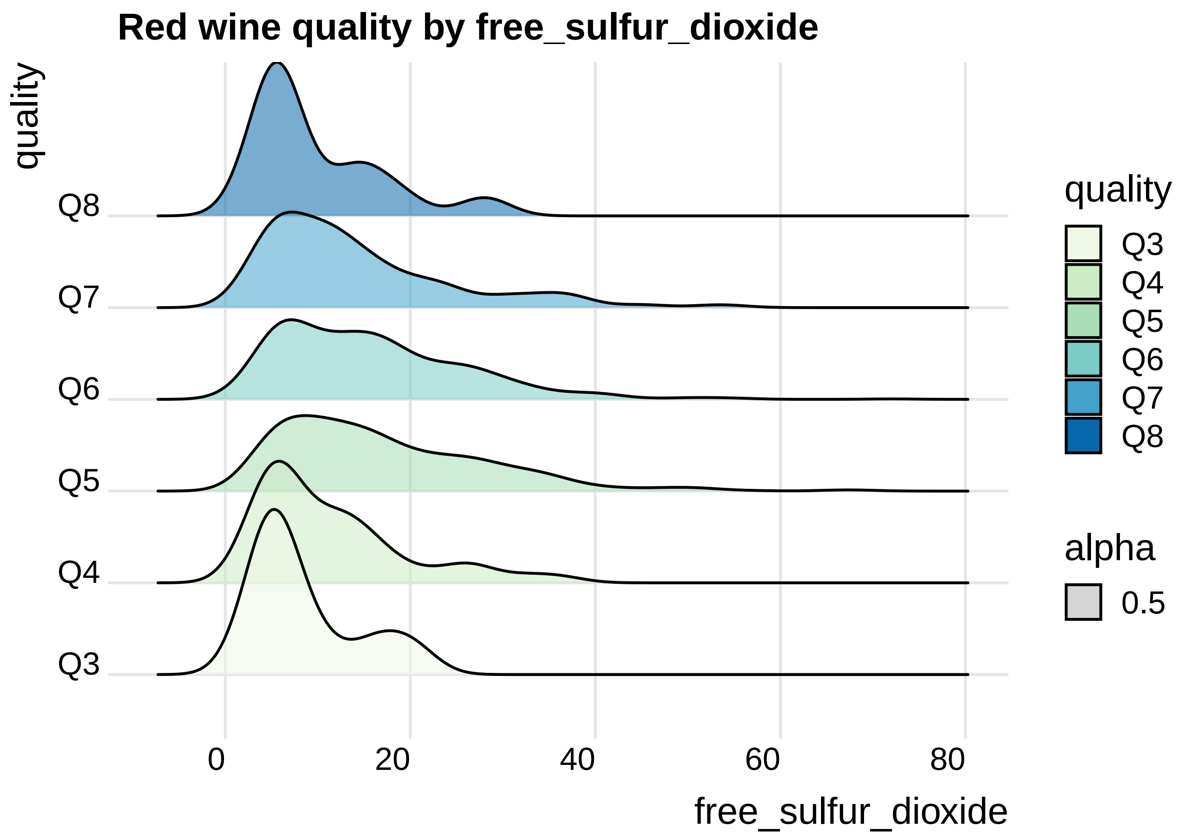
```
##
## [[4]]
```


Red wine quality by residual_sugar
quality
Q8
Q7
Q6
Q5
Q4
Q3
0
5
10
15
residual_sugar
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

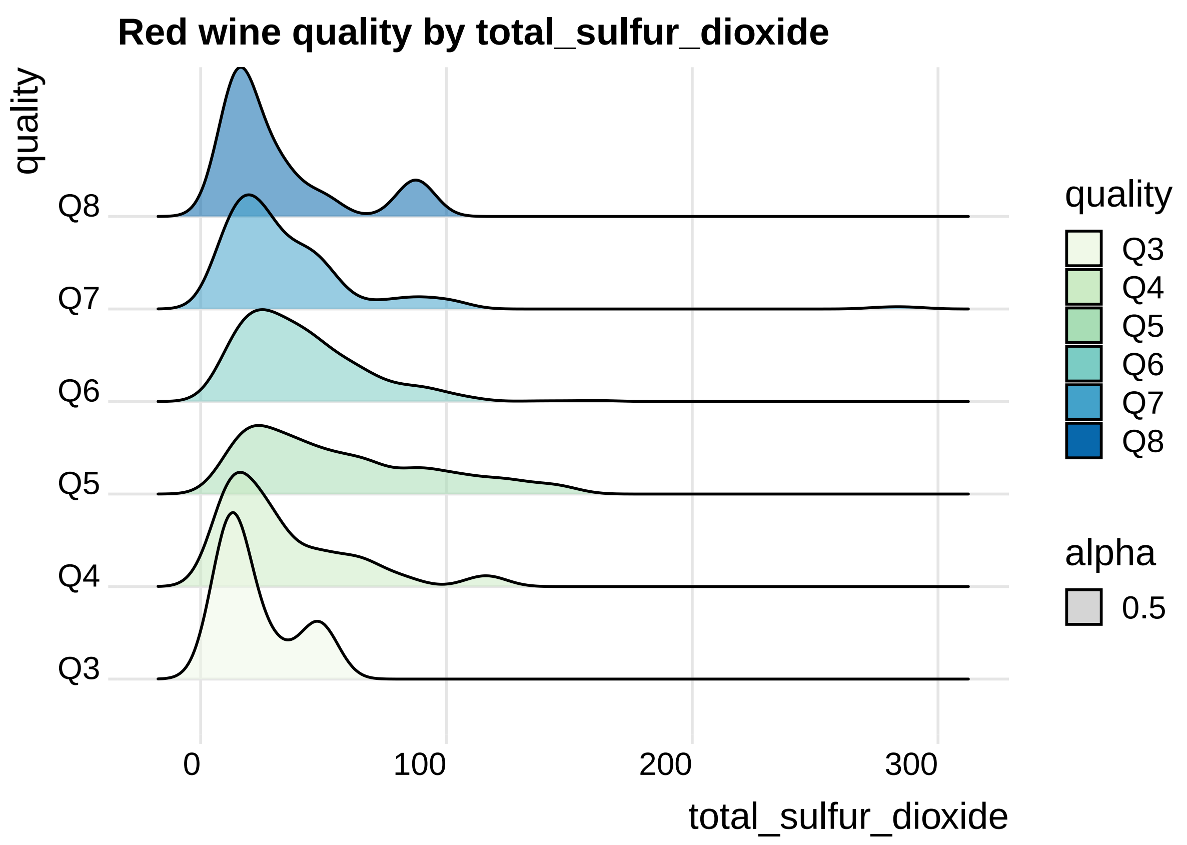
```
##
## [[5]]
```


Red wine quality by chlorides
quality
Q8
Q7
Q6
Q5
Q4
Q3
0.0
0.2
0.4
0.6
chlorides
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

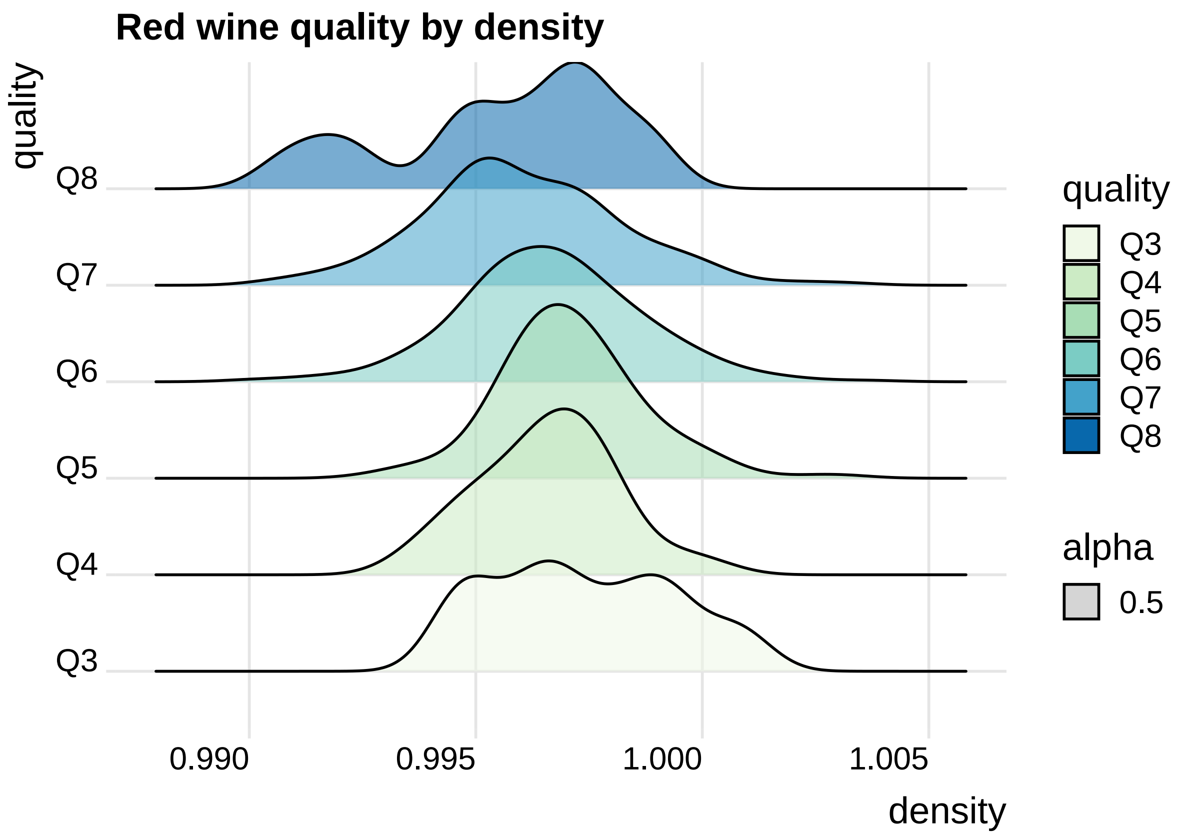
```
##
## [[6]]
```


Red wine quality by free_sulfur_dioxide
quality
Q8
Q7
Q6
Q5
Q4
Q3
0
20
40
60
80
free_sulfur_dioxide
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

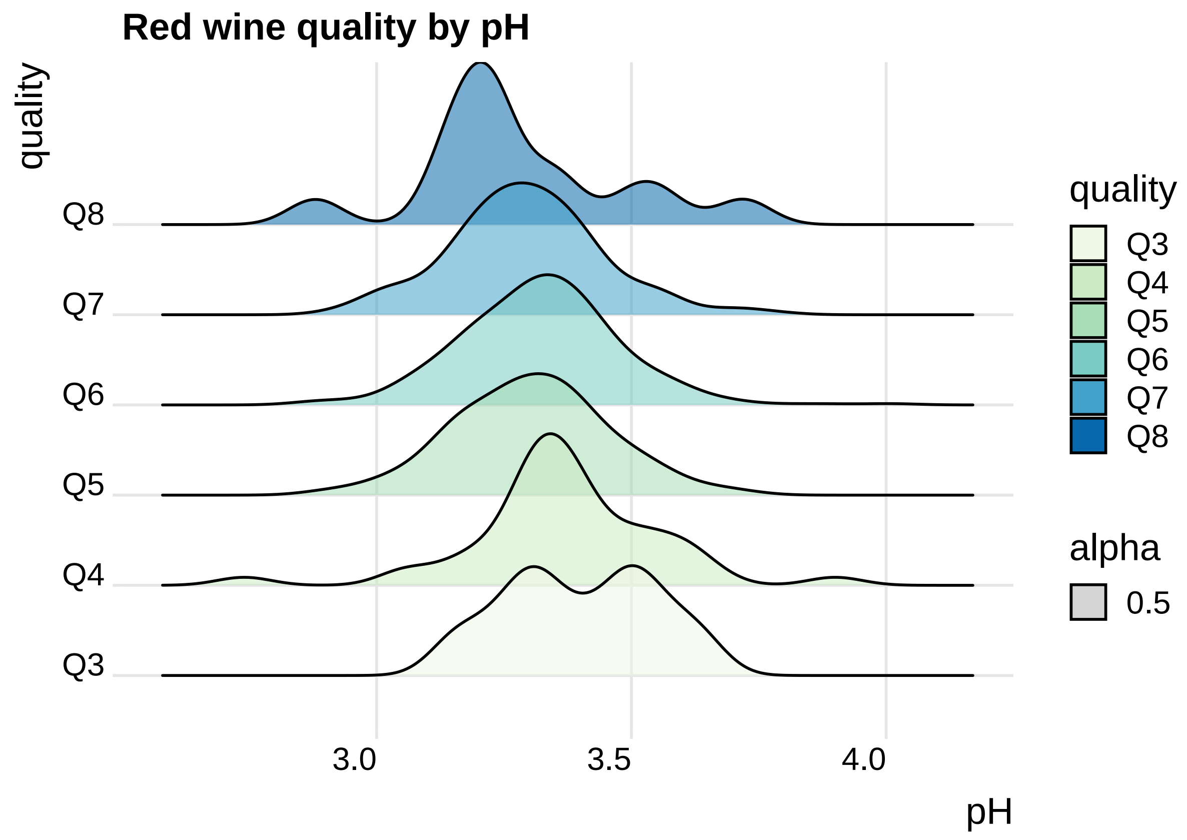
```
##
## [[7]]
```


Red wine quality by total_sulfur_dioxide
quality
Q8
Q7
Q6
Q5
Q4
Q3
0
100
200
300
total_sulfur_dioxide
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

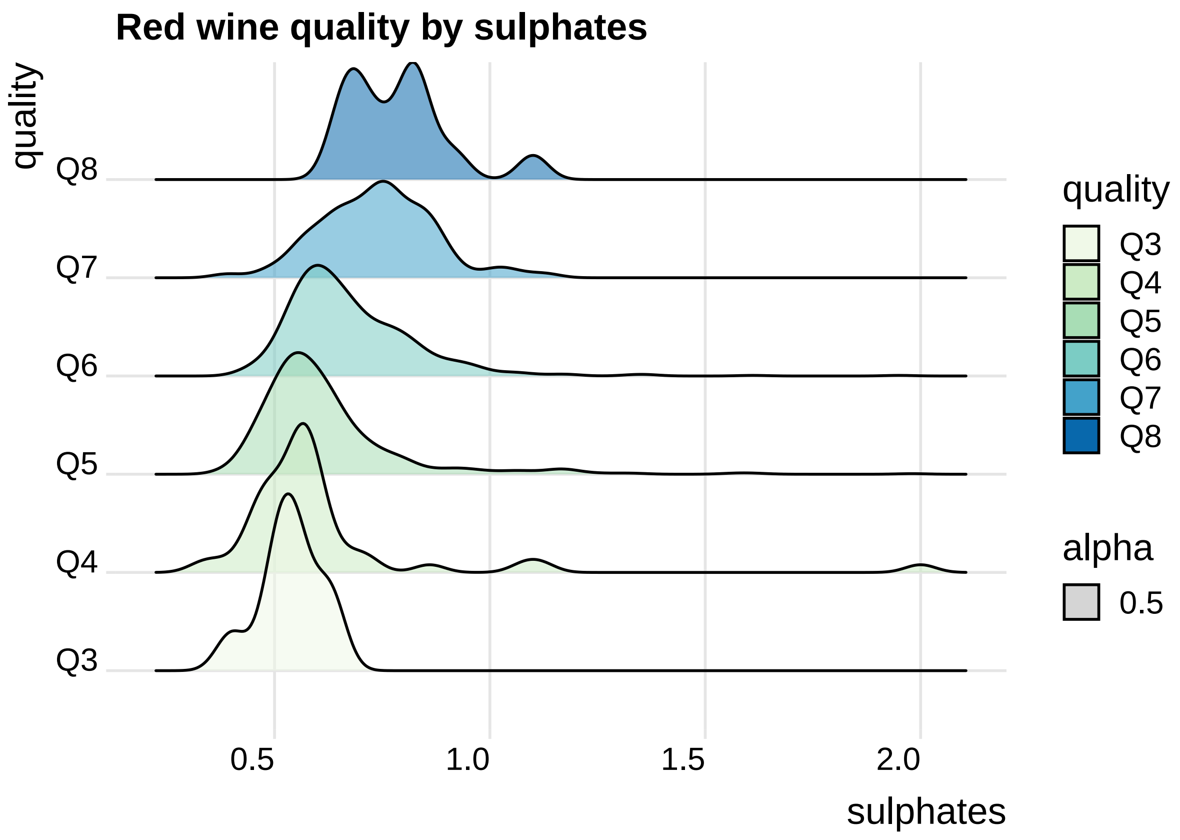
```
##
## [[8]]
```


Red wine quality by density
quality
Q8
Q7
Q6
Q5
Q4
Q3
0.990
0.995
1.000
1.005
density
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

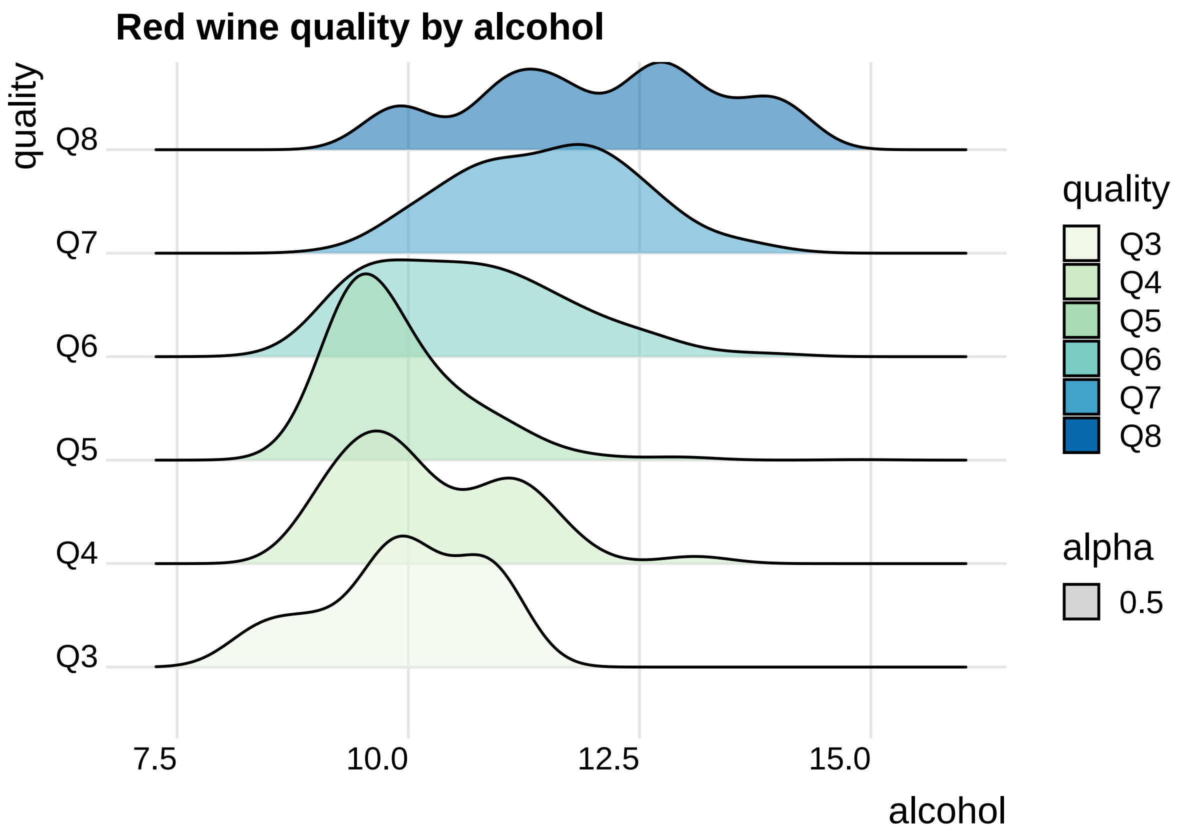
```
##
## [[9]]
```

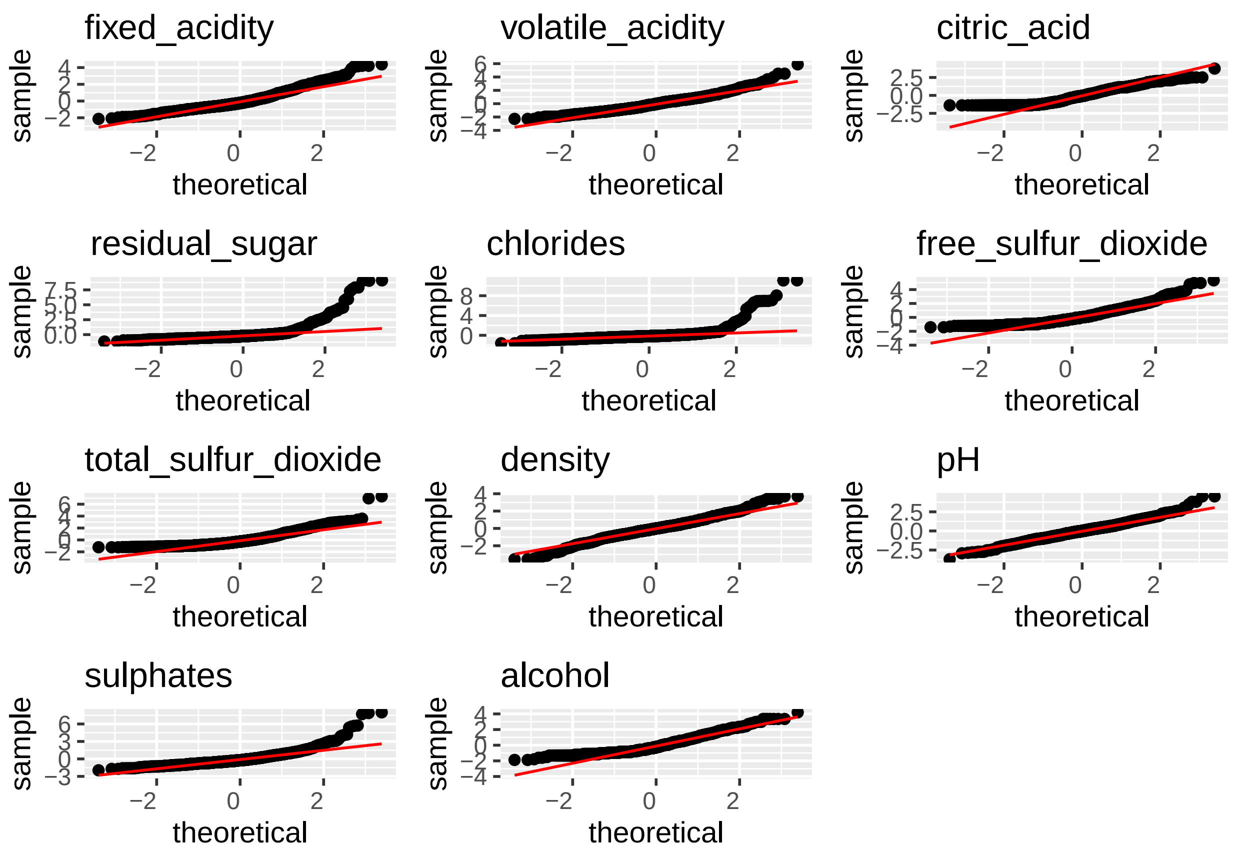

Red wine quality by pH
quality
Q8
Q7
Q6
Q5
Q4
Q3
3.0
3.5
4.0
pH
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

```
##
## [[10]]
```


Red wine quality by sulphates
quality
Q8
Q7
Q6
Q5
Q4
Q3
0.5
1.0
1.5
2.0
sulphates
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5

```
##
## [[11]]
```

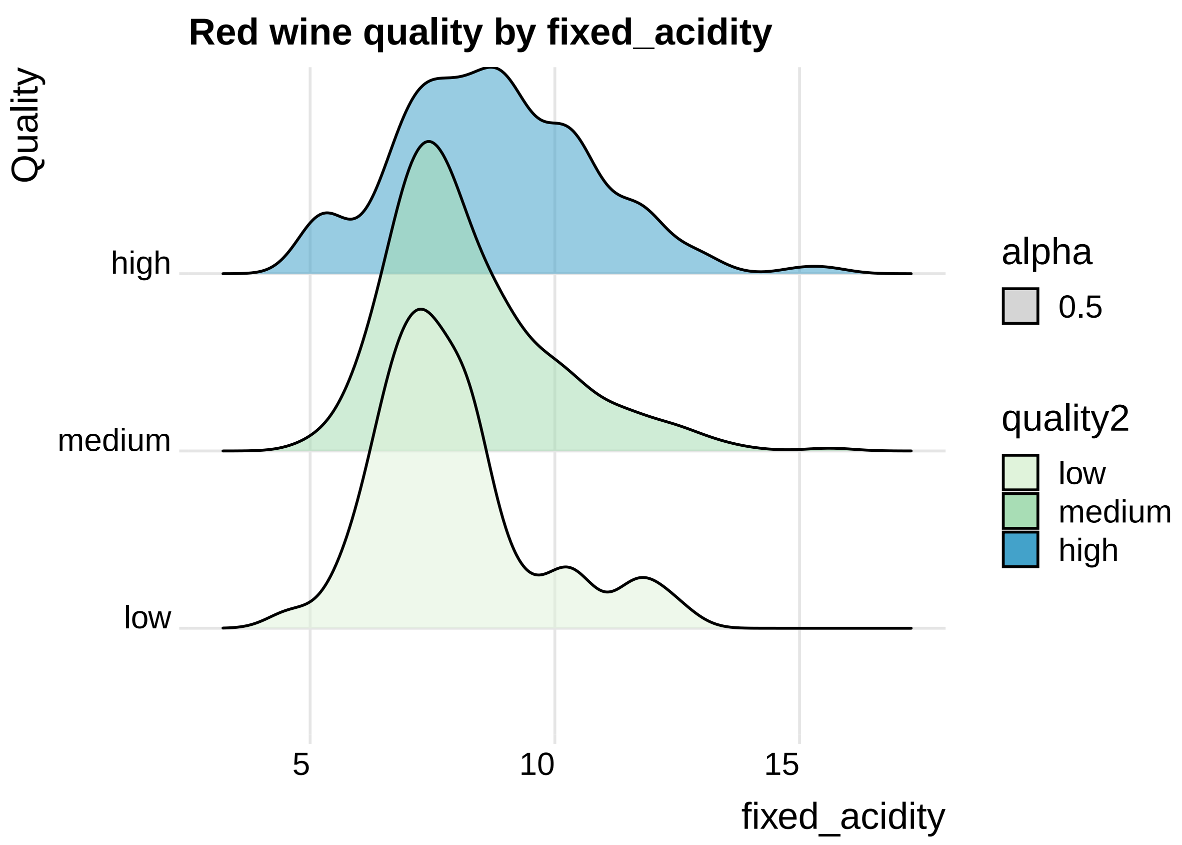

Red wine quality by alcohol
quality
Q8
Q7
Q6
Q5
Q4
Q3
7.5
10.0
12.5
15.0
alcohol
quality
Q3
Q4
Q5
Q6
Q7
Q8
alpha
0.5



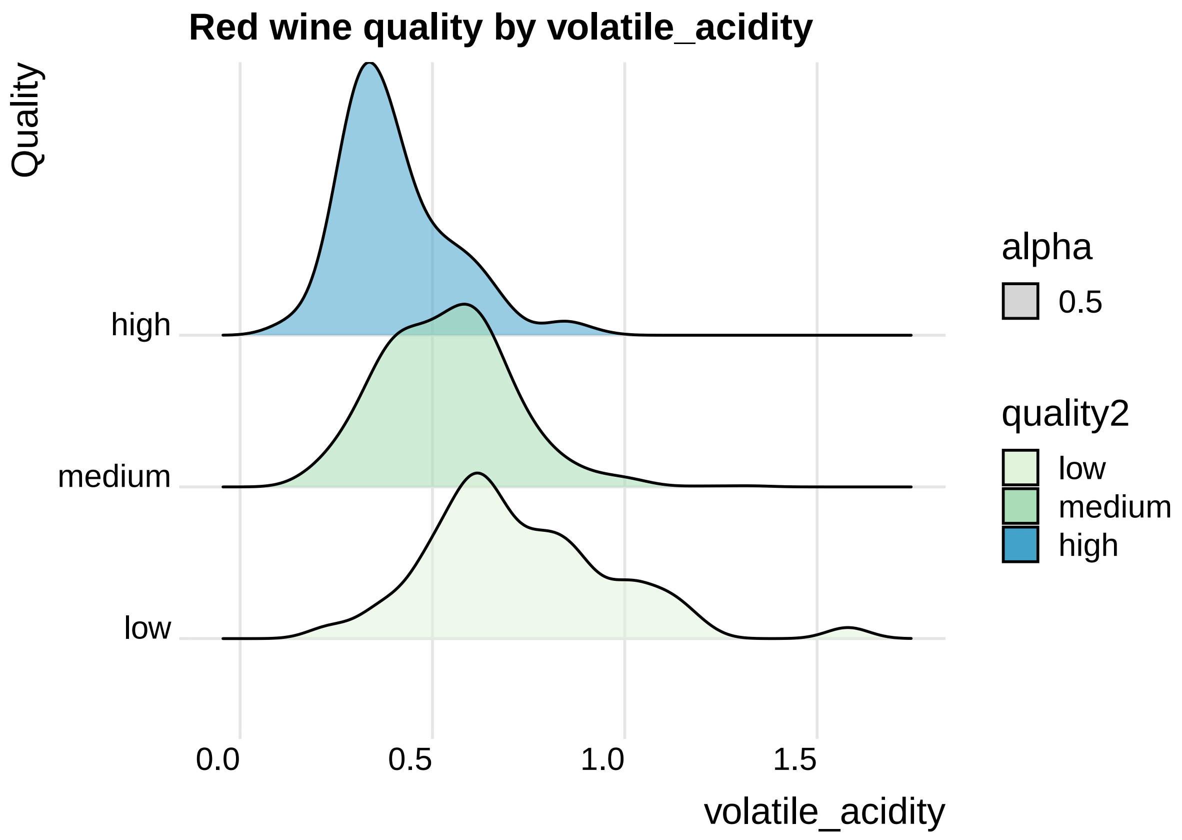
##3.8. Density Ridge##

The density ridge plots shows the different physicochemical variables and its quality. The ridgeline plots are partially overlapping line plots that create the impression of a mountain range and are used for visualizing changes in distributions over time or space.

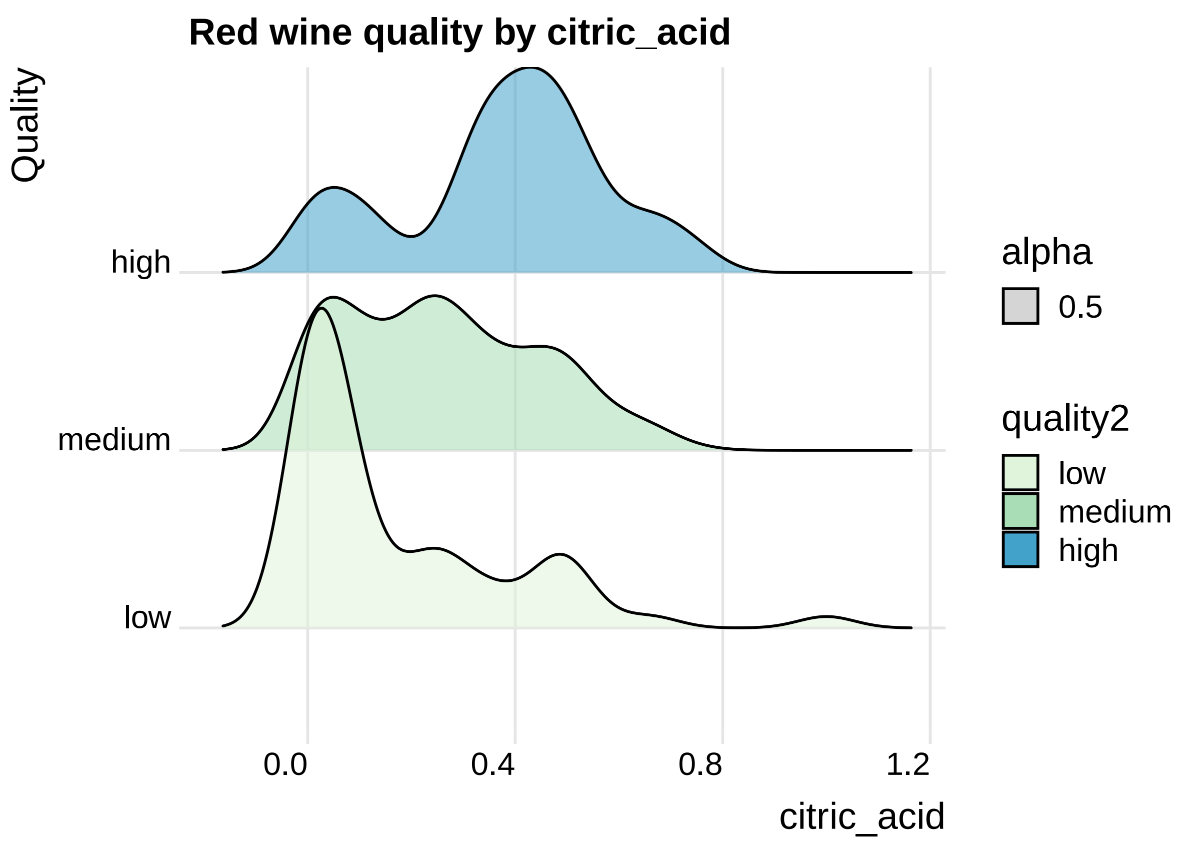
```
## [[1]]
```


Red wine quality by fixed_acidity
Quality
high
medium
low
5
10
15
fixed_acidity
alpha
0.5
quality2
low
medium
high

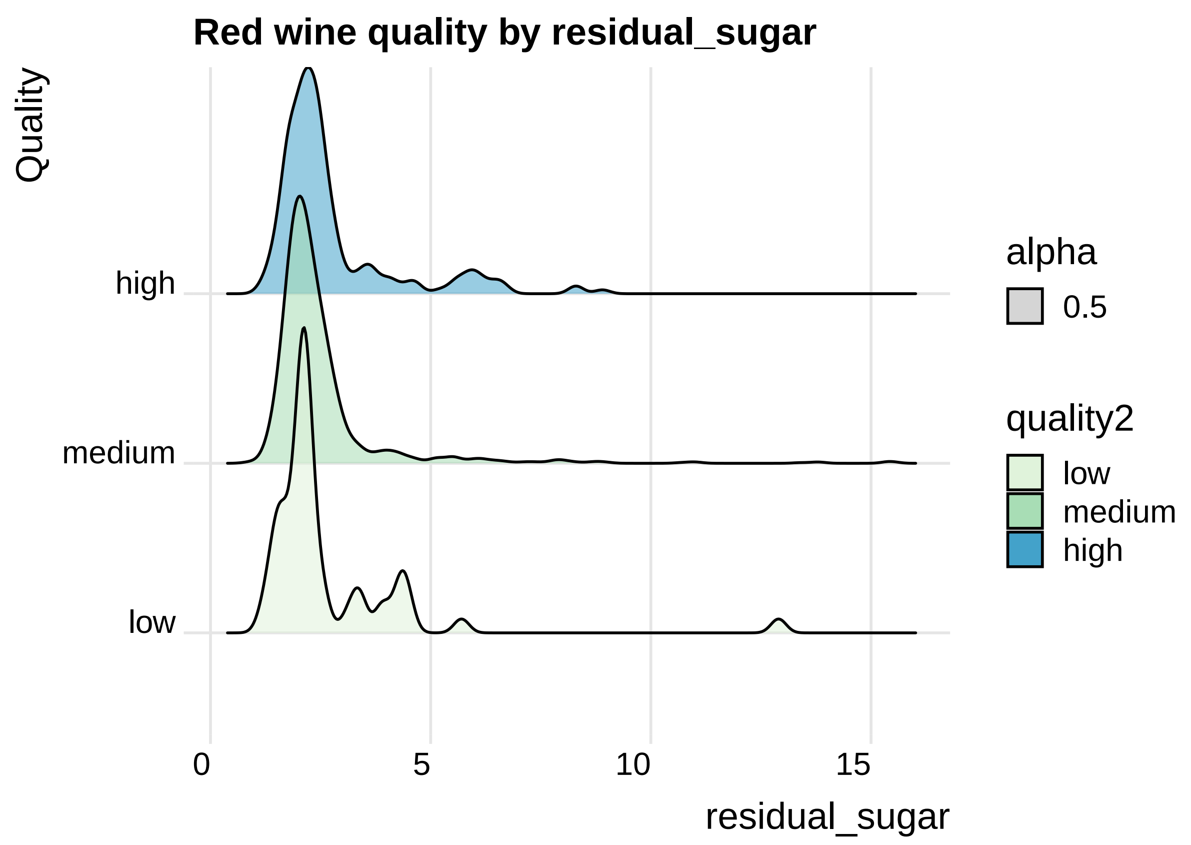
```
##
## [[2]]
```


Red wine quality by volatile_acidity
Quality
high
medium
low
0.0
0.5
1.0
1.5
volatile_acidity
alpha
0.5
quality2
low
medium
high

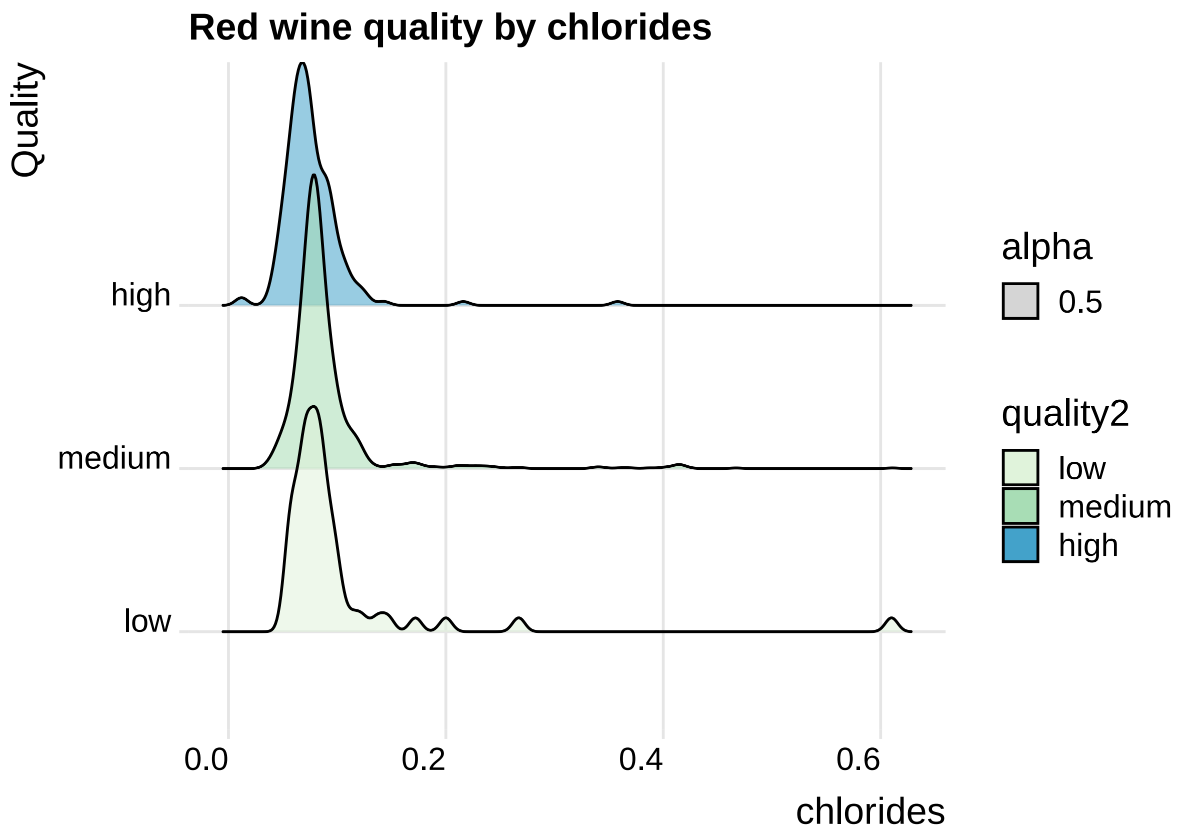
```
##
## [[3]]
```


Red wine quality by citric_acid
Quality
high
medium
low
0.0
0.4
0.8
1.2
citric_acid
alpha
0.5
quality2
low
medium
high

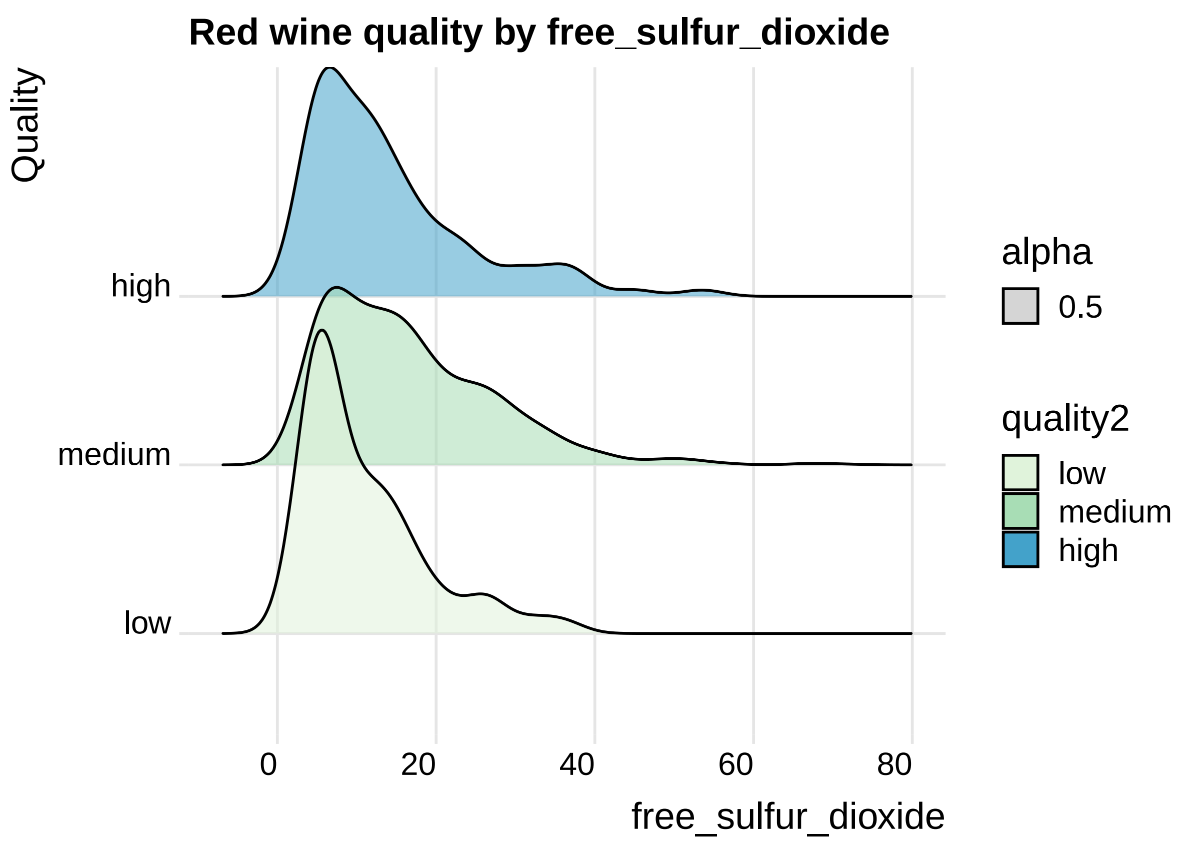
```
##
## [[4]]
```


Red wine quality by residual_sugar
Quality
high
medium
low
0
5
10
15
residual_sugar
alpha
0.5
quality2
low
medium
high

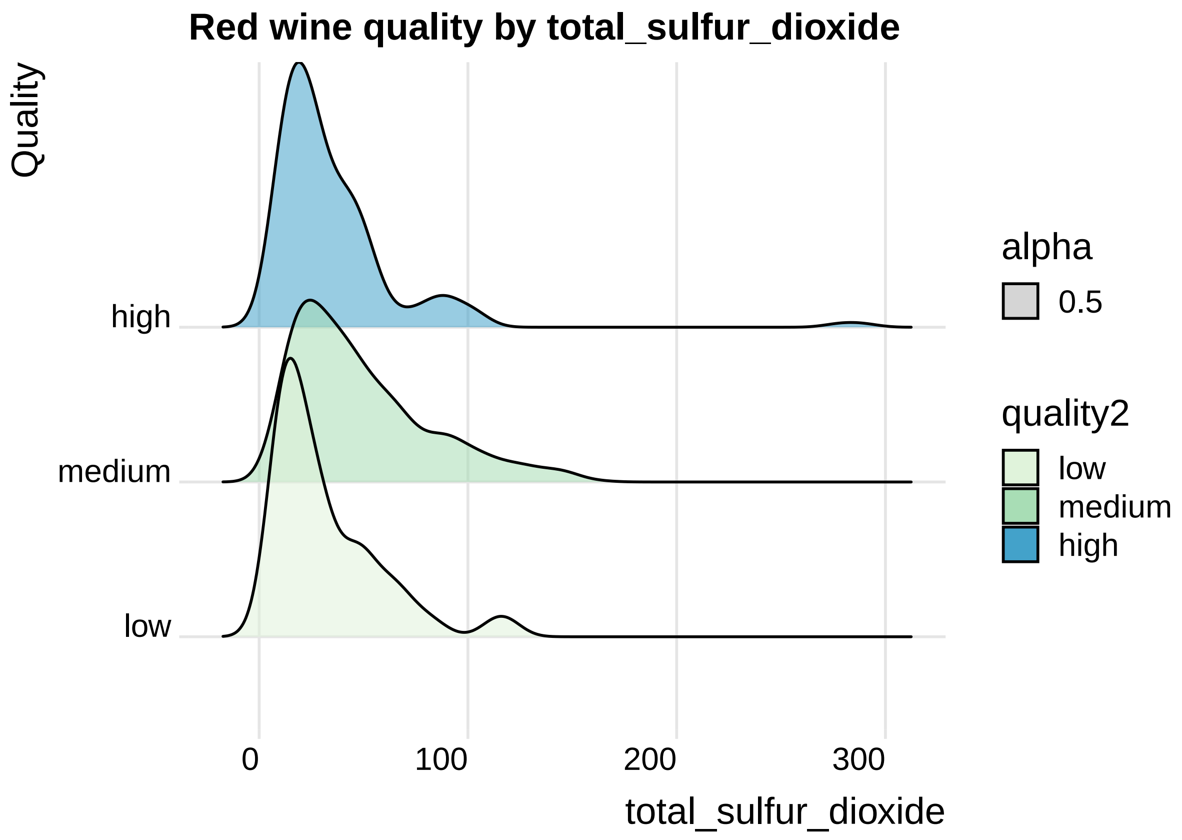
```
##
## [[5]]
```


Red wine quality by chlorides
Quality
high
medium
low
0.0
0.2
0.4
0.6
chlorides
alpha
0.5
quality2
low
medium
high

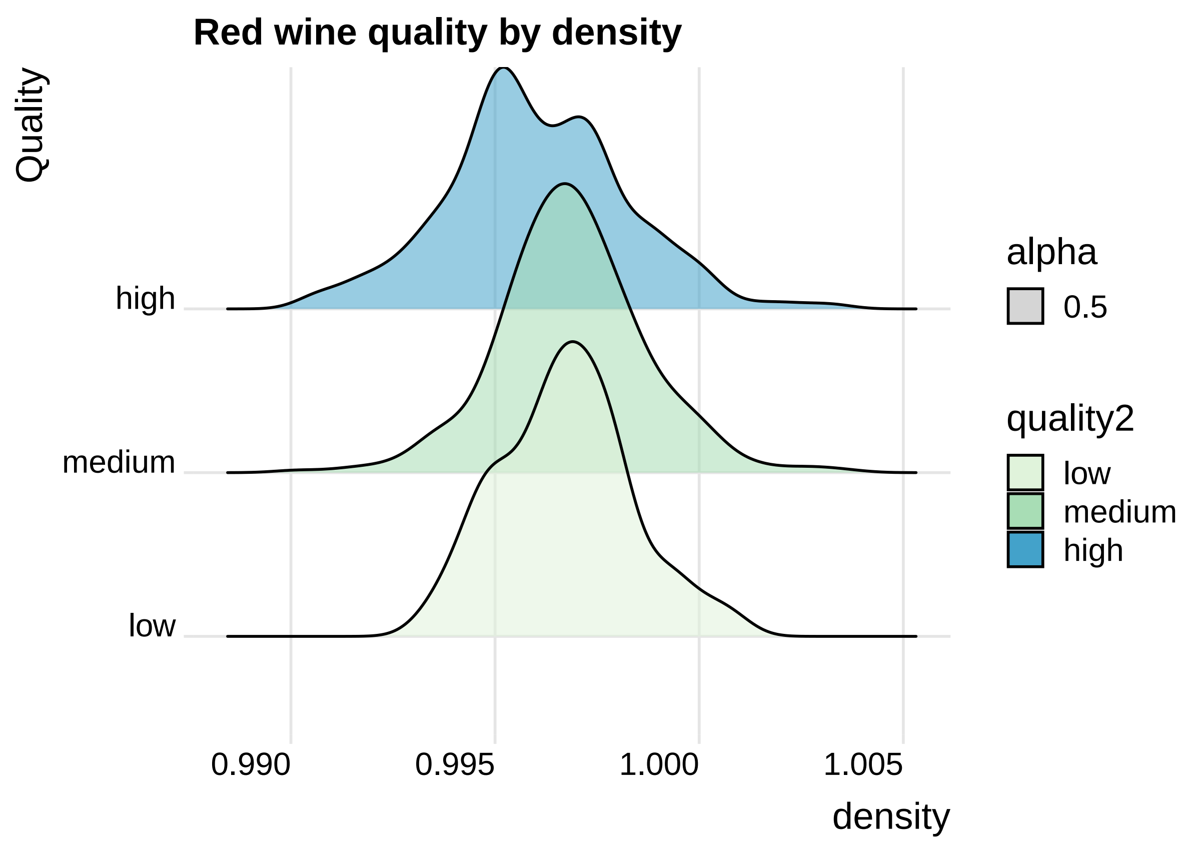
```
##
## [[6]]
```


Red wine quality by free_sulfur_dioxide
Quality
high
medium
low
0
20
40
60
80
free_sulfur_dioxide
alpha
0.5
quality2
low
medium
high

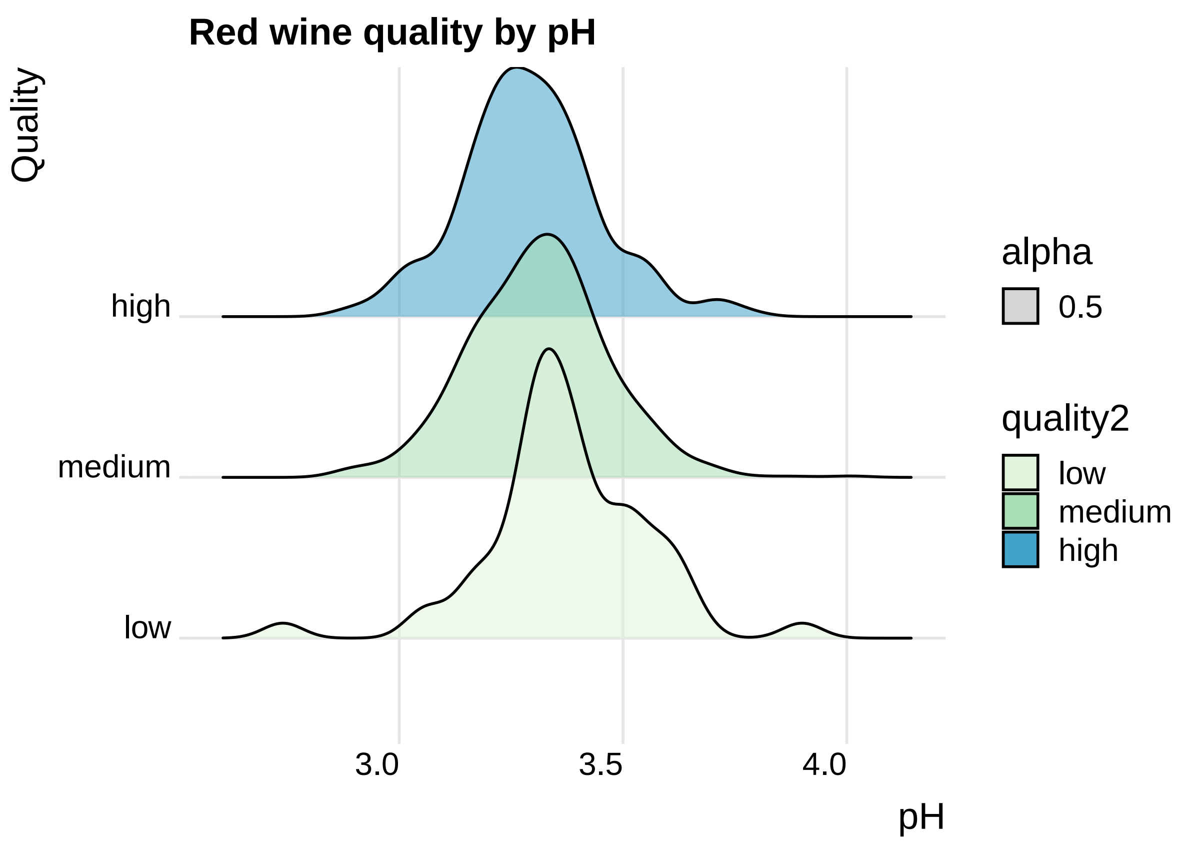
```
##
## [[7]]
```


Red wine quality by total_sulfur_dioxide
Quality
high
medium
low
0
100
200
300
total_sulfur_dioxide
alpha
0.5
quality2
low
medium
high

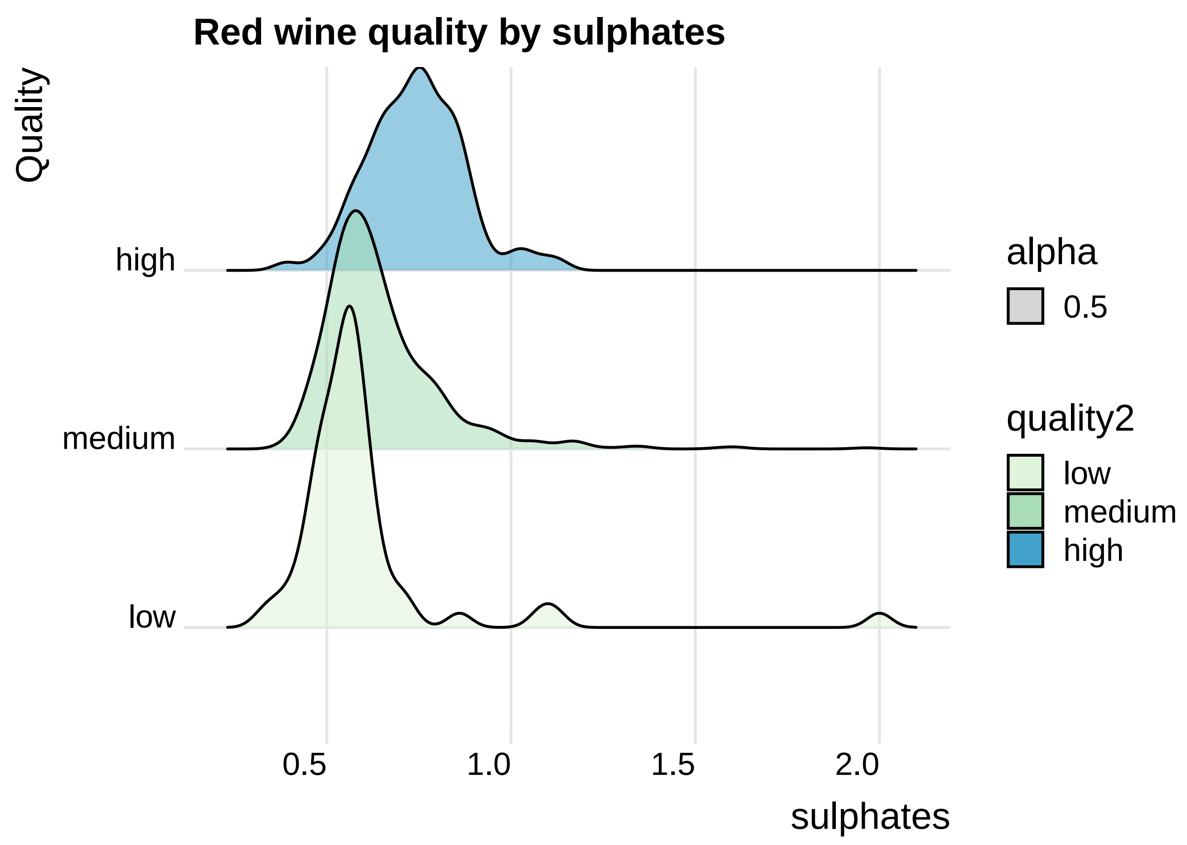
```
##
## [[8]]
```


Red wine quality by density
Quality
high
medium
low
0.990
0.995
1.000
1.005
density
alpha
0.5
quality2
low
medium
high

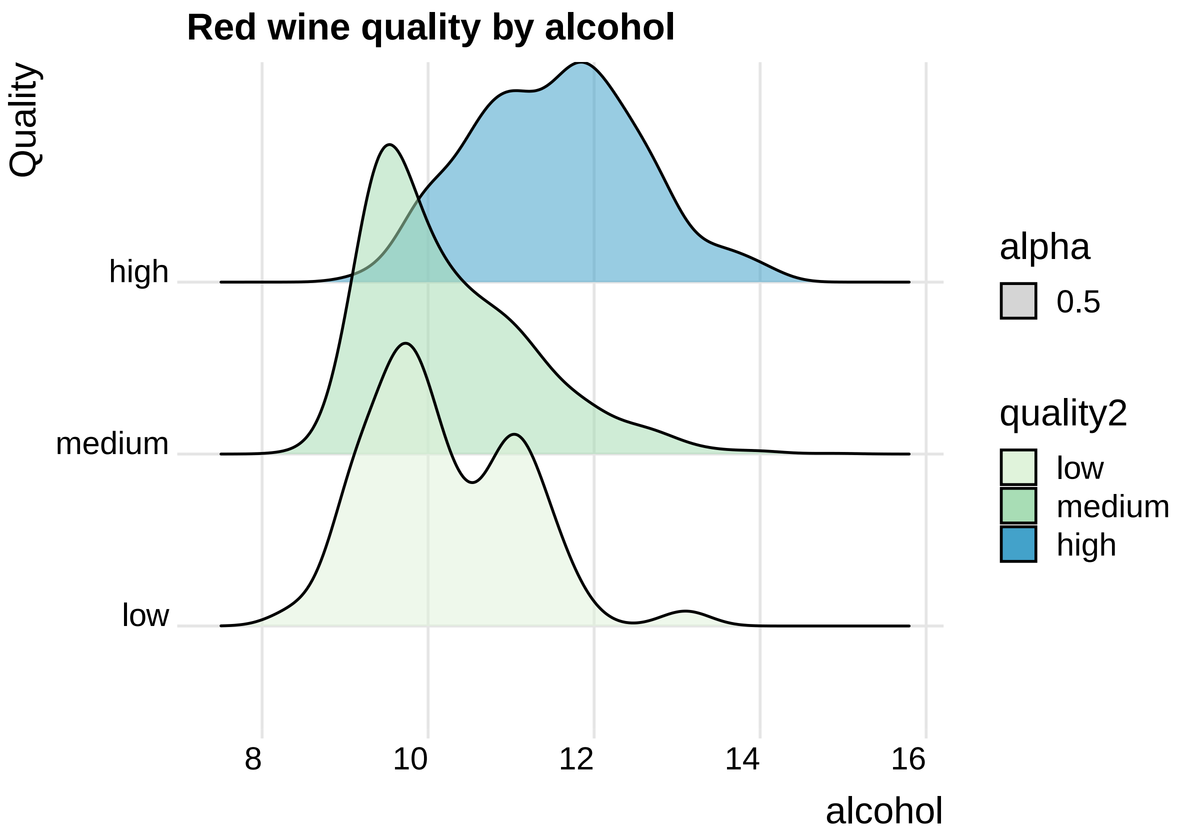
```
##
## [[9]]
```


Red wine quality by pH
Quality
high
medium
low
3.0
3.5
4.0
pH
alpha
0.5
quality2
low
medium
high

```
##
## [[10]]
```


Red wine quality by sulphates
Quality
high
medium
low
0.5
1.0
1.5
2.0
sulphates
alpha
0.5
quality2
low
medium
high

```
##
## [[11]]
```



##3.9. Correlation:##

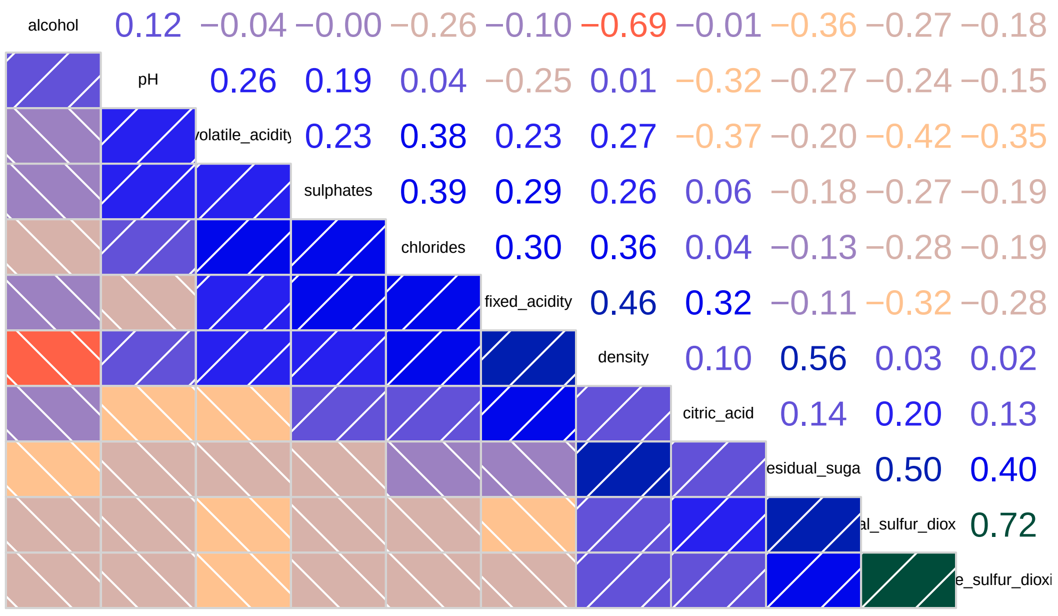
Correlogram is a graph of correlation matrix. It is very useful to highlight the most correlated variables in a data table. In this plot, correlation coefficients is colored according to the value. Correlation matrix can be also reordered according to the degree of association between variables. The exploration done so far shows that volatile acid, chlorides and total sulfur dioxide are good candidates as wine type predictors. The prediction may be lower than expected if any two of these variables are highly correlated.

The correlogram shows the correlation of all predictors in pairs. The diagonal contains the variable names. The correlation of two variables are in the intersection of the variables rows and columns. The color intensity of the boxes and numbers indicate the correlation degree: strong colors indicate high correlation, whereas light colors indicate low correlation.

The assumption in this analysis is that low correlation is between -0.5 and 0.5, and high correlation otherwise. Most colors in the correlogram are light and the correlations are low.

```
## $corrgram
## NULL
```

Correlogram: Wine Physicochemical Properties



```
## $huxtable
## NULL
```

The table below summarizes the pairs of variables with high correlation.

The correlations of volatile acid, chlorides and total sulfur dioxide are low as against the highly correlated variables listed in the table.

```
# Calculate the correlation of all predictors
my_cor <- as.data.frame(cor(train_set[,xcol]))

# Row (r) and column (c) numbers of the correlation matrix
# filtered by high correlated features (cor <= -0.5 or cor >= 0.5)
r <- which(my_cor <=-0.5 | my_cor >= 0.5 & my_cor != 1, arr.ind=TRUE)[,"row"]
c <- which(my_cor <=-0.5 | my_cor >= 0.5 & my_cor != 1, arr.ind=TRUE)[,"col"]

# Create a table with high correlations features only
my_cor_hux <- hux(`Feature 1` = variable_names[r],
                  `Feature 2` = variable_names[c],
                  Correlation = sapply(1:length(r), function(x)
                    my_cor[r[x],c[x]]),
                 add_colnames = TRUE) %>%
  # Format the table
  set_bold(row = 1, everywhere, value = TRUE)          %>%
  set_top_border(row = 1, everywhere, value = 1)    %>%
  set_bottom_border(row = c(1,7), everywhere, value = 1)    %>%
  set_align(row = 1, col = 2:3, value = 'center') %>%
  set_number_format(row = 2:7, col = 3, value = 3)              %>%
  set_caption('High Correlated Features') %>%
  set_position(value = "center")

# Show the table
my_cor_hux
```

Table 7: High Correlated Features

Feature 1	Feature 2	Correlation
Density	Residual sugar	0.556
Total sulfur dioxide	Free sulfur dioxide	0.720
Free sulfur dioxide	Total sulfur dioxide	0.720
Residual sugar	Density	0.556
Alcohol	Density	-0.685
Density	Alcohol	-0.685

```
# Previously, we identified 3 features that may be used in prediction
# of wine type. We create a table to check if the correlation
# between each pair is low.


# Calculate the correlations for volatile acid, chlorides and total sulfur dioxide

# Variable names
pred1 <- c("volatile_acidity", "chlorides", "total_sulfur_dioxide")

# Nice variable names
xpred2 <- c("Volatile acidity", "Chlorides", "Total sulfur dioxide")

# Calculate the correlation of all predictors
my_cor2 <- as.data.frame(cor(train_set[,pred1]))
```

```r
# Create a table with high correlations features only
cor_hux2 <- hux(cor(train_set[,pred1]),
                add_colnames  = FALSE,
                add_rownames = FALSE)

# Set row and column names
rownames(cor_hux2) <- xpred2
colnames(cor_hux2) <- xpred2
cor_hux2 <- add_rownames(cor_hux2, colname = "Feature")
cor_hux2 <- add_colnames(cor_hux2, value = TRUE)

# Format the table
cor_hux2 <- cor_hux2 %>%
  set_bold(row = 1, everywhere, value = TRUE)          %>%
  set_top_border(row = 1, everywhere, value = 1)    %>%
  set_bottom_border(row = c(1,4), everywhere, value = 1)    %>%
  set_align(row = 1, col = 2:ncol(my_cor_hux), value = 'center') %>%
  set_number_format(everywhere, everywhere, value = 3)            %>%
  set_caption('Correlation Matrix - Selected Features') %>%
  set_position(value = "center") %>%
  set_width(value = 0.6)

# Show the table
cor_hux2
```

Table 8: Correlation Matrix - Selected Features

Feature	**Volatile acidity**	**Chlorides**	**Total sulfur dioxide**
Volatile acidity	1.000	0.378	-0.416
Chlorides	0.378	1.000	-0.276
Total sulfur dioxide	-0.416	-0.276	1.000

##3.10. Modeling##

It is observed in data exploration that total sulfur dioxide, chlorides and volatile acidity are good candidates for wine type prediction. The AUC is very high, the distributions have low overlapping areas and the correlations are low.

Although the quality prediction of red wines may be challenging due to the fact that some features with high AUC have low correlations and all features present large distribution overlapping areas. Further more, the prevalence of wines with qualities 3, 4 and 8 is very low.

Subsequently, different modeling approaches would be utilized to predict red and white wines and wine quality.

#3.10.1. Simple Model:#

The simplest model is based on the assumption that all wines are red or white. Since there are more white than red wines, the model predicts all wines are white. The accuracy at 75.4 is higher than 50%, the specificity is 1 and sensitivity 0. The model is good at predicting white wines, but would not be very useful at predicting red wines. A better model should have higher sensitivity, although it may lower specificity and accuracy.

#3.10.2. Random Model:#

The second model uses the sample probabilities of each wine type as a predictor. The ubiquity of the red and white wines is 24.6% and 75.4% respectively, so the red or white is randomly assigned using the proportions. The problem experienced with this approach is that the actual distribution in the population is not known, and the actual distribution may fluctuate over time. To have a better performance any machine learning model should perform better than both models.

#3.10.3. The Single Predictor:#

Since the distributions of total sulfur dioxide, chlorides and volatile acidity stratified by wine type have small overlapping areas, we can find a cutoff value that maximizes the conditional probability.

Pr(Y=k|X=x) Where Y is the outcome and p is the number of predictors.

because three predictors have been selected during data exploration, our model therefore becomes:

Pr(Y=k|X1=x1,X2=x2,X3=x3)=Î²0+Î²1x1+Î²2x2+Î²3x3

where x1 is total sulfur dioxide, x2 is chlorides and x3 is volatile acidity.

Comparatively the difference with the previous model is that linear regression builds a function that best fits the data.

#3.10.4. The K-Nearest Neighbors (KNN):#

The conditional probability are estimated using the k-nearest neighbors algorithm: p(x1,..,xp)=Pr(Y=k|X1=x1,..,Xp=xp) The algorithm calculates the euclidean distance of all predictors, for any point (x1,..,xp) in the multidimensional space that is being predicted, the algorithm determines the distance to k points. Hence, the k nearest points is refereed to as neighborhood.

For k=1 the algorithm finds the distance to a single neighbor. For k equals to the number of samples, the algorithm uses all points. Hence, k is a tuning parameter that can be calculated running the algorithm for several values of k and picking the result with highest accuracy or AUC.

#3.10.4. Classification and Regression Trees:#

The Classification and regression tree(CART) model is one of the oldest and most fundamental algorithms. It is used to predict outcomes based on certain predictor variables that is explaining how a target variableâ€™s values can be predicted based on other values. They are seen as a decision tree where each fork is a split in a predictor variable and each node at the end has a prediction for the target variable. The tree is basically a flow chart of yes or no questions.

However, they do not have the best accuracy, they are hard to train and they are unstable to changes in the data, but are also the basis for other powerful machine learning algorithms like random forest and boosted decision trees.

#3.10.4. Random Forest:#

Random forests or random decision forests are an ensemble learning model for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Random forests improve prediction performance over classification trees by averaging multiple decision trees. The algorithm creates several random subsets of the original data, in this case the training set, and calculates the classification trees, then the final result is the average of all trees.

#3.10.5. Linear and Quadratic Discriminant Analysis:#

Linear discriminant analysis (LDA) is particularly popular because it is both a classifier and a dimensionality reduction technique. Quadratic discriminant analysis (QDA) is a variant of LDA that allows for non-linear separation of data.

In LDA and QDA the assumption is that all predictors are normally distributed and have the same standard deviations.The decision boundary of LDA is linear, while QDA is an extension of LDA with quadratic boundaries. They are all used to solving classification problems where the output variable is categorical. LDA supports both binary and multi-class classification.

The LDA and QDA can be derived from simple probabilistic models which model the class conditional distribution of the data P(X|y=k) for each class k. Predictions can then be obtained by using Bayesâ€™ rule:

Pr(y=k|X)=Pr(X|y=k)Pr(y=k)Pr(X)=Pr(X|y=k)Pr(y=k)âˆ‘lPr(X|y=l)â‹…Pr(y=l) and we select the class k which maximizes this conditional probability.

LDA and QDA work better with few predictors.

#3.10.6. Cross Validation:#

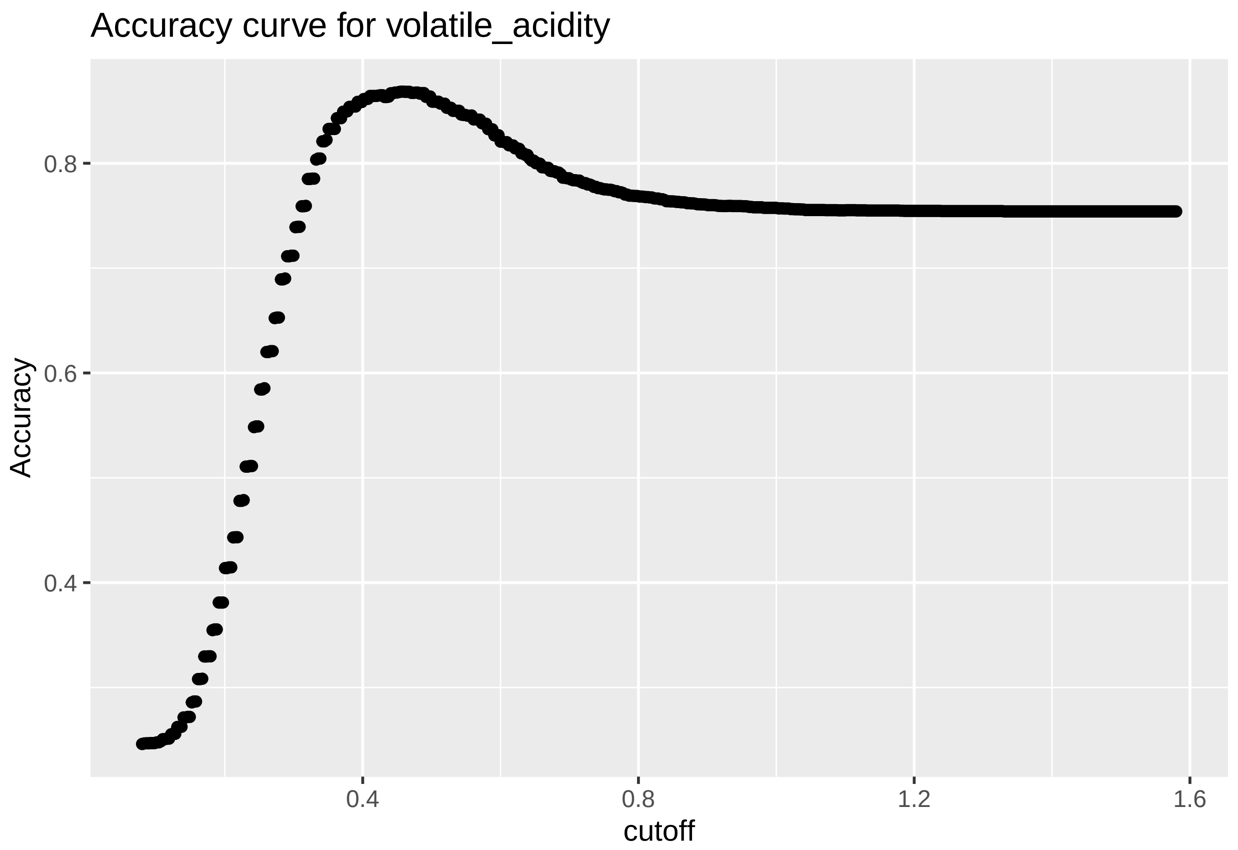
Cross-validation is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. Use cross-validation to detect overfitting.In ML, you can use the k-fold cross-validation method to perform cross-validation. In k-fold cross-validation, you split the input data into k subsets of data (also known as folds).

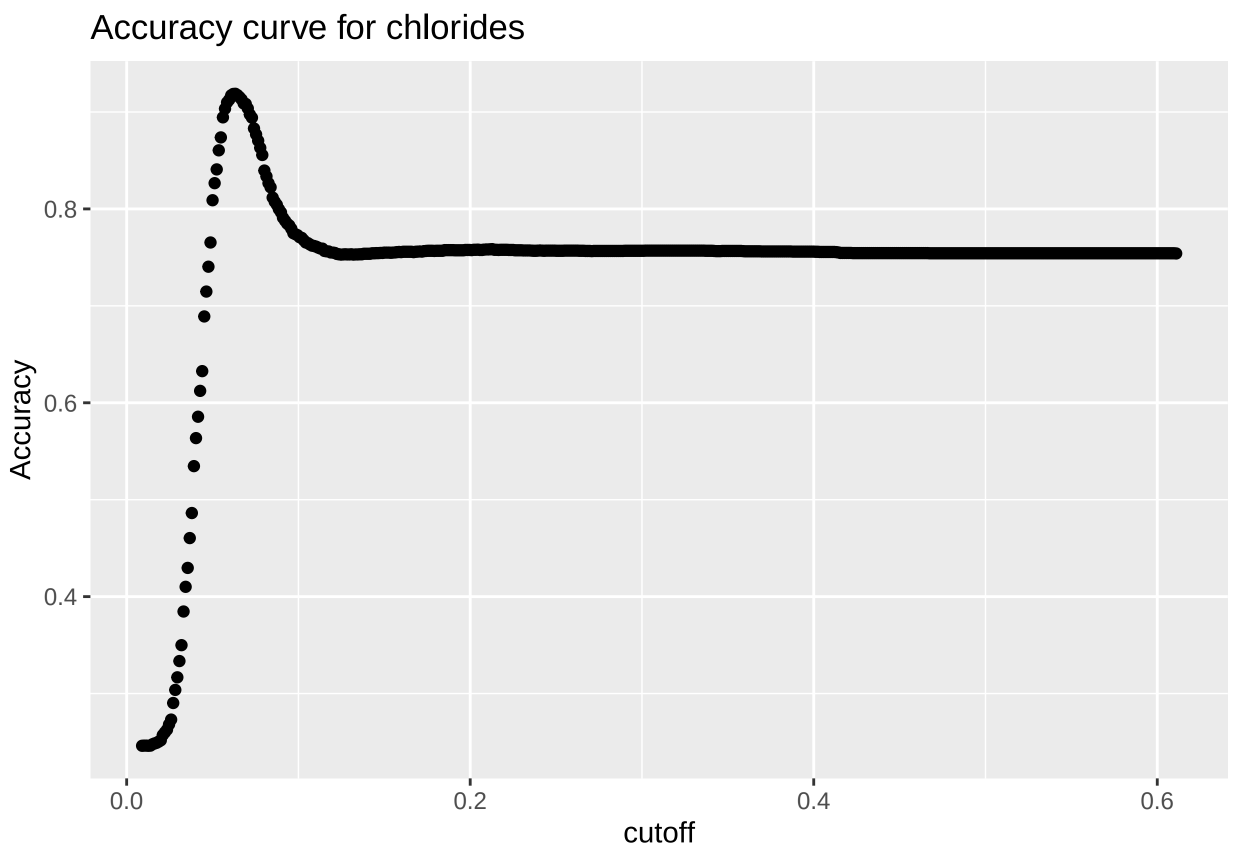
The original dataset is partitioned in the training set used to train the model, and the test set used to predict the values with the trained model. Cross validation partitions the training set in the same way and performs the training and prediction several times. Then, the result with the best RMSE, accuracy, AUC or the chosen metric is selected. This process can be used in conjunction to tuning parameters, for example picking the best k number of neighbors in knn.

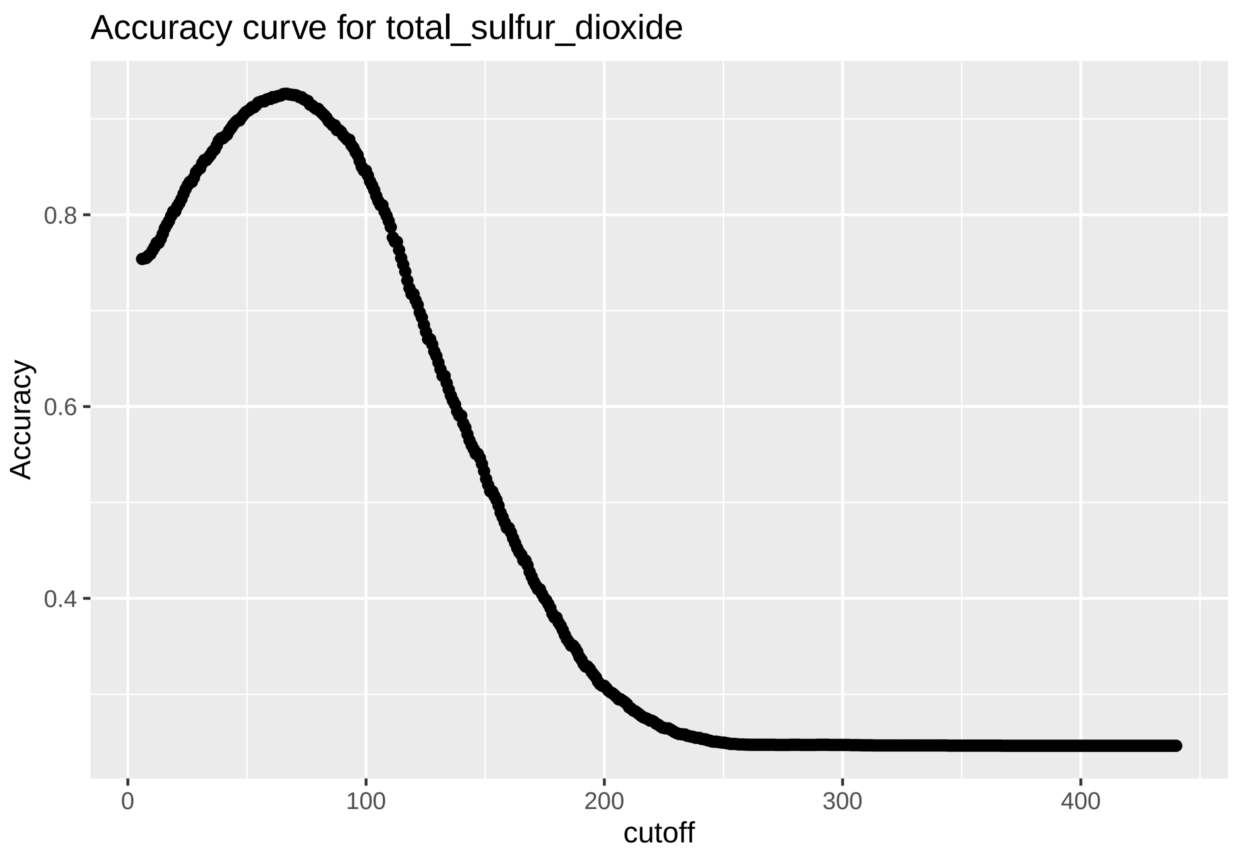
#3.10.7 Ensemble:#

Ensemble is a technique that combines several base models in order to produce one optimal predictive model. Combining the results of several predictions may improve prediction accuracy. This method is called ensemble and consists in selecting the most common class for a given set of observations.

For example, suppose the predictions of wine type for three models as illustrated in the table below. The ensemble is just the majority of votes of the combined models.




Accuracy curve for chlorides
Accuracy
0.4
0.6
0.8
0.0
0.2
0.4
0.6
cutoff



```
# Remove first row with NA
t_results <- t_results[2:nrow(t_results),]

# Combine the results using majority of votes
p_hat_ens <-as_factor(data.frame(preds) %>%
                        mutate(x = as.numeric(preds[,1] == "red") +
                                 as.numeric(preds[,2] == "red") +
                                 as.numeric(preds[,3]  == "red"),
                               p_hat = ifelse(x >=2, "red", "white")) %>%
                        pull(p_hat))

# Update results table
t_results <<- rbind(t_results,
                       data.frame(Feature = "Ensemble",
                                  Accuracy = mean(p_hat_ens == test_set$type),
                                  Sensitivity = sensitivity(p_hat_ens, test_set$type),
                                  Specificity = specificity(p_hat_ens, test_set$type),
                                  stringsAsFactors = FALSE))
as_hux(t_results,
       add_colnames = TRUE) %>%
  # Format header row
  set_bold(row = 1, everywhere, value = TRUE)          %>%
  set_top_border(row = 1, everywhere, value = 1)       %>%
  set_bottom_border(row = c(1,5), everywhere, value = 1)    %>%
  # Format numbers
  set_number_format(row = everywhere, col = 2:4, value = 3)  %>%
  # Format alignment
  set_align(row = everywhere, col = 1,   value = 'left')  %>%
  set_align(row = everywhere, col = 1:4, value = 'right') %>%
  set_caption("Best Performance for Combined Predictions")  %>%
  set_position(value = "center")
```

Table 9: Best Performance for Combined Predictions

Feature	Accuracy	Sensitivity	Specificity
volatile_acidity	0.868	0.638	0.947
chlorides	0.919	0.894	0.947
total_sulfur_dioxide	0.926	0.794	0.971
Ensemble	0.969	0.894	0.994

##3.11. Linear Regression and KNN:## Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Linear regression approximates the multidimensional space of predictors X and outcomes Y into a function. The outcome, which is wine type, is categorical, so we need to convert to number before building the function. Now, 1 would be assigned to red wine and 0 to white wine.

The predicted values are also numeric, so we need to convert back to categories.

Again, only the three main features would be used to predict wine type: total sulfur dioxide, chlorides and volatile acidity.

```
#-------------------
# Linear Regression
#-------------------
# Predict wine type with total_sulfur_dioxide + chlorides + volatile_acidity

# Train the linear regression model
ft_lm <- train_set %>%
  # Convert the outcome to numeric
  mutate(type = ifelse(type == "red", 1, 0)) %>%
  # Fit the model
  lm(type ~ total_sulfur_dioxide + chlorides + volatile_acidity, data = .)
```

```r
# Predict
f_hat_lm <- predict(ft_lm, newdata = test_set)

# Convert the predicted value to factor
p_hat_lm <- factor(ifelse(f_hat_lm > 0.5, "red", "white"))

# Evaluate the results
caret::confusionMatrix(p_hat_lm, test_set$type)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction red white
##      red   151     7
##      white   9   483
##
##                Accuracy : 0.975
##                  95% CI : (0.96, 0.986)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.933
##
##  Mcnemar's Test P-Value : 0.803
##
##             Sensitivity : 0.944
##             Specificity : 0.986
##          Pos Pred Value : 0.956
##          Neg Pred Value : 0.982
##              Prevalence : 0.246
##          Detection Rate : 0.232
##    Detection Prevalence : 0.243
##       Balanced Accuracy : 0.965
##
##        'Positive' Class : red
##
```

```r
#-------------------
# Knn
#-------------------
# Predict wine type with all features
# Train
ft_knn <- knn3(formula = pml, data = train_set, k = 5)
```

```r
# Predict
p_knn <- predict(object = ft_knn,
                 newdata = test_set,
                 type ="class")

# Compare the results: confusion matrix
caret::confusionMatrix(data = p_knn,
                       reference = test_set$type,
                       positive = "red")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction red white
##      red   145    15
##      white  15   475
##
##                Accuracy : 0.954
##                  95% CI : (0.935, 0.969)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.876
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.906
##             Specificity : 0.969
##          Pos Pred Value : 0.906
##          Neg Pred Value : 0.969
##              Prevalence : 0.246
##          Detection Rate : 0.223
##    Detection Prevalence : 0.246
##       Balanced Accuracy : 0.938
##
##        'Positive' Class : red
##
```

```r
# F1 score
F_meas(data = p_knn, reference = test_set$type)
```

```
## [1] 0.906
```

##3.12. Regression tree - rpart:##

This is a regression part of CART. The basic regression trees partition a data set into smaller subgroups and then fit a simple constant for each observation in the subgroup. The partitioning is achieved by successive binary partitions also known as recursive partitioning, based on the different predictors. The constant to predict is based on the average response values for all observations that fall in that subgroup. The regression tree for this studies uses all features to predict wine type.

```
#-------------------
# Regression tree
#-------------------
# Predict wine type with all features

# The "rpart" package trains regression trees and
# "rpart.plot" plots the tree
load_lib(c("rpart", "rpart.plot"))
```

```
## Loading required package: rpart
```

```
## Loading required package: rpart.plot
```

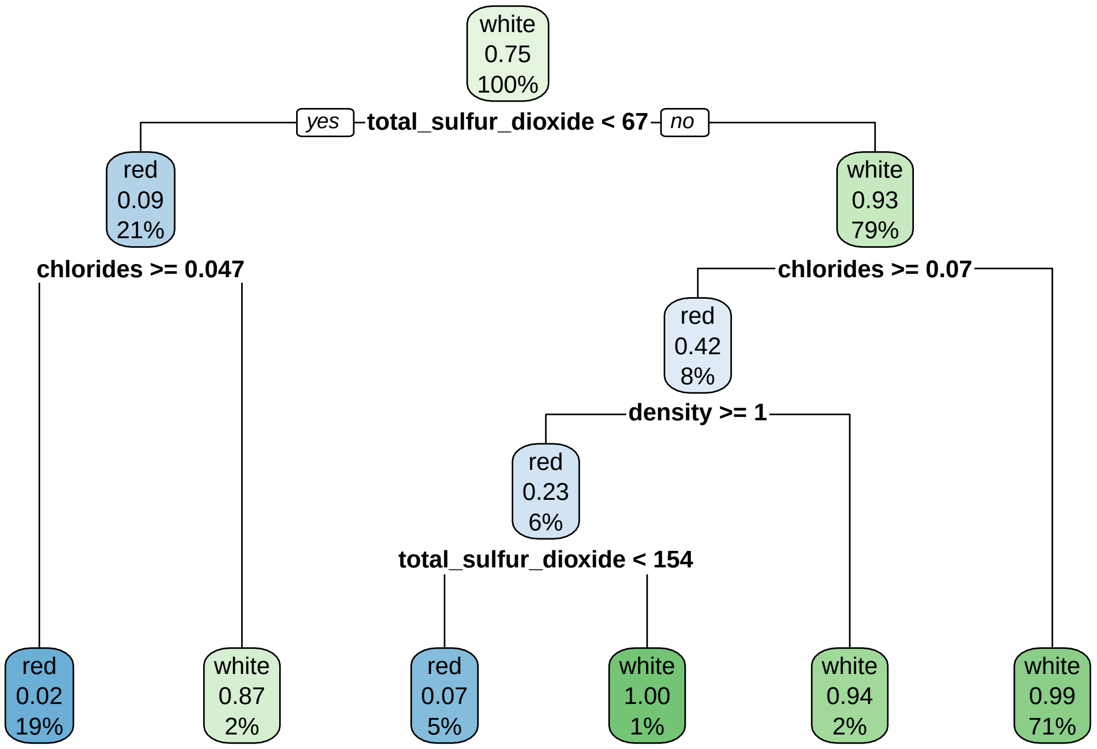
```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

```
## $rpart
## NULL
##
## $rpart.plot
## NULL
```

```
# Train the model
ft_rpart <- rpart::rpart(formula = pml,
                          method = "class",
                          data = train_set)
# Predict
p_rpart <- predict(object = ft_rpart,
                   newdata = test_set,
                   type = "class")

# Compare the results: confusion matrix
caret::confusionMatrix(data = p_rpart,
                       reference = test_set$type,
                       positive = "red")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction red white
##      red   154     2
##      white   6   488
##
##                Accuracy : 0.988
##                  95% CI : (0.976, 0.995)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.967
##
##  Mcnemar's Test P-Value : 0.289
##
##             Sensitivity : 0.963
##             Specificity : 0.996
##          Pos Pred Value : 0.987
##          Neg Pred Value : 0.988
##              Prevalence : 0.246
##          Detection Rate : 0.237
##    Detection Prevalence : 0.240
##       Balanced Accuracy : 0.979
##
##        'Positive' Class : red
##
```

```
# Plot the result
rpart.plot(ft_rpart)
```



```r
# F1 score
F_meas(data = p_rpart, reference = test_set$type)
```

```
## [1] 0.975
```

```r
# Variable importance
caret::varImp(ft_rpart)
```

```
##                       Overall
## chlorides                1773
## density                   259
## fixed_acidity             186
## free_sulfur_dioxide       548
## pH                         39
## residual_sugar            139
## sulphates                 615
## total_sulfur_dioxide     1595
## volatile_acidity         1235
## citric_acid                 0
## alcohol                     0
```

Alcohol and citric acid have no importance at all while chlorides, total sulfur dioxide and volatile acidity are the three most important variables.

Each node shows:

a. the predicted class (red or white),
b. the predicted probability of that class,
c. the percentage of observations in the node. The intensity of a nodeâ€™s color is proportional to the value predicted at the node hence, the resulting tree represents the rule to predict the different wine types.

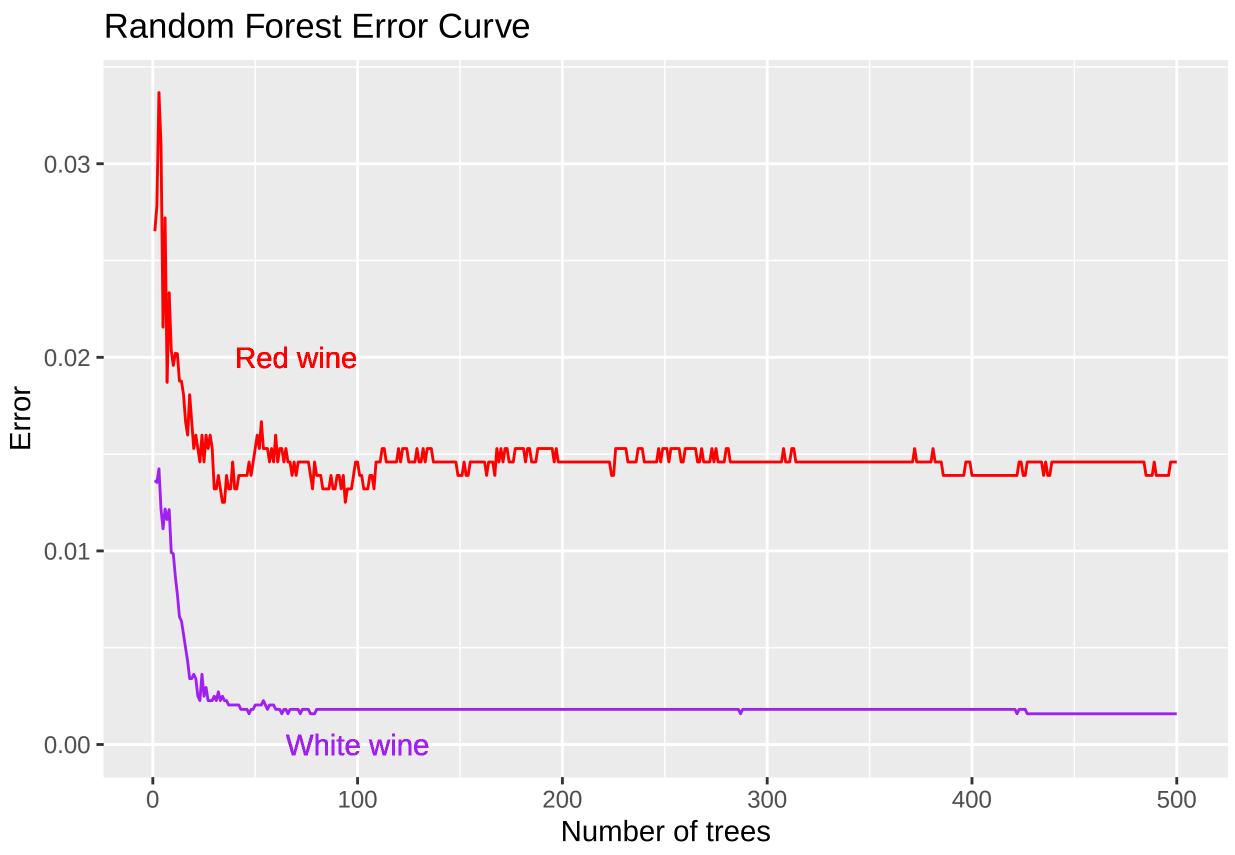
##3.13. Random Forest:## From the result of the random forest high sensitivity and specificity achieved.

```
## $randomForest
## NULL

## Confusion Matrix and Statistics
##
##           Reference
## Prediction red white
##      red   158     0
##      white   2   490
##
##                Accuracy : 0.997
##                  95% CI : (0.989, 1)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.992
##
##  Mcnemar's Test P-Value : 0.48
##
##             Sensitivity : 0.988
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 0.996
##              Prevalence : 0.246
##          Detection Rate : 0.243
##    Detection Prevalence : 0.243
##       Balanced Accuracy : 0.994
##
##        'Positive' Class : red
##
```

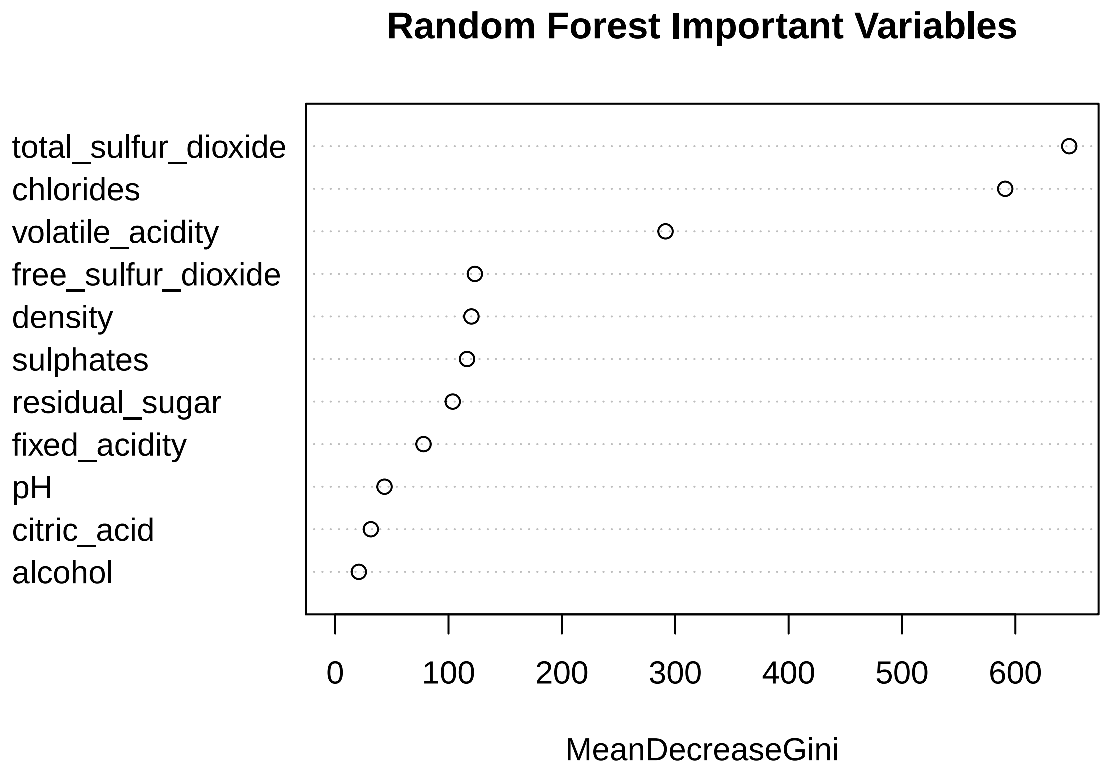
```r
# F1 score
F_meas(data = p_rf, reference = test_set$type)
```

```
## [1] 0.994
```

The error for red wines is higher than white wines this is as a result of lower ubiquity or prevalence.The error decreases as the number of trees grows, stabilizing around 50 trees.



The result shows that alcohol, citric acid and pH are the three most important variables for random forest prediction while chlorides and volatile acidity have low predictive power .



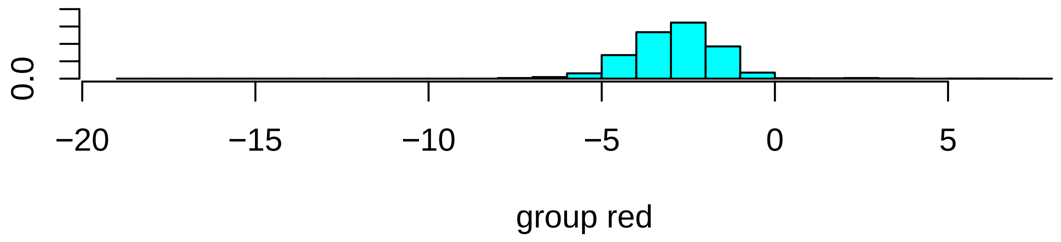
##3.14 Linear Discriminant Analysis - LDA:## LDA uses all features to predict wine type.The group of the red wine fell within the negative scale (-5 to 0) while the white wine fell within positive scale (0 to 5) indicating higher ubiquity or prevalence in the white wine.

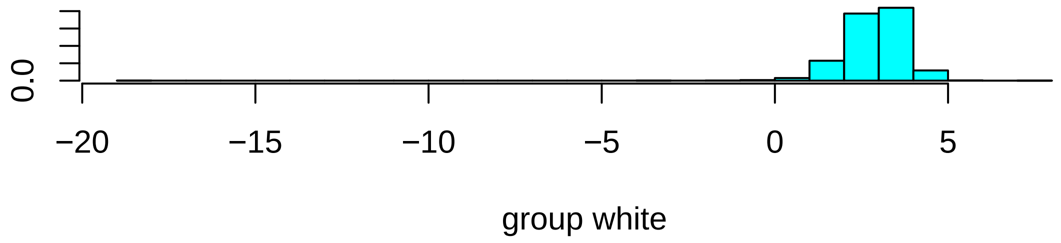
```
## $MASS
## NULL
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction red white
##      red   158     0
##      white   2   490
##
##                Accuracy : 0.997
##                  95% CI : (0.989, 1)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.992
##
```

```
##  Mcnemar's Test P-Value : 0.48
##
##             Sensitivity : 0.988
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 0.996
##              Prevalence : 0.246
##          Detection Rate : 0.243
##    Detection Prevalence : 0.243
##       Balanced Accuracy : 0.994
##
##        'Positive' Class : red
##
```

```
## [1] 0.994
```





##3.15 Quadratic Discriminant Analysis - QDA:##

QDA uses all features to predict wine type, the result further suggest that when red is predicted to be 158 white wine is predicted to be 486 and the accuracy is given as 0.991 and an F score of 98 percent.

```r
#-------------------
# QDA
#-------------------
# Predict wine type with all features
load_lib(c("MASS", "scales"))
```

```
## Loading required package: scales
```

```
## Warning: package 'scales' was built under R version 3.6.3
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:viridis':
##
##     viridis_pal
```

```
## The following object is masked from 'package:huxtable':
##
##     number_format
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
## The following object is masked from 'package:memisc':
##
##     percent
```

```
## $MASS
## NULL
##
## $scales
## NULL
```

```r
# Train the model
ft_qda <- qda(formula = pml, data = train_set)

# Predict
p_qda <- predict(object = ft_qda, newdata = test_set)

# Compare the results: confusion matrix
caret::confusionMatrix(data = p_qda[[1]],
                       reference = test_set$type,
                       positive = "red")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction red white
##      red   158     4
##      white   2   486
##
##                Accuracy : 0.991
##                  95% CI : (0.98, 0.997)
##     No Information Rate : 0.754
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.975
##
##  Mcnemar's Test P-Value : 0.683
##
##             Sensitivity : 0.988
##             Specificity : 0.992
##          Pos Pred Value : 0.975
##          Neg Pred Value : 0.996
##              Prevalence : 0.246
##          Detection Rate : 0.243
##    Detection Prevalence : 0.249
##       Balanced Accuracy : 0.990
##
##        'Positive' Class : red
##
```

```r
# F1 score
F_meas(data = p_qda[[1]], reference = test_set$type)
```

```
## [1] 0.981
```

##3.16 Prediction Results Evaluation:##

Random forest and Linear discriminant analysis (LDA) have the best performance, although the single predictor model and linear regression models used three features as predictors, they showed better performance than knn that used all the features.

```
##               Model Accuracy Sensitivity Specificity
## 1  Single predictor    96.9%       89.4%       99.4%
## 2 Linear Regression    97.5%       94.4%       98.6%
## 3               Knn    95.4%       90.6%       96.9%
## 4  Regression trees    98.8%       96.2%       99.6%
## 5     Random forest    99.7%       98.8%      100.0%
## 6               LDA    99.7%       98.8%      100.0%
## 7               QDA    99.1%       98.8%       99.2%
```

##3.17. Cross Validation and Ensemble:##

All models used previously ran on a single partition of the training set, thereby unable to perfectly identify the different wine type therefore the application of cross validation on 11 classification algorithms and then combining the results would be deployed.

```
## $pROC
## NULL
##
## $plotROC
## NULL

## $e1071
## NULL
##
## $dplyr
## NULL
##
## $fastAdaboost
## NULL
##
## $gam
## NULL
##
## $gbm
## NULL
##
## $import
## NULL
##
```

```
## $kernlab
## NULL
##
## $kknn
## NULL
##
## $klaR
## NULL
##
## $MASS
## NULL
##
## $mboost
## NULL
##
## $mgcv
## NULL
##
## $monmlp
## NULL
##
## $naivebayes
## NULL
##
## $nnet
## NULL
##
## $plyr
## NULL
##
## $ranger
## NULL
##
## $randomForest
## NULL
##
## $Rborist
## NULL
##
## $RSNNS
## NULL
##
## $wsrf
## NULL
```

#3.17.1. Train the models:#

At this point the models would be trained, predictions made, stats calculated and stored in â€˜resultsâ€™ table. Hence the use of standard tuning parameters for all models except knn and random forest.

```
#--------------------------------
# Start parallel processing
#--------------------------------
# The 'train' function in the 'caret' package allows the use of
# parallel processing. Here we enable this before training the models.
# See this link for details:
# http://topepo.github.io/caret/parallel-processing.html
cores <- 4    # Number of CPU cores to use
# Load 'doParallel' package for parallel processing
load_lib("doParallel")
```

```
## $doParallel
## NULL
```

```
cl <- makePSOCKcluster(cores)
registerDoParallel(cl)
set.seed(1234, sample.kind = "Rounding")
# Formula used in predictions
pml <- as.formula(paste("type", "~",
                         paste(xcol, collapse=' + ')))

# Run predictions
preds <- sapply(models, function(model){

  if (model == "knn") {
    # knn use custom tuning parameters
    grid <- data.frame(k = seq(3, 50, 2))
    ft <- caret::train(form = pml,
                       method = model,
                       data = train_set,
                       trControl = contrl,
                       tuneGrid = grid)
  } else if (model == "rforest") {
    # Random forest use custom tuning parameters
    t_grid <- data.frame(mtry = c(1, 2, 3, 4, 5, 10, 25, 50, 100))

    ft <- caret::train(form = pml,
                       method = "rforest",
```

```r
                       data = train_set,
                       trControl = contrl,
                       ntree = 150,
                       tuneGrid = t_grid,
                       nSamp = 5000)
} else {
  # Other models use standard parameters (no tuning)
  ft <- caret::train(form = pml,
                       method = model,
                       data = train_set,
                       trControl = contrl)
}

# Predictions
pred <- predict(object = ft, newdata = test_set)

# Accuracy
acc <- mean(pred == test_set$type)

# Sensitivity
sen <- sensitivity(data = pred,
                    reference = test_set$type,
                    positive = "red")
# Specificity
spe <- specificity(data = pred,
                    reference = test_set$type,
                    positive = "red")

# F1 score
f1 <- F_meas(data = factor(pred), reference = test_set$type)

# AUC
auc_val <- auc(ft$pred$obs, ft$pred$red)

# Store stats in 'results' table
results <<- rbind(results,
                  tibble(
                    Model = model,
                    Accuracy = acc,
                    Sensitivity = sen,
                    Specificity = spe,
                    AUC = auc_val,
                    F1_Score = f1))
  # The predictions will be used for ensemble
  return(pred)
})
```

```
## # weights:  13 (12 variable)
## initial  value 4052.831565
## iter  10 value 582.282216
## iter  20 value 286.472172
## iter  30 value 286.232096
## iter  40 value 285.106941
## final  value 285.105099
## converged
```

```
# Remove the first row of 'results' that contains NAs
results <- results[2:(nrow(results)),]
```

The combination of the best results of all models in a single ensemble is presented and the summarry of the results table.

```
#-------------------------------
# Combine all models
#-------------------------------
# Use votes method to ensemble the predictions
votes <- rowMeans(preds == "red")
p_hat <- factor(ifelse(votes > 0.5, "red", "white"))

# Update the 'results' table
results <<- rbind(results,
                  tibble(
                    Model = "Ensemble",
                    Accuracy = mean(p_hat == test_set$type),
                    Sensitivity = sensitivity(p_hat, test_set$type),
                    Specificity = specificity(p_hat, test_set$type),
                    AUC = auc(p_hat, as.numeric(test_set$type)),
                    F1_Score = F_meas(p_hat, test_set$type)))
```

```
## Setting levels: control = red, case = white
```

```
## Setting direction: controls < cases
```

```
as_hux(results,
    add_colnames = TRUE) %>%
  # Format header row
  set_bold(row = 1, everywhere, value = TRUE)          %>%
  set_top_border(row = 1, everywhere, value = 1)       %>%
  set_bottom_border(row = c(1,13), everywhere, value = 1)    %>%
  # Format numbers
  set_number_format(row = -1, col = 2:6, value = 3)  %>%
  # Format alignment
  set_align(row = everywhere, col = 1,   value = 'left')  %>%
  set_align(row = everywhere, col = 2:6, value = 'right') %>%
  # Title
  set_caption('Model Performance With Cross Validation') %>%
  set_position(value = "center")
```

Table 10: Model Performance With Cross Validation

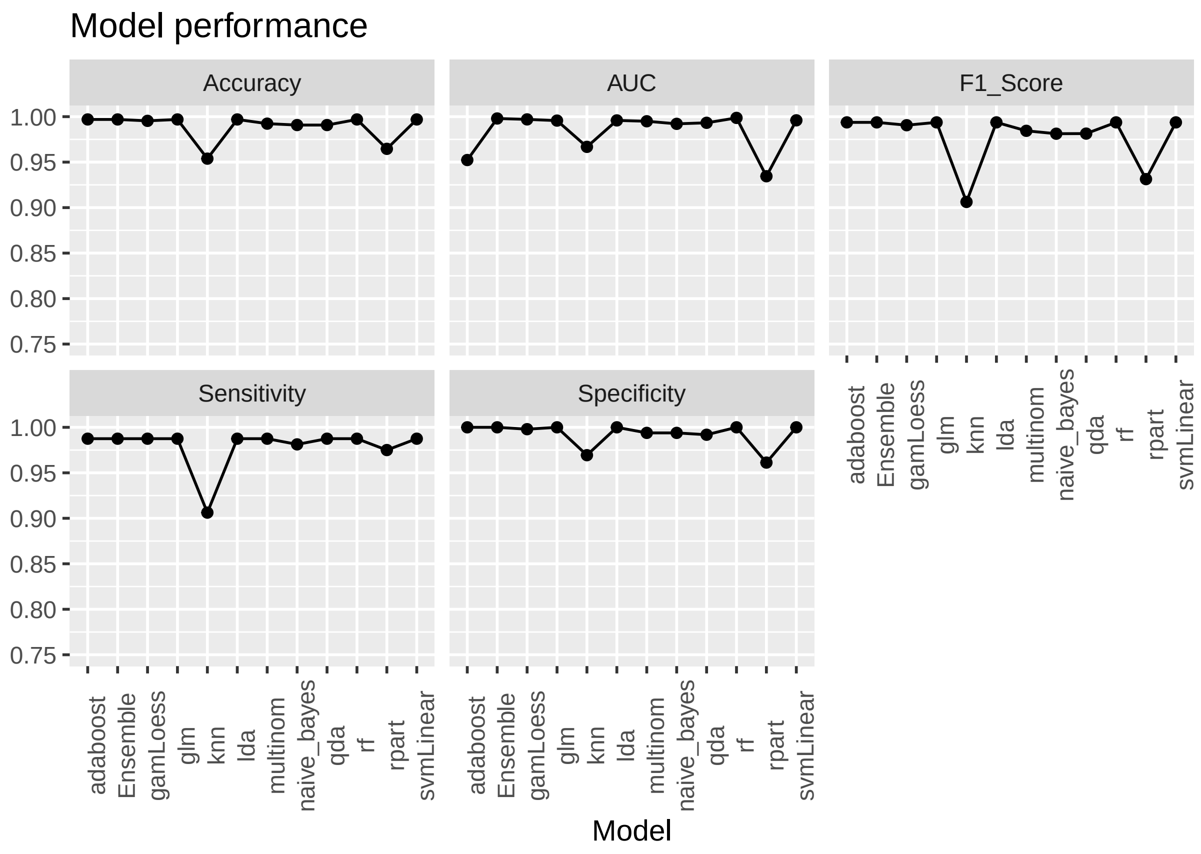
Model	**Accuracy**	**Sensitivity**	**Specificity**	**F1_Score**	**AUC**
glm	0.997	0.988	1.000	0.994	0.996
lda	0.997	0.988	1.000	0.994	0.996
naive_bayes	0.991	0.981	0.994	0.981	0.992
svmLinear	0.997	0.988	1.000	0.994	0.996
rpart	0.965	0.975	0.961	0.931	0.934
knn	0.954	0.906	0.969	0.906	0.967
gamLoess	0.995	0.988	0.998	0.991	0.997
multinom	0.992	0.988	0.994	0.984	0.995
qda	0.991	0.988	0.992	0.981	0.993
rf	0.997	0.988	1.000	0.994	0.999
adaboost	0.997	0.988	1.000	0.994	0.952
Ensemble	0.997	0.988	1.000	0.994	0.998

Generalized linear model (glm) has the best overall performance, than the ensemble.

```
hux(Accuracy    = results[which.max(results$Accuracy),1]$Model,
    Sensitivity = results[which.max(results$Sensitivity),1]$Model,
    Specificity = results[which.max(results$Specificity),1]$Model,
    F_1    = results[which.max(results$F1_Score),1]$Model,
    AUC   = results[which.max(results$AUC),1]$Model,
    add_colnames = TRUE) %>%
    # Format header row
    set_bold(row = 1, col = everywhere, value = TRUE)        %>%
    set_top_border(row = 1, col = everywhere, value = 1)     %>%
    set_bottom_border(row = c(1,2), col = everywhere, value = 1)  %>%
    # Format table
    set_width(value = 0.6)                                   %>%
    set_caption("Best model")                                %>%
    set_position(value = "center")
```

Table 11: Best model

Accuracy	**Sensitivity**	**Specificity**	**F_1**	**AUC**
glm	glm	glm	glm	rf



```
results
```

```
## # A tibble: 12 x 6
##    Model       Accuracy Sensitivity Specificity F1_Score   AUC
##    <chr>          <dbl>       <dbl>       <dbl>    <dbl> <dbl>
##  1 glm            0.997       0.988       1        0.994 0.996
##  2 lda            0.997       0.988       1        0.994 0.996
##  3 naive_bayes    0.991       0.981       0.994    0.981 0.992
##  4 svmLinear      0.997       0.988       1        0.994 0.996
##  5 rpart          0.965       0.975       0.961    0.931 0.934
##  6 knn            0.954       0.906       0.969    0.906 0.967
##  7 gamLoess       0.995       0.988       0.998    0.991 0.997
##  8 multinom       0.992       0.988       0.994    0.984 0.995
##  9 qda            0.991       0.988       0.992    0.981 0.993
## 10 rf             0.997       0.988       1        0.994 0.999
## 11 adaboost       0.997       0.988       1        0.994 0.952
## 12 Ensemble       0.997       0.988       1        0.994 0.998
```

##3.10 Predicting Quality:##

There are 5 samples of quality level 9 in the red and white datasets combined and no sample in the train_set_r dataset while the ubiquity or prevalence of quality levels 3, 4 and 8 is very low in the dataset.

The levels were previously combined to form the low, medium and high quality categories in the train_set_r dataset. The use of cross validation with ensemble would be deployed to predict the combined levels.The models such as glm, gamLoess, qda and adaboost would not work in this case.

```
set.seed(1234, sample.kind = "Rounding")
```

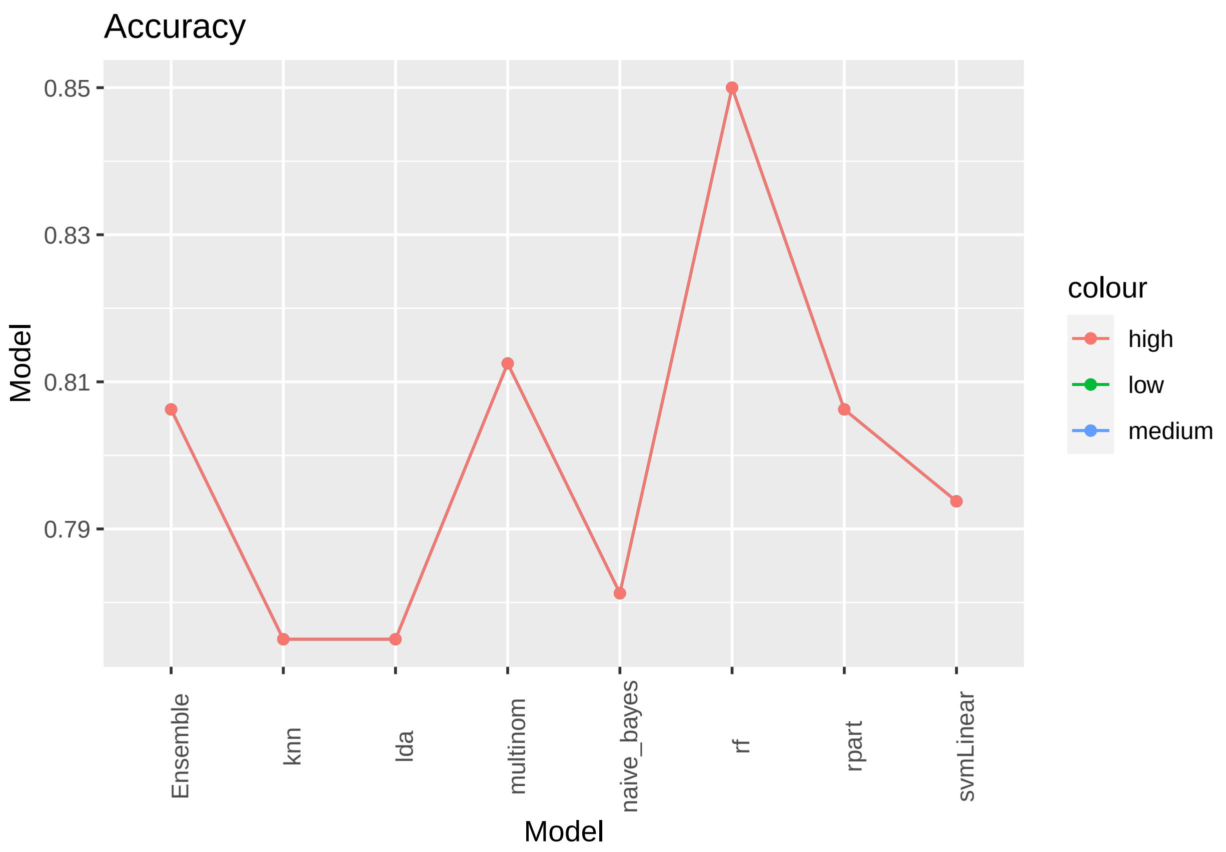
```
## Warning in set.seed(1234, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

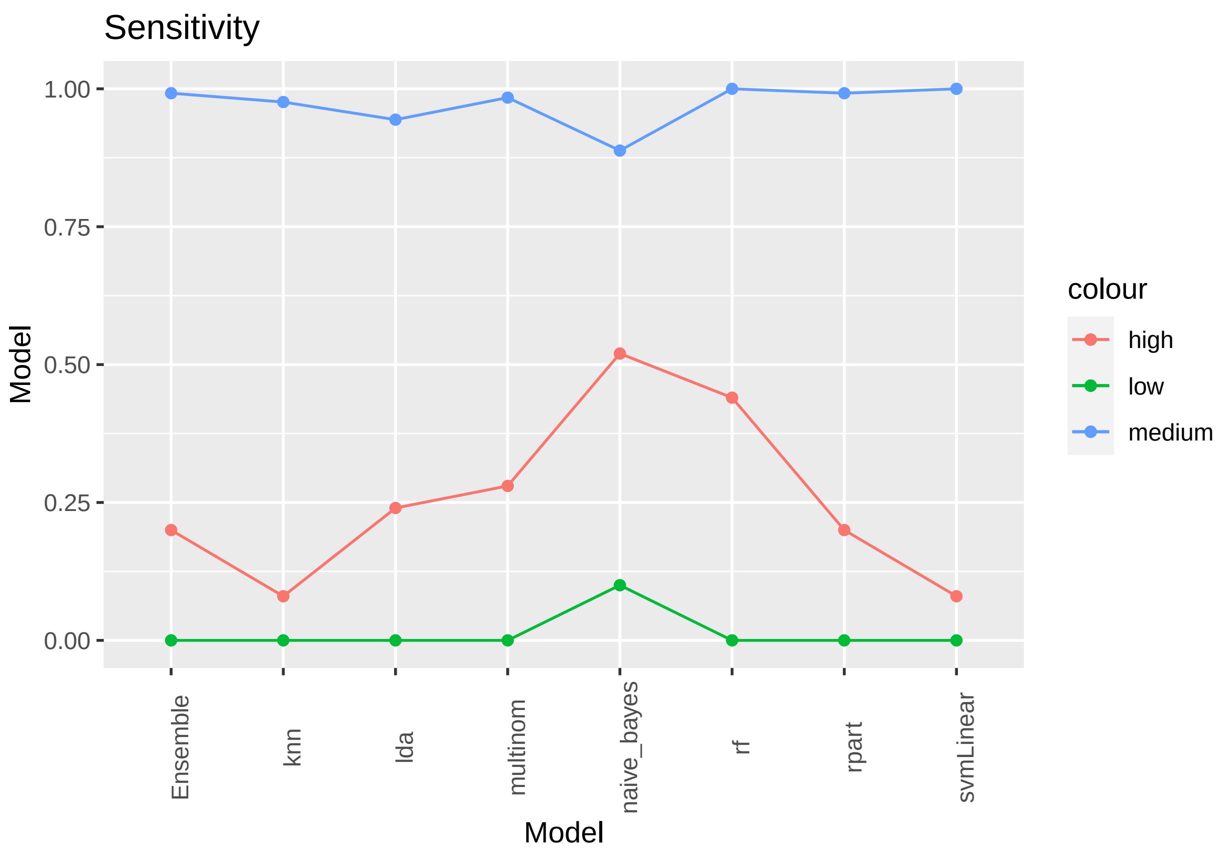
```
# Formula used in predictions
pml_qual <- as.formula(paste("quality2", "~",
                             paste(xcol, collapse=' + ')))

# Define models
#"glm",gamLoess, qda, adaboost
models <- c( "lda", "naive_bayes", "svmLinear", "rpart",
            "knn", "multinom", "rf")

# Create 'results' table. The first row
# contains NAs and will be removed after
# the training
q_results <- tibble(Model = NA,
                          Quality = NA,
                          Accuracy = NA,
                          Sensitivity = NA,
                          Specificity = NA,
                          F1_Score = NA)

preds_q <- sapply(models, function(model){

  print(model)
  if (model == "knn") {
    # knn use custom tuning parameters
    grid <- data.frame(k = seq(3, 50, 2))
    ft <- caret::train(form = pml_qual,
                       method = model,
                       data = train_set_r,
                       trControl = contrl,
                       tuneGrid = grid)
  } else if (model == "rforest") {
    # Random forest use custom tuning parameters
    grid <- data.frame(mtry = c(1, 2, 3, 4, 5, 10, 25, 50, 100))

    ft <- caret::train(form = pml_qual,
                       method = "rforest",
                       data = train_set_r,
                       trControl = contrl,
                       ntree = 150,
                       tuneGrid = grid,
                       nSamp = 5000)
  } else {
    # Other models use standard parameters (no tuning)
    ft <- caret::train(form = pml_qual,
                       method = model,
                       data = train_set_r,
                       trControl = contrl)
  }

  # Predictions
  pred <- predict(object = ft, newdata = test_set_r)

  # Accuracy
  acc <- mean(pred == test_set_r$quality2)
  # Sensitivity
  sen <- caret::confusionMatrix(pred,
                                test_set_r$quality2)$byClass[,"Sensitivity"]
  # Specificity
  spe <- caret::confusionMatrix(pred,
                                test_set_r$quality2)$byClass[,"Specificity"]

  # F1 score
  f1 <- caret::confusionMatrix(pred,
                               test_set_r$quality2)$byClass[,"F1"]

  # Store stats in 'results' table
  q_results <<- rbind(q_results,
                         tibble(Model = model,
                                Quality = levels(test_set_r$quality2),
                                Accuracy = acc,
                                Sensitivity = sen,
                                Specificity = spe,
                                F1_Score = f1))

  # The predictions will be used for ensemble
  return(pred)
})
```

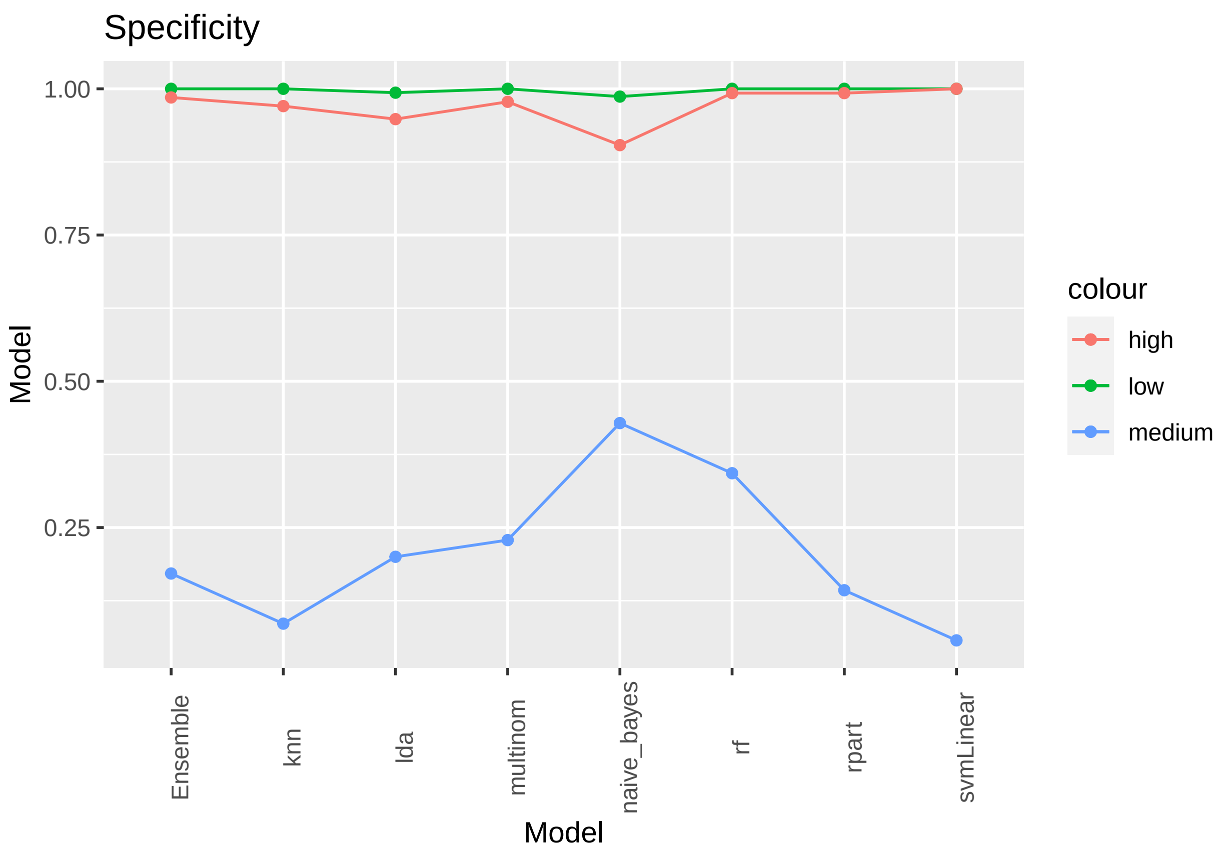
```
## [1] "lda"
## [1] "naive_bayes"
## [1] "svmLinear"
## maximum number of iterations reached 0.000157 -0.000157maximum number of iterations r
## [1] "knn"
## [1] "multinom"
## # weights:  39 (24 variable)
## initial  value 1580.903083
## iter  10 value 869.856753
## iter  20 value 602.009106
## iter  30 value 572.543773
## iter  40 value 571.645387
## iter  50 value 571.590758
## iter  60 value 571.588222
## iter  70 value 571.560614
## iter  80 value 571.367829
## iter  90 value 570.896416
## final  value 570.891543
## converged
## [1] "rf"
```

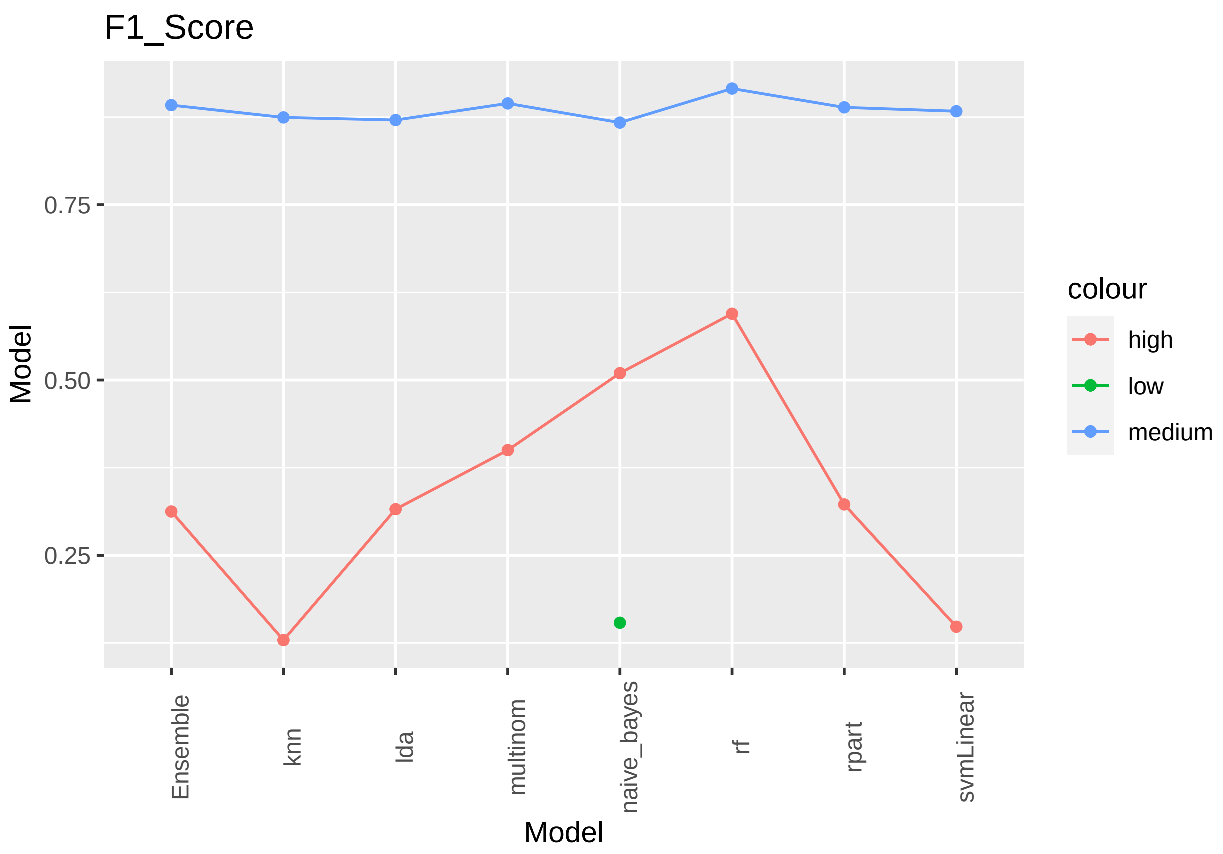
```r
# Remove the first row of 'results' that contains NAs
q_results <- q_results[2:(nrow(q_results)),]
```

The combination of the best results of all models in a single ensemble is presented and the summarized table of results.




Sensitivity
1.00
0.75
0.50
0.25
0.00
Model
colour
high
low
medium
Ensemble
knn
lda
multinom
naive_bayes
rf
rpart
svmLinear
Model


Specificity
1.00
0.75
0.50
0.25
Model
Ensemble
knn
lda
multinom
naive_bayes
rf
rpart
svmLinear
Model
colour
high
low
medium

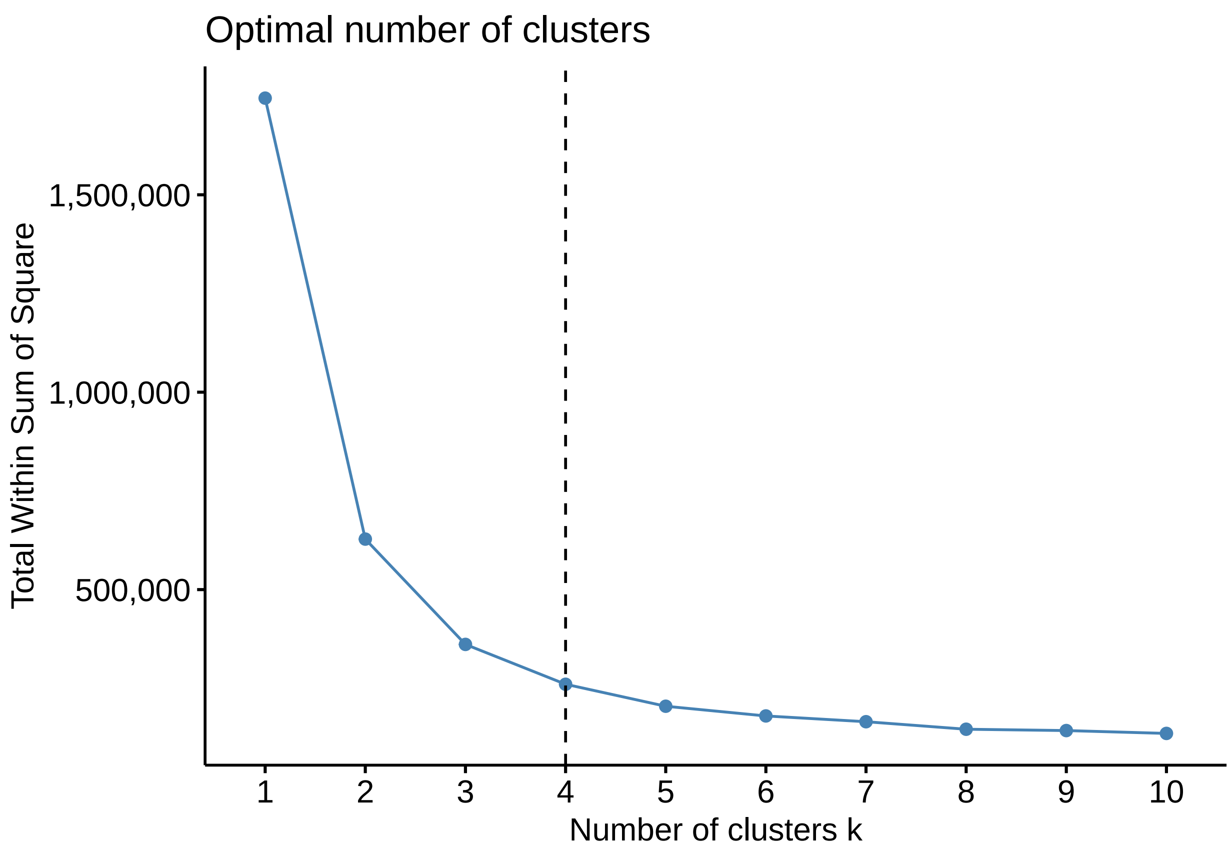


```r
#-------------------------------
# Stop parallel processing used in 'train'
#-------------------------------
stopCluster(cl)
```

##3.11 Clustering:##

The unsupervised learning are used to identify possible clusters. Partitioning methods, such as k-means clustering require the users to specify the number of clusters to be generated. The function fviz_nbclust from the factoextra package determines and visualize the optimal number of clusters using different methods. For the purpose of this analysis clusters are created for the red wines because the properties of the white and the red wines are different.

```
## $factoextra
## NULL
```



From the result of the plot it is observed that the total sum of squares decreases slowly for more than 4 clusters. The Selection of a larger value provides little improvement and increases the overlaps between the clusters.

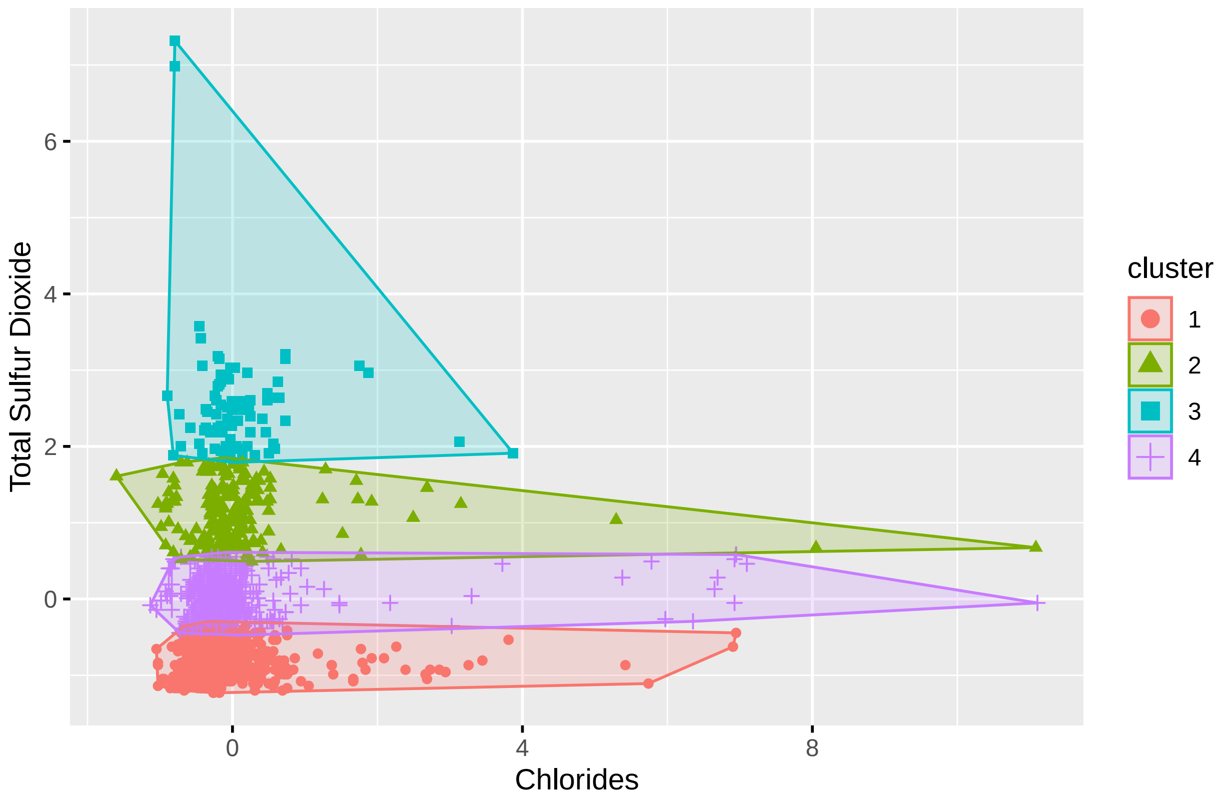
```
# We use 25 random starts for the clusters
k <- train_set %>% filter(type == "red") %>% .[,xcol] %>%
      kmeans(x = ., centers = 4, nstart = 25)

# Calculate cluster means
clus_m <- as_hux(data.frame(t(k$centers)), add_rownames = TRUE)
colnames(clus_m) <- c("Feature", paste("Cluster", 1:4))
clus_m <- add_colnames(clus_m)
clus_m %>%
      # Format header row
      set_bold(row = 1, col = everywhere, value = TRUE)         %>%
      set_top_border(row = 1, col = everywhere, value = 1)      %>%
      set_bottom_border(row = c(1,12), col = everywhere, value = 1)  %>%
      # Format cells
      set_align(row = everywhere, col = 1,   value = 'left')    %>%
      set_align(row = everywhere, col = 2:5, value = 'right')   %>%
      set_number_format(row = 2:nrow(clus_m),
                        col = everywhere, value = 3)            %>%
      # Format table
      set_width(value = 0.7)                                          %>%
      set_position(value = "center")                                  %>%
      set_caption("Features of Cluster Center")
```

Table 12: Features of Cluster Center

Feature	Cluster 1	Cluster 2	Cluster 3	Cluster 4
rownames	X1.000	X2.000	X3.000	X4.000
fixed_acidity	8.519	8.083	7.958	8.168
volatile_acidity	0.520	0.544	0.545	0.524
citric_acid	0.274	0.279	0.322	0.252
residual_sugar	2.403	2.901	3.465	2.374
chlorides	0.084	0.090	0.090	0.090
free_sulfur_dioxide	8.361	24.646	30.478	19.129
total_sulfur_dioxide	20.703	83.537	130.725	47.413
density	0.997	0.997	0.997	0.997
pH	3.306	3.319	3.234	3.330
sulphates	0.648	0.645	0.688	0.672
alcohol	10.598	10.137	9.853	10.410

The fviz_cluster function plots the clusters for the combination of two features. The clusters for total sulfur dioxide and chlorides are hereby ploted

Cluster plot with selected features



###IV. Conclusion:###

This research has explored the physicochemical constituent of red and white wine base on quality and type. According to experts quality is a subjective measure. The report gives a brief summary of model evaluation, explaining the most common metrics used in categorical problems in machine learning. The steps taken in actualizing the research includes, data preparation by downloading and importing the dataset, training and testing the dataset, data exploration and visualization of the features, model building ranging fron simple to a more complex models such as simple linear regression, LDA and QDA, CART using rpart, esemble and cross validation. The result obtained showed that the best predictors have low distribution overlapping area and low correlation among them. The model performance were evaluated and the algorithms were used to predict wine type and quality. Finaly, the utilization of the data to demonstrate clustering.

###V. Recommendation:###

The application of a more efficient machine learning models that would be able to predict red and white wine quality with high accuracy, specificity and sensitivity.

More information on composition of grape varieties in each wine, the mix of experts that evaluated the wine quality and the production year would enrich the dataset and expand its scope. A further research to ascertain the relationships between the different physicochemical properties of the wine and its association and correlation would clarify the issue of Simpson paradox.

###References### Rafael A. Irizarry, (2020) - â€œIntroduction to Data Scienceâ€ - https://rafalab.github.io/dsbook/

Baptiste AuguiÃ©, (2019) - â€œLaying out multiple plots on a pageâ€ - https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html

Yihui Xie, J. J. Allaire, Garrett Grolemund (2019) - â€œR Markdown: The Definitive Guideâ€ - https://bookdown.org/yihui/rmarkdown/

Baptiste Auguie, (2017) - â€œArranging multiple grobs on a pageâ€ https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html

Max Kuhn, (2019) - â€œThe caret Packageâ€ https://topepo.github.io/caret/model-training-and-tuning.html

ROC and AUC - https://stackoverflow.com/questions/31138751/roc-curve-from-training-data-in-caret

Xavier Robin et al, (2019), â€œDisplay and Analyze ROC Curvesâ€ - https://cran.r-project.org/web/packages/pROC/pROC.pdf

Sarang Narkhede. (2018) - â€œUnderstanding AUC - ROC Curveâ€ https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Alboukadel Kassambara, Fabian Mundt, (2019) - â€œExtract and Visualize the Results of Multivariate Data Analysesâ€ - https://cran.r-project.org/web/packages/factoextra/factoextra.pdf

â€œK-Means Clustering in R: Algorithm and Practical Examplesâ€ https://www.datanovia.com/en/lessons/k-means-clustering-in-r-algorith-and-practical-examples/

Teru Watanabe, (2016), â€œQuality and Physicochemical Properties of White Wineâ€ - https://rstudio-pubs-static.s3.amazonaws.com/152060_f4b5dad9dbd742a383ac3b8cee4e679c.html

Hadley Wickham, â€œThe Split-Apply-Combine Strategy for Data Analysisâ€ https://vita.had.co.nz/papers/plyr.pdf

https://stackoverflow.com/questions/57381627/add-text-labels-at-the-top-of-density-plots-to-label-the-different-groups

â€œWhat is Wine?â€ https://waterhouse.ucdavis.edu/tags/what-wine?page=1

â€œCluster Analysis in Râ€ http://girke.bioinformatics.ucr.edu/GEN242/pages/mydoc/Rclustering.html#1_introduction

https://www.edx.org/professional-certificate/harvardx-data-scienceâ†©

https://en.wikipedia.org/wiki/Vinho_Verdeâ†©

https://archive.ics.uci.edu/ml/datasets/Wine+Quality

http://gim.unmc.edu/dxtests/roc3.html

https://archive.ics.uci.edu/ml/datasets/Wine+Quality

https://portal.vinhoverde.pt/pt/estatisticas https://topepo.github.io/caret/variable-importance.html

https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/
â†©

https://scikit-learn.org/stable/modules/lda_qda.htmlâ†©

/pagebreak