HarvardX: PH125.9x Wine Quality Project

Mohammed Bouayoun

09/01/2020

1. Overview

The objective of this Project was to investigate how the physicochemical like alcohol, chlorides, sulphates content varies the quality of Wine. This study analyzed different wine samples of the portuguese "Vinho Verde" wine.

There is no wine without chemistry. Oenology is the science of wine and therefore involves a chemical knowledge of the product.

Modern device allows the winemaker to follow the fermentation and vinification closely. In 2 minutes, the winemaker obtains a complete analysis of his sample (must, fermenting wine, finished wine): sugar content, alcohol content, total acidity, pH, volatile acidity, etc.

2. Introduction

The dataset was downloaded from the UCI Machine Learning Repository https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/.

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine, it's contains the chemical properties of the wine as well as the wine quality score. The twho datasets comes from the paper P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. The goal is to predict human expert taste preferences based on 11 analytical tests (continuous values, such as pH or alcohol levels) that are easy to collect during the wine certification step. The output variable is categorical and ordered, ranging from 3 (low quality) to 9 (high quality). It was downloaded from the University of California Irvine Machine Learning Repository at http://archive.ics.uci.edu/ml/.

3. Loading and cleaning Datasets

```
white_url <-
"https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv"
white <- read.csv(white_url, header = TRUE, sep = ";")

red_url <-
"https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
red <- read.csv(red_url, header = TRUE, sep = ";")
```

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(GGally)) install.packages("GGally", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
if(!require(lattice)) install.packages("lattice", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
if(!require(klaR)) install.packages("klaR", repos = "http://cran.us.r-project.org")
if(!require(kernlab)) install.packages("kernlab", repos = "http://cran.us.r-project.org")
```

Loading library

```
library(tidyverse)
library(caret)
library(lattice)
library(GGally)
library(e1071)
library(MASS)
library(klaR)
library(kernlab)
```

Checking for any missing (NA) values.

```
white <- white[!duplicated(unique(white)), ]
red <- red[!duplicated(unique(red)), ]
```

We add categorical variable **type** to both sets.

```
red['type'] <- 'red'
white['type'] <- 'white'
```

From regression to binary classification

We create another categorical variable, classifying the wines as **bad** (rating 0 to 5), and **good** (rating 6 to 10).

red wine

```
good <- red$quality >= 6
bad <- red$quality < 6
red[good, 'quality'] <- 'good'
red[bad, 'quality'] <- 'bad'
red$quality <- as.factor(red$quality)
```

white win

```
good <- white$quality >= 6
bad <- white$quality < 6
white[good, 'quality'] <- 'good'
white[bad, 'quality'] <- 'bad'
white$quality <- as.factor(white$quality)
```

The two datasets **red** and **white** are joined into one larger dataset **wine**.

```
wine <- rbind(red, white)
```

We delete type variable in dataset red and white.

```
red$type <- NULL
white$type <- NULL
```

4. Data structure and Methodology

4.1. Data structure

Dimension of total wine

```
## [1] 6497   13
```

The wine dataset has 6497 observations, 11 predictors and 1 outcome (quality). All of the predictors are numeric values, outcomes are integer.

Dimension of red wine

```
wine %>% filter(type=="red")  %>% dim
```

```
## [1] 1599   13
```

The red wine dataset has 1599 observations, 11 predictors and 1 outcome (quality). All of the predictors are numeric values, outcomes are integer.

Dimension of white wine

```
wine %>% filter(type=="white")  %>% dim
```

```
## [1] 4898   13
```

The white wine dataset has 4898 observations, 11 predictors and 1 outcome (quality). All of the predictors are numeric values, outcomes are integer.

Table preview

```
## Observations: 6,497
## Variables: 13
## $ fixed.acidity        <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3, 7.8, 7....
## $ volatile.acidity     <dbl> 0.700, 0.880, 0.760, 0.280, 0.700, 0.660, 0.600,...
## $ citric.acid          <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, 0.00, ...
## $ residual.sugar       <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2.0, 6.1...
## $ chlorides            <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.075, 0.069,...
## $ free.sulfur.dioxide  <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 17, 15, 17, 1...
## $ total.sulfur.dioxide <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102, 65, 102...
## $ density              <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0.9978, ...
## $ pH                   <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, 3.39, ...
## $ sulphates            <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, 0.47, ...
## $ alcohol              <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, 10.0, 9.5, 10...
## $ quality              <fct> bad, bad, bad, good, bad, bad, bad, good, good, ...
## $ type                 <chr> "red", "red", "red", "red", "red", "red", "red",...
```

-There are 13 columns in the data set. The data type of all these columns is numeric except type is categorial.
-All numeric columns except the **quality** column is continuous.

The physical properties which are in the data set are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates, alcohol and finally quality. There are physical as well as chemical variables that influence the quality of wines.

Input variables (based on physicochemical tests):

-**fixed acidity** (tartaric acid - g/dm^3)
-**volatile acidity** (acetic acid - g/dm^3)
-**citric acid** (g/dm^3)
-**residual sugar** (g/dm^3)
-**chlorides** (sodium chloride - g/dm^3
-**free sulfur dioxide** (mg/dm^3)
-**total sulfur dioxide** (mg/dm^3)
-**density** (g/cm^3)
-**pH**
-**sulphates** (potassium sulphate - $g/dm3$)
-**alcohol** (% by volume)
-**color** (wine color)

Output variable (based on sensory data):

-**quality** (score between 0 and 10)

4.2 Methodology

-Transforming quality variable from numeric to category good and bad
-Detecting an outliers
-Identifying an optimal subset of variables with backward elimination
-Predicting with Naive Bayes
-Predicting with Logistic Regression
-Predicting with K Nearest Neighbors
-Predicting with Random Forest
-Predicting with Support Vector Machines with linear kernel
-Predicting with Support Vector Machines with Radial Basis Function
-Comparing all models

5. Description of attributes

Fixed acidity : Refers to all organic acids not included under the volatile category include tartaric, malic, citric, and succinic acids which are found in grapes (except succinic). Quantitatively, they control the pH of wine.

Volatile acidity :The volatile acidity of a wine consists of the part of the fatty acids belonging to the acetic series, which is found in wine either in the free state or in salt form. Volatile acidity gives the wine its bouquet, but when the dose of acid is too high, the wine is said to be sour, characterised by notes of glue or vinegar.

Citric acid : This is one of the fixed acids which gives a wine its freshness. Usually most of it is consumed during the fermentation process and sometimes it is added separately to give the wine more freshness.

Total acidity : This is the sum of the volatile acidity and the fixed acidity.

Residual sugar : Refers to the natural sugar from grapes which remains after the fermentation. The residual sugar content of dry wine is generally less than 1.5 g/liter.

Chlorides : The amount of salt in the wine, Chloride concentration in the wine is influenced by terroir and its highest levels are found in wines coming from countries.

Free sulfur : Measure of the amount of SO2 that is not bound to other molecules, and is used to calculate molecular SO2. Is used throughout all stages of the winemaking process to prevent oxidation and microbial growth.
the most signifi cant factor affecting color density of young red wines is not the pH, but the amount of free sulfur dioxide.

Density : The mass per unit volume of wine or must at 20°C. It is expressed in grams per milliliter. It is generally used as a measure of the conversion of sugar to alcohol.

pH : Parallel to the total acidity, the free acid functions are partially dissociated or ionised and release H+ ions into the liquid, which represent the real acidity, whose concentration is expressed as a pH. It depends to a large extent on tartaric acid.

Sulphates : These are mineral salts containing sulfur. Sulphates are to wine as gluten is to food. They are a regular part of the winemaking around the world and are considered essential. They are connected to the fermentation process and affects the wine aroma and flavour.

alcohol : The percent alcohol content of the wine.

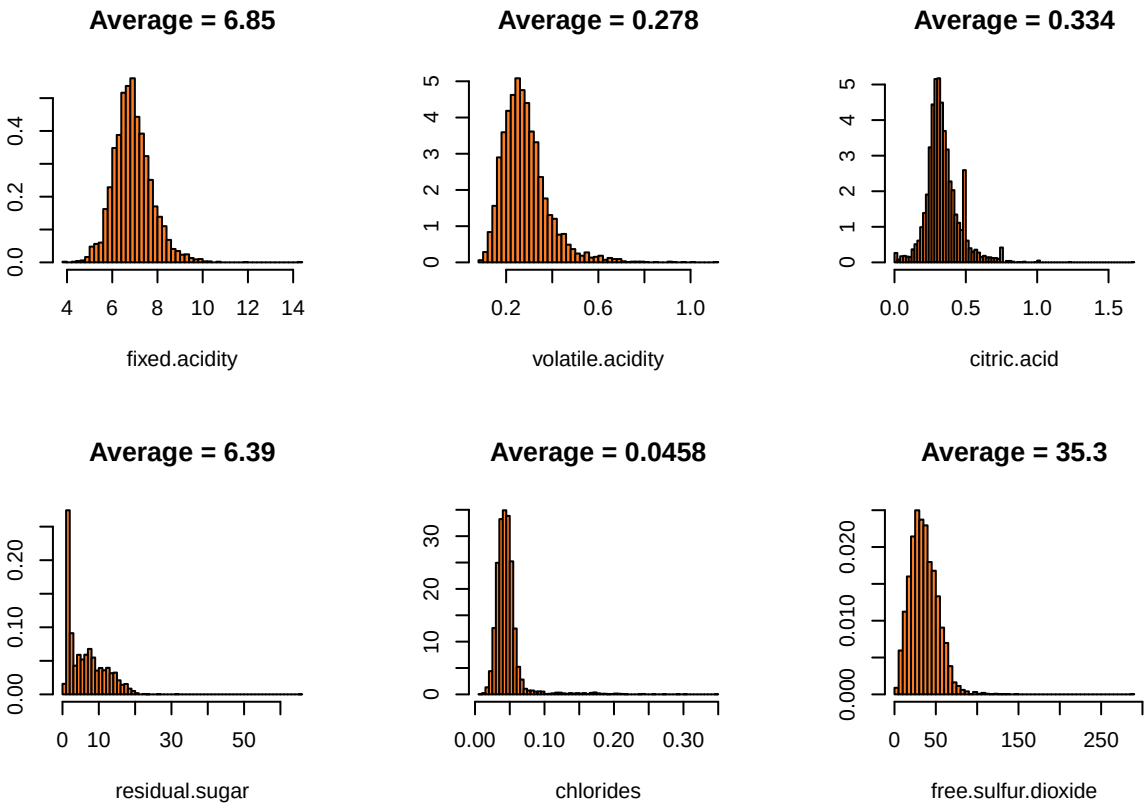
Quality : Wine experts graded the wine quality between 0 (very bad) and 10 (very excellent).

type : red or white color of wine.

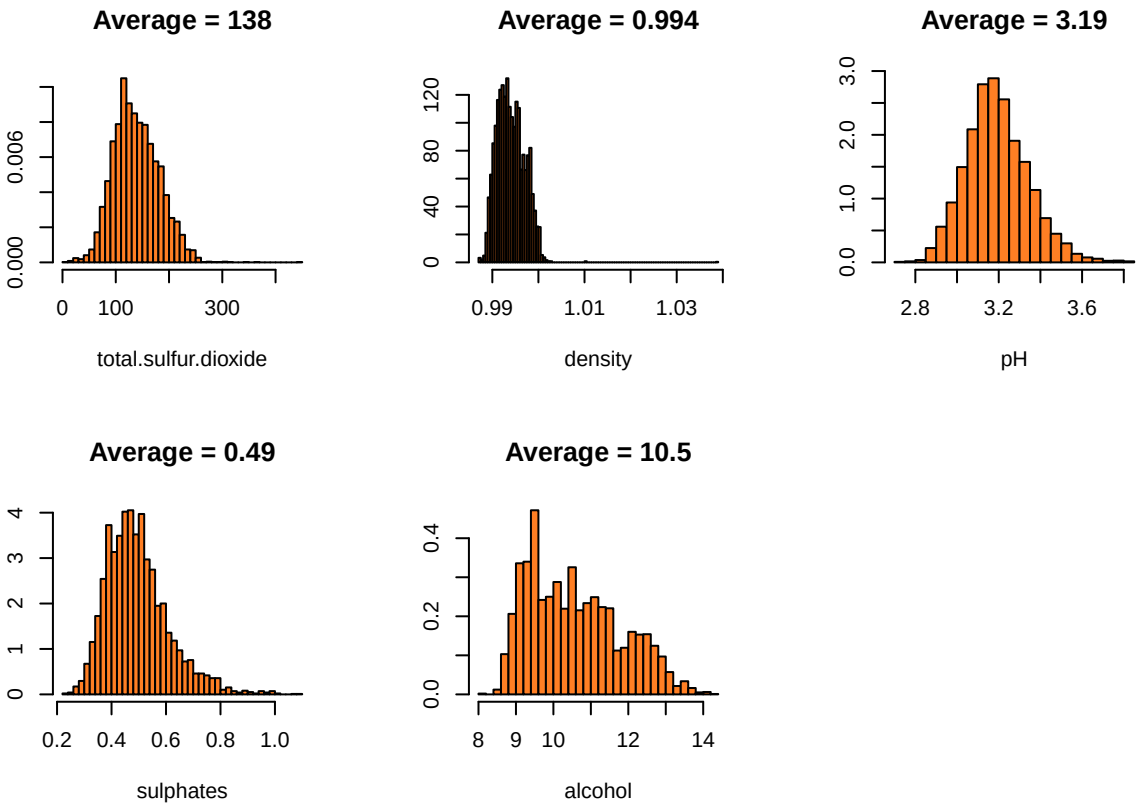
6. Distribution of Single Variables

6.1. Distribution of white wine

```
oldpar = par(mfrow = c(2,3))
for ( i in 1:11 ) {
  truehist(white[[i]], xlab = names(white)[i], col = 'chocolate1',
           main = paste("Average =", signif(mean(white[[i]]),3)))
}
```


Average = 6.85
0.0
0.2
0.4
4
6
8
10
12
14
fixed.acidity
Average = 0.278
0
1
2
3
4
5
0.2
0.6
1.0
volatile.acidity
Average = 0.334
0
1
2
3
4
5
0.0
0.5
1.0
1.5
citric.acid
Average = 6.39
0.00
0.10
0.20
0
10
30
50
residual.sugar
Average = 0.0458
0
10
20
30
0.00
0.10
0.20
0.30
chlorides
Average = 35.3
0.000
0.010
0.020
0
50
150
250
free.sulfur.dioxide

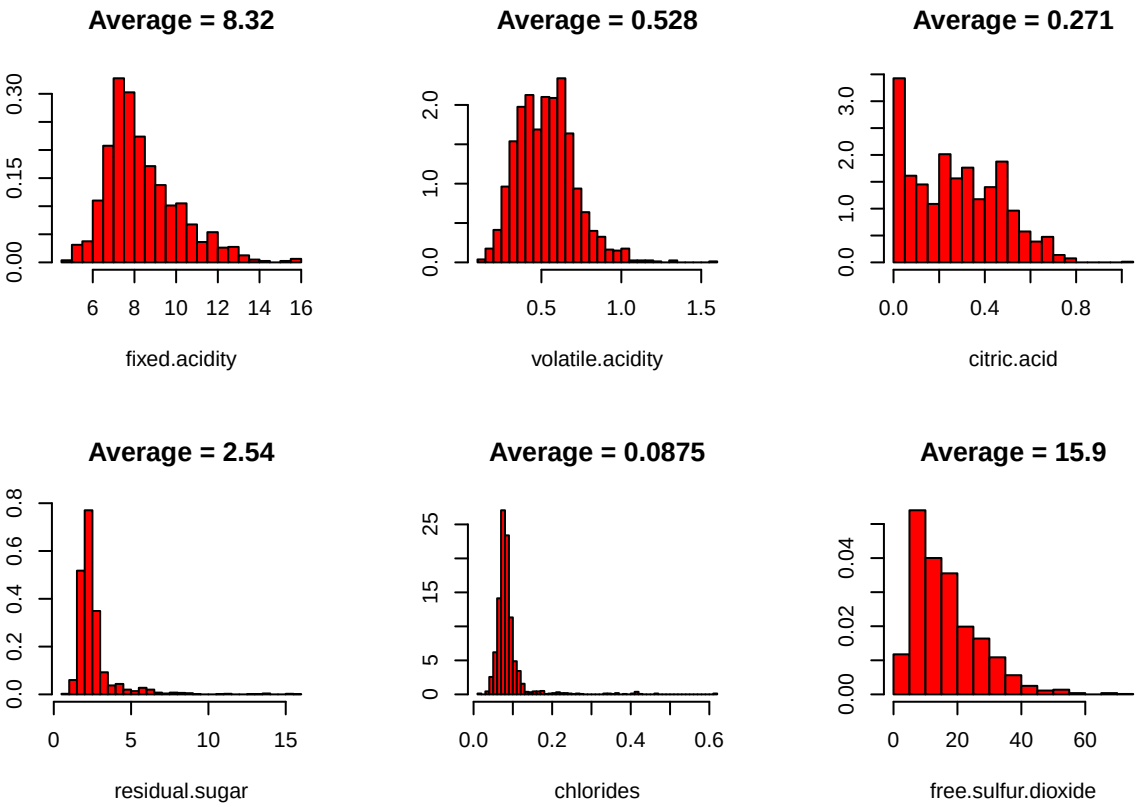
```
par(oldpar)
```

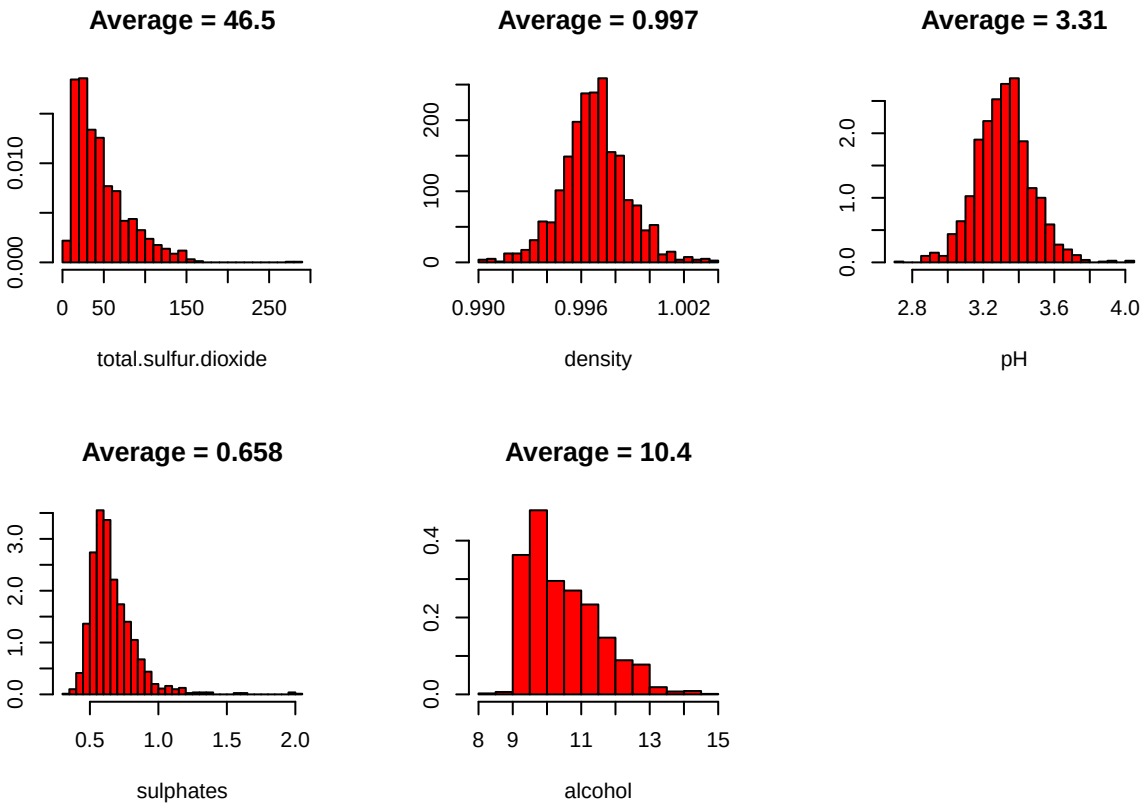


We note that all the variables has positively skewed distributions.
The pH value seems to dispaly a normal distribution.

6.2. Distribution of red wine

```
oldpar = par(mfrow = c(2,3))
for ( i in 1:11 ) {
  truehist(red[[i]], xlab = names(red)[i], col = '#ff0000',
           main = paste("Average =", signif(mean(red[[i]]),3)))
}
```


Average = 8.32
0.00
0.15
0.30
6
8
10
12
14
16
fixed.acidity
Average = 0.528
0.0
1.0
2.0
0.5
1.0
1.5
volatile.acidity
Average = 0.271
0.0
1.0
2.0
3.0
0.0
0.4
0.8
citric.acid
Average = 2.54
0.0
0.2
0.4
0.6
0.8
0
5
10
15
residual.sugar
Average = 0.0875
0
5
15
25
0.0
0.2
0.4
0.6
chlorides
Average = 15.9
0.00
0.02
0.04
0
20
40
60
free.sulfur.dioxide



We note that all the variables has positively skewed distributions.
The pH value seems to dispaly a normal distribution.

7. Outlier detection

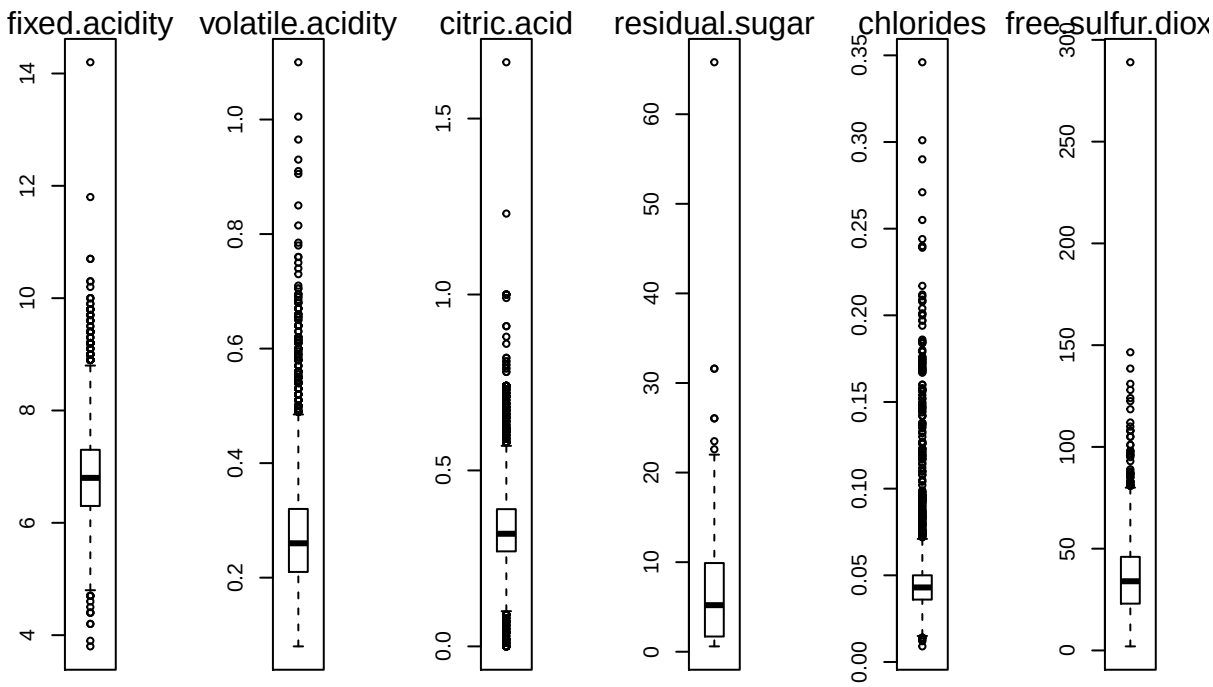
The difference between quartiles Q3 and Q1 is known as the **interquartile range** or **IQR**. The IQR contains the middle 50% of the data and provides a good idea of the spread of the distribution without the outlying observations. In fact, a comparison of the spread in the middle 50% and the full range is useful to detect outliers.

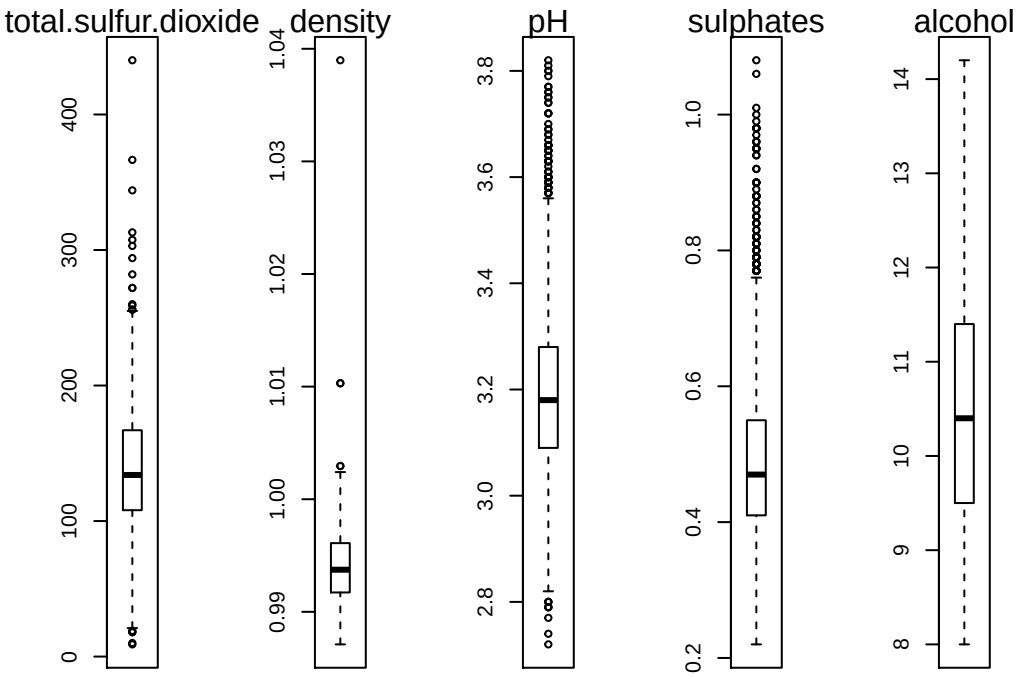
IQR = Q3 - Q1
LOWER_OUTLIER = Q1 - (1.5 x IQR)
UPPER_OUTLIER = Q3 + (1.5 x IQR)

The boxplot utilizes the IQR and constructs two sets of boundaries at 1.5xIQR and 3xIQR distance from Q1 and Q3 on the outer side. The box in the middle represents the spread of the middle 50% of the data. If there are any points outside [Q1 - 1.5 x IQR, Q3 + 1.5 * IQR], those points may be considered outliers.

7.1 White wine outliers

```
oldpar = par(mfrow = c(1,6))
for ( i in 1:11 ) {
  boxplot(white[[i]])
  mtext(names(white)[i], cex = 1, side = 3, line = 0)
}
```

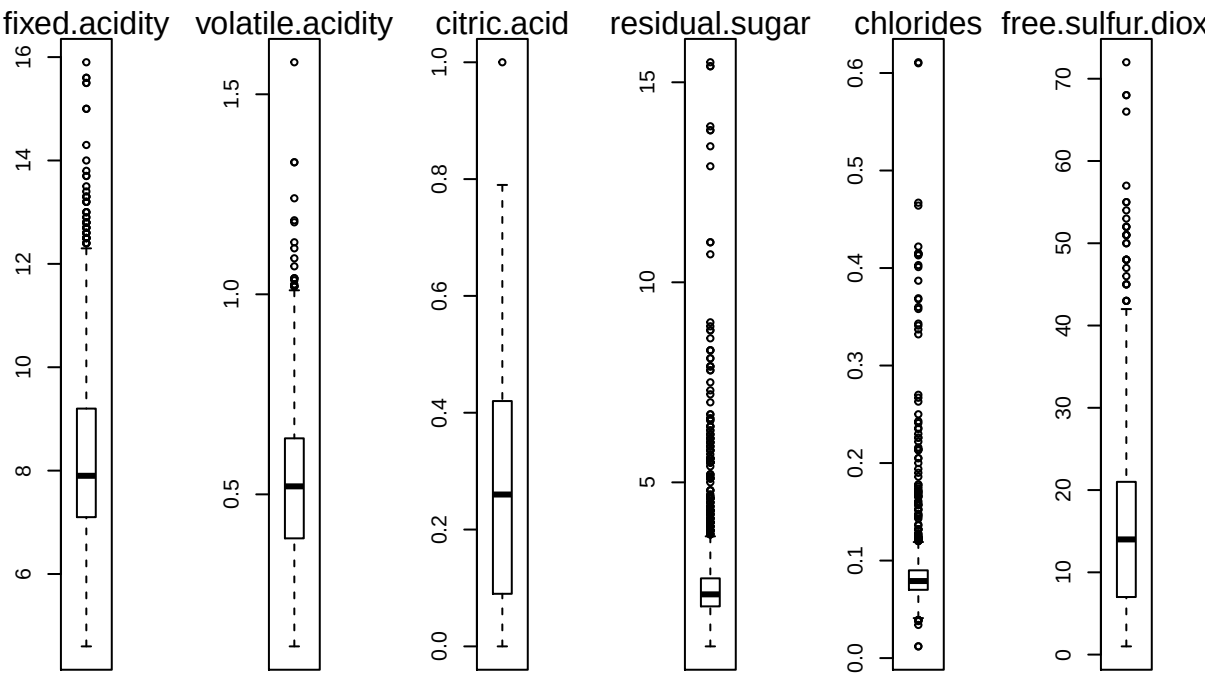

fixed.acidity
volatile.acidity
citric.acid
residual.sugar
chlorides
free.sulfur.diox
4
6
8
10
12
14
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
0
10
20
30
40
50
60
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
50
100
150
200
250
300

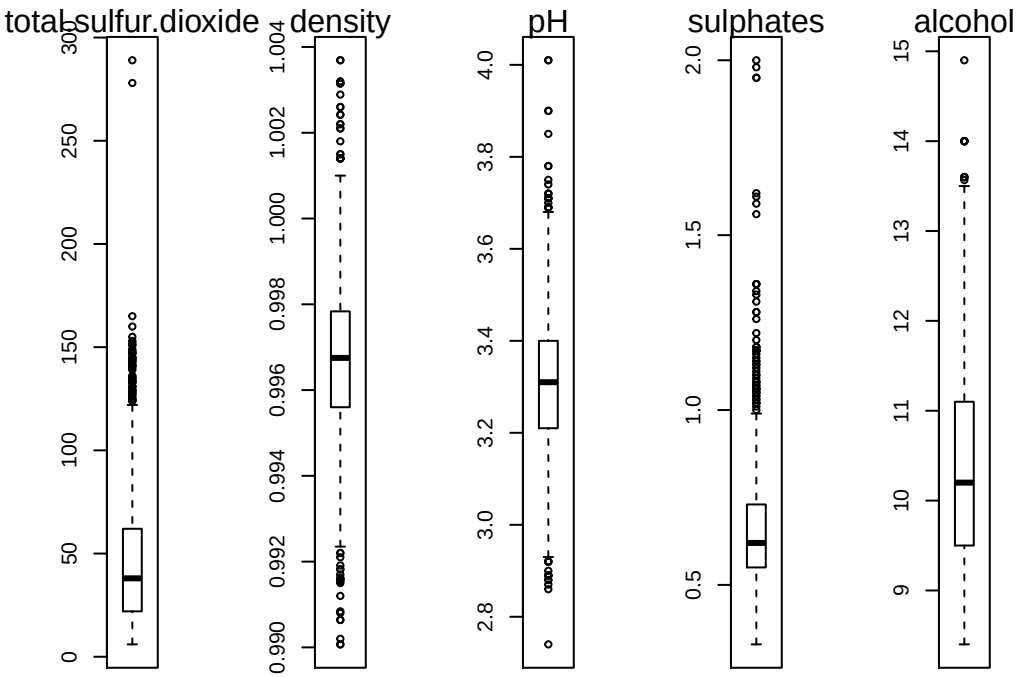


All variables of white wine except alcohol contains outliers.

7.2 red wine outliers

```
oldpar = par(mfrow = c(1,6))
for ( i in 1:11 ) {
  boxplot(red[[i]])
  mtext(names(red)[i], cex = 1, side = 3, line = 0)
}
```


fixed.acidity
volatile.acidity
citric.acid
residual.sugar
chlorides
free.sulfur.diox



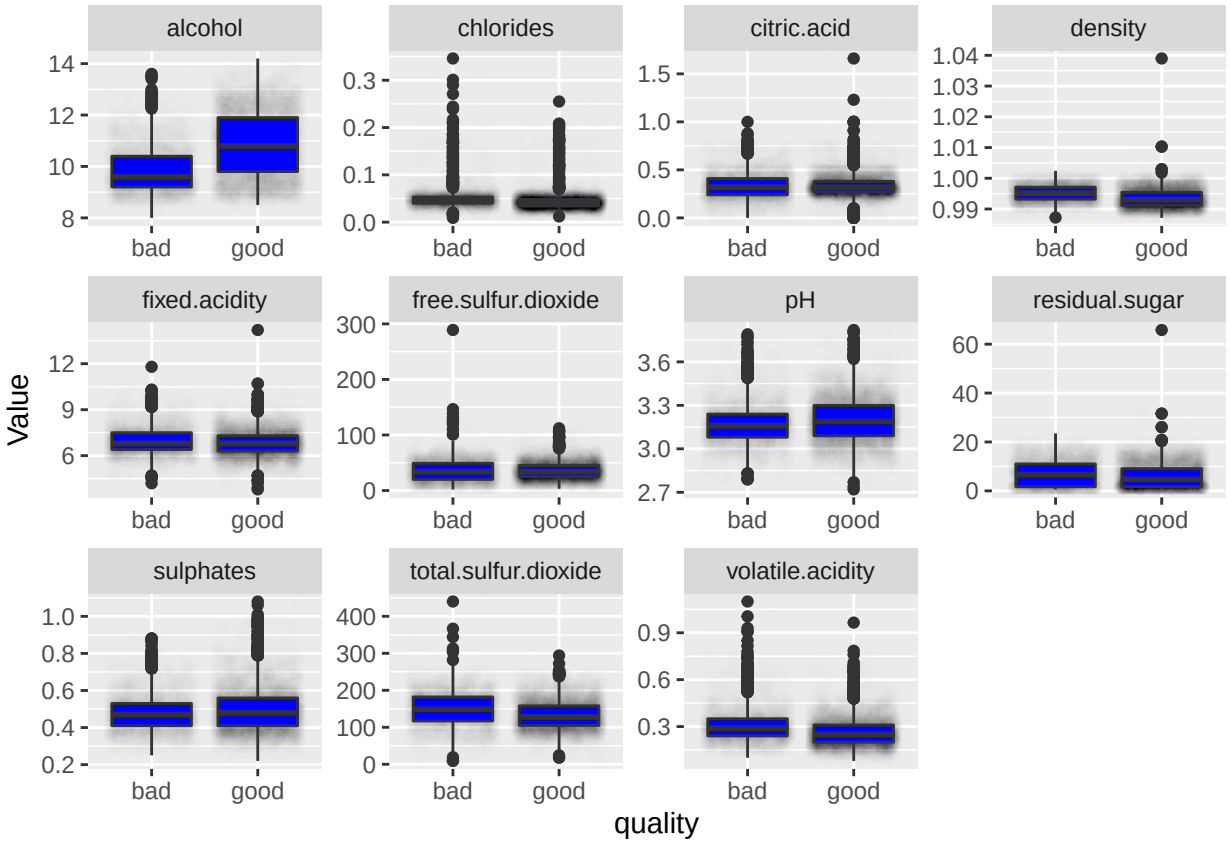
All variables of red wine contains outliers.

8. Variables by quality ranking

Now, we will explore each variable with respect to quality.

8.1. White wine and quality

```
white[1:12] %>%
  gather(key = "Variable", value = "Value", `fixed.acidity`:alcohol) %>%
  ggplot(aes(x = quality, y = Value)) +
  geom_jitter(alpha = 0.005)  +
  geom_boxplot(aes(group = cut_width(quality, 1)), fill = "blue") +
  facet_wrap(~ Variable, scales = "free") +
  guides(fill = TRUE)
```



There exists a positve correlation between alcohol and quality : 0.38328.

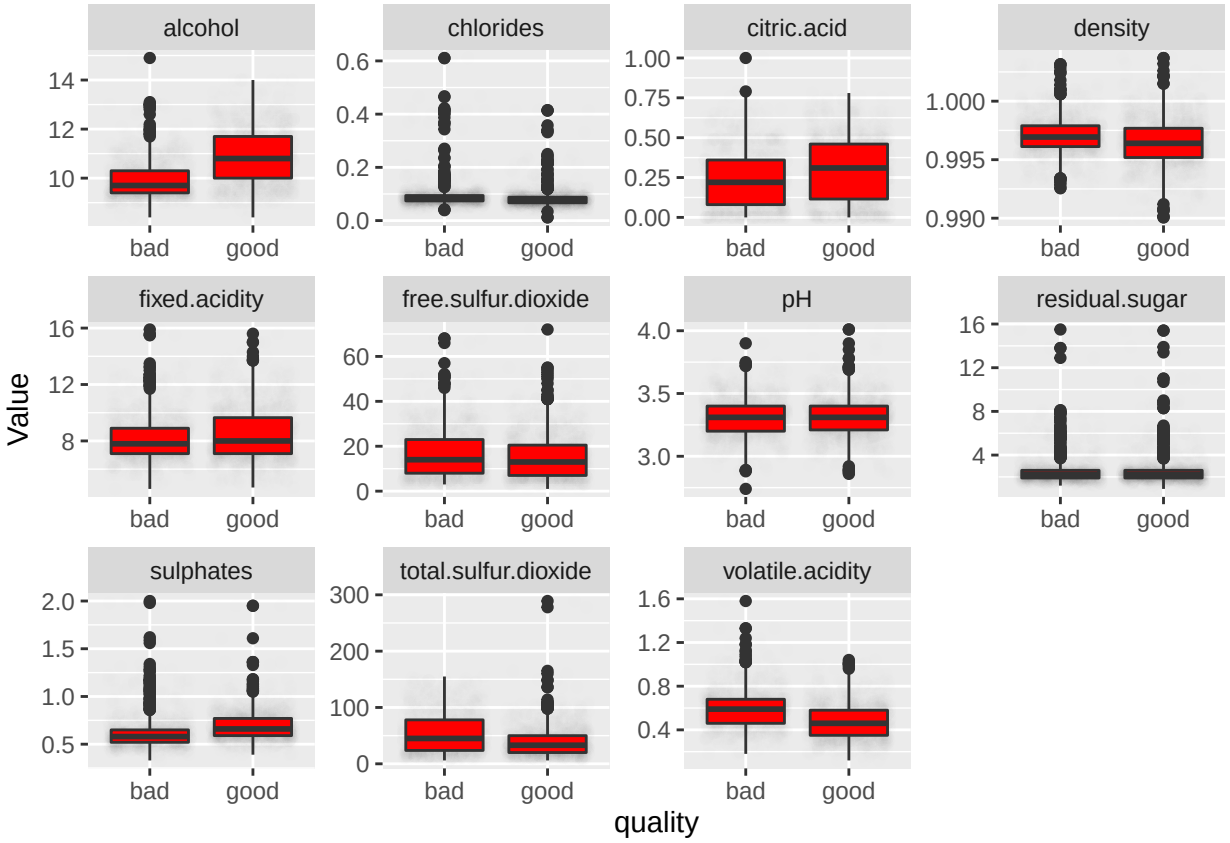
```
cor.test(as.numeric(white$quality), white$alcohol,
         method = 'pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(white$quality) and white$alcohol
## t = 29.036, df = 4896, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3591285 0.4069184
## sample estimates:
##     cor
## 0.38328
```

8.2. Red wine and quality

```
red[1:12] %>%
  gather(key = "Variable", value = "Value", `fixed.acidity`:alcohol) %>%
  ggplot(aes(x = quality, y = Value)) +
  geom_jitter(alpha = 0.005)  +
  geom_boxplot(aes(group = cut_width(quality, 1)), fill = "red") +
```

```
  facet_wrap(~ Variable, scales = "free") +
  guides(fill = TRUE)
```



There exists a positve correlation between alcohol and quality : 0.4347512.
Little positive correlation with citric.acid and quality.

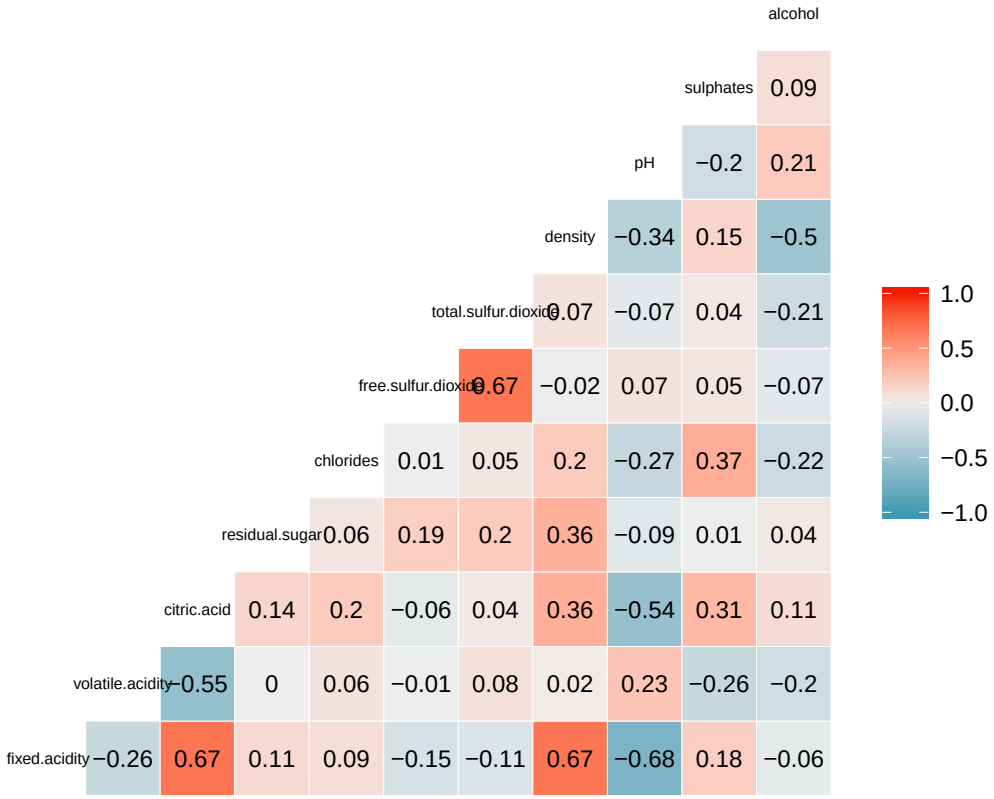
```
cor.test(as.numeric(red$quality), red$alcohol, method = 'pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(red$quality) and red$alcohol
## t = 19.292, df = 1597, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3941298 0.4736773
## sample estimates:
##       cor
## 0.4347512
```

9. Correlations

We can see correlation here for red wine

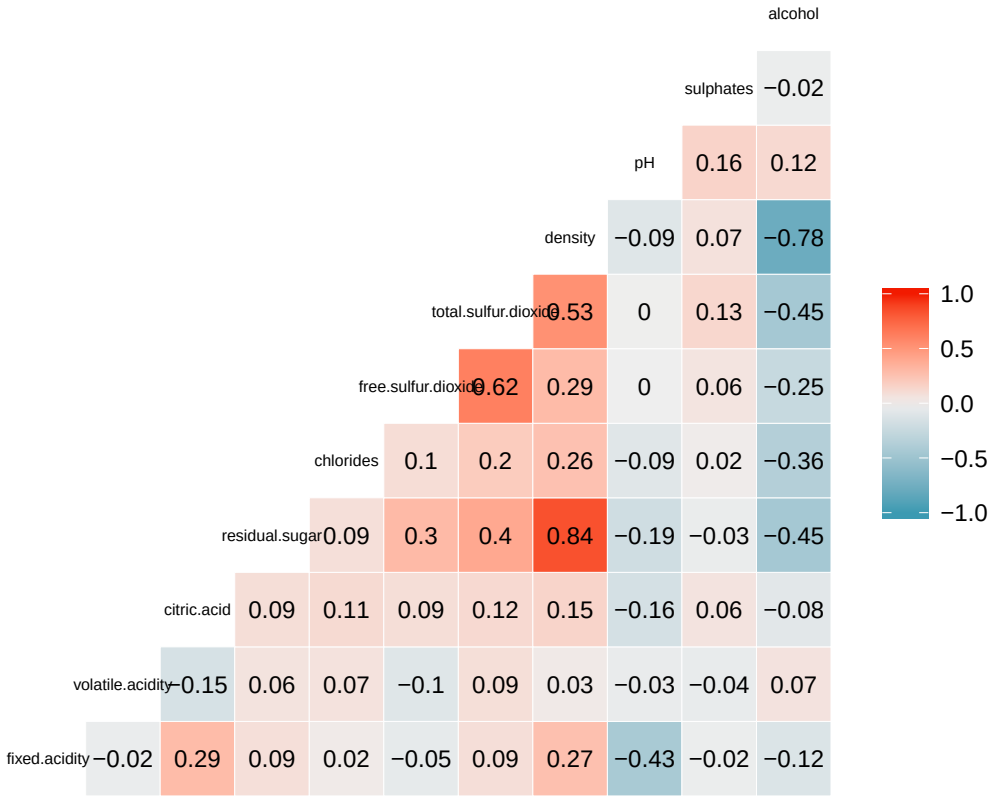
```
ggcorr(red, label = TRUE, label_round = 2,label_size = 3,size = 2)
```



Positive correlation between fixed.acidity and citric.acid
Positive correlation between fixed.acidity and density
Negative correlation between fixed.acidity and pH
Negative correlation between volatile.acidity and citric.acid
Negative correlation between citric.acid and pH
Positive correlation between free.sulfur.dioxide and total.sulfur.dioxide

And correlation for white wine

```
ggcorr(white, label = TRUE, label_round = 2,label_size = 3,size = 2)
```



Positive correlation between density and residual.sugar
Positive correlation between density and total.sulfur.dioxide
Positive correlation between free.sulfur.dioxide and total.sulfur.dioxide
Negative correlation between alcohol and density

10. Data preparation

10.1 Resampling

The k-fold cross validation method involves splitting the dataset into k-subsets. For each subset is held out while the model is trained on all other subsets. This process is completed until accuracy is determine for each instance in the dataset, and an overall accuracy estimate is provided.
We uses 5-fold cross validation for our models

```
ctrl <- trainControl(method = 'cv', number = 5)
```

```
set.seed(1, sample.kind="Rounding")
# red wine

redSplit <- createDataPartition(red$quality,p = 0.75,list = FALSE)
redTrain <- red[redSplit,]
redTest  <- red[-redSplit,]

# white wine
```

```
whiteSplit <- createDataPartition(white$quality,p = 0.75,list = FALSE)
whiteTrain <- white[whiteSplit,]
whiteTest  <- white[-whiteSplit,]
```

10.2. Subset Selection Methods

One way to handle a complex model is to identify an optimal subset of variables from a larger set of predictors.

The stepwise regression (or stepwise selection) consists of iteratively adding and removing predictors, in the predictive model, in order to find the subset of variables in the data set resulting in the best performing model, that is a model that lowers prediction error.

There are three strategies of stepwise regression (James et al. 2014,P. Bruce and Bruce (2017)):

-**Forward selection**, which starts with no predictors in the model, iteratively adds the most contributive predictors, and stops when the improvement is no longer statistically significant.
-**Backward selection** (or backward elimination), which starts with all predictors in the model (full model), iteratively removes the least contributive predictors, and stops when you have a model where all predictors are statistically significant.
-**Stepwise selection** (or sequential replacement), which is a combination of forward and backward selections. You start with no predictors, then sequentially add the most contributive predictors (like forward selection). After adding each new variable, remove any variables that no longer provide an improvement in the model fit (like backward selection).

forward selection and stepwise selection can be applied in the high-dimensional configuration, where the number of samples n is inferior to the number of predictors p, such as in genomic fields.
Backward selection requires that the number of samples n is larger than the number of variables p, so that the full model can be fit.

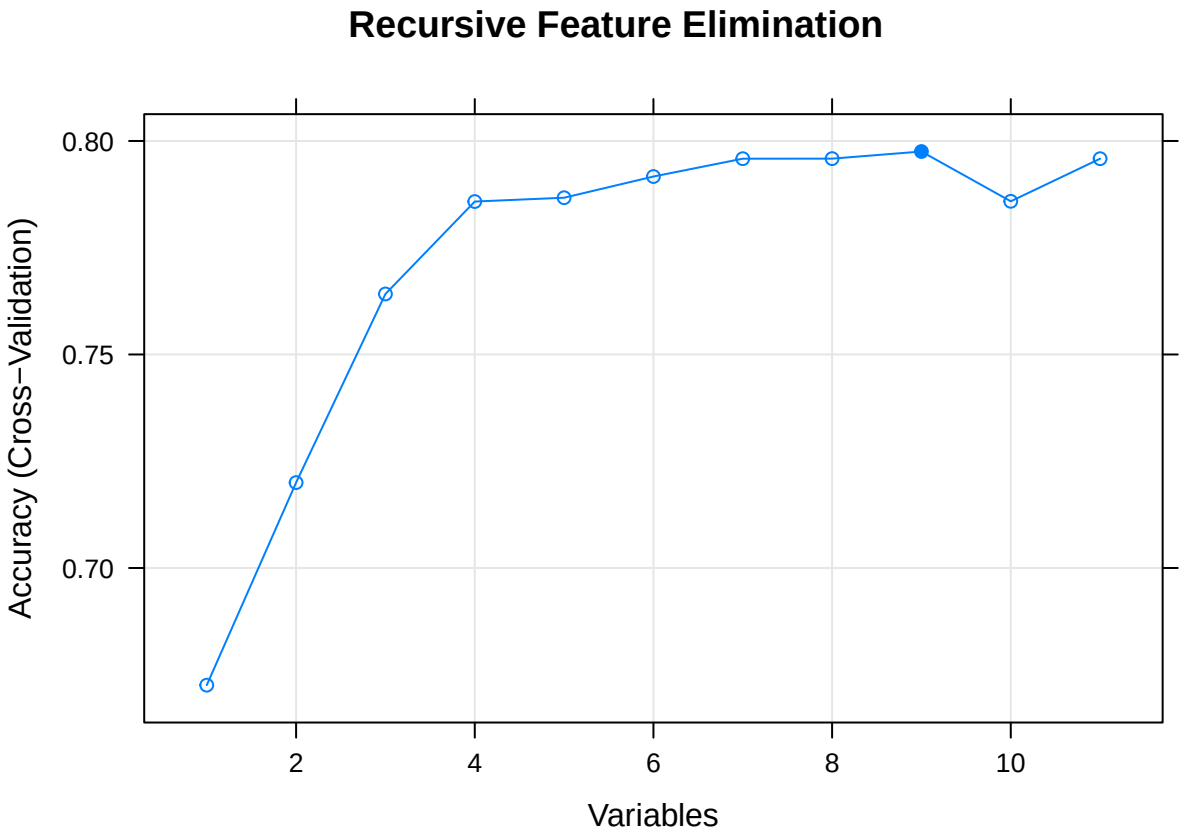
Feature elimination for red wine

```
fitCtrl_redrfe <- rfeControl(functions = rfFuncs, method = 'cv', number = 5)
fit_redrfe <- rfe(quality ~., data = redTrain,
                  sizes = c(1:10),
                  rfeControl = fitCtrl_redrfe)
features_red <- predictors(fit_redrfe)

max(fit_redrfe$results$Accuracy)
```

```
## [1] 0.7975333
```

```
plot(fit_redrfe, type = c('g', 'o'), main = 'Recursive Feature Elimination')
```



The new features for red wine :

```
features_red
```

```
## [1] "alcohol"              "sulphates"            "total.sulfur.dioxide"
## [4] "volatile.acidity"     "density"              "chlorides"
## [7] "free.sulfur.dioxide"  "fixed.acidity"        "pH"
```

The RFE removed : residual.sugar, citric.acid and volatile.acidity from features.

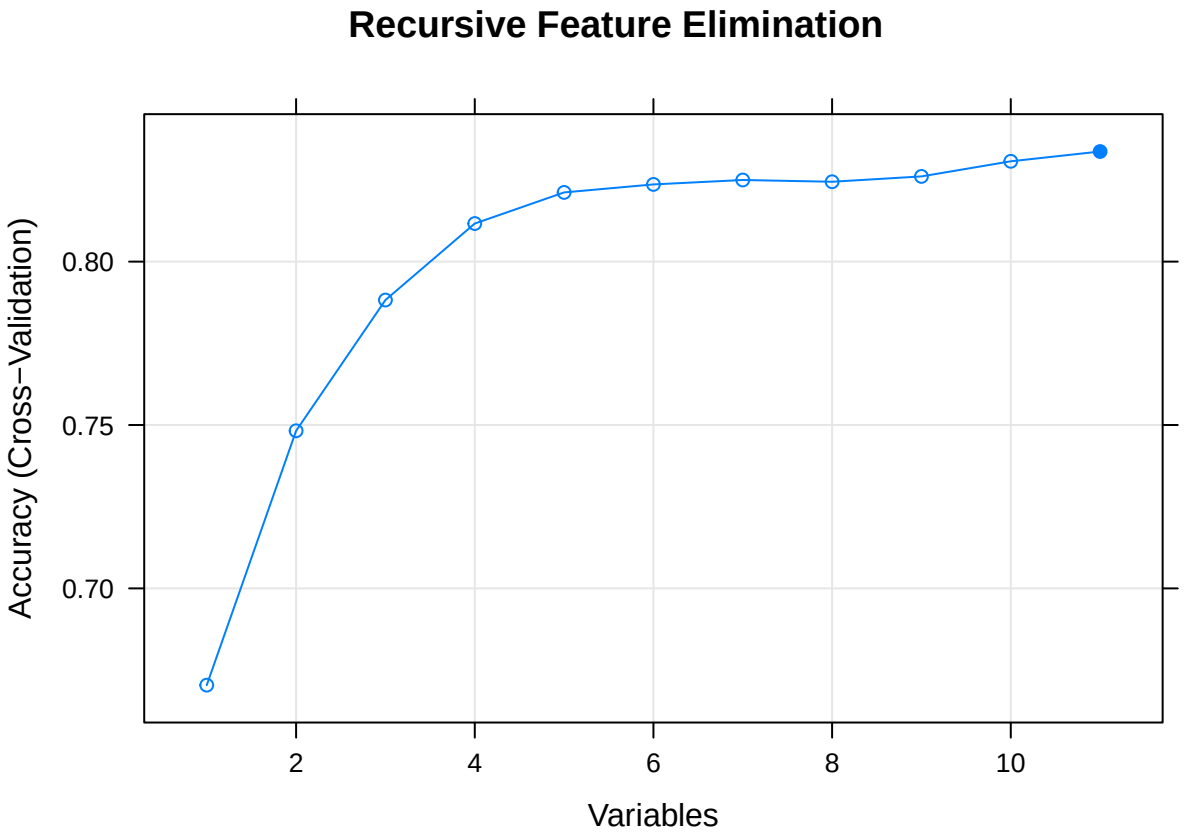
Feature elimination for white wine

```
fitCtrl_whiterfe <- rfeControl(functions = rfFuncs, method = 'cv', number = 5)
fit_whiterfe <- rfe(quality ~., data = whiteTrain,
                sizes = c(1:10),
                rfeControl = fitCtrl_whiterfe)
features_white <- predictors(fit_whiterfe)

max(fit_whiterfe$results$Accuracy)
```

```
## [1] 0.8336926
```

```
plot(fit_whiterfe, type = c('g', 'o'), main = 'Recursive Feature Elimination')
```



The new features for white wine :

```
features_white
```

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

The RFE take all features.

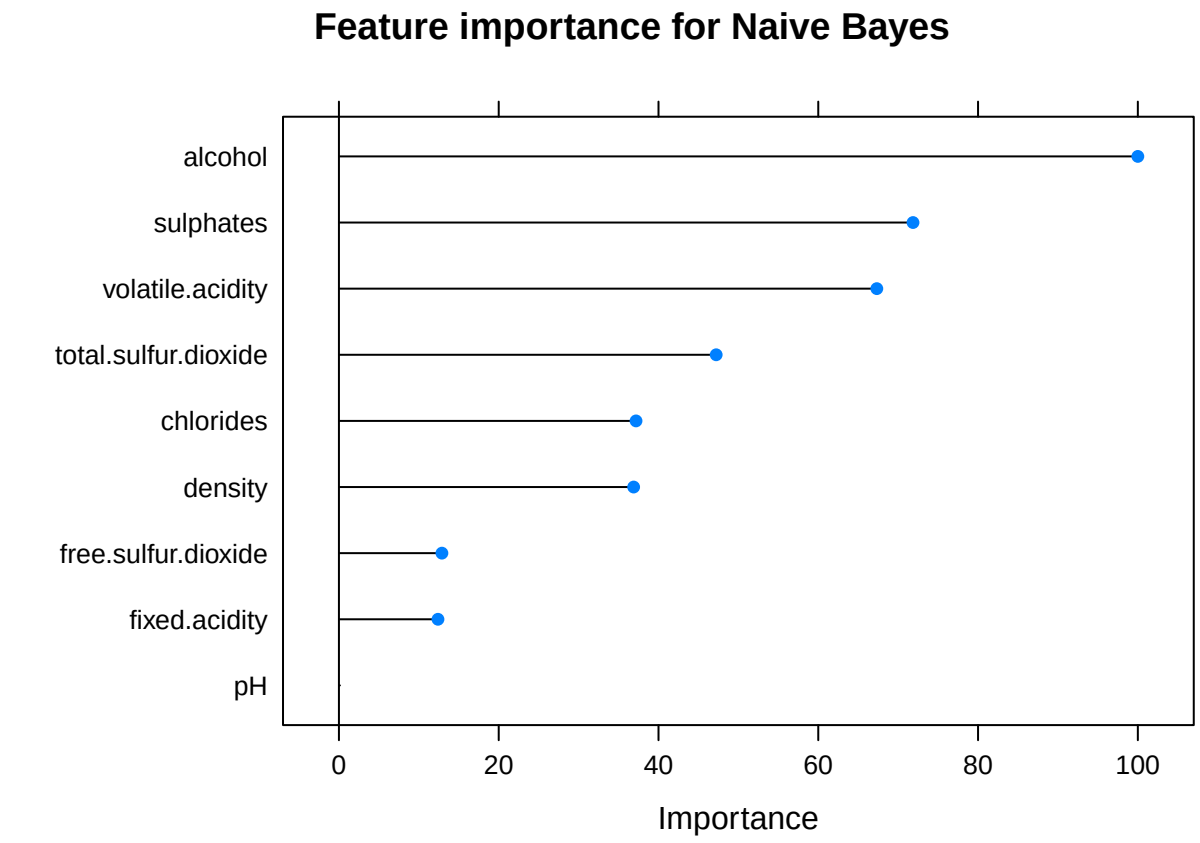
11. Model Prediction

11.1. Naive Bayes classifier

Let's start with the simplest models.
We generate a Naive Bayes model, using 5-fold cross-validation.
For red wine

```
fit_rednaive <- train(x = redTrain[, features_red], y = redTrain$quality,
                method ="nb",
                trControl = ctrl)
predict_rednaive <- predict(fit_rednaive, newdata = redTest[, features_red])
confMat_rednaive <- confusionMatrix(predict_rednaive, redTest$quality, positive = 'good')
```

```
importance_rednaive <- varImp(fit_rednaive, scale = TRUE)
plot(importance_rednaive, main = 'Feature importance for Naive Bayes')
```




```
accuracy_bn_red <-max(fit_rednaive$results$Accuracy)
```

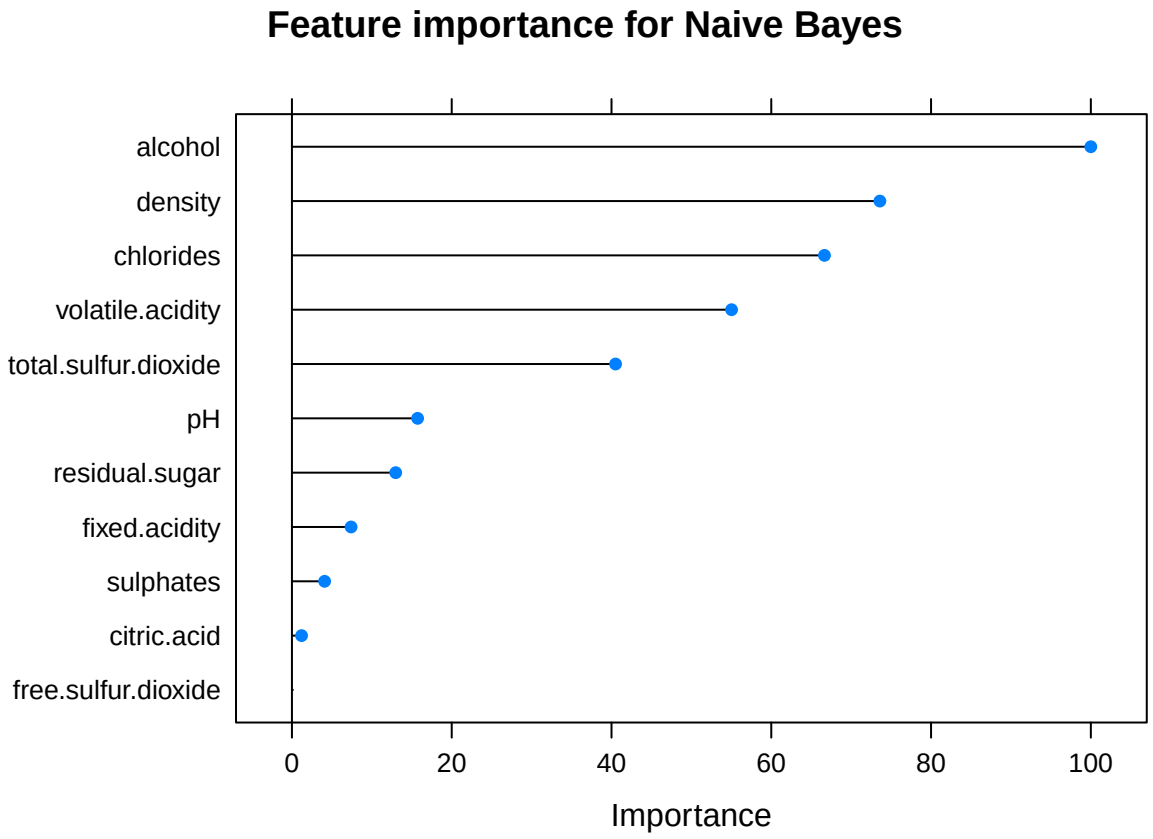
We can see which variables are relatively more important. The following shows the normalized result where the best predictor is given a value of 100, and the worst 0.

```
accuracy_results_red <- data.frame(Method = "Naive Bayes red variables",
 Accuracy = accuracy_bn_red)
accuracy_results_red
```

```
##                       Method  Accuracy
## 1 Naive Bayes red variables 0.7425432
```

For white wine

```
fit_whitenaive <- train(x = whiteTrain[, features_white], y = whiteTrain$quality,
                method ="nb",
                trControl = ctrl)
predict_whitenaive <- predict(fit_whitenaive, newdata = whiteTest[, features_white])
confMat_whitenaive <- confusionMatrix(predict_whitenaive, whiteTest$quality, positive = 'good')
importance_whitenaive <- varImp(fit_whitenaive, scale = TRUE)
plot(importance_whitenaive, main = 'Feature importance for Naive Bayes')
```




```
accuracy_bn_white <- max(fit_whitenaive$results$Accuracy)
```

```
accuracy_results_white <- data.frame(Method = "Naive Bayes white variables",
Accuracy = accuracy_bn_white)
accuracy_results_white
```

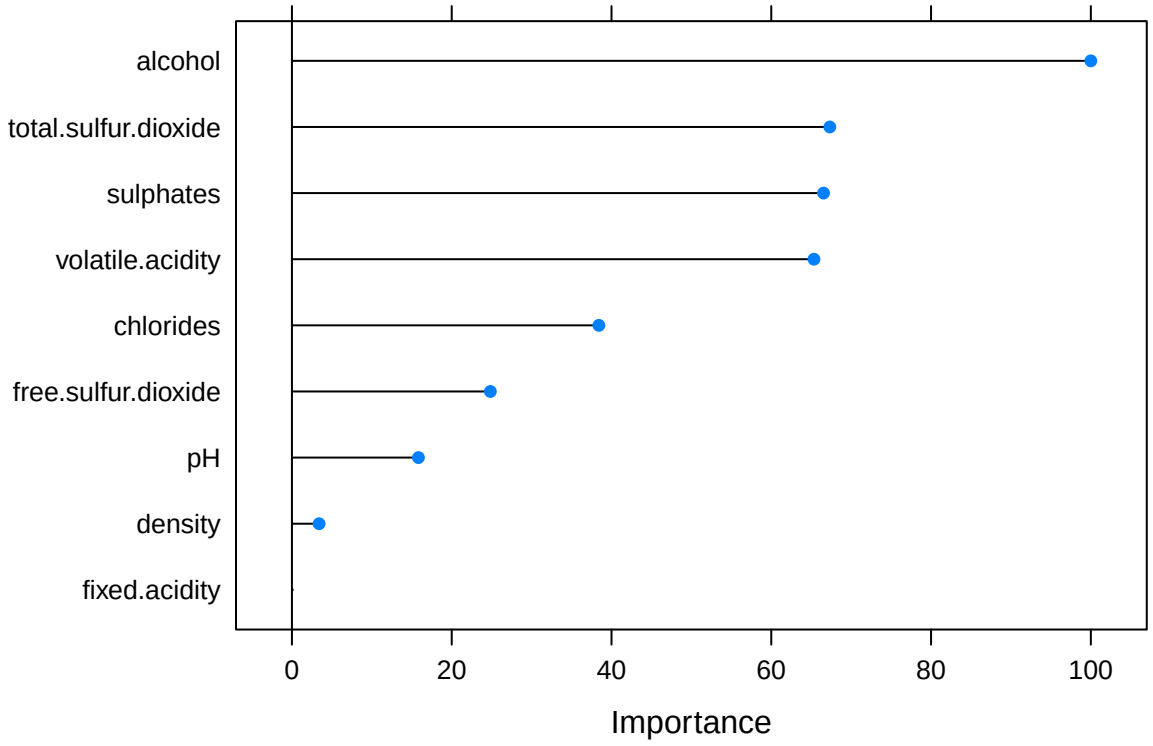
```
##                         Method  Accuracy
## 1 Naive Bayes white variables 0.7223704
```

11.2. Logistic Regression (GLM)

red wine

```
fit_redglm <- train(x = redTrain[, features_red], y = redTrain$quality,
                    method = 'glm',
                    preProcess = 'range',
                    trControl = ctrl)
predict_redglm <- predict(fit_redglm, newdata = redTest[, features_red])
confMat_redglm <- confusionMatrix(predict_redglm, redTest$quality, positive = 'good')
importance_redglm <- varImp(fit_redglm, scale = TRUE)

plot(importance_redglm, main = 'Feature importance for Logistic Regression Red wine')
```




```
accuracy_glm_red <- max(fit_redglm$results$Accuracy)
```

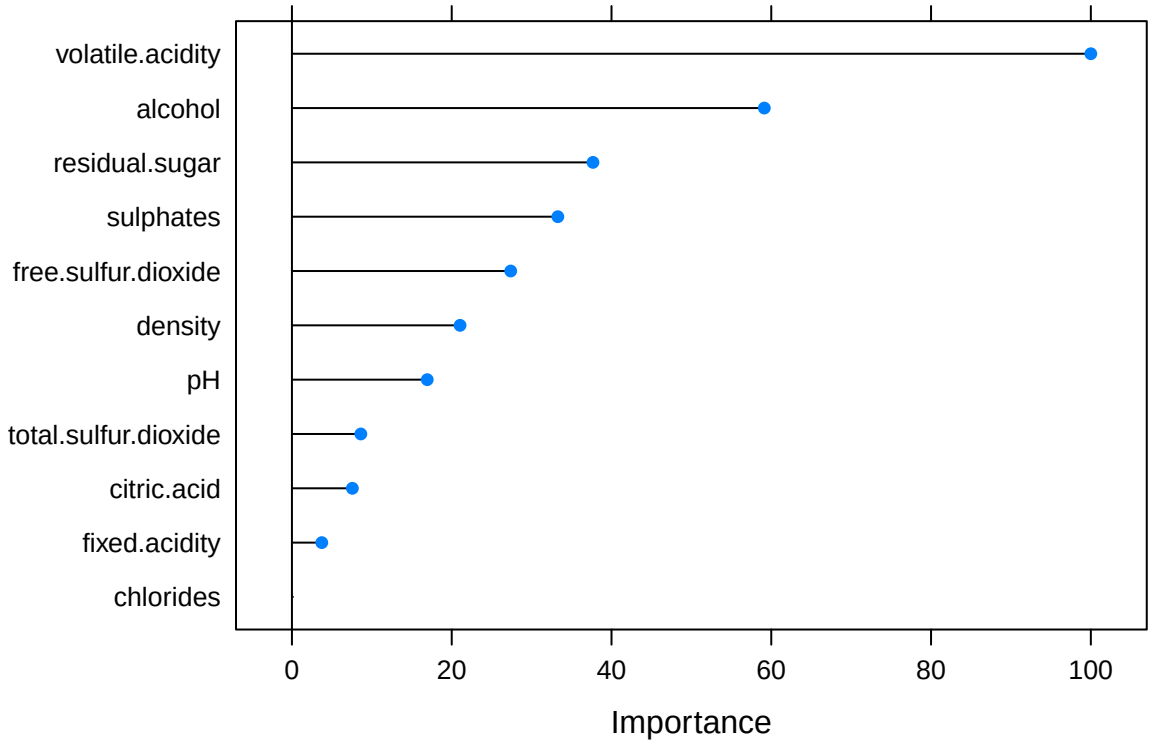
```
accuracy_results_red <- bind_rows(accuracy_results_red,
data_frame(Method=" Logistic Regression Model Red wine",Accuracy = accuracy_glm_red ))
accuracy_results_red
```

```
##                                Method  Accuracy
## 1           Naive Bayes red variables 0.7425432
## 2  Logistic Regression Model Red wine 0.7349564
```

white wine

```
fit_whiteglm <- train(x = whiteTrain[, features_white], y = whiteTrain$quality,
                 method = 'glm',
                 preProcess = 'range',
                 trControl = ctrl)
predict_whiteglm <- predict(fit_whiteglm, newdata = whiteTest[, features_white])
confMat_whiteglm <- confusionMatrix(predict_whiteglm, whiteTest$quality, positive = 'good')
importance_whiteglm <- varImp(fit_whiteglm, scale = TRUE)

plot(importance_whiteglm, main = 'Feature importance for Logistic Regression White wine')
```




```
accuracy_glm_white <- max(fit_whiteglm$results$Accuracy)
```

```
accuracy_results_white <- bind_rows(accuracy_results_white,
data_frame(Method=" Logistic Regression Model White wine",Accuracy = accuracy_glm_white ))
accuracy_results_white
```

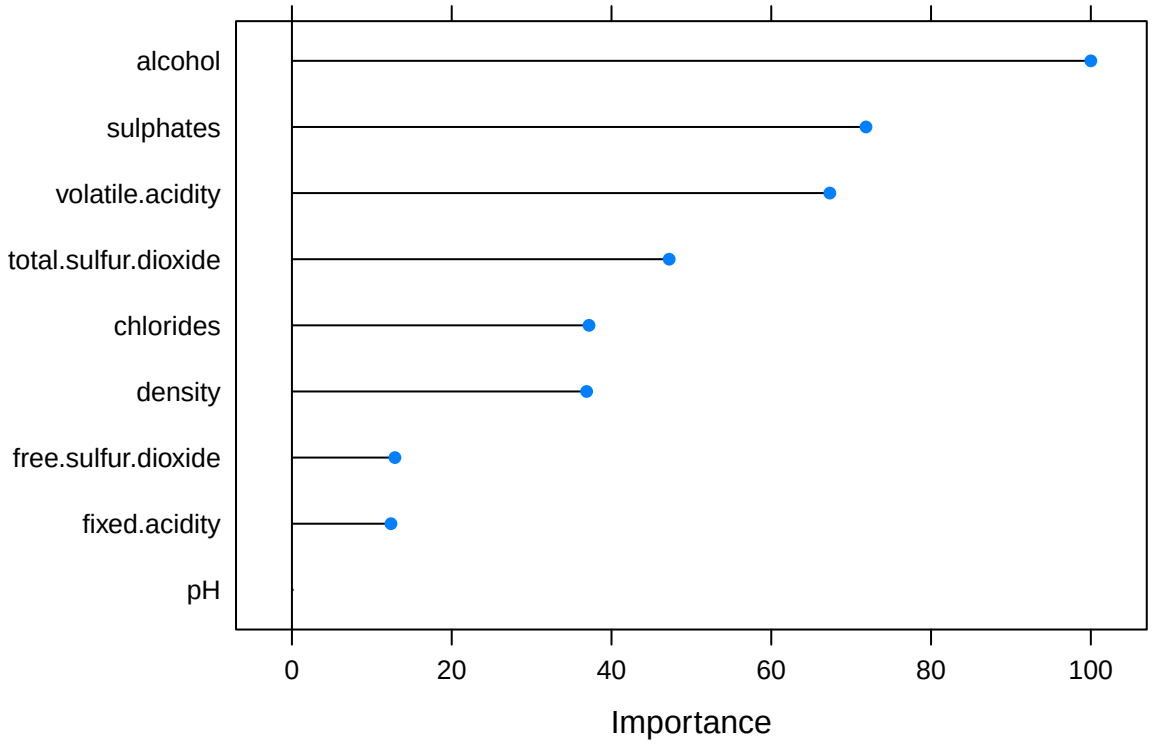
```
##                                 Method  Accuracy
## 1          Naive Bayes white variables 0.7223704
## 2  Logistic Regression Model White wine 0.7433331
```

11.3. K Nearest Neighbors (KNN)

red wine

```
fit_redknn <- train(x = redTrain[, features_red], y = redTrain$quality,
        method = 'knn',
        preProcess = 'range',
        trControl = ctrl,
        tuneGrid = expand.grid(.k = c(3, 5, 7, 9, 11, 15, 21, 25, 31, 41, 51, 75, 101)))
predict_redknn <- predict(fit_redknn, newdata = redTest[, features_red])
confMat_redknn <- confusionMatrix(predict_redknn, redTest$quality, positive = 'good')
importance_redknn <- varImp(fit_redknn, scale = TRUE)

plot(importance_redknn, main = 'Feature importance for K Nearest Neighbors Red wine')
```




```
accuracy_knn_red <- max(fit_redknn$results$Accuracy)
```

```
fit_redknn$bestTune
```

```
##    k
## 7 21
```

The nearest neighbors k=21 works best in terms of accuracy. Now that k is chosen, let's see how the model performs on the test set.

```
accuracy_results_red <- bind_rows(accuracy_results_red,
data_frame(Method=" KNN Model Red wine",Accuracy = accuracy_knn_red ))
accuracy_results_red
```

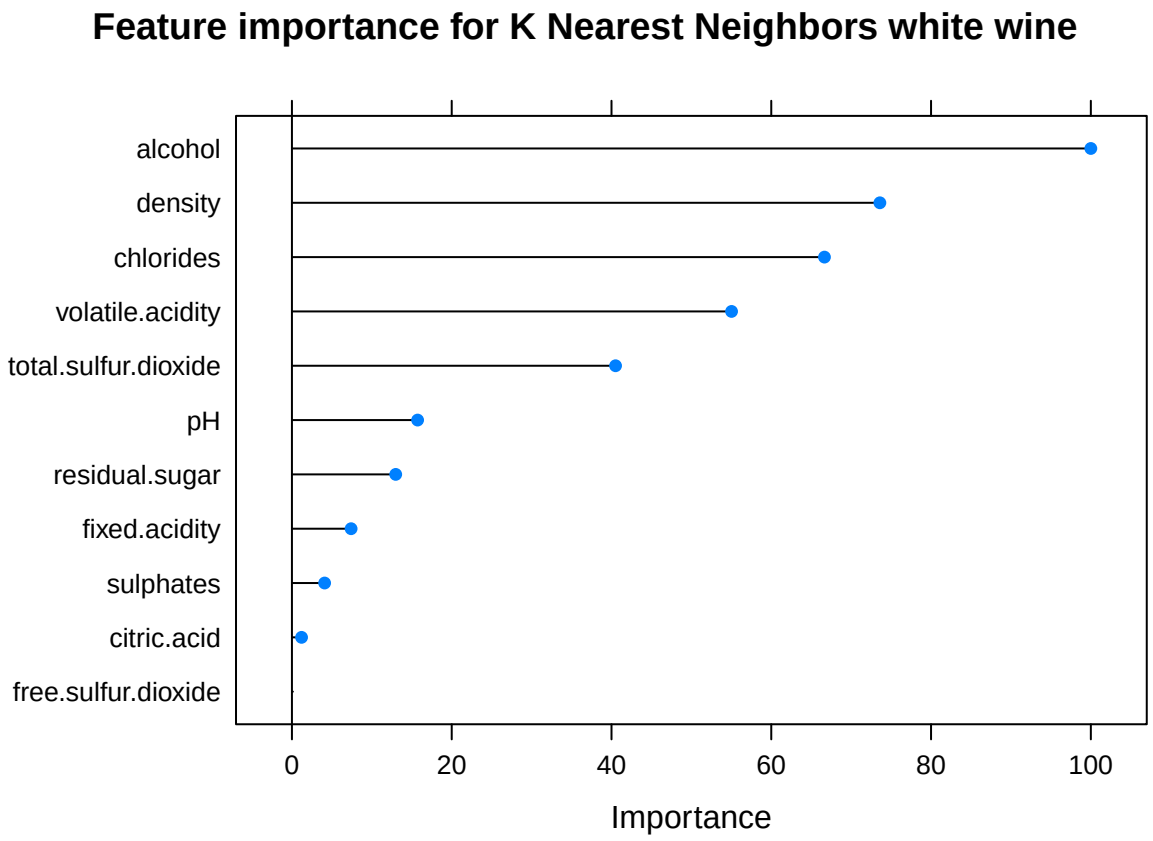
```
##                                Method  Accuracy
## 1           Naive Bayes red variables 0.7425432
## 2  Logistic Regression Model Red wine 0.7349564
## 3                  KNN Model Red wine 0.7391997
```

white wine

```
fit_whiteknn <- train(x = whiteTrain[, features_white], y = whiteTrain$quality,
      method = 'knn',
```

```
      preProcess = 'range',
     trControl = ctrl,
     tuneGrid = expand.grid(.k = c(3, 5, 7, 9, 11, 15, 21, 25, 31, 41, 51, 75, 101)))
predict_whiteknn <- predict(fit_whiteknn, newdata = whiteTest[, features_white])
confMat_whiteknn <- confusionMatrix(predict_whiteknn, whiteTest$quality, positive = 'good')
importance_whiteknn <- varImp(fit_whiteknn, scale = TRUE)

plot(importance_whiteknn, main = 'Feature importance for K Nearest Neighbors white wine')
```




```
accuracy_knn_white <- max(fit_whiteknn$results$Accuracy)
```

```
fit_whiteknn$bestTune
```

```
##   k
## 2 5
```

The nearest neighbors k=5 works best in terms of accuracy. Now that k is chosen, let's see how the model performs on the test set.

```
accuracy_results_white <- bind_rows(accuracy_results_white,
data_frame(Method=" KNN Model white wine",Accuracy = accuracy_knn_white ))
accuracy_results_white
```

```
##                                 Method  Accuracy
```

```
## 1           Naive Bayes white variables 0.7223704
## 2  Logistic Regression Model White wine 0.7433331
## 3                   KNN Model white wine 0.7681025
```

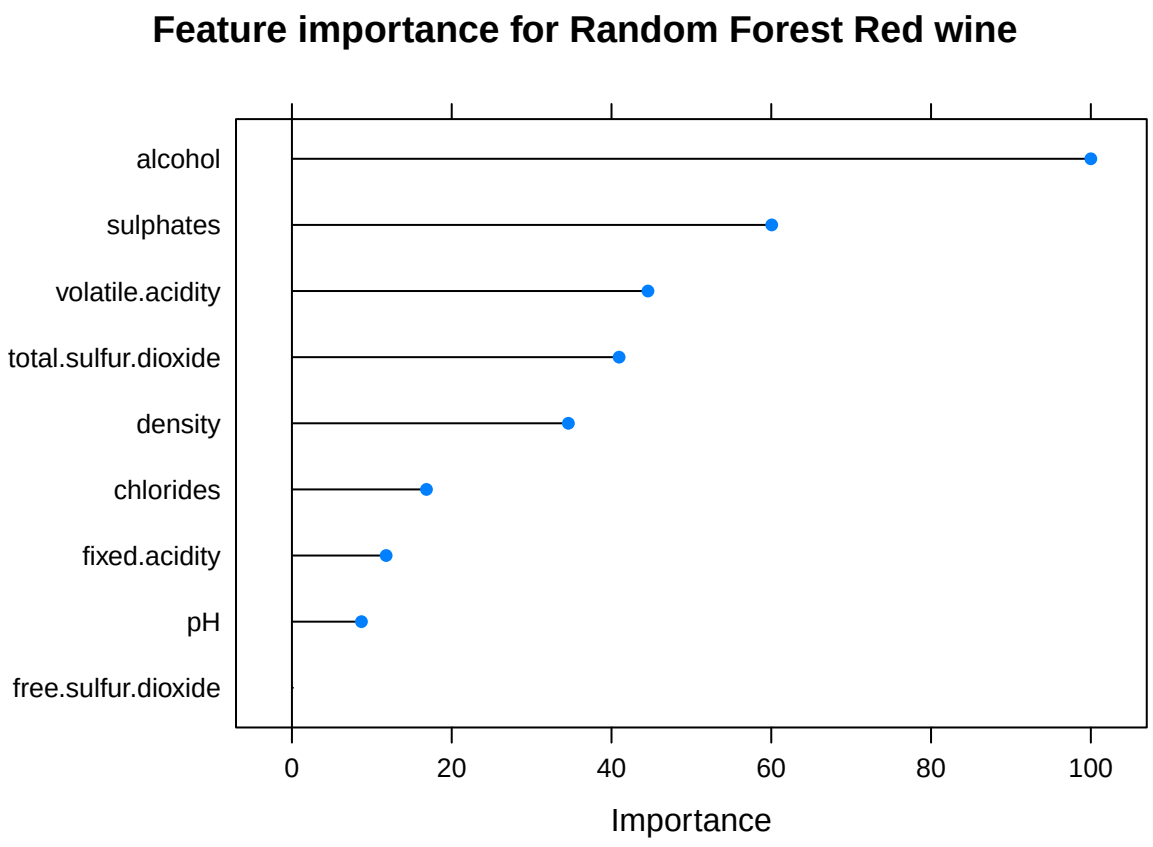
Really KNN doesn't give a good accuracy for red and white wine.

11.4. Random Forest (RF)

Red wine

```
fit_redrf <- train(x = redTrain[, features_red], y = redTrain$quality,
                   method = 'rf',
                   trControl = ctrl,
                  tuneGrid = expand.grid(.mtry = c(2:6)),
                  n.tree = 1000)
predict_redrf <- predict(fit_redrf, newdata = redTest[, features_red])
confMat_redrf <- confusionMatrix(predict_redrf, redTest$quality, positive = 'good')
importance_redrf <- varImp(fit_redrf, scale = TRUE)

plot(importance_redrf, main = 'Feature importance for Random Forest Red wine')
```




```
accuracy_rf_red <- max(fit_redrf$results$Accuracy)
```

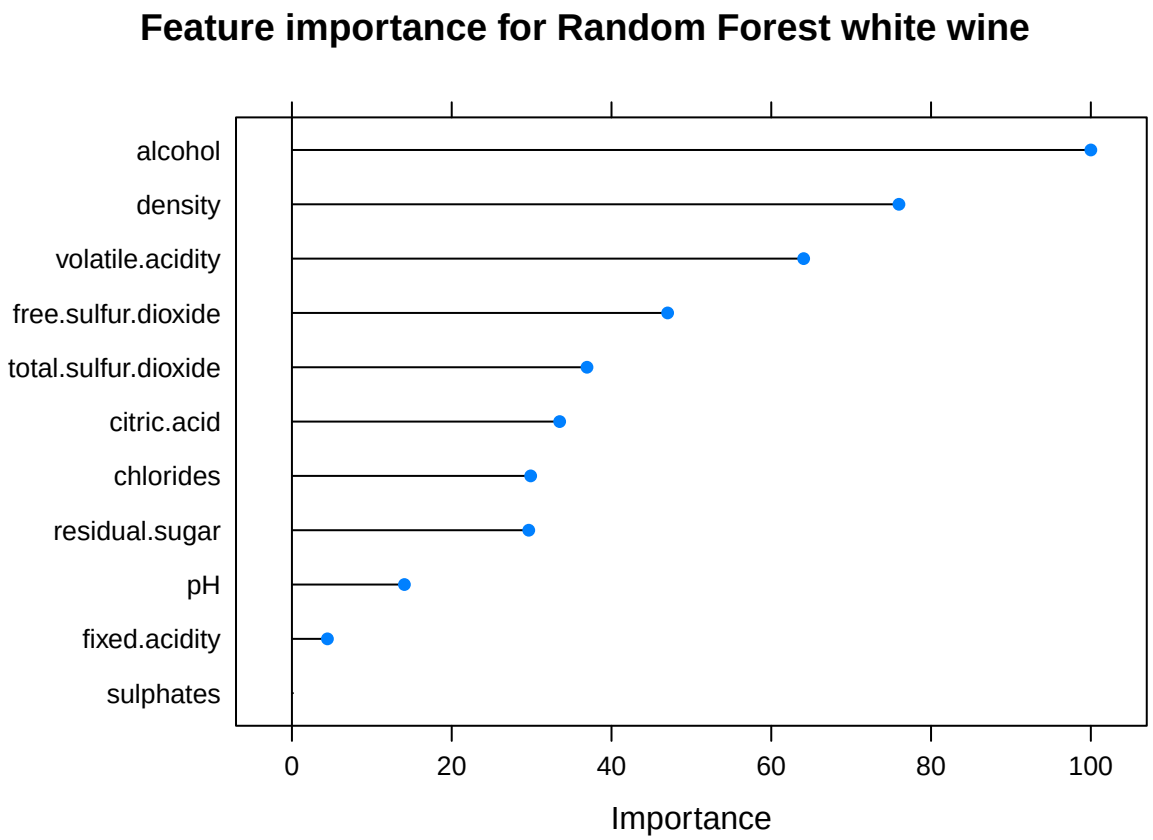
```
accuracy_results_red <- bind_rows(accuracy_results_red,
data_frame(Method=" Random Forest Model Red wine",Accuracy = accuracy_rf_red ))
accuracy_results_red
```

```
##                                Method  Accuracy
## 1           Naive Bayes red variables 0.7425432
## 2  Logistic Regression Model Red wine 0.7349564
## 3                  KNN Model Red wine 0.7391997
## 4        Random Forest Model Red wine 0.7975403
```

White wine

```
fit_whiterf <- train(x = whiteTrain[, features_white], y = whiteTrain$quality,
                 method = 'rf',
                 trControl = ctrl,
                tuneGrid = expand.grid(.mtry = c(2:6)),
                n.tree = 1000)
predict_whiterf <- predict(fit_whiterf, newdata = whiteTest[, features_white])
confMat_whiterf <- confusionMatrix(predict_whiterf, whiteTest$quality, positive = 'good')
importance_whiterf <- varImp(fit_whiterf, scale = TRUE)

plot(importance_whiterf, main = 'Feature importance for Random Forest white wine')
```



```
accuracy_rf_white <- max(fit_whiterf$results$Accuracy)
```

```
accuracy_results_white <- bind_rows(accuracy_results_white,
data_frame(Method=" Random Forest Model white wine",Accuracy = accuracy_rf_white ))
accuracy_results_white
```

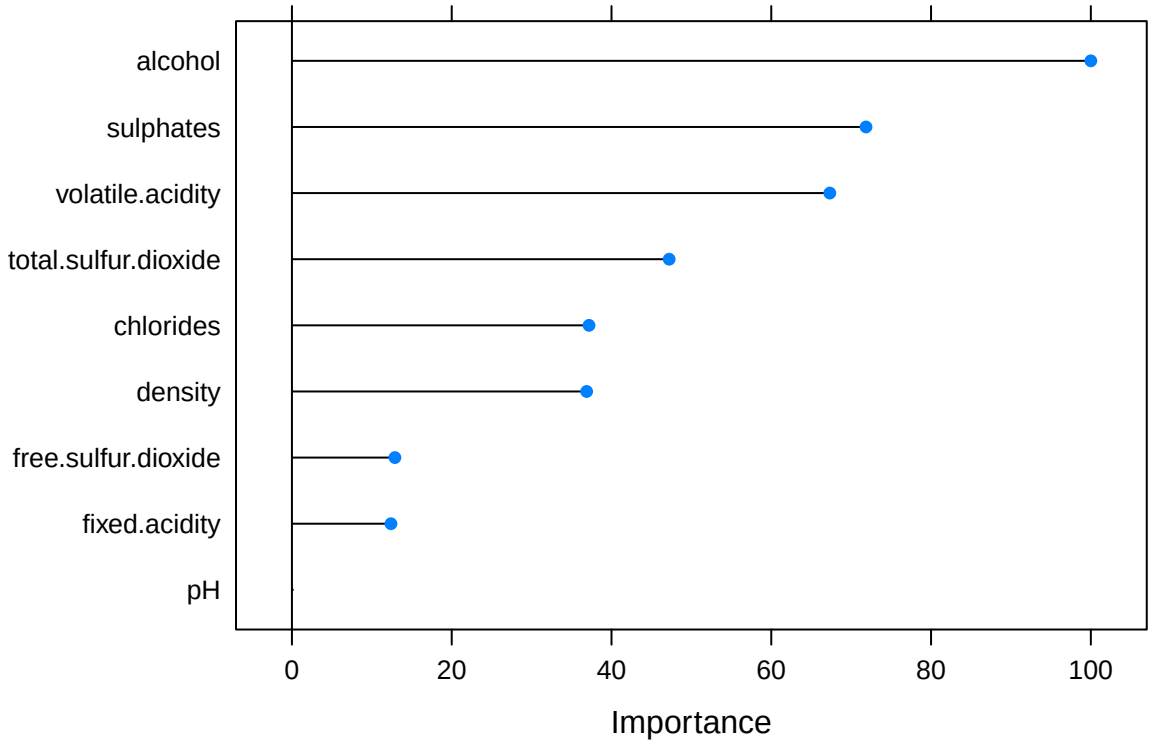
```
##                                 Method  Accuracy
## 1          Naive Bayes white variables 0.7223704
## 2  Logistic Regression Model White wine 0.7433331
## 3                  KNN Model white wine 0.7681025
## 4        Random Forest Model white wine 0.8298893
```

11.5. Support Vector Machines with linear kernel (svmLinear)

red wine

```
#  very slow
fit_redsvm <- train(x = redTrain[, features_red], y = redTrain$quality,
                 method = 'svmLinear',
                 preProcess = 'range',
                 trControl = ctrl,
                 tuneGrid = expand.grid(.C = c(0.001, 0.01, 0.1, 1, 10, 100)))
predict_redsvm <- predict(fit_redsvm, newdata = redTest[, features_red])
confMat_redsvm <- confusionMatrix(predict_redsvm, redTest$quality, positive = 'good')
importance_redsvm <- varImp(fit_redsvm, scale = TRUE)

plot(importance_redsvm, main = 'Feature importance for SVM-Linear red wine')
```




```
accuracy_svm_red <- max(fit_redsvm$results$Accuracy)
```

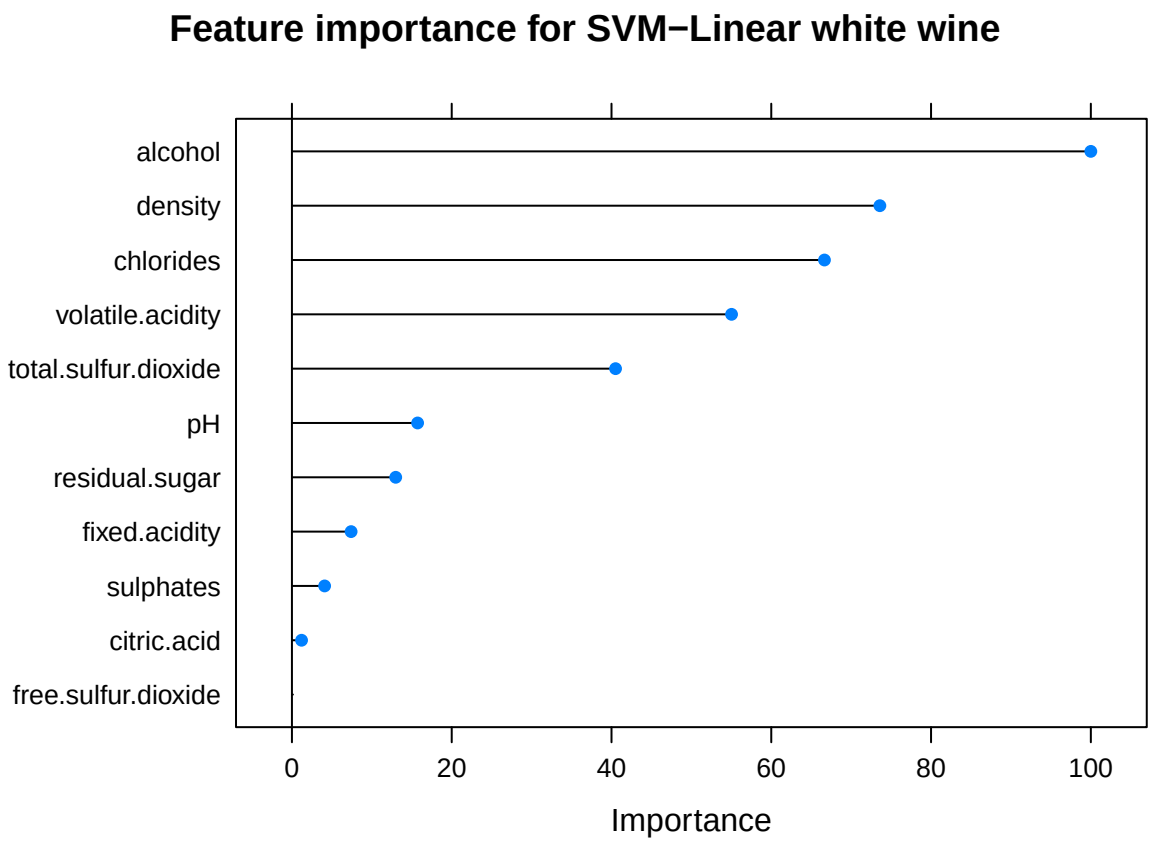
```
accuracy_results_red <- bind_rows(accuracy_results_red,
data_frame(Method=" SVM-Linear Model Red wine",Accuracy = accuracy_svm_red ))
accuracy_results_red
```

```
##                                Method  Accuracy
## 1           Naive Bayes red variables 0.7425432
## 2  Logistic Regression Model Red wine 0.7349564
## 3                  KNN Model Red wine 0.7391997
## 4        Random Forest Model Red wine 0.7975403
## 5           SVM-Linear Model Red wine 0.7300293
```

white wine

```
# very slow
fit_whitesvm <- train(x = whiteTrain[, features_white], y = whiteTrain$quality,
                 method = 'svmLinear',
                 preProcess = 'range',
                 trControl = ctrl,
                 tuneGrid = expand.grid(.C = c(0.001, 0.01, 0.1, 1, 10, 100)))
predict_whitesvm <- predict(fit_whitesvm, newdata = whiteTest[, features_white])
confMat_whitesvm <- confusionMatrix(predict_whitesvm, whiteTest$quality, positive = 'good')
importance_whitesvm <- varImp(fit_whitesvm, scale = TRUE)
```

```
plot(importance_whitesvm, main = 'Feature importance for SVM-Linear white wine')
```




```
accuracy_svm_white <- max(fit_whitesvm$results$Accuracy)
```

```
accuracy_results_white <- bind_rows(accuracy_results_white,
data_frame(Method=" SVM-Linear Model white wine",Accuracy = accuracy_svm_white ))
accuracy_results_white
```

```
##                                 Method  Accuracy
## 1          Naive Bayes white variables 0.7223704
## 2  Logistic Regression Model White wine 0.7433331
## 3                  KNN Model white wine 0.7681025
## 4        Random Forest Model white wine 0.8298893
## 5           SVM-Linear Model white wine 0.7457818
```

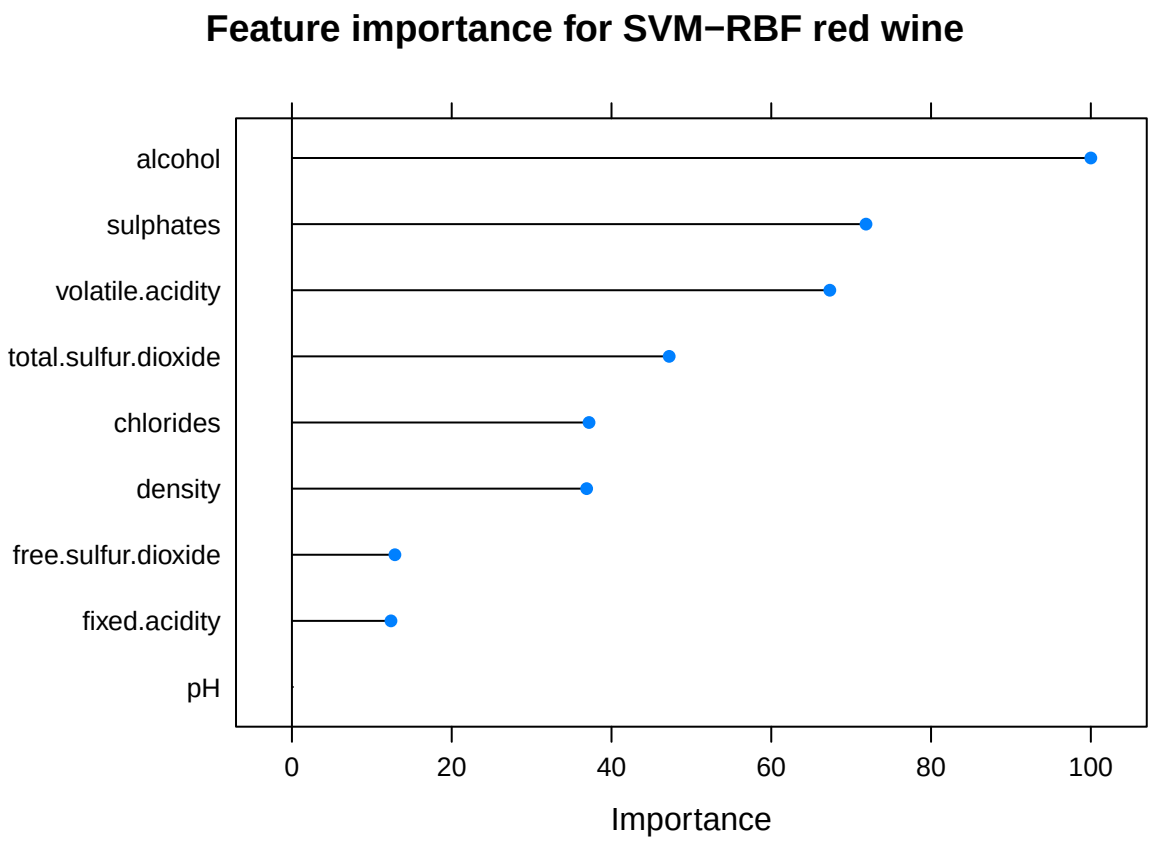
11.6. Support Vector Machines with Radial Basis Function (svmRBF)

Red wine

```
# very slow
fit_redsvmRBF <- train(x = redTrain[, features_red], y = redTrain$quality,
                    method = 'svmRadial',
                    preProcess = 'range',
```

```
                    trControl = ctrl,
                    tuneGrid = expand.grid(.C = c(0.001, 0.01, 0.1, 1, 10, 100),
                                           .sigma = c(0.001, 0.01, 0.1)))
predict_redsvmRBF <- predict(fit_redsvmRBF, newdata = redTest[, features_red])
confMat_redsvmRBF <- confusionMatrix(predict_redsvmRBF, redTest$quality, positive = 'good')
importance_redsvmRBF <- varImp(fit_redsvmRBF, scale = TRUE)

plot(importance_redsvmRBF, main = 'Feature importance for SVM-RBF red wine')
```




```
accuracy_svmRBF_red <- max(fit_redsvmRBF$results$Accuracy)
```

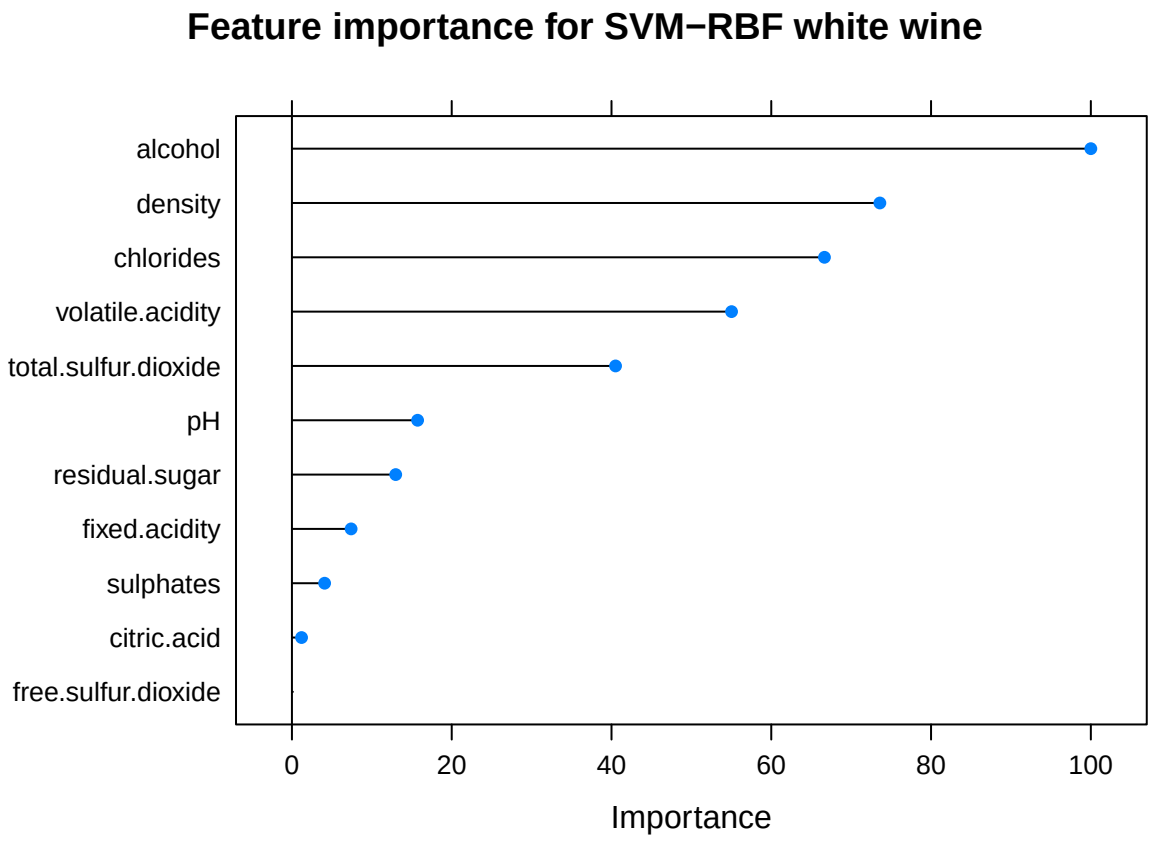
```
accuracy_results_red <- bind_rows(accuracy_results_red,
data_frame(Method=" SVM-RBF Model Red wine",Accuracy = accuracy_svmRBF_red ))
accuracy_results_red
```

```
##                                Method  Accuracy
## 1           Naive Bayes red variables 0.7425432
## 2  Logistic Regression Model Red wine 0.7349564
## 3                  KNN Model Red wine 0.7391997
## 4        Random Forest Model Red wine 0.7975403
## 5           SVM-Linear Model Red wine 0.7300293
## 6              SVM-RBF Model Red wine 0.7550575
```

White wine

```
# very slow
fit_whitesvmRBF <- train(x = whiteTrain[, features_white], y = whiteTrain$quality,
                     method = 'svmRadial',
                     preProcess = 'range',
                     trControl = ctrl,
                     tuneGrid = expand.grid(.C = c(0.001, 0.01, 0.1, 1, 10, 100),
                                            .sigma = c(0.001, 0.01, 0.1)))
predict_whitesvmRBF <- predict(fit_whitesvmRBF, newdata = whiteTest[, features_white])
confMat_whitesvmRBF <- confusionMatrix(predict_whitesvmRBF, whiteTest$quality, positive = 'good')
importance_whitesvmRBF <- varImp(fit_whitesvmRBF, scale = TRUE)

plot(importance_whitesvmRBF, main = 'Feature importance for SVM-RBF white wine')
```




```
accuracy_svmRBF_white <- max(fit_whitesvmRBF$results$Accuracy)
```

```
accuracy_results_white <- bind_rows(accuracy_results_white,
data_frame(Method=" SVM-RBF Model white wine",Accuracy = accuracy_svmRBF_white ))
accuracy_results_white
```

```
##                                Method  Accuracy
## 1           Naive Bayes white variables 0.7223704
## 2  Logistic Regression Model White wine 0.7433331
## 3                  KNN Model white wine 0.7681025
## 4        Random Forest Model white wine 0.8298893
```

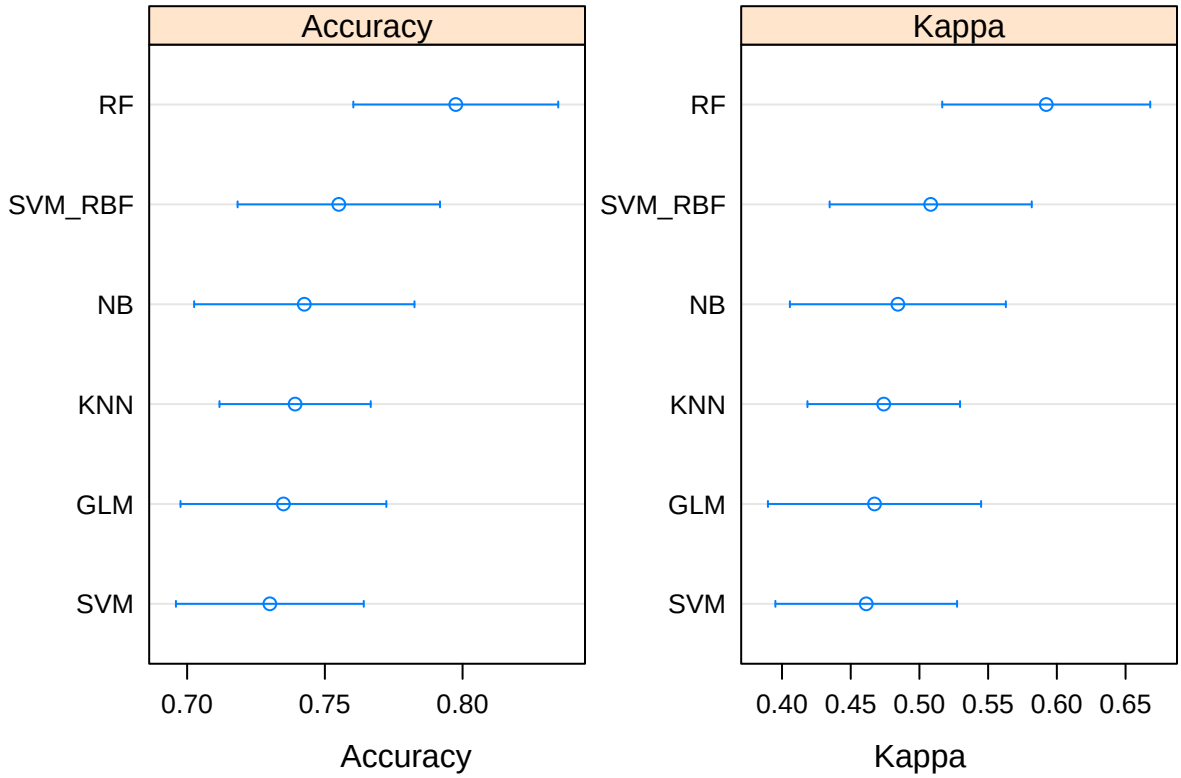
```
## 5           SVM-Linear Model white wine 0.7457818
## 6              SVM-RBF Model white wine 0.7827978
```

12. Results

Comparring all models for red wine

```
models_red <- resamples(list(NB = fit_rednaive, KNN = fit_redknn, GLM = fit_redglm,
                          SVM = fit_redsvm,
                          SVM_RBF = fit_redsvmRBF,
                          RF = fit_redrf))

scales <- list(x=list(relation="free"), y=list(relation="free"))
dotplot(models_red, scales=scales)
```



Accuracy for red wine :

```
accuracy_results_red %>% arrange(desc(Accuracy))
```

```
##                                Method  Accuracy
## 1       Random Forest Model Red wine 0.7975403
## 2             SVM-RBF Model Red wine 0.7550575
## 3          Naive Bayes red variables 0.7425432
## 4                 KNN Model Red wine 0.7391997
```

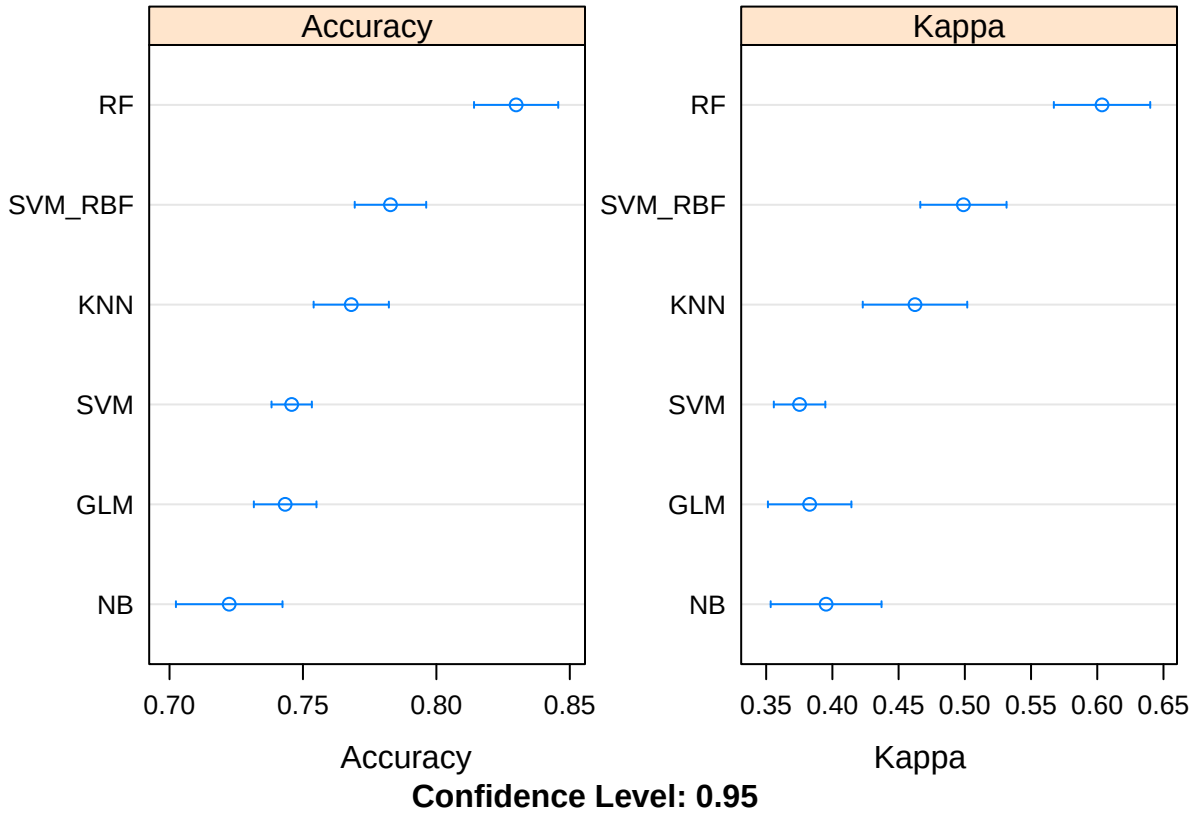
```
## 5  Logistic Regression Model Red wine 0.7349564
## 6           SVM-Linear Model Red wine 0.7300293
```

The best model for red wine is the **Random Forest Model**

Comparring all models for white wine

```
models_white <- resamples(list(NB = fit_whitenaive, KNN = fit_whiteknn, GLM = fit_whiteglm,
                          SVM = fit_whitesvm,
                          SVM_RBF = fit_whitesvmRBF,
                          RF = fit_whiterf))

scales <- list(x=list(relation="free"), y=list(relation="free"))
dotplot(models_white, scales=scales)
```



Accuracy for white wine :

```
accuracy_results_white %>% arrange(desc(Accuracy))
```

```
##                                Method  Accuracy
## 1        Random Forest Model white wine 0.8298893
## 2              SVM-RBF Model white wine 0.7827978
## 3                  KNN Model white wine 0.7681025
## 4            SVM-Linear Model white wine 0.7457818
## 5  Logistic Regression Model White wine 0.7433331
## 6           Naive Bayes white variables 0.7223704
```

The best model for white wine is the **Random Forest Model**

13. Conclusion

The two most important features among all attributes are Sulphur dioxide (both free and total) and Alcohol. Volatile acidity contributes to acidic tastes and have negative correlation to wine quality.
The most important factor to decide the quality of wine is alcohol, higher concentration of alcohol leads to better quality of wine and lower density of wine.

Random forest is a method of 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. This method has the highest score than the other models.