HarvardX Professional Certificate in Data Science

edX

Capstone Project: Quality of Wine

ABSTRACT

Evaluation of quality of wine from dataset provided by UCI (source: Paulo Cortez, University of Minho, Guimarães, Portugal, http://www3.dsi.uminho.pt/pcortez)

Seshan Senga

Contents

Introduction

Wine Quality Data Set from UCI Machine Learning Repository, related to the Portuguese “Vinho Verde” wine https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/

- The dataset contain:
  - 1,599 observations for red wine and 4898 for white wine.
  - 11 continuous numerical independent variables which are physicochemical attributes, some of which may be correlated.
  - 1 discrete numerical response variable which is sensory data in an 1-to-10 scale

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For more details, consult: [Web Link] or the reference [Cortez et al., 2009]. Due to privacy and logistic obstacles, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones). Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, we are not sure if all input variables are relevant. It will be interesting to test feature selection methods.

Attribute Information:

1 - Fixed acidity: most acids involved with wine or fixed or nonvolatile (do not evaporate readily) (tartaric acid - g / dm^3)

2 - Volatile acidity: the amount of acetic acid in wine, which at very high levels can lead to an unpleasant, vinegar taste (acetic acid - g / dm^3)

3 - Citric acid: found in small quantities, citric acid can add ‘freshness’ and flavor to wines (g / dm^3)

4 - Residual sugar: the amount of sugar remaining after fermentation stops, it’s rare to find wines with less than 1 gram/liter and wines with greater than 45 grams/liter are considered sweet (g / dm^3)

5 - Chlorides: the amount of salt in the wine (sodium chloride - g / dm^3

6 - Free sulfur dioxide: the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion; it prevents microbial growth and the oxidation of wine (mg / dm^3)

7 - Total sulfur dioxide: amount of free and bound forms of S02. In low concentrations, SO2 is mostly undetectable in wine, but at free SO2 concentrations over 50 ppm, SO2 becomes evident in the nose and taste of wine (mg / dm^3)

8 - Density: the density of water is close to that of water depending on the percent alcohol and sugar content (g / cm^3)

9 - pH: describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines fall between 3-4 on the pH scale

10 - Sulphates: a wine additive which can contribute to sulfur dioxide gas (S02) levels, which acts as an antimicrobial and antioxidant (potassium sulphate - g / dm3)

11 - Alcohol: the percent alcohol content of the wine (% by volume)

Relevant Papers:

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

The objective of this project is be to find the quality of the wine using various machine learning models.

Data analysis and visualisation (Red Wine)

The following library set will be used to process this data

```
library(ggplot2)# Create Elegant Data Visualisations
library(caret) # For model tuning
library(corrgram) # For correlation matrix
library(corrplot)# For graphical display of correlation matrix
library(gridExtra) # For multi-object layouts
library(car)# For smoother scatterplots
library(MASS) # For MASS function
library(GGally) #extension of ggplot2
library(dplyr) # For select and mutate function
```

The next step is to import the dataset for analysis. In this case, I have already downloaded the data on to my local machine as the antivirus setup in my machine didn't permit me to download from UCI portal directly.

```
#Importing the data of the red wine for the first set of analysis

wine = read.csv("D:/Harvard/winequality-red.csv", sep = ";", header = T)
```

For this analysis we analyse the red wine dataset first followed by the white wine dataset.

View the structure and summary of red wine data to understand the content.

```
> str(wine)
'data.frame':   1599 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.07
 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
>
```

Checking the names of the variables which are part of red wine dataset

```
> #check the names of the variables present in the data.
> colnames(wine)
 [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
 [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
 [7] "total.sulfur.dioxide" "density"              "pH"
[10] "sulphates"            "alcohol"              "quality"
>
```

```
> summary(wine)
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
   chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
 Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
 1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
 Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
 Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
 3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
 Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
       pH          sulphates         alcohol         quality
 Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
 1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
 Median :3.310   Median :0.6200   Median :10.20   Median :6.000
 Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
 3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
 Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
>
```

The next step is to remove any duplicate information in the dataset as part of data cleaning.

```
> #Removing the Duplicate Rows
> wine <- wine[!duplicated(wine), ]
> dim(wine)
[1] 1359   12
```

As you can see the rows are reduced to 1359, from 1599

```
> str(wine)
'data.frame':   1359 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.9 7.3 7.8 7.5 6.7 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.66 0.6 0.65 0.58 0.5 0.58 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0.06 0 0.02 0.36 0.08 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.8 1.6 1.2 2 6.1 1.8 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.075 0.069 0.065 0.073 0.071
 0.097 ...
 $ free.sulfur.dioxide : num  11 25 15 17 13 15 15 9 17 15 ...
 $ total.sulfur.dioxide: num  34 67 54 60 40 59 21 18 102 65 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.3 3.39 3.36 3.35 3.28 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.46 0.47 0.57 0.8 0.54 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 10 9.5 10.5 9.2 ...
 $ quality             : int  5 5 5 6 5 5 7 7 5 5 ...
```

Check for NA in the dataset

```
> #Check for NA in dataset
> sum(is.na(wine))
[1] 0
> |
```

No NA found in the dataset.

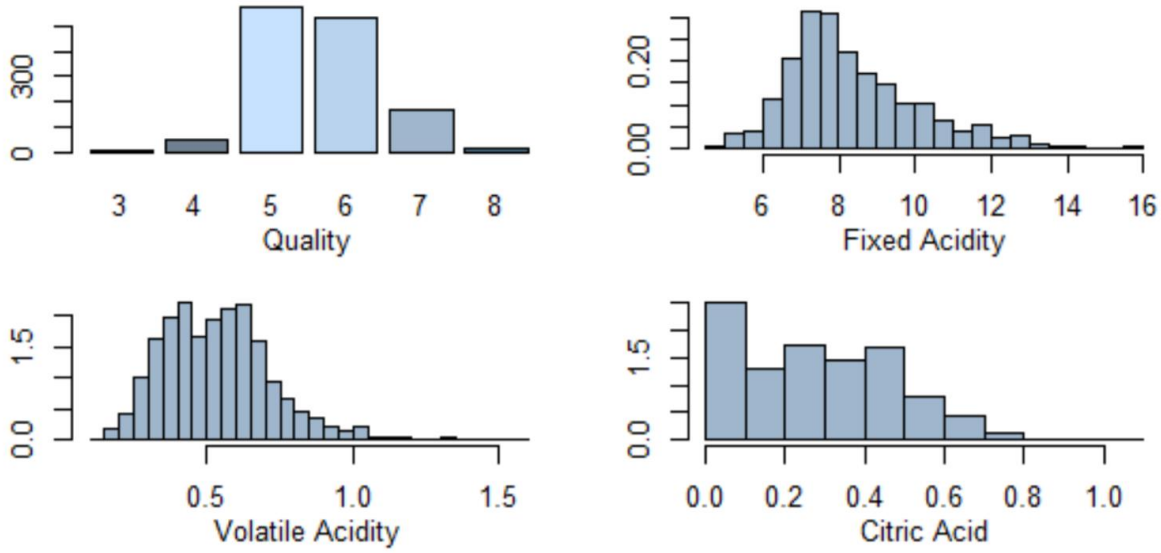
Let's now find out if the response count are skewed.

```
> #Response count in the dataset
> table(wine$quality)

  3   4   5   6   7   8
 10  53 577 535 167  17
> |
```

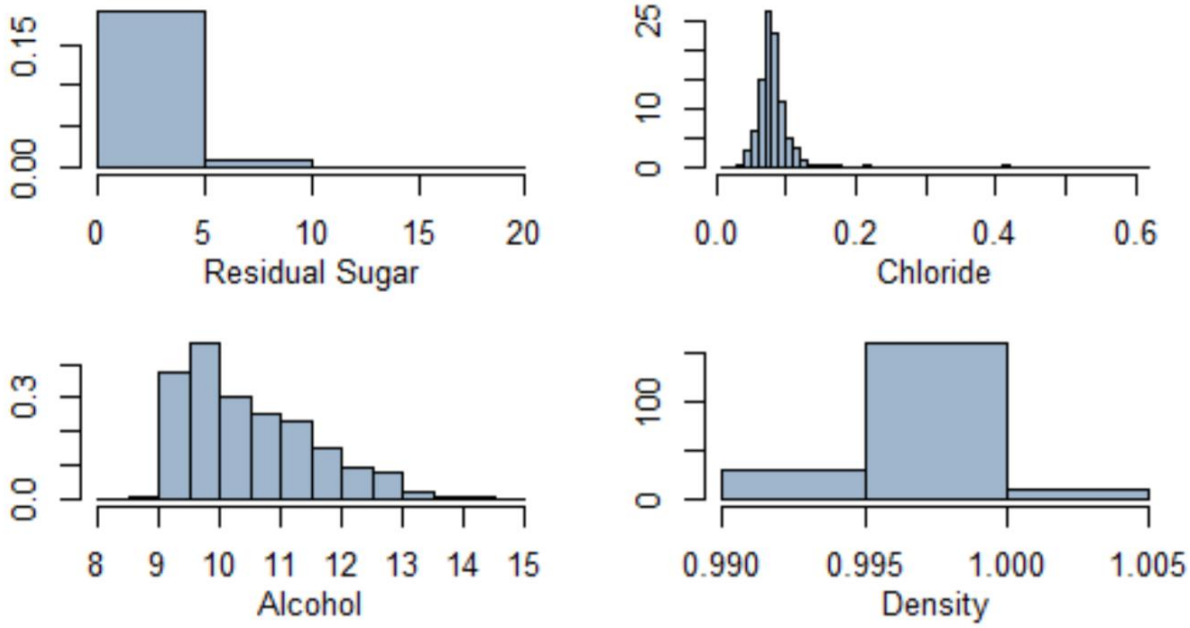
We can see that the class is skewed, it unbalanced.
There are 1359 samples but only 10 points are from 3rd class and only 17 points are from the 8th class. Wines with the lowest and highest quality are rare. Generally, wines contains quality 5 and 6 which is in the medium range (not low, not high).

The next step is to further analysis the data for gather more information. We start by plotting a histogram of the dataset

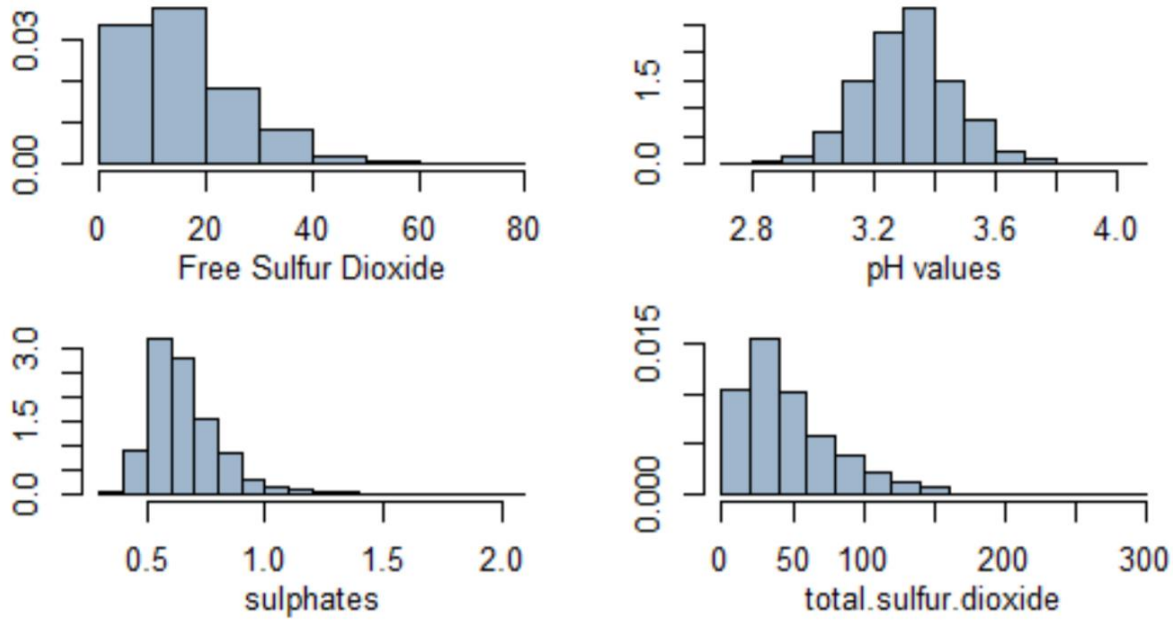
```
> #Plotting the histograms using hist() for the data.
>
> #plot comparison of "QUALITY", "FIXED ACIDITY", "VOLATILE ACIDITY", "CITRIC A
CID"
>
> attach(wine)
>
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) + 0.1)
> barplot((table(quality)), col=c("slateblue4", "slategray", "slategray1", "sla
tegray2", "slategray3", "skyblue4"))
> mtext("Quality", side=1, outer=F, line=2, cex=0.8)
>
>
> truehist(fixed.acidity, h = 0.5, col="slategray3")
> mtext("Fixed Acidity", side=1, outer=F, line=2, cex=0.8)
>
> truehist(volatile.acidity, h = 0.05, col="slategray3")
> mtext("Volatile Acidity", side=1, outer=F, line=2, cex=0.8)
>
> truehist(citric.acid, h = 0.1, col="slategray3")
> mtext("Citric Acid", side=1, outer=F, line=2, cex=0.8)
>
```



```
> #PLOT COMPARISON OF "RESIDUAL SUGAR", "CHLORIDE", "ALCHOL", "DEN
SITY"
>
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) + 0.
1)
>
> truehist(residual.sugar, h = 5, col="slategray3")
> mtext("Residual Sugar", side=1, outer=F, line=2, cex=0.8)
>
> truehist(chlorides, h = 0.01, col="slategray3")
> mtext("Chloride", side=1, outer=F, line=2, cex=0.8)
>
> truehist(alcohol, h = 0.5, col="slategray3")
> mtext("Alcohol", side=1, outer=F, line=2, cex=0.8)
>
>
> truehist(density, h = 0.005, col="slategray3")
> mtext("Density", side=1, outer=F, line=2, cex=0.8)
>
```



```
> #PLOT COMPARISON OF "FREE SULFUR DIOXIDE", "pH VALUES", "SULPHAT
ES", "TOTAL.SULFUR.DIOXIDE"
>
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) + 0.
1)
>
> truehist(free.sulfur.dioxide, h = 10, col="slategray3")
> mtext("Free Sulfur Dioxide", side=1, outer=F, line=2, cex=0.8)
>
> truehist(pH, h = 0.1, col="slategray3")
> mtext("pH values", side=1, outer=F, line=2, cex=0.8)
>
> truehist(sulphates, h = 0.1, col="slategray3")
> mtext("sulphates", side=1, outer=F, line=2, cex=0.8)
>

> truehist(total.sulfur.dioxide, h = 20, col="slategray3")
> mtext("total.sulfur.dioxide", side=1, outer=F, line=2, cex=0.8)
>
```



From the information displayed above, the quality rating ranges from 3 to 8, focusing on 5 and 6.

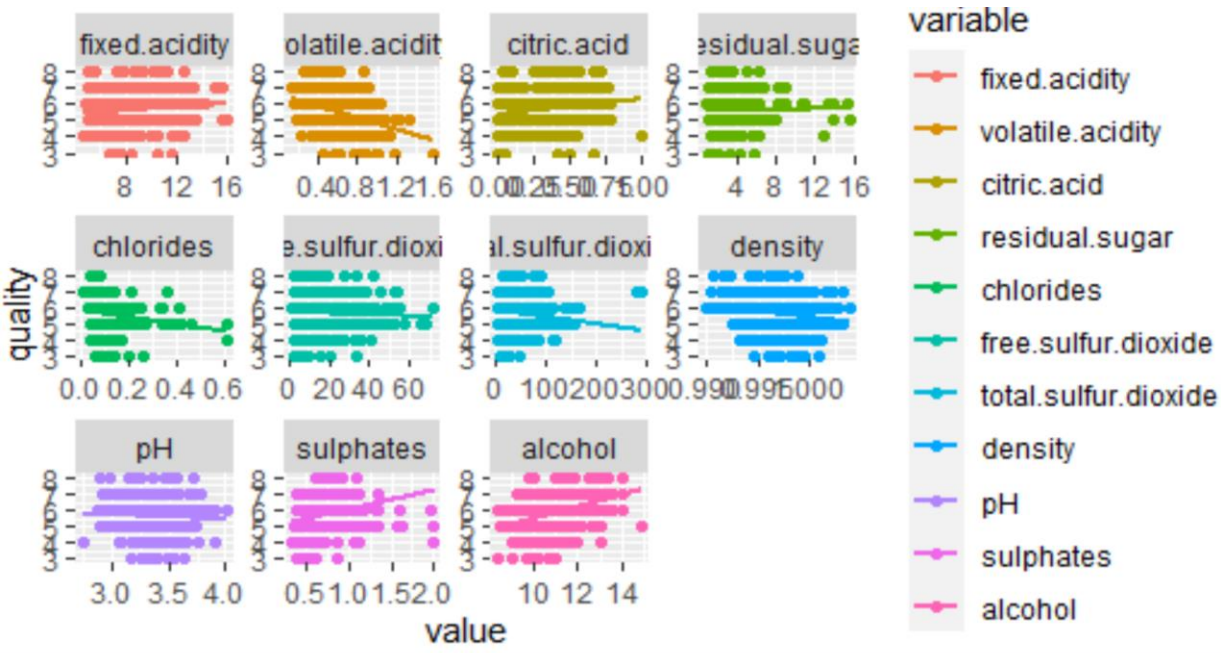
Density and pH are distributed normally, while the majority of other variables are skewed to the right.

Volatile acidity, sulphates and alcohol appear to have the strongest relationship with quality. Free sulfur dioxide and residual sugar appear to have the weakest relationships with quality.

Some of the predicator variables appear to have a fewer outliners.

Let's focus on the relationship between attributes and quality.

```
> #Relationship between each Attribute and Quality (Rating)
>
> red_wine <- melt(wine, "quality")
> ggplot(red_wine, aes(value, quality, color = variable)) +
+   geom_point() +
+   geom_smooth(aes(value,quality, colour=variable), method=lm, se
=FALSE)+
+   facet_wrap(.~variable, scales = "free")
`geom_smooth()` using formula 'y ~ x'
>
```

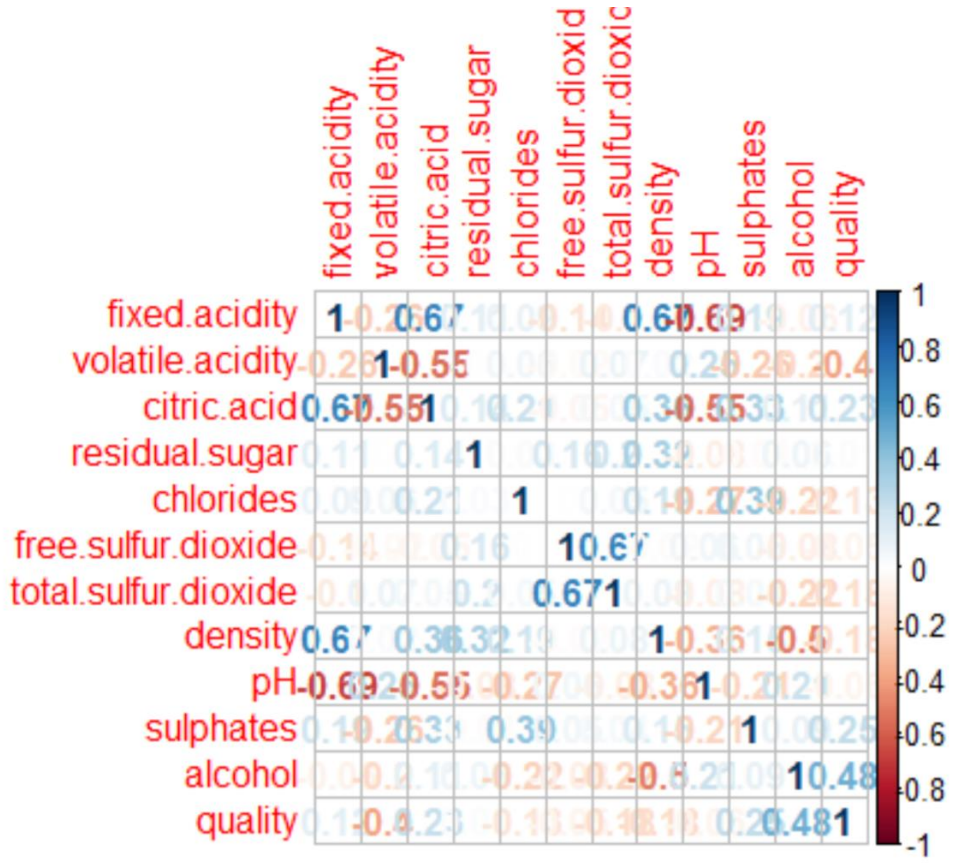


Alcohol and sulphates seem to have a positive correlation with quality

Volatile acidity, chlorides, and total sulfur dioxide appears to have a negative relationship with quality

I will now find the collinearity between the attributes using following code.

```
> #Collinearity between Attributes
>
> par(mfrow = c(1,1))
> cor.wine <- cor(wine)
> corrplot(cor.wine, method = 'number')
>
```



There is a significant and a major positive correlation between density & fixed acidity, fixed acidity & citric acid, as well as free sulfur dioxide & total sulfur dioxide.

There's a significantly negative correlation between fixed acidity & pH

Data Split (Red Wine)

I will convert the quality response of wine into factors first and then split the data into a training and test dataset.

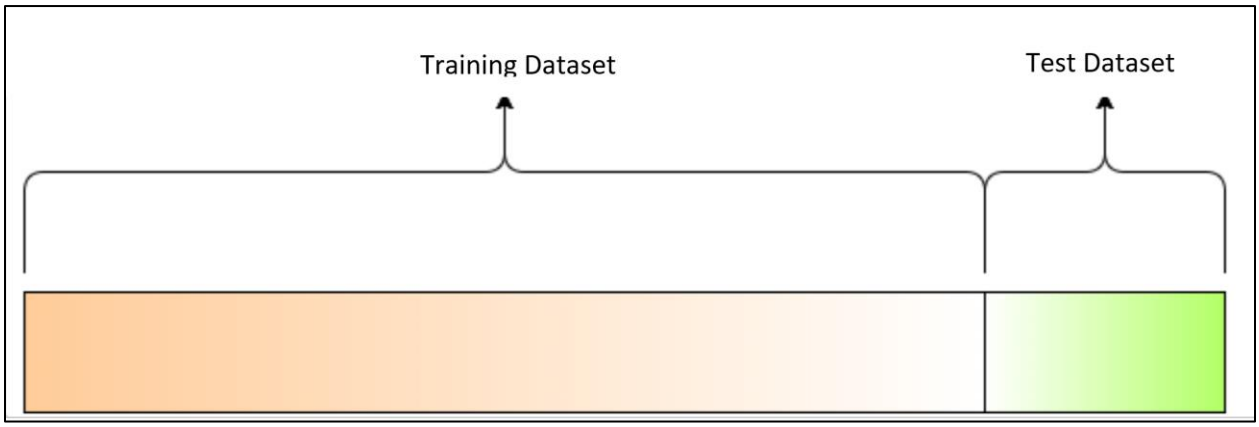
```
> wine$quality <- as.factor(wine$quality)
> str(wine)
'data.frame':   1359 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.9 7.3 7.8 7.5 6.7
 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.66 0.6 0.65 0.58 0.
5 0.58 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0.06 0 0.02 0.36 0.08
 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.8 1.6 1.2 2 6.1 1.8
 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.075 0.069 0.06
5 0.073 0.071 0.097 ...
 $ free.sulfur.dioxide : num  11 25 15 17 13 15 15 9 17 15 ...
 $ total.sulfur.dioxide: num  34 67 54 60 40 59 21 18 102 65 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.3 3.39 3.36 3.
35 3.28 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.46 0.47 0.57
 0.8 0.54 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 10 9.5 10.5 9.2
 ...
 $ quality             : Factor w/ 6 levels "3","4","5","6",..: 3 3 3
 4 3 3 5 5 3 3 ...
> |
```

The quality response of the red wine dataset has been changed to factor. Ensuring that class of the vectors match each other before doing the split into training and test dataset.

The next step is to split the data: before splitting it is important to understand why we split the data into training and test. The reason being overfitting. Overfitting is a major challenge for machine learning. This means that the model has been trained too well and the model believes it as a reliable pattern. Overfitting affects the performance of the model when it is exposed to untrained data, this phenomenon is known as generalisation.

To overcome the problem of overfitting, we will split the data into training and test data set. I am considering splitting two thirds of the dataset into training dataset while using the remaining third as the test dataset.

The model learns this data in order to be generalised to other data in the process, after which model will predict values for the 1/3 test dataset. Finally by calculating RMSE, we can determine the prediction accuracy of the model.



```
> #setting the seed value to 1
> set.seed(1)
>
> #splitting the dataset into train and test (2/3rd for train remainin
g for test)
>
> inTrain <- createDataPartition(wine$quality, p = 2/3, list = F)
> train <- wine[inTrain,]
> test <- wine[-inTrain,]
>
```

Modeling and Evaluation:

I will be using two methods to train dataset and observe the results in this part, I will start with the red wine dataset and then move onto white wine. Finally, comparing the results in the conclusion.

I would be using k-nearest neighbours and random forest to model the data.

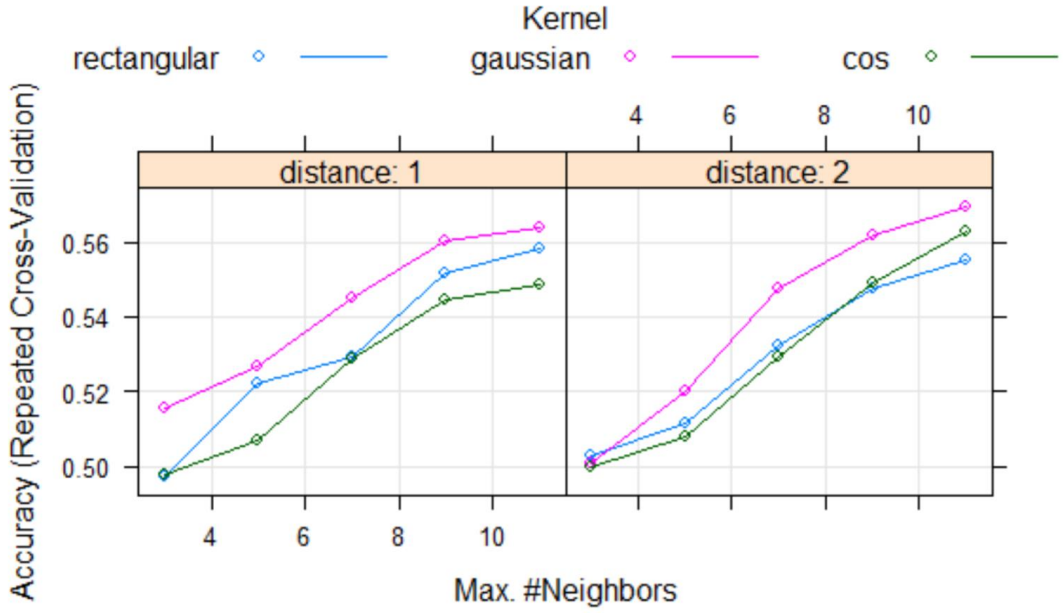
K-nearest neighbours:

I would be using Caret's train function to simplify model tuning. The tunegrid and expand grind functions will be used to combine the selected parameters in possible combinations.

K-nearest neighbours uses distance to classify the response variable. Hence, it is necessary to standardise the predictor variables. This is done to prevent predictors with larger ranges from being over-emphasized by the algorithm. The preprocess argument in the train function will be used to center and scale the predictors, i.e. standardization.

For k-nearest neighbours, five (5) kmax, two (2) distance, and three (3) kernel values will be used. For the distance value, one (1) is the Manhattan distance, and two (2) is the Euclidian distance.

```
> #K-nearest neighbour model
>
> t.ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)
> knn.grid <- expand.grid(kmax = c(3, 5, 7 ,9, 11), distance = c(1, 2),
+                         kernel = c("rectangular", "gaussian", "cos"))
> knn.train <- train(quality ~ ., data = train, method = "knn",
+                    trControl = t.ctrl, tuneGrid = knn.grid,
+                    preProcess = c("center", "scale"))
>
> save(knn.train, file = "knn.train.rda")
>
> # saveRDS(model, "model.rds")
> # my_model <- readRDS("model.rds")
> # This lets you to choose a new name for the object (you don't need to reme
mber the name you used when you saved it)
>
> plot(knn.train)
>
```




```
> kknn.train$bestTune
   kmax distance   kernel
29   11        2 gaussian
>
```

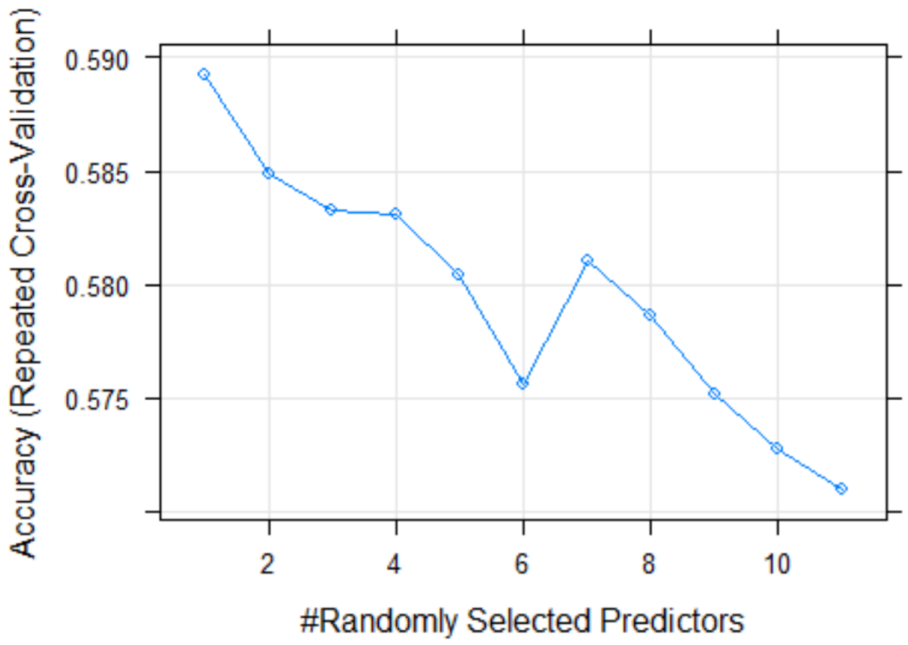
The gaussian kernel of distance 2 has outperformed others.

Let's move on to next model before predicting the results of both these models.

RandomForest

For the random forest, only the mtry hyperparameter is available for tuning. Mtry values of one to eleven (1-11) will be passed into the train function's tuneGrid argument. Mtry is the number of variables randomly sampled as candidates at each split.

```
#Random Forest
rf.grid <- expand.grid(mtry = 1:11)
rf.train <- train(quality ~ ., data = train, method = "rf",
                  trControl = t.ctrl, tuneGrid = rf.grid,
                  preProcess = c("center", "scale"))

save(rf.train, file = "rf.train.rda")
plot(rf.train)

load("rf.train.rda")
```




```
> rf.train$bestTune
  mtry
1    1
> |
```

An mtry of one (1) is likely using univariate decision trees. I'm going to keep mtry = 1 as the best value.

Best model prediction

The models we're trained using the accuracy metric. Kappa is important as well and we will take that into account when evaluating model performance.

K-Nearest Neighbour Prediction Results:

```
> kknn.predict <- predict(kknn.train, test)
> confusionMatrix(kknn.predict, test$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8
         3   0   0   0   0   0   0
         4   0   1   1   0   1   0
         5   3  13 128  61   5   0
         6   0   3  60  98  28   1
         7   0   0   3  19  20   4
         8   0   0   0   0   1   0

Overall Statistics

               Accuracy : 0.5489
                 95% CI : (0.5016, 0.5955)
    No Information Rate : 0.4267
    P-Value [Acc > NIR] : 1.238e-07
```

```
                  Kappa : 0.2737

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7
Sensitivity          0.000000 0.058824   0.6667   0.5506  0.36364
Specificity          1.000000 0.995381   0.6822   0.6618  0.93418
Pos Pred Value            NaN 0.333333   0.6095   0.5158  0.43478
Neg Pred Value       0.993333 0.964206   0.7333   0.6923  0.91337
Prevalence           0.006667 0.037778   0.4267   0.3956  0.12222
Detection Rate       0.000000 0.002222   0.2844   0.2178  0.04444
Detection Prevalence 0.000000 0.006667   0.4667   0.4222  0.10222
Balanced Accuracy    0.500000 0.527102   0.6744   0.6062  0.64891
                     Class: 8
Sensitivity          0.000000
Specificity          0.997753
Pos Pred Value       0.000000
Neg Pred Value       0.988864
Prevalence           0.011111
Detection Rate       0.000000
Detection Prevalence 0.002222
Balanced Accuracy    0.498876
> |
```

RandomForest Prediction Results

```
> rf.predict <- predict(rf.train, test)
> confusionMatrix(rf.predict, test$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8
         3   0   0   0   0   0   0
         4   0   0   0   0   0   0
         5   3  12 155  63   4   0
         6   0   5  37 107  33   2
         7   0   0   0   8  17   3
         8   0   0   0   0   1   0

Overall Statistics

               Accuracy : 0.62
                 95% CI : (0.5734, 0.665)
    No Information Rate : 0.4267
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3729

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity          0.000000  0.00000   0.8073   0.6011  0.30909 0.000000
Specificity          1.000000  1.00000   0.6822   0.7169  0.97215 0.997753
Pos Pred Value            NaN      NaN   0.6540   0.5815  0.60714 0.000000
Neg Pred Value       0.993333  0.96222   0.8263   0.7331  0.90995 0.988864
Prevalence           0.006667  0.03778   0.4267   0.3956  0.12222 0.011111
Detection Rate       0.000000  0.00000   0.3444   0.2378  0.03778 0.000000
Detection Prevalence 0.000000  0.00000   0.5267   0.4089  0.06222 0.002222
Balanced Accuracy    0.500000  0.50000   0.7447   0.6590  0.64062 0.498876
> |
```

The RandomForest method performed better than K-nearest neighbours with an accuracy of 62 and Kappa of 37. All of the models did an insufficient job at identifying red wines of two lowest and two highest classes.

Now let's move on to white wine for analysis to understand if these models perform better than red wine.

Data analysis and visualisation (White Wine)

I will be using the same libraries for this analysis and the steps will be followed, as with the red wine.

Let's begin by downloading the dataset

```
#Importing the data of the white wine for the second step of analysis

white = read.csv("D:/Harvard/winequality-white.csv", sep = ";", header = T)
```

Next step would be to find more information about the dataset

```
> #viewing structure of the data and summary
> str(white)
'data.frame':   4898 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 .
 $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
 $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
>
```

```
> #check the names of the variables present in the data.
> colnames(white)
 [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
 [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
 [7] "total.sulfur.dioxide" "density"              "pH"
[10] "sulphates"            "alcohol"              "quality"
>
```

```
> #summary to better understand the dataset
> summary(white)
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900
 1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700   1st Qu.:0.03600
 Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200   Median :0.04300
 Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391   Mean   :0.04577
 3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900   3rd Qu.:0.05000
 Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800   Max.   :0.34600
 free.sulfur.dioxide total.sulfur.dioxide    density             pH          sulphates
 Min.   :  2.00      Min.   :  9.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200
 1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100
 Median : 34.00      Median :134.0        Median :0.9937   Median :3.180   Median :0.4700
 Mean   : 35.31      Mean   :138.4        Mean   :0.9940   Mean   :3.188   Mean   :0.4898
 3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500
 Max.   :289.00      Max.   :440.0        Max.   :1.0390   Max.   :3.820   Max.   :1.0800
    alcohol         quality
 Min.   : 8.00   Min.   :3.000
 1st Qu.: 9.50   1st Qu.:5.000
 Median :10.40   Median :6.000
 Mean   :10.51   Mean   :5.878
 3rd Qu.:11.40   3rd Qu.:6.000
 Max.   :14.20   Max.   :9.000
>
```

Removing all the duplicate rows

```
> #Removing the Duplicate Rows
> white <- white[!duplicated(white), ]
> dim(white)
[1] 3961   12
> str(white)
'data.frame':   3961 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 6.2 8.1 8.1 8.6 7.9 6.6 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.32 0.22 0.27 0.23 0.18 0.16 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.16 0.43 0.41 0.4 0.37 0.4 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 7 1.5 1.45 4.2 1.2 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.045 0.044 0.033 0.035 0.04 0.044 ...
 $ free.sulfur.dioxide : num  45 14 30 47 30 28 11 17 16 48 ...
 $ total.sulfur.dioxide: num  170 132 97 186 136 129 63 109 75 143 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.995 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.18 3.22 2.99 3.14 3.18 3.54 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.47 0.45 0.56 0.53 0.63 0.52 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.6 11 12 9.7 10.8 12.4 ...
 $ quality             : int  6 6 6 6 6 6 5 5 5 7 ...
```

From 4898 rows to 3961, approximately 937 rows have been removed due to duplication. This can impact final results.

Checking for any NA in the dataset.

```
> #Check for NA in dataset
> sum(is.na(white))
[1] 0
> |
```

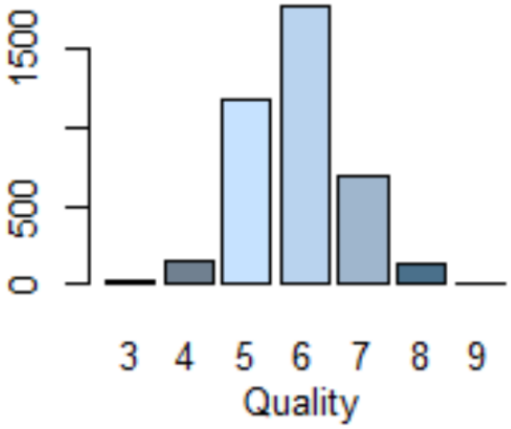
Let's move on to the response count of various classes of white wine. Similar to what was done for the red wine.

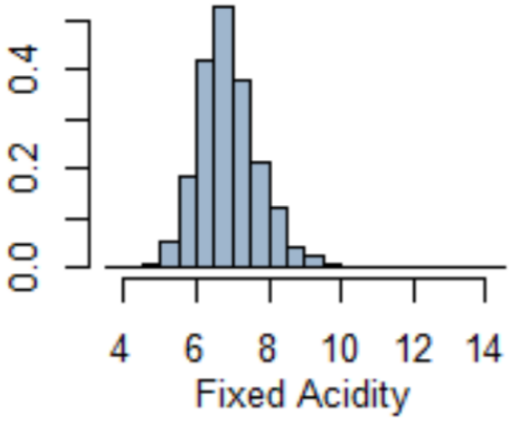
```
> #Response count in the dataset
> table(white$quality)

   3    4    5    6    7    8    9
  20  153 1175 1788  689  131    5
> |
```

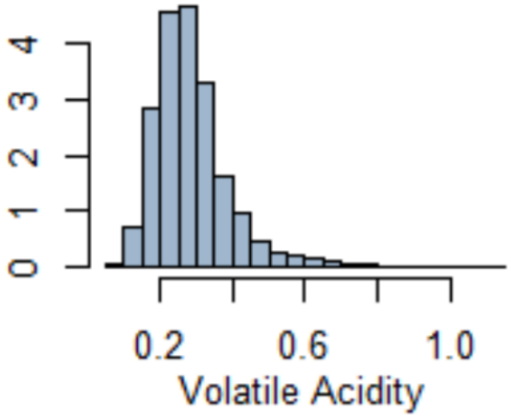
As observed the class 5 to 7 of white wine have many responses and responses are skewed, similar to the red wine.

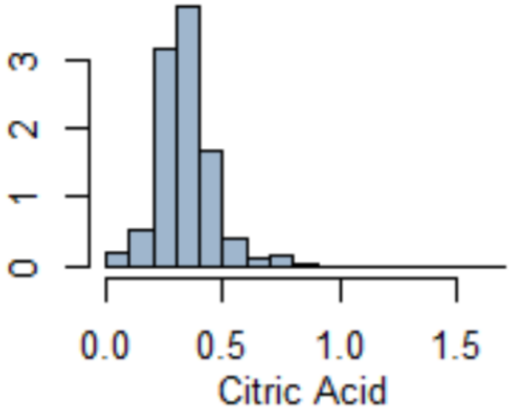
The next step is to further analyse the data to gather more information. We start by plotting the histogram of the dataset

```
> #Plotting the histograms using hist() for the data.
>
> #plot comparison of "QUALITY", "FIXED ACIDITY", "VOLATILE ACIDITY",
 "CITRIC ACID"
>
> attach(white)
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) + 0.1)
> barplot((table(quality)), col=c("slateblue4", "slategray", "slategra
y1", "slategray2", "slategray3", "skyblue4"))
> mtext("Quality", side=1, outer=F, line=2, cex=0.8)
>
>
> truehist(fixed.acidity, h = 0.5, col="slategray3")
> mtext("Fixed Acidity", side=1, outer=F, line=2, cex=0.8)
>
> truehist(volatile.acidity, h = 0.05, col="slategray3")
> mtext("Volatile Acidity", side=1, outer=F, line=2, cex=0.8)
>
> truehist(citric.acid, h = 0.1, col="slategray3")
> mtext("Citric Acid", side=1, outer=F, line=2, cex=0.8)
> |
```

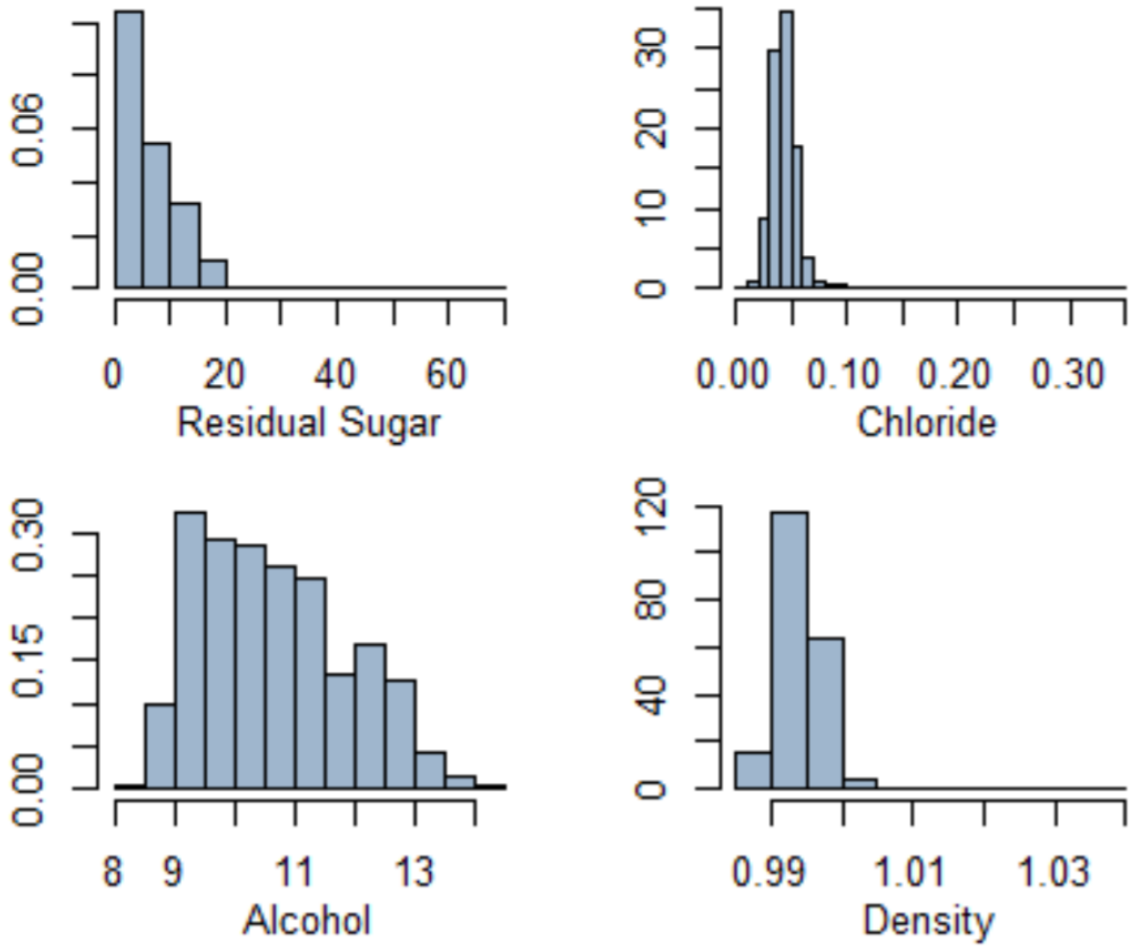

0
500
1500
3
4
5
6
7
8
9
Quality


0.0
0.2
0.4
4
6
8
10
12
14
Fixed Acidity

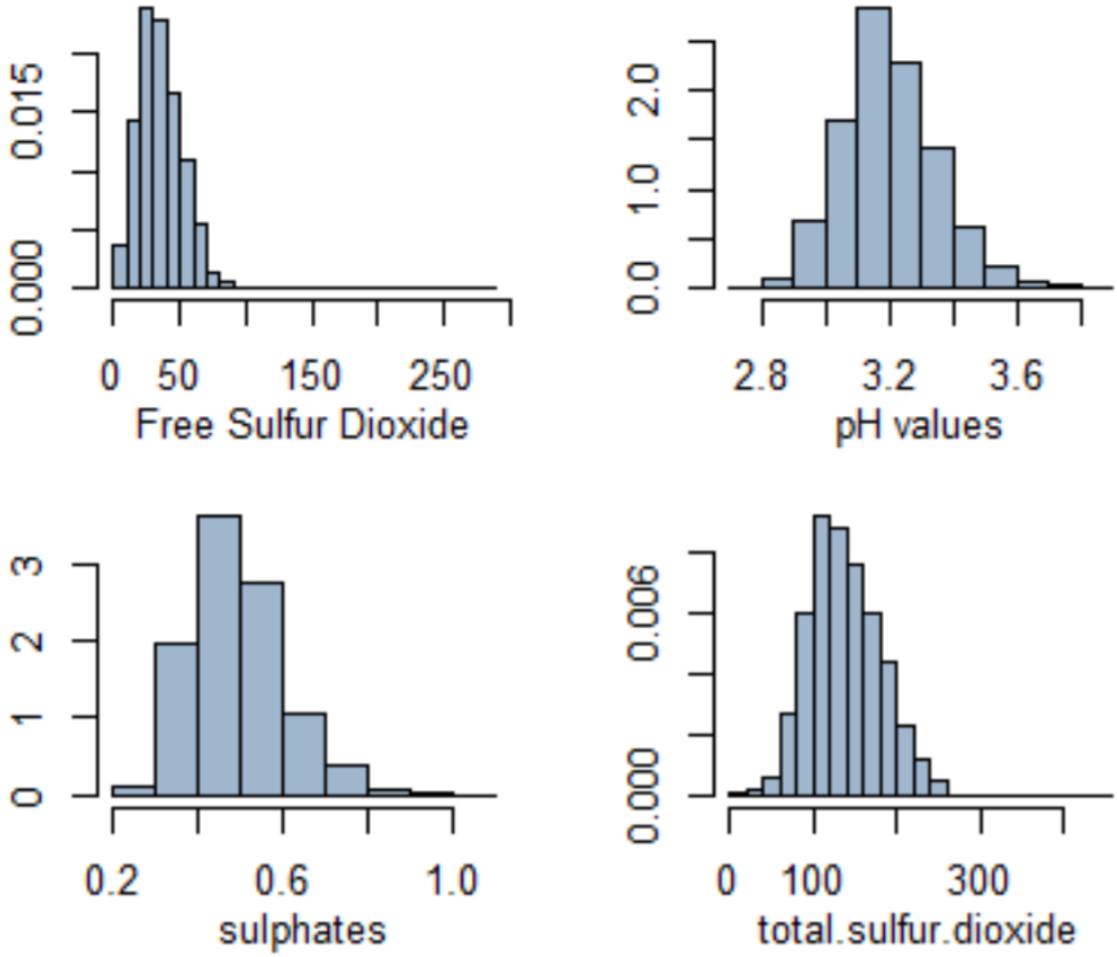

0
1
2
3
4
0.2
0.6
1.0
Volatile Acidity


0
1
2
3
0.0
0.5
1.0
1.5
Citric Acid

```
> #PLOT COMPARISON OF "RESIDUAL SUGAR", "CHLORIDE", "ALCHOL", "DENSITY"
>
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) + 0.1)
>
> truehist(residual.sugar, h = 5, col="slategray3")
> mtext("Residual Sugar", side=1, outer=F, line=2, cex=0.8)
>
> truehist(chlorides, h = 0.01, col="slategray3")
> mtext("Chloride", side=1, outer=F, line=2, cex=0.8)
>
 >
 > truehist(alcohol, h = 0.5, col="slategray3")
 > mtext("Alcohol", side=1, outer=F, line=2, cex=0.8)
 >
 >
 > truehist(density, h = 0.005, col="slategray3")
 > mtext("Density", side=1, outer=F, line=2, cex=0.8)
 >
```



```
> #PLOT COMPARISON OF "FREE SULFUR DIOXIDE", "pH VALUES", "SULP
HATES", "TOTAL.SULFUR.DIOXIDE"
>
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) +
 0.1)
>
> truehist(free.sulfur.dioxide, h = 10, col="slategray3")
> mtext("Free Sulfur Dioxide", side=1, outer=F, line=2, cex=0.
8)
>
> truehist(pH, h = 0.1, col="slategray3")
> mtext("pH values", side=1, outer=F, line=2, cex=0.8)
>
>
> truehist(sulphates, h = 0.1, col="slategray3")
> mtext("sulphates", side=1, outer=F, line=2, cex=0.8)
>
>
> truehist(total.sulfur.dioxide, h = 20, col="slategray3")
> mtext("total.sulfur.dioxide", side=1, outer=F, line=2, cex=0.
8)
>
```



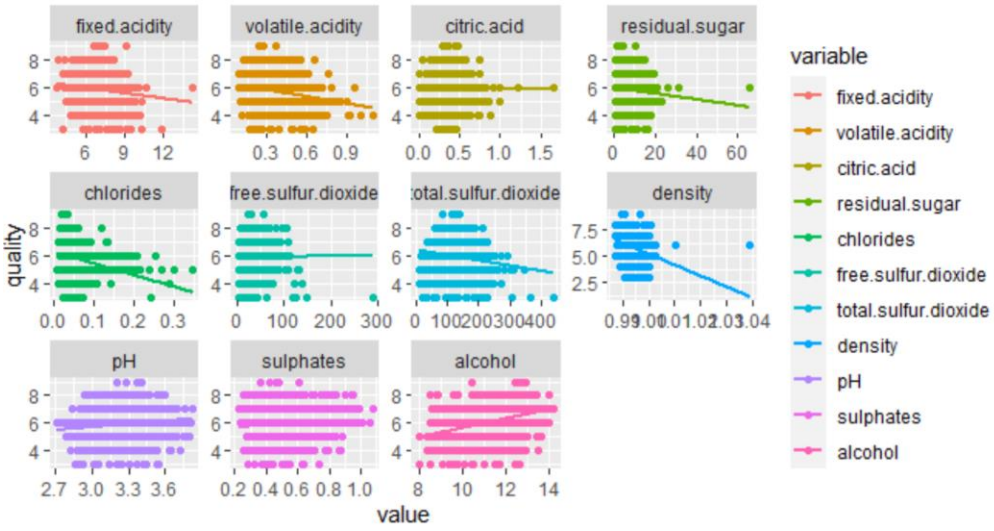
The first thing that stands out in the plots is the presence of outliers for the majority of the predictor variables. The UCI wine dataset was cleaned prior to its posting, which would eliminate errors. However, the outliers are interesting.

Let's focus on relationship between attributes and quality.

```
> #Relationship between each Attribute and Quality (Rating)
>
> white_wine <- melt(white, "quality")
> ggplot(white_wine, aes(value, quality, color = variable)) +
+   geom_point() +
+   geom_smooth(aes(value,quality, colour=variable), method=lm, se=FALSE)+
+   facet_wrap(.~variable, scales = "free")
`geom_smooth()` using formula 'y ~ x'
>
```

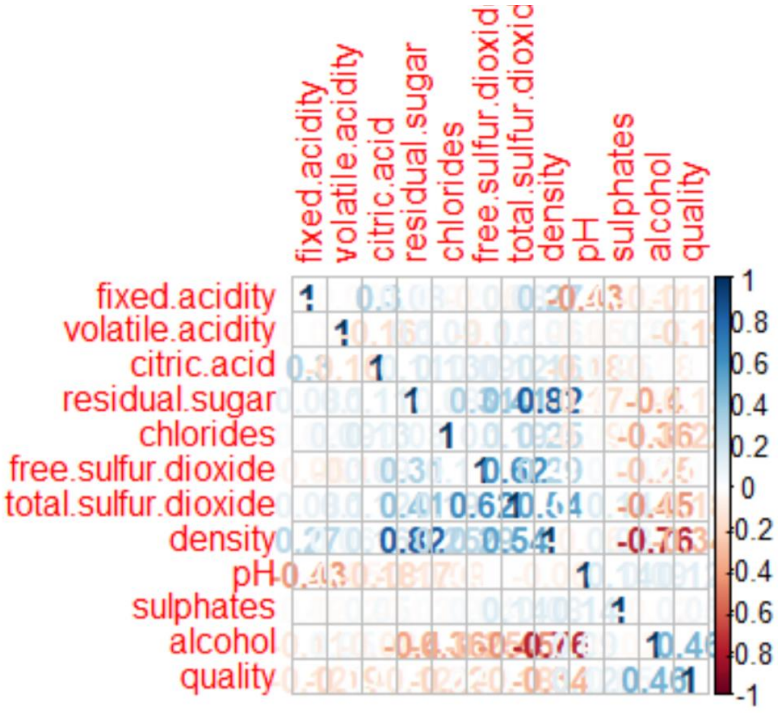
Alcohol and sulphates seem to have a positive correlation with quality.

While volatile acidity, chlorides and total sulfur dioxide seem to have negative relationship with quality.



I will now pinpoint the collinearity between the attributes using the following code.

```
> #Collinearity between Attributes
> par(mfrow = c(1,1))
> cor.white <- cor(white)
> corrplot(cor.white, method = 'number')
>
```



Weak relationships between quality and citric.acid, free.sulfur.dioxide and sulphates can be seen in the above correlation plot.

Density has a 0.82 correlation with residual.sugar and a -0.76 correlation with alcohol. This is a greater concern if we were planning on training regression and/or linear models. However, after studying the data, I've decided that non-linear classification models will be more appropriate than regression. Ive concluded that the relationship between the response and predictor variables is more complex than what a regression and/or linear model can capture.

Data Split (White Wine)

I will firstly convert the quality response of white wine into factors, before splitting the data into training and test dataset.

It will follow the same steps as that of the red wine.

```
> white$quality <- as.factor(white$quality)
> white$quality_as_factor = NULL
>
```

```
> #setting the seed value to 1
> set.seed(1)
>
> #splitting the dataset into train and test (2/3rd for train remaining for test)
>
> inTrain <- createDataPartition(white$quality, p = 2/3, list = F)
> train <- white[inTrain,]
> test <- white[-inTrain,]
>

> #Structure and summary after split
> str(train)
'data.frame':   2644 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 6.2 8.1 8.1 8.6 7.9 6.6 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.32 0.22 0.27 0.23 0.18 0.16 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.16 0.43 0.41 0.4 0.37 0.4 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 7 1.5 1.45 4.2 1.2 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.045 0.044 0.033 0.035 0.04 0.044
 ...
 $ free.sulfur.dioxide : num  45 14 30 47 30 28 11 17 16 48 ...
 $ total.sulfur.dioxide: num  170 132 97 186 136 129 63 109 75 143 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.995 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.18 3.22 2.99 3.14 3.18 3.54 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.47 0.45 0.56 0.53 0.63 0.52 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.6 11 12 9.7 10.8 12.4 ...
 $ quality             : Factor w/ 7 levels "3","4","5","6",..: 4 4 4 4 4 4 3 3 3 5 ...
> str(test)
'data.frame':   1317 obs. of  12 variables:
 $ fixed.acidity       : num  6.6 6.2 7.4 6.4 7 7.4 8.5 7 6.6 7.4 ...
 $ volatile.acidity    : num  0.17 0.66 0.34 0.31 0.28 0.27 0.24 0.31 0.24 0.18 ...
 $ citric.acid         : num  0.38 0.48 0.42 0.38 0.39 0.48 0.39 0.26 0.27 0.31 ...
 $ residual.sugar      : num  1.5 1.2 1.1 2.9 8.7 1.1 10.4 7.4 1.4 1.4 ...
 $ chlorides           : num  0.032 0.029 0.033 0.038 0.051 0.047 0.044 0.069 0.057 0.05
8 ...
 $ free.sulfur.dioxide : num  28 29 17 19 32 17 20 28 33 38 ...
 $ total.sulfur.dioxide: num  112 75 171 102 141 132 142 160 152 167 ...
 $ density             : num  0.991 0.989 0.992 0.991 0.996 ...
 $ pH                  : num  3.25 3.33 3.12 3.17 3.38 3.19 3.2 3.13 3.22 3.16 ...
 $ sulphates           : num  0.55 0.39 0.53 0.35 0.53 0.49 0.53 0.46 0.56 0.53 ...
 $ alcohol             : num  11.4 12.8 11.3 11 10.5 11.6 10 9.8 9.5 10 ...
 $ quality             : Factor w/ 7 levels "3","4","5","6",..: 5 6 4 5 4 4 4 4 4 5 ...
>
```

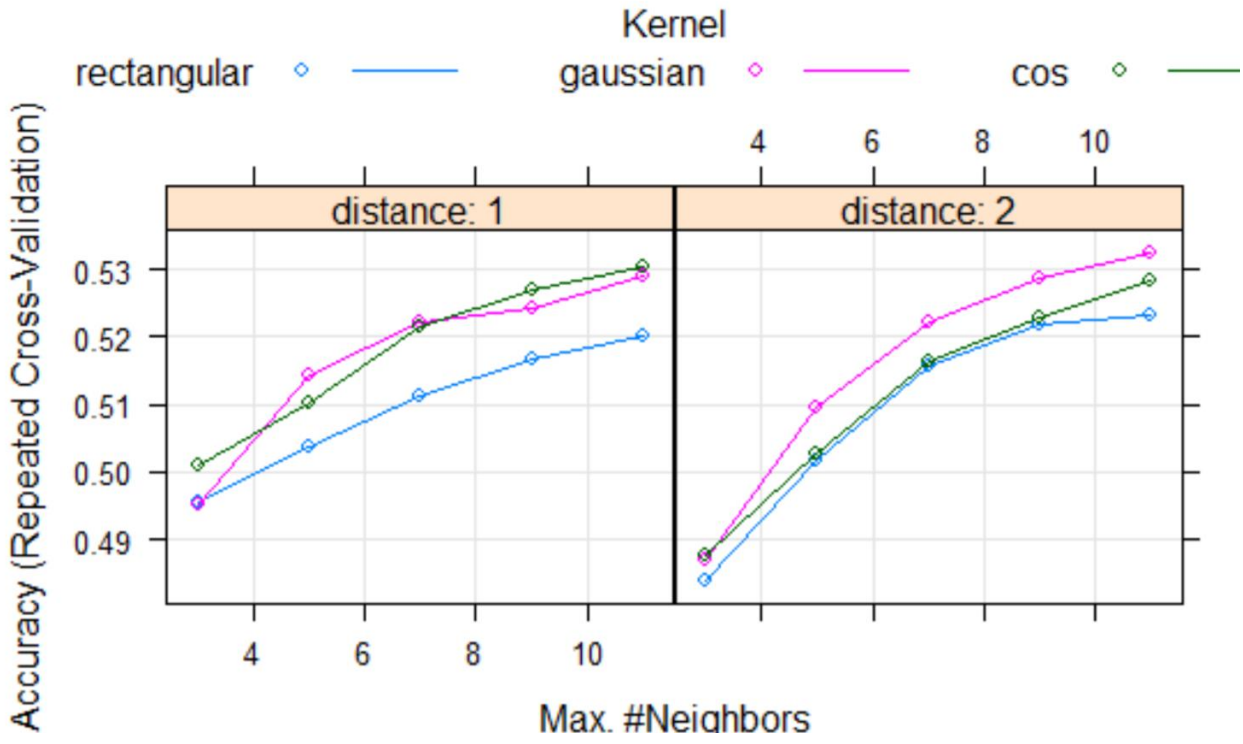
Modeling and Evaluation:

I will use two methods to train dataset and observe the results in this part, I will be using k-nearest neighbours and random forest to model the data.

K-nearest neighbours:

Just like with red wine, we will use the same codes to process this dataset.

```
> ##K-nearest neighbour model
>
> t.ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)
> knn.grid <- expand.grid(kmax = c(3, 5, 7 ,9, 11), distance = c(1, 2),
+                         kernel = c("rectangular", "gaussian", "cos"))
> knn.train <- train(quality ~ ., data = train, method = "knn",
+                    trControl = t.ctrl, tuneGrid = knn.grid,
+                    preProcess = c("center", "scale"))
>
> save(knn.train, file = "knn.train.rda")
>
>
> # saveRDS(model, "model.rds")
> # my_model <- readRDS("model.rds")
> # This lets you to choose a new name for the object (you don't need to remember the na
me you used when you saved it)
>
> plot(knn.train)
>
```




```
> load("knn.train.rda")
> knn.train$bestTune
   kmax distance   kernel
29   11        2 gaussian
>
```

Once again gaussian kernel outperformed others just like in red wine. The best value for k is 11.

Moving on to the next model before predicting the results of these models.

RandomForest

For the random forest, only the mtry hyperparameter is available for tuning. Mtry values of one to 11 (1-11) will be passed into the train function's tuneGrid argument. Mtry is the number of variables randomly sampled as candidates at each split.

```
> #Random Forest
>
> rf.grid <- expand.grid(mtry = 1:11)
> rf.train <- train(quality ~ ., data = train, method = "rf",
+                   trControl = t.ctrl, tuneGrid = rf.grid,
+                   preProcess = c("center", "scale"))
>
> save(rf.train, file = "rf.train.rda")
> plot(rf.train)
>
```

```
> load("rf.train.rda")
> rf.train$bestTune
  mtry
2    2
>
```

We get mtry value of 2. It is not univariate as that the case of red wine.

Best model prediction

The models were trained using the accuracy metric. Models will be evaluated in the same fashion as that of the red wine.

K-Nearest Neighbour prediction Results:

```
> #Predicted accurancy
> kknn.predict <- predict(kknn.train, test)
> confusionMatrix(kknn.predict, test$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8   9
         3   0   0   0   0   0   0   0
         4   0   6   1   0   0   0   0
         5   2  30 217 132   6   0   0
         6   4  11 154 380 131  20   0
         7   0   4  19  81  89  20   1
         8   0   0   0   3   3   3   0
         9   0   0   0   0   0   0   0

Overall Statistics

               Accuracy : 0.5277
                 95% CI : (0.5003, 0.555)
    No Information Rate : 0.4525
    P-Value [Acc > NIR] : 2.696e-08

                  Kappa : 0.2661

 Mcnemar's Test P-Value : NA
```

```
Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8  Class: 9
Sensitivity          0.000000 0.117647   0.5550   0.6376  0.38865 0.069767 0.0000000
Specificity          1.000000 0.999210   0.8164   0.5562  0.88511 0.995290 1.0000000
Pos Pred Value            NaN 0.857143   0.5607   0.5429  0.41589 0.333333       NaN
Neg Pred Value       0.995444 0.965649   0.8129   0.6499  0.87307 0.969419 0.9992407
Prevalence           0.004556 0.038724   0.2969   0.4525  0.17388 0.032650 0.0007593
Detection Rate       0.000000 0.004556   0.1648   0.2885  0.06758 0.002278 0.0000000
Detection Prevalence 0.000000 0.005315   0.2938   0.5315  0.16249 0.006834 0.0000000
Balanced Accuracy    0.500000 0.558429   0.6857   0.5969  0.63688 0.532529 0.5000000
> |
```

RandomForest prediction Results:

```
> #Accurancy prediction
> rf.predict <- predict(rf.train, test)
> confusionMatrix(rf.predict, test$quality)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8   9
         3   0   0   0   0   0   0   0
         4   0   9   3   1   0   0   0
         5   2  27 224 118   6   0   0
         6   4  15 162 432 155  23   0
         7   0   0   2  44  68  18   1
         8   0   0   0   1   0   2   0
         9   0   0   0   0   0   0   0


 Overall Statistics

                Accuracy : 0.5581
                  95% CI : (0.5308, 0.5851)
     No Information Rate : 0.4525
     P-Value [Acc > NIR] : 1.045e-14

                   Kappa : 0.2932

  Mcnemar's Test P-Value : NA


 Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8  Class: 9
Sensitivity          0.000000 0.176471   0.5729   0.7265  0.29694 0.046512 0.0000000
Specificity          1.000000 0.996840   0.8359   0.5021  0.94026 0.999215 1.0000000
Pos Pred Value            NaN 0.692308   0.5957   0.5467  0.51128 0.666667       NaN
Neg Pred Value       0.995444 0.967791   0.8225   0.6895  0.86402 0.968798 0.9992407
Prevalence           0.004556 0.038724   0.2969   0.4525  0.17388 0.032650 0.0007593
Detection Rate       0.000000 0.006834   0.1701   0.3288  0.05163 0.001519 0.0000000
Detection Prevalence 0.000000 0.009871   0.2855   0.6014  0.10099 0.002278 0.0000000
Balanced Accuracy    0.500000 0.586656   0.7044   0.6143  0.61860 0.522863 0.5000000
> |
```

Random Forest model were the better performing model, of the two. Random Forest returned an accuracy of 55.8 and a Kappa of 29.3. Both models did a poor job at identifying white wines of the 2 lowest and 2 highest classes.

Conclusion:

In this report, I used two datasets of UCI (vinho verde wine) to predict the quality. The quality is based on physicochemical attributes that was initially explained.

The report started with data cleaning and preparation by importing the dataset on to R. After which dataset was analysed separately for red and white wine. This was explained in the data visualisation section wherein the impact of these parameters on quality of wine was analysed.

Moving towards the modeling phase, two models were used in this analysis for both wine types and their accuracy was determined.

These model could only predict the accuracy of average quality wines and it couldn't predict accurately the highest and lowest quality wines. The accuracy for the white wine is more balanced across the classes. However, further work have to be done on the dataset to improve the predictions across low and high quality wine.