UNIVERSITÀ DEGLI STUDI DI MILANO

WINE QUALITY PREDICTION USING SUPERVISED LEARNING TECHNIQUES

Statistical Learning, Deep Learning and Artificial Intelligence

Sai Spandana Adivishnu (03180A)

A.A. 2022-2023

1 Abstract

This project focuses on the analysis and assessment of wine quality using a dataset comprising physicochemical properties and sensory scores. Through exploratory data analysis, feature engineering, and the implementation of machine learning algorithms (k-nearest neighbors and random forest), the study aims to understand the factors influencing wine quality and build predictive models.The random forest model exhibited superior performance compared to the k-NN model, highlighting its suitability for wine quality assessment. These findings could have significant implications for the wine industry, enabling producers to evaluate and enhance wine quality based on objective chemical attributes. Future research could explore additional algorithms and feature engineering techniques to further improve the predictive accuracy and interpretability of wine quality models.

Key Factors :- k-nearest neighbors (KNN) algorithm, Random forest

Contents

2 Introduction

Figure 1: Wine Quality

In this project, we focused on developing predictive models to assess the quality of wines based on their chemical characteristics. We used a dataset that included physicochemical attributes and sensory scores of different wines. Our goal was to uncover the relationships between these attributes and the perceived quality of the wines.

To achieve this, we implemented two machine learning algorithms: k-nearest neighbors (k-NN) and random forest. The k-NN algorithm predicts the quality of a wine by considering the quality of its nearest neighbors in the feature space. On the other hand, random forest constructs an ensemble of decision trees, utilizing random subsets of features at each split to make predictions.

Before training the models, we conducted exploratory data analysis to gain insights into the dataset's distribution and identify any correlations between the attributes. We also performed preprocessing tasks such as data cleaning and normalization to ensure data quality.

The results of our analysis showed that both the k-NN and random forest models were effective in predicting wine quality. However, the random forest model exhibited slightly better performance, achieving higher overall accuracy and robustness compared to the k-NN model.

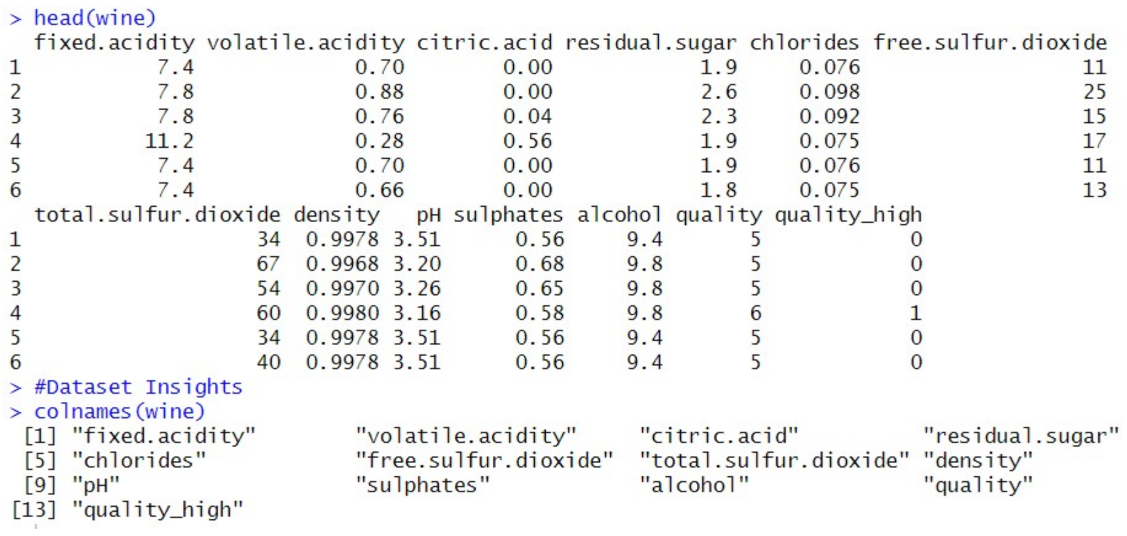
3 Data Preposessing

This data is collected from the kaggle. This data consists of 1599 observations and 13 variables with the information about physicochemical attributes and sensory scores of different wines such as

- fixed.acidity
- volatile.acidity
- citric.acid
- residual.sugar
- chlorides
- free.sulfur.dioxide
- total.sulfur.dioxide
- density
- pH
- sulphates
- alcohol
- quality [Which is our Response variable or Target]

3.1 Data Insights

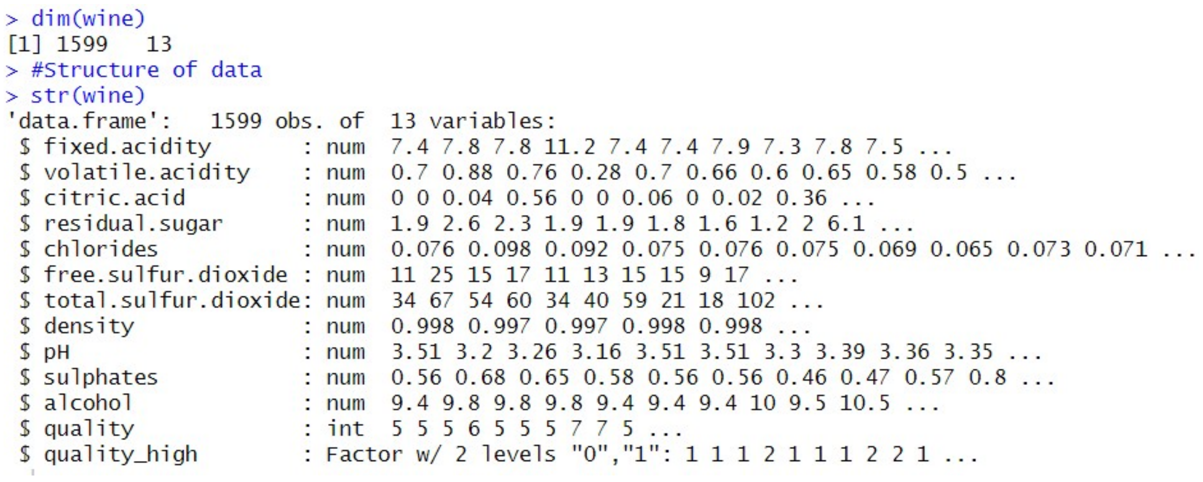
Here is the imported wine dataset.



```
> head(wine)
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide
1           7.4             0.70        0.00            1.9     0.076                  11
2           7.8             0.88        0.00            2.6     0.098                  25
3           7.8             0.76        0.04            2.3     0.092                  15
4          11.2             0.28        0.56            1.9     0.075                  17
5           7.4             0.70        0.00            1.9     0.076                  11
6           7.4             0.66        0.00            1.8     0.075                  13
  total.sulfur.dioxide density   pH sulphates alcohol quality quality_high
1                   34  0.9978 3.51      0.56     9.4       5            0
2                   67  0.9968 3.20      0.68     9.8       5            0
3                   54  0.9970 3.26      0.65     9.8       5            0
4                   60  0.9980 3.16      0.58     9.8       6            1
5                   34  0.9978 3.51      0.56     9.4       5            0
6                   40  0.9978 3.51      0.56     9.4       5            0
> #Dataset Insights
> colnames(wine)
 [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"          "residual.sugar"
 [5] "chlorides"            "free.sulfur.dioxide"  "total.sulfur.dioxide" "density"
 [9] "pH"                   "sulphates"            "alcohol"              "quality"
[13] "quality_high"
```

Figure 2: Data Insights

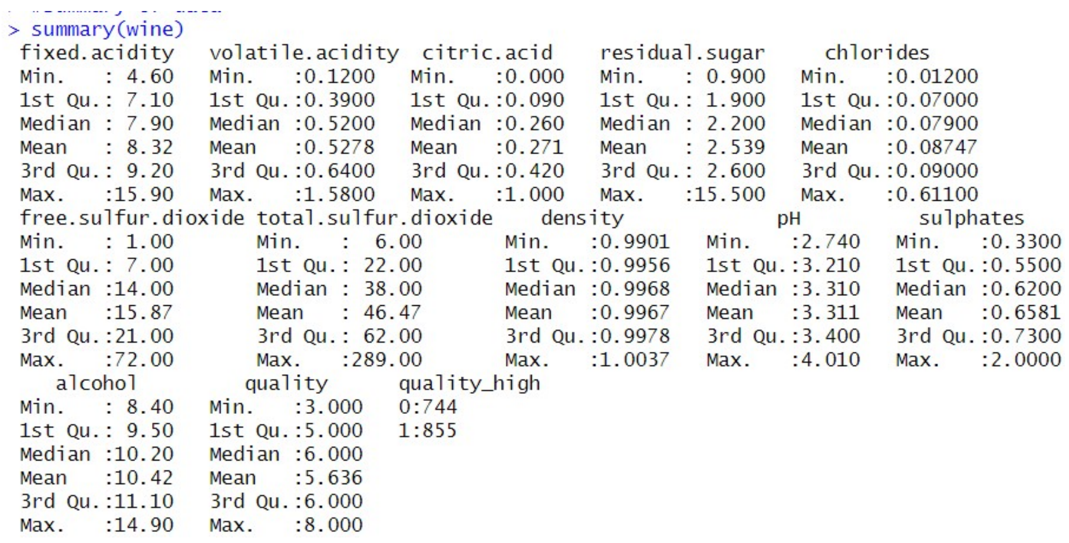
The target and predictor variables are well separated, as mentioned in the section under "About the Dataset." The quality is the target variable, and the other variables will be utilized as predictors.



```
> dim(wine)
[1] 1599   13
> #Structure of data
> str(wine)
'data.frame':   1599 obs. of  13 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
 $ quality_high        : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 2 2 1 ...
```

Figure 3: Structure of Data

Overall, it appears to be good; no predictors appear to have incorrect data types, but since we intend to utilize classification models, we must convert the quality column to a categorical format. We either factorize the column as is or categorize the numbers into certain categories/groups because the peek() method shows us that the quality spans from 0 to 10 and appears to only employ integers.



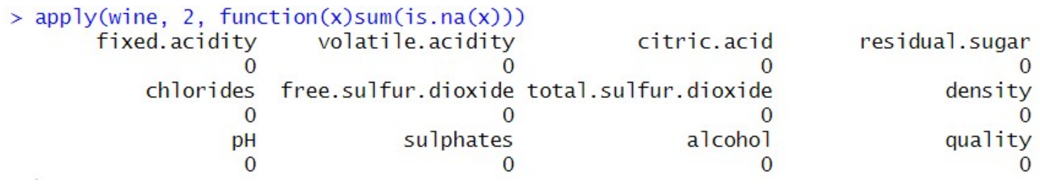
```
> summary(wine)
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar     chlorides      
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900   Min.   :0.01200  
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900   1st Qu.:0.07000  
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200   Median :0.07900  
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539   Mean   :0.08747  
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600   3rd Qu.:0.09000  
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500   Max.   :0.61100  
 free.sulfur.dioxide total.sulfur.dioxide    density             pH          sulphates     
 Min.   : 1.00       Min.   :  6.00       Min.   :0.9901   Min.   :2.740   Min.   :0.3300  
 1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500  
 Median :14.00       Median : 38.00       Median :0.9968   Median :3.310   Median :0.6200  
 Mean   :15.87       Mean   : 46.47       Mean   :0.9967   Mean   :3.311   Mean   :0.6581  
 3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300  
 Max.   :72.00       Max.   :289.00       Max.   :1.0037   Max.   :4.010   Max.   :2.0000  
    alcohol         quality      quality_high
 Min.   : 8.40   Min.   :3.000   0:744       
 1st Qu.: 9.50   1st Qu.:5.000   1:855       
 Median :10.20   Median :6.000               
 Mean   :10.42   Mean   :5.636               
 3rd Qu.:11.10   3rd Qu.:6.000               
 Max.   :14.90   Max.   :8.000               
```

Figure 4: Summary of Data

The output provides summary of key variables related to wine quality. It includes physico-chemical attributes such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. The statistics for each attribute, including minimum, maximum, median, mean, and quartile values, are presented. The quality variable represents the sensory evaluation score, ranging from 3 to 8, with a mean value of 5.636. The presence of the variable "quality_high" indicates whether a wine is considered high quality (1) or not (0). This summary provides a concise overview of the dataset, offering insights into the range and distribution of the

3.1.1 Checking Missing Values

There is no null values in any predictors. It seems safe to go to further step.

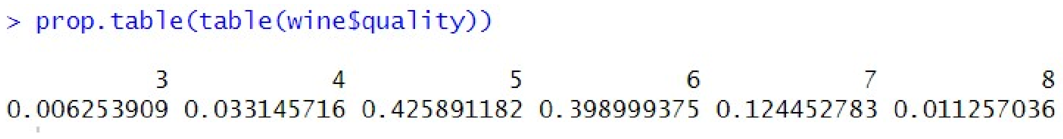


```
> apply(wine, 2, function(x)sum(is.na(x)))
       fixed.acidity     volatile.acidity          citric.acid       residual.sugar
                   0                    0                    0                    0
           chlorides  free.sulfur.dioxide total.sulfur.dioxide              density
                   0                    0                    0                    0
                  pH            sulphates              alcohol              quality
                   0                    0                    0                    0
```

Figure 5: Checking for Null Values

3.2 Adjusting Target Variables for classification models

I want to check for the proportion of the target variable.



```
> prop.table(table(wine$quality))

          3           4           5           6           7           8 
0.006253909 0.033145716 0.425891182 0.398999375 0.124452783 0.011257036 
```

Figure 6: Adjusting Target

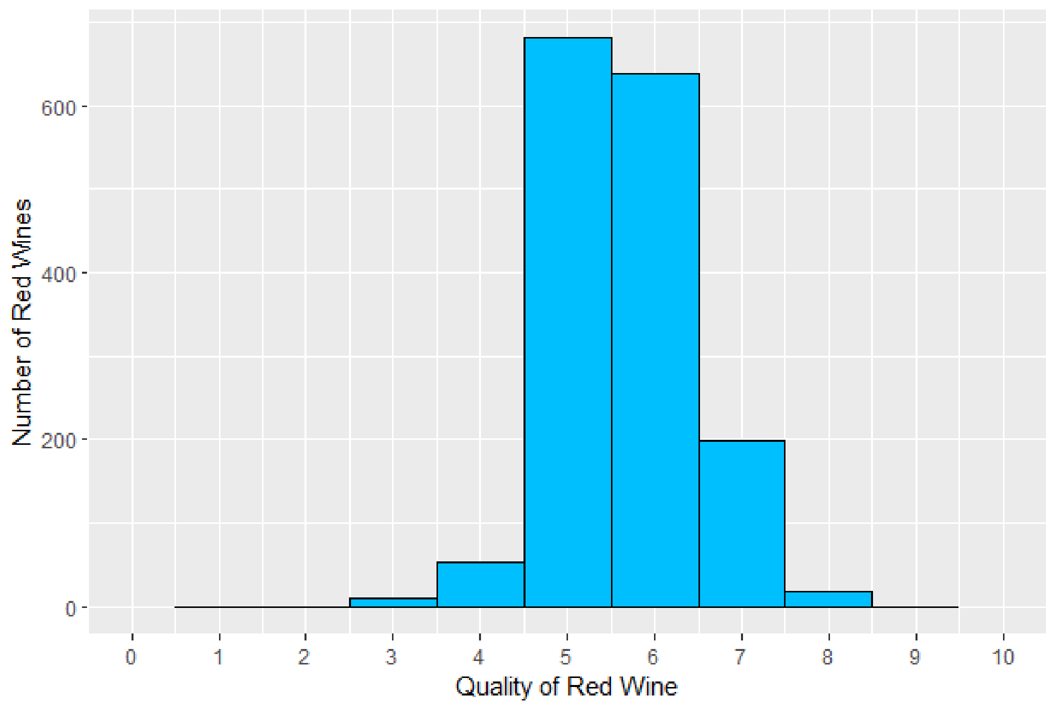


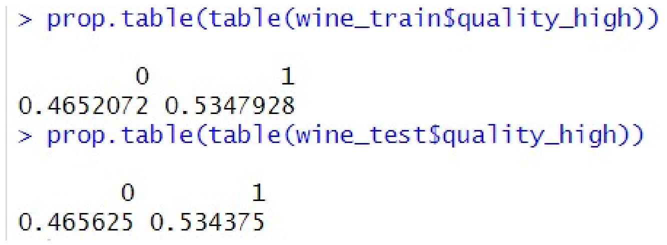
Figure 7: Visualization of Target Variable

If we intend to use classification machine learning models, representing the wine quality ratings as different numbers may not be appropriate, as the numbers indicate an order of ratings. While simple regression models can predict numeric targets, for classification models, we need to categorize the target variable. Additionally, it is important to address class imbalance to ensure balanced representation.

Initially, the plan was to categorize ratings of at least 7 as high quality (or 1) and the rest as low quality (or 0). However, considering the distribution of the data with a focus on values 5 and 6, it would be better to split them into separate classes. Hence, the new approach is to categorize ratings 6 and above as high quality and the rest as low quality. A new column called 'quality_high' will be created to reflect this categorization.

3.3 Train-Test Splitting

Next, we will separate the data into train and test ones, with the default of 80:20 split, using strata of quality_high to keep the balanced proportion. To make the random sampling stay.

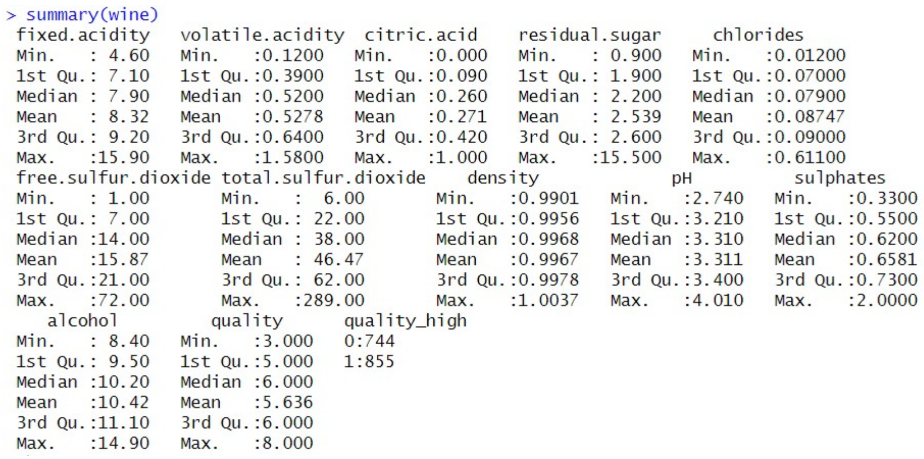


```
> prop.table(table(wine_train$quality_high))

        0         1 
0.4652072 0.5347928 
> prop.table(table(wine_test$quality_high))

       0        1 
0.465625 0.534375 
```

Figure 8: Train and Test Split

4 Exploratory Data Analysis and Data Visualization

4.1 Overall Summary Statistics



```
> summary(wine)
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar     chlorides      
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900   Min.   :0.01200  
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900   1st Qu.:0.07000  
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200   Median :0.07900  
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539   Mean   :0.08747  
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600   3rd Qu.:0.09000  
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500   Max.   :0.61100  
 free.sulfur.dioxide total.sulfur.dioxide    density             pH          sulphates     
 Min.   : 1.00       Min.   :  6.00       Min.   :0.9901   Min.   :2.740   Min.   :0.3300  
 1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500  
 Median :14.00       Median : 38.00       Median :0.9968   Median :3.310   Median :0.6200  
 Mean   :15.87       Mean   : 46.47       Mean   :0.9967   Mean   :3.311   Mean   :0.6581  
 3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300  
 Max.   :72.00       Max.   :289.00       Max.   :1.0037   Max.   :4.010   Max.   :2.0000  
    alcohol          quality      quality_high
 Min.   : 8.40   Min.   :3.000   0:744       
 1st Qu.: 9.50   1st Qu.:5.000   1:855       
 Median :10.20   Median :6.000               
 Mean   :10.42   Mean   :5.636               
 3rd Qu.:11.10   3rd Qu.:6.000               
 Max.   :14.90   Max.   :8.000               
```

Figure 9: Train and Test Split

KEYFINDINGS:

Some of the predictors, such as 'fixed.acidity', 'total.sulfur.dioxide', 'free.sulfur.dioxide', and 'sulphates', appear to have outliers. This is evident from the maximum values, which are significantly larger than the mean, median, or 3rd quartile. On the other hand, 'density' and 'pH' exhibit more favorable distributions, as the maximum values are relatively closer to the median and 3rd quartile. Although addressing the outliers is important, for the current analysis, we will proceed with the data as is and observe how it impacts our models.

4.2 Data Visualization

I will divide the predictors based on scales that are comparable because they appear to have different scales. I'll leave out density and pH because they seem to have a fairly normal scale and might fill our plot.

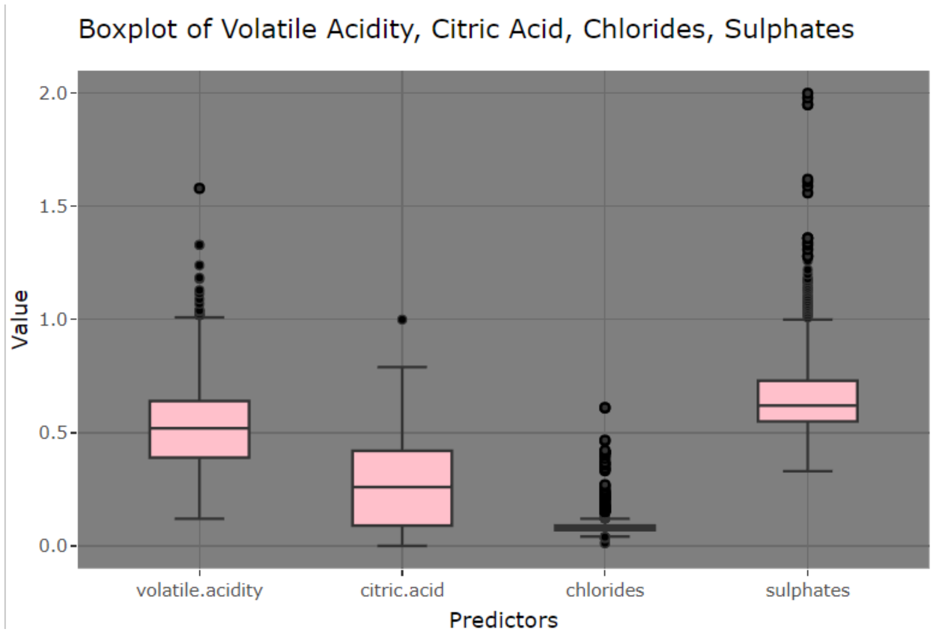


Figure 10: BoxPlot of Volatile Acidity, Citiric Acid, Chlorides and Sulphates

As you can see in the above plot the Critic acid demonstrate relatively normal and well-distributed patterns compared to others.

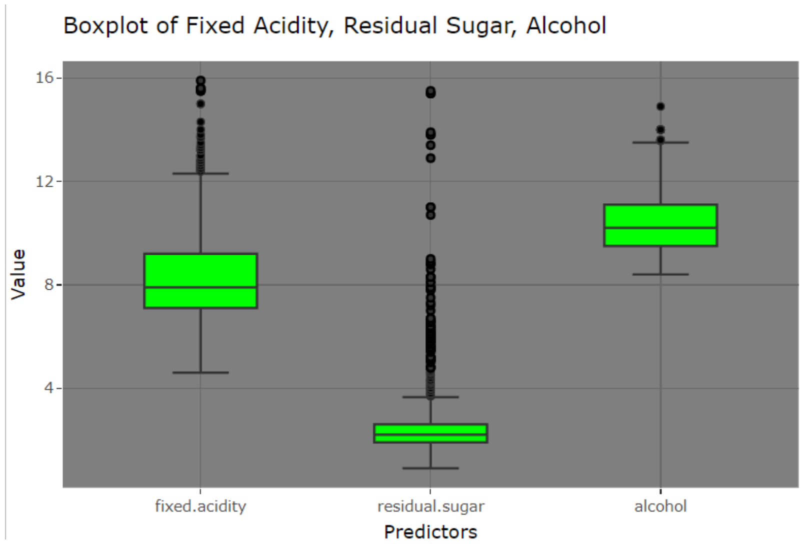


Figure 11: Boxplot of Fixed Acidity, Residual Sugar, Alcohol

In this plot we can identify Alcohol seemes to be normal and well distributed.

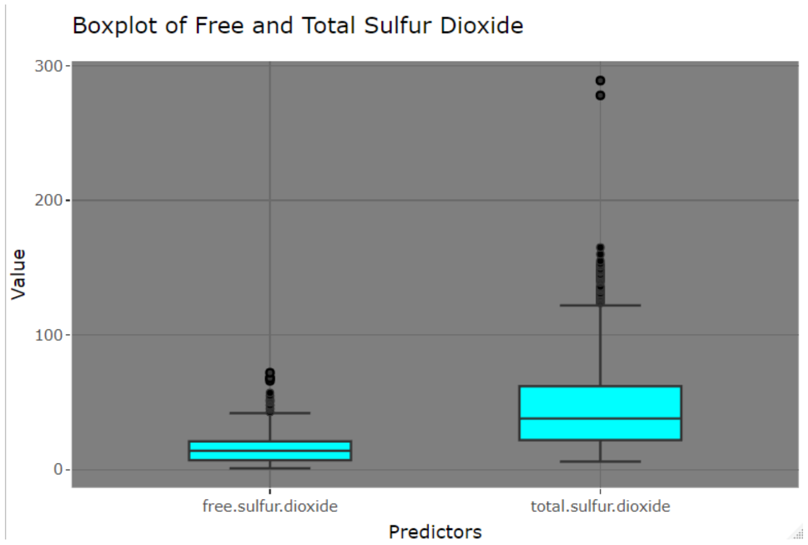


Figure 12: Boxplot of Free and Total Sulfur Dioxide

I coundnt find any variable exhibit a significant number of outliers.

KEYFINDINGS:

However, two predictors, namely 'alcohol' and 'citric.acid', demonstrate relatively normal and well-distributed patterns. These variables have a smaller number of outliers, and their boxplots show the median positioned near the center. This indicates that 'alcohol' and 'citric.acid' may have more reliable and consistent distributions compared to other predictors in the dataset.

4.3 Correlation Between Predictors

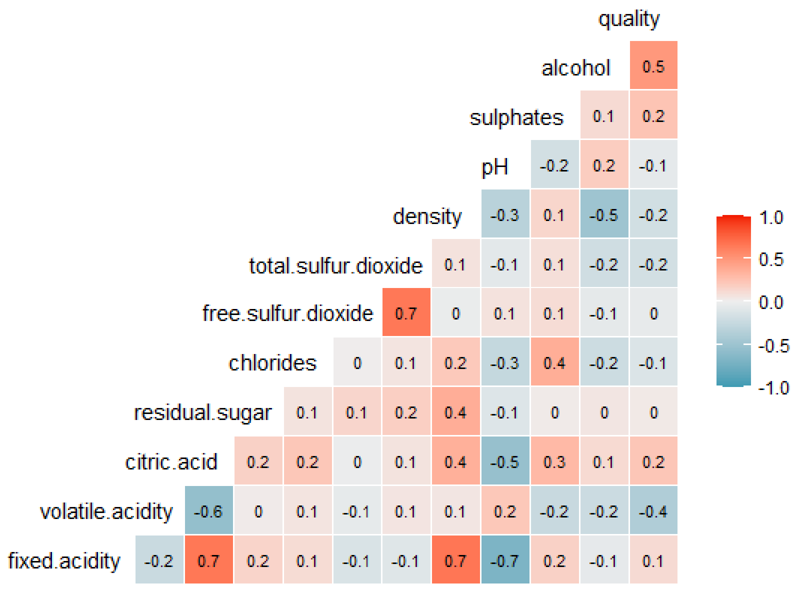


Figure 13: Correlation Plot

KEY FINDINGS:

1. The target variable, 'quality', appears to have a positive correlation with 'alcohol' and a negative correlation with 'volatile.acidity'.
2. Conversely, 'quality' does not seem to have a significant correlation with 'free.sulfur.dioxide' and 'residual.sugar'.
3. There are potential strong correlations between certain variables that have similar names. For example, 'free.sulfur.dioxide' and 'total.sulfur.dioxide' may exhibit a strong correlation, as well as 'fixed.acidity' and 'volatile.acidity'. These strong correlations suggest the possibility of multicollinearity, which can adversely affect some of our models.

4.4 Scatterplots for Strong Correlated Variables

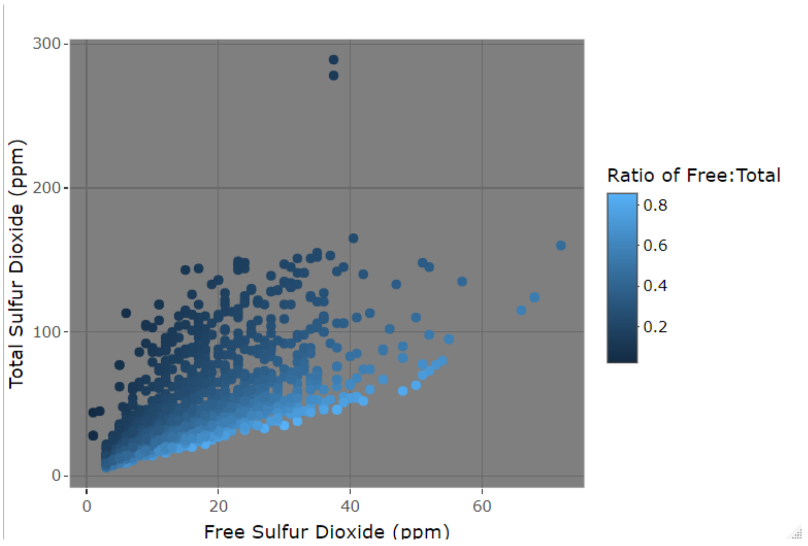


Figure 14: Correlation between Free Sulfur Dioxide and Total Sulfur Dioxide

KEY FINDINGS:

While a trend line can be observed from the bottom left to the top right, indicating a positive correlation between the two variables, the scatter plot also reveals a significant amount of randomness in the data point distribution. As a result, I don't believe that these two variables require any special feature engineering to capture their individual linearity.

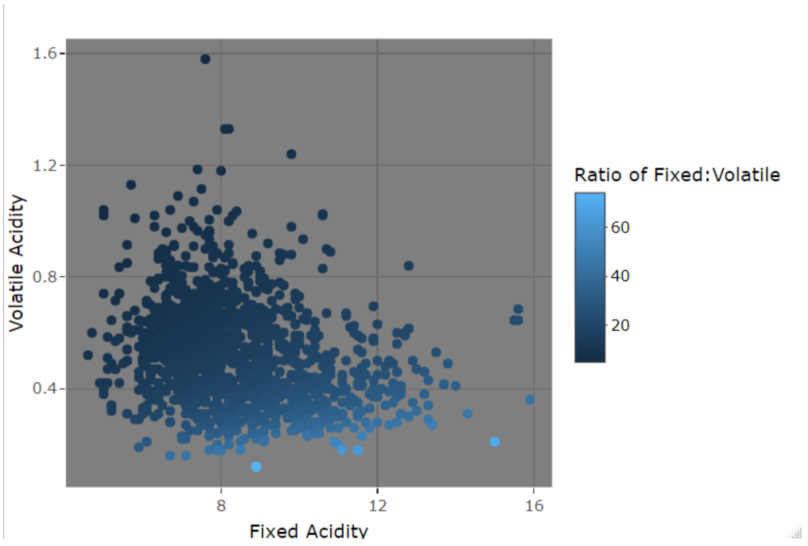


Figure 15: Correlation between Fixed and Volatile Acidity

KEY FINDINGS:

Despite the strong correlation observed in the scatter plot, there is no clear linear pattern that suggests one variable can accurately predict the other. In other words, there is no evident relationship indicating a clear linearity between the two variables. Hence, using this combination of two variables in our analysis can be done safely, as there is no indication of one variable being a strong predictor of the other.

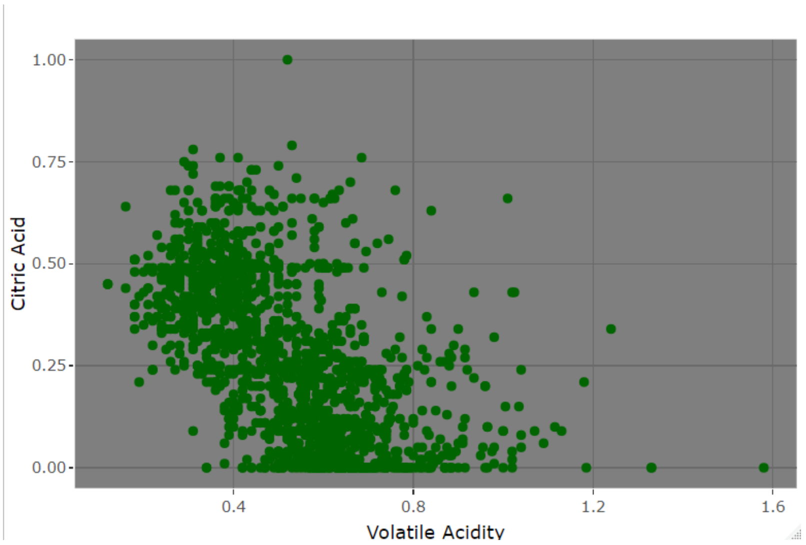


Figure 16: Correlation between Volatile Acidity and Citric Acid

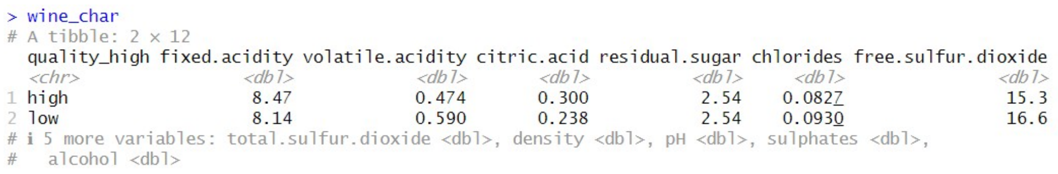
KEY FINDINGS:

Although a trend line can be roughly approximated from the top-left to the bottom-right of the chart, it is important to note that the data points exhibit a considerable amount of random dispersion. Therefore, the linearity isn't really seen in these two variables.

4.5 Characteristics of Top-Rated vs Lowest-Rated Red Wines

To identify the characteristics that distinguish our top-rated red wine from the rest, we will compare them based on specific qualities that stand out. In order to represent the entire group for each characteristic (predictor), using the mean value would be more appropriate than other measures like the median. By considering the mean, even outliers that potentially accentuate a particular characteristic can be taken into account, making it more distinctive compared to others.

To accomplish this, we will divide the data into two groups: red wines rated 6-8, and red wines rated below 6. Subsequently, we will summarize each predictor using the mean() function to gain insights into the average values of the predictors for each group.



```
> wine_char
# A tibble: 2 × 12
  quality_high fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide
  <chr>                <dbl>            <dbl>       <dbl>          <dbl>     <dbl>               <dbl>
1 high                  8.47            0.474       0.300           2.54    0.0827                15.3
2 low                   8.14            0.590       0.238           2.54    0.0930                16.6
# ℹ 5 more variables: total.sulfur.dioxide <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
#   alcohol <dbl>
```

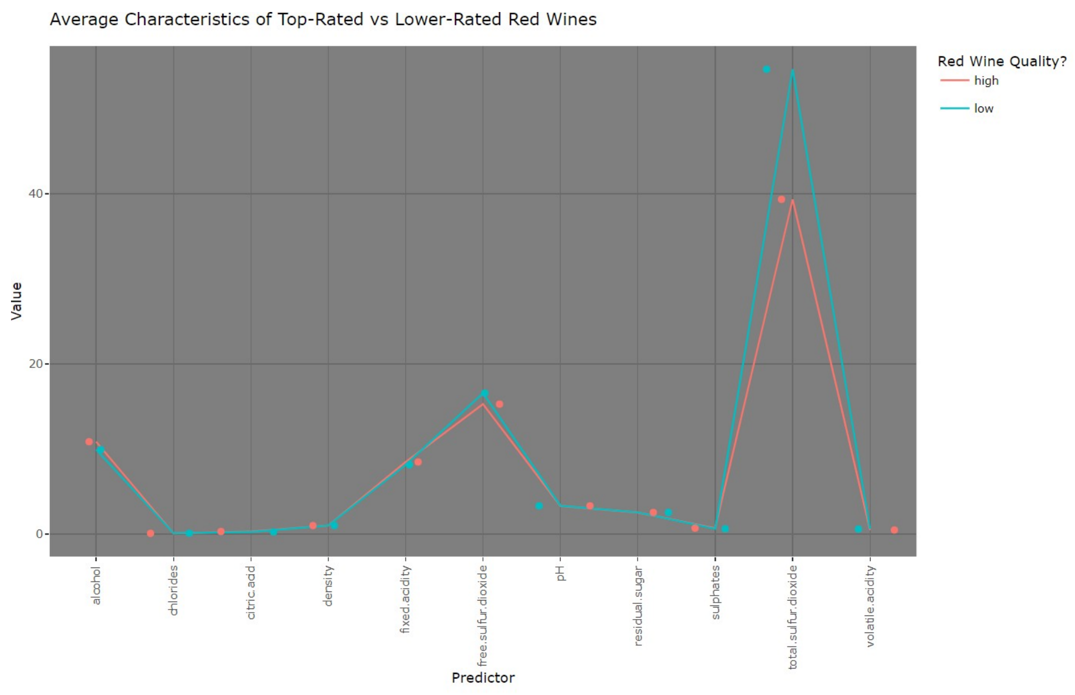


Figure 17: Average Characteristics of Top-Rated vs Lower-Rated Red Wines

KEY FINDINGS:

When comparing the average values of predictors between top-rated red wines and lower-rated red wines, several observations can be made:

- Top-rated red wines tend to have a significantly lower amount of Total Sulfur Dioxide compared to lower-rated red wines, as depicted in the visualization of predictor averages.
- While other predictors show only slight differences, they can still be helpful in classifying higher quality red wines:
    1. Alcohol, Citric Acid, Sulphates, and Fixed Acidity tend to be slightly higher in higher quality red wines.
    2. On the other hand, Chlorides, Free Sulfur Dioxide, and Volatile Acidity tend to be slightly lower in higher quality red wines.
- Density, pH, and Residual Sugar appear to be relatively consistent across all red wines, regardless of their quality rating.

5 Model Building

5.1 K-NN model

The k-nearest neighbors (K-NN) algorithm is a machine learning model used for classification and regression tasks. It operates based on the principle that similar data points tend to have similar labels or values. In the case of classification, K-NN assigns a label to a new data point by considering the labels of its nearest neighbors. The number of neighbors to consider, represented by the parameter k, is determined by the user.

The algorithm calculates the distance between the new data point and each existing data point in the training set, using a chosen distance metric such as Euclidean distance. The k nearest neighbors are then determined, and the majority class among them is assigned as the predicted label for the new data point. In regression, K-NN predicts the value of the target variable by averaging the values of the k nearest neighbors. K-NN is a simple yet effective algorithm, but its performance can be influenced by the choice of k and the distance metric. It is a non-parametric model, meaning it does not make explicit assumptions about the underlying data distribution.

In the k-nearest neighbors (K-NN) algorithm, the distance between data points is a crucial component in determining the nearest neighbors. There are various distance metrics that can be used to measure the similarity or dissimilarity between two data points. Here are some commonly used distance metrics in K-NN:

Distance functions

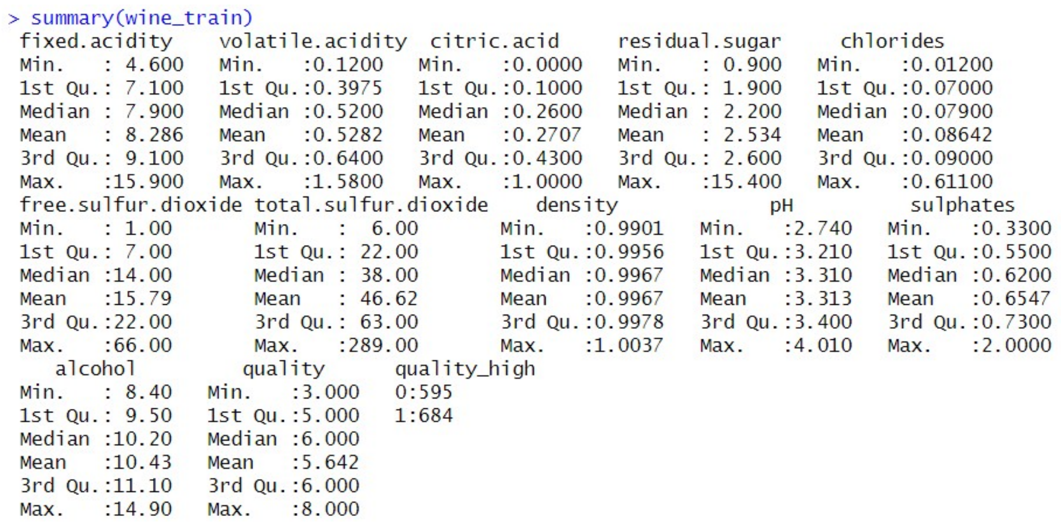
Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

5.1.1 Data Preprocessing

Let's check the dataset summary and check the range of the data (minimum-maximum values of each predictors).



```
> summary(wine_train)
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides      
 Min.   : 4.600   Min.   :0.1200   Min.   :0.0000   Min.   : 0.900   Min.   :0.01200  
 1st Qu.: 7.100   1st Qu.:0.3975   1st Qu.:0.1000   1st Qu.: 1.900   1st Qu.:0.07000  
 Median : 7.900   Median :0.5200   Median :0.2600   Median : 2.200   Median :0.07900  
 Mean   : 8.286   Mean   :0.5282   Mean   :0.2707   Mean   : 2.534   Mean   :0.08642  
 3rd Qu.: 9.100   3rd Qu.:0.6400   3rd Qu.:0.4300   3rd Qu.: 2.600   3rd Qu.:0.09000  
 Max.   :15.900   Max.   :1.5800   Max.   :1.0000   Max.   :15.400   Max.   :0.61100  
 free.sulfur.dioxide total.sulfur.dioxide    density             pH          sulphates     
 Min.   : 1.00       Min.   :  6.00       Min.   :0.9901   Min.   :2.740   Min.   :0.3300  
 1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500  
 Median :14.00       Median : 38.00       Median :0.9967   Median :3.310   Median :0.6200  
 Mean   :15.79       Mean   : 46.62       Mean   :0.9967   Mean   :3.313   Mean   :0.6547  
 3rd Qu.:22.00       3rd Qu.: 63.00       3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300  
 Max.   :66.00       Max.   :289.00       Max.   :1.0037   Max.   :4.010   Max.   :2.0000  
    alcohol          quality      quality_high
 Min.   : 8.40   Min.   :3.000   0:595       
 1st Qu.: 9.50   1st Qu.:5.000   1:684       
 Median :10.20   Median :6.000               
 Mean   :10.43   Mean   :5.642               
 3rd Qu.:11.10   3rd Qu.:6.000               
 Max.   :14.90   Max.   :8.000               
```

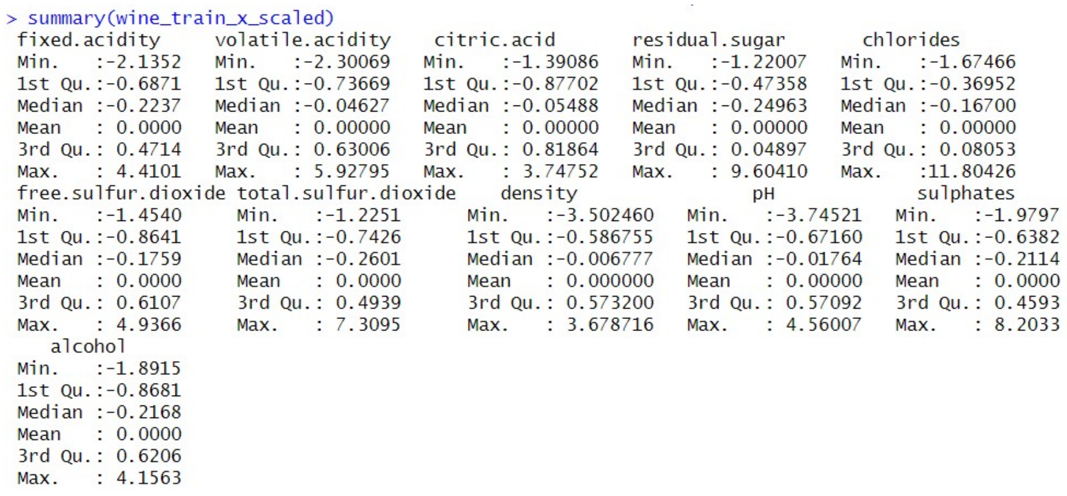
Figure 18: Summary of Data

KEY FINDINGS:

To ensure fair and unbiased representation of each predictor in the k-nearest neighbors (K-NN) algorithm, it is important to address the issue of varying scales among the predictors. Since predictors with larger numbers can dominate the distance calculation, it is necessary to normalize the numerical columns and bring them to a similar scale. This involves excluding the quality column, which serves as the target predictor (y), and scaling the remaining predictors (x). By scaling the predictors, each variable will contribute equally to the distance calculation and prevent any bias towards predictors with larger values. This step is crucial for accurate and reliable predictions in the K-NN method.

5.1.2 Data Scaling

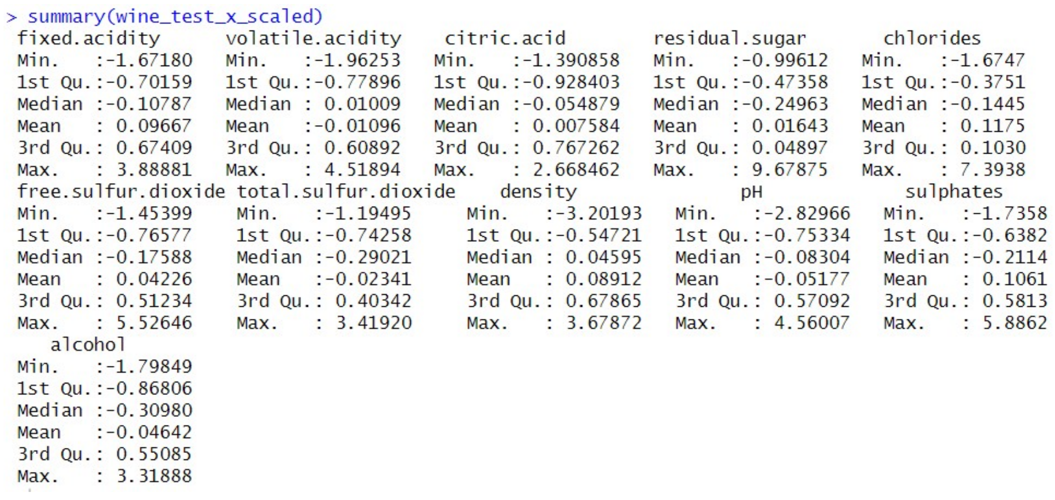
Before scaling the data points, it is important to separate the target variable (labels) from the predictors. In this case, we should exclude the quality column as it is no longer needed for evaluation in the k-nearest neighbors (K-NN) model or any classification models. The target variable will remain unchanged, while the predictors will be scaled to ensure consistency in their scales. This separation allows us to focus solely on the predictors and their relationships without considering the quality column.



```
> summary(wine_train_x_scaled)
 fixed.acidity      volatile.acidity    citric.acid        residual.sugar       chlorides
 Min.   :-2.1352   Min.   :-2.30069   Min.   :-1.39086   Min.   :-1.22007   Min.   :-1.67466
 1st Qu.:-0.6871   1st Qu.:-0.73669   1st Qu.:-0.87702   1st Qu.:-0.47358   1st Qu.:-0.36952
 Median :-0.2237   Median :-0.04627   Median :-0.05488   Median :-0.24963   Median :-0.16700
 Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000
 3rd Qu.: 0.4714   3rd Qu.: 0.63006   3rd Qu.: 0.81864   3rd Qu.: 0.04897   3rd Qu.: 0.08053
 Max.   : 4.4101   Max.   : 5.92795   Max.   : 3.74752   Max.   : 9.60410   Max.   :11.80426
 free.sulfur.dioxide total.sulfur.dioxide    density                 pH              sulphates
 Min.   :-1.4540     Min.   :-1.2251      Min.   :-3.502460   Min.   :-3.74521   Min.   :-1.9797
 1st Qu.:-0.8641     1st Qu.:-0.7426      1st Qu.:-0.586755   1st Qu.:-0.67160   1st Qu.:-0.6382
 Median :-0.1759     Median :-0.2601      Median :-0.006777   Median :-0.01764   Median :-0.2114
 Mean   : 0.0000     Mean   : 0.0000      Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000
 3rd Qu.: 0.6107     3rd Qu.: 0.4939      3rd Qu.: 0.573200   3rd Qu.: 0.57092   3rd Qu.: 0.4593
 Max.   : 4.9366     Max.   : 7.3095      Max.   : 3.678716   Max.   : 4.56007   Max.   : 8.2033
    alcohol
 Min.   :-1.8915
 1st Qu.:-0.8681
 Median :-0.2168
 Mean   : 0.0000
 3rd Qu.: 0.6206
 Max.   : 4.1563
```

Figure 19: Scaled data of Train set

For the testing ones, we need to use the parameters from the scaled training dataset because it is supposed to be unseen data that follows the same "rules" with the training data. This

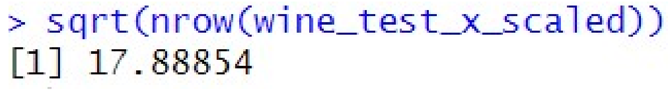


```
> summary(wine_test_x_scaled)
 fixed.acidity       volatile.acidity     citric.acid          residual.sugar       chlorides
 Min.   :-1.67180   Min.   :-1.96253   Min.   :-1.390858   Min.   :-0.99612   Min.   :-1.6747
 1st Qu.:-0.70159   1st Qu.:-0.77896   1st Qu.:-0.928403   1st Qu.:-0.47358   1st Qu.:-0.3751
 Median :-0.10787   Median : 0.01009   Median :-0.054879   Median :-0.24963   Median :-0.1445
 Mean   : 0.09667   Mean   :-0.01096   Mean   : 0.007584   Mean   : 0.01643   Mean   : 0.1175
 3rd Qu.: 0.67409   3rd Qu.: 0.60892   3rd Qu.: 0.767262   3rd Qu.: 0.04897   3rd Qu.: 0.1030
 Max.   : 3.88881   Max.   : 4.51894   Max.   : 2.668462   Max.   : 9.67875   Max.   : 7.3938
 free.sulfur.dioxide total.sulfur.dioxide    density                 pH               sulphates
 Min.   :-1.45399    Min.   :-1.19495     Min.   :-3.20193    Min.   :-2.82966   Min.   :-1.7358
 1st Qu.:-0.76577    1st Qu.:-0.74258     1st Qu.:-0.54721    1st Qu.:-0.75334   1st Qu.:-0.6382
 Median :-0.17588    Median :-0.29021     Median : 0.04595    Median :-0.08304   Median :-0.2114
 Mean   : 0.04226    Mean   :-0.02341     Mean   : 0.08912    Mean   :-0.05177   Mean   : 0.1061
 3rd Qu.: 0.51234    3rd Qu.: 0.40342     3rd Qu.: 0.67865    3rd Qu.: 0.57092   3rd Qu.: 0.5813
 Max.   : 5.52646    Max.   : 3.41920     Max.   : 3.67872    Max.   : 4.56007   Max.   : 5.8862
    alcohol
 Min.   :-1.79849
 1st Qu.:-0.86806
 Median :-0.30980
 Mean   :-0.04642
 3rd Qu.: 0.55085
 Max.   : 3.31888
```

Figure 20: Scaled data of Test set

scaling method involves transforming the data by subtracting the mean and dividing by the standard deviation, resulting in a distribution with a mean of 0 and a standard deviation of 1. By applying the scale() function, we can easily perform this z-score scaling on the predictors in the training dataset.

5.1.3 Finding Optimum K



```
> sqrt(nrow(wine_test_x_scaled))
[1] 17.88854
```

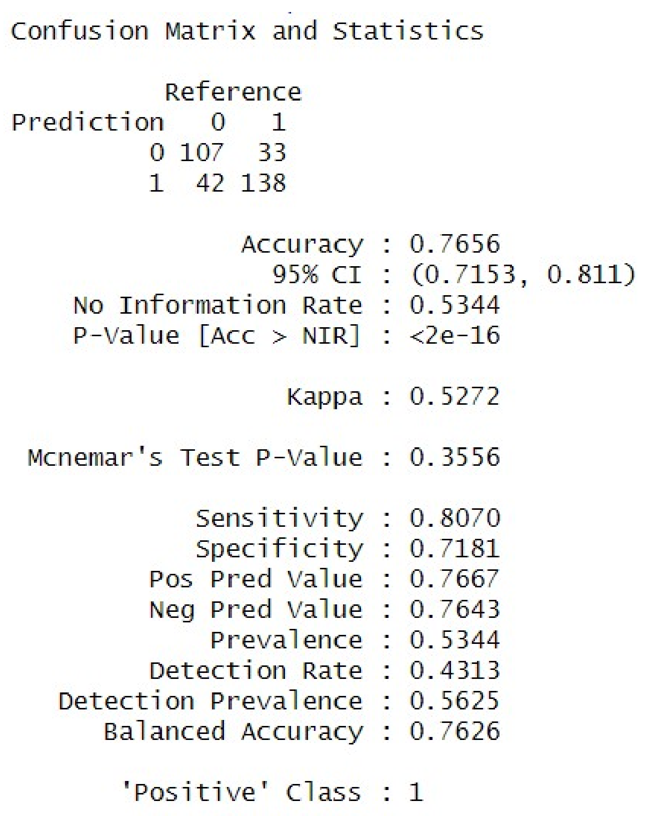
Figure 21: Finding of Optimum K

Since the target variable in our dataset has 2 categories or classes, it is recommended to choose an odd number for the value of k in the k-nearest neighbors (K-NN) algorithm. This helps to avoid any tie situations during the majority-voting process in the classification algorithm. Based on our analysis, the optimal value for k is approximately 17.88. Therefore, we will try different odd values of k, such as 15 and 19, in addition to k=17, when fine-tuning our model.

5.1.4 Model Fitting and Evaluation

Model Fitting k-NN is categorized as a "black-box" machine learning model, therefore we are not able to see its algorithm when it's working and cannot interpret its components. We can only use the result after the model has been fitted.

Model Evaluation A commonly used method to evaluate the performance of a classification model is by analyzing the confusion matrix. In our case, since the classes in the target variable are well-balanced (50:50), we have chosen to focus on accuracy and precision as evaluation metrics. Accuracy represents the overall correctness of the model's predictions, while precision emphasizes the ability of the model to correctly identify good quality wines based on their qualities. Given the preference for precise predictions of good quality wines, this evaluation approach provides a clear and interpretable assessment of the model's performance.

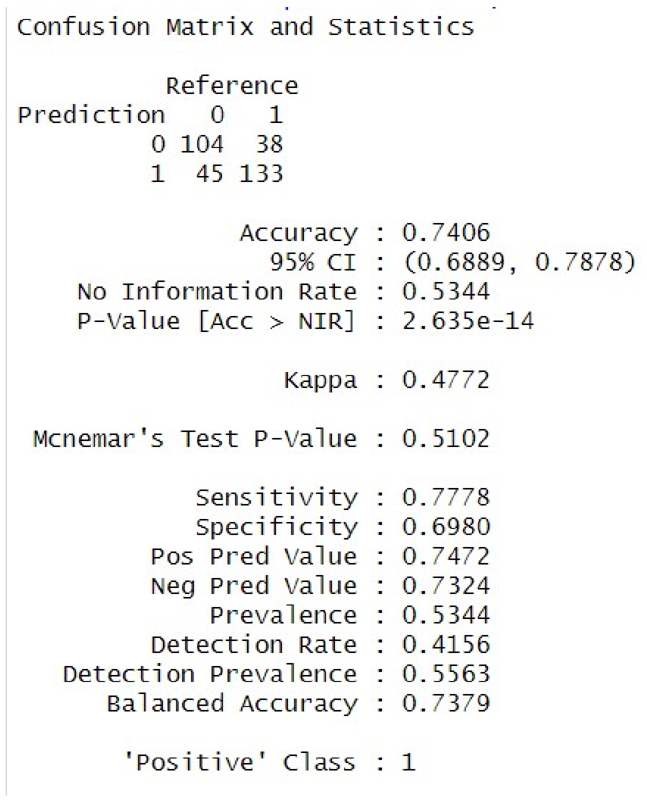


```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 107  33
         1  42 138

               Accuracy : 0.7656
                 95% CI : (0.7153, 0.811)
    No Information Rate : 0.5344
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5272

 Mcnemar's Test P-Value : 0.3556

            Sensitivity : 0.8070
            Specificity : 0.7181
         Pos Pred Value : 0.7667
         Neg Pred Value : 0.7643
             Prevalence : 0.5344
         Detection Rate : 0.4313
   Detection Prevalence : 0.5625
      Balanced Accuracy : 0.7626

       'Positive' Class : 1
```

Figure 22: Confusion matrix at K=17

Using K=17   KEY FINDINGS:

The Accuracy at K= 17 is 76.5% and Precision / Pos Pred Value: 76.60%.

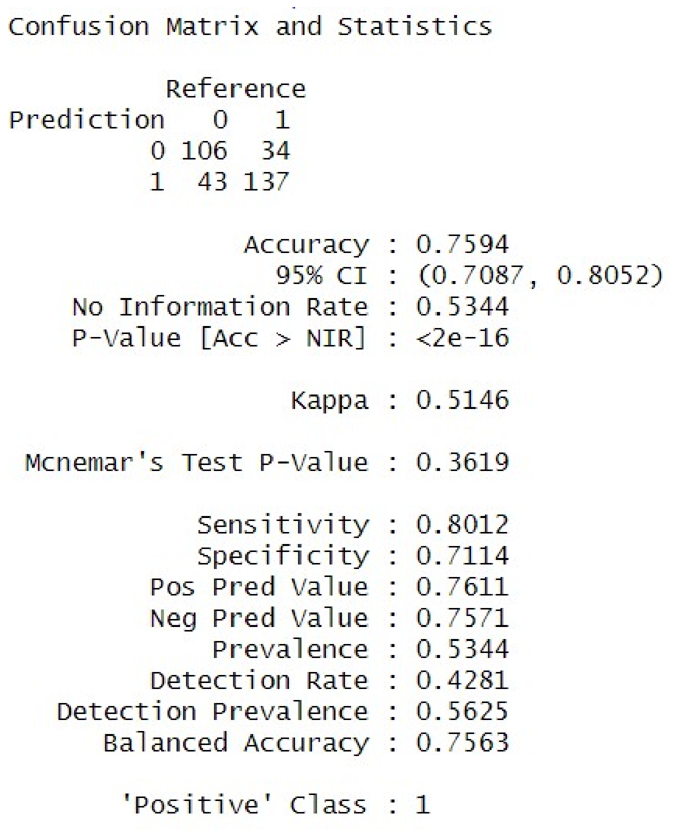


```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 104  38
         1  45 133

               Accuracy : 0.7406
                 95% CI : (0.6889, 0.7878)
    No Information Rate : 0.5344
    P-Value [Acc > NIR] : 2.635e-14

                  Kappa : 0.4772

 Mcnemar's Test P-Value : 0.5102

            Sensitivity : 0.7778
            Specificity : 0.6980
         Pos Pred Value : 0.7472
         Neg Pred Value : 0.7324
             Prevalence : 0.5344
         Detection Rate : 0.4156
   Detection Prevalence : 0.5563
      Balanced Accuracy : 0.7379

       'Positive' Class : 1
```

Figure 23: Confusion matrix at K=15

Using k=15 KEY FINDINGS:

The Accuracy at K= 15 is 74% and Precision / Pos Pred Value: 74.7%. I found it is Slightly lower compared to our original and optimum k.



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 106  34
         1  43 137

               Accuracy : 0.7594
                 95% CI : (0.7087, 0.8052)
    No Information Rate : 0.5344
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5146

 Mcnemar's Test P-Value : 0.3619

            Sensitivity : 0.8012
            Specificity : 0.7114
         Pos Pred Value : 0.7611
         Neg Pred Value : 0.7571
             Prevalence : 0.5344
         Detection Rate : 0.4281
   Detection Prevalence : 0.5625
      Balanced Accuracy : 0.7563

       'Positive' Class : 1
```

Figure 24: Confusion matrix at K=19

Using k=19 KEY FINDINGS:

The Accuracy at K= 19 is 75% and Precision / Pos Pred Value: 76.1%. Even lower compared to that of our original and optimum k.

5.1.5 Conclusions

KEY FINDINGS:

1. After evaluating multiple k values, we have determined that the optimal k for our k-NN model is 17, which is the closest odd number to the calculated optimum k. The accuracy of our model is 76.5%, indicating the overall correctness of its predictions.
2. Additionally, the precision of our model is 76.60%., highlighting its ability to accurately identify and classify good quality wines based on their qualities.

5.2 Random Forest

The Random Forest model is a powerful ensemble learning method that combines multiple decision trees to make predictions. It works by creating a multitude of decision trees on different subsets of the training data and then combining their predictions to determine the final output. This approach helps to reduce overfitting and improve the overall accuracy and robustness of the model.

In simpler terms, this model is basically making multiple Decision Tree models with random predictors/variables to use, then either use the majority voting system for classification cases, or mean of the targets for regression ones.

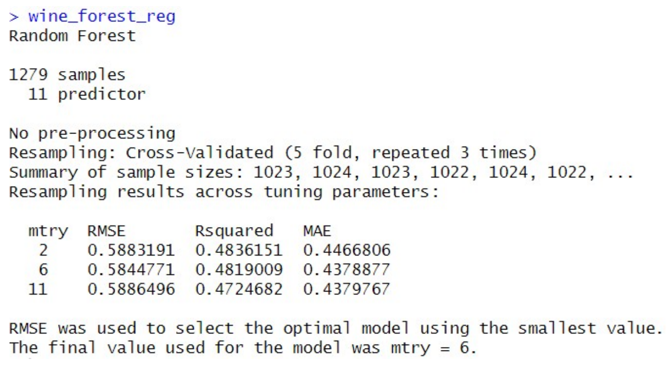
This model is called to have one of the best and accurate predictions amongst most models. The general downside though, to do multiple modeling, this method will need resources like processors, RAM, and most likely, time, to compute the predicted result.

5.2.1 Model Fitting

Since it is said that Random Forest can be utilized for both regression and classification models, let's try both of them. I will attach the code in the chunk below, but they will be commented, as sometimes, model fitting for Random Forest could take hours. Although it would depend on how many folds and repetition will the cross-validation of the data be done, and the number of observations and variables themselves. Therefore, I will originally run the code but save them in RDS file so the model can be used and evaluated easily, and takes less time.

For fitting Regression model it took me 5 minutes and when it comes to fitting the Classification Model it took 1-2 minutes.

5.2.2 Model Evaluation



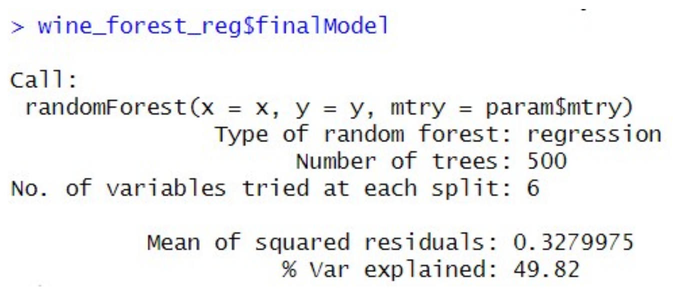
```
> wine_forest_reg
Random Forest

1279 samples
  11 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 1023, 1024, 1023, 1022, 1024, 1022, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
   2    0.5883191  0.4836151  0.4466806
   6    0.5844771  0.4819009  0.4378877
  11    0.5886496  0.4724682  0.4379767

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 6.
```

Figure 25: Model Evaluation for Reggression Model [Random Forest]

Regression Model The "mtry" parameter in Random Forest refers to the number of predictors randomly selected at each split when constructing decision trees. In our model, we observed that after several iterations, the algorithm determined that using 6 predictors provided the lowest Root Mean Squared Error (RMSE), which is an estimation of the model's error.

Random Forest incorporates its own form of cross-validation known as Out-of-Bag (OOB) Error. During training, the algorithm randomly samples data for each decision tree and evaluates the model's performance on the remaining "out-of-bag" samples that were not included in the training set for that particular tree. This OOB evaluation acts as a form of validation, allowing the model to assess its performance on unseen data without the need for additional cross-validation or train-test splitting techniques.

By leveraging the OOB Error, Random Forest provides an internal evaluation mechanism that helps ensure robust and reliable performance. It eliminates the need for separate validation sets or explicit cross-validation, streamlining the model evaluation process.

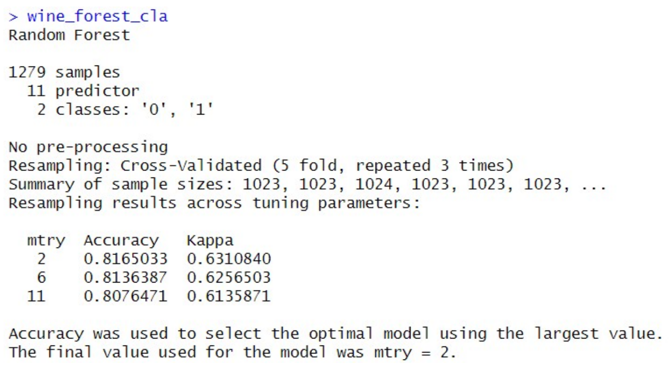


```
> wine_forest_reg$finalModel

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 6

          Mean of squared residuals: 0.3279975
                    % Var explained: 49.82
```

Figure 26: Regression Final Model

KEY FINDINGS:

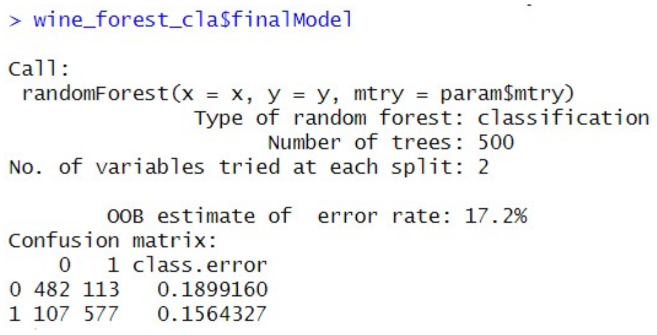
The algorithm valued its model accuracy at 49.82%, which is quite low. But if we compare this to our first linear regression model, this one is actually improved by a little.



```
> wine_forest_cla
Random Forest

1279 samples
  11 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 1023, 1023, 1024, 1023, 1023, 1023, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8165033  0.6310840
   6    0.8136387  0.6256503
  11    0.8076471  0.6135871

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

Figure 27: Model Evaluation for Classification Model [Random Forest]

Clasiffication Model After repeated attempts, the algorithm has determined that the optimal value for mtry is 2. This means that at each node of the Decision Tree, the algorithm will randomly select 2 predictors to consider for splitting. This process is repeated until the Decision Tree is fully constructed.



```
> wine_forest_cla$finalModel

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 17.2%
Confusion matrix:
    0   1 class.error
0 482 113   0.1899160
1 107 577   0.1564327
```

Figure 28: Classification Final Model

As previously mentioned, OOB (Out-Of-Bag) is a term used in Random Forest to refer to randomly sampled observations that are treated as unseen data for model evaluation. In this case, the OOB error rate is calculated to be 17.2%, which corresponds to an accuracy of 82.8%., the highest among all the models considered thus far.

The accuracy for the above output can be calculated by subtracting the OOB error rate (17.2%) from 100%:

Accuracy = 100% - OOB error rate
= 100% - 17.2%
= 82.8%

Therefore, the accuracy of the Random Forest model in the above output is approximately 82.8%.

5.2.3 Conclusions

- Random Forest stands out as the top performer. In terms of regression, it demonstrates an ability to explain 49.82% of the variability in the target variable, leaving the remaining 50.18% unexplained, likely due to factors beyond the predictors used in the model.
- Switching to classification, the Random Forest model achieves an impressive Accuracy of 82.8% and Precision at 83.6%. These metrics signify the model's ability to correctly classify instances and make precise predictions, making it the most effective model among the alternatives.

6 Comparing Models

1. The k Nearest Neighbour Model work best in a scaled and numerical predictors environment, which we can provide easily, although the interpretation of the model components for this one is not one of its strength. Its Accuracy is at 76.5%, while the Precision is 76.60%.
2. The Random Forest Model is still the best performer compared to of the K-NN classification method that we have tried in this report. Showing Accuracy at 82.8% and Precision at 83.6%, the only downside to this model is that it consumes time and resources for its computation to finish.

7 Final Conclusion

Random Forest is a highly reliable machine learning method that strikes a balance between accuracy, precision, and resource efficiency, as demonstrated in our comparative analysis of different models in the report. From our analysis, the following predictors consistently emerged as influential factors in determining red wine quality:

1. Alcohol: The alcohol content of the wine has a significant impact on its overall quality rating.
2. Volatile Acidity: The presence of volatile acids in the wine, which contribute to its aroma, strongly influences its quality.
3. Sulphates: The level of sulphates in the wine plays a crucial role in determining its quality, as it affects various sensory aspects.
4. Total Sulfur Dioxides: The total amount of sulfur dioxide present in the wine has a notable influence on its quality attributes.

Based on these findings, we can confidently conclude that these predictors are among the key factors that contribute to the determination of red wine quality.

8 APPENDIX

8.1 GitHub Link

You can find R files on given GitHub link
`https://github.com/Spandanaadivishnu/Supervised_Learning_Project.git`

8.2 R- Code

```r
#Setting Working Directory
getwd()
setwd("C:/Users/SaiSpandana/OneDrive/Desktop/Supervised_Learning_Project")

#Loding packages
library(readr)  #Loading the data
library(dplyr) # library for data manipulation
library(tidyr)
library(glue)
library(ggplot2)
library(plotly)
library(GGally)
library(rsample)
library(MASS)
library(performance)
library(lmtest)
library(car)
library(gtools)
library(caret)
library(class)
library(e1071)
library(ROCR)
library(partykit)
library(gridExtra)
library(randomForest)
#Loding the Data
wine <- read.table("winequality-red.csv", sep = ";", header = TRUE)
view(wine)

#Dataset Insights
colnames(wine)

#Feature of data
head(wine)

#Dimension of data
dim(wine)
```

```
#Structure of data
str(wine)

#Summary of data
summary(wine)

#Checking for the null values
apply(wine, 2, function(x)sum(is.na(x)))

# Unique valules in the target variable
prop.table(table(wine$quality))

# Draw a histogram for a given dataframe and variable
# Use deparse() and substitute() functions to decode column name from
# a variable passed as an argument to the function, to be displayed
# on x axis (xlab())
draw_hist <- function(dataframe, variable)
{
  # Save histogram definition to the plot variable
  plot <- ggplot(data = dataframe, aes(x = variable)) +
    geom_histogram(color = 'black', fill = '#099DD9') +
    xlab(deparse(substitute(variable)))
  return(plot)
}
# Build a matrix of small histograms with 3 columns
# using customly defined draw_hist() function
grid.arrange(draw_hist(wine, wine$fixed.acidity),
             draw_hist(wine, wine$volatile.acidity),
             draw_hist(wine, wine$citric.acid),
             draw_hist(wine, wine$residual.sugar),
             draw_hist(wine, wine$chlorides),
             draw_hist(wine, wine$free.sulfur.dioxide),
             draw_hist(wine, wine$total.sulfur.dioxide),
             draw_hist(wine, wine$density),
             draw_hist(wine, wine$pH),
             draw_hist(wine, wine$sulphates),
             draw_hist(wine, wine$alcohol),
             draw_hist(wine, wine$quality),
             ncol = 3)

#Visuvalizing the dependent variable
# Plot a histogram of quality values
ggplot(data = wine, aes(x = quality)) +
  geom_histogram(color = 'black', fill = 'deepskyblue1', binwidth = 1) +
  # Used to show 0-10 range, even if there are no values close to 0 or 10
  scale_x_continuous(limits = c(0, 10), breaks = seq(0, 10, 1)) +
  xlab('Quality of Red Wine') +
  ylab('Number of Red Wines')

wine$quality_high <- as.factor(ifelse(wine$quality>=6, 1, 0))
glimpse(wine$quality_high)
prop.table(table(wine$quality_high))

#Cross validation
#Splitting the data
RNGkind(sample.kind = "Rounding")
set.seed(123)

# index sampling
index_wine <- initial_split(wine, prop = 0.8, strata = "quality_high")

# splitting
wine_train <- training(index_wine)
wine_test <- testing(index_wine)

#checking proportions on separated dataframes
prop.table(table(wine_train$quality_high))
prop.table(table(wine_test$quality_high))

#Exploratory data analysis and visualization
#box plot of variables
p2 <- ggplot(data = stack(wine %>% dplyr::select(volatile.acidity, citric.acid, chlor
             mapping = aes(x = ind, y = values)) +
  geom_boxplot(fill = "pink")+
  theme_dark()+
  labs(title = "Boxplot of Volatile Acidity, Citric Acid, Chlorides, Sulphates",
       x = "Predictors",
       y = "Value")

ggplotly(p2)

p3 <- ggplot(data = stack(wine %>% dplyr::select(fixed.acidity, residual.sugar, alcol
             mapping = aes(x = ind, y = values)) +
  geom_boxplot(fill = "green")+
  theme_dark()+
  labs(title = "Boxplot of Fixed Acidity, Residual Sugar, Alcohol",
       x = "Predictors",
       y = "Value")
```

```
ggplotly(p3)

p4 <- ggplot(data = stack(wine %>% dplyr::select(free.sulfur.dioxide, total.sulfur.d
             mapping = aes(x = ind, y = values)) +
  geom_boxplot(fill = "cyan")+
  theme_dark()+
  labs(title = "Boxplot␣of␣Free␣and␣Total␣Sulfur␣Dioxide",
       x = "Predictors",
       y = "Value")

ggplotly(p4)

#Checking correlations between predictors
ggcorr(wine_train, label = T, hjust = 0.9, label_size = 3, layout.exp = 3)

#By observing the plot there is a strong correlation b/w free.sulfur.dioxide,total.s
# volatile.acidity

#Scatter plots for strong correlated variables
#Free.sulfur.dioxide and Total.sulfur.dioxide
p5 <- ggplot(data = wine %>% mutate(label = glue("Free␣Sulfur␣Dioxide␣=␣{free.sulfur
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Total␣Sulfur␣Dioxide␣=␣{total.sulfur.diox
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Ratio␣=␣{round(free.sulfur.dioxide/total.
                 mapping = aes(x = free.sulfur.dioxide, y = total.sulfur.dioxide, te
  geom_point(aes(color = free.sulfur.dioxide/total.sulfur.dioxide))+
  theme_dark()+
  labs(x = "Free␣Sulfur␣Dioxide␣(ppm)",
       y = "Total␣Sulfur␣Dioxide␣(ppm)",
       color = "Ratio␣of␣Free:Total")

ggplotly(p5, tooltip = "label")

#Fixed Acidity and Volatile Acidity
p6 <- ggplot(data = wine %>% mutate(label = glue("Fixed␣Acidity␣=␣{fixed.acidity},
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Volatile␣Acidity␣=␣{volatile.a
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Ratio␣=␣{round(fixed.acidity/v
                 mapping = aes(x = fixed.acidity, y = volatile.acidity, text = label
  geom_point(aes(color = fixed.acidity/volatile.acidity))+
  theme_dark()+
  labs(x = "Fixed␣Acidity",
       y = "Volatile␣Acidity",
       color = "Ratio␣of␣Fixed:Volatile")

ggplotly(p6, tooltip = "label")
```

```
#Fixed Acidity and pH
p7 <- ggplot(data = wine %>% mutate(label = glue("Fixed Acidity = {fixed.acidity},
                                                   pH = {pH}")),
                  mapping = aes(x = fixed.acidity, y = pH, text = label))+
  geom_point(color = "aquamarine")+
  theme_dark()+
  labs(x = "Fixed Acidity",
       y = "pH")

ggplotly(p7, tooltip = "label")

#Volatile Acidity and Citric Acid
p8 <- ggplot(data = wine %>% mutate(label = glue("Volatile Acidity = {volatile.acidit
                                                   Citric Acid = {citric.acid}"))
                  mapping = aes(x = volatile.acidity, y = citric.acid, text = label))+
  geom_point(color = "DarkGreen")+
  theme_dark()+
  labs(x = "Volatile Acidity",
       y = "Citric Acid")

ggplotly(p8, tooltip = "label")

table(wine$quality_high)

#Finding the means
wine_char <- wine %>%
  mutate(quality_high = ifelse(quality>=6, "high", "low")) %>%
  group_by(quality_high) %>%
  summarise_all(mean) %>%
  dplyr::select(-quality)

wine_char

p9 <- wine_char %>%
  pivot_longer(cols = -quality_high, names_to = "names", values_to = "values")%>%
  mutate(label = glue("Red Wine Quality? {quality_high}
                      Average of {names} = {round(values,2)}")) %>%
  ggplot(mapping = aes(x=names, y=values))+
  geom_line(aes(group = quality_high, color = quality_high))+
  geom_jitter(mapping = aes(x=names, y=values, color = quality_high, text = label))+
  theme_dark()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  labs(title="Average Characteristics of Top-Rated vs Lower-Rated Red Wines",
       x="Predictor",
       y="Value",
```

```r
        color="Red Wine Quality?")

ggplotly(p9, tooltip = "label")

#Classification Model: k-NN
#Pre-processing of data
summary(wine_train)

#Separating Predictors and Target Variables
wine_train_x <- wine_train %>%
  dplyr::select(-c("quality", "quality_high"))

wine_train_y <- wine_train %>%
  pull(quality_high)

wine_test_x <- wine_test %>%
  dplyr::select(-c("quality", "quality_high"))

wine_test_y <- wine_test %>%
  pull(quality_high)



#Scaling the data
wine_train_x_scaled <- scale(wine_train_x)
wine_test_x_scaled <- scale(wine_test_x,
                            center = attr(wine_train_x_scaled, "scaled:center"),
                            scale = attr(wine_train_x_scaled, "scaled:scale"))
summary(wine_train_x_scaled)
summary(wine_test_x_scaled)

#Finding Optimum k
sqrt(nrow(wine_test_x_scaled))

#Fitting the model
wine_knn_pred_k17 <- knn(train = wine_train_x_scaled,
                         test = wine_test_x_scaled,
                         cl = wine_train_y,
                         k=17)

wine_knn_pred_k15 <- knn(train = wine_train_x_scaled,
                         test = wine_test_x_scaled,
                         cl = wine_train_y,
                         k=15)

wine_knn_pred_k19 <- knn(train = wine_train_x_scaled,
                          test = wine_test_x_scaled,
                          cl = wine_train_y,
                          k=19)

#Model evaluation
#Using k=17 (closest number to optimun)
confusionMatrix(data = wine_knn_pred_k17,
                reference = wine_test_y,
                positive = "1")
#Using k=15 (closest number to optimun)
confusionMatrix(data = wine_knn_pred_k15,
                reference = wine_test_y,
                positive = "1")
#Using k=19 (closest number to optimun)
confusionMatrix(data = wine_knn_pred_k19,
                reference = wine_test_y,
                positive = "1")

#Model fitting of Random forest
#Fitting the regression model of Random forest
 set.seed(314)

ctrl <- trainControl(method = "repeatedcv",
                      number = 5, # k-fold
                      repeats = 3) # repetition

wine_forest_reg <- train(quality ~ .,
                          data = wine_train %>% dplyr::select(-quality_high),
                          method = "rf", # random forest
                          trControl = ctrl)

saveRDS(wine_forest_reg, "wine_forest_reg.RDS") # saving model

#Fitting classification model
wine_forest_cla <- train(quality_high ~ .,
                         data = wine_train %>% dplyr::select(-quality),
                         method = "rf", # random forest
                         trControl = ctrl)

saveRDS(wine_forest_cla, "wine_forest_cla.RDS") # saving model

#Model Evaluation
wine_forest_reg <- readRDS("wine_forest_reg.RDS")
wine_forest_reg
```

```
wine_forest_reg$finalModel

#Classification model
wine_forest_cla <- readRDS("wine_forest_cla.RDS")
wine_forest_cla
wine_forest_cla$finalModel
#Confusion matrix
wine_forest_cla_pred <- predict(object = wine_forest_cla,
                                newdata = wine_test,
                                type = "raw")

confusionMatrix(data = wine_forest_cla_pred,
                reference = wine_test$quality_high,
                positive = "1")
```

9 References

- http://www.sthda.com/english/articles/25-clusteranalysis-in-r-practical-guide/
- https://archive-beta.ics.uci.edu/dataset/109/wine
- Breiman and Cutler's Random Forests for Classification and Regression
- https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/