INDIVIDUAL ASSIGNMENT 1

TECHNOLOGY PARK MALAYSIA

CT046-3-M-AML

APPLIED MACHINE LEARNING

ASSIGNMENT 1 (PART -B)

HAND OUT DATE: 26 JULY 2021

HAND IN DATA: 6 SEPTEMBER 2021

WEIGHTAGE: 60%

INSTRUCTIONS TO CANDIDATES:

1 Submit your assignment at the administrative counter.

2 students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing).

3 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.

4 Cases of plagiarism will be penalized.

5 The assignment should be bound in an appropriate style (comb bound or stapled).

6 Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.

7 You must obtain 50% overall to pass this module.

Acknowledgment

First and foremost, I would like to thank my lecturer Professor De Mandava Rajeswari for introducing this project to the Applied Machine Learning class as it would greatly benefit us in terms of gaining more knowledge on machine learning using the R program. She has been very helpful in giving guidance and feedback on my work to keep me on track with my project.

Not to forget my classmates who were very helpful in guiding and helping me throughout my project red on wine analysis.

Abstract

Nowadays, quality has become an important factor for consumers, which in turn has led various industries to explore various ways of ensuring the high quality of their products. In today's era, the consumption of wine has rapidly increased as it is no longer consumed for social purposes but also view to be able to give health benefits to its consumers, especially those dealing with heart problems (Castaldo *et al.*, 2019). Moreover, consumers have not taken one step ahead by taking into consideration the quality of wines through their certifications. Nowadays, manufacturers are beginning to utilize software to automate validation tasks, which has led to a great reduction in time spent to rate the product through an expert. Thus, in this project, we are going to study the dataset of the red wine by reviewing the usage of various machine learning models used to predict wine quality by previous researches. Red wine is produced through a process of extracting the color and flavor components from the skin of the grapes. The input variables of the dataset include volatile acidity, chlorides, fixed acidity, citric acid, residual sugar, total sulfur dioxide, free sulfur dioxide, density, pH, alcohol, and sulfate, white the output variable is quality which has values ranging from 0 to 10, 0 being very bad and 10 being excellent. Through previous work, we have deduced that the dataset studies are unbalanced and will need some resampling work before it is used in the machine learning models. Thus, this work would also propose the use of the Synthetic Minority Over-Sampling Technique to balance the dataset before using it in three different machine learning models, which are Random Forest, Support Vector Machine, and K Nearest Neighbour.

Table of contents

Chapter 1: Introduction

1.1 Introduction

Portugal is reported to be the 11th wine producer globally with a market share of 3.17% (Cortez *et al.*, 2009). The wine sector is very famous in Portugal as it is known as a major driver of the country's economy and also a very important sector responsible for maintaining rural areas by providing employment and contributes to the stability of the economy (Carmen *et al.*, 2021). The production of wine begins with the fermentation of grapes and other fruits with lower and this beverage is known to have lower alcohol content.

Portugal's wine industry takes the process of certification and evaluation of its quality very seriously as it is essential to prevent the wine from being contaminated before it is distributed out to the consumers. Ensuring its quality is of paramount importance as staying in the industry, given the many competitions in the market is not easy. Thus, many wine industries are eagerly searching for new technologies to cater to its recent growth in demand. "Wineinformatics" is the latest data science application that utilizes wine as the domain knowledge, which incorporates data science and wine-related datasets, including physicochemical laboratory data and wine review (Chen *et al.*, 2018). Assessment of wine quality is usually done is two methods which are physicochemical test and sensory test (Gupta, 2018). Lab tests are usually sufficient to carry out physicochemical tests without the need of a human expert, however, a human expert is required for sensory tests. With the huge demand for wine, it is challenging to ensure quality with experts as this will contribute to the rise in cost. Thus, with the development of technology of machine learning, manufactures are now starting to depend on various techniques to predict the wine quality with human interference. Not only will this save a lot of time, but it will also indirectly contribute to cutting costs for the manufacturers. Furthermore, manufactures can get hands-on with tuning the wine quality by calibrating various criteria during the development phase.

From the literature that has been reviewed in Chapter 2, we can deduce that a huge scope is still available for improvement. In this project, we will be conducting an experiment operating on the wine quality dataset that is readily available on the UCI machine learning repository by (Cortez *et al.*, 2009). The dataset includes two types of wine, however, in this project we will work on red wine.

1.2 Problem Statement

Although much past work has successfully incorporated the use of various machine learning models to predict the quality of wine, there has been no research done to fix the unbalanced red wine dataset. The wine quality values vary from 3 to 9, usually evaluated by the blind taste, which then graded that wine ranging from 0 (very bad) to 10 (excellent). The quality variable as explained by (Sharma, 2017) is being skewed to the left as the mean was reported to be 5.818, which is lower than the median of 6.000. This shows that there is a pull towards the lower side of the quality spectrum compared to the high-quality wines.

1.3 Research Goal

This project aims to study the unbalanced red wine dataset and propose the Synthetic Minority Over-Sampling Technique to rectify the minority class to balance the dataset. SMOTE functions by oversampling the minority class as it uniquely selects the instances that require resampling to eliminate overfitting issues identified in traditional oversampling methods. Furthermore, the balanced dataset will then be used to train 3 different machine learning models which are Random Forest (RF), Support Vector Machine (SVM) and K-Nearest Neighbour. Its performance will be measured with the results of the models without the resampling technique.

1.4 Research Objectives

1. To balance the dataset using the resampling technique SMOTE to rectify the minority class.
2. To predict wine quality using three machine learning models comprising of Random Forest, Support Vector Machine and K-Nearest Neighbour through the R programming language.
3. To determine the Person correlation coefficient and carry out feature selection of variables that contribute most to wine quality.
4. Calculate the performance of each model by evaluating its True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values to deduce the accuracy, precision, recall and f1 score.

Chapter 2: Literature Review

2.1 Introduction

This section investigated and discussed related works done by past researchers on wine quality analysis, which consists of different techniques comparisons using various machine learning models and how well they performed to increase accuracy. This is particularly important to find a gap that could be implemented in future experiment.

2.2 Related work

Gupta and C, (2021) conducted an experiment using the UCL machine learning repository on wine analysis, which ad 12 features. 11 of those features being independent variables and another the dependent variable, which is the target variable in this analysis. No data cleaning was implemented in this dataset as the author reported no missing values or outliers. They then proceed to use the correlation tool to select dominant features, which reported that volatile acidity shared a correlation with quality as well as the concentration of acid whereby both contributed to better wine quality. Furthermore, alcohol was another important feature that also contributed to the quality of the wine. On the contrary to the positive correlation, the concentration of chloride seemed to have had a negative correlation with the targeted variables. They deduced that the key attributes were acidity, sugar level, chlorides, sulphur, alcohol, pH and density. Various machine learning methods were implemented such as Decision Tree, Random Forest, Support Vector Machine, K Nearest Neighbor and MP5 Model. Before that, the dataset was split into 75:25 ratios. The author concluded that MP5 emerged as the model with the highest accuracy of 81.81% for the red wine dataset and 83.27% for the white wine dataset. They also reported that most wines available in the dataset were rated as average quality with a very low amount of poor and excellent being quality wines.

Next, another work by (Pawar *et al.,* 2019) also conducted a similar approach on the same dataset. However, the experiment was done only using red wine. Similarly, they also reported that the dataset was unordered and unbalanced as there were many more average quality wines than excellent or poor ones. Coefficient of determination was used to measure the correlation of the prediction and the actual values. The dataset was split into 80:20% and trained using various models such as Random Forest, Decision Tree, Support Vector Machine

and Stochastic Gradient Descent. To convert the labels into numeric formats, label encoding was implemented. The author reported that volatile acidity and residual sugar were not as an impact on quality as the rest of the variables, thus, they were eliminated. Conversely, they found that citric acid has a directly proportional relationship with quality as well as free Sulphur dioxide. The model that performed best was Random Forest with an accuracy of 87.33%. Another work by (Er, 2016) also reported that Random Forest emerged as the best performing model in his work to analyze the wine dataset. However, the author has a slightly different approach by applying two different methods of k-fold cross-validation and percentage split mode. For this experiment, k = 10 was chosen as it gave the best classification result. An accuracy of 99.52% and 99.46% was achieved using the two methods using Random Forest for both the red and white wine dataset. Furthermore, PCA was employed to increase the classification success as the number of features was reduced. Once again, Random Forest managed to outperform other models which were K Nearest Neighbour and Support Vector Machine with an accuracy of 71.232% for cross-validation and 73.4375% for percentage split.

Dahal *et al.*, (2021) implemented various machine learning models such as the Ridge Regressor (RR), Support Vector Machine, Gradient Boosting Regressor (GBR) and multi-layer Artificial Neural Network to predict the quality analysis. The author chose to focus on the red wine dataset. He first performed a descriptive analysis and Pearson correlation coefficient (r) was applied to find the association between the variables. They reported that the amount of alcohol had the highest r value while citric acid had the lower correlation with the targeted variable. To standardize the unbalance data, standardization and normalization were performed as they found total Sulphur dioxide values to be extremely larger compared to the chlorides. The data was split using a 3:1 ratio to prevent the occurrence of overfitting, which could affect the execution of the machine learning algorithms (Dahal and Gautam, 2020). The GBR model emerged as the superior model with the highest R-value of 0.6057 and lowest MSE value of 0.3741. They also mentioned that the least important feature was free sulphur acid.

In addition, (Kumar *et al.*, 2020) implement Random Forest, Support Vector Machine and Naïve Bayes on the same dataset of red wine. They calculated the confusion matric, relevant performance measures and made a comparison between the models used to analyze the dataset. The dataset was also split into a 70:30 ratio as previous work discussed earlier. They concluded that Support Vector Machine performed the best with an accuracy of 67.25% on the testing dataset. Another author (Gupta, 2018) also reported its best performance model to be Support Vector Machine. He first determined the reliance of the target variables with

other independent variables. The dataset was also pre-processed using linear transformation as they found larger amplitudes on some variables compared to others such as sulfates vs sulfur dioxide. A regression summary on the independent variable was made which described the dependency on the targeted variable individually. The value of adjusted R^2 showed 35.61% dependency of quality on all predictors. Important variables were reported to be volatile acidity, chlorides, free sulfur dioxide, total sulfur dioxide, pH, sulfates, and alcohol. The author then suggested that SVM would perform better if selected predictors only were used.

Lastly, (Sinha and Kumar, 2020) also worked on the same dataset like previously described. Data cleaning was not necessary as no empty columns were found. They also reported that fewer wines rated poor or excellent compared to average quality wines. They also performed statistical analysis to understand the dataset better. A correlation matrix was performed and deduced that volatile acid had a relation to quality as well as high content of alcohol contributed to more pleasant-tasting wine. Contrastingly, high-quality wine is negatively associated with volatile acidity. This may be due to an unpleasant smell indicating that the wine is spoilt. The models used in this analysis were Stochastic descent gradient, Support Vector Machine, and Random Forest. Random Forest outperformed other models with an accuracy of 87.33%.

2.3 Comparison of Previous Methods

Table 1: Comparison of methods by previous research

Citation	Dateset	EDA	Model	Pre-processing	Fine Tuning	Result	Future Work
Gupta and C, 2021	12 variables	Outlier and missing analysis	J48	Not mentioned	Confusion matrix	Accuracy: 0.5600 (red) Accuracy: 0.6936 (white)	Broad data set may be used and other machine learning techniques
	11 independent variables	Correlation matrix	KNN		Confidence interval, p- value	Accuracy: 0.6139 (red) Accuracy: 0.6052 (white)	
	1 target variable	Feature analysis	SVM		Performance measure	Accuracy: 0.6234 (red) Accuracy: 0.6372 (white)	
	UCI Machine learning repository on wine dataset 4898 white wine 1599 red wine sample		CART		Split ratio 75:25	Accuracy: 0.7075 (red) Accuracy: 0.7819 (white)	
			RF			Accuracy: 0.7325 (red) Accuracy: 0.7639 (white)	
			M5P			Accuracy: **0.8181** (red) Accuracy**: 0.8327** (white)	
			Feature selection			Acidity, Sugar content, Chlorides, Sulfur, Alcohol, pH, Density	
Pawar *et al.,* 2019	12 variables	Outlier and missing analysis	RF	Feature elimination	Split ratio 80:20	Accuracy: **0.8733**	Not mentioned
	11 independent variables	Feature analysis	SVM		Label- encoding	Accuracy: 0.8500	
	1 target variable		Stochastic Gradient Descent		Confusion matrix	Accuracy: 0.8100	
	UCI Machine learning repository on wine dataset		Feature selection		Performance measure	Citric acid Sulphur dioxide	

Dahal *et al.*, 2021	12 variables	Outlier and missing analysis	RR	Feature scaling	Split ratio 3:1	MSE: 0.3869, R = 0.5897	Possible to obtain better performance for ANN if training set is increased,
	11 independent variables	Statistical analysis	SVM			MSE: 0.3862, R =0.5971	
	1 target variable	Person correlation coefficient	GBR			**MSE: 0.3741 R = 0.6057**	
	UCI Machine learning repository on wine dataset 4898 white wine 1599 red wine sample		ANN			MSE: 0.4, R = 0.6057	
			Feature selection			Alcohol, Sulphates, Volatile acidity, Total sulfur dioxide, Chlorides, Density, Fixed acidity, pH	
Er, 2016	12 variables 4898 white wine 1599 red wine sample	Outlier and missing analysis	KNN	PCA	10-fold cross validation	Accuracy: 0.648 (red) (cross validation) Accuracy:0.678(red) (percentage split) Accuracy: 0.647 (white)(cross validation) Accuracy: 0.67.8 (white) (percentage split)	Not mentioned
	11 independent variables	Statistical analysis	RF		Split ratio 80:20	Accuracy: **0.712** (red) (cross validation) Accuracy:0.734 (red) (percentage split) Accuracy: **0.699** (white) (cross validation) Accuracy: 0.674 (white) (percentage split)	
	1 target variable		SVM		Confusion matrix	Accuracy: 0.580(red) (cross validation) Accuracy:0.569 (red) (percentage split) Accuracy: 0.522(white)(cross validation)	
	UCI Machine learning repository on wine dataset				Performance measure		

						Accuracy: 0.512 (white) (percentage split)	
Kumar *et al.*, 2020	12 variables	Outlier and missing analysis	RF		Split ratio 70:30	Accuracy: 0.6583	Not mentioned
	11 independent variables		SVM		Confusion matrix	Accuracy: **0.6725**	
	1 target variable		NB		Performance measure	Accuracy: 0.5590	
	UCI Machine learning repository on wine dataset 4898 white wine 1599 red wine sample						
Gupta, 2018	12 variables	Outlier and missing analysis	Linear regression	Linear transformation		Red wine: $R^2 = 0.3606$ Adjusted $R^2 = 0.3561$ $P < 0.000$ White wine: $R^2 = 0.3068$ Adjusted $R^2 = 0.2803$ $P < 0.000$	To experiment on larger dataset and other machine learning algorithms.
	11 independent variables	Feature analysis	SVM			Values not mentioned, however, it is the best performance model	
	1 target variable		NN			Red wine: Test error: 0.1460 White wine: Test error: 0.2075	

	UCI Machine learning repository on wine dataset 4898 white wine sample 1599 red wine sample		Feature selection			Volatile acidity, Chlorides, Free sulfur dioxide, Total sulfur dioxide, pH, Sulphates, alcohol	
Mahima *et al.*, 2020	12 variables	Outlier and missing analysis	RF		Split ratio	RMSE: 0.6322 (red) RMSE: 0.6430 (white)	Not mentioned
	11 independent variables	Correlation matrix	KNN + RF			RMSE: 0.584 (red) RMSE: 0.541 (white)	
	1 target variable UCI Machine learning repository on wine dataset 1599 red wine sample		Feature selection			Sulphur dioxide (both free and total), Alcohol, Sulphates, Volatile acidity, Citric acid	
Sinha and Kumar, 2020	12 variables	Outlier and missing analysis	Stochastic descent gradient		Confusion matrix	Accuracy: 0.81	
	11 independent variables	Correlation matrix	SVM		Performance measure	Accuracy: 0.85	
	1 target variable	Statistical analysis	RF			Accuracy: 0.8733	
	UCI Machine learning repository on wine dataset 1599 red wine sample		Feature selection			pH, Volatile acidity, Alcohol	

2.4 Summary

Table 2: Summary of techniques used by previous work

Citation	Train/test split	Outlier & missing value analysis	Correlation tool	Feature analysis	Statistical analysis	Confusion matrix	Performance evaluation	PCA	Cross validation (check paper)
		Random Forest							
(Gupta and C, 2021)	YES	YES	YES	YES		YES	YES		
Pawar *et al.*, 2019	YES	YES		YES					
Er, 2016	YES				YES	YES	YES	YES	YES
Kumar *et al.*, 2020	YES	YES				YES	YES		
Mahima *et al.*, 2020	YES	YES	YES						
Sinha and Kumar, 2020		YES	YES	YES	YES	YES	YES		
		Support Vector Machine							
(Gupta and C, 2021)	YES	YES	YES	YES		YES	YES		
Pawar *et al.*, 2019	YES	YES		YES					
Dahal *et al.*, 2021	YES	YES	YES	YES	YES				
Er, 2016	YES				YES	YES	YES	YES	YES
Kumar *et al.*, 2020	YES	YES				YES	YES		
Gupta, 2018		YES	YES						
Sinha and Kumar, 2020		YES	YES	YES	YES	YES	YES		

		K- Nearest Neighbour							
(Gupta and C, 2021)	YES	YES	YES	YES		YES	YES		
Er, 2016	YES				YES	YES	YES	YES	YES
		J48							
(Gupta and C, 2021)	YES	YES	YES	YES					

CART									
(Gupta and C, 2021)	YES	YES	YES	YES		YES	YES		
MP5									
(Gupta and C, 2021)	YES	YES	YES	YES		YES	YES		
Stochastic Gradient Descent									
Pawar *et al.*, 2019	YES	YES		YES					
Sinha and Kumar, 2020		YES	YES	YES	YES	YES	YES		
Ridge Regressor									
Dahal *et al.*, 2021	YES	YES	YES	YES	YES				
Gradient Boosting Regressor									
Dahal *et al.*, 2021	YES	YES	YES	YES	YES				
Ridge Regressor									
Dahal *et al.*, 2021	YES	YES	YES	YES	YES				
Artificial Neural Network									
Dahal *et al.*, 2021	YES	YES	YES	YES	YES				
Naïve Bayes									
Kumar *et al.*, 2020	YES	YES				YES	YES		

2.4.1 Gap Analysis

From the work that was reviewed and studied earlier, it is seen that most of the authors reported an unbalance dataset which is skewed as most of the quality rated as average while very few were reported to be poor or excelled quality. The following question will try to address the gap to achieve our objectives.

1. Can the resampling technique using SMOTE able to fix the skewed dataset to balance data?
2. Will a balanced dataset provide better accuracy for the models proposed for future experiment?

2.4.2 Scope Analysis

Based on the analysis we have identified, a few key steps can be taken to reduce the scope to achieve our objective.

1. Balance the dataset using SMOTE by over-sampling the minority class.
2. Implement the balanced dataset in the models proposed: Random Forest, Support Vector Machine
3. Make a comparison between the models proposed using confusion matric and some performance measures to find out the superior model.
4. Make a comparison between the accuracy of the models before and after the resampling technique being applied.
5. Perform feature selection to out the important features for the particular model
6. Perform a correlation matrix to find out the reliance of independent variables on the target variable.

Chapter 3: Method and Dataset

3.1 Dataset Description

In this section, we will discuss in detail the dataset that is will be implemented for a future experiment. The background and its features are explained in detail in sections 3.1.1 and 3.1.2.

3.1.1 Introduction

The source of the dataset was prepared by Paulo Cortez (Cortez *et al.*, 2009) at the University of Minho, Guimaraes, Portugal. The original samples were collected from May 2004 to February 2007 are of original samples that were tested at the official certification entity (CVRVV). The datasets which comprise red and white wines are variants of the Portuguese "Vinho Verde". In this dataset, only the physicochemical (inputs) and sensory (output) variables are revealed to protect its privacy. There are two ways to view this dataset, which are classification or regression tasks. They are unbalanced and unordered as reported above by previous works, as there are many more average quality wines than excellent or poor ones. Its characteristics are multivariate with 4898 instances and 12 features, where 11 features are independent variables and 1 feature being the targeted variable. 1599 instances were reported in was red wine dataset and 3299 instances for the case of white wine. In this case, the quality variable is being used as the targeted variable classed as 0 ranging from 0 to 10 (0 being poor and 10 being excellent). The author also suggested that feature selection could be used as not all inputs may be relevant in predicting the analysis.

3.1.2 Description of features

Table 3: Dataset description

Attributes	Description	Structure
Fixed acidity	Is the amount of fixed acids per volume (tartaric acid etc)	Numeric
Volatile acidity	Amount of volatile acids such as acetic acid Important to distill out from the wine as high levels can cause to a nasty, vinegary taste	Numeric
Citric acid	Contributes to the 'freshness' of wine flavor	Numeric
Residual sugar	The measure of natural sugar remained even after the fermentation process ends or is stopped	Numeric
Chlorides	The measure of salt found in the wine	Numeric
Free sulphur dioxide	Able to prevent microbial growth and the oxidation of wine, important during aging of wine	Numeric
Total sulphur dioxide	The total amount of bound and free sulfur dioxide	Numeric
Density	Weight of a specific volume of wine to the same volume of water	Numeric
pH	specifies the acidity or basicity of the wine where most wines, in general, are between 3-4 on the pH scale.	Numeric
Sulphates	Acts as an antimicrobial and antioxidant	Numeric
Alcohol	Measured in % col or alcohol by volume	Numeric
Quality	The final grade which represents the quality of scores	Factor

3.2 Exploratory Analysis

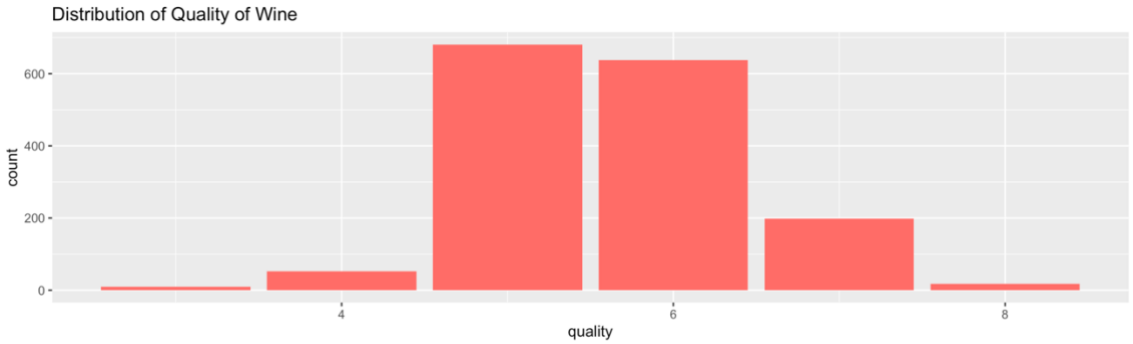
```
> raw <- read.csv("~/Desktop/APU/AML/ASSIGNMENT/AML Assignment/winequality-red.csv", sep =";")
> head(raw)
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol quality
1           7.4             0.70        0.00            1.9     0.076                   11                   34  0.9978 3.51      0.56     9.4       5
2           7.8             0.88        0.00            2.6     0.098                   25                   67  0.9968 3.20      0.68     9.8       5
3           7.8             0.76        0.04            2.3     0.092                   15                   54  0.9970 3.26      0.65     9.8       5
4          11.2             0.28        0.56            1.9     0.075                   17                   60  0.9980 3.16      0.58     9.8       6
5           7.4             0.70        0.00            1.9     0.076                   11                   34  0.9978 3.51      0.56     9.4       5
6           7.4             0.66        0.00            1.8     0.075                   13                   40  0.9978 3.51      0.56     9.4       5
> dim(raw)
[1] 1599   12
> str(raw)
'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 0.073 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 ...
```

```
> library(psych)
> describe(raw)
                      vars    n  mean    sd median trimmed   mad  min    max  range skew kurtosis   se
fixed.acidity            1 1599  8.32  1.74   7.90    8.15  1.48 4.60  15.90  11.30 0.98     1.12 0.04
volatile.acidity         2 1599  0.53  0.18   0.52    0.52  0.18 0.12   1.58   1.46 0.67     1.21 0.00
citric.acid              3 1599  0.27  0.19   0.26    0.26  0.25 0.00   1.00   1.00 0.32    -0.79 0.00
residual.sugar           4 1599  2.54  1.41   2.20    2.26  0.44 0.90  15.50  14.60 4.53    28.49 0.04
chlorides                5 1599  0.09  0.05   0.08    0.08  0.01 0.01   0.61   0.60 5.67    41.53 0.00
free.sulfur.dioxide      6 1599 15.87 10.46  14.00   14.58 10.38 1.00  72.00  71.00 1.25     2.01 0.26
total.sulfur.dioxide     7 1599 46.47 32.90  38.00   41.84 26.69 6.00 289.00 283.00 1.51     3.79 0.82
density                  8 1599  1.00  0.00   1.00    1.00  0.00 0.99   1.00   0.01 0.07     0.92 0.00
pH                       9 1599  3.31  0.15   3.31    3.31  0.15 2.74   4.01   1.27 0.19     0.80 0.00
sulphates               10 1599  0.66  0.17   0.62    0.64  0.12 0.33   2.00   1.67 2.42    11.66 0.00
alcohol                 11 1599 10.42  1.07  10.20   10.31  1.04 8.40  14.90   6.50 0.86     0.19 0.03
quality                 12 1599  5.64  0.81   6.00    5.59  1.48 3.00   8.00   5.00 0.22     0.29 0.02
```

```
> #creating a new factorial variable called "Rating"
> raw$rating <- ifelse(raw$quality < 5, 'poor', ifelse(raw$quality < 7, 'average', 'excellent'))
> raw$rating <- ordered(raw$rating, levels = c('poor','average','excellent'))
> table(raw$rating)

     poor   average excellent
       63      1319       217
> summary(raw)
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar     chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900   Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200   Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539   Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500   Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
       pH          sulphates        alcohol        quality          rating
 Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000   poor     :  63
 1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000   average  :1319
 Median :3.310   Median :0.6200   Median :10.20   Median :6.000   excellent: 217
 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
```

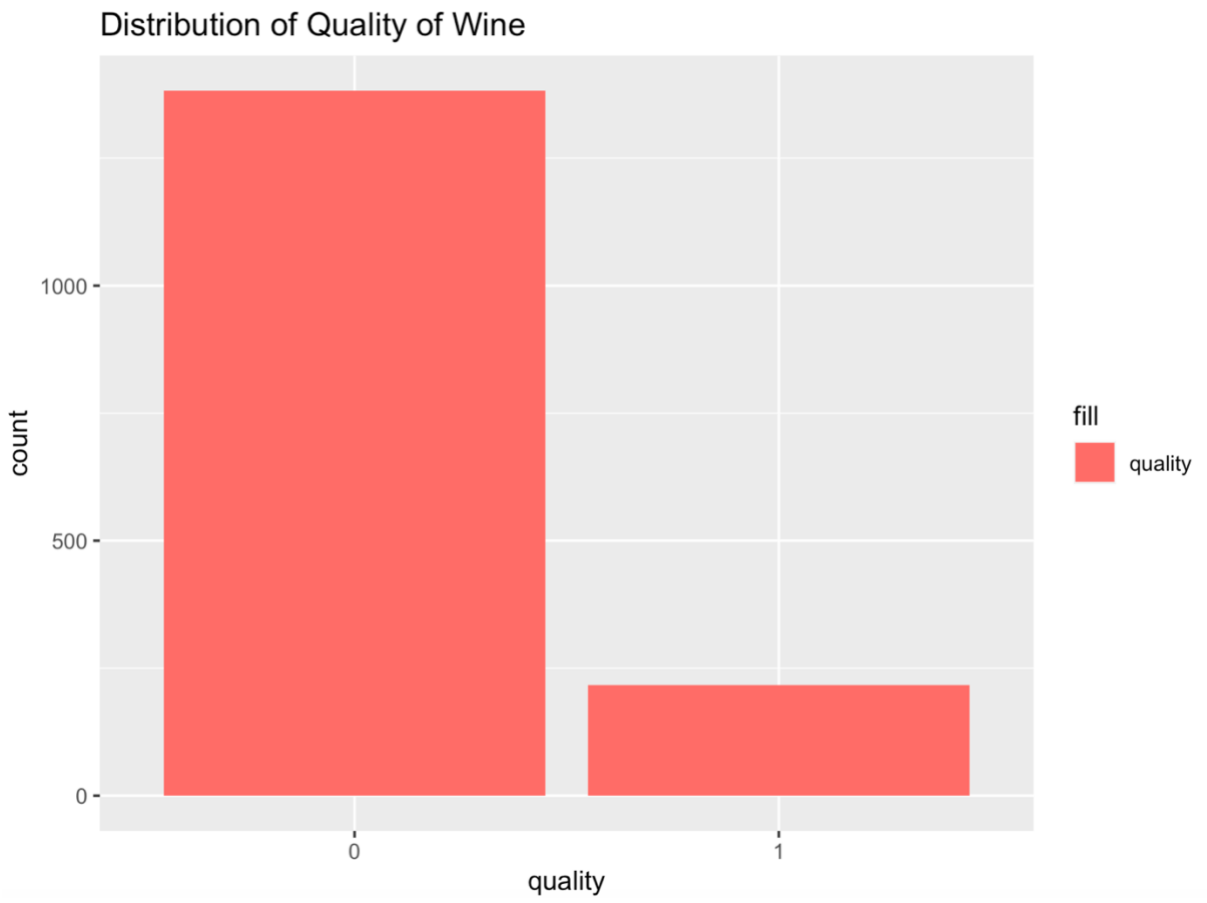
We can deduce that the dimension of the data set has 1599 samples with 12 variables. The structure of the data has 11 numeric variables and one of them being an integer. The library ‘psych’ was loaded so that we will be able to use the describe() function. This function describes the dataset in detail by reporting the mean, standard deviation, minimum and maximum values, skewness, etc.

```
> library(ggplot2)
> quality.wine <- ggplot(raw)+
+   geom_bar(aes(quality, fill = "quality"))+
+   labs(title = "Distribution of Quality of Wine", xlab = "Rating")
```



Through the plots created above, we can observe that most of the wine is rated to be on average 5 or 6 with the least ratings being 8 and below 4. We have transformed the quality variable into a binary factor of 0 and 1. 0 being poor quality and 1 being an excellent quality of the wine.

```
quality.wine <- ggplot(data)+
  geom_bar(aes(quality, fill = "quality"))+
  labs(title = "Distribution of Quality of Wine", xlab = "Rating"); quality.wine
```

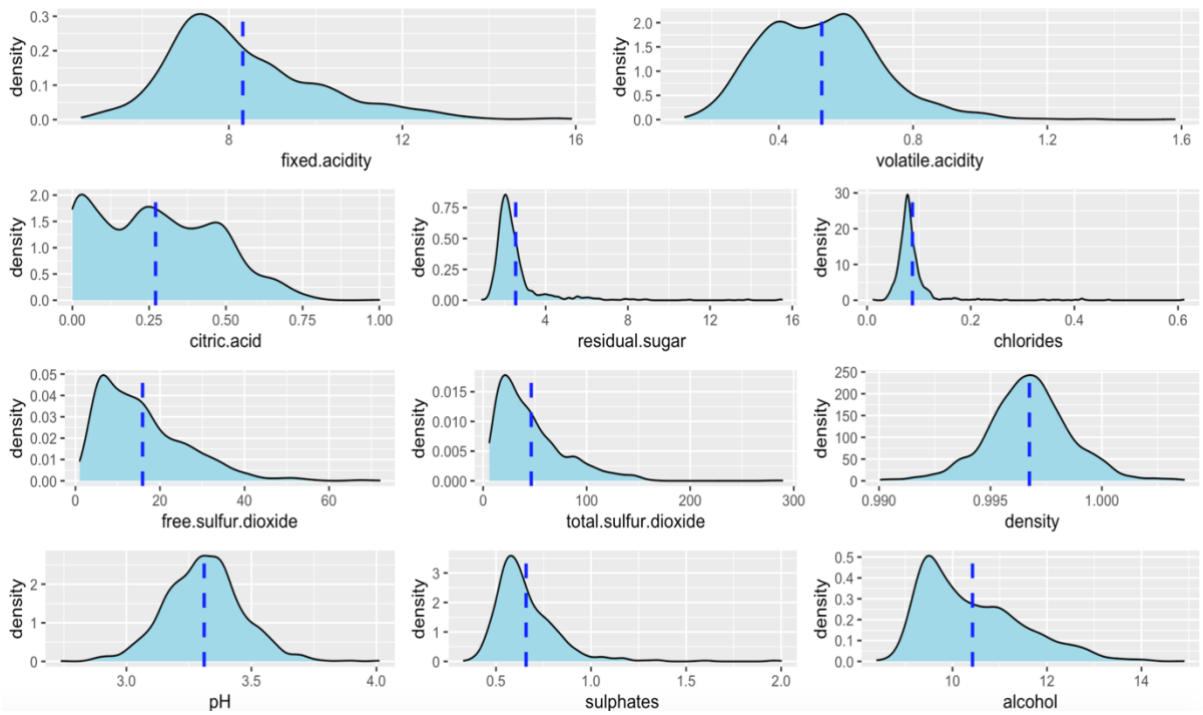


```
> fixed.acidity <- ggplot(raw, aes(fixed.acidity)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(fixed.acidity)),
+              color="blue", linetype="dashed", size=1); fixed.acidity
>
> # Volatile.acidity
> volatile.acidity <- ggplot(raw, aes(volatile.acidity)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(volatile.acidity)),
+              color="blue", linetype="dashed", size=1); volatile.acidity
>
> # Citric.acid
> citric.acid <- ggplot(raw, aes(citric.acid)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(citric.acid)),
+              color="blue", linetype="dashed", size=1); citric.acid
>
> # Residual.sugar
> residual.sugar <- ggplot(raw, aes(residual.sugar)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(residual.sugar)),
+              color="blue", linetype="dashed", size=1); residual.sugar
>
```

```
> # Chlorides
> chlorides <- ggplot(raw, aes(chlorides)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(chlorides)),
+              color="blue", linetype="dashed", size=1); chlorides
>
> # Free.sulfur.dioxide
> free.sulfur.dioxide <- ggplot(raw, aes(free.sulfur.dioxide)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(free.sulfur.dioxide)),
+              color="blue", linetype="dashed", size=1); free.sulfur.dioxide
>
> # Total.sulfur.dioxide
> total.sulfur.dioxide <- ggplot(raw, aes(total.sulfur.dioxide)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(total.sulfur.dioxide)),
+              color="blue", linetype="dashed", size=1); total.sulfur.dioxide
>
> # Density
> density <- ggplot(raw, aes(density)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(density)),
+              color="blue", linetype="dashed", size=1); density
>
```

```
> # pH
> pH <- ggplot(raw, aes(pH)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(pH)),
+              color="blue", linetype="dashed", size=1); pH
>
> # Sulphates
> sulphates <- ggplot(raw, aes(sulphates)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(sulphates)),
+              color="blue", linetype="dashed", size=1); sulphates
>
> # alcohol
> alcohol <- ggplot(raw, aes(alcohol)) +
+   geom_density(fill = "lightblue")+
+   geom_vline(aes(xintercept=mean(alcohol)),
+              color="blue", linetype="dashed", size=1); alcohol
>
> first_row = plot_grid(fixed.acidity,volatile.acidity,nrow =1)
> second_row = plot_grid(citric.acid,residual.sugar,chlorides, nrow = 1)
> third_row = plot_grid(free.sulfur.dioxide,total.sulfur.dioxide,density, nrow = 1)
> forth_row = plot_grid(pH,sulphates,alcohol, nrow = 1)
```

```
> plot_grid(first_row,second_row,third_row,forth_row,nrow = 4,ncol =1)
```



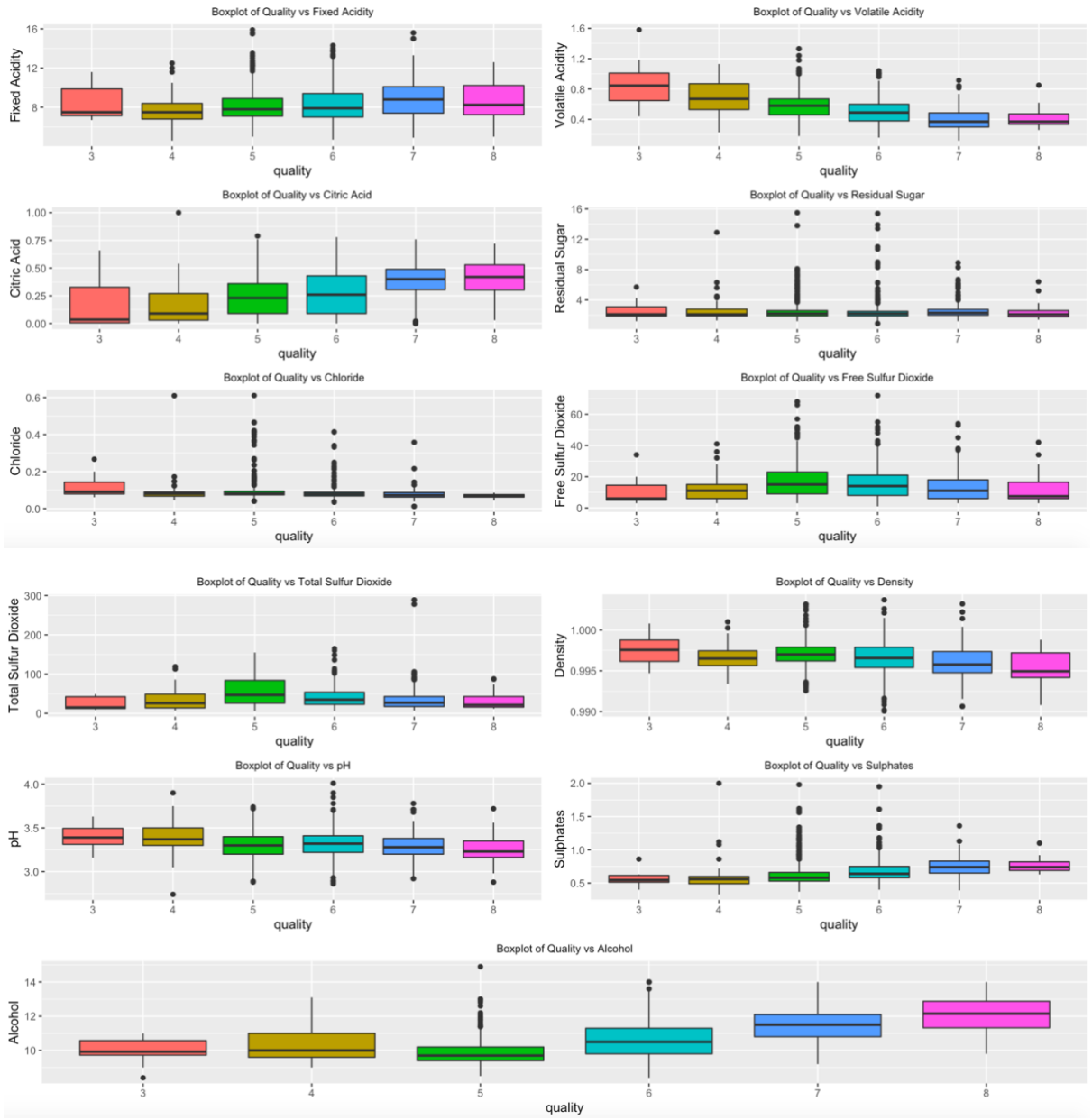
The density plot shows us the distribution of all the predictors of red wine. We can observe that most of the variables are positively skewed such as the residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, alcohol, and sulphates. The density and pH plots are observed to be normally distributed. In addition, we can also observe the distribution of volatile acidity and citric acid to be bimodal with multiple peaks. Volatile acidity has two peaks at approximately

0.4 and 0.6 while citric acid has peaks at about 0.05, 0.2, 0.5 and almost no data at higher values of the variable. Moreover, for the citric acid plot, a positively skewed long-tailed pattern can be seen with possible outliers in the higher range.

```
> g1 <- ggplot(raw,aes(factor(quality), fixed.acidity, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Fixed Acidity", title = "Boxplot of Quality vs Fixed Acidity")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g2 <- ggplot(raw,aes(factor(quality), volatile.acidity, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Volatile Acidity", title = "Boxplot of Quality vs Volatile Acidity")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g3 <- ggplot(raw,aes(factor(quality), citric.acid, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Citric Acid", title = "Boxplot of Quality vs Citric Acid")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g4 <- ggplot(raw,aes(factor(quality), residual.sugar, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Residual Sugar", title = "Boxplot of Quality vs Residual Sugar")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g5 <- ggplot(raw,aes(factor(quality), chlorides, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Chloride", title = "Boxplot of Quality vs Chloride")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
```

```
> g6 <- ggplot(raw,aes(factor(quality), free.sulfur.dioxide, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Free Sulfur Dioxide", title = "Boxplot of Quality vs Free Sulfur Dioxide")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g7 <- ggplot(raw,aes(factor(quality), total.sulfur.dioxide, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Total Sulfur Dioxide", title = "Boxplot of Quality vs Total Sulfur Dioxide")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g8 <- ggplot(raw,aes(factor(quality), density, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Density", title = "Boxplot of Quality vs Density")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g9 <- ggplot(raw,aes(factor(quality), pH, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "pH", title = "Boxplot of Quality vs pH")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g10 <- ggplot(raw,aes(factor(quality), sulphates, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Sulphates", title = "Boxplot of Quality vs Sulphates")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
> g11 <- ggplot(raw,aes(factor(quality), alcohol, fill = factor(quality)))+
+   geom_boxplot()+
+   labs(x = "quality", y = "Alcohol", title = "Boxplot of Quality vs Alcohol")+
+   theme(legend.position = 'none', plot.title = element_text(size = 9, hjust = 0.5))
>
```

```
> first_row2 = plot_grid(g1,g2,nrow =1)
> second_row2 = plot_grid(g3,g4, nrow = 1)
> third_row2 = plot_grid(g5,g6, nrow = 1)
> forth_row2 = plot_grid(g7,g8, nrow = 1)
> fifth_row = plot_grid(g9,g10, nrow = 1)
> sixth_row = plot_grid(g11, nrow = 1)
> plot_grid(first_row2,second_row2,third_row2, nrow = 3,ncol =1)
> plot_grid(forth_row2,fifth_row,sixth_row,nrow = 3,ncol =1)
>
```



We can clearly see that all the variables have outliers that have to be treated before proceeding with resampling techniques and machine learning models.

3.3 Data preprocessing

This section is the initial phase of machine learning where we have prepared the data before testing in with the proposed models. This part is important and requires to be performed systematically to produce accurate results.

1) Data cleaning

After loading the data, we first look for missing values present. As the dataset was initially clean, for the purpose of this project, we have introduced missing values using the prodNA function from the missForest package. We set the proportion of missing values to be 5%.

```
library(missForest)
library(DataExplorer)
set.seed(1111)
data <- prodNA(data, noNA = 0.05)
plot_missing (data)
colSums(sapply(data,is.na))
```

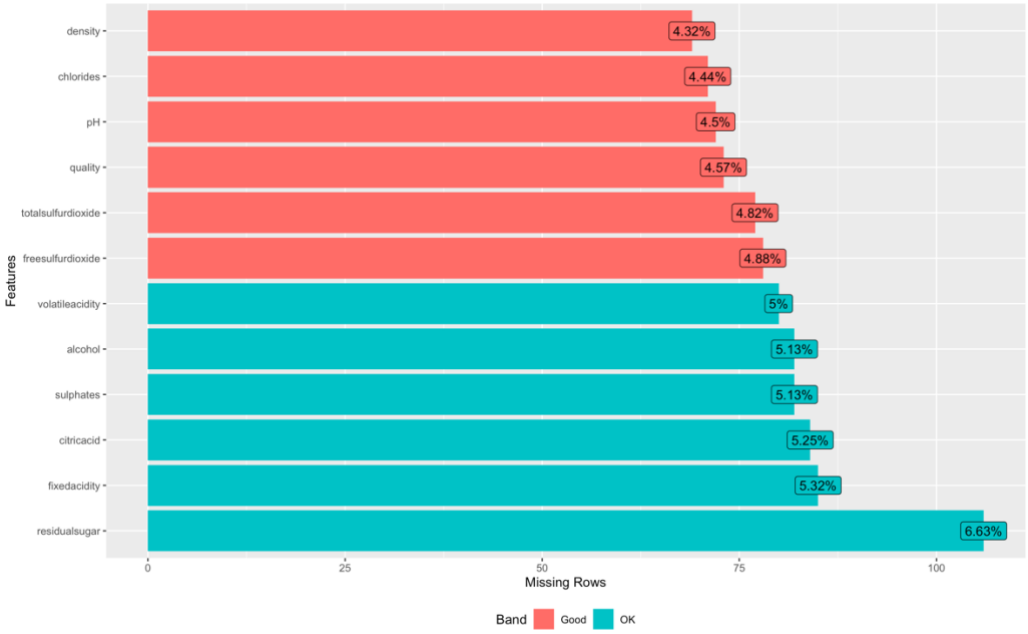


Figure 1: plot frequency of missing values for each feature

```
> colSums(sapply(data,is.na))
      fixedacidity    volatileacidity
                85                 80
       citricacid      residualsugar
                84                106
         chlorides  freesulfurdioxide
                71                 78
totalsulfurdioxide            density
                77                 69
                pH          sulphates
                72                 82
           alcohol            quality
                82                 73
```

The above function tells us the number of missing values in each variable that needs to be imputed. Missing values are obstacles in predictive modeling. Although it would be easier to delete those missing values, it is not recommended as it leads to loss of information. Hence, to impute the data, we have utilized the MICE (Multivariate Imputation via Chained Equations) package, a frequently used package in R, which produces numerous imputations instead of only one imputation, such as mean or median to take care of uncertainty in missing values. Missing data in MICE are assumed to be Missing at Random, meaning the chances of a value being missing is dependent only on the observed value and can be predicting using them. Various methods such as the PMM (predictive Mean Matching) for numerical variables, logreg (Logistic Regression) for binary variables, and polyreg (Bayesian polytomous regression) for factorial variables are being implemented by the package.

```
#Impute
library(mice)
impute <-mice(data,m=3,seed =123)
print(impute)
```

'm' here symbolizes 3 imputed data sets. After analyzing the imputations generated and compared with the median value of each variable, we have selected the 3rd imputation to be the best using the complete () function.

```
#complete data
data_impute <-complete(impute,3)
```

2) Data Resampling

For this project, we have proposed to implement two resampling techniques that could overcome the imbalanced dataset, which are the downSample and upSample techniques. When it comes to classification, an imbalance in the observed class can have a significant negative effect during the model fitting as the trained model would be highly influenced by the majority class, which in this case is the 'poor' wine quality. This can cause an increase in sensitivity rate but a decrease in specificity rate.

a) Down-sample

```
set.seed(123)
data_downsampled <- data_normalized
data_downsampled <- downSample(x = data_downsampled[,-12], y = data_downsampled$quality, list = F, yname = "quality")
inspect1<-inspect_cat(data_downsampled)
show_plot(inspect1)
prop.table(table(data_downsampled$quality))
```

Figure 2: Proportion of the dataset after downsampling

The down-sample technique randomly sample the dataset so that all classes have equal frequency as the minority class.

b) Up-sample

```
set.seed(1234)
data_upsampled <- data_normalized
data_upsampled <- upSample(x = data_upsampled[,-12], y = data_upsampled$quality, list = F, yname = "quality")
inspect1<-inspect_cat(data_upsampled)
show_plot(inspect1)
prop.table(table(data_upsampled$quality))
```

The up-sampling method augments the size of the minority class for the classes to be equal by sampling with replacement.

3) Data scaling

```
# normalize function - each variable can be seen with different ranges
normalize <- function(x){
  return (
    (x - min(x))/(max(x) - min(x))
  )}

# normalize our data
data_normalized <- data_impute
data_normalized[,-12] <- sapply(data_normalized[,-12], normalize)
summary(data_normalized)
```

As the initial data can be seen with different ranges, thus, we have normalized the data. The variables do not contribute fairly when they are being analyzed at different scales. Hence, a higher emphasis is being put on variables with larger ranges. By normalizing the variables, each variable can contribute equally to the analysis. We have implemented the min-max normalization for this project to normalize the data. The formula for the min-max normalization is:

$$(X - \min(X))/(\max(X) - \min(X))$$

For each value of a variable, we simply find the distance a certain value is from the minimum value and divide it by the range. Notice that each column (figure 4) has values ranging from 0 to 1.

```
> summary(data)
  fixedacidity    volatileacidity    citricacid     residualsugar      chlorides       freesulfurdioxide
 Min.   : 4.600   Min.   :0.1200   Min.   :0.0000   Min.   : 0.900   Min.   :0.01200   Min.   : 1.00
 1st Qu.: 7.100   1st Qu.:0.3900   1st Qu.:0.0900   1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00
 Median : 7.900   Median :0.5200   Median :0.2600   Median : 2.200   Median :0.07900   Median :14.00
 Mean   : 8.323   Mean   :0.5279   Mean   :0.2721   Mean   : 2.544   Mean   :0.08732   Mean   :15.91
 3rd Qu.: 9.200   3rd Qu.:0.6400   3rd Qu.:0.4300   3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:22.00
 Max.   :15.900   Max.   :1.5800   Max.   :1.0000   Max.   :15.500   Max.   :0.61000   Max.   :72.00
 NA's   :85       NA's   :80       NA's   :84       NA's   :106      NA's   :71        NA's   :78
 totalsulfurdioxide    density             pH          sulphates       alcohol       quality
 Min.   :  6.00     Min.   :0.9901   Min.   :2.74   Min.   :0.330   Min.   : 8.40   0   :1315
 1st Qu.: 22.00     1st Qu.:0.9956   1st Qu.:3.21   1st Qu.:0.550   1st Qu.: 9.50   1   : 211
 Median : 38.00     Median :0.9967   Median :3.31   Median :0.620   Median :10.20   NA's:  73
 Mean   : 46.37     Mean   :0.9967   Mean   :3.31   Mean   :0.657   Mean   :10.43
 3rd Qu.: 62.00     3rd Qu.:0.9978   3rd Qu.:3.40   3rd Qu.:0.730   3rd Qu.:11.10
 Max.   :289.00     Max.   :1.0037   Max.   :4.01   Max.   :2.000   Max.   :14.90
 NA's   :77         NA's   :69       NA's   :72     NA's   :82      NA's   :82
```

Figure 3: summary of data before normalization

```
> summary(data_normalized)
  fixedacidity    volatileacidity    citricacid     residualsugar       chlorides       freesulfurdioxide
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
 1st Qu.:0.2212   1st Qu.:0.1849   1st Qu.:0.0900   1st Qu.:0.06849   1st Qu.:0.09699   1st Qu.:0.08451
 Median :0.2920   Median :0.2740   Median :0.2600   Median :0.08904   Median :0.11204   Median :0.18310
 Mean   :0.3299   Mean   :0.2803   Mean   :0.2716   Mean   :0.11253   Mean   :0.12547   Mean   :0.20913
 3rd Qu.:0.4071   3rd Qu.:0.3562   3rd Qu.:0.4300   3rd Qu.:0.11644   3rd Qu.:0.13043   3rd Qu.:0.29577
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
 totalsulfurdioxide    density             pH           sulphates         alcohol       quality
 Min.   :0.00000    Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   0:1380
 1st Qu.:0.05654    1st Qu.:0.4060   1st Qu.:0.3701   1st Qu.:0.1317   1st Qu.:0.1692   1: 219
 Median :0.10954    Median :0.4868   Median :0.4488   Median :0.1737   Median :0.2769
 Mean   :0.14256    Mean   :0.4898   Mean   :0.4494   Mean   :0.1963   Mean   :0.3125
 3rd Qu.:0.19788    3rd Qu.:0.5675   3rd Qu.:0.5197   3rd Qu.:0.2395   3rd Qu.:0.4154
 Max.   :1.00000    Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Figure 4: Summary of data after normalization

4) Data Splitting

In this section, we have partitioned the data into train and test sets with a 70-30 ratio respectively. The training set is where we implement our machine learning models on while the test set is where we evaluate the performance of our models on. We have split the data to ensure that our models do not overlearn the correlation of data it is trained on. We have prepared 3 types of data, which are without resampling, over-sampling, and under-sampling. We have successfully implemented these datasets in the models which will be evaluated in the next sections.

```
# Data Partition - Downsample
set.seed(111)
ind_downsample <- sample(2,nrow(data_downsampled), replace = T, prob = c(0.7,0.3))
train_downsample <- data_downsampled[ind_downsample==1,]
test_downsample <- data_downsampled[ind_downsample==2,]
```

```
# Data Partition - Upsample
set.seed(112)
ind_upsample <- sample(2,nrow(data_upsampled), replace = T, prob = c(0.7,0.3))
train_upsample <- data_upsampled[ind_upsample==1,]
test_upsample <- data_upsampled[ind_upsample==2,]
```

```
# Data Partition - No resampling
set.seed(113)
ind <- sample(2,nrow(data_normalized), replace = T, prob = c(0.7,0.3))
train <- data_normalized[ind==1,]
test <- data_normalized[ind==2,]
```

5) Cross- validation

```
#cross-validation
control <- trainControl(
  method = "cv",
  number = 10)
```

Cross-validation is usually used in machine learning to approximate the skill of a machine learning model on unseen data. It typically results in a less biased estimate of the model skill as it makes sure that every observation from the dataset is given a chance to appear in the training and test set. For this method, we have set the value of k to be 10.

Table 1 shows the models and their dataset to be implemented in the next section.

Table 4: Models and dataset used

No	Experiment	Used Dataset		Model
		Train set	Test set	
Logistic Regression				
1.	No resampling	train	test	LR4
2.	Down-sample	train_downsample	test_downsample	LR4_ds
3.	Up-sample	train_upsample	test_upsample	LR3_us
Naïve Bayes				
1.	No resampling	train	test	naïve
2.	Down-sample	train_downsample	test_downsample	naïve_downsampled
3.	Tune-downsample	train_downsample	test_downsample	naïve_tuned_ds
4.	Up-sample	train_upsample	test_upsample	naïve_upsampled
5.	Tune- upsample	train_upsample	test_upsample	naïve_tuned-us
Random Forest				
1.	No resampling	train	test	rf
2.	Down-sample	train_downsample	test_downsample	rf_downsample
3.	Tune-downsample	train_downsample	test_downsample	rf_downsampled_tuned
3.	Up-sample	train_upsample	test_upsample	rf_upsample

3.4 Model Implementation

In this experiment, we have chosen three machine learning models, which are logistic regression, Naïve Bayes, and random forest.

Logistic Regression – No resampling

Logistic regression is considered to be a linear method with predictions modified using the logistic function. It has an S-shaped curve that can accommodate any real-valued number and designate it into values between 0 and 1. If the output produced by the functions is above, it gets included in classification '1'. However, if the output value is less than 0.5, is it classified as '0'. For the first experiment, we have implemented the model without resampling, followed by down-sampling and finally with the up-sampling technique.

```
#Model
LR <- glm(formula = quality ~ ., data = train, family = binomial())
summary(LR)


Call:
glm(formula = quality ~ ., family = binomial(), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7548  -0.6069  -0.1327   0.6719   2.5971

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        -0.32432    1.46690  -0.221 0.825021
fixedacidity        1.13159    2.18582   0.518 0.604671
volatileacidity    -7.47282    2.17844  -3.430 0.000603 ***
citricacid          0.46746    1.52228   0.307 0.758787
residualsugar       3.76654    1.71511   2.196 0.028085 *
chlorides          -3.77874    3.07120  -1.230 0.218556
freesulfurdioxide   0.07432    1.43978   0.052 0.958833
totalsulfurdioxide -4.65063    1.84881  -2.515 0.011888 *
density            -2.81360    1.98050  -1.421 0.155418
pH                 -1.36462    2.26657  -0.602 0.547131
sulphates           8.17457    1.80524   4.528 5.95e-06 ***
alcohol             5.36303    1.37227   3.908 9.30e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

'Pr(>|z|' represents the p-value, which corresponds to the z-statistic. Smaller p values (below 0.05) are considered more significant. We have used the backward selection technique, which includes all feature in the model, iteratively eliminate the least contributing features, and end when we have successfully generated a model with statistically significant features.

```
LR2 <- glm(formula = quality ~ volatileacidity + residualsugar + chlorides +
             totalsulfurdioxide + density + pH + sulphates + alcohol,
           data = train, family = binomial())
summary(LR2)
```

```
Call:
glm(formula = quality ~ volatileacidity + residualsugar + chlorides +
    totalsulfurdioxide + density + pH + sulphates + alcohol,
    family = binomial(), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7535  -0.6070  -0.1280   0.6621   2.6288

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)         0.2376     1.2713   0.187 0.851737
volatileacidity    -8.0042     1.7765  -4.506 6.62e-06 ***
residualsugar       3.7828     1.7921   2.111 0.034794 *
chlorides          -4.1669     2.8765  -1.449 0.147450
totalsulfurdioxide -4.6567     1.3751  -3.386 0.000708 ***
density            -1.7187     1.3510  -1.272 0.203311
pH                 -2.6166     1.5062  -1.737 0.082354 .
sulphates           8.3279     1.8088   4.604 4.14e-06 ***
alcohol             5.8106     1.2321   4.716 2.40e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
LR3 <- glm(formula = quality ~  volatileacidity + residualsugar +
             totalsulfurdioxide + pH + sulphates + alcohol,
           data = train, family = binomial())
summary(LR3)
```

```
Call:
glm(formula = quality ~ volatileacidity + residualsugar + totalsulfurdioxide +
    pH + sulphates + alcohol, family = binomial(), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7688  -0.5853  -0.1440   0.6706   2.6923

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.3463     0.8103  -1.661  0.09662 .
volatileacidity     -8.5208     1.7538  -4.858 1.18e-06 ***
residualsugar        2.7306     1.7242   1.584  0.11327
totalsulfurdioxide  -3.9850     1.3132  -3.035  0.00241 **
pH                  -1.7231     1.4086  -1.223  0.22122
sulphates            6.9071     1.5620   4.422 9.78e-06 ***
alcohol              6.8219     1.0875   6.273 3.54e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
LR4 <- glm(formula = quality ~  volatileacidity +
             totalsulfurdioxide + sulphates + alcohol,
           data = train, family = binomial())
summary(LR4)

Call:
glm(formula = quality ~ volatileacidity + totalsulfurdioxide +
    sulphates + alcohol, family = binomial(), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7378  -0.6525  -0.1482   0.6707   2.7315

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        -1.6203     0.6823  -2.375   0.0176 *
volatileacidity    -9.2579     1.6432  -5.634 1.76e-08 ***
totalsulfurdioxide -3.1456     1.2620  -2.492   0.0127 *
sulphates           6.6979     1.5085   4.440 8.99e-06 ***
alcohol             6.7190     1.0251   6.555 5.58e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 432.41  on 311  degrees of freedom
Residual deviance: 269.74  on 307  degrees of freedom
AIC: 279.74

Number of Fisher Scoring iterations: 5
```

Each one-unit change in volatile acidity will decrease the log odds of getting excellent wine by 9.258 and its p-value indicates that it is very significant in determining excellent wine. In contrast, each unit increase in alcohol will increase the log odds of targeting excellent wine by 6.719 and its p-value also shows that it is highly significant in determining excellent wine. The difference between the null and residual deviance portrays a good fit model, where the more prominent the difference is, the better the model.

```r
#Prediction on Train Set
pred_lr_train <- predict(LR4, train, type = 'response')
pred_lr_train1 <- ifelse(pred_lr_train>0.5,1,0)
tab_train <- table(Predicted = pred_lr_train1, Actual =  as.factor(train$quality))
confusionMatrix(tab_train, positive = '1')
```

```
Confusion Matrix and Statistics

         Actual
Predicted   0   1
        0 125  30
        1  34 123

               Accuracy : 0.7949
                 95% CI : (0.7458, 0.8383)
    No Information Rate : 0.5096
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5898

 Mcnemar's Test P-Value : 0.7077

            Sensitivity : 0.8039
            Specificity : 0.7862
         Pos Pred Value : 0.7834
         Neg Pred Value : 0.8065
             Prevalence : 0.4904
         Detection Rate : 0.3942
   Detection Prevalence : 0.5032
      Balanced Accuracy : 0.7950

       'Positive' Class : 1
```

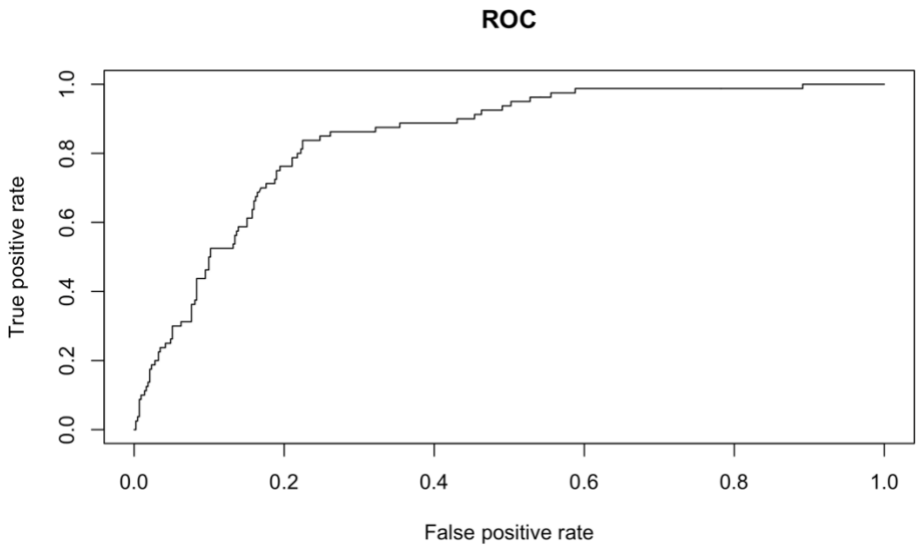
```r
#Prediction on Test Set
pred_lr_test <- predict(LR4, test, type = 'response')
pred_lr_test1 <- ifelse(pred_lr_test>0.5,1,0)
tab_test <- table(Predicted = pred_lr_test1, Actual =  as.factor(test$quality))
confusionMatrix(tab_test, positive = '1')
```

```
Confusion Matrix and Statistics

         Actual
Predicted   0   1
        0 337  16
        1  95  64

               Accuracy : 0.7832
                 95% CI : (0.7449, 0.8182)
    No Information Rate : 0.8438
    P-Value [Acc > NIR] : 0.9999

                  Kappa : 0.4137

 Mcnemar's Test P-Value : 1.327e-13

            Sensitivity : 0.8000
            Specificity : 0.7801
         Pos Pred Value : 0.4025
         Neg Pred Value : 0.9547
             Prevalence : 0.1562
         Detection Rate : 0.1250
   Detection Prevalence : 0.3105
      Balanced Accuracy : 0.7900

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude a good fit for the model as the difference in accuracy between the two datasets is small. An accuracy of 79.5% was achieved on the training set and 78.32% on the test set.

```
#ROC
library(ROCR)
pred_LR <- predict(LR4,test, type = 'response')
ROC <- prediction(pred_LR, labels = test$quality )
plot(performance(ROC,"tpr", "fpr"),
     main = "ROC")
```

The receiver Operating Characteristic (ROC) curve is a known method to visualize the trade-offs between sensitivity and specificity in a binary classifier.



One way to quantify how well the logistic regression model does at classifying data is to calculate AUC (Area Under Curve). The closer the value is to 1, the better the model is. We can conclude that there is an 84.31% probability of predicting whether or not a wine is poor or excellent in quality.

```
# AUC
auc_ROCR_n <- performance(ROC, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.8431279
```

Logistic Regression – Down-sample

```
LR_ds <- glm(formula = quality ~ ., data = train_downsample, family = binomial())
summary(LR_ds)

Call:
glm(formula = quality ~ ., family = binomial(), data = train_downsample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7548  -0.6069  -0.1327   0.6719   2.5971

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        -0.32432    1.46690  -0.221 0.825021
fixedacidity        1.13159    2.18582   0.518 0.604671
volatileacidity    -7.47282    2.17844  -3.430 0.000603 ***
citricacid          0.46746    1.52228   0.307 0.758787
residualsugar       3.76654    1.71511   2.196 0.028085 *
chlorides          -3.77874    3.07120  -1.230 0.218556
freesulfurdioxide   0.07432    1.43978   0.052 0.958833
totalsulfurdioxide -4.65063    1.84881  -2.515 0.011888 *
density            -2.81360    1.98050  -1.421 0.155418
pH                 -1.36462    2.26657  -0.602 0.547131
sulphates           8.17457    1.80524   4.528 5.95e-06 ***
alcohol             5.36303    1.37227   3.908 9.30e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
LR2_ds <- glm(formula = quality ~ fixedacidity + volatileacidity + residualsugar + chlorides +
            totalsulfurdioxide + density + pH + sulphates + alcohol,
          data = train_downsample, family = binomial())
summary(LR2_ds)

Call:
glm(formula = quality ~ fixedacidity + volatileacidity + residualsugar +
    chlorides + totalsulfurdioxide + density + pH + sulphates +
    alcohol, family = binomial(), data = train_downsample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7533  -0.6061  -0.1296   0.6716   2.6239

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        -0.2824     1.4603  -0.193  0.84667
fixedacidity        1.4245     1.9639   0.725  0.46824
volatileacidity    -7.8583     1.7859  -4.400 1.08e-05 ***
residualsugar       3.8315     1.7072   2.244  0.02482 *
chlorides          -3.5822     2.9809  -1.202  0.22947
totalsulfurdioxide -4.4908     1.3897  -3.231  0.00123 **
density            -2.7729     1.9729  -1.406  0.15986
pH                 -1.4261     2.2235  -0.641  0.52127
sulphates           8.2303     1.7950   4.585 4.54e-06 ***
alcohol             5.5007     1.2951   4.247 2.16e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
LR3_ds <- glm(formula = quality ~  volatileacidity + residualsugar + chlorides +
            totalsulfurdioxide + density + sulphates + alcohol,
          data = train_downsample, family = binomial())
summary(LR3_ds)

Call:
glm(formula = quality ~ volatileacidity + residualsugar + chlorides +
    totalsulfurdioxide + density + sulphates + alcohol, family = binomial(),
    data = train_downsample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7000  -0.6230  -0.1373   0.6981   2.7175

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)         -0.9604     1.0651  -0.902  0.36723
volatileacidity     -9.2024     1.6736  -5.499 3.83e-08 ***
residualsugar        3.9972     1.8102   2.208  0.02723 *
chlorides           -3.2571     2.9001  -1.123  0.26139
totalsulfurdioxide  -4.4229     1.3727  -3.222  0.00127 **
density             -1.0929     1.3069  -0.836  0.40304
sulphates            8.1102     1.7602   4.608 4.07e-06 ***
alcohol              5.6530     1.2147   4.654 3.26e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

LR4_ds <- glm(formula = quality ~  volatileacidity + residualsugar +
               totalsulfurdioxide + sulphates + alcohol,
             data = train_downsample, family = binomial())
summary(LR4_ds)

Call:
glm(formula = quality ~ volatileacidity + residualsugar + totalsulfurdioxide +

    sulphates + alcohol, family = binomial(), data = train_downsample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7240  -0.6070  -0.1431   0.6946   2.7452

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.8439     0.7008  -2.631  0.00851 **
volatileacidity     -9.3312     1.6449  -5.673 1.41e-08 ***
residualsugar        3.1192     1.6992   1.836  0.06641 .
totalsulfurdioxide  -3.9886     1.3263  -3.007  0.00264 **
sulphates            7.0699     1.5353   4.605 4.13e-06 ***
alcohol              6.4544     1.0314   6.258 3.90e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

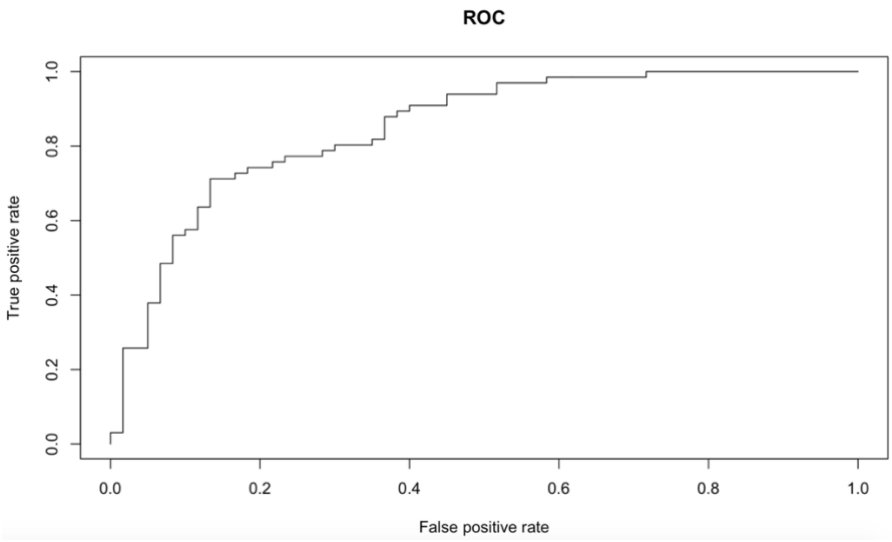
Each one-unit change in volatile acidity will decrease the log odds of getting excellent wine by 9.331 and its p-value indicates that it is very significant in determining excellent wine. In contrast, each unit increase in alcohol will increase the log odds of targeting excellent wine by 6.454 and its p-value also shows that it is highly significant in determining excellent wine.

Prediction

```
#Prediction on Train Set
pred_lr_train_ds <- predict(LR4_ds, train_downsample, type = 'response')
pred_lr_train_ds1 <- ifelse(pred_lr_train_ds>0.5,1,0)
tab_train_ds <- table(Predicted = pred_lr_train_ds1, Actual =  as.factor(train_downsample$quality))
confusionMatrix(tab_train_ds, positive = '1')

Confusion Matrix and Statistics

         Actual
Predicted   0   1
        0 129  32
        1  30 121

               Accuracy : 0.8013
                 95% CI : (0.7526, 0.8441)
    No Information Rate : 0.5096
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6023

 Mcnemar's Test P-Value : 0.8989

            Sensitivity : 0.7908
            Specificity : 0.8113
         Pos Pred Value : 0.8013
         Neg Pred Value : 0.8012
             Prevalence : 0.4904
         Detection Rate : 0.3878
   Detection Prevalence : 0.4840
      Balanced Accuracy : 0.8011

       'Positive' Class : 1

#Prediction on Test Set
pred_lr_test_ds <- predict(LR4_ds, test_downsample, type = 'response')
pred_lr_test1_ds <- ifelse(pred_lr_test_ds>0.5,1,0)
tab_test_ds <- table(Predicted = pred_lr_test1_ds, Actual =  as.factor(test_downsample$quality))
confusionMatrix(tab_test_ds, positive = '1')

Confusion Matrix and Statistics

         Actual
Predicted  0  1
        0 48 17
        1 12 49

               Accuracy : 0.7698
                 95% CI : (0.6865, 0.8401)
    No Information Rate : 0.5238
    P-Value [Acc > NIR] : 1.066e-08

                  Kappa : 0.5404

 Mcnemar's Test P-Value : 0.4576

            Sensitivity : 0.7424
            Specificity : 0.8000
         Pos Pred Value : 0.8033
         Neg Pred Value : 0.7385
             Prevalence : 0.5238
         Detection Rate : 0.3889
   Detection Prevalence : 0.4841
      Balanced Accuracy : 0.7712

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude a good fit for the model as the difference in accuracy between the two datasets is small. An accuracy of 80.13% was achieved on the training set and 76.98% on the test set. It is interesting to note that, down-sampling the dataset resulted in a lower value of accuracy on the test set.

```
#ROC
library(ROCR)
pred_LR_ds <- predict(LR4_ds,test_downsample, type = 'response')
ROC_ds <- prediction(pred_LR_ds, labels = test_downsample$quality )
plot(performance(ROC_ds,"tpr", "fpr"),
     main = "ROC")
```




```
# AUC
auc_ROCR_n <- performance(ROC_ds, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.8487374
```

We can conclude that there is 84.87% probability in predicting whether or not a wine is poor or excellent. This is only slightly higher than the AUC value without sampling.

Logistic Regression – Up-sample

```
#Model training
LR_us <- glm(formula = quality ~ ., data = train_upsample, family = binomial())
summary(LR_us)

Call:
glm(formula = quality ~ ., family = binomial(), data = train_upsample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1687  -0.5498  -0.0915   0.6581   2.2522

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.0868     0.5985  -1.816   0.0694 .
fixedacidity         3.9910     0.8961   4.454 8.44e-06 ***
volatileacidity     -4.3242     0.8113  -5.330 9.83e-08 ***
citricacid          -0.4450     0.6304  -0.706   0.4803
residualsugar        6.7701     0.8968   7.549 4.37e-14 ***
chlorides           -5.1547     1.0718  -4.809 1.51e-06 ***
freesulfurdioxide   -0.3562     0.6087  -0.585   0.5584
totalsulfurdioxide  -4.4330     0.7721  -5.742 9.38e-09 ***
density             -6.0429     0.8405  -7.190 6.49e-13 ***
pH                  -0.2717     0.8859  -0.307   0.7591
sulphates           10.5999     0.8853  11.973  < 2e-16 ***
alcohol              4.9127     0.5708   8.606  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2726.8  on 1966  degrees of freedom
Residual deviance: 1581.7  on 1955  degrees of freedom
AIC: 1605.7

Number of Fisher Scoring iterations: 5

LR2_us <- glm(formula = quality ~ fixedacidity + volatileacidity + citricacid + residualsugar + chlorides +
                                  totalsulfurdioxide + density + sulphates + alcohol,
                                  data = train_upsample, family = binomial())
summary(LR2_us)

Call:
glm(formula = quality ~ fixedacidity + volatileacidity + citricacid +
    residualsugar + chlorides + totalsulfurdioxide + density +
    sulphates + alcohol, family = binomial(), data = train_upsample)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-3.11044  -0.54227  -0.08835   0.66044   2.26333

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.2708     0.4221  -3.011  0.00261 **
fixedacidity         4.1590     0.6926   6.005 1.91e-09 ***
volatileacidity     -4.2533     0.7975  -5.333 9.66e-08 ***
citricacid          -0.3481     0.6139  -0.567  0.57072
residualsugar        6.8961     0.8779   7.855 4.00e-15 ***
chlorides           -5.0905     1.0448  -4.872 1.10e-06 ***
totalsulfurdioxide  -4.6875     0.5798  -8.085 6.23e-16 ***
density             -6.1799     0.7642  -8.086 6.14e-16 ***
sulphates           10.5443     0.8775  12.017  < 2e-16 ***
alcohol              4.8392     0.5380   8.995  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2726.8  on 1966  degrees of freedom
Residual deviance: 1582.2  on 1957  degrees of freedom
AIC: 1602.2

Number of Fisher Scoring iterations: 5
```

```
LR3_us <- glm(formula = quality ~  fixedacidity + volatileacidity + residualsugar + chlorides +
                                   totalsulfurdioxide + density + sulphates + alcohol,
                                   data = train_upsample, family = binomial())
summary(LR3_us)

Call:
glm(formula = quality ~ fixedacidity + volatileacidity + residualsugar +
    chlorides + totalsulfurdioxide + density + sulphates + alcohol,
    family = binomial(), data = train_upsample)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-3.10010  -0.54483  -0.08972   0.65910   2.26440

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        -1.2712     0.4222  -3.011   0.0026 **
fixedacidity        3.9213     0.5496   7.135 9.70e-13 ***
volatileacidity    -3.9758     0.6262  -6.349 2.16e-10 ***
residualsugar       6.8495     0.8731   7.845 4.32e-15 ***
chlorides          -5.3075     0.9757  -5.440 5.34e-08 ***
totalsulfurdioxide -4.7770     0.5583  -8.557  < 2e-16 ***
density            -6.2059     0.7623  -8.141 3.92e-16 ***
sulphates          10.5391     0.8779  12.005  < 2e-16 ***
alcohol             4.7408     0.5084   9.326  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2726.8  on 1966  degrees of freedom
Residual deviance: 1582.5  on 1958  degrees of freedom
AIC: 1600.5

Number of Fisher Scoring iterations: 5
```

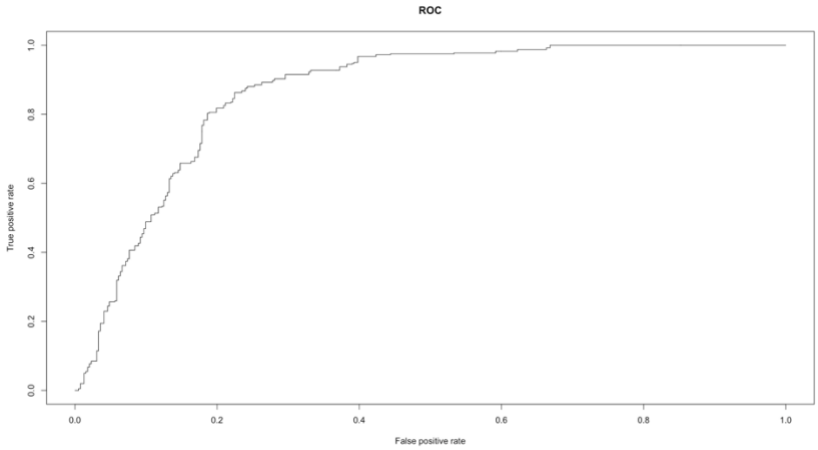
We can observe more variables being statistically significant when up-sampling was implemented. Furthermore, the standard errors are very low for all the variables, indicating they are significant in determining the target variables. Each one-unit change in volatile acidity will decrease the log odds of getting excellent wine by 3.976 and its p-value indicates that it is very significant in determining excellent wine. In contrast, each unit increase in alcohol will increase the log odds of targeting excellent wine by 4.740 and its p-value also indicates that it is very significant in determining excellent wine.

Prediction

```
#Prediction on Train Set
pred_lr_train_us <- predict(LR3_us, train_upsample, type = 'response')
pred_lr_train_us1 <- ifelse(pred_lr_train_us>0.5,1,0)
tab_train_us <- table(Predicted = pred_lr_train_us1, Actual =  as.factor(train_upsample$quality))
confusionMatrix(tab_train_us, positive = '1')

Confusion Matrix and Statistics

         Actual
Predicted   0   1
        0 790 168
        1 198 811

               Accuracy : 0.8139
                 95% CI : (0.796, 0.8309)
    No Information Rate : 0.5023
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6279

 Mcnemar's Test P-Value : 0.1296

            Sensitivity : 0.8284
            Specificity : 0.7996
         Pos Pred Value : 0.8038
         Neg Pred Value : 0.8246
             Prevalence : 0.4977
         Detection Rate : 0.4123
   Detection Prevalence : 0.5130
      Balanced Accuracy : 0.8140

       'Positive' Class : 1

#Prediction on Test Set
pred_lr_test_us <- predict(LR3_us, test_upsample, type = 'response')
pred_lr_test_us1 <- ifelse(pred_lr_test_us>0.5,1,0)
tab_test_us <- table(Predicted = pred_lr_test_us1, Actual =  as.factor(test_upsample$quality))
confusionMatrix(tab_test_us, positive = '1')

Confusion Matrix and Statistics

         Actual
Predicted   0   1
        0 308  67
        1  84 334

               Accuracy : 0.8096
                 95% CI : (0.7805, 0.8363)
    No Information Rate : 0.5057
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6189

 Mcnemar's Test P-Value : 0.1929

            Sensitivity : 0.8329
            Specificity : 0.7857
         Pos Pred Value : 0.7990
         Neg Pred Value : 0.8213
             Prevalence : 0.5057
         Detection Rate : 0.4212
   Detection Prevalence : 0.5271
      Balanced Accuracy : 0.8093

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude a good fit for the model as the difference in accuracy between the two datasets is small. An accuracy of 81.39% was achieved on the training set and 80.96% on the test set, which surpasses both datasets without resampling and down-sampling.

```
#ROC
library(ROCR)
pred_LR_us <- predict(LR3_us,test_upsample, type = 'response')
ROC_us <- prediction(pred_LR_us, labels = test_upsample$quality )
plot(performance(ROC_us,"tpr", "fpr"),
     main = "ROC")
```




```
# AUC
auc_ROCR_n <- performance(ROC_us, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n

[1] 0.8623785
```

We can conclude that there is 86.23% probability of predicting whether or not a wine is poor or excellent. The AUC value for up-sampling surpasses the down-sampling and without sampling dataset AUC values.

Naïve Bayes – No resampling

The Bayes theorem was named after Rev. Thomas Bayes, which functions on conditional probability (Saxena, 2017). Naïve Bayes classifier is a form of a statistical method based on the Bayes' theorem, which calculates the probability that a certain variable belonging to a certain class. It gets its name 'naïve' by assuming the variables being trained by the models are independent of each other (Prabhakaran, 2018). The Naïve Bayes considers the class with the highest probability to be the possible class (Saxena, 2017). Similar to the model implemented previously, we have tested using the no resampling method, down-sampling, and up-sampling techniques.

```
# Model
naive <- naiveBayes(train[-12],train$quality, laplace = 1,trControl = control)
```

Prediction

```
#Predicting train set
pred_naive_train <- predict(naive, train)
cm_naive_train <- confusionMatrix(pred_naive_train, train$quality, positive = "1")
cm_naive_train
```

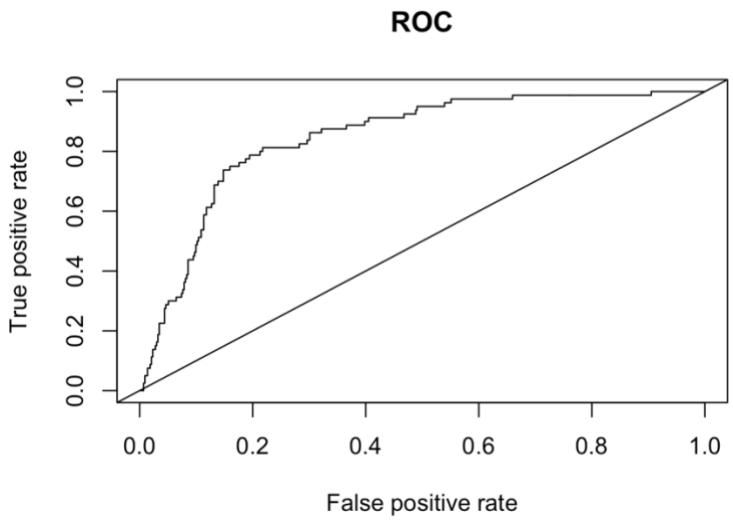
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 846  51
         1 102  88

               Accuracy : 0.8592
                 95% CI : (0.8371, 0.8794)
    No Information Rate : 0.8721
    P-Value [Acc > NIR] : 0.9048

                  Kappa : 0.4544

 Mcnemar's Test P-Value : 5.294e-05

            Sensitivity : 0.63309
            Specificity : 0.89241
         Pos Pred Value : 0.46316
         Neg Pred Value : 0.94314
             Prevalence : 0.12787
         Detection Rate : 0.08096
   Detection Prevalence : 0.17479
      Balanced Accuracy : 0.76275

       'Positive' Class : 1
```

```
#Predicting test set
pred_naive_test <- predict(naive, test)
cm_naive_test <- confusionMatrix(pred_naive_test, test$quality, positive = "1")
cm_naive_test
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 375  26
         1  57  54

               Accuracy : 0.8379
                 95% CI : (0.8031, 0.8688)
    No Information Rate : 0.8438
    P-Value [Acc > NIR] : 0.6690690

                  Kappa : 0.469

 Mcnemar's Test P-Value : 0.0009915

            Sensitivity : 0.6750
            Specificity : 0.8681
         Pos Pred Value : 0.4865
         Neg Pred Value : 0.9352
             Prevalence : 0.1562
         Detection Rate : 0.1055
   Detection Prevalence : 0.2168
      Balanced Accuracy : 0.7715

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude a good fit for the model as the difference in accuracy between the two datasets is small. An accuracy of 85.392% was achieved on the training set and 83.79% on the test set.

```
#ROC
library(ROCR)
pred_naive <- predict(naive,test, type = 'raw')[,2]
ROC <- prediction(pred_naive, labels = test$quality)
plot(performance(ROC,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```



```
# AUC
auc_ROCR_n <- performance(ROC, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.8457176
```

We can conclude that there is 84.57% probability in predicting whether or not a wine is poor or excellent.

Naïve Bayes – Down-sample

```
#model building
library(e1071)
naive_downsmpled <- naiveBayes(train_downsample[-12],train_downsample$quality, laplace = 1, trControl = control)
```

Prediction

```
#Predicting train set
pred_naive_ds_train <- predict(naive_downsmpled, train_downsample)
cm_naive_ds_train <- confusionMatrix(pred_naive_ds_train, train_downsample$quality, positive = "1")
cm_naive_ds_train
```

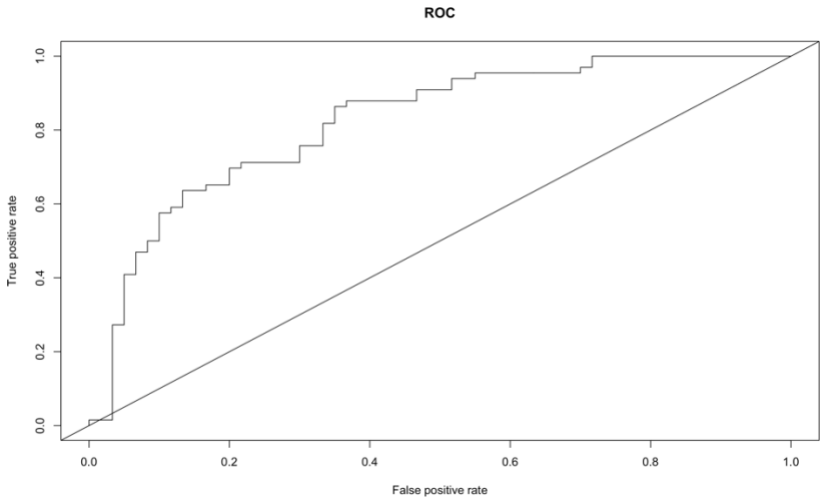
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 119  26
         1  40 127

               Accuracy : 0.7885
                 95% CI : (0.7389, 0.8325)
    No Information Rate : 0.5096
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5775

 Mcnemar's Test P-Value : 0.1096

            Sensitivity : 0.8301
            Specificity : 0.7484
         Pos Pred Value : 0.7605
         Neg Pred Value : 0.8207
             Prevalence : 0.4904
         Detection Rate : 0.4071
   Detection Prevalence : 0.5353
      Balanced Accuracy : 0.7892

       'Positive' Class : 1
```

```r
#Predicting test set
pred_naive_ds_test <- predict(naive_downsmpled, test_downsample)
cm_naive_ds_test <- confusionMatrix(pred_naive_ds_test, test_downsample$quality, positive = "1")
cm_naive_ds_test
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 44 19
         1 16 47

               Accuracy : 0.7222
                 95% CI : (0.6354, 0.7983)
    No Information Rate : 0.5238
    P-Value [Acc > NIR] : 4.213e-06

                  Kappa : 0.4444

 Mcnemar's Test P-Value : 0.7353

            Sensitivity : 0.7121
            Specificity : 0.7333
         Pos Pred Value : 0.7460
         Neg Pred Value : 0.6984
             Prevalence : 0.5238
         Detection Rate : 0.3730
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.7227

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude that the model is slightly overfitting as the training set accuracy is larger than the test set. An accuracy of 78.85% was achieved on the training set and 72.22% on the test set, which is lower than the values achieved without resampling. Overfitting is recognized as a modeling inaccuracy that presents bias to the model as it relates similarly to the dataset, thus, making it only applicable to its data set and almost inapplicable to other datasets. This may cause the failure in predicting future observations with significant accuracy values.

```r
#ROC
library(ROCR)
pred_naive_ds <- predict(naive_downsmpled,test_downsample, type = 'raw')[,2]
ROC_ds <- prediction(pred_naive_ds, labels = test_downsample$quality)
plot(performance(ROC_ds,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```




```
# AUC
auc_ROCR_n <- performance(ROC_ds, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.8219697
```

We can conclude that there is 82.2% probability in predicting whether or not a wine is poor or excellent.

Naïve Bayes – Up-sample

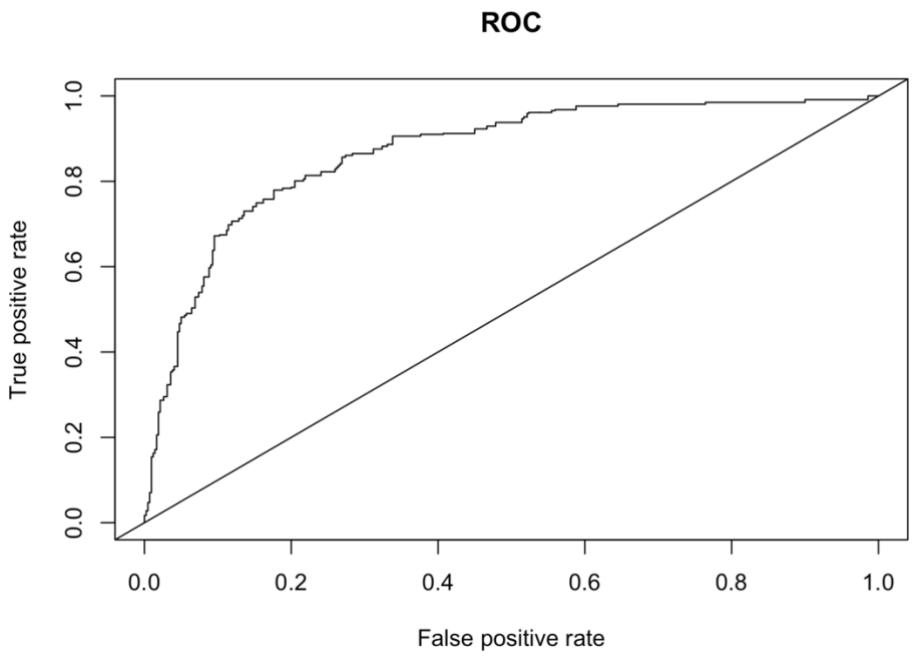
```
#model building
library(e1071)
naive_upsmpled <- naiveBayes(train_upsample[-12],train_upsample$quality, laplace = 1, trControl = control)

#Predicting train set
pred_naive_us_train <- predict(naive_upsmpled, train_upsample)
cm_naive_us_train <- confusionMatrix(pred_naive_us_train, train_upsample$quality, positive = "1")
cm_naive_us_train
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 719 173
         1 241 740

               Accuracy : 0.779
                 95% CI : (0.7595, 0.7976)
    No Information Rate : 0.5125
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5585

 Mcnemar's Test P-Value : 0.0009917

            Sensitivity : 0.8105
            Specificity : 0.7490
         Pos Pred Value : 0.7543
         Neg Pred Value : 0.8061
             Prevalence : 0.4875
         Detection Rate : 0.3951
   Detection Prevalence : 0.5238
      Balanced Accuracy : 0.7797

       'Positive' Class : 1
```

```
#Predicting test set
pred_naive_us_test <- predict(naive_upsmpled, test_upsample)
cm_naive_us_test <- confusionMatrix(pred_naive_us_test, test_upsample$quality, positive = "1")
cm_naive_us_test

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 334  93
         1  86 374

               Accuracy : 0.7982
                 95% CI : (0.7702, 0.8241)
    No Information Rate : 0.5265
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5956

 Mcnemar's Test P-Value : 0.6538

            Sensitivity : 0.8009
            Specificity : 0.7952
         Pos Pred Value : 0.8130
         Neg Pred Value : 0.7822
             Prevalence : 0.5265
         Detection Rate : 0.4216
   Detection Prevalence : 0.5186
      Balanced Accuracy : 0.7980

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude a good fit for the model as the difference in accuracy between the two datasets is small. An accuracy of 77.9% was achieved on the training set and 79.82% on the test set.

```
#ROC
pred_naive_us <- predict(naive_upsampled,test_upsample, type = 'raw')[,2]
ROC_us <- prediction(pred_naive_us, labels = test_upsample$quality)
plot(performance(ROC_us,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```




```
# AUC
auc_ROCR_n <- performance(ROC_us, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.868237
```

We can conclude that there is 86.82% probability in predicting whether or not a wine is poor or excellent.

Parameter Tuning- Upsample

To improve the readings of accuracy for the resampling methods, we have performed parameter tuning. The Naïve Bayes model can be tuned using 'usekernel' parameter that lets us utilize a kernel density approximation for continuous variables versus a Gaussian density approximation, 'adjust' permits the bandwidth of the kernel density to be adjusted and 'fL', which lets us include the Laplace smoother.

```
#tuning grid set up
search_grid <- expand.grid(
  usekernel = c(TRUE, FALSE),
  fL = 0:5,
  adjust = seq(0, 5, by = 1)
)
```

```
#Model tuned
library(caret)
x = train_upsample[,-12]
y = train_upsample$quality

naive_tuned_us <- train(x = x,y = y,method = "nb",trControl = control,tuneGrid = search_grid)
naive_tuned_us
```

Prediction

```
#Predicting train set tuned
pred_naive_train_tuned_us <- predict(naive_tuned_us , train_upsample)
cm_naive_train_tuned_us <- confusionMatrix(pred_naive_train_tuned_us, train_upsample$quality, positive = "1")
cm_naive_train_tuned_us
```

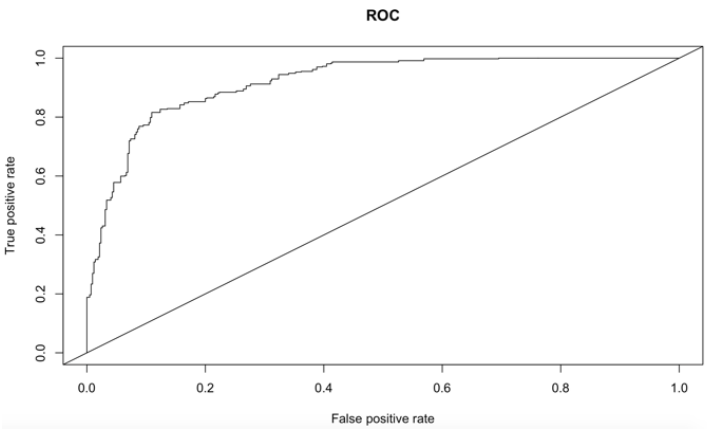
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 762 137
         1 198 776

               Accuracy : 0.8211
                 95% CI : (0.803, 0.8383)
    No Information Rate : 0.5125
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6426

 Mcnemar's Test P-Value : 0.001045

            Sensitivity : 0.8499
            Specificity : 0.7937
         Pos Pred Value : 0.7967
         Neg Pred Value : 0.8476
             Prevalence : 0.4875
         Detection Rate : 0.4143
   Detection Prevalence : 0.5200
      Balanced Accuracy : 0.8218

       'Positive' Class : 1
```

```
#Predicting test set tuned
pred_naive_test_tuned_us <- predict(naive_tuned_us, test_upsample)
cm_naive_test_tuned_us <- confusionMatrix(pred_naive_test_tuned_ds, test_upsample$quality, positive = "1")
cm_naive_test_tuned_us
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 339  69
         1  81 398

               Accuracy : 0.8309
                 95% CI : (0.8046, 0.855)
    No Information Rate : 0.5265
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6603

 Mcnemar's Test P-Value : 0.3691

            Sensitivity : 0.8522
            Specificity : 0.8071
         Pos Pred Value : 0.8309
         Neg Pred Value : 0.8309
             Prevalence : 0.5265
         Detection Rate : 0.4487
   Detection Prevalence : 0.5400
      Balanced Accuracy : 0.8297

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude a good fit for the model as the difference in accuracy between the two datasets is small. An accuracy of 82.11% was achieved on the training set and 83.09% on the test set. Clearly, tuning the parameters of the model helped improve the accuracy of the up-sampling model.

```
#ROC tuned
library(ROCR)
pred_naive_tuned_us <- predict(naive_tuned_us,test_upsample, type = 'prob')[,2]
ROC_naive_tuned_us <- prediction(pred_naive_tuned_us, labels = test_upsample$quality)
plot(performance(ROC_naive_tuned_us,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```




```
# AUC tuned
auc_ROCR_n <- performance(ROC_naive_tuned_us, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.9184358
```

An AUC value of 0.9184 was achieved after tuning, indicating a 91.84% of predicting the wine quality to be poor or excellent.

Parameter Tuning- Down-sample

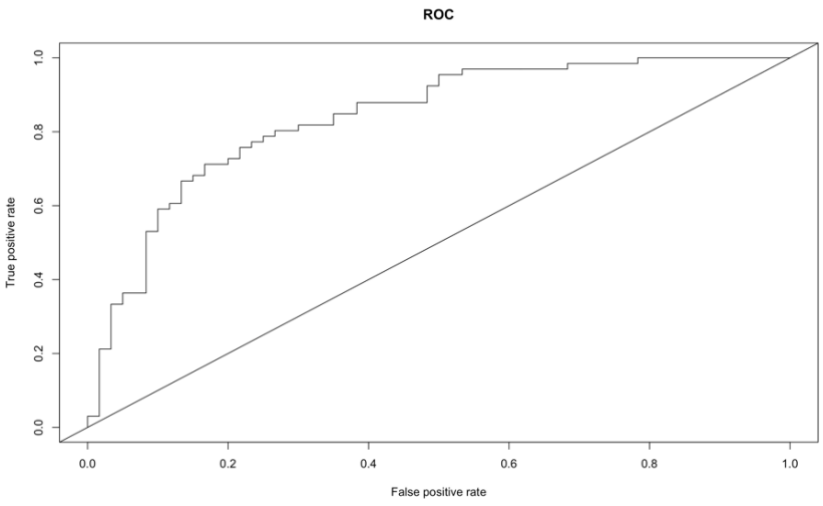
```
#Model tuned
library(caret)
x = train_downsample[,-12]
y = train_downsample$quality

naive_tuned_ds <- train(x = x,y = y,method = "nb",trControl = control,tuneGrid = search_grid)
naive_tuned_ds

#Predicting train set tuned
pred_naive_train_tuned_ds <- predict(naive_tuned_ds , train_downsample)
cm_naive_train_tuned_ds <- confusionMatrix(pred_naive_train_tuned_ds, train_downsample$quality, positive = "1")
cm_naive_train_tuned_ds

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  20
         1  28 133

               Accuracy : 0.8462
                 95% CI : (0.8012, 0.8843)
    No Information Rate : 0.5096
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6925

 Mcnemar's Test P-Value : 0.3123

            Sensitivity : 0.8693
            Specificity : 0.8239
         Pos Pred Value : 0.8261
         Neg Pred Value : 0.8675
             Prevalence : 0.4904
         Detection Rate : 0.4263
   Detection Prevalence : 0.5160
      Balanced Accuracy : 0.8466

       'Positive' Class : 1
```

```r
#Predicting test set tuned
pred_naive_test_tuned_ds <- predict(naive_tuned_ds, test_downsample)
cm_naive_test_tuned_ds <- confusionMatrix(pred_naive_test_tuned_ds, test_downsample$quality, positive = "1")
cm_naive_test_tuned_ds
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 46 15
         1 14 51

               Accuracy : 0.7698
                 95% CI : (0.6865, 0.8401)
    No Information Rate : 0.5238
    P-Value [Acc > NIR] : 1.066e-08

                  Kappa : 0.539

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.7727
            Specificity : 0.7667
         Pos Pred Value : 0.7846
         Neg Pred Value : 0.7541
             Prevalence : 0.5238
         Detection Rate : 0.4048
   Detection Prevalence : 0.5159
      Balanced Accuracy : 0.7697

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude that there is still overfitting of the dataset due to the accuracy of the training set being higher than the test set even after tuning. Although, it is worth noting that the accuracy has increased almost 5% higher after tuning.

```r
#ROC tuned
library(ROCR)
pred_naive_tuned_ds <- predict(naive_tuned_ds,test_downsample, type = 'prob')[,2]
ROC_naive_tuned_ds <- prediction(pred_naive_tuned_ds, labels = test_downsample$quality)
plot(performance(ROC_naive_tuned_ds,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```



```r
# AUC tuned
auc_ROCR_n <- performance(ROC_naive_tuned_ds, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.8391414
```

We can conclude that there is an 83.91% probability in predicting whether or not a wine is poor or excellent, which is just slightly higher than down-sampling without tuning.

Random Forest – No sampling

One of the ensemble methods includes the Random Forest method which is known as the combination of various tree predictors, where each tree relies on a random independent dataset and all the trees are equally distributed in the forest (Xuan, Zheng and Jiang, 2018). To train each tree, a subset of the training set is sampled arbitrarily (Niveditha, Abarna and Akshaya, 2019). Where a feature is being chosen from a random subset of the whole feature set, each node is then split. Consequently, the diversity of the forest is being increased leading to a more powerful general prediction. When being implemented as a classifier, the Random Forest will consider the majority vote as the predicted class. For this model, we have used the random forest function from the 'random forest package.

```
#Model
library(randomForest)
set.seed(225)
rf <- randomForest(quality~., train, trControl = control)
rf
```

Prediction

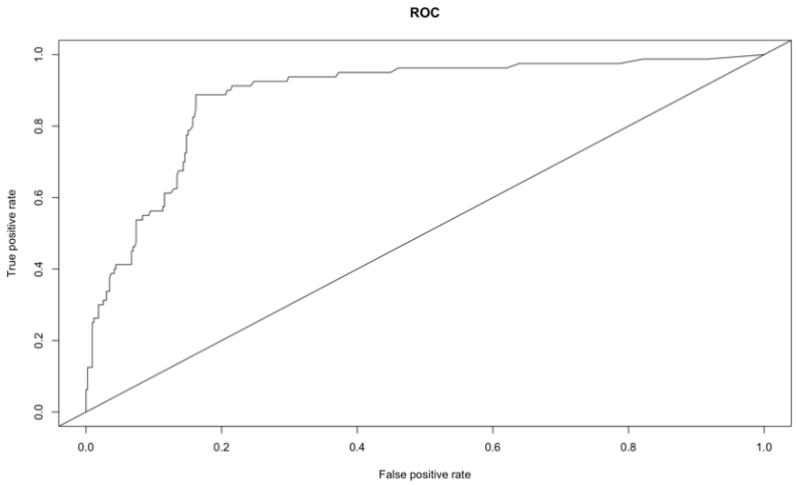
```
#Prediction- Train set
library(caret)
pred_rf_train <- predict(rf,train)
confusionMatrix(pred_rf_train,train$quality, positive ='1')

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 947   0
         1   1 139

               Accuracy : 0.9991
                 95% CI : (0.9949, 1)
    No Information Rate : 0.8721
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9959

 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9989
         Pos Pred Value : 0.9929
         Neg Pred Value : 1.0000
             Prevalence : 0.1279
         Detection Rate : 0.1279
   Detection Prevalence : 0.1288
      Balanced Accuracy : 0.9995

       'Positive' Class : 1
```

```r
#Prediction- Test set
pred_rf_test <- predict(rf,test)
confusionMatrix(pred_rf_test,test$quality, positive ='1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 414  49
         1  18  31

               Accuracy : 0.8691
                 95% CI : (0.8368, 0.8971)
    No Information Rate : 0.8438
    P-Value [Acc > NIR] : 0.0615754

                  Kappa : 0.4107

 Mcnemar's Test P-Value : 0.0002473

            Sensitivity : 0.38750
            Specificity : 0.95833
         Pos Pred Value : 0.63265
         Neg Pred Value : 0.89417
             Prevalence : 0.15625
         Detection Rate : 0.06055
   Detection Prevalence : 0.09570
      Balanced Accuracy : 0.67292

       'Positive' Class : 1
```

From the prediction of the test and train set, we can that the model is overfitting as the difference in accuracy between the two datasets is high. An accuracy of 99.91% was achieved on the training set and 86.91% on the test set. This occurrence means that the model trained on the training set does not generalize well on the test set. This is probably due to an imbalance dataset where excellent wines are being predicted in the minority class and poor-quality wines are the majority class. As the data is imbalanced, the model may overlearn the majority class too well resulting in low accuracy when predicting the minority class, which in turn led to an overfitting issue.

```r
#ROC
pred_rf <- predict(rf, test, type = 'prob')
ROC <- prediction(pred_rf[,2], labels = test$quality )
plot(performance(ROC,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```




```r
# AUC
auc_ROCR_n <- performance(ROC, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.8839844
```

We can conclude that there is 88.39% probability in predicting whether or not a wine is poor or excellent.

Random Forest – Down-sample

```
#Model
library(randomForest)
set.seed(223)
rf_downsample <- randomForest(quality~., train_downsample, trControl = control)
rf_downsample
```

```
Call:
 randomForest(formula = quality ~ ., data = train_downsample,      trControl = control)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 18.59%
Confusion matrix:
    0   1 class.error
0 131  28   0.1761006
1  30 123   0.1960784
```

Prediction

```
#Prediction- Train set
library(caret)
pred_rf_ds_train <- predict(rf_downsample,train_downsample)
confusionMatrix(pred_rf_ds_train,train_downsample$quality, positive ='1')
```

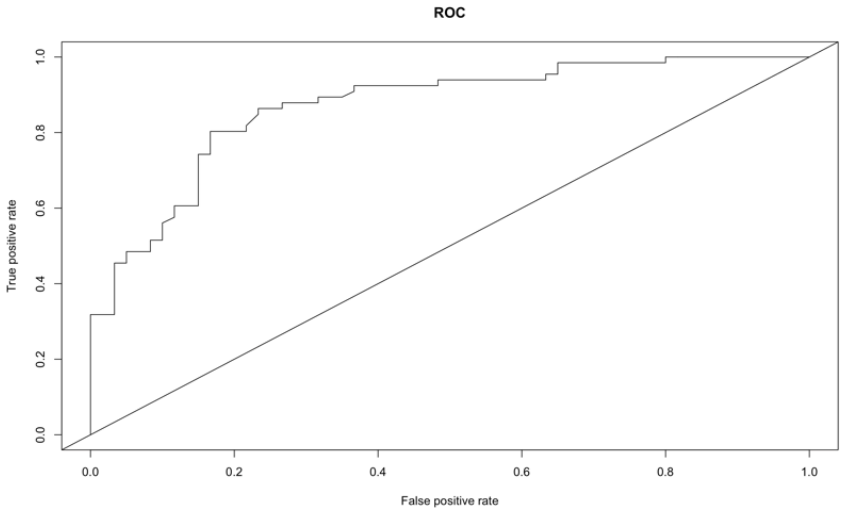
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 159   0
         1   0 153

               Accuracy : 1
                 95% CI : (0.9882, 1)
    No Information Rate : 0.5096
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4904
         Detection Rate : 0.4904
   Detection Prevalence : 0.4904
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

```r
#Prediction- Test set
pred_rf_ds_test <- predict(rf_downsample,test_downsample)
confusionMatrix(pred_rf_ds_test,test_downsample$quality, positive ='1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 47 12
         1 13 54

               Accuracy : 0.8016
                 95% CI : (0.7212, 0.8673)
    No Information Rate : 0.5238
    P-Value [Acc > NIR] : 8.412e-11

                  Kappa : 0.602

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8182
            Specificity : 0.7833
         Pos Pred Value : 0.8060
         Neg Pred Value : 0.7966
             Prevalence : 0.5238
         Detection Rate : 0.4286
   Detection Prevalence : 0.5317
      Balanced Accuracy : 0.8008

       'Positive' Class : 1
```

Similar to the case of down-sampling the dataset for the naïve bayes model, we can see an overfit in the mode as the accuracy of training set (100%) is much higher than the test set (80.16%).

```r
#ROC
pred_rf_ds <- predict(rf_downsample, test_downsample, type = 'prob')
ROC_ds <- prediction(pred_rf_ds[,2], labels = test_downsample$quality )
plot(performance(ROC_ds,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```




```r
# AUC
auc_ROCR_n <- performance(ROC_ds, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n
```

```
[1] 0.8689394
```

We can conclude that there is 86.89% probability in predicting whether or not a wine is poor or excellent, which is just slightly lower than without sampling.

Random Forest – Up-sample

```
#Model
library(randomForest)
set.seed(223)
rf_upsample <- randomForest(quality~., train_upsample, trControl = control)
rf_upsample

Call:
 randomForest(formula = quality ~ ., data = train_upsample, trControl = control)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 4%
Confusion matrix:
    0   1 class.error
0 892  68 0.070833333
1   7 906 0.007667032
```

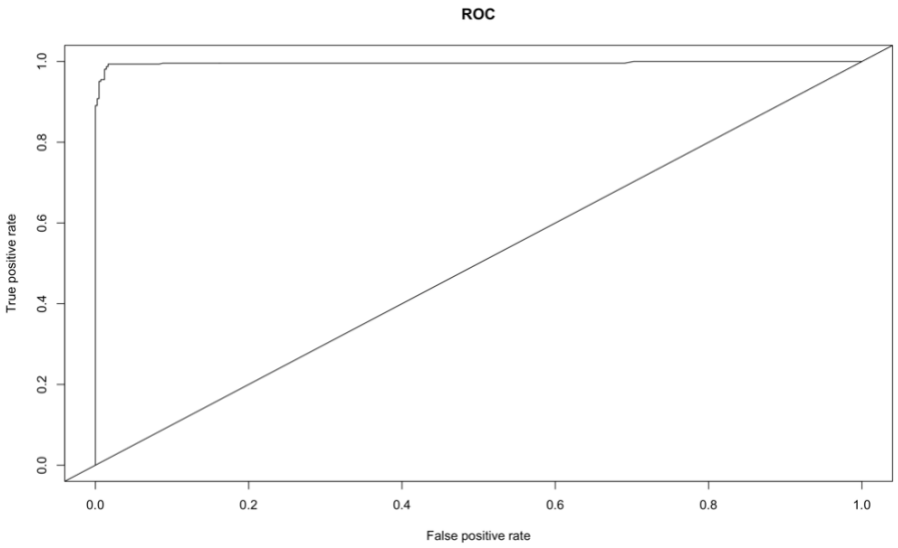
Prediction

```
#Prediction- Train set
library(caret)
pred_rf_us_train <- predict(rf_upsample,train_upsample)
confusionMatrix(pred_rf_us_train,train_upsample$quality, positive ='1')

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 957   0
         1   3 913

               Accuracy : 0.9984
                 95% CI : (0.9953, 0.9997)
    No Information Rate : 0.5125
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9968

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 1.0000
            Specificity : 0.9969
         Pos Pred Value : 0.9967
         Neg Pred Value : 1.0000
             Prevalence : 0.4875
         Detection Rate : 0.4875
   Detection Prevalence : 0.4891
      Balanced Accuracy : 0.9984

       'Positive' Class : 1
```

```
#Prediction- Test set
pred_rf_us_test <- predict(rf_upsample,test_upsample)
confusionMatrix(pred_rf_us_test,test_upsample$quality, positive ='1')

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 402   3
         1  18 464

               Accuracy : 0.9763
                 95% CI : (0.964, 0.9853)
    No Information Rate : 0.5265
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9524

 Mcnemar's Test P-Value : 0.00225

            Sensitivity : 0.9936
            Specificity : 0.9571
         Pos Pred Value : 0.9627
         Neg Pred Value : 0.9926
             Prevalence : 0.5265
         Detection Rate : 0.5231
   Detection Prevalence : 0.5434
      Balanced Accuracy : 0.9754

       'Positive' Class : 1
```

From the prediction of the test and train set, we can conclude a good fit for the model as the difference in accuracy between the two datasets is small. An accuracy of 99.84% was achieved on the training set and 97.63% on the test set.

```
#ROC
pred_rf_us <- predict(rf_upsample, test_upsample, type = 'prob')
ROC_us <- prediction(pred_rf_us[,2], labels = test_upsample$quality )
plot(performance(ROC_us,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```



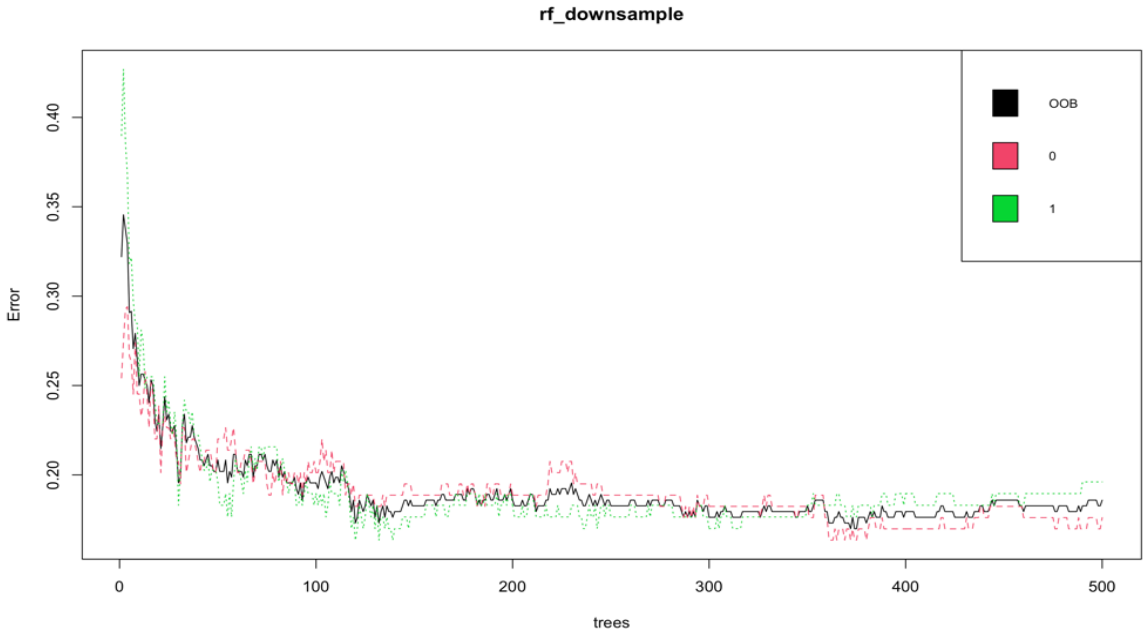

```
# AUC
auc_ROCR_n <- performance(ROC_us, measure = "auc")
auc_ROCR_n <- auc_ROCR_n@y.values[[1]]
auc_ROCR_n

[1] 0.9960538
```

We can conclude that there is 99.61% probability in predicting whether or not a wine is poor or excellent, which surpasses both the down-sampled and without sampling AUC scores.

Parameter Tuning – Down-sample

To achieve better accuracy for the down-sampled dataset, we have implemented parameter tuning. The two main parameters in the tuning process are the ‘mtry’ and ‘ntree’ parameters. ‘mtry’ represents the number of variables randomly sampled as candidates at each split while ntree is the number of trees to grow. From the plot below, it is obvious that the default value of ntree = 500 needs to increase, hence, we have set it to be at 750.

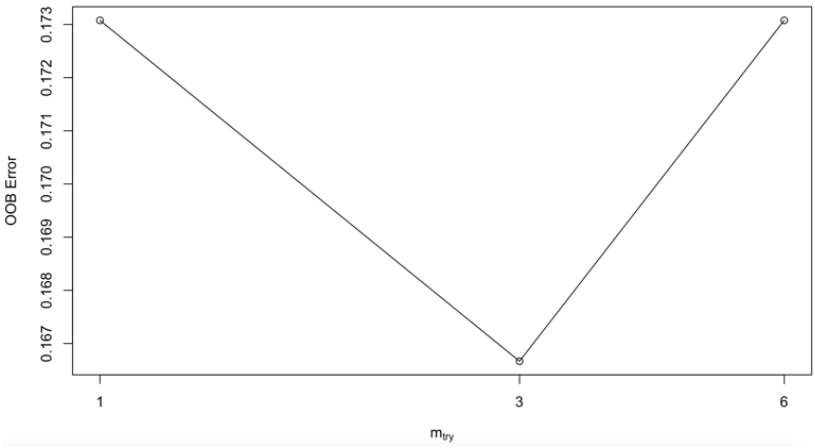
```
plot(rf_downsample)
legend("topright", colnames(rf_upsample$err.rate),col=1:6,cex=0.8,fill=1:6)
```



```
#Tune for mtry
train <- as.data.frame(train_downsample)
tuneRF(train_downsample[-12], train_downsample[,12],
       stepFactor = 0.5,
       plot = TRUE,
       ntreeTry = 750,
       trace = TRUE,
       improve = 0.05)
```

```
mtry = 3  OOB error = 16.67%
Searching left ...
mtry = 6        OOB error = 17.31%
-0.03846154 0.05
Searching right ...
mtry = 1        OOB error = 17.31%
-0.03846154 0.05
      mtry  OOBError
1.OOB    1 0.1730769
3.OOB    3 0.1666667
6.OOB    6 0.1730769
```

At ntree = 750, we have generated mtry = 3, which gives the lowest OOBError value.




```
#tuned model
rf_downsampled_tuned <- randomForest(quality~., train_upsample,
                           ntree = 750,
                           mtry = 3,
                           importance = TRUE,
                           promixity = TRUE)
rf_downsampled_tuned
```

```
Call:
 randomForest(formula = quality ~ ., data = train_upsample, ntree = 750,      mtry = 3, importance = TRUE, promixity = TRUE)
               Type of random forest: classification
                     Number of trees: 750
No. of variables tried at each split: 3

        OOB estimate of  error rate: 3.9%
Confusion matrix:
    0   1 class.error
0 894  66 0.068750000
1   7 906 0.007667032
```

Figure 5: After tuning

```
Call:
 randomForest(formula = quality ~ ., data = train_downsample,      trControl = control)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 18.59%
Confusion matrix:
    0   1 class.error
0 131  28   0.1761006
1  30 123   0.1960784
```

Figure 6: Before tuning

By tuning the parameters of mtry and ntree, we have successfully reduced the OOBError from 18.59% to 3.9%

Prediction

```
#Prediction- Train set tuned
pred_rf_train_tuned_ds <- predict(rf_downsampled_tuned,train_downsample)
confusionMatrix(pred_rf_train_tuned_ds,train_downsample$quality, positive ='1')
```

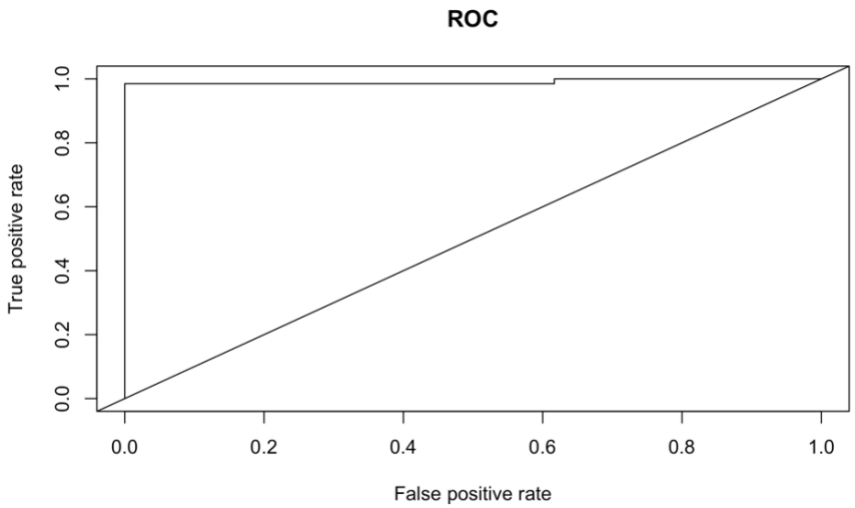
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 159   1
         1   0 152

               Accuracy : 0.9968
                 95% CI : (0.9823, 0.9999)
    No Information Rate : 0.5096
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9936

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9935
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9938
             Prevalence : 0.4904
         Detection Rate : 0.4872
   Detection Prevalence : 0.4872
      Balanced Accuracy : 0.9967

       'Positive' Class : 1
```

```r
#Prediction- Test set tuned
pred_rf_test_tuned_ds <- predict(rf_downsampled_tuned,test_downsample)
confusionMatrix(pred_rf_test_tuned_ds,test_downsample$quality, positive ='1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 59  1
         1  1 65

               Accuracy : 0.9841
                 95% CI : (0.9438, 0.9981)
    No Information Rate : 0.5238
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9682

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9848
            Specificity : 0.9833
         Pos Pred Value : 0.9848
         Neg Pred Value : 0.9833
             Prevalence : 0.5238
         Detection Rate : 0.5159
   Detection Prevalence : 0.5238
      Balanced Accuracy : 0.9841

       'Positive' Class : 1
```

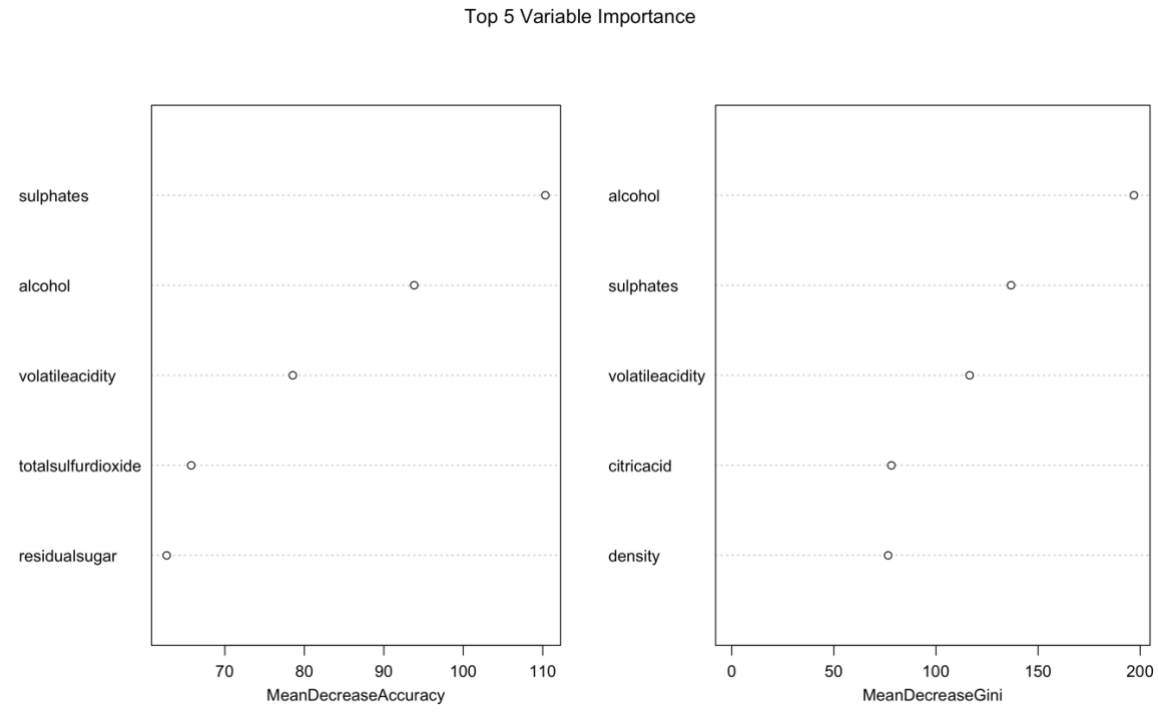
From the prediction of the test and train set, we can conclude a good fit for the model as the difference in accuracy between the two datasets is small. An accuracy of 99.35% was achieved on the training set and 98.41% on the test set, which surpasses the up-sampling and without sampling methods.

```r
#ROC- tuned
pred_rf_tuned_ds <- predict(rf_downsampled_tuned, test_downsample, type = 'prob')
ROC_tuned_ds <- prediction(pred_rf_tuned_ds[,2], labels = test_downsample$quality )
plot(performance(ROC_tuned_ds,"tpr", "fpr"),
     main = "ROC")
abline(a=0,b=1)
```



We can conclude that there is 99.07% probability in predicting whether or not a wine is poor or excellent.

```r
#Important Variables
varImp(rf_downsampled_tuned)
varImpPlot(rf_downsampled_tuned,
           sort = T,
           n.var = 5,
           main = 'Top 5 Variable Importance')
```



Furthermore, we have also included the top 5 variables which contributed the most to the tuned down-sampled mode which are sulphates, alcohol, volatile acidity, total sulfur dioxide, and residual sugar. The mean decrease Gini graph measures how pure the nodes are at the end of the tree without each variable, while the mean decrease accuracy graph tells us how worse the model performances are without each variable. We can conclude that the sulphate and alcohol variables contribute the most in predicting excellent wine quality.

3.5 Analysis and Recommendation

Table 5: Summary of the model analysis

No	Experiment	Accuracy	Specificity	Sensitivity	AUC score
Logistic Regression					
1.	No resampling	78.32	78.01	80.00	84.31
2.	Down-sample	76.98	80.00	74.24	84.87
3.	**Up-sample**	**80.96**	**78.57**	**83.29**	**86.23**
Naïve Bayes					
1.	No resampling	83.79	86.81	67.50	84.57
2.	Down-sample	72.22	73.33	71.21	82.20
3.	Tune- down-sample	76.98	76.67	77.27	83.91
4.	Up-sample	79.82	79.52	80.09	86.82
5.	**Tune- up-sample**	**83.09**	**80.71**	**85.22**	**91.84**
Random Forest					
1.	No resampling	86.91	95.83	38.75	88.40
2.	Down-sample	80.16	78.33	81.82	86.89
3.	**Tune- down-sample**	**98.41**	**98.33**	**98.48**	**99.07**
4.	Up-sample	97.63	95.71	99.36	99.61

From our experiment, we can observe that generally, the down-sampling method produced the lowest value of accuracies. This is probably due to the decrease in observations as down-sampling is known to reduce the majority class to match the minority class. Thus, the model does have enough observations to learn during the training phase. For the logistic regression model, the model that performed the best was with the up-sampling method with an accuracy of 80.96%, followed by the no resampling method with 78.32% and the down-sampling technique with 76.98%.

As for the Naïve Bayes model, the model that performed the best was the up-sampling after tuning method with an accuracy of 83.09% followed by no resampling technique.

Although the no resampling technique has an accuracy only slightly higher of 83.79%, its lower sensitivity value of 67.50% is why we do not consider it as the best model.

However, we can observe that the down-sampling method after tuning performed the best with the random forest model with an accuracy of 98.41% followed by the up-sampling method with 97.63% accuracy. The down-sampling method after tuning also produced the highest values for specificity and sensitivity. Specificity being the accuracy of negative cases (poor quality) that are classified correctly and sensitivity is the amount of actual positive cases (excellent wine) that are being predicted as positive.

Overall, we can conclude that the model that outperformed the rest was the down-sampling technique after parameter tuning with the random forest model. In addition, our model has also outperformed the accuracy values of other authors such as (Gupta and C, 2021) with random forest accuracy of 73.25%, (Pawar *et al.,* 2019) with random forest accuracy of 87.33%, and (Kumar *et al.*, 2020) with random forest accuracy of 65.83%.

Furthermore, we have also identified the top 2 features that contributed the most to predicting excellent wine qualities for this model which are sulphates and alcohol. Sulphates are known to maintain freshness and protect wines from oxidation and unwanted bacteria while the alcohol generally affects the overall taste, texture, and structure of the wine itself, thus, why they are the two most important features.

Conclusion

This report uses the red wine dataset of Portuguese “Vinho Verde” wine to predict the quality of the wine based on their physicochemical characteristics.

We have successfully carried three types of experiments which are the without resampling, down-sampling and up-sampling techniques to be used in three machine learning models which are the logistic regression, Naïve Bayes, and the random forest. We have also successfully performed the parameter tunings for the Naïve Bayes and random forest models, where we found that the random forest model that has been down-sampled and tuned outperformed all the other models generating the highest accuracy, sensitivity, and specificity scores. Moreover, we have also identified two important features that played major roles in predicting excellent wine qualities, which are sulphates and alcohol.

For future work, we suggest that the experiment include both the red and white wine dataset for a larger dataset. Furthermore, we also suggest using other resampling techniques such as the Synthetic Minority Oversampling Technique (SMOTE) along with more machine learning models such as the support vector machine (SVM), XGBoost, Decision Tree, and artificial neural network (ANN) to further improve the accuracy of the models.

From the analysis carried out above, we can conclude that it is very important that we implement the resampling technique in a future experiment to balance out the dataset beforehand. This is to ensure that better performance of the machine learning models is achieved. This analysis is particularly important as the results will benefit many as it can be used by wine manufacturers to improve the quality of future wine and certification bodies are also able to implement it in their quality control phase. Not to forget the end consumers that purchase these beverages are able to make a selective and wise decision on the quality of wine before purchasing them.

References

Castaldo, L. *et al.* (2019) 'Red wine consumption and cardiovascular health', *Molecules*, 24(19). doi: 10.3390/molecules24193626.

Chen, B. *et al.* (2018) 'Wineinformatics: A quantitative analysis of wine reviewers', *Fermentation*, 4(4), pp. 1–16. doi: 10.3390/fermentation4040082.

Cortez, P. *et al.* (2009) 'Modeling wine preferences by data mining from physicochemical properties', *Decision Support Systems*, 47(4), pp. 547–553. doi: 10.1016/j.dss.2009.05.016.

Dahal, K. R. *et al.* (2021) 'Prediction of Wine Quality Using Machine Learning Algorithms', *Open Journal of Statistics*, 11(02), pp. 278–289. doi: 10.4236/ojs.2021.112015.

Dahal, K. R. and Gautam, Y. (2020) 'Argumentative Comparative Analysis of Machine Learning on Coronary Artery Disease', *Open Journal of Statistics*, 10(04), pp. 694–705. doi: 10.4236/ojs.2020.104043.

Er, Y. (2016) 'The Classification of White Wine and Red Wine According to Their Physicochemical Qualities', *International Journal of Intelligent Systems and Applications in Engineering*, 4(Special Issue-1), pp. 23–26. doi: 10.18201/ijisae.265954.

Gupta, M. and C, V. (2021) 'A Study and Analysis of Machine Learning Techniques in Predicting Wine Quality', *International Journal of Recent Technology and Engineering*, 10(1), pp. 314–321. doi: 10.35940/ijrte.a5854.0510121.

Gupta, Y. (2018) 'Selection of important features and predicting wine quality using machine learning techniques', *Procedia Computer Science*, 125, pp. 305–312. doi: 10.1016/j.procs.2017.12.041.

Kumar, S., Agrawal, K. and Mandan, N. (2020) 'Red wine quality prediction using machine learning techniques', *2020 International Conference on Computer Communication and Informatics, ICCCI 2020*, (January 2020). doi: 10.1109/ICCCI48352.2020.9104095.

Pawar, D., Mahajan, A. and Bhoithe, S. (2019) 'Wine Quality Prediction using Machine Learning Algorithms', *International Journal of Computer Applications Technology and Research*, 8(9), pp. 385–388. doi: 10.7753/ijcatr0809.1010.

Report, T. *et al.* (2021) 'Report Name : Portuguese Wine Exports Rise in 2020 Despite COVID-19 Challenges'.

Sharma, N. (2017) 'Wine Quality Prediction Using Machine Learning'. Available at: https://medium.com/themlblog/wine-quality-prediction-using-machine-learning-59c88a826789.

Sinha, A. and Kumar, A. (2020) 'Wine Quality and Taste Classification Using Machine Learning Model', *International Journal of Innovative Research in Applied Sciences and Engineering*, 4(4), pp. 715–721. doi: 10.29027/ijirase.v4.i4.2020.715-721.