DATA ANALYSIS PROJECT II

ADVANCED ANALYSIS OF RED WINE QUALITY

GROUP 6

15332-MALEESHA NILUMINDA

15378-SHANALIE RANASINGHE

15669-GANESHI UMAYANGANA

ABSTRACT

Maintaining and growing a customer base in the current competitive industry depends heavily on the quality of the product. The market is overflowing with different varieties of wine, and the wine industry is no different. "How do we pick the best one?" is a question frequently. This research focuses on the chemical characteristics of red wine to answer this specific topic.

We performed a thorough analysis of the Red Wine quality dataset from the UCI Machine Learning Repository using R software. The objective is to identify the critical chemical elements influencing the quality of red wine. The results are intended to assist customers and winemakers in making knowledgeable decisions about the selection and production of wine.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Wine can be both very simple and incredibly complex. It's an alcoholic drink made by fermenting grape juice. Most wine, as we know, is made with grapes, but it can technically be made from other fruits too, such as apples, blueberries, and strawberries.

Why have grapes become the standard? There are two main reasons. Grapes contain acids; malic, tartaric, and citric acids that preserve the wine, allowing it to be aged for decades or even centuries. Secondly, grapes have a much higher sugar content than other fruits, allowing them to ferment successfully and produce complex wines.

Wine is the most widely consumed beverage globally, and its values are considered important. The quality of wine is always important for its consumers, and mainly for producers in the present competitive market to raise their revenue.

Historically, wine quality used to be determined by testing at the end of the production; to reach the level, one already spends lots of time and money. Every person has their own opinion about taste, so identifying a quality based on a person's taste is challenging.

With the development of technology, manufacturers started to rely on various device testing in the development phases. So, they can have a better idea about wine quality. This helped in accumulating lots of data with various parameters such as the quality of different chemicals and temperatures used during the production, and the quality of wine produced. One can adjust the variables that directly affect the wine's quality during this process. This provides the producer with a greater understanding of how to adjust various development process parameters to

optimize the wine's quality. Additionally, this might produce wines with a variety of flavors and, ultimately, might create a new brand. Therefore, it is crucial to analyze the fundamental factors that affect wine quality. In this work, we have demonstrated how machine learning (ML) may be used to predict wine quality by determining the optimal parameter that determines wine quality.

2. DESCRIPTION OF THE QUESTIONS

Our main goal in this exploratory analysis is to better understand the complex relationship that exists between wine's physicochemical characteristics and its perceived quality, which is represented by the qualitative variable "Quality" and is scaled from 0 to 10. Acknowledging the industry's critical need for wine quality evaluation, we aim to accomplish the following main goals:

1. Identification of Impactful Physicochemical Properties
2. Determination of Optimal Physicochemical Levels
3. Development of a Predictive Model for Quality Assessment

While acknowledging that personal preferences may occasionally deviate from the objective standard of quality, our goal in doing this analysis is to promote a more objective approach to wine quality prediction. Our goal is to give the wine industry a strong foundation for improving the quality of their goods and enabling better informed decision-making by breaking down quality evaluation into quantifiable and explicable variables.

3. DESCRIPTION OF THE DATASET

This Red Wine Quality Dataset was taken from Kaggle. It contains 12 different properties of wine. One of them is “Quality” which is the response variable for the study and the remaining variables are based on physicochemical factors. There are 1599 observations produced in the Vinho-Verde region of Portugal.

Variable	Description	Data Type
Quality	Based on sensory data (a score between 0-10)	Categorical
Fixed acidity	These are non-volatile acids that do not evaporate readily(g/dm3)	Numeric
Volatile acidity	are high acetic acid in wine which leads to an unpleasant vinegar taste(g/dm3)	Numeric
Citric acid	Acts as a preservative to increase acidity (small quantities add freshness and flavor to wines) (g/dm3)	Numeric
Residual sugar	The amount of sugar remaining after fermentation stops. The key is to have a perfect balance between — sweetness and sourness (wines > 45g/dm3 are sweet)	Numeric
Chlorides	The amount of salt in the wine(g/dm3)	Numeric
Free sulfur dioxide	So2 is used for the prevention of wine by oxidation and microbial spoilage(mg/dm3)	Numeric
Total sulfur dioxide	Is the amount of free and bound forms of SO2(mg/dm3)	Numeric
Density	The density of wine is close to that of water depending on the present alcohol and sugar content. (g/dm3)	Numeric

pH	the level of acidity-free Sulfur Dioxide: it prevents microbial. Basic wine is on a scale from 0(very acidic) to 14(very basic)	Numeric
Sulphates	A wine additive that contributes to SO2 levels and acts as an antimicrobial and antioxidant. Which preserves freshness and protects wine from oxidation and bacteria(g/dm3)	Numeric
Alcohol	Percent of alcohol present in wine (% by volume)	Numeric

Table 1:- Variable Description

4. IMPORTANT RESULTS OF DESCRIPTIVE ANALYSIS

- **Response variable - Quality of wine**

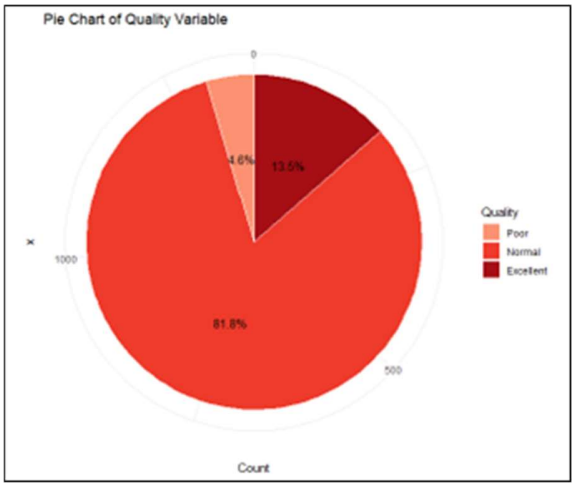


Figure 2:Pie chart of uality Variable

The "quality" variable assigns a rating, ranging from 0 to 10, to each red wine sample as determined by an expert in the field. Their categories have an order. So quality is an ordinal categorical variable. For the convenience of further analysis, the quality arable was recorded as, **Poor (1-4), Normal (5-6), and Excellent (7-10)**

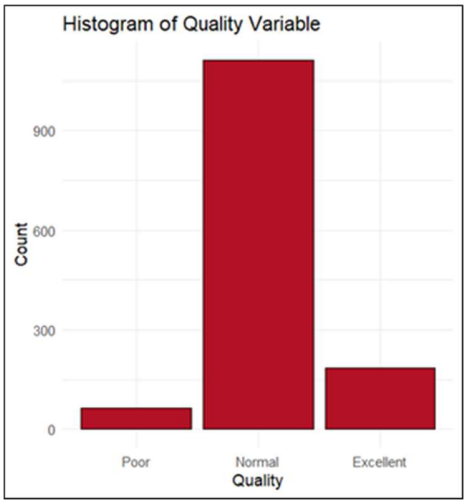


Figure 1:Histogram of Quality Variable

A majority of the wines were Normal while there were fewer rated Excellent or Poor quality categories.

- **Univariate Analysis Results**

Variable name	Mode value	Majority Range	Permissible Ranges
Fixed acidity	7.2000 (g/dm3)	6.8 - 9.3 (g/dm3)	Max limit - 16 (g/dm3)
Volatile acidity	0.5000(g/dm3)	0.38 - 7.8 (g/dm3)	Max limit 0.4 (g/dm3)
Citric acid	0.0000(g/dm3)	0.0 - 0.43 (g/dm3)	0.275 - 0.875 (g/dm3)
Residual sugar	2.0000(g/dm3)	1.9 - 2.8 (g/dm3)	Depending on the type of red wine
Chlorides	0.0800(g/dm3)	0.05 - 0.18 (g/dm3)	Max limit - 0.611 (g/dm3)
Free sulfur dioxide	6.0000(mg/dm3)	5 - 20 (mg/dm3)	Usually around 70 (mg/dm3)
Total sulfur dioxide	28.000(mg/dm3)	22 - 62 (mg/dm3)	Maximum Limit - 289 (mg/dm3)
Density	0.9968(g/dm3)	0.9956 - 0.9978 (g/dm3)	Maximum limit - 1.004 (g/dm3)
pH	3.3000	3.2 - 3.4	3 - 4
Sulphates	0.5400(g/dm3)	0.45 - 0.72 (g/dm3)	0.5 - 1.0 (g/dm3)
Alcohol	9.5000(g/dm3)	8.8 - 11.1 (g/dm3)	10% -13%

Table 2:Summary of univariate analysis

- **Bivariate Analysis Results**

  Associations with Wine Quality. (Spearman's Correlation Results)

Fixed acidity	Volatile acidity	Citric acid	Residual sugar	Chlorides	Free sulfur dioxide	Total sulfur dioxide	Density	pH	Sulphates	Alcohol
0.12	-0.321	0.225	0.054	-0.127	-0.02	-0.091	-0.128	-0.100	0.303	0.337

Table 3: Correlation of predictor variables with Wine Quality

Variable name	Distribution
Fixed acidity	High Fixed Acidity → High Quality
Volatile acidity	Low Volatile Acidity → High Quality
Citric acid	High Citric Acid → High Quality
Residual sugar	No clear association was observed with quality. The spread of the boxplot is low. Can't identify any difference between quality categories.
Chlorides	No clear association was observed with quality. The spread of the boxplot is low. Can't identify any difference between quality categories
Free sulfur dioxide	High Free SO2 → High Quality
Total sulfur dioxide	High Total SO2 → High Quality
Density	No clear association was observed with quality
pH	Low pH → High Quality
Sulphates	High Sulfite → High Quality
Alcohol	High Alcohol Level → High Quality

Important correlations among predictors.

- Free sulfur dioxide with total sulfur dioxide ->(0.79)
- pH with fixed acidity ->(-0.71)
- Fixed acidity with citric acid ->(0.66)
- Density with fixed acidity -> (0.63)
- Citric with volatile acidity ->(-0.61)
- pH with citric acid -> (-0.56)

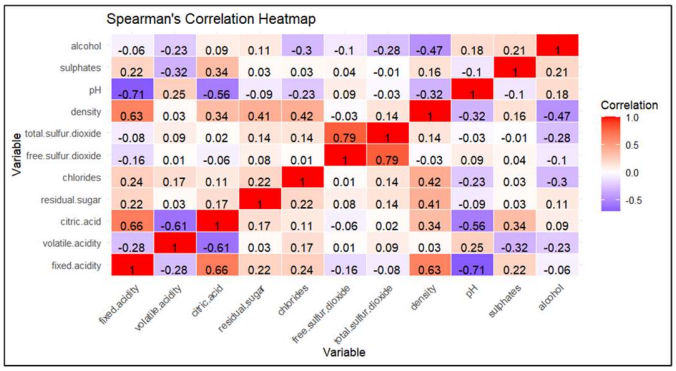


Figure 3: Spearman's Correlation Heatmap

Table 4:Summary of Predictor variations with Wine Quality

Figure 4: Fixed Acidity vs Citric Acid

Citric acid is a fixed acid found in small quantities in wine . Therefore they have a strong positive relationship.

Figure 5:Volatile Acidity vs Citric Acid

Citric acid may inhibit the growth or activity of acetic acid bacteria, which are responsible for converting alcohol into acetic acid. By inhibiting these bacteria, citric acid can indirectly reduce the formation of volatile acidity

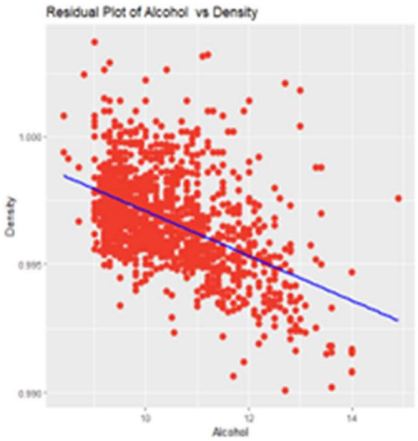


Figure 7:Alcohol vs Density

The density of wine demonstrates an approximate negative correlation with alcohol content and slight positive correlation with residual sugar content. This suggests that the addition of sugar, alcohol, and other supplementary ingredients, aimed at enhancing the quality of the wine, may contribute to a decrease in the overall density of the liquid.

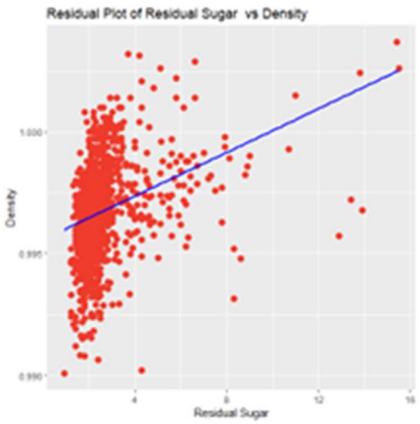
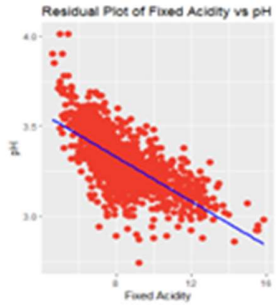
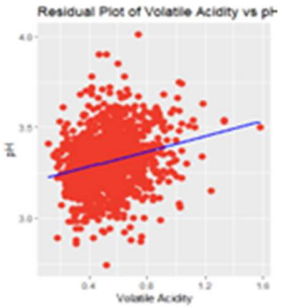


Figure 6:Residual Sugar vs Density





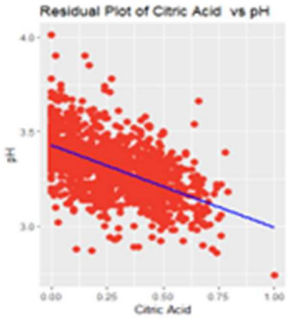


Figure 8;Fixed acidity vs pH

Acids with pH plots have negative relationships except the plot volatile acidity and pH. That's because of the pH scale(when acid level increases pH level decreases). Here volatile acidity has a positive relationship with pH, maybe that's due to Simpson's paradox

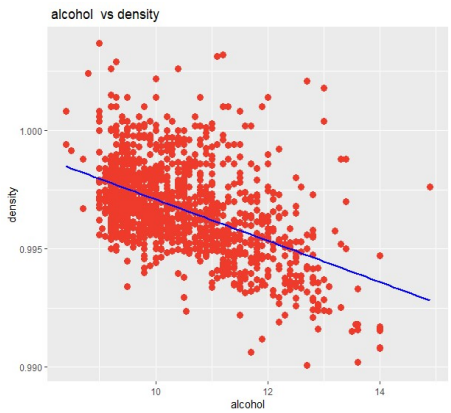


Figure 10:Alcohol vs Density

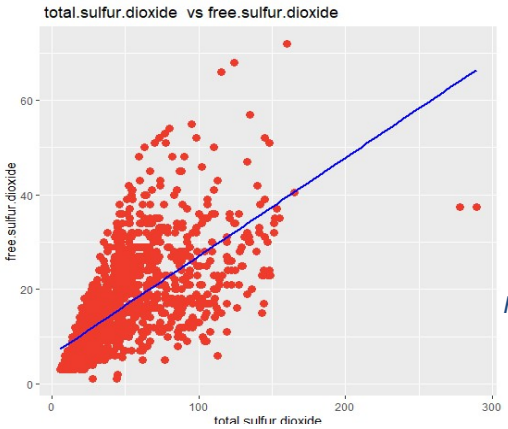


Figure 9: Total SO2 vs free SO2

With a higher amount of alcohol, the density decreases, which was also confirmed for red wine samples.

Total SO2 is the portion of SO2 that is free in the wine plus the portion that is bound to other chemicals in the wine such as aldehydes, pigments, or sugars -(*Iowa State University of Science and Technology.*) therefore they have a strong positive relationship.

Identifying the clusters and significantly correlated predictors

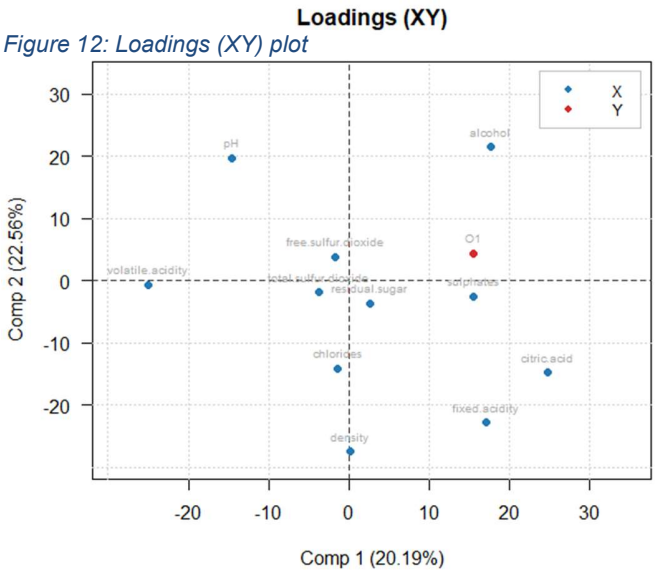


Figure 12: Loadings (XY) plot

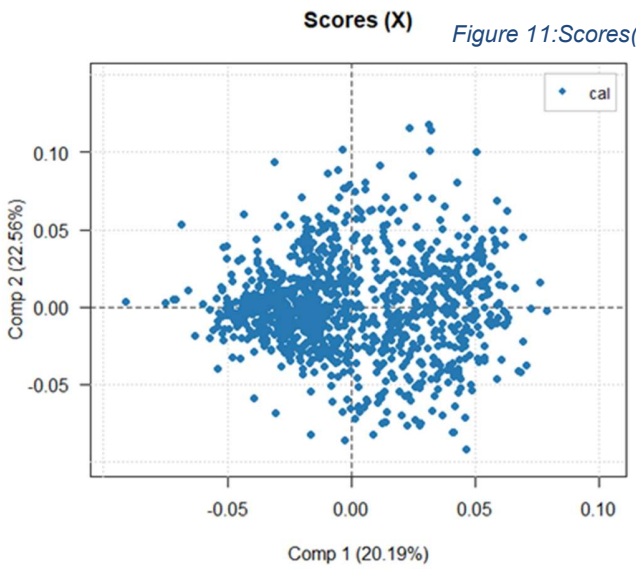


Figure 11:Scores(X) plot

- PLSR was performed on the data to identify any clusters among the observations as well as to identify any significantly correlated predictors.
- By considering the Scores(XY) sometimes, there may be some considerable clusters. But not exactly say about that using only eye inspections.
- According to Loading(XY) here implies that there are strong correlations among some predictors and responses, where some predictors are orthogonal to the response as well.
  Alcohol, Citric acid, fixed acidity and sulphates -These explanatory variables form a small angle with the response variable. - These are positively correlated predictors.
  pH, volatile acidity, Free sulfur dioxide, Volatile acidity, Chloride - These variables form a large angle (close to 180) - These are Negatively correlated predictors.

Important Findings

Using the findings of bivariate analysis, the best measurements for quality are,

- High fixed Acidity
- Low Volatile Acidity
- High Citric Acidity
- High Sulfite Concentration
- High Alcohol level

- Volatile Acidity and Citric acidity negatively Correlated
- Fixed Acidity and Citric Acidity Positively Correlated
- Total SO2 and Fixed S02 are positively correlated

High correlations exist between the variables as discussed above. These correlations can be handled using Penalized Logistic Regression and other Machine Learning Approaches which were proposed under the descriptive analysis. The distributions of most of the predictors were slightly skewed, and this will be addressed using data scaling and transformations under the advanced

5. IMPORTANT RESULTS IN ADVANCED ANALYSIS

Our 'Red wine quality' dataset response variable quality is an ordinal variable with 0 to 10. We recategorized these levels with three categories. Hence Multiclass ordinal logistic regression methods were used for the model building. (As a benchmark model).

Quality categories have imbalanced data

Because there were substantially more observations under the "Normal" Wine Quality category, the dataset was imbalanced, hence Sampling techniques were used as a remedy to balance the dataset. We used up-sampling, down-sampling, weighted method, and SMOTE techniques.

- Up-Sampling: Increasing the number of instances in the minority class to balance class distribution.
- Down-Sampling: Reducing the number of instances in the majority class to balance class distribution.
- Weighted method: Assigning different weights to classes during model training to account for imbalances.
- SMOTE: Generating synthetic samples for the minority class to balance class distribution.

Outlier Analysis

Certain expected patterns in the variables were not found under the descriptive analysis, and several unexplained relationships were noted. Additionally, skewed distributions for a number of predictor factors were noted. Since creating a classification/prediction model is the goal, outlier analysis is carried out on the training set. When an ordinal model responds ordinally, there won't be any outliers. After that, a custom function makes sure there are no outliers common to the predictor variable by checking for them and using the DBSCAN method to confirm it. . Finally, the same method is used to check for outliers in each predictor, resulting in the appearance of 275 outliers. These outliers are then subjected to square root and log10 transformations, which reveal 326 and 241 outliers, respectively, that were discovered using the IQR boundaries method.

Table 5:Tranformation outlier percentages	Transformation		
	Original data	**Log10**	**Square root**
Outlier percentages for predictors (by observing individual predictors)	**4.36%**	**3.68%**	**4.98%**

Although they were outside of the IQR ranges of the predictors in the provided dataset, the observations that were flagged as outliers by all of those mentioned outlier detection techniques were within desirable ranges by international wine standards. As a result, every model was applied to the entire training dataset and no observations were eliminated from it. It is also a very small percentage.

Ordinary Logistic regression with all variables

Quality, the response variable, is categorical. Using all the predictor variables, standard ordinal logistic regression was then conducted.

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.1818	0.9109	0.4068	0.5500	0.6170	0.6418	0.8440

Table 6:Results of Ordinary Logistic Regression with all predictor variables

Best subset method

Here we used the best subset selection and fit the model. Using the best subset selection we got a model with 5 variables.

Variables	Volatile acidity	Chlorides	pH	Sulphates	Alcohol
Coefficients	-0.4488	-0.8552	-0.2593	0.3894	0.1233

Table 7:Best Subset selection

Ordinary Logistic regression after best subset selection

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.1818	0.9124	0.4516	0.5500	0.6362	0.6667	0.8475

Table 8:Results of Ordinary Logistic Regression after best subset selection

Ordinary Logistic regression with smote

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.3478	0.8948	0.3571	0.6835	0.6106	0.6168	0.8191

Table 9:Results of Ordinary Logistic Regression with smote

Ordinary Logistic regression with up-sampling

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.1449	0.6757	0.4800	0.6507	0.6596	0.7870	0.5674

Table 10:Ordinary logistic regression with up sampling:

Ordinary Logistic regression with down-sampling

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.1219	0.6359	0.4035	0.6268	0.5787	0.7006	0.5142

Table 11:Ordinary Logistic regression with down sampling:

Ordinary Logistic regression with weights

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.1205	0.610	0.4640	0.6250	0.6170	0.7715	0.5106

Table 12:Ordinary Logistic regression with weights

Multinomial Logistic regression

We applied Multinomial Logistic regression as the benchmark model.

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.3077	0.9061	0.3934	0.5982	0.6213	0.6377	0.8369

Table 13:Results of Multinomial Logistic Regression with all predictor variables

VIF Values

Fixed acidity	Volatile acidity	Citric acid	Residual sugar	Chlorides	Free SO2	Total SO2	Density	pH	Sulphates	Alcohol
8.8918	2.0112	3.7809	1.7708	1.5155	2.0824	2.3013	7.0799	3.6337	1.5402	2.8535
8.8345	1.8829	3.6366	1.7060	1.2806	2.2324	2.5508	7.7239	3.4498	1.4368	2.6711

Table 14:VIF values obtained for the multinomial logistic regression model with all predictor

Multinomial Logistic regression after best subset selection

Given that multinomial logistic regression with all predictors revealed a modest level of multicollinearity, the best subset selection technique was used to minimize the dimensionality of the data. The best subset of predictors have been found to be all variables excluding density, free sulfur dioxide, and fixed acidity variables. The class-wise statistics and VIF values that were obtained when a subset of the predictors from feature selection were subjected to multinomial logistic regression are displayed in the tables below. When comparing VIF values and statistics to multinomial logistic regression, which takes into account all predictors, a little decrease was seen.

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.1667	0.9102	0.4516	0.5482	0.6340	0.6667	0.844

Table 15: multinomial logistic regression model after best subset selection

		Volatile acidity	Chlorides	pH	Sulphates	Alcohol
VIF	2	1.2315	1.3055	1.2799	1.2558	1.1641
	3	1.1031	1.1776	1.2378	1.1153	1.0999

Table 16:VIF values obtained for the multinomial logistic regression model with all predictor

Multinomial Logistic regression with SMOTE

Test	F1_Scor			Accuracy			Overall Accuracy
	Poor	Normal	Excellent	Poor	Normal	Excellent	
	0.400	0.8986	0.3934	0.6889	0.6319	0.6377	0.8262

Table 17: multinomial logistic regression model under SMOTE.

VIF Values

Fixed acidity	Volatile acidity	Citric acid	Residual sugar	Chlorides	Free SO2	Total SO2	Density	pH	Sulphates	Alcohol
8.6865	2.0440	3.7454	1.7793	1.4703	2.0931	2.2822	6.7263	3.6029	1.5408	2.9154
8.6806	1.8891	3.6497	1.7048	1.2754	2.2318	2.5494	7.5592	3.3922	1.4456	2.6352

Table 18: VIF values obtained for the multinomial regression model with predictors selected under SMOTE

KNN (K Nearest Neighbor)

K-nearest neighbors (KNN) algorithm is a type of supervised ML Algorithm, one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.The following values were found on test data using SMOTE sampling to adjust for imbalance and changing the value of k to get the best results.

	Precision	Recall	Accuracy		F1_Score	
			Class-wise	Overall	Class-wise	Overall
Poor	0.9717	009434	0.9575	0.8575	0.7715	0.7715
Normal	0.8233	0.9763	0.8998		0.8997	
Excellent	0.9796	0.9013	0.9405		0.7423	

Table 19:Results obtained for the KNN classifier with hyperparameter tuning.

Random Forest

Decision trees are the basic learning model used by the categorization method Random Forest. Since each tree will make a different mistake, aggregating the findings of several trees ought to produce results that are more accurate than those of a single tree, according to the basic premise of Random Forest. In this manner, the model use averaging to increase prediction accuracy and manage over-fitting while fitting multiple decision tree classifiers on different subsamples of the dataset. For SMOTE data,

	Precision	Recall	Accuracy		F1_Score	
			Class-wise	Overall	Class-wise	Overall
Poor	0.3333	0.1000	0.5463	0.8582	0.1538	0.8786
Normal	0.8793	0.9617	0.6511		0.9187	
Excellent	0.6818	0.4054	0.6884		0.5054	

Table 20:: Results of the Random Forest Classifier under the default parameters

In order to improve the model used hyper- parameter tuning

	Precision	Recall	Accuracy		F1_Score	
			Class-wise	Overall	Class-wise	Overall
Poor	0.5000	0.2000	0.5963	0.8617	0.2857	0.8794
Normal	0.8828	0.9617	0.6617		0.9206	
Excellent	0.6818	0.4054	0.6884		0.5085	

Table 21:Table 12: Results of Random Forest Classifier after conducting hyperparameter tuning.

Final Model Using Random Forest Classifier,

Above various types of Machine Learning Models, we decided that Random Forest Classifier performs best with the following variables most important model building.

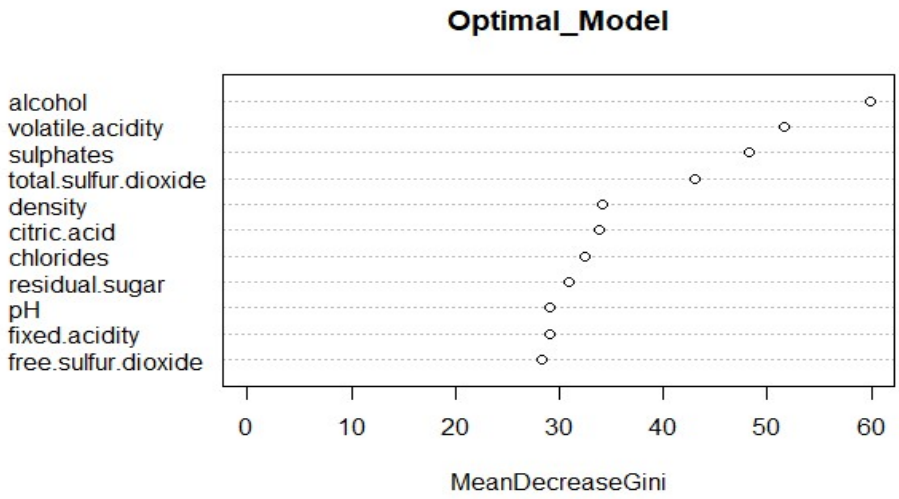


Figure 13:Optimal model obtained from random forest

6. ISSUES ENCOUNTERD AND SOULUTIONS

- Every sample of red wine in the dataset is assigned a quality score, ranging from 0 to 10. To further simplify the analysis and lower its complexity, this problem was reclassified into three levels: poor, normal, and excellent.
- The dataset was unbalanced since there were more observations under the "Normal" Wine Quality category than under the other two levels. As a result, sampling techniques were applied to each training algorithm. (Normal: 81.8%, Poor: 4.6%, Excellent: 13.5%)
- Significant background study was done before the analysis due to the predictors' representation of physicochemical and sensory data, making it difficult to evaluate correlations between the variables.
- There are just sensory output variables and physicochemical predictors available. No details about the kind of grape used, the type of wine, the brand, the selling price, etc. As a result, the analysis has limitations and might only be relevant to a certain quantity of Vinho-Verde wine at a given period.
- It can be subjective to assign weights to variables that affect wine quality, which might result in different conclusions. To get over this Organize a group of knowledgeable winemakers, sommeliers, and wine reviewers to decide on weight allocations and evaluate the relative value of various elements.

7. DISCUSSION AND CONCLUSIONS

	Accuracy	F1_score
Ordinary Logistic(All variables)	0.8440	0.1818
Ordinary Logistic(best subset)	0.8475	0.1818
Ordinary Logistic (Up sampling)	0.5676	0.1449
Ordinary Logistic(Down sampling)	0.5142	0.1219
Ordinary Logistic(smote)	0.8191	0.3478
Ordinary Logistic(weighted)	0.5106	0.1205
Multiple Logistic(All variables)	0.8369	0.3077
Multiple Logistic(Best subset)	0.8440	0.1667
Multiple Logistic(Smote)	0.8262	0.4000
KNN(Smote)	0.8575	0.7715
Random Forest Classifier	0.8617	0.8794

Table 22:Summary of Final Results obtained for all the models

The Random Forest Classifier, which has a comparatively high-Test Accuracy and F1 score, is the best model for predicting the quality of the Red Wine dataset, as revealed by the stepwise Advanced Analysis.

8. APPENDIX

```
################################################## smote model ################################
set.seed(100)
train_smote = SMOTE(X =  train_data[,-12],
                    target = train_data[,12],
                    dup_size = 3)
train_smote = train_smote$data # extract only the balanced dataset
view(train_smote)
train_smote$class = as.factor(train_smote$class)
prop.table(table(table(train_smote$class)))

best_model_smote = regsubsets(class~.,data = train_smote, nvmax = 11)
best_model_smote.summary = summary(best_model_smote)
which.max(best_model_smote.summary$adjr2)
par(mar = c(3, 3, 2, 2) + 0.05)
plot(best_model_smote.summary$adjr2,xlab="Number of Variables",ylab="adjr2")
points(8,best_model_smote.summary$adjr2[8],pch=20,col="red")
coef(best_model_smote,8)

ologitModel1 = polr(class ~ fixed_acidity + volatile_acidity +residual_sugar +
                    chlorides +total_sulfur_dioxide + pH + sulphates + alcohol ,data =train_smote)
ologitModel1

pred3  = predict(ologitModel1,newdata = train_smote,type = "class")
(length(pred3))
(conf_matrix = confusionMatrix(pred3, train_smote$class))

pred4  = predict(ologitModel1,newdata =test_data,type = "class")
(length(pred4))
(conf_matrix = confusionMatrix(pred4,test_data$quality))
```

```
#Checking the algorithm
set.seed(1)
train_pred=knn(train_smote[-12], train_smote[-12], train_smote[,12], k=3,prob = TRUE,use.all=TRUE)
train_acc=mean(train_pred == train_smote[,12])

#KNN models will always get 100% accuracy on the training set when K=1
#To select the value of K we need to create a validation set
set.seed(1)
valid_pred= knn(train, valid, outcome_train, k=1,prob = TRUE,use.all=TRUE)
valid_acc=mean(valid_pred == outcome_valid)

cat('Training Accuracy:   ', train_acc, '\n',
    'Validation Accuracy: ', valid_acc, sep='')

set.seed(1)
train_acc=c()
valid_acc=c()

k_range=2:100
for (i in k_range){
  set.seed(1)
  train_pred =knn(train, train, outcome_train, k=i,prob = TRUE,use.all=TRUE)
  train_acc =c(train_acc, mean(train_pred == outcome_train))

  set.seed(1)
  valid_pred=knn(train, valid, outcome_train, k=i,prob = TRUE,use.all=TRUE)
  valid_acc=c(valid_acc, mean(valid_pred == outcome_valid))

}
max(valid_acc)
```

```
################################################## weighted-sample model ##########################

class_freq = c(48, 906, 144)  # Number of observations for each quality level
weights = 1 / class_freq
#weights = c(20.5,1,6.3)
train_data$weights = weights[train_data$quality]

best_model_weight = regsubsets(quality ~.-weights,data = train_data,weights =train_data$weights,  nvmax = 11)
best_model_weight.summary = summary(best_model_weight)
which.max(best_model_weight.summary$adjr2)
par(mar = c(3, 1, 1, 1) + 0.1)
plot(best_model_weight.summary$adjr2,xlab="Number of Variables",ylab="adjr2")
points(10,best_model_weight.summary$adjr2[10],pch=20,col="red")
coef(best_model_weight,10)

ologitModel_weight = polr(quality ~ fixed.acidity +volatile.acidity +citric.acid +  residual.sugar+ chlorides+
                            free.sulfur.dioxide + total.sulfur.dioxide  + pH + sulphates + alcohol
                          ,weights =train_data$weights, data =train_data)
ologitModel_weight

##### so the best model in ologitmodel which didnt do to the original data set smote,up or down sampling
pred9  = predict(ologitModel_weight ,newdata = train_data,type = "class")
(length(pred9))
(conf_matrix = confusionMatrix(pred9, train_data$quality))
```

```
################################################## down-sample model ##############################
set.seed(100)
best_model_down = regsubsets(quality~.,data = train_down, nvmax = 11)
best_model_down.summary = summary(best_model_down)
which.max(best_model_down.summary$adjr2)
par(mar = c(3, 3, 2, 2) + 0.05)
plot(best_model_up.summary$adjr2,xlab="Number of Variables",ylab="adjr2")
points(6,best_model_smote.summary$adjr2[6],pch=20,col="red")
coef(best_model_down,6)

ologitModel3 = polr(quality ~ volatile_acidity + residual_sugar +chlorides + pH + sulphates + alcohol ,data =train_down)
ologitModel3

pred7  = predict(ologitModel3,newdata = train_down,type = "class")
(length(pred7))
(conf_matrix = confusionMatrix(pred7, train_down$quality))

pred8  = predict(ologitModel3,newdata =test_data,type = "class")
(length(pred8))
(conf_matrix = confusionMatrix(pred8,test_data$quality))

#############################################################################
#                Best Subset Selection
#############################################################################
regfit.full<-regsubsets(quality~.,data=train)
summary(regfit.full)
#It gives by default best-subsets up to size 7;
#lets increase that to 19, i.e. all the variables
regfit.full<-regsubsets(quality~.,data=train, nvmax=11)
regfull.summary<-summary(regfit.full)
summary(regfit.full)
names(regfull.summary)
```

```
###Multinomial Logistic Regression All Explanatory Variables ##################

# Fit the multinomial logistic regression model
mlogitModel <- multinom(quality ~. ,data =train, maxit = 1000)
mlogitModel

# Calculate VIF values

labels = rownames(coefficients(mlogitModel))
ref = setdiff(mlogitModel$lab,labels)
t(sapply(labels,function(i){
  train$quality=as.numeric(train$quality==i)
  vif(glm(quality ~ .,data=train,family="binomial"))
}))

# determine the value of K
k=which.max(valid_acc)
k
#k=1 gives the hightest validation accuracy
#recalculating the training and validation accuracies for k=10 model

set.seed(1)
train_pred=knn(train, train, outcome_train, k,prob = TRUE,use.all=TRUE)
train_acc=mean(train_pred == outcome_train)

set.seed(1)
valid_pred = knn(train, valid, outcome_train, k,prob = TRUE,use.all=TRUE)
valid_acc = mean(valid_pred == outcome_valid)

#Test Accuracy when k=10
set.seed(1)
test_pred=knn(train,test, outcome_train, k,prob = TRUE,use.all=TRUE)
test_acc=mean(test_pred == outcome_test)


cat('Training Accuracy:   ', train_acc, '\n',
    'Validation Accuracy: ', valid_acc, '\n',
    'Test accuracy: ', test_acc, sep='')

T=table(outcome_test)
T
# Confusion Matrix
S=confusionMatrix(test_pred,outcome_test,mode = "everything")
S
```

```
######################################## optimal tree 1000 ################################
set.seed(100)
# Fitting Random Forest to the train dataset with 1000 trees
Optimal_Model= randomForest(x = smoteTrain[-12],y = smoteTrain$class, ntree = 1000, mtry =Min)
print(Optimal_Model)
predict_train = predict(Optimal_Model)


#############################################################################################
om_pred=predict(Optimal_Model,newdata=test)
Z=confusionMatrix(om_pred,test$quality, mode='everything')
Z

#Weighted F1 Score
s = as.matrix(confusionMatrix(as.factor(om_pred),test$quality))
n = sum(s) # number of instances
s
n
nc = nrow(s) # number of classes
rowsums = apply(s, 1, sum) # number of instances per class
colsums = apply(s, 2, sum) # number of predictions per class
colsums
diag = diag(s)  # number of correctly classified instances per class
diag
precision = diag / colsums
precision
recall = diag / rowsums
recall
f1 = 2*precision * recall / (precision + recall)
f1
```

```
############################################ 500 trees ################################
#fitting random forest to the train data
set.seed(100)
rf_model <- randomForest(x=smoteTrain[-12],
                         y =smoteTrain$class,
                         ntree = 500)

# Print the summary of the model
print(rf_model)
```

```
#with 1000 trees
rf_model1 <- randomForest(x=smoteTrain[-12],y =smoteTrain$class, ntree = 1000)
print(rf_model1)

#############################################################################################
rf1_pred=predict(rf_model1,newdata=test[-12])
#confusion matrix
A=confusionMatrix(rf1_pred,test$quality,mode = "everything")
A
```

```
#error rate dataframe for all the trees
oob_error1=data.frame(
  Trees=rep(1:nrow(rf_model1$err.rate), 4),
  Type=rep(c("OOB","Poor","Normal","Excellent"), each = nrow(rf_model1$err.rate)),
  Error=c(rf_model1$err.rate[,"OOB"], rf_model1$err.rate[,"Poor"],
          rf_model1$err.rate[,"Normal"],rf_model1$err.rate[,"Excellent"]))

ggplot(data = oob_error1, aes(x = Trees, y = Error, color = Type)) + geom_line() +labs(x = "Number of
```

```
#testing accuracy of model

oob_values=numeric()
for(i in 1:11){
  model2=randomForest(x = smoteTrain[-12],
                      y = smoteTrain$class,
                      mtry = i,
                      ntree = 1000)

  oob_values[i]=model2$err.rate[nrow(model2$err.rate),1]
}
plot(oob_values)
min(oob_values)
Min=which.min(oob_values)
Min
```

```
########################k fold cross validation ########################################
library(caret)
# Define the training control
ctrl <- trainControl(method = "cv", number = 5)  # 5-fold cross-validation
# Define the range of mtry values to test
mtry_grid <- expand.grid(mtry = c(2, 3, 4, 5))  # Example values, adjust as needed
# Get the best mtry value
best_mtry <- rf_grid$bestTune$mtry
######### mtry value using k fold cross validation
# Check variable names and presence in the dataset
all_vars <- all(names(smoteTrain) %in% c("class"))  # Include all predictor variable names
if (!all_vars) {
  missing_vars <- setdiff(c("class"), names(smoteTrain))
  print(paste("Missing variables:", missing_vars))
}
# Verify formula
print(formula(smoteTrain$class ~ .))
# Perform grid search
rf_grid <- train(
  class ~ .,
  data = smoteTrain,
  method = "rf",
  trControl = ctrl,
  tuneGrid = mtry_grid
)
best_mtry <- rf_grid$bestTune$mtry
print(best_mtry)
#####################################################################
#best_mtry=4
```

```
############## outlier analysis ################

########## DBSCAN ##################
eps = 0.5
minPts = 5

# Apply DBSCAN

result = dbscan(train_data[-12], eps = eps, MinPts = minPts)
outliers_db <- train_data[result$cluster == -1, ]
view(outliers_db)
count_outliers <- nrow(outliers_db)
print(count_outliers)

###################### common outliers ################
detect_common_outliers <- function(data) {
  outliers_list <- list()  # Create a list to store outliers for each column

  # Iterate over each column in the dataset
  for (i in 1:ncol(data)) {
    if (is.numeric(data[[i]])) {  # Check if the column is numeric
      Q1 <- quantile(data[[i]], 0.25)  # Calculate the first quartile (25th percentile)
      Q3 <- quantile(data[[i]], 0.75)  # Calculate the third quartile (75th percentile)
      IQR <- Q3 - Q1  # Calculate the interquartile range (IQR)
      threshold <- 1.5  # Set the threshold for outlier detection (1.5 times the IQR)

      # Identify outliers using the IQR method and store their indices
      outliers_list[[i]] <- which(data[[i]] < (Q1 - threshold * IQR) | data[[i]] > (Q3 + threshold * IQR))
    }
  }

  # Find the common outliers across all columns
  common_outliers <- Reduce(intersect, outliers_list)

  return(common_outliers)
}
```

```
(outliers_com = detect_common_outliers(train_data))
length(outliers_com)
(length(data_set)/length(outliers_com))*100  #outlier percentage

#For each predictor###############

fit <- polr(quality ~ ., data = train_data)

# Get the predictor variables
predictor_variables = names(train_data)[!names(train_data) %in% "quality"]  # Exclude the response variable

par(mar=c(1, 1, 1, 1))  # Adjust the margin values as needed

# Create boxplots for each predictor variable
for (predictor in predictor_variables) {
  boxplot(train_data[[predictor]], main = predictor, xlab = predictor, ylab = "Value")
}

#################################################

detect_outliers = function(data) {
  outliers = numeric()
  for (i in 1:ncol(data)) {
    if (is.numeric(data[[i]])) {  # Check if the column is numeric
      Q1 = quantile(data[[i]], 0.25)
      Q3 = quantile(data[[i]], 0.75)
      IQR = Q3 - Q1
      threshold = 1.5
      outliers = c(outliers, which(data[[i]] < (Q1 - threshold * IQR) | data[[i]] > (Q3 + threshold * IQR)))
    }
  }
  unique(outliers)
}
```

```r
# Detect outliers in dataset ##################
outliers = detect_outliers(train_data)
length(outliers)
(length(data_set)/length(outliers))*100  #outlier percentage
#train_data
# Detect outliers in trasnformed dataset ####################

### use log10 transformation to skewd prdictors and create a new data set ######
transformed_dataset <- train_data %>%
  mutate(
    `citric acid` = log10(`citric acid`),
    `residual sugar` = log10(`residual sugar`),
    chlorides = log10(chlorides),
    `free sulfur dioxide` = log10(`free sulfur dioxide`),
    `total sulfur dioxide` = log10(`total sulfur dioxide`),
    alcohol = log10(alcohol)
  )
view(transformed_dataset)

outliers_tr = detect_outliers(transformed_dataset)
length(outliers_tr )
(length(transformed_dataset)/length(outliers_tr ))*100  #outlier percentage
```

```r
### use sqrt transformation to skewd prdictors and create a new data set ##############
transformed_dataset2 = train_data %>%
  mutate(
    `citric acid` = sqrt(`citric acid`),
    `residual sugar` = sqrt(`residual sugar`),
    chlorides = sqrt(chlorides),
    `free sulfur dioxide` = sqrt(`free sulfur dioxide`),
    `total sulfur dioxide` = sqrt(`total sulfur dioxide`),
    alcohol = sqrt(alcohol)
  )

view(transformed_dataset2)
outliers_tr2 = detect_outliers(transformed_dataset2)
length(outliers_tr2 )
(length(transformed_dataset2)/length(outliers_tr2 ))*100  #outlier percentage
```

R codes: https://drive.google.com/drive/folders/1iwVtcFlICpjBeMmw0I23W_E1usFlfm16?usp=sharing

9. REFERENCES

1. https://www.gourmethunters.com/blog/en/find-out-more-about-wine-and-volatile-acidity/
2. https://www.statology.org/smote-in-r/
3. https://mda.tools/docs/pls--distances-and-outliers.html
4. https://rpubs.com/mbaumer/knn
5. https://scales.arabpsychology.com/stats/how-to-find-the-range-in-r-with-examples/
6. https://www.statology.org/ridge-regression-in-r/
7. https://www.kaggle.com/code/ssudeep/red-wine-quality-prediction-using-ridge-regression
8. https://www.researchgate.net/publication/320072358_Comparing_Ridge_and_LASSO_estimators_for_data_analysis
9. https://rpubs.com/esuess/RandomForest
10. https://stats.oarc.ucla.edu/r/dae/multinomial-logistic-regression/ - ordinal logistic regression model
11. https://morewinemaking.com/articles/testing_wine_must#:~:text=The%20typical%20pH%20range%20for%20red%20wines%20is%20between%203.5%20and%203.8 . : outlier details
12. https://morewinemaking.com/articles/testing_wine_must#:~:text=Optimally%2C%20the%20pH%20of%20a,in%20the%203.4%2D3.6%20range.
13. https://waterhouse.ucdavis.edu/whats-in-wine/fixed-acidity#:~:text=The%20predominant%20fixed%20acids%20found,2%2C000%20mg%2FL%20succinic%20acid. : acidity
14. https://www.rdocumentation.org/packages/ordinal/versions/2023.12-4/topics/clm2