Wine Quality: Leveraging Machine Learning Paradigms to Predict Wine Quality Based on Physiochemical Parameters

A Predictive Approach to Wine Classification Using CARTs and Random Forests

1. Introduction:

Winemaking dates back nearly 8,000 years, originating in Mesopotamia and ancient Georgia, where fruits and berries were mashed and fermented into alcoholic beverages. Over time, viticulture spread worldwide, evolving into both an art and a science, with classification systems developed based on subjective qualities like taste, aroma, and balance.

This study leverages Classification and Regression Trees (CARTs) and Random Forests (RFs) to model wine quality based on physiochemical data. Using a dataset by Cortez et al. (2009) on Portuguese *Vinho Verde* wine, the study classifies wine into two categories: low quality (ratings ≤5) and high quality (ratings ≥6). The models analyses various physiochemical features such as density, alcohol content, volatile acidity, and others. They are evaluated using standard performance metrics, including accuracy, F1-score, recall, and precision. Comparing CARTs and RFs provides insight into how machine learning can enhance traditional wine classification, leading to scalable, objective, and reliable automated wine rating systems whilst also providing key insights into what drives the quality of wine.

2. Data Cleaning and Exploratory Data Analysis:

Before building the two models selected, it is important to understand the dataset and clean it up to ensure the model is learns from the dataset as required. Exploratory Data Analysis (EDA) was initiated by viewing the two datasets for red and white wine respectively. The datasets were investigated for missing values, inaccuracies or incorrect datatypes, duplicates and data corruption. All value types were confirmed as numeric. The datasets were then merged, with two additional columns being created – one to specify wine type and the other to specify a quality label ("high" or "low" – this was done at the end of the cleaning and analysis process to minimize risks of encountering false duplicates and the like). Other than the addition of the *wine_type* and *quality_label* columns, the data was clean and required no further cleaning or modification. Various distribution plots of the different physiochemical properties were generated (see *Appendix,* Fig 2.1 to 2.12). A scatterplot matrix was generated for each of the wine types and the combined dataset, to get a better understanding of the distributions and interrelations of the various features.

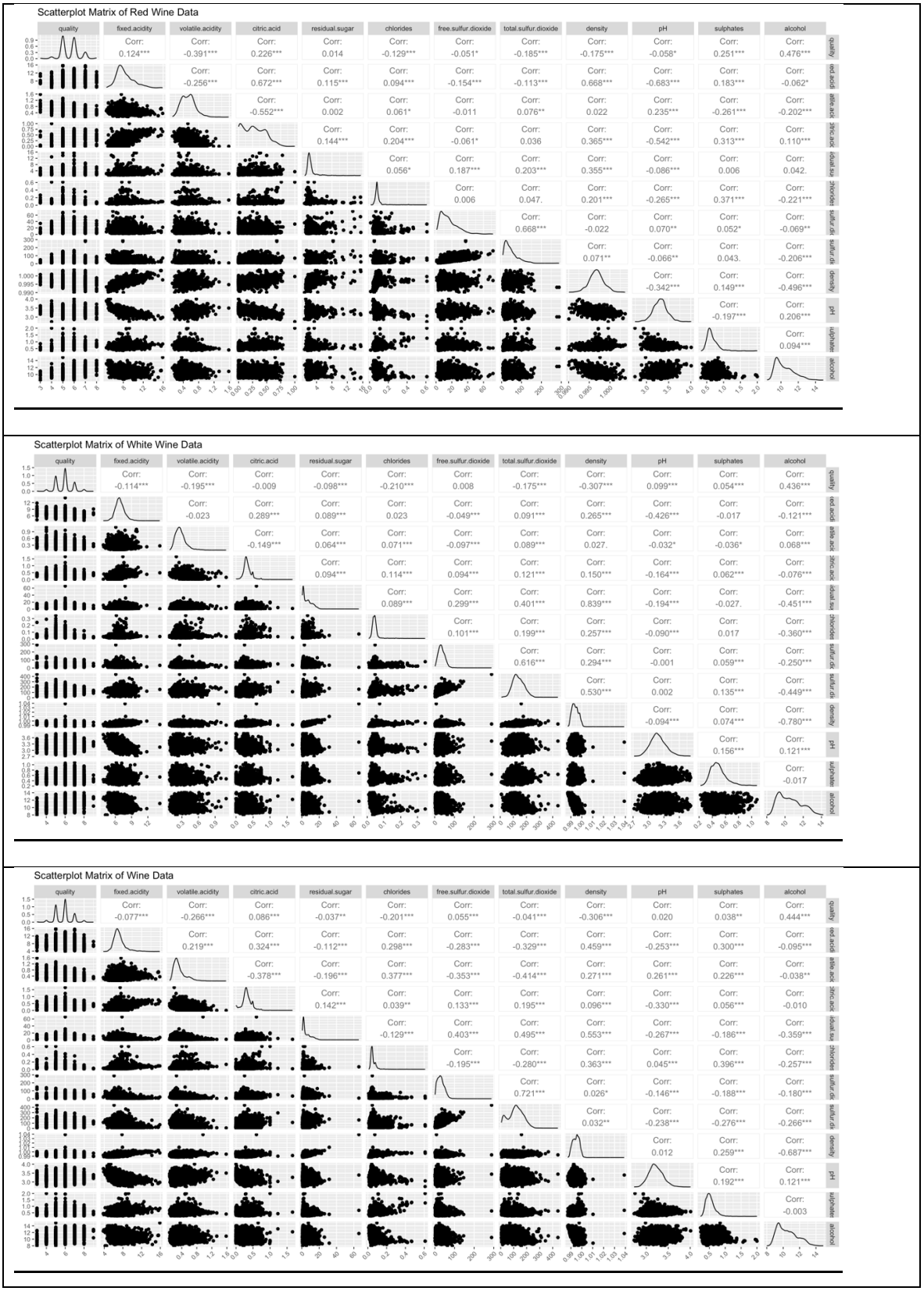


Fig 2.13 – Scatterplot Matrices of the red wine, white wine and combined wine datasets

Even before conducting a thorough analysis via ML models, it is already clear that *alcohol, density* and *volatile.acidity* will be key drivers for wine quality. Additionally, it was observed that multiple variables exhibited high correlation with one another indicating multicollinearity in the dataset, which must be addressed during model construction and validation. This is further visualized via a correlation heatmap.

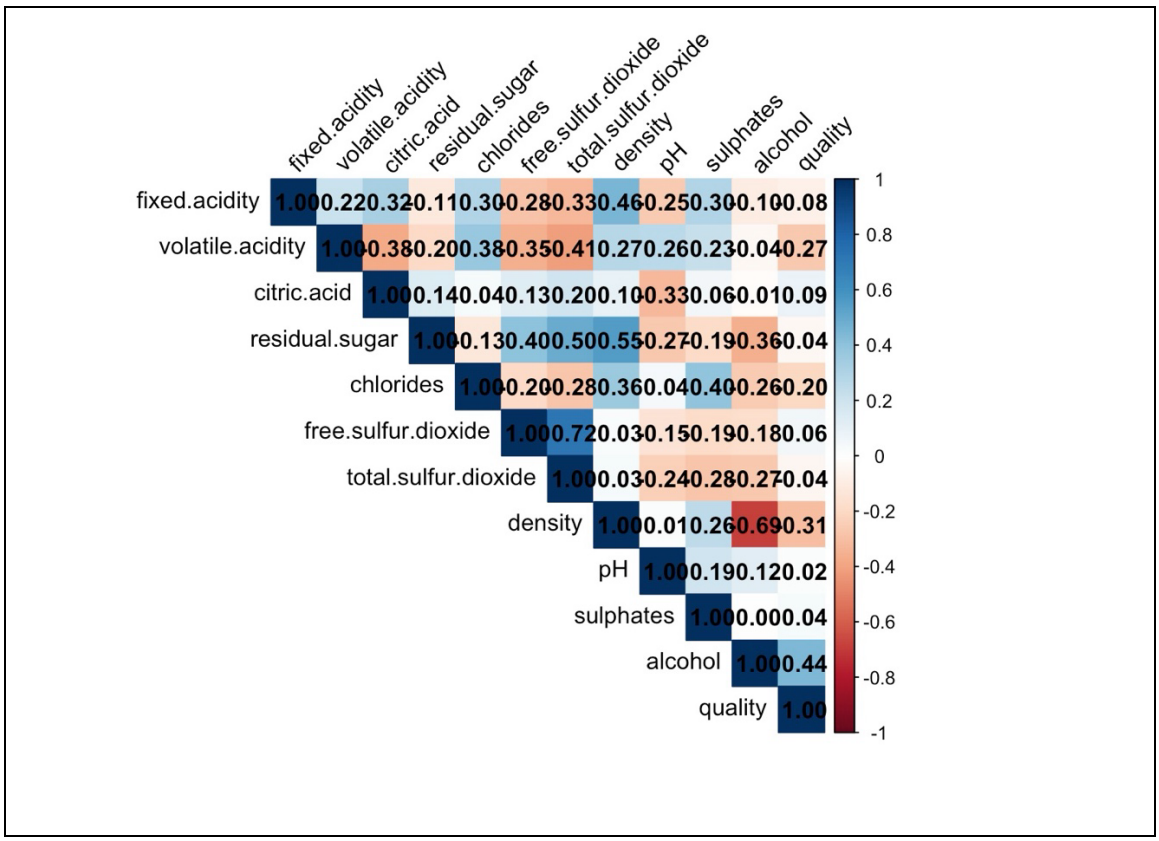


Fig 2.14 – Correlation heatmap

After viewing a brief density distribution for wine quality (colour-coded by wine type), the relationships between *alcohol, volatile.acidity* and quality were visualised (see *Appendix*, Fig 2.15 and 2.16).

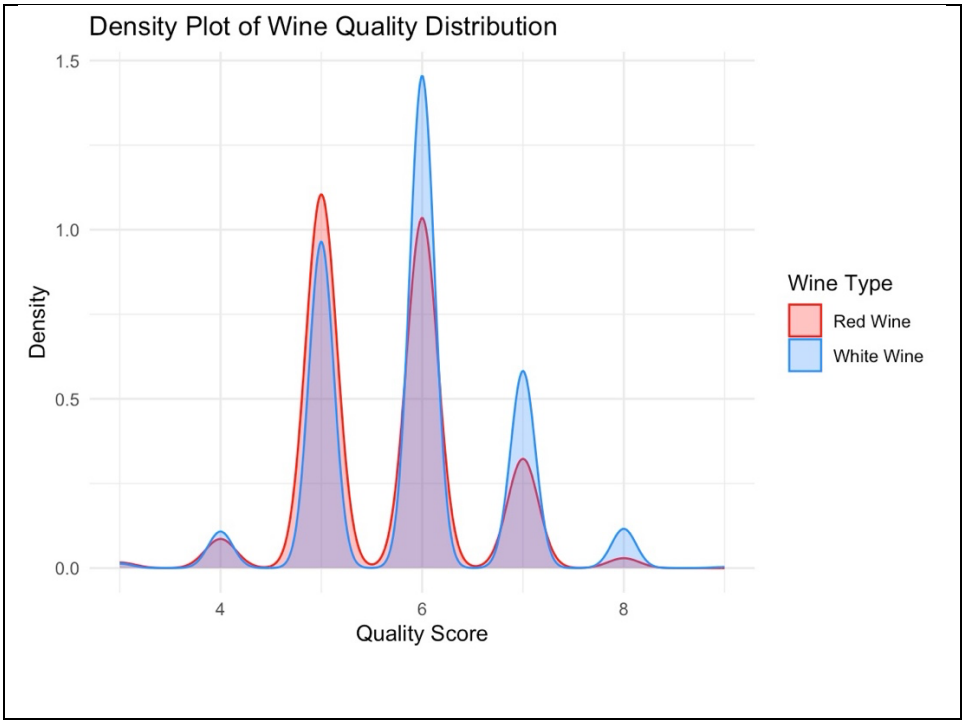


Fig 2.15 – Wine Quality Distribution, a density plot.

From the offset, white wines dominate the dataset, and there is a visible class imbalance with a larger percentage of "high" quality wines than "low" quality wines. Furthermore, white wine has a seemingly larger proportion of high-quality wines than red. Combined, the class imbalance issue is exacerbated – this will be addressed during model generation.

Fig 2.16 – Boxplot of quality, showing class imbalance.

3. Modelling:

The two Machine Learning models chosen for this analysis were CARTs and Random Forest. As the quality variable was ordinate (wine quality ratings on a simple scale of 1-10), the CART was built as a Classification Tree instead of a Regression Tree, breaking the wine quality ratings into two bins – wines rated less than 6 were classified as low-quality ("low") and wines rated 6 and above were rated as high-quality ("high"). To this end, an additional column named "*quality_label*" was added to the dataset, indicating which quality bin a particular sample of wine belonged to.

A Classification Tree and a Random Forest were chosen for two main reasons. Firstly, it was to do the exact problem at hand – the classification of wine on the basis of its quality rating. The target variable, while numeric, is categorical and non-continuous, which makes it a simple classification problem. Secondly, Classification Trees and Random Forests go hand-in-hand – in fact, a Random Forest is merely an aggregation of multiple Classification Trees, mitigating the weaknesses of a single tree (such as its proclivity to overfit the data and its inability to generalize well) whilst enhancing its strengths, accounting for a greater number of influential features and capturing more complex, non-linear relationships. Thus, a Classification Tree followed by a Random Forest was seen as the best option.

In the case of our wine dataset, a CT was trained on the combined wine dataset to model the various physiochemical properties of wine against its quality label, selecting whether it falls into one two categories or *labels* – "high" or "low". It works by splitting the dataset into subsets based on 'if-else' rules, creating a branching, tree-like structure, populated by splits called "nodes". At each node, it uses measures like Gini Impurity and entropy to decide how to best split the dataset further. The goal is to keep splitting the data and minimizing the Gini Impurity, until it cannot be reduced further. As an added note for reference, Gini impurity is defined as the probability of incorrectly classifying a randomly chosen element if it were randomly labelled according to the class distribution within the node, calculated as:

$$G(k) = 1 - \Sigma P(i)^2$$

Where *P(i)* is the probability of an element being present in the *i*-th class. At the end, once no more splits are possible (*i.e.* G(k) is reduced to near 0), it creates leaf nodes – the final predicted classes that the dataset can be placed in.

To build the CT, the *rpart()* function was used, with *quality_label* as the response and all variables except *quality* and *wine_type* as the predictors; *quality* was removed as it led to immense overfitting, with the tree "cheating" by learning the relationships instead of predicting

them, while *wine_type* was removed to improve accuracy and recall. In figure 3.1 below, we can see how the resultant CT (after pruning and optimization) creates a decision structure with nodes and branches.

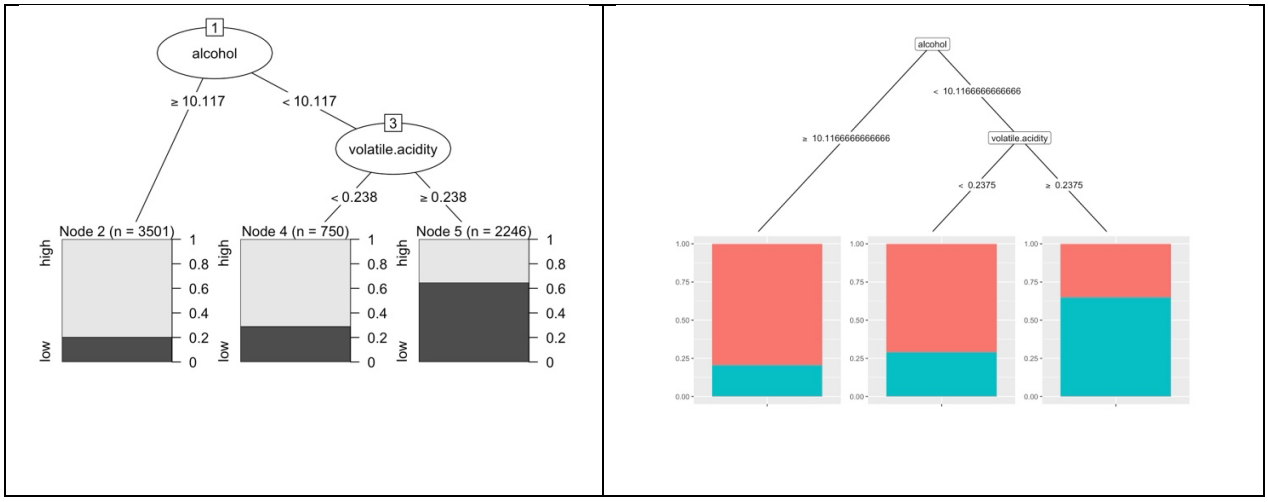


Figure 3.1 – Two plots showing the decision tree built by classifying the wine dataset as per "quality_label".

From these plots, we can see that the tree chooses two splits and divides the data up into 3 leaves or classes. The first variable that most affects the decision making is *alcohol*, which is a measure of the alcohol content of the wine. High alcohol wines are largely classified as high quality, with a smaller percentage classified as low quality. For wines with alcohol lower than a threshold value, the tree considers the next most important variable influencing wine quality classification, *volatile.acidity*, which is a measure of the gaseous acids (such ethyl acetate and acetic acid) present in the wine. Here, the algorithm splits the dataset into two leaves, classifying most of the high-quality wines into the leaf corresponding to lower volatile acidity, indicating that low levels of volatile acidity indicate higher wine quality. Volatile acids significantly impact wine's aroma, mouthfeel, and taste - key factors in subjective classification by connoisseurs and sommeliers. The classification tree suggests that while volatile acidity is important, lower levels are preferred (Vilela-Moura et al., 2010). This aligns with theories emphasising balanced acidity while preserving a wine's characteristic bouquet (Crespo et al., 2023).

The Classification Tree resulting from initial explorations was not entirely perfect. To fine tune and "prune" it so as to create a more definitive classification, the cost complexity of the resulting tree was considered using the *printcp()* function in conjunction with the *rpart()* function used to generate the tree. The cost complexity of a model is a trade-off between performance on training and validation data and its level of complexity. The *printcp()* produced a table of the best splits possible in our tree, shown below:

model	CP	nsplit	rel error	xerror	xstd	
1	0.14597	0	1.0000	1.0000	0.016296	
2	0.13255	1	0.85403	0.85864	0.015706	
3	0.0100	2	0.72148	0.72148	0.015021	*

Table 1 – CP Table for the Classification Tree

From the table above, our goal was to go for the model with the minimum value of *xerror* or cross-validation error, which in this case is model 3, with an *xerror* = 0.72148, the model with 2 splits (indicated with a "*" on the table). Thus, having used the cross-validation error as our model selection criteria, a model selector was initiated with goal of picking the model with lowest *xerror* from a CP table. The resulting pruned model was then tested on test data, and the model parameters were evaluated using a confusion matrix and a calculation of performance scores. The final pruned model did not significantly improve on the model pre-pruning, with accuracy and precision remaining the same, but other criteria were ever so slightly improved. More fine tuning was attempted by modifying the *rpart.control()* parameters, setting values for maxdepth, minsplit, minbucket and cp to control how deep the tree goes, the maximum number of splits per node, the maximum number of leaves, and the minimum required decrease in Gini Impurity. These procedures did not result in any change in the model, indicating that the model at the default settings for *rpart.control()* was already the best model after pruning via *printcp()*. The final pruned tree had an accuracy of 73.53%, a recall of 80.77%, a precision of 78.15% and an F1 score of 79.44%. From these results, we can conclude that the tree has a high accuracy and identifies both high- and low-quality wines well (due to good Precision, Sensitivity and F1 scores).

For our second model, a Random Forest was selected as our model of choice. An RF is an aggregation of single trees to create a forest (hence the name Random Forest). The Random Forest utilises techniques known as bootstrapping and bagging, where each tree is trained on random samples (or a "bootstrap sample") of the dataset (each tree learns slightly different patterns as a result) and the "majority vote", *i.e.* the prediction of the majority of the trees in the forest, is output as the final prediction (known as "bagging"). In the case of an RF made of Regression trees, bagging is done by averaging the outputs of all models to get the final output.

For the Random Forest, two models were produced – one Classification model, and one Regression model. This was done for experimental purposes, to compare the performances of both and confirm that the classification model is a better choice for the problem at hand. To train the model, the *randomForest()* function was used – this function contains all the necessary tools needed to apply bootstrapping and bagging to build the forest. First, the dataset was

randomly split into a training and a testing set. The classification forest was then built by treating the *quality_label* variable as the response variable while all the other variables were treated as predictors, save for the *quality* variable, which was excluded as including it resulted in a model that overfit – this follows the results obtained earlier with the CT, where the inclusion of the *quality* variable led to an inaccurate model that displayed extreme overfitting. After a rigorous testing and tuning phase, the Random Forest was fine-tuned by increasing the number of trees to 10,000 and setting the random sampling, mtry, to 3 after a 5-fold cross validation where the lowest OOB (Out-Of-Bag) error rate was found at an mtry value of 3. This resulted in the best model as per evaluative statistics, achieving an accuracy of 82.8% and an OOB error rate of 17.75%. Precision and recall came in at 84.1% and 89.24% respectively. This showed that the model did a good job of classifying "high" and "low" quality wines with very low error rates. However, there is still some weakness to the model as its Specificity of 72.19% showed that there is a risk of misclassifying a small portion of the low-quality wines as high quality by accident. This is largely due to the issue of class imbalance, as there is a much higher quantity of high quality wines in the dataset than low quality wines – attempts were made to address this issue by setting class weights, modifying the *classwt* parameter in the *randomForest()* function to compensate for the underrepresented classes. Furthermore, undersampling of the majority class was also attempted. However, neither of these attempts led to any change in the model evaluative parameters.

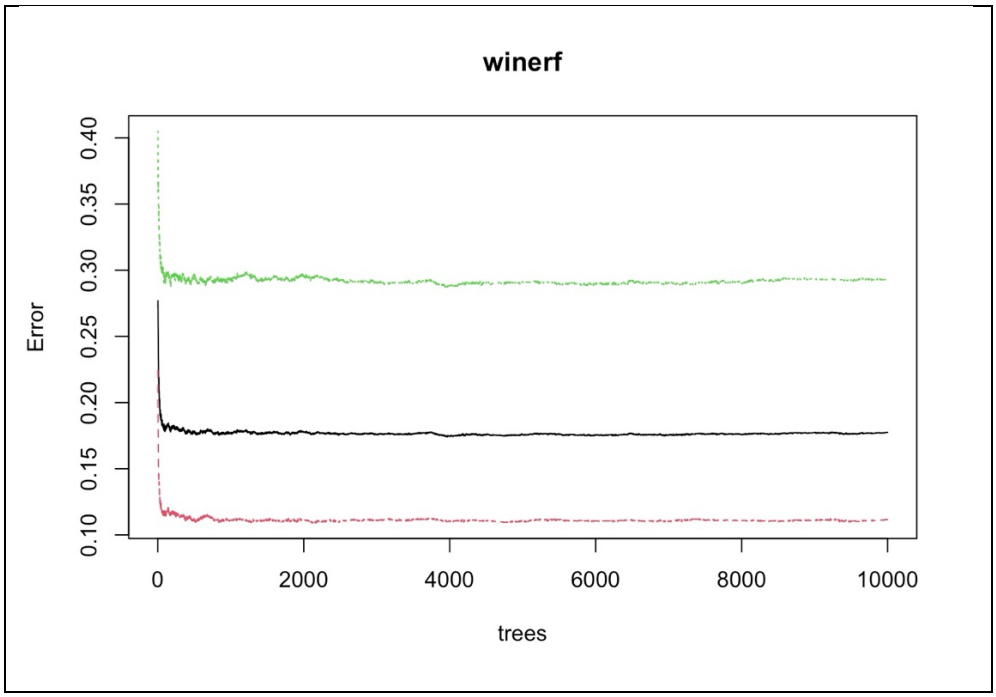


Figure 3.2 – Plot of the error rate of the winerf Random Forest vs. number of trees in the forest

The figure above shows how the error rate for our model reduces as the number of trees are increased, becoming flatter and flatter as we approach our *ntree* parameter of 10,000 trees.

While this approach has allowed us to minimize error rates and generalize the dataset better, it is worth mentioning that our parameter of 10,000 trees is computationally expensive.

A deeper evaluation of our RF was done by an MDS plot, which showed us how the model classified and grouped the data, and a variable importance plot, showing which variables were the key drivers for the predictions made by the model.

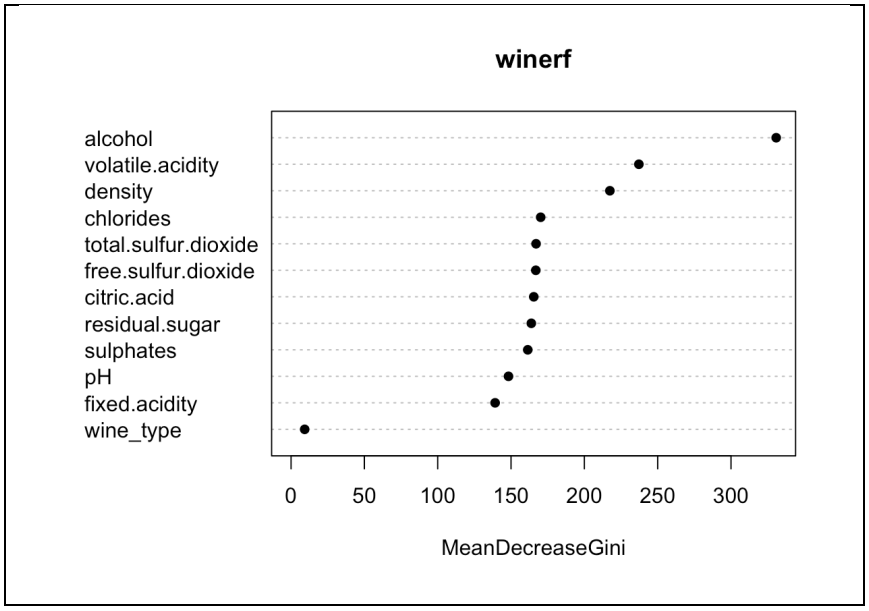


Figure 3.3 – Variable Importance Plot for the classification Random Forest

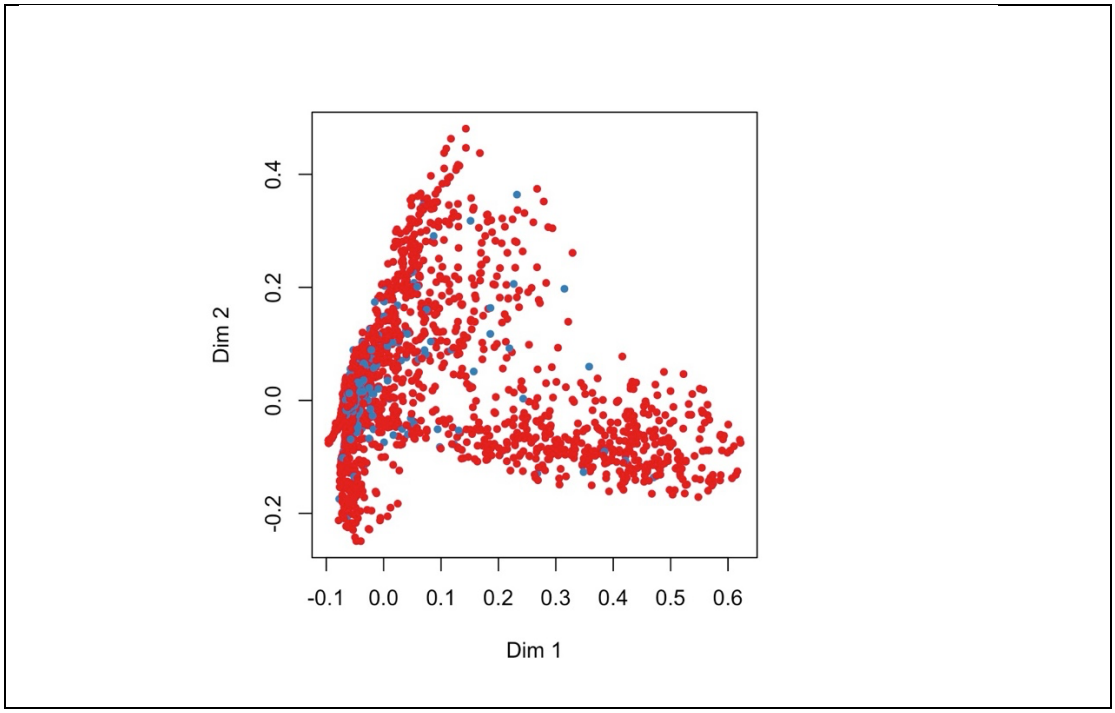


Figure 3.4 – Multi Dimensional Scaling Plot of the Random Forest

The classification Random Forest showed that in addition to *alcohol* and *volatile.acidity*, *density* was also a key factor in determining wine quality. These findings were explored further by plotting the interrelationships between these variables (see fig 3.5 to fig 3.7 in th), showing the wines with low density, low volatile acidity and high alcohol content tend to be rated highly.

Initial predictive explorations using Random Forests treated the design of the model as classification model. However, for comparative purposes, a Regression Forest was also designed. To do this, the response variable was *quality*, treating the ratings as continuous values between 1 and 10, while the other features were treated as response. No features were excluded. A study of the model accuracy and the predictions graph (fig. 3.8 and fig. 3.9 in the *Appendix)* revealed that this model did not do as well as the classification model. As such, due to its lower accuracy (77.8%) than the classification tree, it was rejected as the model of choice for the study.

4. Model Comparison:

Classification Trees and Random Forests go together – indeed, it would be fair to say that a Random Forest is merely an expansion of the Classification Tree, combining several such trees to form a "forest". From a theoretical perspective, when comparing the two models, while a singular tree is easier to visualize and interpret, an RF is a far better method as its ability to aggregate many such trees means that it doesn't overfit as much, generalizes the data and accounts for complex relationships better due to bootstrapping and bagging, and is more robust and stable. Looking at the raw data, the RF performed as expected, far exceeding the decision tree, visualized in the table below.

Metric	Classification Tree	RandomForest (Classification)
Accuracy	73.5%	82.8%
Kappa	0.42	0.63
Balanced Accuracy	70.9%	80.71%
Precision	78.15%	84.10%
Recall	80.77%	89.24%
Specificity	61.03%	72.19%

Table 2 – Comparison of CT and RF models

These statistics make it clear that RF model far outperforms the singular classification tree. The higher accuracy score shows that the RF generalizes the data much better than a tree. This is a direct validation of the idea of bagging and bootstrapping – by training the trees on random samples of data and aggregating them, the RF can mitigate any chances of overfitting or any risk of variance. Furthermore, higher Precision, Recall and Specificity scores indicate that the model captures more complex non-linear relationships, picking out higher and lower quality wines much better than the CT which is unable to capitalize on these. The numbers also indicate that the RF model is more robust and handles class imbalance (a particular problem in this dataset) much better than a single tree as it can pick out the minority class ("low") much better, indicated by its superior balanced accuracy. A final note on the model comparison relates to variable importance - while the decision tree chose two variables (*alcohol* & *volatile.acidity*) as the most important variables, the RF added a third variable, *density*, as a variable driving quality rating. The RF is a much better choice as a model, due to its superior performance; however, CTs are easier to interpret and less computationally expensive, offering a quick and simple method to perform a preliminary predictive analysis.

5. Results and Conclusion:

The performance of the classification tree and random forest models highlights the strengths and trade-offs of different machine learning approaches. The classification tree achieved 73.5% accuracy, with lower specificity (61.03%), indicating difficulty in correctly classifying low-quality wines. In contrast, the random forest significantly outperformed the classification tree, achieving 82.8% accuracy, with higher sensitivity (89.24%) and specificity (72.19%), making it more reliable for classification. The ensemble approach of random forests reduced overfitting, leading to better generalization.

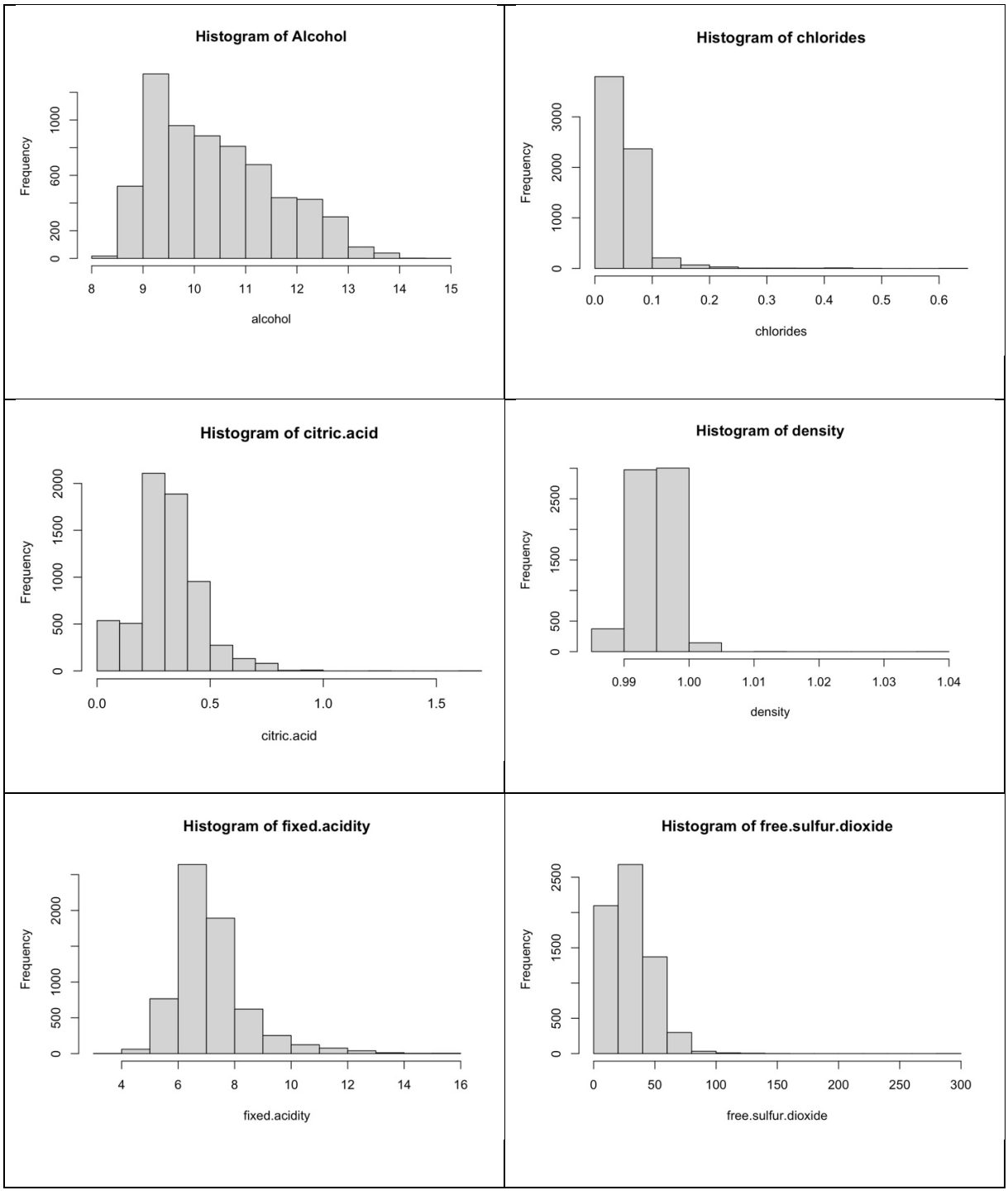
Despite performance improvements, several limitations exist. The dataset is limited to Portuguese Vinho Verde wines, which may affect generalizability to other wine types. Additionally, strong correlations between features (e.g., alcohol content and density) may influence model performance. The binary classification of wine quality (low vs. high) removes granularity, limiting nuanced quality distinctions. Finally, the random forest model, while highly accurate, lacks interpretability, making it harder to extract simple decision rules.

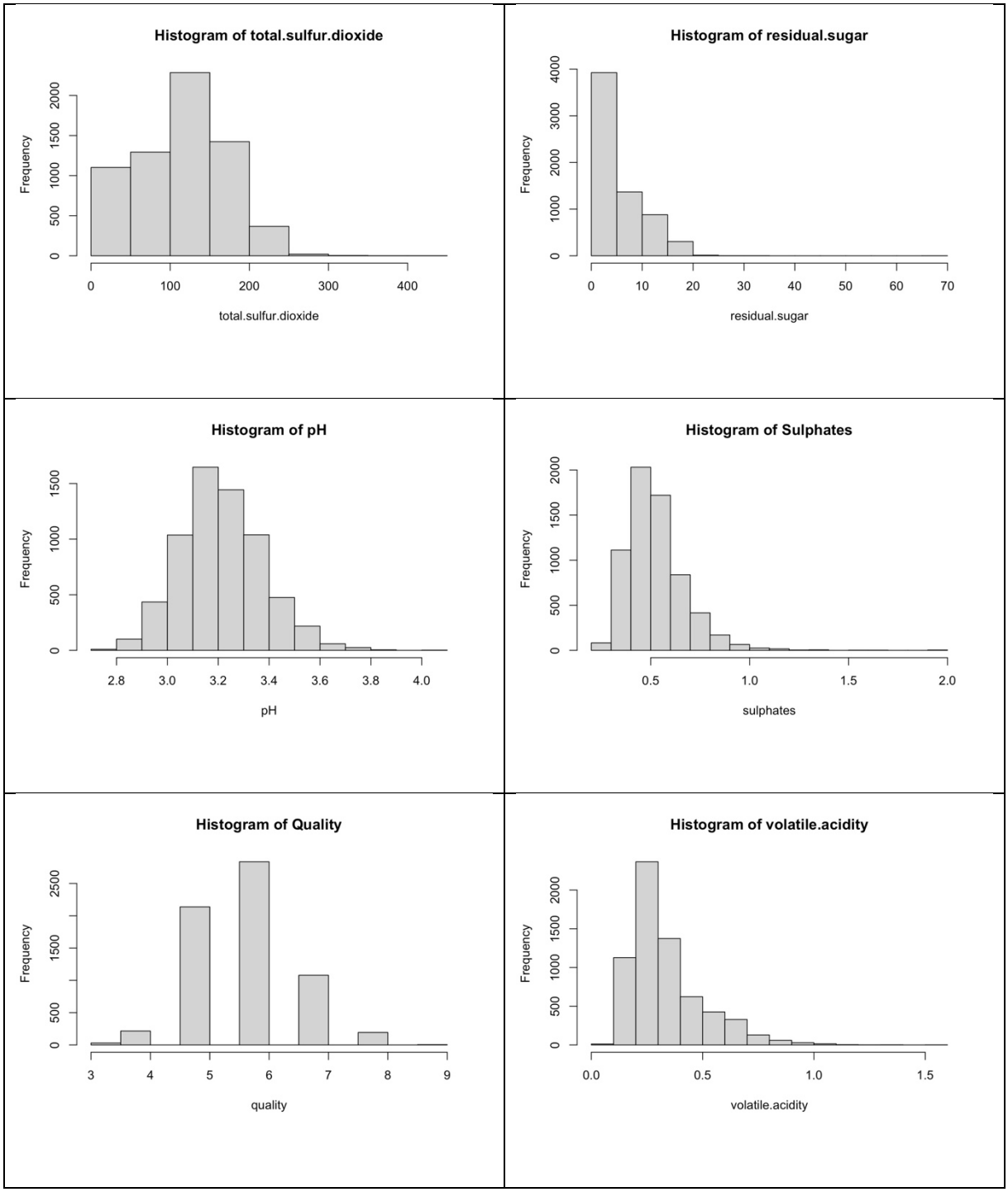
To enhance model performance, expanding the dataset to include diverse wine types would improve generalization. Applying feature engineering techniques like Principal Component Analysis (PCA) could reduce redundancy among correlated variables. Additionally, alternative models such as Neural Nets or Ridge/Lasso/Elastic-Net regression or other ensemble methods may further enhance classification accuracy. Addressing class imbalance through SMOTE could improve the prediction of minority classes. Future studies could also explore a multi-class classification approach to better reflect varying quality levels, instead of a binary high-low classification.

Bibliography:

1. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Wine Quality [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C56S3T.
2. Cortez, P., Cerdeira, A.L., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst., 47*, 547-553.
3. Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: springer.
4. Vilela-Moura, A, Côrte-Real, M, Sousa, MJ, Mendes-Faia, A, Schuller, D, Chaves, SR, & Silva, RD, 2010, 'The impact of acetate metabolism on yeast fermentative performance and wine quality: reduction of volatile acidity of grape musts and wines' *Applied Microbiology and Biotechnology*, vol. 89, no. 2, pp. 271–280, doi: 10.1007/s00253-010-2898-3.
5. Gambetta, JM, Jeffery, DW, Schmidtke, LM, Cozzolino, D, Wang, J, & Bastian, SEP, 2016, 'Relating Expert Quality Ratings of Australian Chardonnay Wines to Volatile Composition and Production Method', *American Journal of Enology and Viticulture*, vol. 68, no. 1, pp. 39–48, doi: 10.5344/ajev.2016.16058
6. Crespo, J, García, M, Arroyo, T, Romero, V, & Cabellos, JM, 2023, 'Influence of Native Saccharomyces cerevisiae Strains on Malvasia aromatica Wines.', *Frontiers in Bioscience-Elite*, vol. 15, no. 3, p. 18, doi: 10.31083/j.fbe1503018.
7. Gislason, PO, Sveinsson, JR, & Benediktsson, JA, 2004, 'Random forest classification of multisource remote sensing and geographic data', vol. 2, pp. 1049–1052, institute of electrical electronics engineers, doi: 10.1109/igarss.2004.1368591.
8. Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.
9. Supriatna, DJI, Saputra, H, & Hasan, K, 2023, 'Enhancing the Red Wine Quality Classification Using Ensemble Voting Classifiers', *Infolitika Journal of Data Science*, vol. 1, no. 2, pp. 42–47, doi: 10.60084/ijds.v1i2.95.

APPENDIX


Histogram of Alcohol
Frequency
0
200
600
1000
8
9
10
11
12
13
14
15
alcohol
Histogram of chlorides
Frequency
0
1000
2000
3000
0.0
0.1
0.2
0.3
0.4
0.5
0.6
chlorides
Histogram of citric.acid
Frequency
0
500
1000
1500
2000
0.0
0.5
1.0
1.5
citric.acid
Histogram of density
Frequency
0
500
1500
2500
0.99
1.00
1.01
1.02
1.03
1.04
density
Histogram of fixed.acidity
Frequency
0
500
1000
2000
4
6
8
10
12
14
16
fixed.acidity
Histogram of free.sulfur.dioxide
Frequency
0
500
1500
2500
0
50
100
150
200
250
300
free.sulfur.dioxide



Figures 2.1 – 2.12 – Histograms of variable distributions.

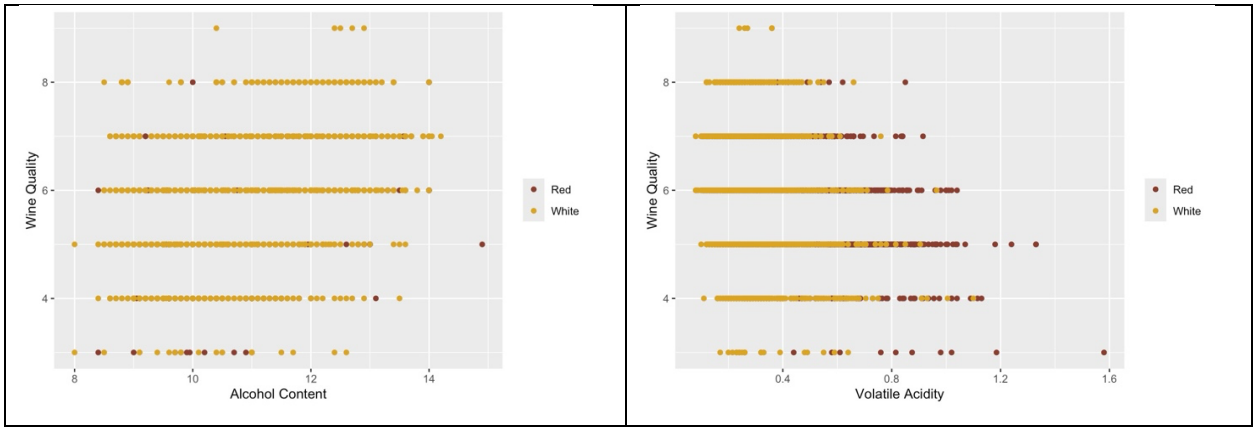
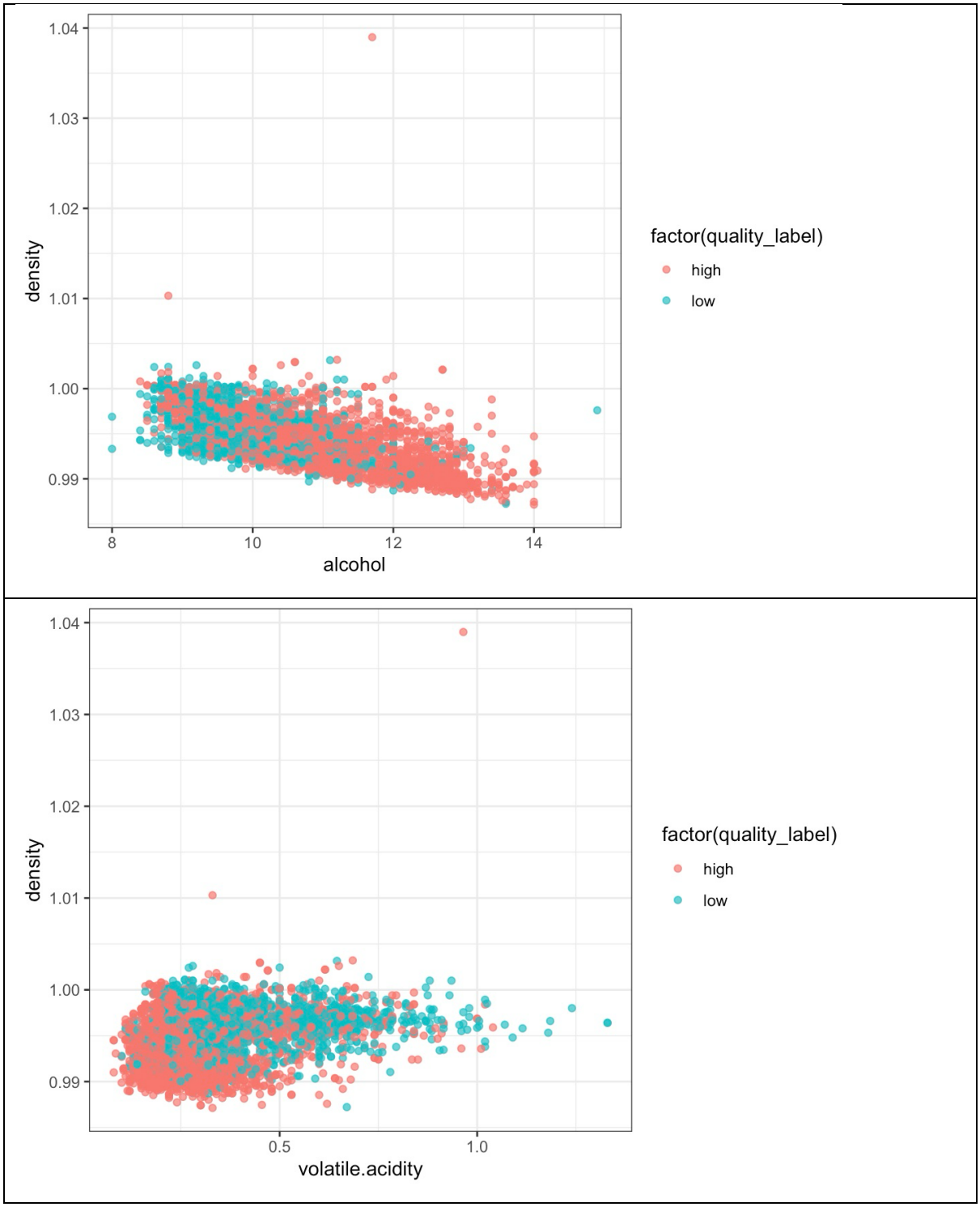
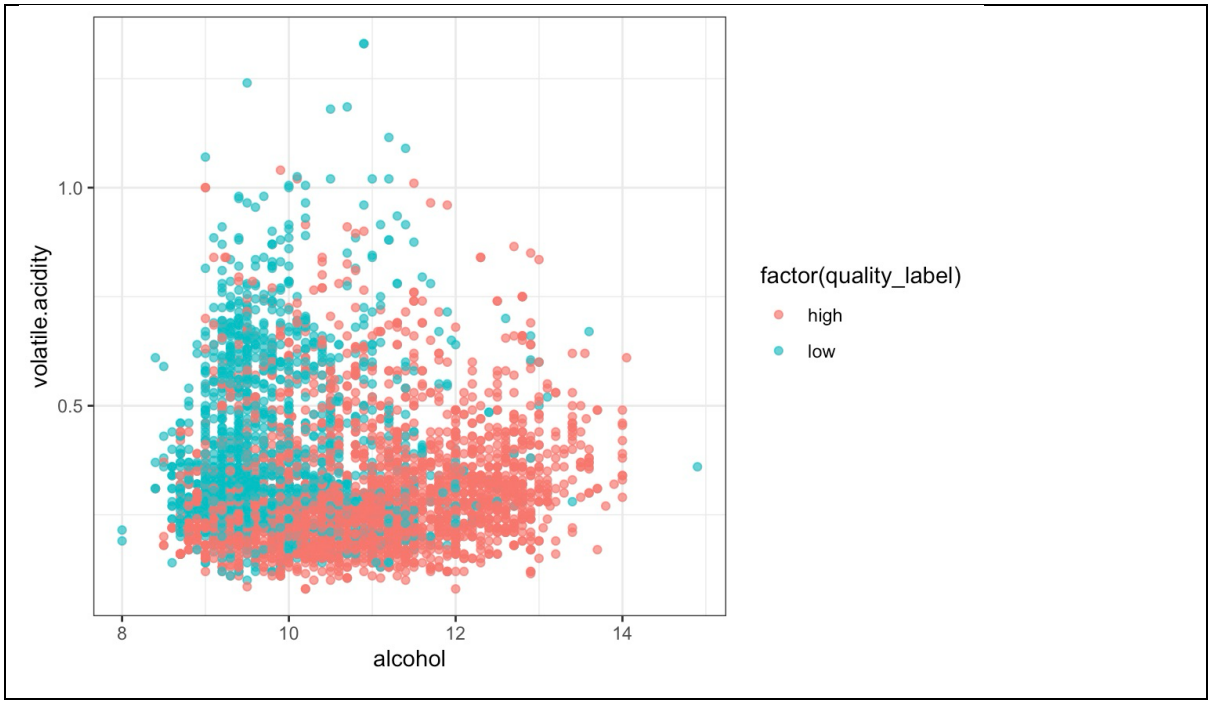


Figure 2.15 & 2.16 – Quality as related to alcohol content and volatile acidity


1.04
1.03
1.02
1.01
1.00
0.99
density
8
10
12
14
alcohol
factor(quality_label)
high
low
1.04
1.03
1.02
1.01
1.00
0.99
density
0.5
1.0
volatile.acidity
factor(quality_label)
high
low



Figures 3.5, 3.6, & 3.7 – Comparisons of the three most important variables driving the RF.

```
Call:
 randomForest(formula = quality ~ ., data = train_wine, ntree = 1000,      mtry = 3)
               Type of random forest: regression
                     Number of trees: 1000
No. of variables tried at each split: 3

          Mean of squared residuals: 0.1662728
                    % Var explained: 78
```

Fig 3.8 – Regression Random Forest stats

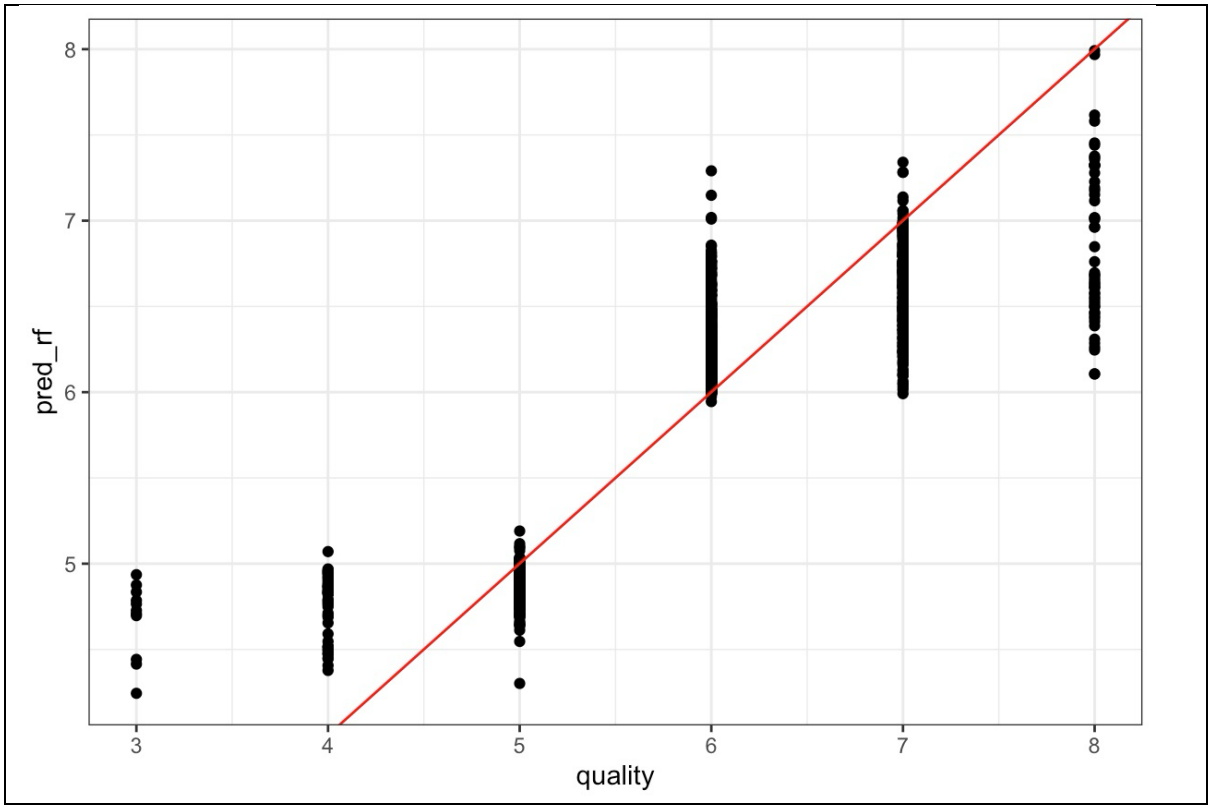


Fig 3.9 – Regression RF predictions vs actual values

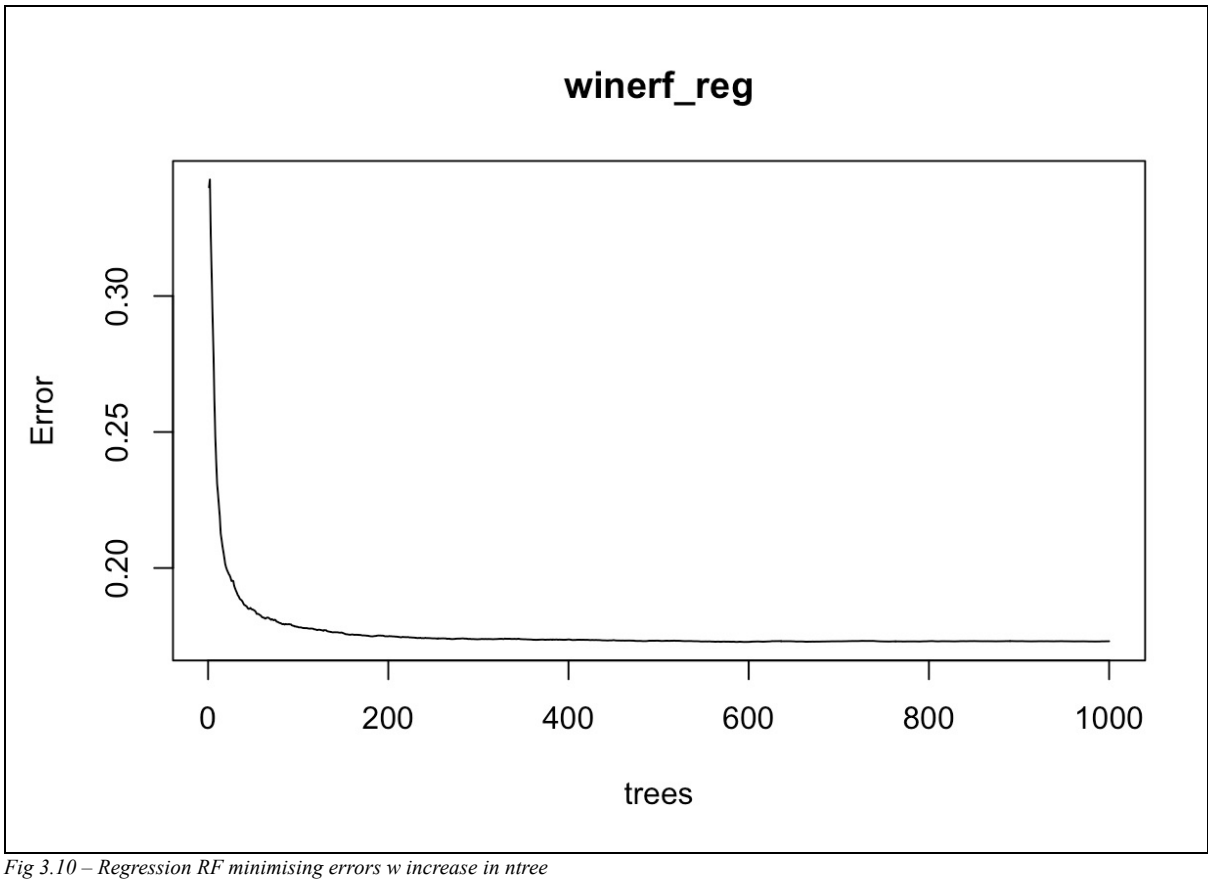


Fig 3.10 – Regression RF minimising errors w increase in ntree