Wine Quality

David Dyer - 1786617

April 2020

1 Introduction

In this project, we will be using 'Wine Quality Data Set' which is based on red and white variants of the Portuguese "Vinho Verde" wine (P. Cortez and Reis, 2009). We will be using this data set to see if we can predict wine quality based on several physicochemical properties of the wine, such as fixed acidity and alcohol level. Each wine has a quality rating which is from 0 (the worse) to 10 (the best).

2 Pre-processing

The 'Wine Quality Data Set' has two data sets: one for white wine (consisting of 4,898 wines) and another for red wines (consisting of 1,599 wines). We will merge these into one data set consisting of 6,497 wines. We will add type as another variable to the combined data set, assigning a value of 0 for white wines and 1 for red wines.

To run either regression or classification techniques, we need to check our data set to make sure that there is no missing data. As the data set is so large, it would take a considerable amount of time to check ourselves. Instead, we can use the 'visdat' package to check that our data does not have any missing (NA) values.

Figure 1: Missing values test

As can be seen from Figure 1 there are no missing values so we can proceed to analyse the data.

3 Data Analysis

Our first step is to see if any of the variables are highly correlated with each other. If variables are highly correlated with each other, then this suggests multicollinearity is present, which would affect the quality of our results. To investigate if there are any problems we can produce a correlation matrix (James et al., 2017).

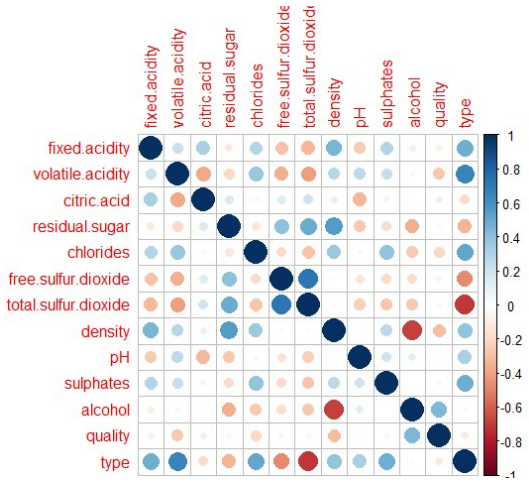

Figure 2: Correlation between variables

As seen from Figure 2, we can see that there is correlation between some variables. For example, density and alcohol have a correlation of -0.687 (3 dp) and total sulfur dioxide and residual sugar have a correlation of 0.459 (3 dp).

Next, we can have a look at how quality varies due to the other predictor values. First, we can view the overall distribution of the wines based on their quality.

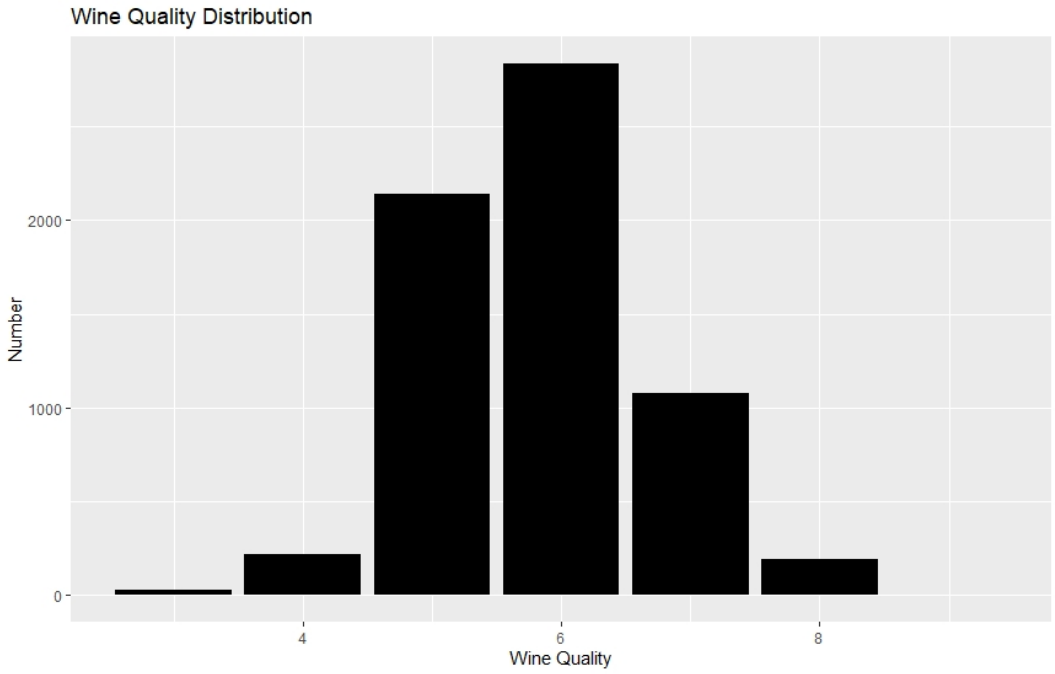

Figure 3: Wine quality distribution

From Figure 3, we can see that no wines are rated lower than a 3 or higher than an 8. We can also see that the mode quality is 6 and that the data is approximately symmetric due to it having a skewness value of 0.1895 (4 d.p.). Next, we can use scatter plots to see the relationship between quality and other variables.

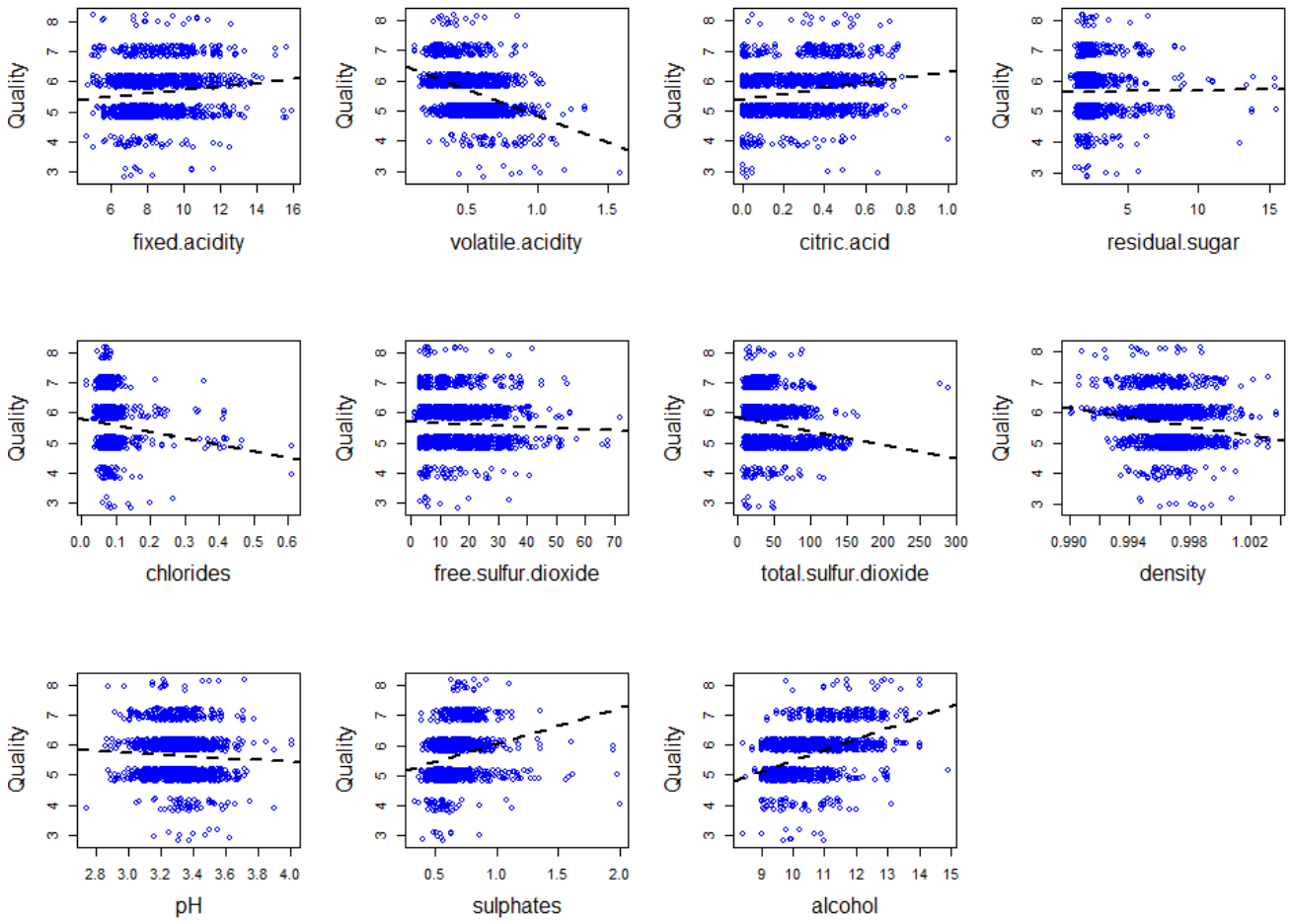

Figure 4: Wine quality compared to other variables

From Figure 4, it is evident that there are some clear correlations between wine quality and some of the other physicochemical properties. For example, quality seems to be positively correlated with alcohol (Pearson rank of 0.4443) but negatively correlated with volatile acidity (Pearson rank of -0.2657) and chlorides (Pearson rank of -0.2007). Due to the correlation existing between these variables and quality, it means these properties will play a significant role in the classification models.

3.1 Classification - Decision Trees

To produce our models, we will randomly split our data into a training set and a validation set using an 80:20 ratio.

Due to the likely presence of multicollinearity (predictors being correlated with other predictors) in our data, as seen in Figure 2, we will use classification instead of regression. The reason for this is that one of the assumptions for linear regression is that there is no multicollinearity. First, instead of the quality rating of 0 to 10, we will instead label wines that have a rating of above 5

as good and the rest as bad. Splitting the wines into these two classes, in our training set we end up with more good wines than bad as can be seen in Figure 5.

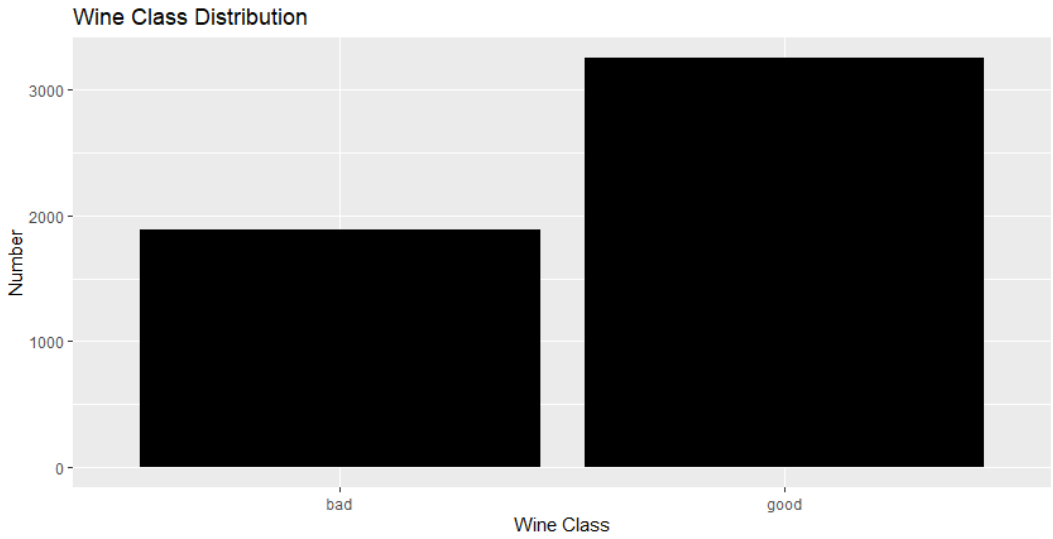

Figure 5: Wine class distribution

The first classification technique we will use is a classification decision tree. We will use our training set to create the tree and use the complexity parameter value of 0.01 (which is the default value in r).

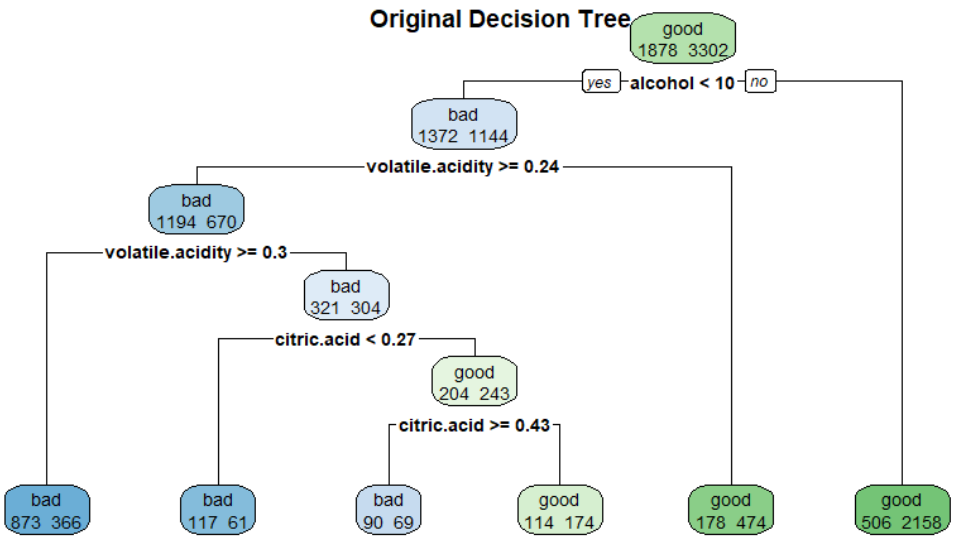

Figure 6: Decision tree

As can be seen in Figure 6, the decision tree uses 3 variables (alcohol, volatile acidity and citric acid) to classify if a wine is good or bad. By adding up the correctly classified wines at each terminal node of our original decision tree (Figure 6), we see that the tree classifies the training

set's wines correctly 75.02% of the time. To test the quality of our decision tree, we will use our test data test. We will see how many of the wines in the testing set are correctly classified by our decision tree. A way of visualising this is by creating a confusion matrix.

Table 1: Confusion matrix for original decision tree

	Actually Bad	**Actually Good**
Predicted Bad	274	136
Predicted Good	232	675

From Table 1, we can see that the decision tree correctly classified the testing data set's wines 72.06% of the time. Hence our training error is 24.98% and our test error is 27.94%.

To improve our decision tree, we can see if pruning it will lead to an improvement at the rate of accuracy when classifying the test set. We can plot the cross-validation relative error against the size of the tree, as seen in Figure 7. The size of the tree depends upon the complexity parameter (cp) value, the bigger the cp value, the smaller the tree.

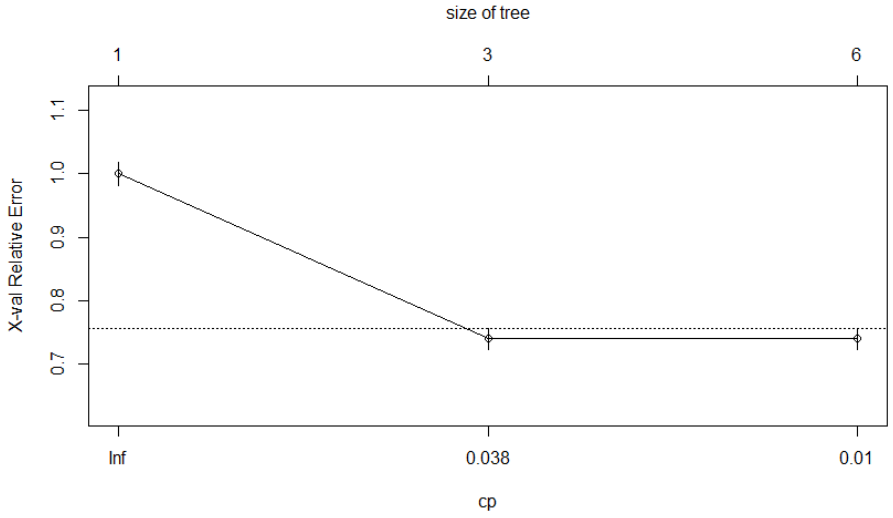

Figure 7: Optimum size of tree

From Figure 7, we can see that the cross-validation relative error is the same (or very similar) for a tree of size 3 and 6. This means we can prune our tree and change it from a tree of size 6 to a tree of size 3 by specifying a cp value of 0.038 instead of default r value of 0.01.

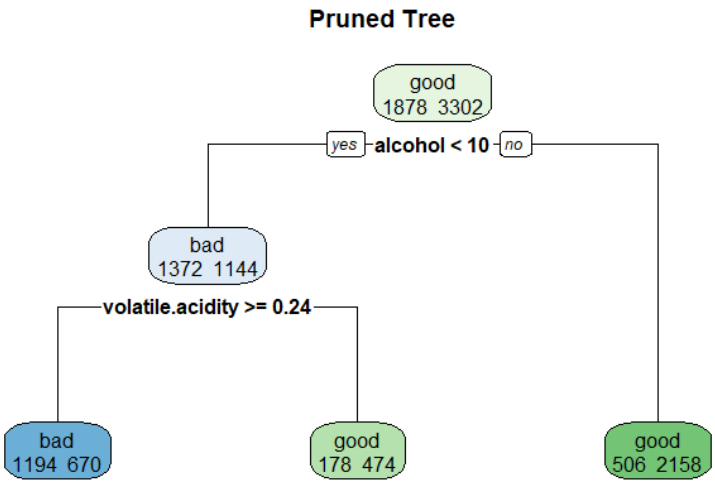

Figure 8: Pruned tree

By adding up the correctly classified wines at each terminal node of our pruned decision tree (Figure 8), we see that the tree classifies the training set's wines correctly 73.86% of the time. Again, we can produce a confusion matrix to see how the pruned decision tree classifies the test set's wine.

Table 2: Confusion matrix for pruned decision tree

	Actually Bad	**Actually Good**
Predicted Bad	318	177
Predicted Good	188	634

From Table 2, we can see that the decision tree correctly classified the test data set's wines 72.29% of the time. Hence our training error is 26.14%, and our test error is 27.71%. The training error has increased by 1.16%, but the test error has decreased by 0.23%. This suggests that the original decision was overfitting the training data leading to a more unsatisfactory test set performance. The pruned decision tree with fewer splits leads to lower variance and better interpretation at the cost of a little bias (which caused the training error to increase).

3.2 Classification - Random Forest

Instead of using decision trees, we will use another classification technique: random forest. The random forest is a classification algorithm consisting of many decisions trees. First, we will use the standard random forest function of r from the package 'randomForest' by not specifying the *ntree* or the *mtry* value. The *ntree* value refers to the number of trees that are grown and is defaulted to 500. Whereas the *mtry* value is the number of variables available for splitting at each tree

node, the default (in classification models) is the square root of the number of predictor variables (rounded down) so in our case it will be three (Liaw, n.d.). After running the random forest with these specified values, we get our final model. The wines in our training set are correctly classified 83.45% meaning the training error is 16.55%. We can test the quality of this model by testing how well it classifies our test set.

Table 3: Confusion matrix for random forest

	Actually Bad	**Actually Good**
Predicted Bad	362	88
Predicted Good	144	723

As seen in Table 3, the random forest correctly classifies the testing set 82.38%, meaning the testing error is 17.62%. To try to improve our model, we can look at how the error rate of the random forest change depending on the number of trees. We can graph our results to show how these error rates change.

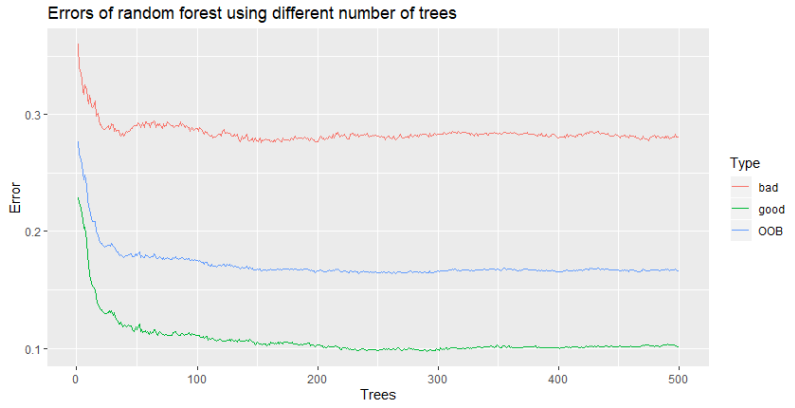

Figure 9: Errors of random forest for different number of trees

As can be seen from Figure 9, the error rates tail off when the tree value is around 250. This means that in our final random forest model, we will use a *ntree* value of 250 (meaning that will be the total number of trees).

Next, we will look to see what the optimum *mtry* value. We will do this by plotting the out-of-bag errors at different *mtry* values. We want the out-of-bag error to be as small as possible.

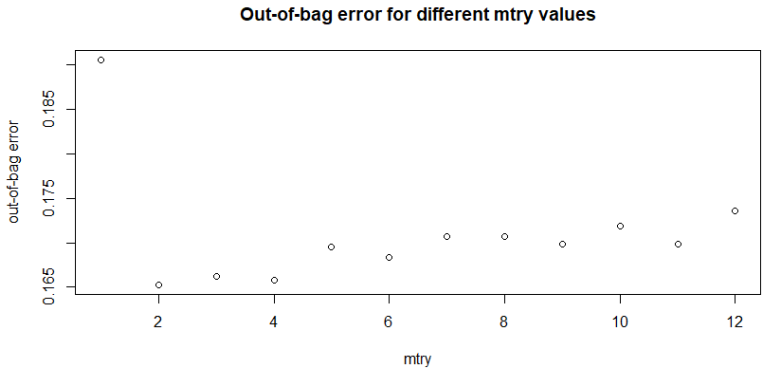

Figure 10: Errors of random forest for different *mtry* values

We can see from Figure 10 that the optimum *mtry* value is 2 as this is where the out-of-bag error is lowest.

Hence from our results from Figure 9 and Figure 10, the error rates are minimised for our random forest when the number of trees is 250 and the *mtry* value is 2. Therefore, we will run another random forest where we specify these two values.

After running the random forest with these specified values, we get our final model. The wines in our training set are correctly classified 83.53% meaning the training error is 16.47%. We can test the quality of this model by testing how well it classifies our test set.

Table 4: Confusion matrix for final random forest

	Actually Bad	**Actually Good**
Predicted Bad	364	83
Predicted Good	142	728

Using this model, the test set is correctly classified 82.92% of the time, this means the testing error is 17.08%. This is an improvement on all the classification models that we previously made and tested. On top of this, this model is more straightforward than our previous random forest model as it only has 250 trees instead of 500 and the *mtry* value is now 2 instead of 3.

4 Conclusion

We have tried classification techniques to produce models to predict the quality of wine based on many predictor values. To see which model is best, we can look at the training and test errors for each model.

Table 5: Comparison of the four classification models

Test	**Training Error (%)**	**Test Error (%)**
Original Decision Tree	24.98	27.94
Pruned Decision Tree	26.14	27.71
Original Random Forest	16.55	17.62
Optimised Random Forest	16.47	17.08

To decide on which model is the best, we look at the test errors for each model. From Table 5, we can clearly see that the optimised random forest is the best as it has the lowest test error out of the four models. On top of this it also has the lowest training error showing that it is the model that fits the training set best too (although this is of less importance than the test error). This model that we designed had 250 trees and had 2 variables available for splitting at each tree node and classified 82.92% of the wines in our testing set correctly.

A further investigation we could do is instead of splitting the wines into good and bad, we could see how well the classification techniques work in predicting each quality rating (0-10). Another thing that could be changed in a future project is to scale the variables or remove anomalies before doing the tests/creating the models.

References

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2017), *An introduction to statistical learning: with applications in R*, Springer.

Liaw, A. (n.d.), 'Classification And Regression With Random Forest', Available at: `https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest`. (Last accessed on 2 April 2020).

P. Cortez, A. Cerdeira, F. A. T. M. and Reis, J. (2009), 'Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553', Available at: `https://archive.ics.uci.edu/ml/datasets/Wine+Quality`. (Last accessed on 2 April 2020).