STAT 515 Final Project: Prediction of Wine Quality

Nebojsa Hrnjez

G01337837

Table of Contents

Project Description

A wine rating is a subjective measure with many difficulties. Often classifications of quality and taste are dependent on the opinions of "wine experts". This process comes with many problems, particularly concerning personal biases based on the reputation of vineyards or the price of the wine. It would be advantageous to the industry as a whole if wine could be classified based solely on its properties. Utilizing data analysis methods is an easy and cost-effective way to classify based on the facts and not on implications.

The data for this analysis is from a study with a similar objective. By performing data analysis methods taught in the STAT 515 class on physiochemical properties of wines, these methods will be trained and be able to predict the quality. This project will use three classifiers to replicate the performance of wine experts: regression, decision tree, and random forest. The main measure of performance is the classification rate, which measures the number of correctly predicted classifications versus the actual classification. In a real-world application when the classifier is trained sufficiently, it will be able to predict wines that have not been previously classified.

There are three research questions that this analysis will explore. The primary question is directly related to the performance of the classifier: Can a model be built that can predict the quality of wine? Additionally, through further analysis of classifier performance, the key input variables will be examined and if there are different key variables for red and white wine.

Through exploring this data set, analyzing it, and then applying the models taught in the STAT 515 class an appreciation and display of the knowledge gained will be presented.

Data Set

Data Source

This data was originally sourced from the UC Irvine Machine Learning Repository (Cortez P. , Cerdeira, Almeida, Matos, & Reis, 2009). Upon further inspection, it is noted that this dataset is from a previous study into modeling wine preferences by data mining from physicochemical properties (Cortez P. , Cerdeira, Almeida, Matos, & Reis, 2009).

Data Description

From this study, there are two distinct datasets one for white wine and one for red. The white wine dataset has 4898 instances and the red wine dataset has 1599 instances.

Each of the datasets has the same 11 input attributes and the same output attribute. Table 1 provides a list of the attributes in each data set, all input attributes are based on physiochemical tests on the wine samples and are numerical, there being a range of values with no clear pattern. The output variable is a median score of at least 3 evaluations made by wine experts. The output variable was initially thought to be numerical due to the 1 to 10 range of values but after initial exploration was deemed to be ordinal.

Data Context

As previously mentioned, this dataset was created for a study (Cortez P. , Cerdeira, Almeida, Matos, & Reis, 2009) in applying data mining methods to predict potentially subjective classifications using physiochemical properties.

The two datasets are related to red and white wine of the Portuguese "Vinho Verde" wine. In each of these data sets, there is no information about grape types, wine brands, and prices due to privacy and logistic issues.

Data Inspection and Preparation

This dataset was originally presented as a ".csv" file. I initially used the read_csv() function in the R package readr (Wickham & Hester, readr: Read Rectangular Text Data. R package Version 2.1.0, 2021) but had difficulty getting the data into a data frame properly due to the delimiter in the .csv file. I proceeded to use the "text to column" function within "Excel" to pre-process the data before importing it into excel.

Once the data was accessible the code seen in Code 1 was used to import the dataset into R and verify if there were any missing values and omit them. Once the dataset was complete I converted it to a data frame using the as.data.frame() function. This was done for both the red and white wine datasets.

There was very little inspection and preparation to be done to this dataset as it was already cleaned for the study it was created for.

Exploratory analysis and research questions

Before any analysis, two initial research questions were created based solely on the attributes and context of the data set:

1. Can a model be built to predict wine ratings?
2. If so, what are the key variables in predicting wine ratings?

Initially, it was believed that this dataset could be predicted using linear regression, based on values within the data set and the nature of the values. To explore the data the L.I.N.E assumptions discussed in class were examined along with other preliminary analysis techniques.

P-value and Correlation Plot

The first step is to create a simple multiple linear regression using all the available data and all input variables through the lm() function (labeled w.mlm and r.mlm) in the stats package (Team, R Core, 2013) as seen in Code 2. A summary was outputted as a preliminary exploratory analysis using the summary() function. This output is seen in Code 3.

Certain variables may not be statistically significant due to p-values greater than 0.05. Additionally, when looking at the white and red datasets there are differences in which variables have high p-values.

White wine variables include: Citric Acid (0.81759), Chlorides (0.65097), and Total Sulfur Dioxide (0.44979)

Red wine variables include: Fixed Acidity (0.3357), Citric Acid (0.2150), Residual Sugar (0.2765) and Density (0.4086)

Next, the correlation between variables was examined using the corrplot() function from the corrplot package (Wei & Simko, 2021) originally shown in class and seen in Code 4. In graphs 1 & 2 we again note differences in which variables have high and low correlation not only with the output variable, Quality but also amongst the input variables themselves.

White wine variables of interest include Citric Acid and Free Sulfur Dioxide due to a low correlation with Quality and Density to Residual Sugar and Alcohol due to the highly negative and positive correlations respectively.

Red wine variables of interest include Residual Sugar due to a low correlation with Quality.

Individually, these techniques do not determine which variables are key but allow the exploration of removing variables to improve model performance. Additionally, the results in this preliminary exploration showed that there are differences in the interactions of variables when looking at white or red wine. This leads to the final research question:

3. Are there different key variables in red and white wine?

In a subsequent section of this report, full datasets will be used in addition to datasets with removed variables when examining model performance.

L.I.N.E Assumptions

The next exploratory technique is the examine the L.I.N.E assumptions of linear regression models. To start the plot() function from base R (Team, R Core, 2013) was used on the previously created w.mlm and r.mlm regressions as seen in Code 5. In graphs 3 & 4, the output for Code 5, the points in the Residual vs Fitted and Scale-Location have a distinct pattern not normally seen in these plots. These patterns are due to the fact that while the response variable is a rating from 1 to 10, it is not a numerical variable but an ordinal one. Each of the rankings has meaning relative to each other and the values in the output variable are integers and not decimal.

The L.I.N.E assumptions will be tested using a variety of R packages as well as several functions created by Professor Richard Sigman within the R script “rss_regress_funcs_v2.R”. (Sigman, 2021). All of these exploratory measures will be applied exclusively to w.mlm and r.mlm throughout this section.

Linearity

Linearity is the assumption that there exists a linear relationship between the outcome variable and the independent variables. This is most easily checked using a residuals vs fitted plot. Furthermore, non-linearity impacts are less severe and the resulting biases may be small if the multiple R2 values are below 0.40.

To implement these tests the residFit() function (Sigman, 2021) was called on both w.mlm and r.mlm to generate the residuals vs fitted plot as seen in Code 6 with the output shown in Graphs 5 & 6 respectively. The summary for each regression was already generated in Code 2 with output seen in Code 3.

In the white wine dataset, the residuals vs fitted plot indicate linearity. The downward slope between fitted values 3 and 4 may be due to the lack of instances at that extreme and also the outlier labeled 2782. As previously mentioned, the points ended up being plotted in a distinct pattern this is due to the ordinal nature of the response variable. Additionally, we see in Code 3 that the multiple R2 value is 0.2819 which is well below the 0.40 threshold.

The red wine dataset shows a different residuals vs fitted plot. While the pattern in plotted points remains the same, we see higher fluctuations in the regression line but not extreme enough to indicate non-linearity. The R2 value for the red wine regression is higher at 0.3606 but still below the 0.40 threshold.

Independent-errors

The independent errors assumption stipulates that the residuals should be uncorrelated. While difficult to detect using data analysis alone, there are numerous tools to help indicate the presence of correlated errors, it also is important to note that this assumption only particularly applies when the data is a repeated measure taken at numerous time points.

Since our data is not ordered a response vs order plot would not be informative. Two additional measures of error correlation, the autocorrelation function or acf() function from the stats package (Team, R Core, 2013) and the Durbin-Watson test or dwt() function from the cars package (Fox, Weisberg, & Price, 2021) can be seen in Code 7. To show the independent-errors assumption holds, the values for lag 1 and beyond must be within the 95% confidence interval lines for the autocorrelation function and the D-W statistic must be around 2.

For both the white and red wine datasets, the ACF plot, shown in Graphs 7 & 8 respectively, we see that a number of the lags are very close to the 95% interval line but at numerous points exceed it.

Again, both the white and red wine datasets have similar results, seen in Code 8. Both the autocorrelation value and D-W statistic are outside the values of 0 and 2 respectively to assume independent errors, they are very close and likely the assumption can be held without much concern. When considering that this assumption is also more effectively used on time-series datasets, which this is not, there are no major concerns in regards to this assumption.

Normality

Normality assumes the residuals are normally distributed. The main avenues for verifying this assumption are a histogram plot and a Q-Q plot, additionally, the skewness() and kurtosis() functions from the moments package (Komsta & Novomestky, 2015) can give numerical values of the patterns seen in the histogram plot. Looking at Code 9, the histogram is produced using the geom.histogram() function from the ggplot2 package (Wickham, ggplot2: Elegant Graphics for Data Analysis, version 3.3.5, 2021), and the Q-Q plot was created using the normalQQ() function (Sigman, 2021).

Looking at Graphs 9 & 10 for the white wine dataset, it would appear that the normality assumption is satisfied but with some interesting features. The histogram appears centered around 0 and the low skewness value, 0.073, would agree that it appears normally distributed. It should be noted that both the histogram and the kurtosis value, 4.1, would indicate that there may be outliers or more "extreme" values than normal. The Q-Q plot appears to also agree with the previous statements along with indicating potential outliers at the extremes.

Graphs 11 & 12 for the red wine dataset show almost identical results to the white wine dataset. Again the residuals appear normally distributed, but with a higher than normal amount of "extreme" values as indicated by a kurtosis value of 3.71. The Q-Q plot agrees with the previous statement for this dataset as well.

From the initial normality tests, there was an indication of the possibility of outliers that would have a large influence on any potential model. Two methods for identifying these observations are a residuals vs leverage plot and the Cook's Distance plot, both of which are available through the "rss_regress_funcs_v2.R" script (Sigman, 2021).

The residuals vs leverage and Cook's Distance plots seen in Graph 13 & 14 for the white wine dataset both indicate the observation at "2782" as an outlier. For the red wine dataset, neither of the plots seen in Graph 15 & 16 indicate any outliers have a large influence.

Equal-variances (Homoscedasticity)

The equal-variances ensure that residuals do not vary systematically and are tested by examining a residuals vs fitted plot and a scale-location plot. The residuals vs fitted plot was already created as seen in Graphs 5 & 6. To create the scale-location plot we used the scaleLocation() function (Sigman, 2021) as seen in Code 11.

Both the residuals vs fitted plots and the scale-location plots seen in Graph 17 & 18 for the white and red wine datasets respectively show a very distinct pattern. This can be explained by the ordinal nature of the response variable. This however isn't the type of pattern that usually appears in these plots. The points are normally scattered and evenly distributed for both axes, when patterns do form it is usually in the points forming a general shape like a cone or pipe shape. When the linear regression model predicts output values, it does so with the possibility of any value between two integers (i.e. 1.1, 1.2, 1.3, or possibly to more decimal places) where the actual output is only integer values between 1 to 10. This makes the equal-variances assumption difficult to determine, however, this is not a major concern as even if this assumption is not satisfied the coefficients, conditional means, and predicted means are still considered unbiased just not minimum variance.

Multicollinearity

The final assumption is not a L.I.N.E assumption but one that applies only to multiple linear regressions. The multicollinearity assumption is when there exists a high correlation between two predictors generally this threshold is about 0.80 on a correlation plot. The two main functions used to determine multicollinearity are the correlation plot already produced in Graphs 1 & 2, and the Variance inflation factor (VIF) produced using the vif() function from the car package (Fox, Weisberg, & Price, 2021) and seen in Code 12. VIF values over 10 indicate multicollinearity and those predictors are candidates for removal.

The correlation plot for the white wine dataset seen in Graph 1 indicates that there is a high correlation between Density and Residual Sugar. There is also a high correlation between Density and Alcohol but it does not cross the 0.80 threshold. Examining the vif() function output at Code 13, it agrees with our previous analysis as both Density (28.23) and Residual Sugar (12.64) have values above 10, Alcohol is close (7.71) but does not cross the threshold.

The red wine dataset contained no indications of multicollinearity or excessively high VIF values. One would expect that since the predictors are the same for both datasets that the correlations would be the same but this is not the case. The differences could be explained by the number of observations between the white and red wine dataset, 4898 vs 1599 respectively, fewer observations preventing certain patterns from forming. Another possibility is that the interactions between chemical properties are distinctly different between white and red wine causing different levels of correlation.

Results Interpretation

Examining all exploratory analysis results allows for the removal of some attributes and observations. This will potentially lead to better data analysis results and ultimately stronger prediction. The exploratory analysis will be rerun and only results discussed with the pruned datasets and will appear in the compile report for said R script and will be used in the following sections of this report.

The results of the white wine dataset exploratory analysis indicated two courses of action be taken.

1. Remove the observation "2782", as it was highlighted in multiple plots. This was deemed a low-risk decision as removing a single observation from a population of almost 5000 likely would not have many consequences. When the exploratory analysis was rerun with a dataset excluding "2782" no significant changes in observations were noted. The VIF values for Density and Residual Sugar increased to 36.46 and 16.099 respectively.

2. Removed the predictor "Density" from the dataset. This decision is based on this predictor's high correlation with other predictors as well as its very high VIF value. When the exploratory analysis was rerun excluding "Density" there were fewer predictors with high p-values but most of the plots looked very similar. The only predictor whose p-value remained high was "Citric Acid" at 0.776 but this was the only thing that stood out. "Citric Acid" is a candidate

The results of the red wine dataset exploratory analysis did not yield any compelling information for observation or predictor removal. A few predictors had high p-values but this was the only factor that indicated these predictors should be removed. The exploratory analysis did yield that the third research question is important as the white and red wine data sets showed vastly different observations for each predictor.

In concluding the exploratory analysis, this dataset and the white and red wine subsets were revealed to be more complicated than initially anticipated. The ordinality of the response variable caused some of the L.I.N.E assumption tests to be difficult to read or otherwise not useful. The data was pruned based on the evidence available and the fact that the response variable was realized to be ordinal is valuable information in itself.

Data Analysis

Upon realizing that the response variable was ordinal a moment of panic set in. The STAT 515 class did not distinctly cover how to analyze ordinal variables, and they are unique because they can be considered either classification or prediction problems. Additionally, ordinal regression and ordinal tree models do exist but would be an exercise in self-learning and may not yield sufficient results for this final project.

For ease of comparison, it was decided to treat this data as a classification problem, meaning any predictive outputs are simply rounded to the nearest whole integer and then compared to the actual output and classified as correct or incorrect. This classification rate is considered the measure of success with a higher correct classification indicating a better model.

Methods and Software Used

For this project three main data analysis techniques were used: Multiple Linear Regression, Classification Trees, and Random Forests all using R studio. For each model a "standard" version taught in class was run and an ordinal version was attempted. Each model was initially run with a simple randomly generated train/test split at an 80/20 ratio using the sample() function from base R (Team, R Core, 2013) as seen in Code 14. Each model was then run again using 10-fold cross-validation to ensure accuracy and consistency in the results. This was achieved using the cvTools() function from the cvTools package (Alfons, 2012) and a for loop using base R (Team, R Core, 2013) as seen in Code 15.

The process is almost the same for both the white and red wine datasets, therefore only white dataset code will be shown. Additionally, within the white dataset, we will be examining two subsets, one with all the predictors and one with all the predictors except Density, therefore only the first subset will be shown. Any unique situations that present themselves in either scenario will also be shown.

Multiple Linear Regression

This method was the easiest to implement but also did not produce particularly successful results. Two main functions were used in this section: lm() from the stats package (Team, R Core, 2013) for multiple linear regression and polr() from the MASS package (Ripley, 2021) to perform ordinal logistic regression.

As previously stated both lm() and polr() were called to train initial models using training and test subsets seen in Code 14 for both "dfw" (white wine all predictors), "dfw2" (white wine all predictors less Density), and "dfr" (red wine all predictors). This process was then repeated using 10-fold cross-validation as seen in Code 15, with the mean of the 10 classification rates calculated taken to get an accurate result for that method. This results in 8 different correct classification rates calculated.

Multiple linear regression using lm() was used exclusively and no other functions were used like regsubsets() from the leaps package to determine any best combination. This decision was made after completing the initial data analysis for the oral presentation of this project seen in the "White_MLR_Intial.pdf" compile report. That initial analysis showed that the selection of predictors was relatively random, with the "best" subset changing each time the function was run. As discussed at the start of this section, the predicted outputs for this model needed to be rounded using round() found in base R (Team, R Core, 2013) to be able to compare to the test set outputs. This can be seen in the last line of Code 15 and the process is the same for both iterations.

The polr() function was chosen instead of the clm() function from the ordinal package due to a larger amount of information available about that function. The basic explanation of this function is that it fits outputs based on proportional

odds. For this project, the classification of each output was used (highest probability) and not the probabilities for each possible classification. This allowed easy calculation of the correct classification rate and no rounding was required.

The entirety of this code can be seen in the "White_MLR_Final.pdf" and "Red_AllModels_Final.pdf" compile reports. Code 16 illustrates an example of the process using the lm() function through both the training/test set and the cross-validation. This process is repeated for both "dfw" and "dfw2" as well as "dfr". The polr() function follows the same process and simply replaces the lm() function and rounding is removed.

Classification Trees

The classification tree method followed almost the same process as the multiple linear regression method. There were two distinct differences: no rounding was needed as "class" was specified in the type argument of the predict function and some pruning was done to select the minimum error tree using the "cp" value.

The functions used were the rpart() function from the package of the same name (Therneau & Atkinson, 2019) for the classification tree. rpartScore() function also from the package of the same name (Galimberti, Soffritti, & Di Maso, 2012) was used for the classification tree for ordinal responses.

The standard classification tree was pruned in a process similar to the one shown in lecture. An initial tree was created with a very small "cp" or complexity parameter, which is the minimum improvement in the relative error required to add an additional node. The "cp" value that produced the minimum relative error was then selected and a new tree was trained and new outputs predicted using this "cp" value. This method was used for both the training/test set and the cross-validation set and can be seen in Code 17 and 18 respectively. Results were recorded for both the pre and post-pruned trees.

The ordinal classification tree had two measures of pruning: a specific argument, prune, that was based on either total misclassification rate or total misclassification cost and the previously used "cp" value. Total misclassification rate was used as this is our measure of success for this project and the "cp" was selected in the same way as the standard classification tree. The same process is seen in Code 17 and 18 was applied to the ordinal classification tree, with rpartScore() simply replacing rpart().

Additionally, plots for certain trees were created using the rpart.plot() from the rpart.plot package (Milborrow, 2021) to help visualize solutions and also results of pruning, and will be discussed in further detail in the results section. The entirety of this code can be seen in "White_CART_Final.pdf" and "Red_AllModels_Final.pdf".

Random Forests

This method again followed the process explained in the previous two sections. Again no rounding was needed but the response variable Quality did need to be transformed into a factor using the function as.factor() from base R (Team, R Core, 2013) to keep the predicted values as integers to allow for comparison to actual output.

The functions used were the randomForest() from the package randomForest (Breiman & Cutler, 2018) for the standard random forest and ordfor() from the ordinalForest package (Hornung, 2021) for the ordinal forest. An example of this model can be seen in Code 19, the process is the same for both randomForest() and ordfor().

For the standard random forest, the number of predictors (the mtry argument) was maintained at the maximum amount of predictors. Initial analysis performed for the oral report for this project indicated no significant change in classification accuracy when changing the number of variables considered for defining a node's splitting logic, this can be seen in the "White_classificationtree_ordinal.pdf" compile report. The importance() and varImpPlot() functions from the randomForest package (Breiman & Cutler, 2018) were used to determine the important variables based on MDA and MDI.

The ordfor() function required some tuning of iteration and quantity of tree arguments. The arguments seen in Code 20: nset, ntreeperdiv, ntreefinal, nbest, npermtrial all pertain to either score sets, number of trees, or number of permutations. Initially, the default values for all these arguments were used but the run time to solve these forests was taking over 20 min for the training/test set and even longer for cross-validation sets therefore all arguments were reduced by a factor of 10. Doing this did not reduce classification accuracy at all and therefore was deemed an appropriate course of action. The ordinal forest models output important variables as part of the model and are accessed using ordfor$varimp and bases importance on misclassification rate. The important variables will be discussed in the results section.

The entirety of this code can be seen in "White_RF_Final.pdf" and "Red_AllModels_Final.pdf".

Results

The results of the data analysis can be seen in Table 2 and are visualized in Graphs 19 and 20. For all subsets, there is a clear indication that "Forest" based methods performed the best, on average classifying 13.73% and 11.96% better than "Regression" and "Tree" based methods across all methods and subsets respectively. While these results are a significant improvement, they are likely the result of overfitting. Note that the "Tree" based methods, in general, performed slightly better than regression methods. Looking at Graphs 21 and 20 we see the plotted unpruned and pruned classification trees for the "dfw" data set using a training/test set. Both trees had a correct classification rate of ~55% and both trees have a high number of leaf nodes. Now apply this to a random forest that generates hundreds of trees, eventually, every observation would be covered. And finally, when you consider that the majority of outputs exist in the ratings 5 to 7, overfitting is not unlikely in this scenario.

Comparing "Standard" and "Ordinal" types of the same method shows no significant advantage in classification rate amongst all methods and subsets. This is not entirely surprising due to the nature of the ordinal response. The dataset contains an ordinal response variable that was created by taking the median score of at least three wine expert scores, which I believe forces the response variable into that ordinal classification. Without seeing the original scores for each observation, it is impossible to know for certain but certain scenarios come into question. If a bottle of wine was rated by three experts with scores of 2, 5, and 5 respectively the median would be 5 and the mean would be 4, this is an extreme example but illustrates that the potential for significant error exists. Additionally, how the experts gave their scores is also unknown, scores may have been given in decimals as well as integers.

Amongst the data subsets, the red wine dataset was on average classified at a higher rate than either of the white wine datasets. When looking at the two white wine datasets, dfw (all predictors) and dfw2 (all predictors less Density), there is a slight increase in classification rate for the dfw2 data set, which indicates that removing Density as discussed in the exploratory analysis was a valid decision. Comparing the red and white wine datasets, it is probable that the white wine dataset set suffers from noise polluting the output, having more than 3 times the observations.

Important Variables

This initial analysis done for the oral presentation of this project revealed a lot of information about variable selection as it pertains to this dataset. Amongst all models, there was not a large change in classification accuracy when sub-setting by various predictors, this can be seen specifically in the initial compile reports.

For this project, the MDI and MDA values were taken from the random forest models and the important variables from the ordinal forest as seen in the "White_RF_Final.pdf" and "Red_AllModels_Final.pdf" compile reports. Looking at Table 3 some clear patterns are present. The most important variable across all subsets was Alcohol and this is expected as alcohol is one of the vital components to wine, this is also verified as Alcohol is the first predictor used to split the classification trees. Consistently, Volatile Acidity, often associated with oxidation problems (Gardner, 2015) in wine was found in the top four for all models. For white wine Table 3 shows that Alcohol, Volatile Acidity, and Free Sulfur Dioxide were the top three respectively for all models. For red wine Alcohol, Sulphates, and Total Sulfur Dioxide were considered key variables.

Conclusions

The goal for this project was to create and hopefully answer hypothesis questions based on a data set using techniques taught in the STAT515 class. The three questions related to this project were:

1. Can a model be built to predict wine ratings?
2. If so, what are the key variables in predicting wine ratings?
3. Are there different key variables in red and white wine?

Multiple models were built to predict wine ratings, some with more success than others. At a classification rate of close to 70% for all datasets and training methods, Random Forests are the clear choice for predicting this dataset. There are however some concerns with the classification. As seen in Graphs 5 and 6 but also in examining the initial date sets for white and red wine the majority of the response values are between 5 and 7, there a very few values at the extremes. This means that values at these extremes would likely be misclassified. There is also a question of whether the response variable is actually ordinal or if it was simply coerced into that state when the data set was created.

Key variables were identified for the dataset as a whole and the specific white and red subsets but the significance of these variables outside of Alcohol and Volatile Acidity is questionable. The initial data analysis showed that changing predictors used in any model set did little to improve classification accuracy.

Overall, this project raises the question of if physiochemical properties of wines are effective predictors in wine quality as rated by "experts". Wine experts are often easily fooled about the quality and even which type of wine they are drinking (Pomeroy, 2014). The physiochemical inputs could very well paint a clear picture but because the output is based on opinion and those opinions come with many unpredictable. Even the study this data was created for using the Support Vector Machines classifier achieved similar accuracies in the 57-68% range for a tolerance level of 0.5 and only found higher accuracies when increasing the tolerance level (Cortez P. , Cerdeira, Almeida, Matos, & Reis, 2009). That study also found that the majority of classification occurred in the 5 to 7 range.

Challenges and Further Analysis

The two biggest challenges were the ordinal nature of the response variable and the computation time required for cross-validation of "Tree" and "Forest" models. Since ordinal response variables were not covered in STAT 515, using a combination of evaluating as a classification problem and attempting to learn ordinal classification methods allowed for analysis of this dataset. Additionally, the iterations, number of trees, and conditions for the "Tree" and "Forest" models had to be substantially reduced from the initial setting due to the extended run time. Finally, this project may have suffered from over-ambitiousness, completing three datasets worth of modeling and over 10 different models for each dataset the analysis of each model may have been negatively impacted.

Further analysis would include considering this as a predictive problem, and creating a tolerance range around the predicted value as seen in Code 21 and Graph 23. This would allow for a better comparison amongst models as a majority of models had very similar classification rates and adding a tolerance rate would give an additional measure of performance. Models that have high classification rates and require low tolerance rates would be considered the best. This is also similar to the method used by the study this data set was created for.

Bibliography

Alfons, A. (2012, May 14). *cvTools: Cross-validation tools for regression models, version 0.3.2*. Retrieved from r-project: https://cran.r-project.org/web/packages/cvTools/index.html

Breiman, L., & Cutler, A. (2018, March 22). *randomForest: Random Forests for Classification and Regression, version 4.6-14*. Retrieved from r-project: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 547-553.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009, October 07). *Wine Quality.* Retrieved from UC Irvine Machine Learning Repository: https://archive-beta.ics.uci.edu/ml/datasets/wine+quality

Fox, J., Weisberg, S., & Price, B. (2021, November 06). *car: Companion to Applied Regression, version 3.0-12*. Retrieved from r-project: https://cran.r-project.org/web/packages/car/index.html

Galimberti, G., Soffritti, G., & Di Maso, M. (2012, Decemeber 08). *rpartScore: Classification trees for ordinal responses, Version 1.0-1*. Retrieved from r-project: https://cran.r-project.org/web/packages/rpartScore/index.html

Gardner, D. M. (2015, September 30). *Volatile Acidity in Wine*. Retrieved from PennState Extension: https://extension.psu.edu/volatile-acidity-in-wine

Hornung, R. (2021, June 25). *ordinalForest: Ordinal Forests: Predication and Variable Ranking with Ordinal Target Variables, version 2.4-2*. Retrieved from r-project: https://cran.r-project.org/web/packages/ordinalForest/index.html

Komsta, L., & Novomestky, F. (2015, January 05). *moments: Moments, cumulants, skewness, kurtosis and related tests, version 0.14*. Retrieved from r-project: https://cran.r-project.org/web/packages/moments/index.html

Kuhn, M. (2021, October 09). *caret: Classification and Regression Training, version 6.0-90*. Retrieved from r-project: https://cran.r-project.org/web/packages/caret/caret.pdf

Milborrow, S. (2021, July 24). *Plot 'rpart' Models: An Enhanced Version of 'plot.rpart', version 3.1.0*. Retrieved from r-project: https://cran.r-project.org/web/packages/rpart.plot/rpart.plot.pdf

Pomeroy, R. (2014, August 18). *The Legendary Study That Embarrassed Wine Experts Across the Globe*. Retrieved from RealClear Science: https://www.realclearscience.com/blog/2014/08/the_most_infamous_study_on_wine_tasting.html

Ripley, B. (2021, May 03). *MASS: Support Functions and Datasets for Venables and Ripley's MASS, version 7.3-54*. Retrieved from r-project: https://cran.r-project.org/web/packages/MASS/index.html

Sigman, R. (2021). rss_regress_funcs_v2.R. *STAT515: Applied Statistics and Visualizations for Analytics*.

Team, R Core. (2013). *R: A language and environment for Statistical Computing*. Retrieved from R: Foundation for Statstical Computing: http://www.R-project.org/

Therneau, T., & Atkinson, B. (2019, April 10). *rpart: Recursive partitioning for classification, regression, and survival trees. Version 4.1-15*. Retrieved from r-project: https://cran.r-project.org/web/packages/rpart/rpart.pdf

Wei, T., & Simko, V. (2021, November 18). *corrplot: Visualization of a Correlation Matrix*. Retrieved from r-project: https://cran.r-project.org/web/packages/corrplot/index.html

Wickham, H. (2021, June 25). *ggplot2: Elegant Graphics for Data Analysis, version 3.3.5*. Retrieved from r-project: https://cran.r-project.org/web/packages/ggplot2/index.html

Wickham, H., & Hester, J. (2021, November 11). *readr: Read Rectangular Text Data. R package Version 2.1.0*. Retrieved from r-project: https://cran.r-project.org/web/packages/readr/index.html

Tables

Attribute Name	Units	Input or Output Variable
Fixed Acidity	g(tartaric acid)/dm^3	Input
Volatile Acidity	g(acetic acid)/dm^3	Input
Citric Acid	g/dm^3	Input
Residual Sugar	g/dm^3	Input
Chlorides	g(sodium chloride)/dm^3	Input
Free sulfur dioxide	mg/dm^3	Input

Attribute Name	Units	Input or Output Variable
Total sulfur dioxide	mg/dm^3	Input
Density	g/cm^3	Input
pH		Input
Sulphates	$g(potassium\ sulphate)/dm^3$	Input
Alcohol	vol%	Input
Quality	Score from 0 to 10	Output

Table 1: Attribute description

Data Analysis Results

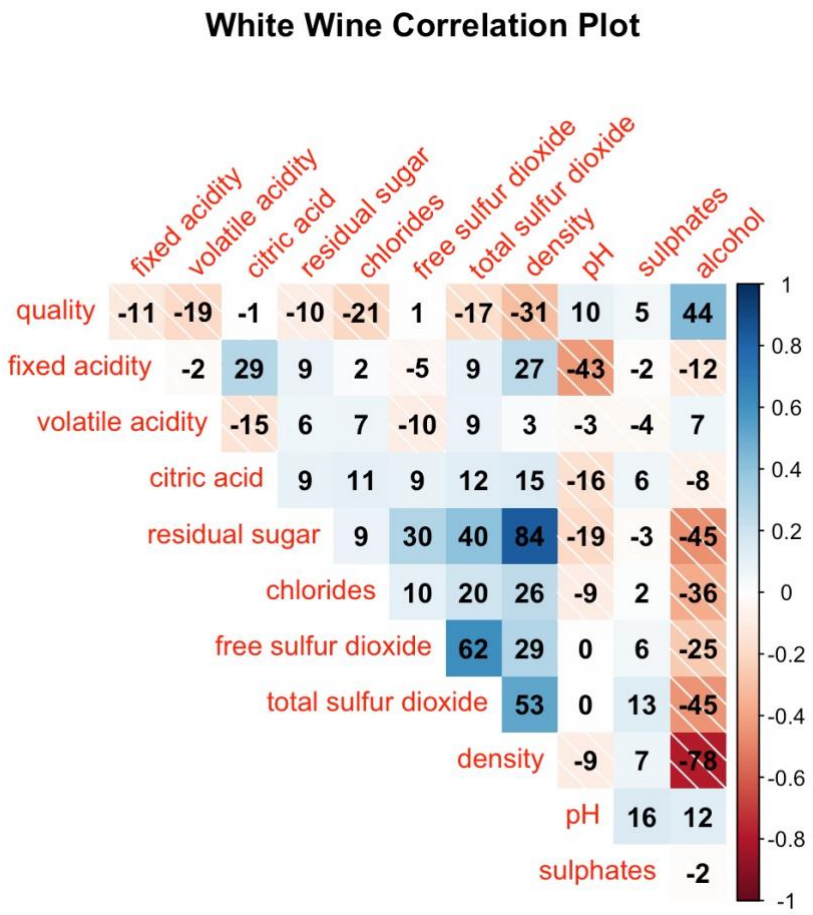
		Model / Sub-Model							
		Regression		Tree				Forest	
Data Set		MLR	OLR	CART	Ordinal Tree	Pruned CART	Pruned Ordinal ..	Ordinal Forest	Random Forest
White Wine (all predictors)	Training/Test Set	53.63%	51.48%	55.77%	55.77%	55.46%	55.36%	67.93%	68.23%
	Cross-Validation Set	51.79%	52.58%			54.69%	55.3%	68.18%	69.08%
White Wine (all predictors less Density)	Training/Test Set	53.32%	54.34%	54.44%	56.89%	56.49%	58.63%	68.23%	68.13%
	Cross-Validation Set	51.54%	52.73%			55.65%	55.48%	69.19%	69.92%
Red Wine	Training/Test Set	60.19%	60.82%	56.11%	57.99%	56.74%	60.82%	67.4%	68.97%
	Cross-Validation Set	59.41%	58.85%			60.78%	59.54%	69.67%	70.48%

Table 2: Data Analysis Results Table

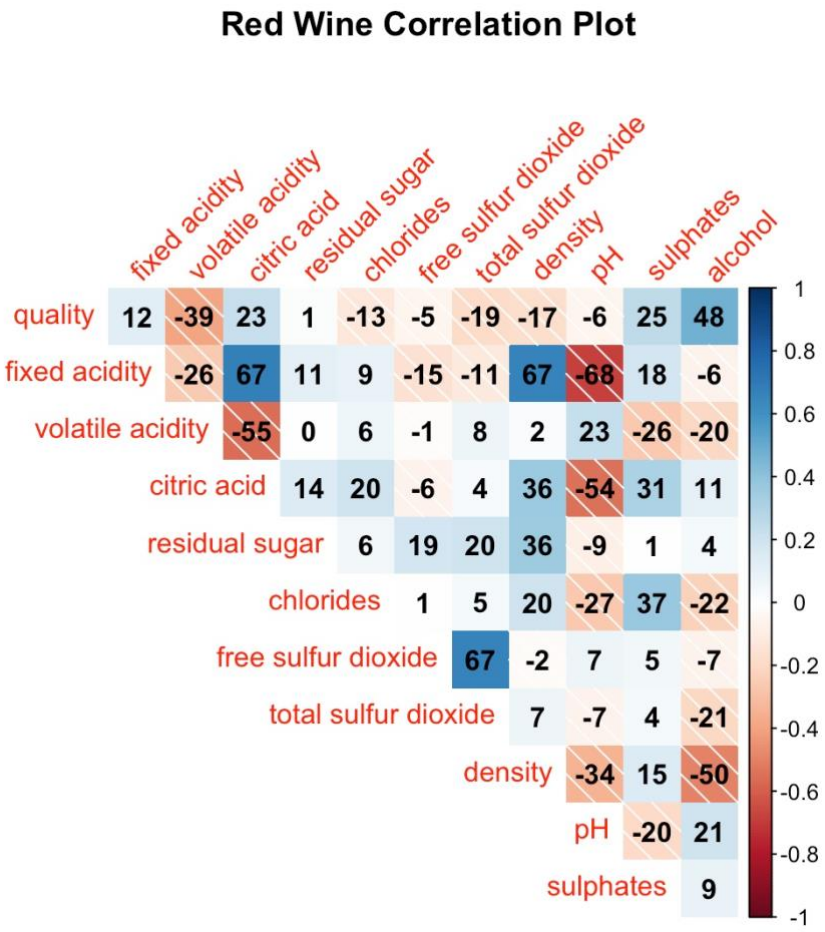
	RANK 1	RANK 2	RANK 3	RANK 4
w.rf MDA	alcohol	volatile.acidity	free.sulfur.dioxide	chlorides
w.rf MDI	alcohol	volatile.acidity	free.sulfur.dioxide	total.sulfur.dioxide
w.rf2 MDA	alcohol	volatile.acidity	free.sulfur.dioxide	residual.sugar
w.rf MDI	alcohol	volatile.acidity	free.sulfur.dioxide	residual.sugar
w.ordfor	alcohol	volatile.acidity	free.sulfur.dioxide	residual.sugar
w.ordfor2	alcohol	volatile.acidity	free.sulfur.dioxide	chlorides
r.rf MDA	alcohol	sulphates	total.sulfur.dioxide	volatile.acidity
r.rf MDI	alcohol	volatile.acidity	sulphates	total.sulfur.dioxide
r.ordfor	alcohol	sulphates	volatile.acidity	total.sulfur.dioxide

Table 3: Variable Importance ranking

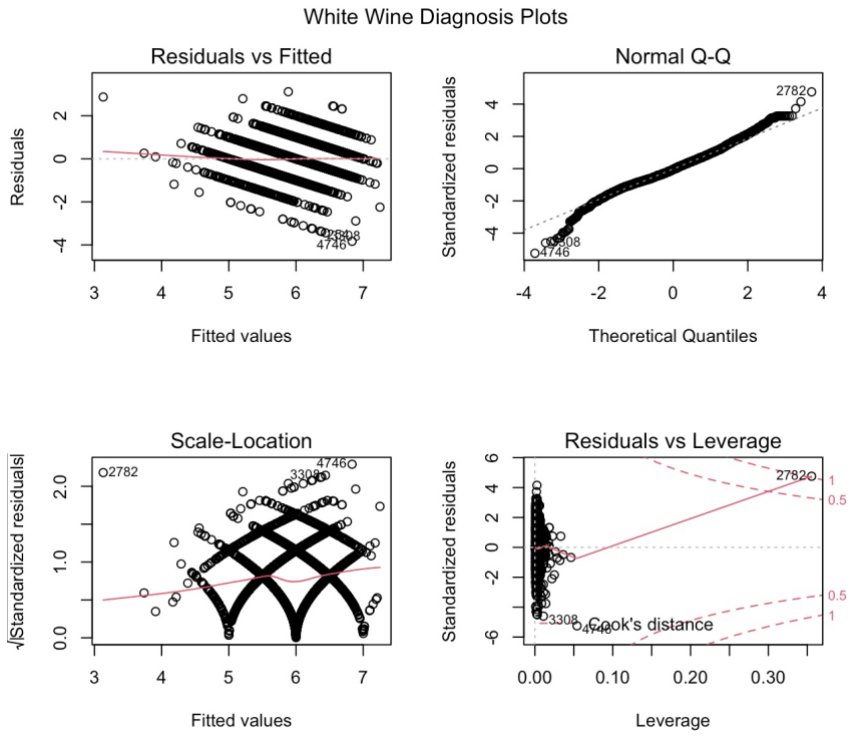
Graphs



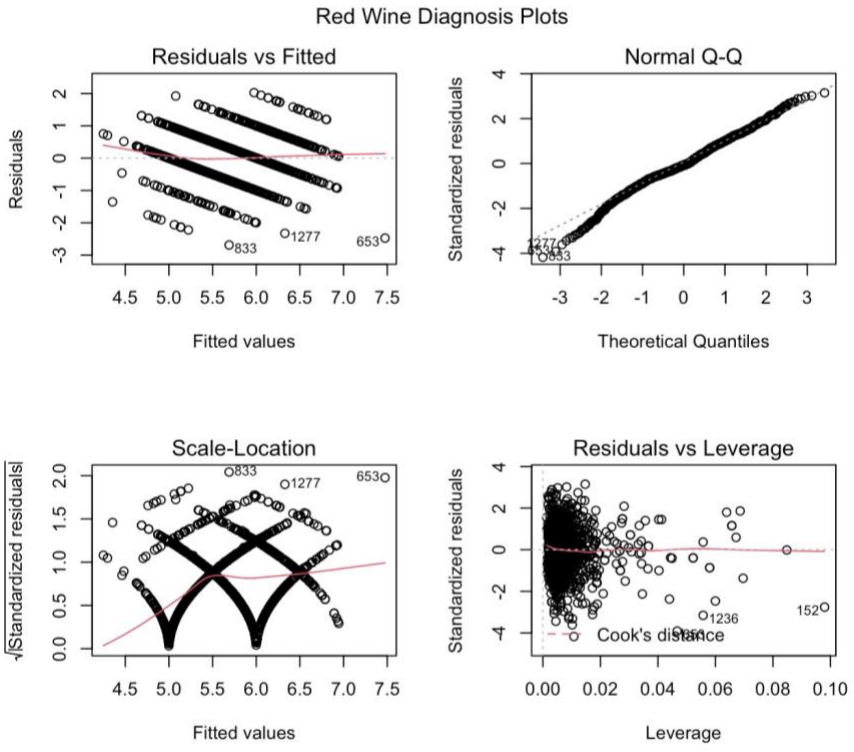
Graph 1: White Wine Correlation Plot



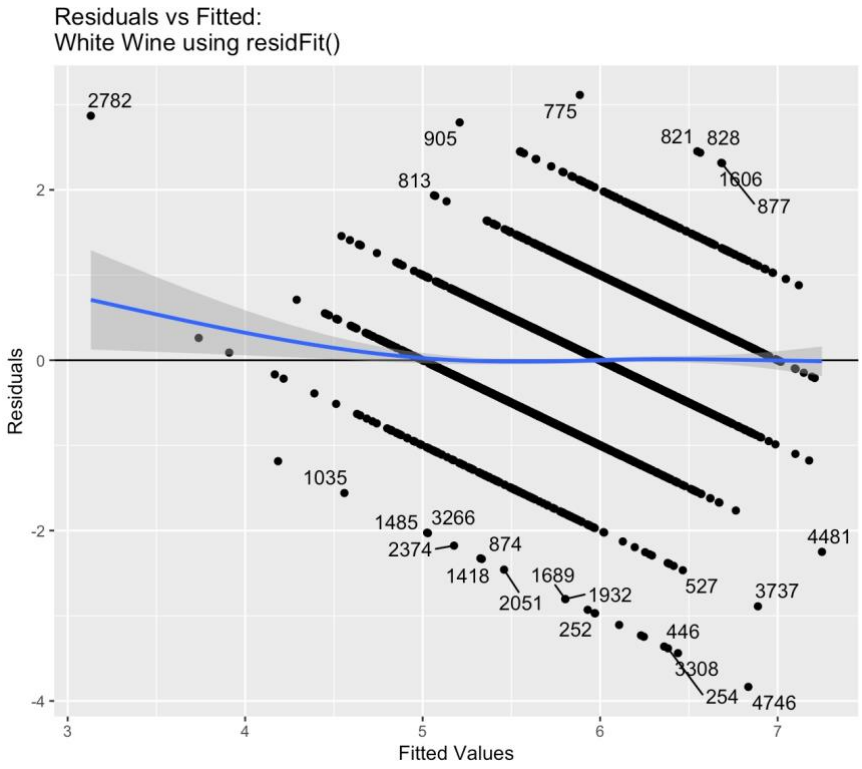
Graph 2: Red Wine Correlation Plot



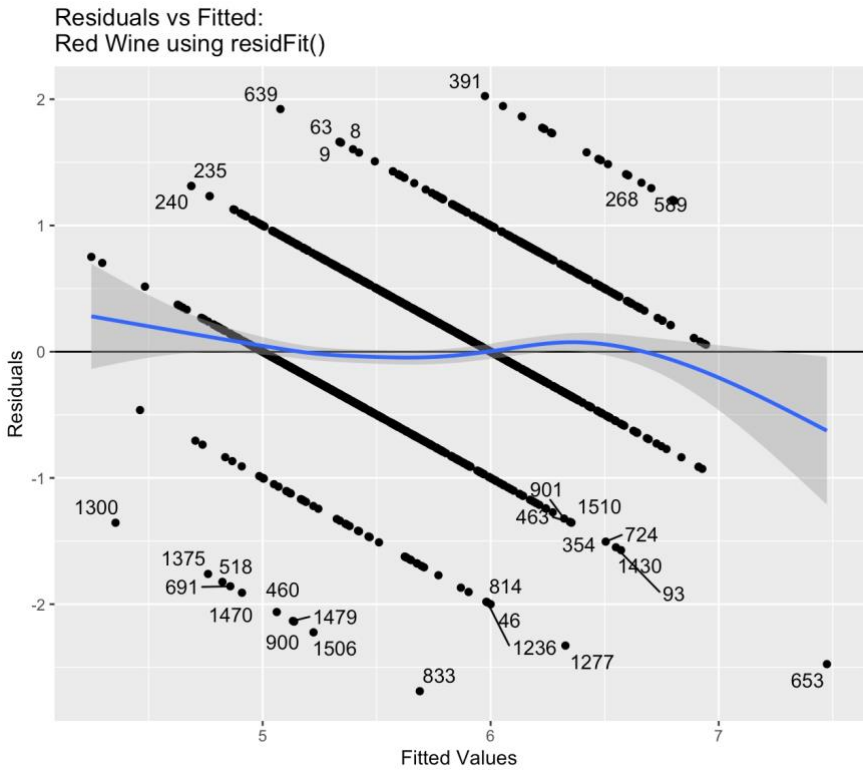
Graph 3: White Wine Diagnosis Plots



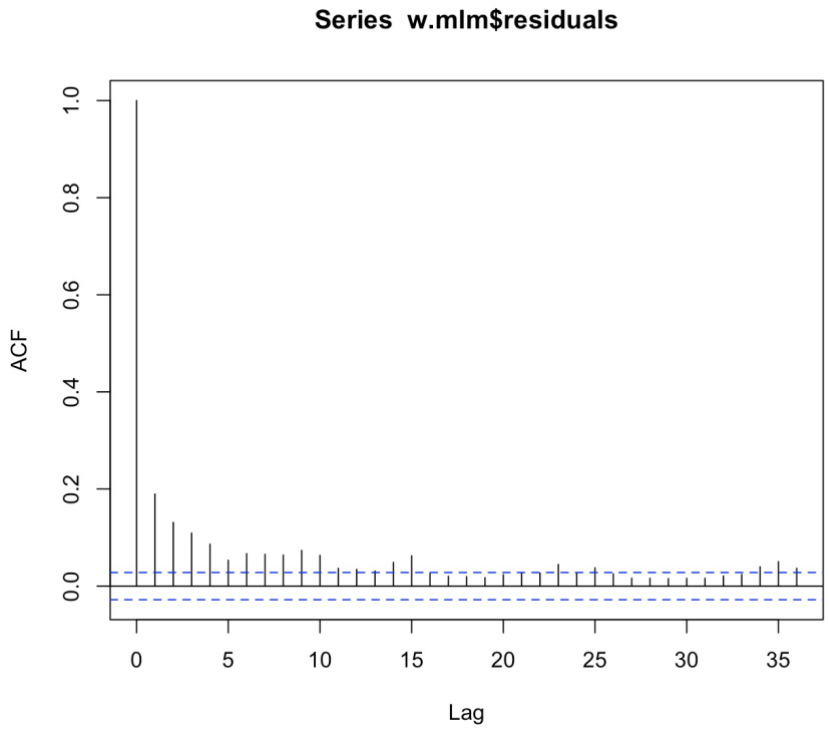
Graph 4: Red Wine Diagnosis Plots



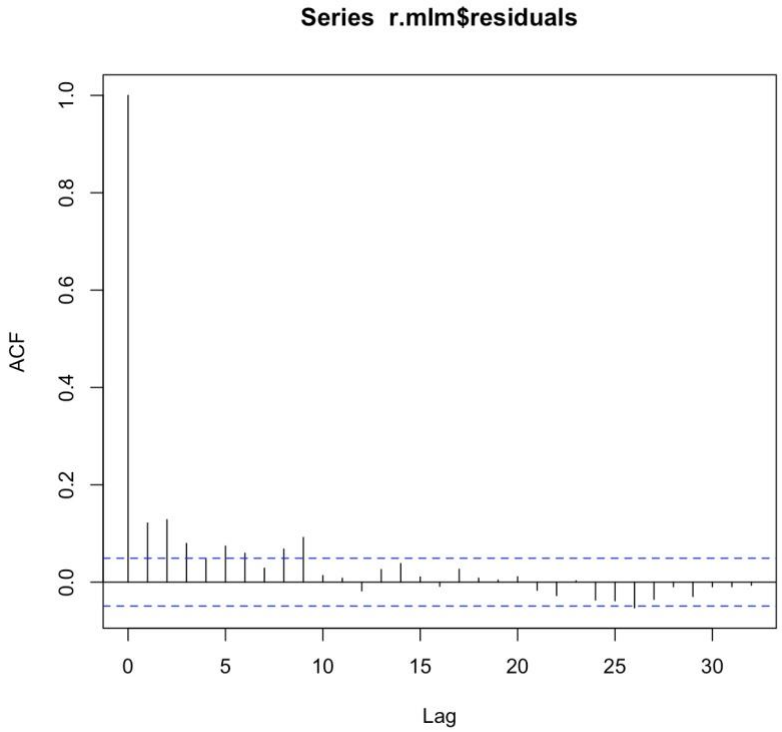
Graph 5: Residuals vs Fitted plot for White wine



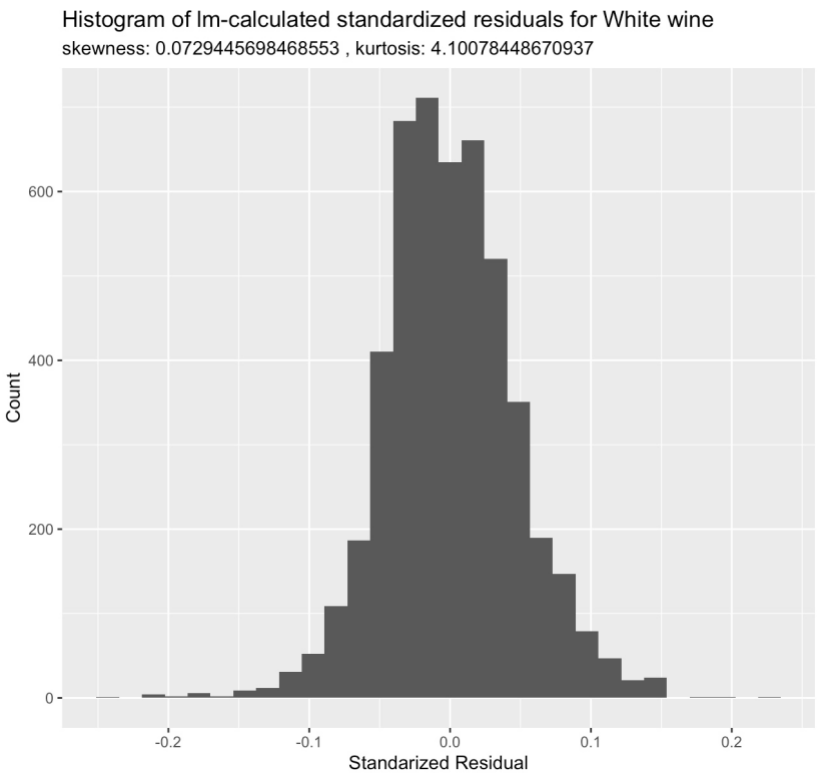
Graph 6: Residuals vs Fitted plot for Red wine



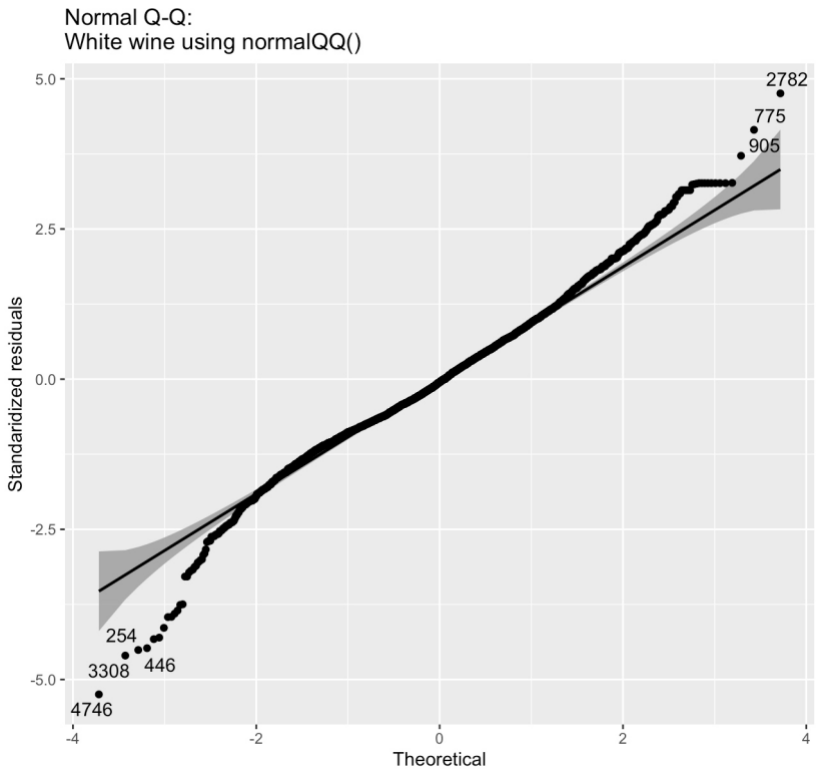
Graph 7: ACF plot for white wine



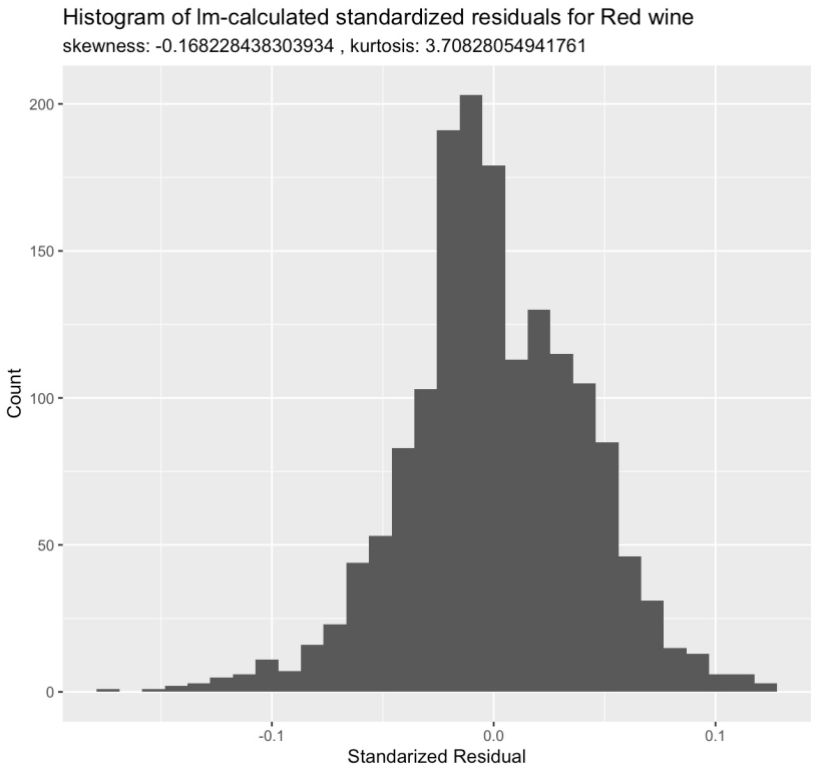
Graph 8: ACF plot for red wine



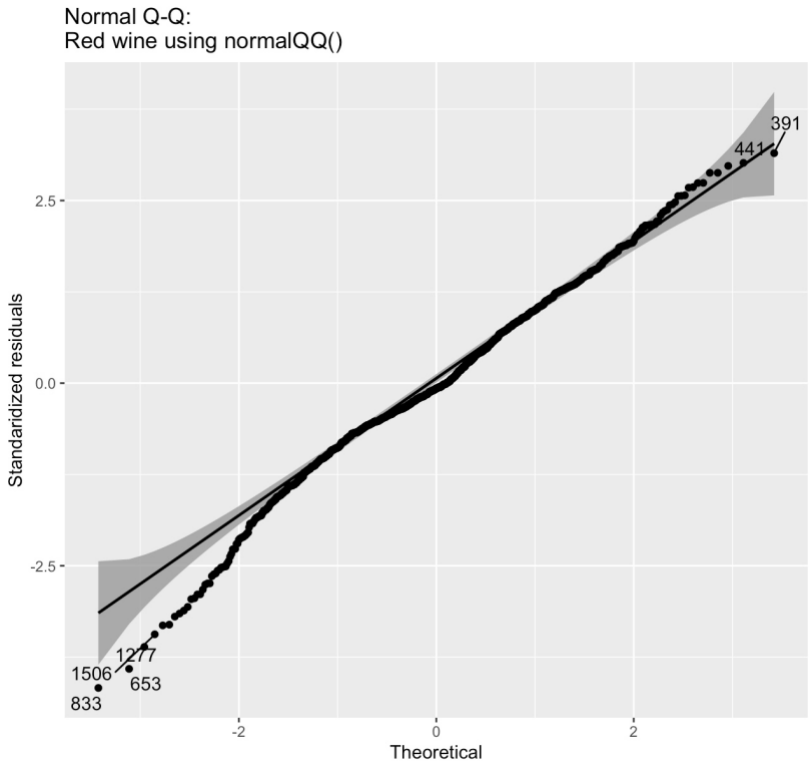
Graph 9: White wine histogram



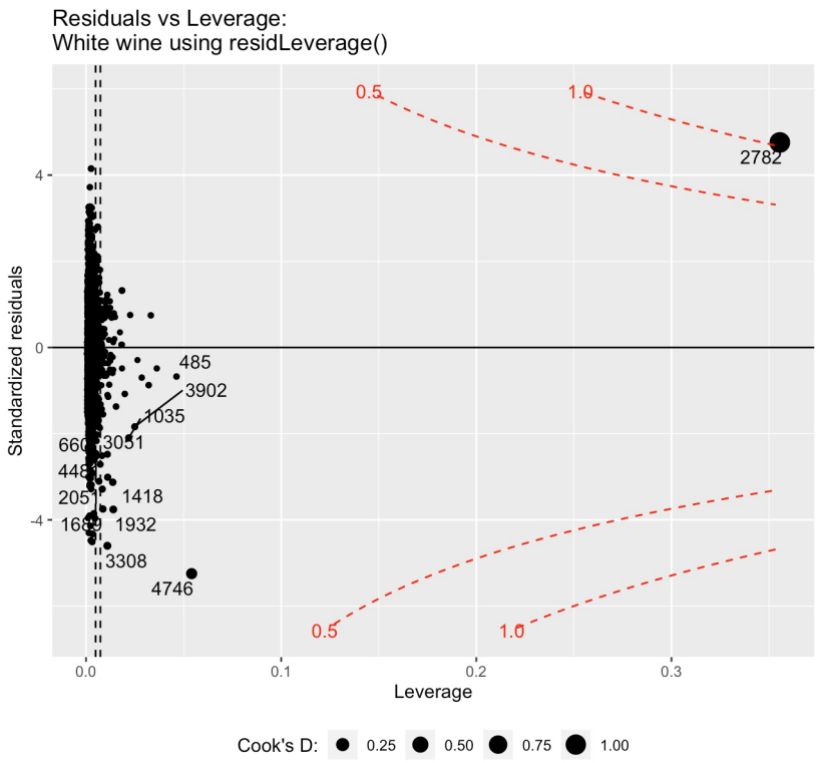
Graph 10: Q-Q plot for white wine



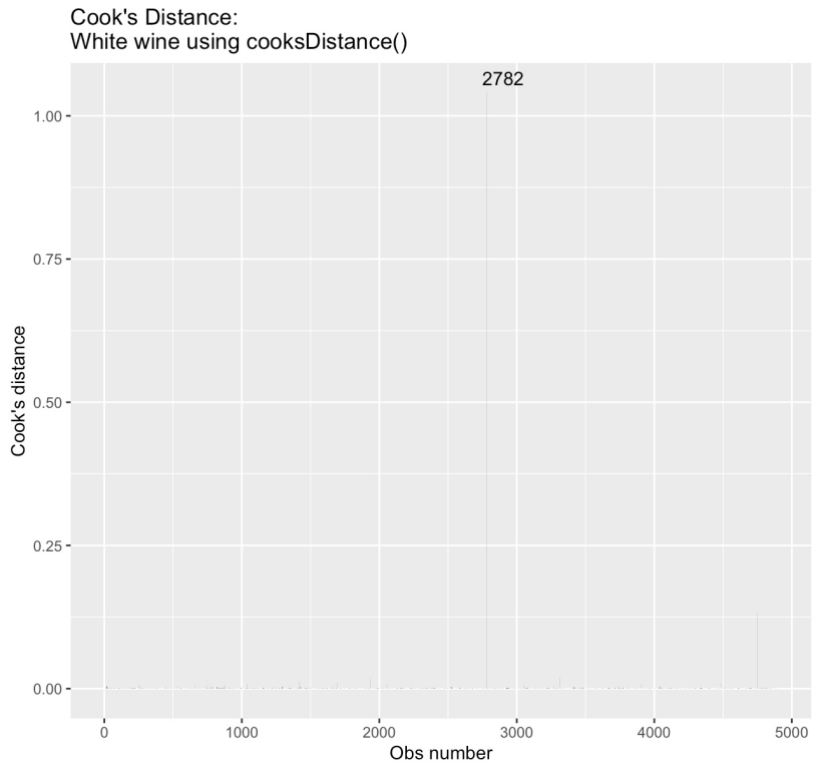
Graph 11: Red wine histogram



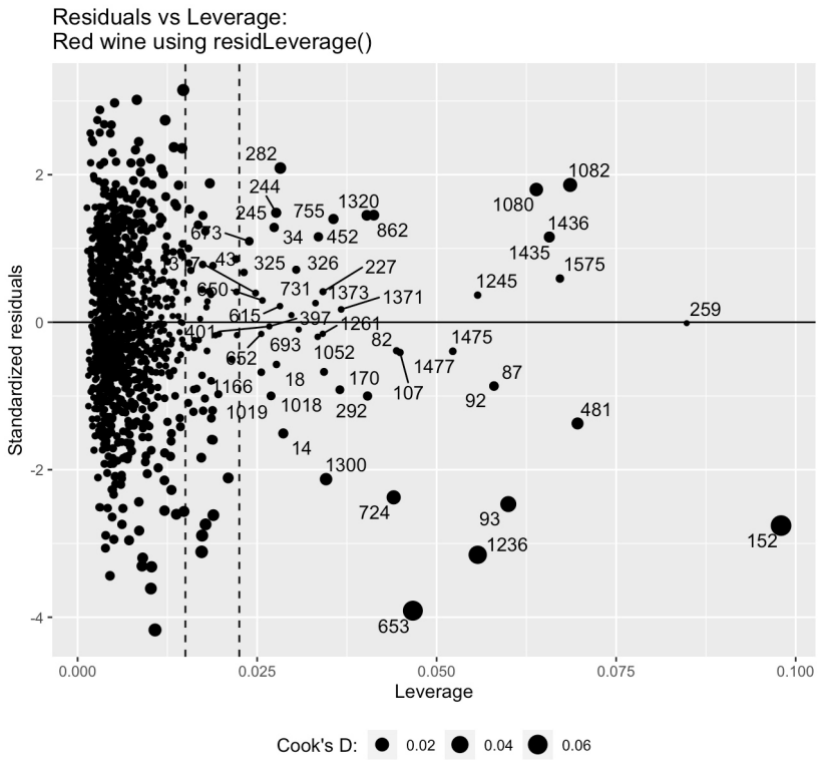
Graph 12: Q-Q plot for red wine



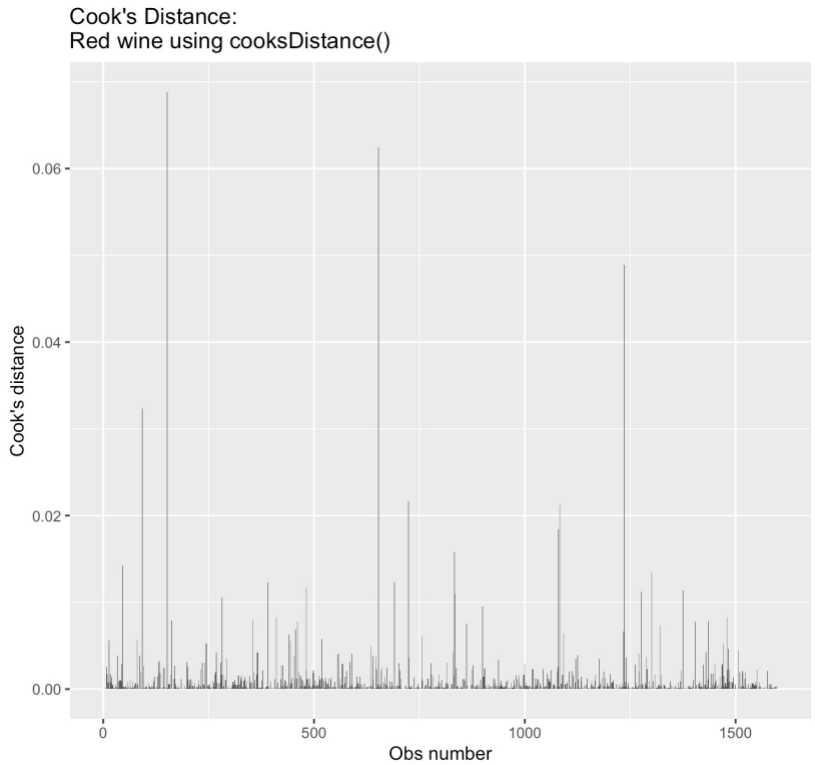
Graph 13: Residuals vs Leverage for White wine



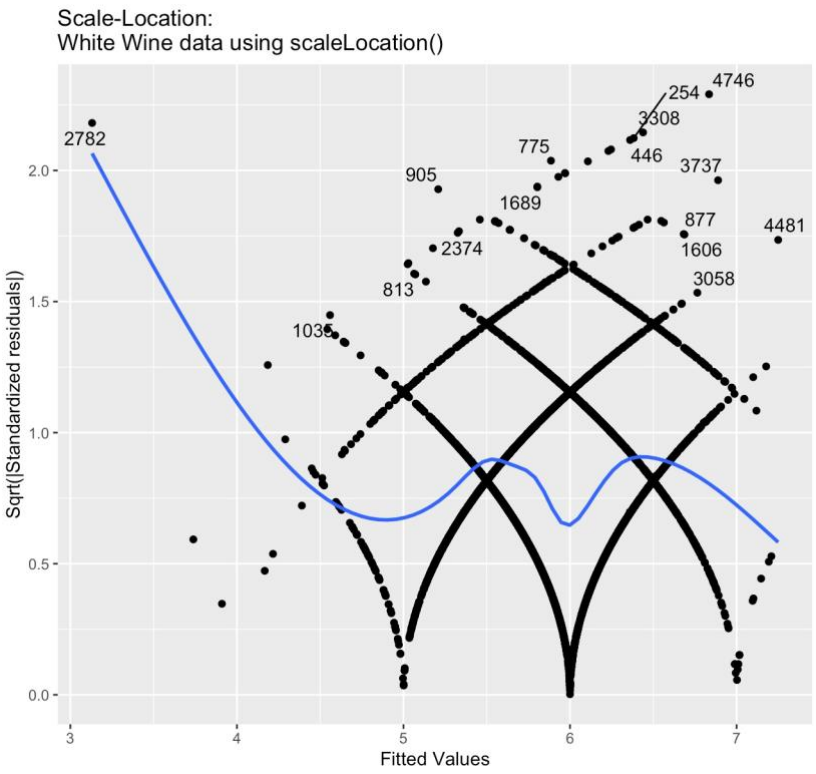
Graph 14: Cook's Distance plot for White wine



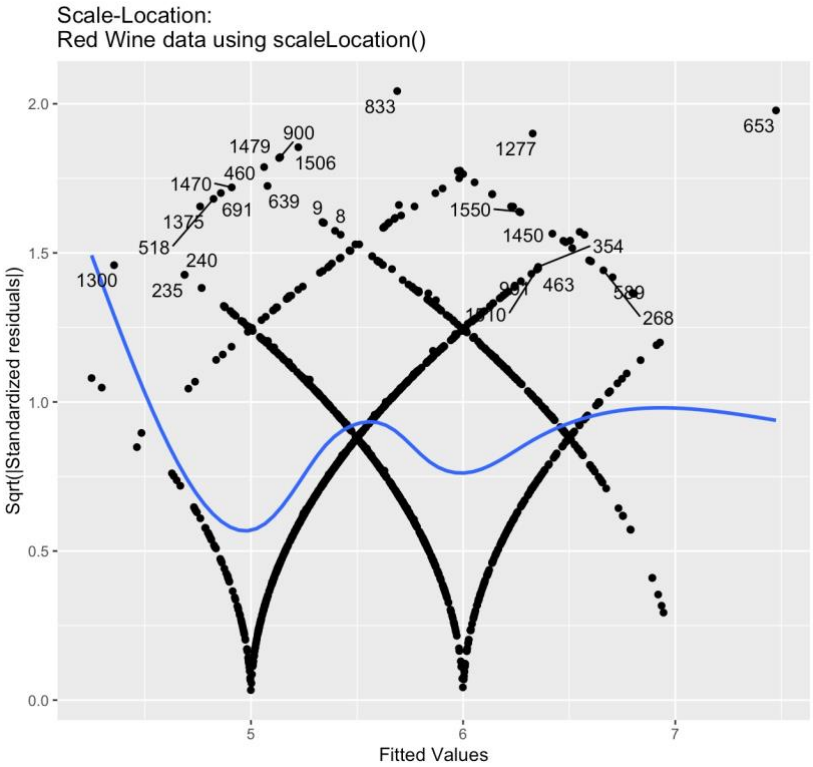
Graph 15:Residuals vs Leverage for Red wine



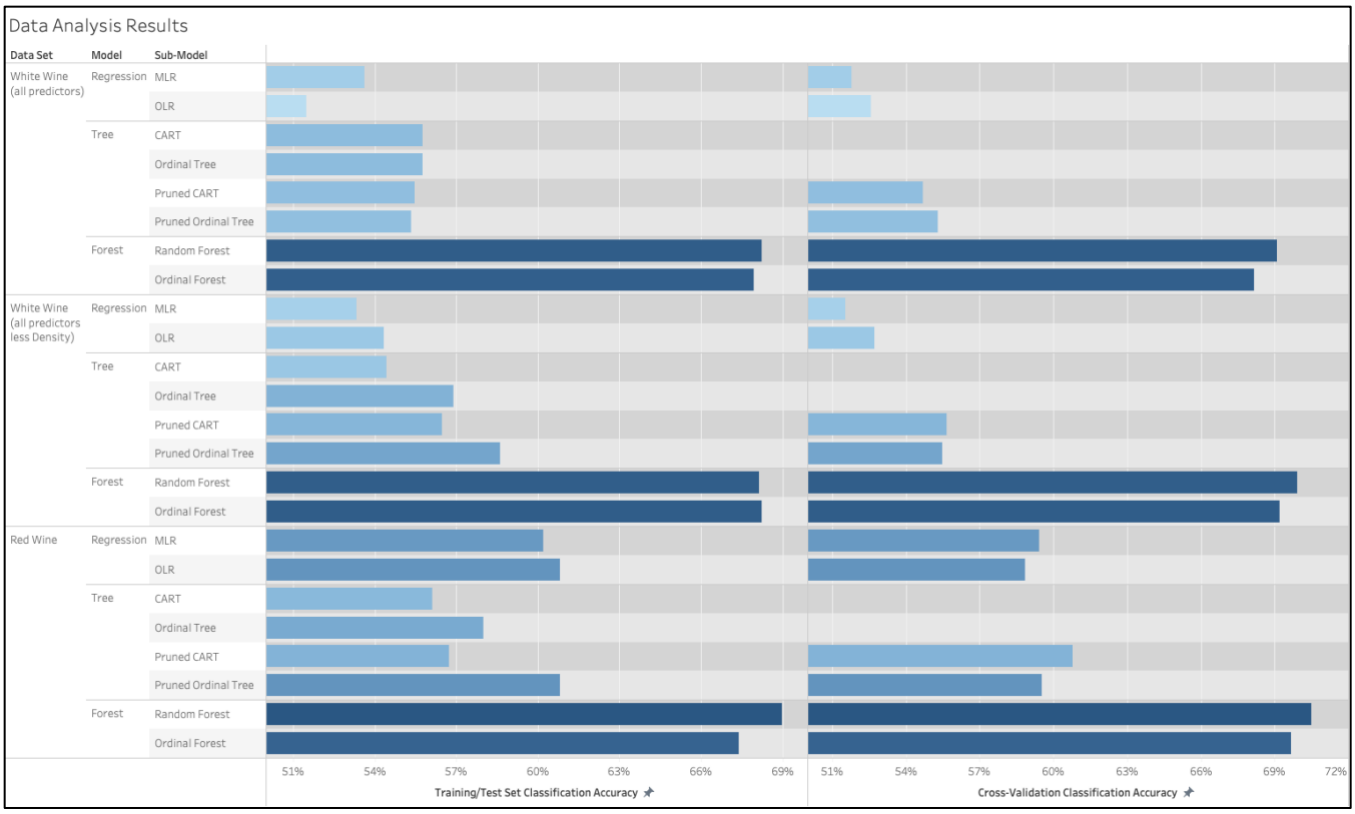
Graph 16: Cook's Distance plot for Red wine



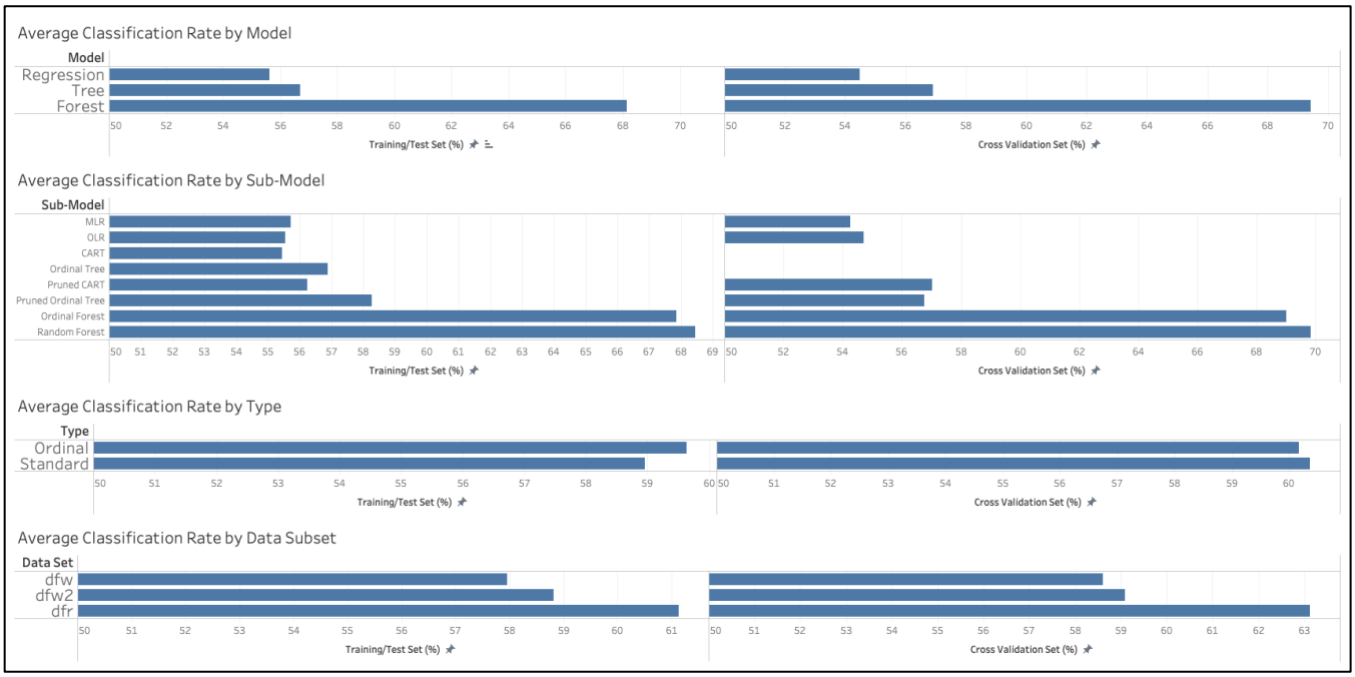
Graph 17: Scale-location plot for White wine



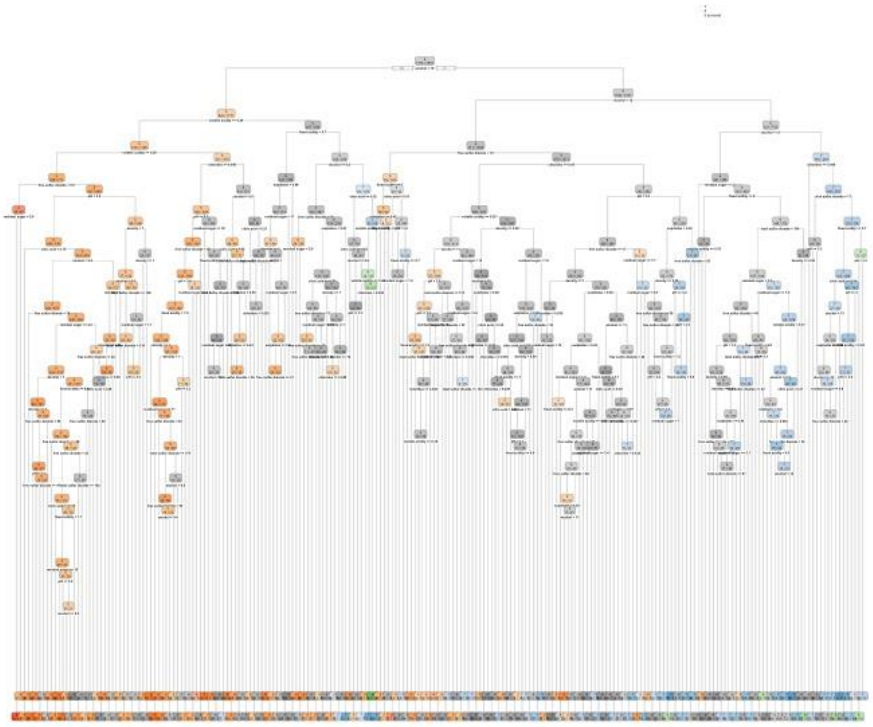
Graph 18: Scale-location plot for Red wine



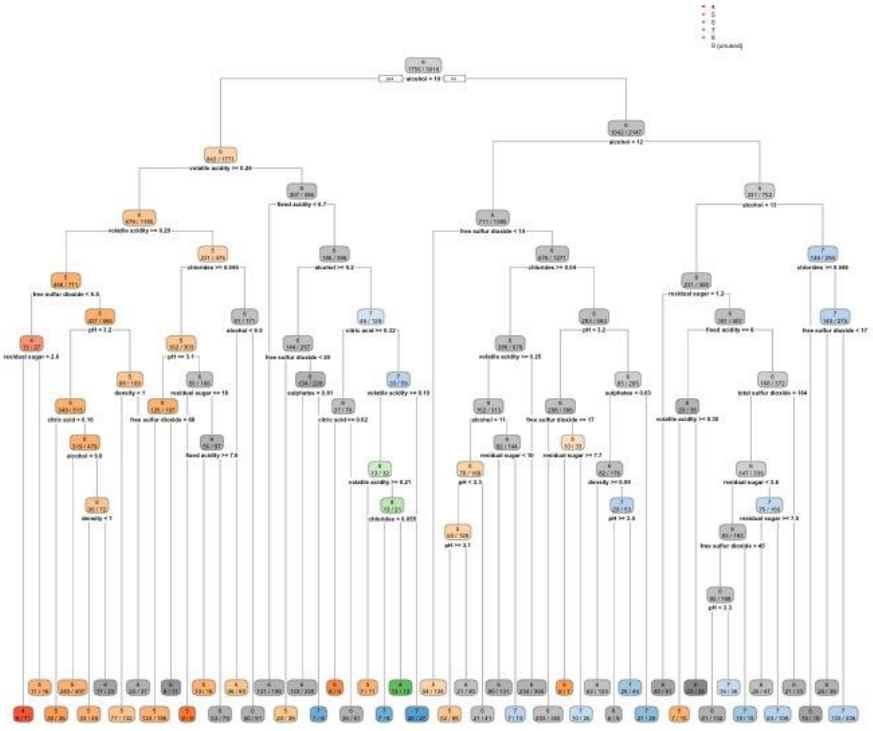
Graph 19:Overall Visualization of Data Analysis Results



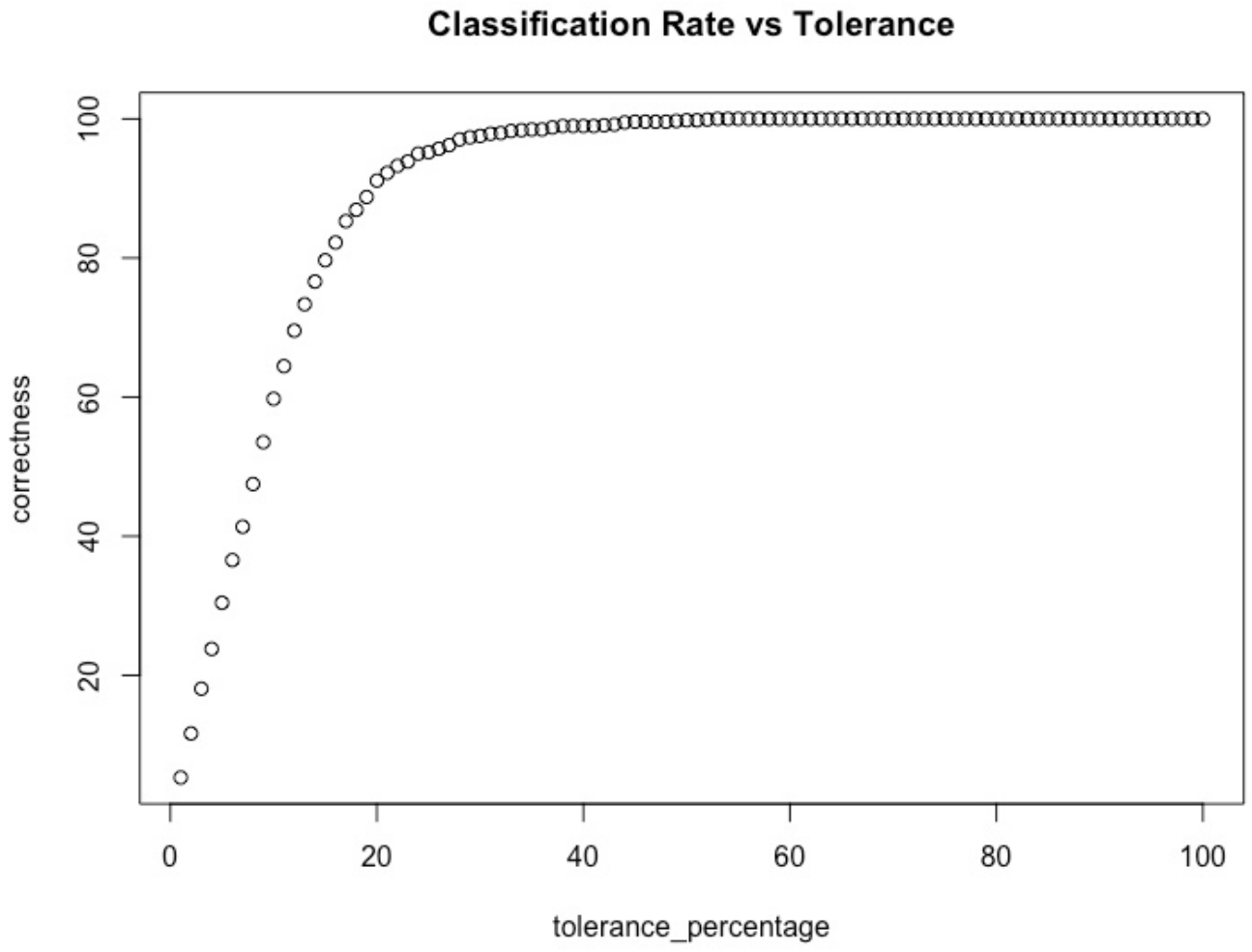
Graph 20: Visualization of Average classification rate by different categories



Graph 21: Unpruned dfw classification tree with training/test set



Graph 22:Pruned dfw classification tree with training/test set



Graph 23: Classification Rate vs Tolerance for future analysis

Code

```
white <- read_csv("winequality-white.csv")

sum(is.na(white))
white <- na.omit(white)


white <- as.data.frame(white)
```

```
red <- read_csv("winequality-red.csv")

sum(is.na(red))
red <- na.omit(red)


dfr <- as.data.frame(red)
```

Code 1: Importing and checking data sets

```
w.mlm <- lm(quality ~.,data = dfw)

summary(w.mlm)
```

```
r.mlm <- lm(quality ~.,data = dfr)

summary(r.mlm)
```

Code 2: Simple multiple linear regression and summary

```
Call:
lm(formula = quality ~ ., data = dfw)

Residuals:
    Min      1Q  Median      3Q     Max
-3.8348 -0.4934 -0.0379  0.4637  3.1143

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            1.502e+02  1.880e+01   7.987 1.71e-15 ***
`fixed acidity`        6.552e-02  2.087e-02   3.139  0.00171 **
`volatile acidity`    -1.863e+00  1.138e-01 -16.373  < 2e-16 ***
`citric acid`          2.209e-02  9.577e-02   0.231  0.81759
`residual sugar`       8.148e-02  7.527e-03  10.825  < 2e-16 ***
chlorides             -2.473e-01  5.465e-01  -0.452  0.65097
`free sulfur dioxide`  3.733e-03  8.441e-04   4.422 9.99e-06 ***
`total sulfur dioxide` -2.857e-04  3.781e-04  -0.756  0.44979
density               -1.503e+02  1.907e+01  -7.879 4.04e-15 ***
pH                     6.863e-01  1.054e-01   6.513 8.10e-11 ***
sulphates              6.315e-01  1.004e-01   6.291 3.44e-10 ***
alcohol                1.935e-01  2.422e-02   7.988 1.70e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7514 on 4886 degrees of freedom
Multiple R-squared:  0.2819,    Adjusted R-squared:  0.2803
F-statistic: 174.3 on 11 and 4886 DF,  p-value: < 2.2e-16
```

```
Call:
lm(formula = quality ~ ., data = dfr)

Residuals:
     Min       1Q   Median       3Q      Max
-2.68911 -0.36652 -0.04699  0.45202  2.02498

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            2.197e+01  2.119e+01   1.036   0.3002
`fixed acidity`        2.499e-02  2.595e-02   0.963   0.3357
`volatile acidity`    -1.084e+00  1.211e-01  -8.948  < 2e-16 ***
`citric acid`         -1.826e-01  1.472e-01  -1.240   0.2150
`residual sugar`       1.633e-02  1.500e-02   1.089   0.2765
chlorides             -1.874e+00  4.193e-01  -4.470 8.37e-06 ***
`free sulfur dioxide`  4.361e-03  2.171e-03   2.009   0.0447 *
`total sulfur dioxide` -3.265e-03  7.287e-04  -4.480 8.00e-06 ***
density               -1.788e+01  2.163e+01  -0.827   0.4086
pH                    -4.137e-01  1.916e-01  -2.159   0.0310 *
sulphates              9.163e-01  1.143e-01   8.014 2.13e-15 ***
alcohol                2.762e-01  2.648e-02  10.429  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.648 on 1587 degrees of freedom
Multiple R-squared:  0.3606,    Adjusted R-squared:  0.3561
F-statistic: 81.35 on 11 and 1587 DF,  p-value: < 2.2e-16
```

Code 3: Output from Code 2

```r
corrplot(
  cor(dfw),
  type = "upper",
  method = "shade",
  title = "White Wine Correlation Plot",
  mar=c(0,0,2,0),
  diag = FALSE,
  tl.srt = 45,
  addCoef.col = "black",
  addCoefasPercent = TRUE
)
```

```r
corrplot(
  cor(dfr),
  type = "upper",
  method = "shade",
  title = "Red Wine Correlation Plot",
  mar=c(0,0,2,0),
  diag = FALSE,
  tl.srt = 45,
  addCoef.col = "black",
  addCoefasPercent = TRUE
)
```

Code 4: Correlation Plot code

```r
par(mfrow = c(2, 2))
plot(w.mlm)
mtext("White Wine Diagnosis Plots", side = 3, line = -2, outer = TRUE)
```

```r
par(mfrow = c(2, 2))
plot(r.mlm)
mtext("Red Wine Diagnosis Plots", side = 3, line = -2, outer = TRUE)
```

Code 5: Initial L.I.N.E assumption plots on simple multiple linear regressions

```r
residFit(w.mlm, title2 = "White Wine using residFit()")

summary(w.mlm)
```

```
residFit(r.mlm, title2 = "Red Wine using residFit()")

summary(r.mlm)
```

Code 6: Code to determine Linearity assumption

```
acf(w.mlm$residuals)

dwt(w.mlm)
```

```
acf(r.mlm$residuals)

dwt(r.mlm)
```

Code 7: Code to determine Independent-Errors assumption

```
> dwt(w.mlm)
 lag Autocorrelation D-W Statistic p-value
   1       0.1896474      1.620592       0
 Alternative hypothesis: rho != 0
> acf(r.mlm$residuals)
>
> dwt(r.mlm)
 lag Autocorrelation D-W Statistic p-value
   1        0.121429       1.75714       0
 Alternative hypothesis: rho != 0
```

Code 8: Durbin-Watson Test function outputs

```r
w.skew <- skewness(w.mlm$residuals)
w.kurt <- kurtosis(w.mlm$residuals)

white.standard_res = rstandard(w.mlm, sd = 16)
ggplot() +
  geom_histogram(aes(white.standard_res)) +
  labs (
    x = "Standarized Residual",
    y = "Count",
    title = "Histogram of lm-calculated standardized residuals for White wine",
    subtitle = paste("skewness:",w.skew,",","kurtosis:",w.kurt, sep = " ")
  )

#Histogram plot

normalQQ(w.mlm, title2 = "White wine using normalQQ()")

#QQ plot
r.skew <- skewness(r.mlm$residuals)
r.kurt <- kurtosis(r.mlm$residuals)

red.standard_res = rstandard(r.mlm, sd = 16)
ggplot() +
  geom_histogram(aes(red.standard_res)) +
  labs (
    x = "Standarized Residual",
    y = "Count",
    title = "Histogram of lm-calculated standardized residuals for Red wine",
    subtitle = paste("skewness:",r.skew,",","kurtosis:",r.kurt, sep = " ")
  )

#Histogram plot

normalQQ(r.mlm, title2 = "Red wine using normalQQ()")

#QQ plot
```

Code 9: Initial Normality assumption tests for White and Red wine

```r
residLeverage(w.mlm, title2 = "White wine using residLeverage()")

#Residuals vs Leverage plot

cooksDistance(w.mlm, title2 = "White wine using cooksDistance()")

#Cook's Distance plot
```

```
residLeverage(r.mlm, title2 = "Red wine using residLeverage()")

#Residuals vs Leverage plot

cooksDistance(r.mlm, title2 = "Red wine using cooksDistance()")

#Cook's Distance plot
```

Code 10: Normality: Outlier identification for White and Red wine

```
residFit(w.mlm, title2 = "White wine using residFit()")

scaleLocation(w.mlm, title2 = "White Wine data using scaleLocation()")

#Equal-variances:
residFit(r.mlm, title2 = "Red wine using residFit()")

scaleLocation(r.mlm, title2 = "Red Wine data using scaleLocation()")

#Equal-variances:
```

Code 11: Equal-variances code

```
corrplot(
  cor(dfw),
  type = "upper",
  method = "shade",
  diag = FALSE,
  tl.srt = 45,
  addCoef.col = "black",
  addCoefasPercent = TRUE
)

vif(w.mlm)
```

```r
corrplot(
  cor(dfr),
  type = "upper",
  method = "shade",
  diag = FALSE,
  tl.srt = 45,
  addCoef.col = "black",
  addCoefasPercent = TRUE
)

vif(r.mlm)
```

Code 12: Multicollinearity test for White and Red wine

```
> vif(w.mlm)
       `fixed acidity`     `volatile acidity`          `citric acid`       `residual sugar`             chlorides  `free sulfur dioxide`
              2.691435               1.141156               1.165215              12.644064               1.236822               1.787880
`total sulfur dioxide`              density                     pH              sulphates                alcohol
              2.239233              28.232546               2.196362               1.138540               7.706957
>
> #Multicollinearity:
> vif(r.mlm)
       `fixed acidity`     `volatile acidity`          `citric acid`       `residual sugar`             chlorides  `free sulfur dioxide`
              7.767512               1.789390               3.128022               1.702588               1.481932               1.963019
`total sulfur dioxide`              density                     pH              sulphates                alcohol
              2.186813               6.343760               3.329732               1.429434               3.031160
```

Code 13: vif() function output for White and Red wine

```r
set.seed(100)

test <- sample(1:nrow(dfw), size = nrow(dfw)/5)
train <- (-test)

#Training and Test sets for dfw

dfw.train <- dfw[train,]
dfw.test <- dfw[test,]

#Create a test and training data set, 20/80 split:
```

Code 14: Code for training/test set used throughout the project

```r
k <- 10 #number of folds

folds <- cvFolds(nrow(dfw), K=k)
folds2 <- cvFolds(nrow(dfw2), K=k)

w.mlm.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))
w.mlm.cv.class2 <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

#Preparing both datasets for cross-validation

for(i in 1:k){
  tr.mlr <- dfw[folds$subsets[folds$which != i],]
  te.mlr <- dfw[folds$subsets[folds$which == i],]

  w.mlm <- lm(quality~., data = tr.mlr)
  w.mlm.pred <- predict(w.mlm, newdata = te.mlr)

  w.mlm.cv.class[i] <- mean(round(w.mlm.pred, digits = 0)==te.mlr$quality)
}
```

Code 15: Example of 10-fold cross-validation code

```r
w.mlm.train2 <- lm(
  quality ~.,
  data = dfw2.train
)
#Create linear regression with all predictors using the training dataset

w.mlm.predict2 <- predict(w.mlm.train2, newdata = dfw2.test)
w.mlm.predict.rounded2 <- round(w.mlm.predict2, digits = 0)

#Round the predicted to a integer so it can be compared to the test set for
# classification

(con_mat <- table(w.mlm.predict.rounded2, dfw2.test$quality))

w.mlm.acc2 <- round(mean(w.mlm.predict.rounded2==dfw2.test$quality)*100,
                    digits = 2)

#Create the confusion matrix and calculate the proportion correct

MLR.results[2,] <- c("dfw2 MLR w/ Train/Test", w.mlm.acc2)

#dfw2 train/test multi linear regression

k <- 10 #number of folds

folds <- cvFolds(nrow(dfw), K=k)
folds2 <- cvFolds(nrow(dfw2), K=k)

w.mlm.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))
w.mlm.cv.class2 <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

#Preparing both datasets for cross-validation

for(i in 1:k){
  tr.mlr <- dfw[folds$subsets[folds$which != i],]
  te.mlr <- dfw[folds$subsets[folds$which == i],]

  w.mlm <- lm(quality~., data = tr.mlr)
  w.mlm.pred <- predict(w.mlm, newdata = te.mlr)

  w.mlm.cv.class[i] <- mean(round(w.mlm.pred, digits = 0)==te.mlr$quality)
}

w.mlm.cv.class

w.mlm.cv.class <- mean(w.mlm.cv.class)
print(paste("The average outputs correctly predicted is",
            round(w.mlm.cv.class*100,digits =2),"%",sep=" "))
```

Code 16: Example of code for multiple linear regression using training/test sets and cross-validation

```r
w.tree2 <- rpart(quality~., data = dfw.train2, method = "class", cp = 0.000001)

w.tree.pred2 <- predict(w.tree2, newdata = dfw.test2, type = "class")

w.tree.class2 <- mean(w.tree.pred2 == dfw.test2$quality)

CART.results[3,] <- c("dfw2 CART w/ Train/Test", round(w.tree.class2*100,
                                                       digits = 2))
#dfw2 simple Classifcation Tree

rpart.plot(w.tree2, extra = 2, digits = 2)

#plot dfw2 simple classification tree

cp2 <- data.frame(w.tree2$cptable)
min.cp2 <- which.min(cp2$xerror)
cp2 <- cp2$CP[min.cp2]

#Find cp with minimum relative error for dfw2 classification tree

w.tree.prun2 <- prune(w.tree2, cp = cp2)

w.tree.pred2 <- predict(w.tree.prun2, newdata = dfw.test2, type = "class")

w.tree.class2 <- mean(w.tree.pred2 == dfw.test2$quality)

CART.results[4,] <- c("dfw2 pruned CART w/ Train/Test", round(w.tree.class2*100,
                                                             digits = 2))

#Created pruned min-error tree for dfw with train/test set
```

Code 17: Pre and post pruned classification tree using training/test set

```
k <- 10 #number of folds

folds <- cvFolds(nrow(dfw), K=k)
folds2 <- cvFolds(nrow(dfw2), K=k)

w.tree.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))
w.tree.cv.class2 <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

#Preparing both datasets for cross-validation

for(i in 1:k){
  tr.tree <- dfw[folds$subsets[folds$which != i],]
  te.tree <- dfw[folds$subsets[folds$which == i],]

  w.tree.cv <- rpart(quality~., data = tr.tree, method = "class",
                     cp = 0.000001)

  cp.cv <- data.frame(w.tree.cv$cptable)
  min.cp.cv <- which.min(cp.cv$xerror)
  cp.cv <- cp.cv$CP[min.cp.cv]

  w.tree.prun.cv <- prune(w.tree.cv, cp = cp.cv)

  w.tree.pred.cv <- predict(w.tree.prun.cv, newdata = te.tree, type = "class")

  w.tree.cv.class[i] <- mean(w.tree.pred.cv == te.tree$quality)

}

w.tree.cv.class

w.tree.cv.class <- mean(w.tree.cv.class)
print(paste("The average outputs correctly predicted is",
            round(w.tree.cv.class*100,digits =2),"%",sep=" "))

CART.results[5,] <- c("dfw CART w/ 10-fold CV", round(w.tree.cv.class*100,
                                                      digits=2))
```

Code 18: Pruned classification trees using cross-validation

```r
#Training and Test sets for dfw2

w.rf2 <- randomForest(quality~., data = dfw.train2, mtry =10, ntree = 500,
                      importance = TRUE)

w.rf.pred2 <- predict(w.rf2, newdata = dfw.test2)

w.rf.class2 <- mean(w.rf.pred2 == dfw.test2$quality)

RF.results[2,] <- c("dfw2 random forest w/ training/test set",
                    round(w.rf.class2*100,digits=2))

importance(w.rf2)
varImpPlot(w.rf2)

#dwf2 random forest with training/test set

k <- 10 #number of folds

folds <- cvFolds(nrow(dfw), K=k)
folds2 <- cvFolds(nrow(dfw2), K=k)

w.rf.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))
w.rf.cv.class2 <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

#preparing both datasets for cross-validation

for(i in 1:k){
  tr.rf <- dfw[folds$subsets[folds$which != i],]
  te.rf <- dfw[folds$subsets[folds$which == i],]

  w.rf.cv <- randomForest(quality~., data = tr.rf, mtry =11, ntree = 500,
                          importance = TRUE)

  w.rf.pred.cv <- predict(w.rf.cv, newdata = te.rf)

  w.rf.cv.class[i] <- mean(w.rf.pred.cv == te.rf$quality)

}

w.rf.cv.class

w.rf.cv.class <- mean(w.rf.cv.class)
print(paste("The average outputs correctly predicted is",
            round(w.rf.cv.class*100,digits =2),"%",sep=" "))

RF.results[3,] <- c("dfw Random Forest w/ 10-fold CV",
                    round(w.rf.cv.class*100,digits=2))

#dfw random forest with cross-validation
```

Code 19: Random Forest code using training/test set and cross-validation set

```
w.ordfor <- ordfor("quality", data = dfw.train, mtry = 11,
                   nsets = 100, ntreeperdiv = 10, ntreefinal = 500,
                   nbest = 1, npermtrial = 50)
```

Code 20: Example of ordfor() argument tuning

```
correctness <- rep(NA, 100)

for(j in 1:100){
tol_value <- j*0.01
tolx <- w.mlm.predict+(w.mlm.predict*tol_value)
tolz <- w.mlm.predict-(w.mlm.predict*tol_value)
yorn <- 0
for(i in 1:nrow(dfw.test)){
  yorn[i]<-between(dfw.test$quality[i],tolz[i],tolx[i])
}
yorn
correctness[j] <- sum(yorn)/length(yorn)
}
correctness <- correctness*100
tolerance_percentage <- 1:100
plot(tolerance_percentage,correctness,
     main = "Classification Rate vs Tolerance")
```

Code 21: Tolerance range for predictive modeling