**Vinho Verde Wine Quality**

**Abby Lavoie**

**Math 748, Fall 2022**

**Abstract:** Vinho Verde white wines make up a large percentage of wines produced in the Minho region of Portugal with exports to the United States and Germany growing yearly. There is a thorough certification process that involves analyzing physiochemical properties as well as sensory quality, including visual and gustatory analysis. Due to the changing focus on the environment and increasing concern over clean water access, it is important to minimize the environmental impact of the wine making process. Machine learning techniques can be applied to reliably predict the quality of these wines to help vintners focus on producing only the top quality or at least to reduce their efforts regarding low quality wines to save resources and minimize environmental damage. This paper explores and compares several analytical classification methods including several linear methods, K-Nearest Neighbors, Support Vector Machines, Classification trees, and Random Forest. The best predictor of quality was produced by Random Forest classification. The most important predictor of quality identified by all methods was alcohol. Future analysis should include wines from diverse areas and a more objective measure of quality as well as relation to consumer preferences and habits.

**Introduction**

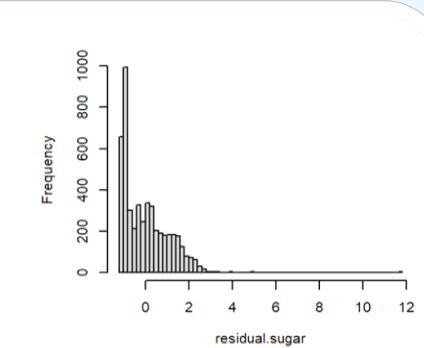
Vinho Verde wines are grown in the northern region of Portugal known as the Minho region. This region is known for diverse microclimates and prevalence of access to clean water, however this has been changing in recent years. Historically most of the inhabitants of this region have private vineyards and personal production of Vinho Verde wines. In recent times there has been a move toward larger commercial vineyards. Recent research has determined that the largest impact on the environment is due to fertilizers used during the growth of grapes. Any efforts to reduce this aspect of wine production would have a positive impact on the environment and longevity of the wine tradition in this region.

The name Vinho Verde translates literally as young wines referring to their short fermentation time (3- 6 months). They are characterized by carbonation and medium-alcohol levels and are enjoyed as a light, refreshing alcoholic beverage typically in the summer. They encompass red, white, and rosé varieties though the majority, about 86%, are white. Almost 30% of the wines exported from Portugal are Vinho Verde. Most of these exports go to the United States and Germany. These wines have a strict certification process where they undergo about 20 different tests (CVRVV) that include basic chemical analyses as well as various sensory tests. The focus of this work is to determine the most important of the chemical analyses and to develop models to predict the gustatory quality.

**Data Cleaning/Exploration**

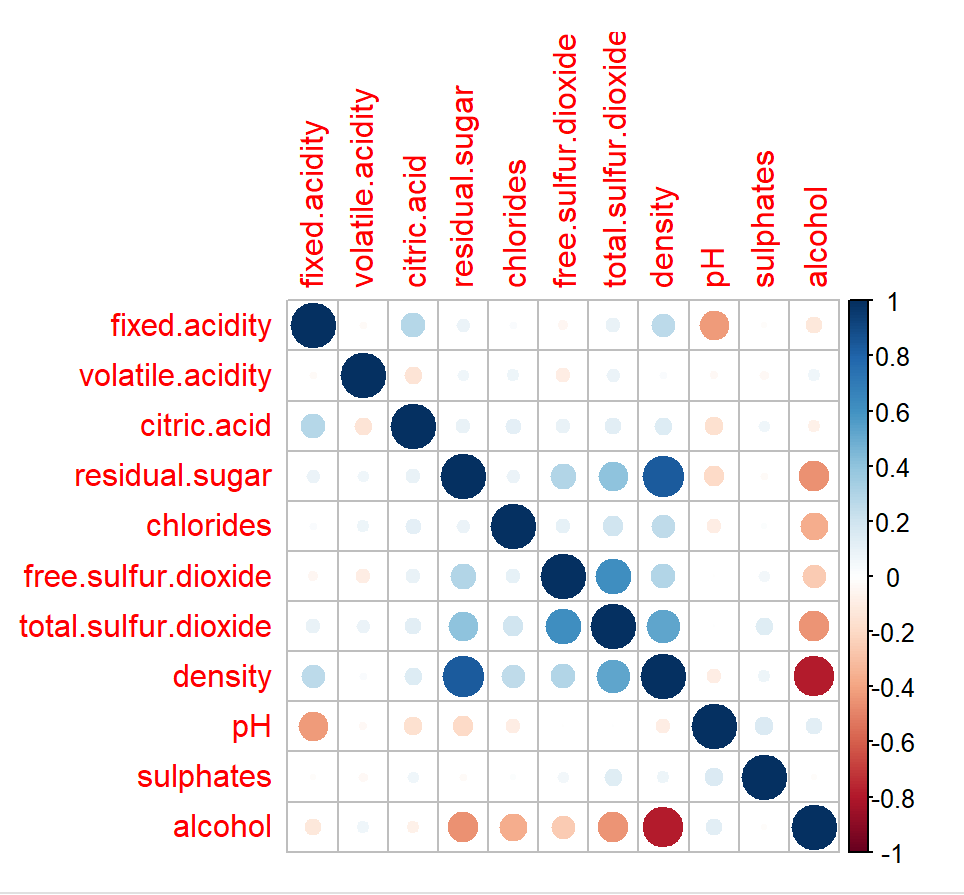
The data set of focus includes only white varieties of the Vinho Verde wines. This data was collected over a period of 3 years and includes only the eleven chemical factors that had no missing data to preserve all 4898 samples. The eleven predictor variables, all measured in terms of concentration, are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. The response variable is taste quality which was produced by taking the median score of at least 3 different expert wine tasters using a scale of 0 (bad) to 10 (great). The ranking of quality in this dataset only includes values as low as 3 and as high as 9, although very few samples of these (20 and 5, respectively). The most common quality ranking was 6 (2198 samples with this score) and the next most common was 5 (1457 samples). Since the rankings were so unevenly represented, I chose to create a binary quality measure where rankings of 6+ were considered “good” represented by 0 (3258 samples) and all the other rankings were “poor” represented by 1 (1640 samples). I chose to represent the poor quality as 1 because it seemed more important to be able to predict which wines are low quality to reduce the efforts and resources focused on their production.

My next focus was exploring the feature variables in terms of magnitude, distribution, and relation to each other. The magnitudes were much different, and the distributions were mostly skewed or non-normal so I chose to center and scale them. Scaling the variables largely reduced the uneven distributions. To illustrate this effect please see the before and after histograms of one of the least normally distributed variables, residual sugar. In the before-histogram it is clear that this variable is heavily right skewed but after scaling it is more symmetrical. Since the normality assumption behind many methods does not need to be strictly true, I decided not to apply a transformation to the data.

 Chart, histogram

Description automatically generated

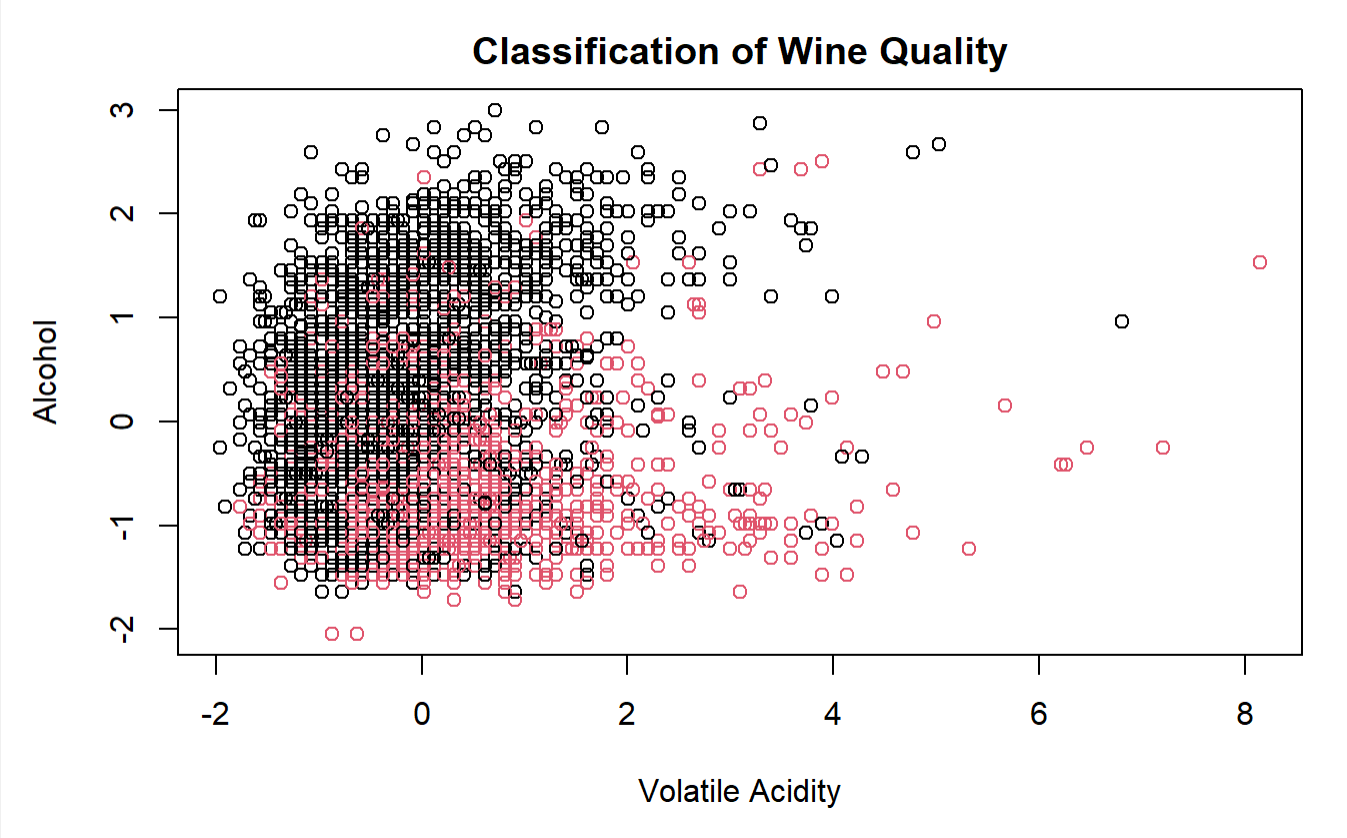
Feature selection methods typically work best without strongly correlated features so I next analyzed the pairwise correlations. I discovered that many of the features were not strongly correlated strongly (please see the correlation plot below). However, density was highly correlated with residual sugar (0.84) and alcohol (-0.78). The primary researchers of this dataset did not find density to be in the top 7 important variables for predicting quality and so I removed this feature from the data. Next I looked at pairwise plots to see if there were interesting relationships I wanted to keep in mind for further analysis. Scatterplots of different pairs of variables hinted that alcohol would help separate the two classes of qualities which I kept in mind as I moved into feature selection.



Before going any further, I first randomly separated the observations into a training set (3918 observations) and testing set (980 observations). Both the training and testing sets preserved the 2:1 proportion of good to poor quality wines found in the overall data (2601:1317 in the training set and 657:323 in the testing set). From this point, every method applied to the data was first applied to the training set and then challenged/optimized/checked with the test set.

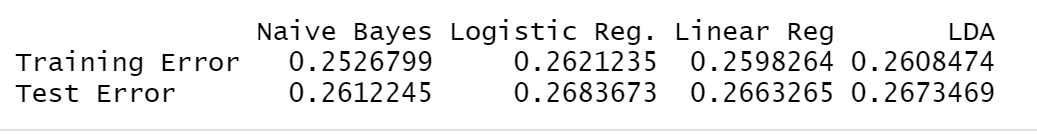
**Feature Selection**

Feature selection methods can be used to reduce the dimensions of the data and to improve interpretability with little or no cost to prediction accuracy. I used the regsubsets() feature in R and the exhaustive search criterion to find the best models of all sizes from 1 to 10 predictors. From the plot of adjusted R-squared values I saw the biggest increase from the models using 1 to 2 variables. After that the measure changed very slowly. Using the test set to choose the model which minimized the cross-validation error resulted in the 5-variable model. The difference in cross-validation errors between the 2 and 5 variable models was only 2.5%. Going forward to the linear classifier analysis, I used both the best 2-predictor and best 5-predictor subsets to test if sacrificing interpretability resulted in an appreciable increase in prediction accuracy (would it produce more than a 2.5% increase in accuracy to use 5 predictors?). The best model found through cross-validation contains these 5 predictors: fixed acidity, volatile acidity, residual sugar, sulphates, and alcohol. The fitted model from the feature selection process is: 0.33594414 + 0.02530612(fixed acidity) + 0.12418652(volatile acidity) - 0.06114755(residual sugar) - 0.02625800(sulphates) - 0.21605245(alcohol). Since the best overall model (5-factors) improved the test classification error of the best 2-factor model by only 2.55%, I believe the improvement in interpretation and visualization is justification for dropping 3 of the factors. This fitted model from the feature selection process is: 0.3355817 + 0.1180736(volatile acidity) - 0.1907672(alcohol). Both models include alcohol (as did the best 1-predictor model) and volatile acidity. The quality of the wine seems to be distinctly separable using alcohol and volatile acidity (please see graph below).



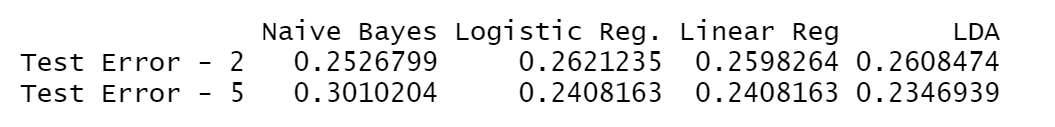
**Analysis/Results**

Now with these two subsets of features I moved on to the first analytical method: linear classifiers. I compared the results of Naive Bayes, Linear Regression, LDA, and Logistic Regression for both the 2 and 5 factor models. The logistic regression analysis resulted in the model: which can be interpreted as: for a one unit increase in alcohol as long as volatile acidity is fixed, the odds of poor quality decreases by 66.2% and for a one unit increase in volatile acidity with alcohol fixed the odds of poor quality increases by 88%. This model suggests that poor quality Vinho Verde wine is characterized by low alcohol and high volatile acidity. The linear regression analysis produced this model: . For a one unit increase in alcohol with volatile acidity fixed at some level, the predicted quality will increase toward good by 0.1908 units (since 1 is poor and 0 is good quality). A one unit increase in volatile acidity with alcohol fixed at some level will produce a decrease in predicted good quality of 0.1181. Both models (please see the plot below for visualizing these two models) produced a larger coefficient for alcohol which implies that variable is at least slightly more important to predicting quality than volatile acidity. The lowest test classification error was associated with the Naïve Bayes method followed by the linear regression. Please see the table below for the training and testing classification error associated with all four linear classifiers. Both the linear and logistic regression models produced a similar decision boundary. Please see the scatterplot below for illustration of these.

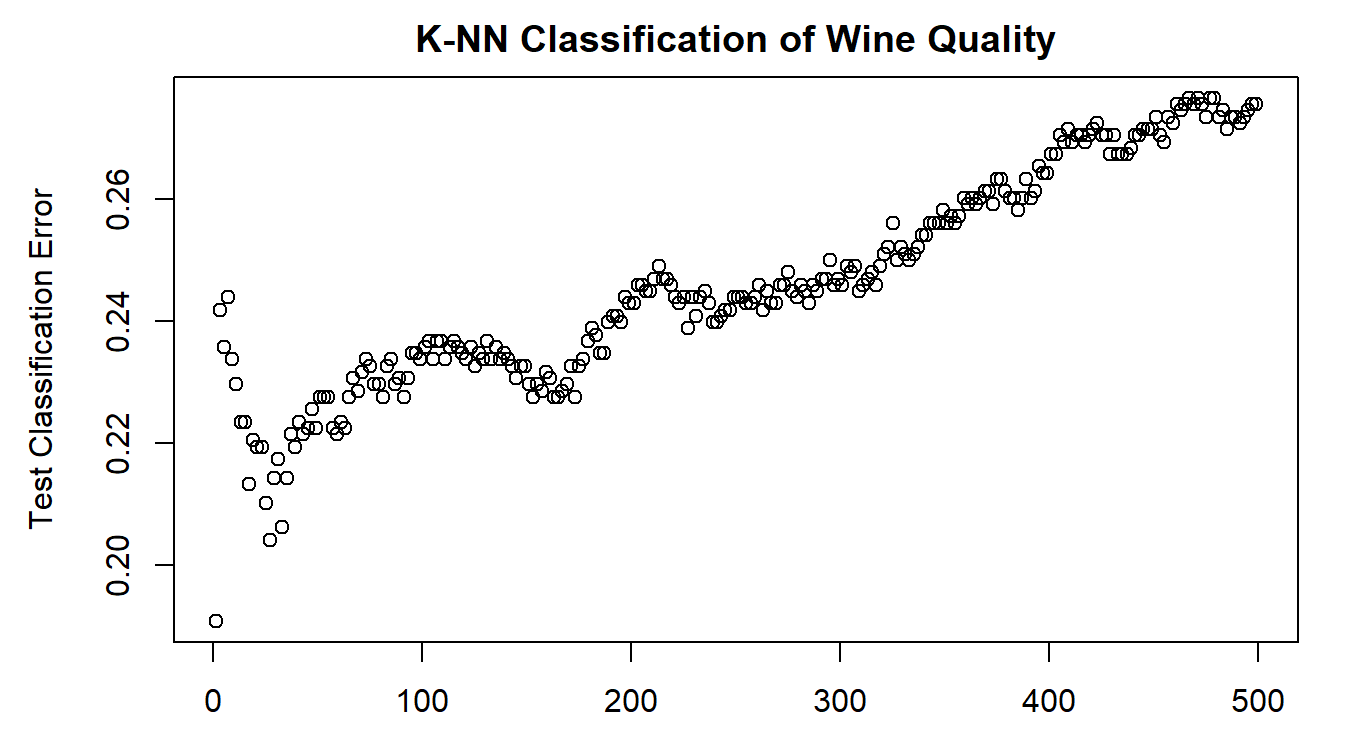
Chart, scatter chart

Description automatically generated

The model produced from logistic regression analysis using 5 factors is as follows: . This model gives alcohol a bigger influence on quality than the 2-predictor model; for a one unit increase in alcohol with all other variables fixed at some constant level the odds of poor wine decreases by about 73%. The magnitude of the coefficient of volatile acidity also increased in this model and gives a similar interpretation as the 2-predictor model. The linear regression model is This model also produced a marginal increase in the coefficients of alcohol and volatile acidity. The 5-variable model did not improve the test error more than about 2.5% and in the case of the Naïve Bayes it worsened which I believe is due to overfitting. Please see the table below for a comparison of the test classification error for the 2 vs. 5 predictor models. These results confirm that the improvement in classification accuracy is not necessarily worth the sacrifice of interpretability.

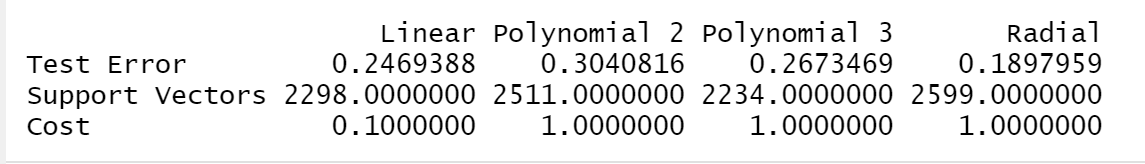


The next method applied to this dataset was K-Nearest Neighbors. In order to optimize the value of K, I used a sequence of values from 1 to 500 (by 2, so K=1, 3, 5, …, 499) and found the test classification error. As the plot below demonstrates, the error was optimized for small values of K and generally increased as K approached 500. With K=500 there were too many ties and test error could not be calculated. The minimum test error of 0.1908163 occurred for k=1. The next lowest test error was 0.2040816 for k=14 (please see second plot below for close up of K=1, …,20). The test error rate found here was lower than any of the linear classifiers.

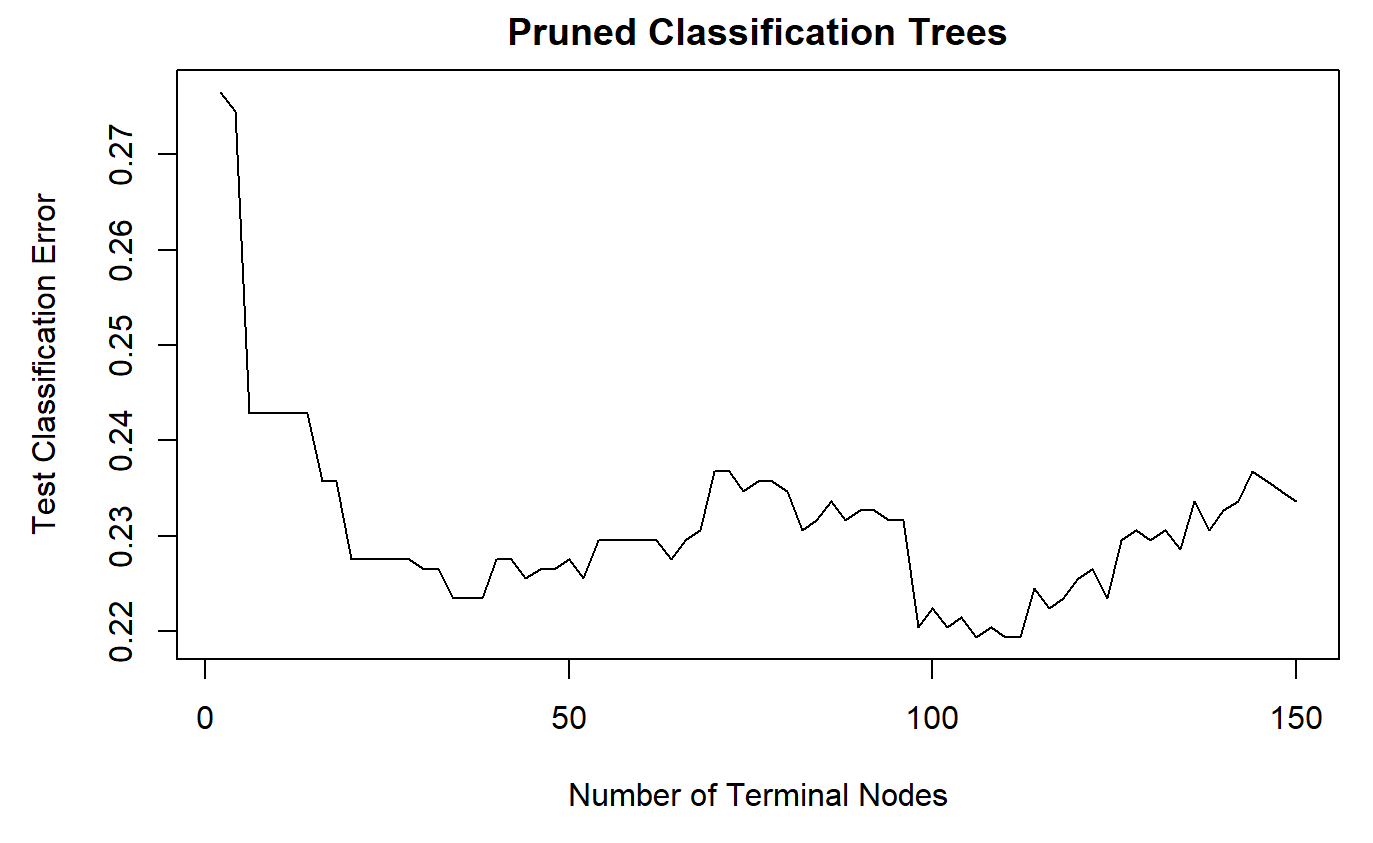
Chart, scatter chart

Description automatically generated

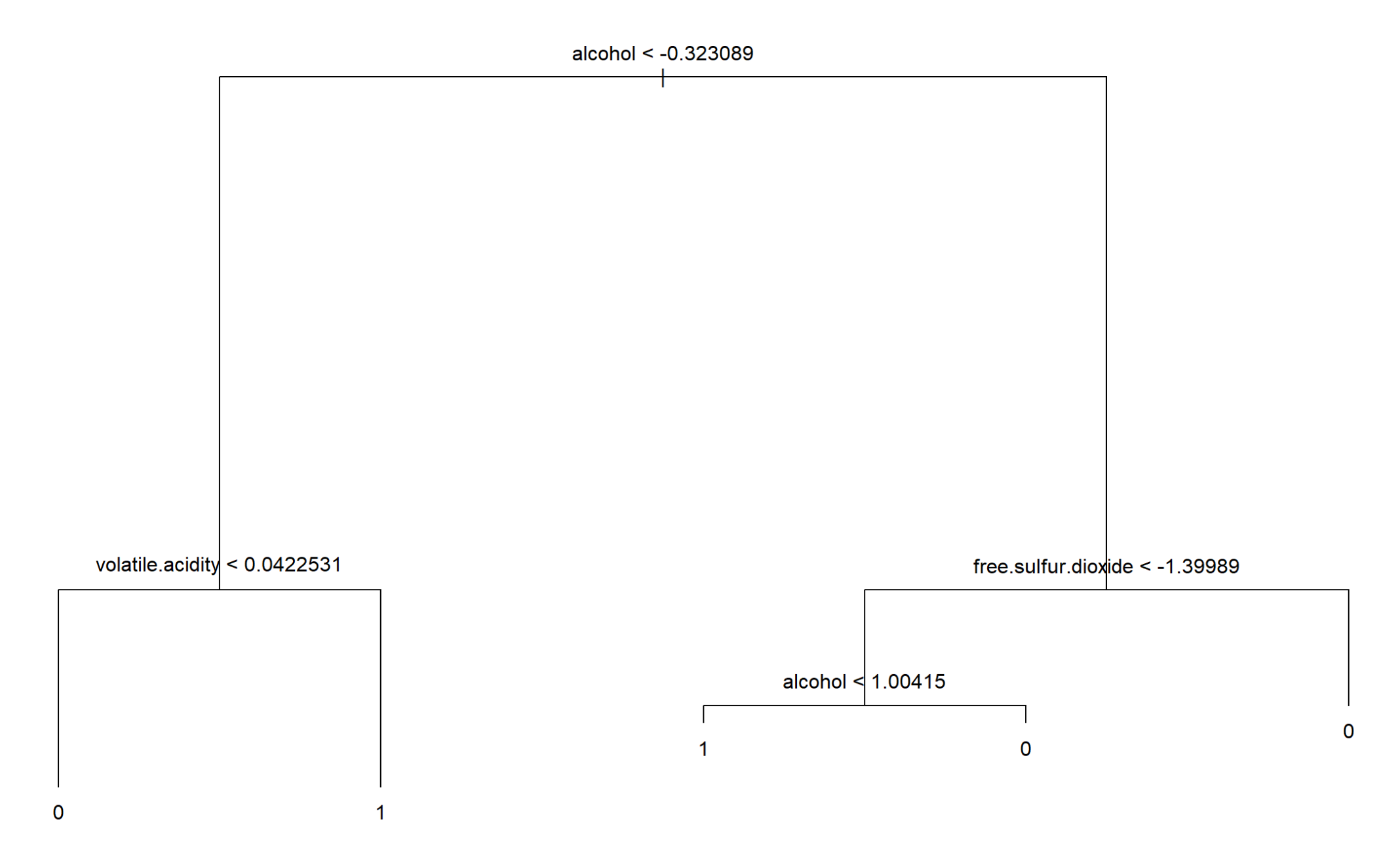
The next method I applied to the data was SVM using different kernels and costs. I first used a linear kernel and tuned the model by comparing many values of the cost (0.001, 0.01, 0.1, 1,5,10, and 100). The best model used a cost of 0.1. While this model produced a comparable test error to the linear classifiers, I was hopeful that by exploring other kernels I could improve this. Next, I used the polynomial kernel and compared polynomials of degree 2 and 3. Due to limitations of my computer I was unable to compare different values of cost and gamma, so I chose a popular level of gamma (0.1) and compared several values of cost. The optimal cost determined was 1 for both polynomial models. These produced worse test errors than the linear model and increased the number of support vectors necessary. The last support vector machine I applied used a radial kernel. My computer was able to compare the results of several values of cost (0.1,1,10,100,1000) and gamma (0.5,1,2,3,4). The ideal combination found used a cost of 1 and gamma of 0.5 This model produced the lowest test error (0.1897959) of all the support vector approaches and the other methods. Below is a chart comparing the different support vector methods and includes the test error, number of vectors, and cost used.



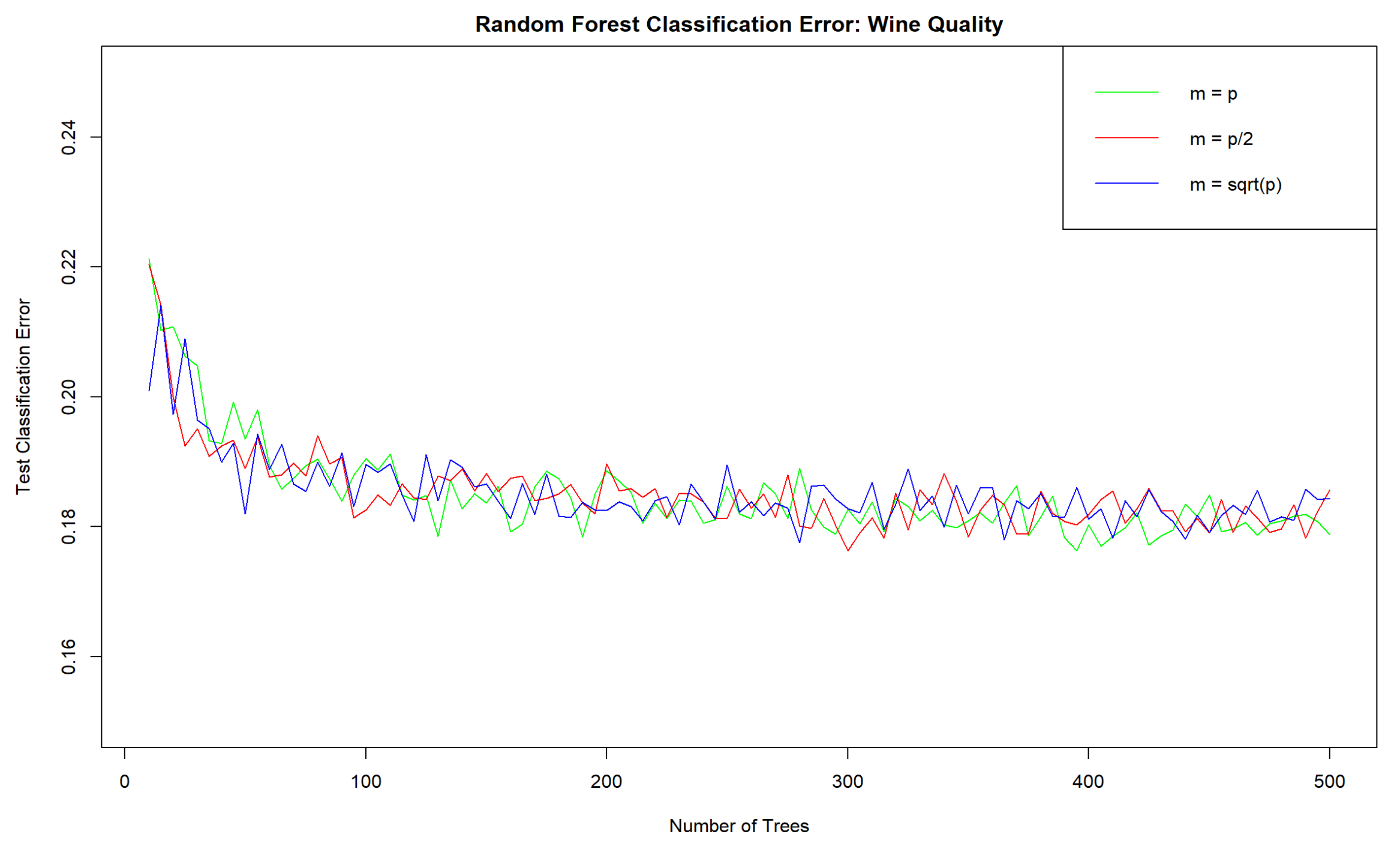
The next method I chose to apply to the data was chosen for interpretability and nonlinearity; a classification tree using the Gini index to guide the branching process. The original tree produced had many terminal nodes and was far too complicated for interpretation, so cross-validation pruning was applied to find the best tree. (Please see plot below for test classification error vs. number of terminal nodes.) This resulted in a tree with 106 terminal nodes.



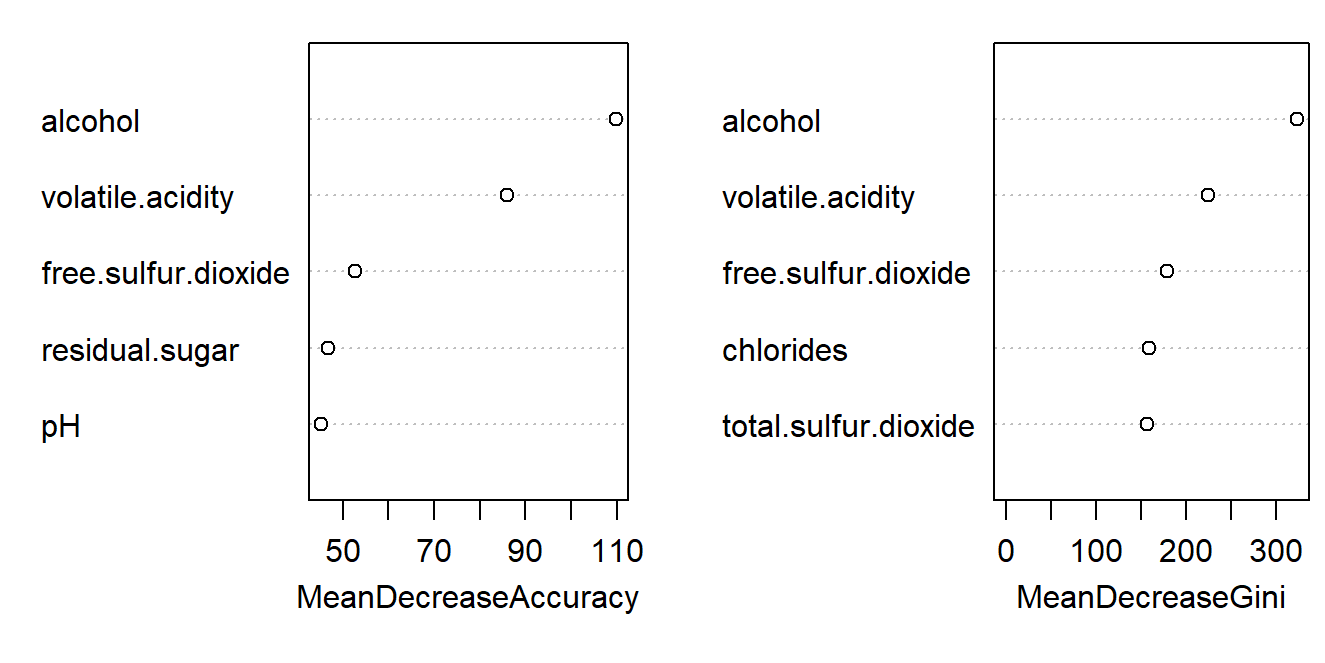
The easiest to interpret with only 3 misclassifications more that the cross-validated best was a tree with 5 terminal nodes. The splits for this tree were made using alcohol, volatile acidity, and free sulfur dioxide. Please see the tree below for a graphical representation of the following interpretation. If the alcohol was less than -0.32 and the volatile acidity was less than 0.04 then the wine was predicted to be good. However, if the alcohol was less than -0.32 and the volatile acidity was more than 0.04 the wine was predicted to be poor. The interpretation for wines with alcohol measured to be larger than -0.32 is a little more complex. For these wines, if the free sulfur dioxide was less than -1.4 and the alcohol was less than 1, then the wine was predicted to be poor. However, if the free sulfur dioxide was less than -1.4 and the alcohol was greater than 1 then the wine was predicted to be good. Also, if the alcohol was greater than -0.32 and the free sulfur dioxide was greater than -1.4, the wine was predicted to be good. To summarize, if alcohol is less than -0.32 then the next most important quality is volatile acidity for separating good and poor quality wines. If the alcohol is between -0.32 and 1, then free sulfur dioxide will help differentiate between good and poor quality wines. Overall, this provides reasonably easy to follow guidelines for monitoring the chemical analysis and has a similar interpretation to the linear classifiers with 2-predictors. However, the test classification error for the full tree was 0.24286 and for the best cross-validated tree (106 terminal nodes) was 0.21939. So, while the trees provide an easy to interpret guide for predicting quality, the test error is almost as high as the linear classifiers and would not be reasonable to follow for discarding wines.



A better application of classification trees, in terms of prediction error but not interpretability, is through the Random Forest analysis which is the final analytical method applied to the data set. Random Forest improves on the prediction accuracy of a single classification tree by using bootstrap aggregation of B trees with only a subset, m, of predictors considered at each split. Three different values of m were considered and compared: (i.e. bagging where all predictors are considered), , and where p is the total number of predictors, in this case 10. The number of trees used for aggregation went from 1 to 500. Please see the plot below for this comparison.



From this plot it is obvious that no single definition of m was the best overall nor a specific number of trees although, in general, the test classification error improves as B approaches 500. The minimum test error achieved here was 0.1763064 when and B=300. This test error is an improvement on all the previous methods applied. Exploring this combination more, I can see that the most important variable is alcohol which agrees with all previous methods. Please see charts below for importance plots based on Mean Decrease Accuracy and Mean Decrease Gini. The next most important variables determined by the Random Forest method are volatile acidity and free sulfur dioxide which agrees with the easiest to interpret classification tree discussed above.



**Comparison/Summary**

The test classification error of the best of each method (as well as any important features of the method) applied to the data are as follows: Best Linear Classifier: 0.2408163 (5-predictor model, linear and logistic regression), Best KNN: 0.1908163 (K=0) and next best: 0.2040816 (K=14), Best SVM: 0.1897959 (Radial kernel, cost =1, gamma=0.5, support vectors= 2599), Best Classification Tree: 0.2193878 (106 terminal nodes), Best Random Forest: 0.1763064 (, 300 trees). Overall, Random Forest achieved the lowest test classification error of 0.176.

Each method identified the same two most important variables for classifying these wines as good or poor quality: alcohol and volatile acidity with alcohol as the topmost important variable. The next most important variables differed depending on the method: Random Forest identified free sulfur dioxide then chlorides whereas feature selection identified residual sugar, sulfates, and fixed acidity. One of the hallmarks of Vinho Verde wines is the medium alcohol levels so I am not surprised that alcohol concentration was so important to the quality classification.

**Limitations**

The biggest change I made to the dataset was to group the response variable into two categories from the original seven. This limited the conclusions I could make but it did make interpretation easier. My grouping was somewhat arbitrarily made. Perhaps it would have been more meaningful to group the data into poor, mediocre, and good wines rather than just poor and good. The different quality ratings were unbalanced with very few samples from the extremes. With my binary grouping I believe downsampling would have been a good technique to employ prior to using the classification methods I chose to employ here. If I had preserved the original categories then downsampling would not be a viable option since the smallest category only had five samples.

For this dataset, expert tasters were allowed to assign one of eleven values (0-10). This rating scale presents some concerns, specifically did the rankings have the same meaning to each expert taster? Was the median the most representative of these rankings? Perhaps it would have been more helpful to have each taster’s ratings separate from the others. The quality score reported in the dataset came from at least three different tasters. It could be possible that having different numbers of tasters for different scores had an impact on the final quality score perhaps causing the large representation of wines ranked 5 and 6. I also wonder if the difference between the rankings had the same meaning; for example, is the difference in quality between 8 and 9 the same as the difference in quality between 5 and 6. I made the split between 5 and 6 for good and poor wines, but is a ranking of 5 actually poor? Is my split reflective of where experts would make the split between good and poor?

All the analytical methods were impacted by my initial choice to get rid of density based on its high correlation with other features. I think it would have been better to perform LASSO on the dataset as a feature selection process. The results from the nonlinear classification methods could have been different if I had run the entire dataset, including density. This could have possibly reduced the test classification error further. My results of the important variables did not vary much from the original researchers, but it might have if I had included density.

For the SVM method, I did not have the computing power to test different values of both tuning parameters for the polynomial and radial kernels, cost and gamma. With more computing power I may have found a different optimal combination. I also limited myself to just two degrees for the polynomial kernel. Potentially a higher degree would have produced better results.

The data itself limited the exploration possible. I believe other factors, like carbon dioxide, would possibly have been as important as alcohol as one of the hallmarks of Vinho Verde wines is the carbonation. There were many variables left out of the original dataset due to missing values in some of the samples, but I wonder if it would have been more valuable to include the features and reduce the sample size.

**Future Directions**

I believe the future will bring greater concern on minimizing impact to clean water resources so future datasets should include the resources used such as water, fertilizer, etc. in the hope that quality can be maximized while minimizing the environmental damage. In terms of other aspects of the dataset itself, the general ranking of price for consumer, idea of quality from marketing, and consumer ratings should be included if possible. These features would add meaningful dimensions to what could be learned from this data.

Expert tastes are potentially different from the consumer tastes. It would be interesting to use consumer purchase habits/feedback to classify wines as good/bad (or any number of k-classes). Do expert tastes relate to/predict purchase numbers, price? For this type of learning to be more meaningful, ability to predict consumer tastes would be important. Although there is a certain group of consumers who will listen to expert tastes/opinions, there will always be those who follow their own tastes.

This wine set is somewhat outdated at this point (being from wines more than a decade ago). The physiochemical tests are still standard for assessing the physical quality of the wine, but perhaps expert wine tastes have changed. Would the models built with this dataset be applicable to more current Vinho Verde wines from the Minho region? I also wonder if the models built here could be applied to wines from other countries/regions and other varieties of white wines. A new data collection would bring the opportunity to utilize Vilela et al.’s standardized sensory lexicon to address the subjectivity of taste tester ratings. This could shed light and bring more meaning to the quality assessment of the wines as it removes some of the subjectivity of the quality measure.

For further analysis of this dataset, I believe it would be meaningful to compare only the most extreme classes by using upsampling to balance the classes as well as to increase the sample size from five to twenty. It may be better to look at the quality in terms of three groups with downsampling. I could have grouped the wines such that the good ones were ratings of 7-9, the mediocre wines as 5-6, and poor wines as 4 and lower. Grouping them this way would produce very unbalanced groups which would require up or downsampling. I could also apply a cost to the larger groups. It would also be interesting to apply a neural network and compare the results to the other methods as it seems as though the non-linear approaches worked the most reliably.

**References**

* P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.
* A. Brochado and F. Oliveira. Brand equity in the Portuguese vinho verde “green wine” market. International Journal of Wine Business Research, 30(1):2-18, 2018.
* B. Neto, A. C. Dias, and M. Machado. Life cycle assessment of the supply chain of a Portuguese wine: from viticulture to distribution. International Journal of Life Cycle Assessment, 18:590-602, 2013.
* P. Quinteiro, A. C. Dias, L. Pina, B. Neto, B. G. Ridoutt, and L. Arroja. Addressing the freshwater use of a Portuguese wine (‘vinho verde’) using different LCA methods. Journal of Cleaner Production, Elsevier, 68:46-55, 2014.
* A. Vilela, C. Marquez, and E. Correia. Structural equation modelling (SEM) applied to sensory profile of Vinho Verde monovarietal wines. Food Research International, Elsevier, 111:650-660, 2018.
* A. Brochado and F. Oliveira. Brand equity in the Portuguese vinho verde “green wine” market. International Journal of Wine Business Research, 30(1):2-18, 2018.
* The Viticulture Commission of the Vinho Verde Region (2022) <https://www.cipvv.pt/en/vinho-verde/certification-of-vinho-verde-wines/>

**R Code Appendix**

library(caret)

library(Hmisc)

library(DMwR2)

library(corrplot)

library(e1071)

library(leaps)

library("FSelector")

library(BiocGenerics)

library(MASS)

library(class)

library(ggplot2)

library(randomForest)

library(tree)

library(olsrr)

library(FSelector)

library(klaR)

white\_wine <- read.csv("~/Grad School/Fall 22/M 748/Project/winequality-white.csv", sep=";")

dim(white\_wine)

names(white\_wine)

head(white\_wine)

hist(white\_wine$quality, xlab="Quality", main="Histogram of White Wine Quality Response")

summary(white\_wine)

summary(factor(white\_wine$quality)) # number of obs in each rating class

#NA's

sum(is.na(white\_wine))

#Variable Ranges: Table, pre-standardized

var\_mins <- c(min(white\_wine$fixed.acidity), min(white\_wine$volatile.acidity), min(white\_wine$citric.acid), min(white\_wine$residual.sugar), min(white\_wine$chlorides),min(white\_wine$free.sulfur.dioxide), min(white\_wine$total.sulfur.dioxide), min(white\_wine$density), min(white\_wine$pH), min(white\_wine$sulphates), min(white\_wine$alcohol))

var\_maxs <- c(max(white\_wine$fixed.acidity), max(white\_wine$volatile.acidity), max(white\_wine$citric.acid), max(white\_wine$residual.sugar), max(white\_wine$chlorides), max(white\_wine$free.sulfur.dioxide), max(white\_wine$total.sulfur.dioxide), max(white\_wine$density), max(white\_wine$pH), max(white\_wine$sulphates), max(white\_wine$alcohol))

var\_info <- rbind(var\_mins, var\_maxs)

colnames(var\_info)=c("Fixed Acidity", "Volatile Acidity", "Citric Acid", "Residual Sugar", "Chlorides", "Free SO2", "Total SO2", "Density", "pH", "Sulphates", "Alcohol")

rownames(var\_info) = c("Min", "Max")

var\_info

#Recode Outcome, Quality, and split into a vector. Histogram

qual\_bi <- ifelse(white\_wine$quality <6, 1,0)

hist(qual\_bi, xlab="Quality", main="Histogram of White Wine Quality Response")

summary(factor(qual\_bi)) #of obs in each group

#Histogram of pre-scaled

par(mfrow=c(3,4))

hist(white\_wine[,1:11])

par(mfrow=c(1,1))

#Near-Zero Variance?

nearZeroVar(white\_wine[,1:11]) #None are near-zero, will not drop any at this stage

#Test of Normality:

lap\_test <- lapply(white\_wine, shapiro.test)

norm\_res <- numeric(11)

for(i in 1:11){

norm\_res[i] = lap\_test[i]

}

norm\_res #All fail the test of normality

#Check Skew

skew\_check <- lapply(white\_wine, skewness)

skew\_res <- numeric(11)

for(j in 1:11){

skew\_res[j] = skew\_check[j]

}

skew\_res #Var 2,3,4,5,6 skewed

#Scale predictors:

wine\_scale <- scale(white\_wine[,1:11], center=TRUE, scale=TRUE)

wine\_new <- data.frame(wine\_scale, qual\_bi)

#Histograms of predictors after scaling

par(mfrow=c(3,4))

hist(wine\_new[,1:11])

par(mfrow=c(1,1))

#Variable Ranges:Table, post-standardized

var\_scale\_min <- c(min(wine\_new$fixed.acidity), min(wine\_new$volatile.acidity), min(wine\_new$citric.acid), min(wine\_new$residual.sugar), min(wine\_new$chlorides), min(wine\_new$free.sulfur.dioxide), min(wine\_new$total.sulfur.dioxide), min(wine\_new$density), min(wine\_new$pH), min(wine\_new$sulphates), min(wine\_new$alcohol))

var\_scale\_max <- c(max(wine\_new$fixed.acidity), max(wine\_new$volatile.acidity), max(wine\_new$citric.acid), max(wine\_new$residual.sugar), max(wine\_new$chlorides), max(wine\_new$free.sulfur.dioxide), max(wine\_new$total.sulfur.dioxide), max(wine\_new$density), max(wine\_new$pH), max(wine\_new$sulphates), max(wine\_new$alcohol))

var\_scale\_info <-rbind(var\_scale\_min, var\_scale\_max)

colnames(var\_scale\_info)=c("Fixed Acidity", "Volatile Acidity", "Citric Acid", "Residual Sugar", "Chlorides", "Free SO2", "Total SO2", "Density", "pH", "Sulphates", "Alcohol")

rownames(var\_scale\_info) = c("Min", "Max")

var\_scale\_info

#Variable Scaling

wine\_new$volatile.acidity <- scale(wine\_new$volatile.acidity)

wine\_new$residual.sugar <- scale(wine\_new$residual.sugar)

wine\_new$chlorides <- scale(wine\_new$chlorides)

wine\_new$free.sulfur.dioxide <-scale(wine\_new$free.sulfur.dioxide)

wine\_new$citric.acid <- scale(wine\_new$citric.acid)

#Check histograms again

hist(wine\_new$residual.sugar, main="Histogram of Standardized Residual Sugar", xlab = "Standardized Residual Sugar")

cors <- cor(wine\_new[,1:11])

corrplot(cors)

which(abs(cors)>0.5)

cors[which(abs(cors)>0.7)]

#drop density

names(wine\_new) #to find index number of density

wine\_new <- wine\_new[,-8]

names(wine\_new) #to ensure density was dropped. Now only 10 predictors

#black=0, red=1

plot(wine\_new$fixed.acidity, wine\_new$sulphates, col=factor(wine\_new$qual\_bi))

plot(wine\_new$volatile.acidity, wine\_new$free.sulfur.dioxide, col=factor(wine\_new$qual\_bi))

par(mfrow=c(1, 3))

plot(wine\_new$residual.sugar, wine\_new$alcohol, col=factor(wine\_new$qual\_bi), xlim=c(0,5), main="Quality of Wine", xlab="Residual Sugar", ylab="Alcohol")

plot(wine\_new$fixed.acidity, wine\_new$pH, col=factor(wine\_new$qual\_bi), main="Quality of Wine", xlab="Fixed Acidity", ylab="pH")

plot(wine\_new$citric.acid, wine\_new$chlorides, col=factor(wine\_new$qual\_bi), main="Quality of Wine", xlab="Citric Acid", ylab="Chlorides")

par(mfrow=c(1,1))

set.seed(748)

train <- sample(1:nrow(wine\_new),0.8\*nrow(wine\_new))

wine\_train <- wine\_new[train,]

wine\_test <- wine\_new[-train,]

summary(factor(wine\_train$qual\_bi))

summary(factor(wine\_test$qual\_bi))

chart <- rbind(summary(factor(wine\_train$qual\_bi)), summary(factor(wine\_test$qual\_bi)))

rownames(chart) = c("Training Set", "Testing Set")

colnames(chart) = c("Good", "Poor")

chart

regfit.full <- regsubsets(qual\_bi~.,data=wine\_train,nvmax=10)

reg.summary <- summary(regfit.full)

reg.summary$outmat

plot(reg.summary$adjr2, ylab = "Adjusted R^2", main="Best Subset Selection")

#the top 2 predictors are alcohol and volatile acidity

plot(wine\_train$alcohol, wine\_train$volatile.acidity, col=factor(wine\_train$qual\_bi))

#this plot shows that these two predictors offer fairly decent separation b/n good and poor quality classification

wine\_testy <- wine\_test$qual\_bi

test.mat <- model.matrix(qual\_bi~.,data=wine\_test)

#Use test set/CV to find best model:

set.seed(748)

val.errors <- rep(NA,10)

for(i in 1:10){

coefi <- coef(regfit.full,id=i)

pred <- test.mat[,names(coefi)]%\*%coefi

pred.y <- ifelse(pred>=0.5, 1,0)

val.errors[i] <- sum(pred.y != wine\_testy)

}

val.errors

which.min(val.errors)

plot(val.errors, main="Best Subset Selection", ylab = "Cross Validation Errors")

coef(regfit.full, 5)

val.errors[5]-val.errors[2] #25 fewer errors for the 5-factor model

25/980 #only 2.55% improvement in test classification error

val.errors[5]/980

coef(regfit.full, 2)

plot(wine\_new$volatile.acidity, wine\_new$alcohol, col=factor(wine\_new$qual\_bi), ylab="Alcohol", xlab="Volatile Acidity", main="Classification of Wine Quality")

#First Naive Bayes

nB.fit <- naiveBayes(qual\_bi ~ alcohol + volatile.acidity, wine\_train)

nB.trainpred <- predict(nB.fit, wine\_train)

nB.testpred <- predict(nB.fit, wine\_test)

tr.table <- table(nB.trainpred, wine\_train$qual\_bi)

te.table <- table(nB.testpred, wine\_test$qual\_bi)

nB.trerr <- (742+248)/3918

nB.teerr <- (190+66)/980

#2nd Logistic Regression

glm.fit <- glm(qual\_bi~alcohol+volatile.acidity, wine\_train, family="binomial")

glm.probs <- predict(glm.fit, type="response")

glm.pred <- rep(0,3918)

glm.pred[glm.probs>.5]=1

glm.tr.tab <- table(glm.pred,wine\_train$qual\_bi)

trainglm.err <- mean(glm.pred != wine\_train$qual\_bi)

glm.testprobs <- predict(glm.fit, wine\_test, type="response")

glm.testpred <- rep(0, 980)

glm.testpred[glm.testprobs>0.5]=1

glm.te.tab <- table(glm.testpred, wine\_test$qual\_bi)

testglm.err <- mean(glm.testpred != wine\_test$qual\_bi)

#3rd Linear Regression

lm.fit <- lm(qual\_bi~alcohol+volatile.acidity, wine\_train)

pred.lmtrain <- predict(lm.fit, wine\_train)

pred.lmtrain <- ifelse(pred.lmtrain <0.5, 0, 1)

lm.tr.tab <- table(pred.lmtrain, wine\_train$qual\_bi)

pred.lmtest <- predict(lm.fit, wine\_test)

pred.lmtest <- ifelse(pred.lmtest <0.5, 0, 1)

lm.te.tab <- table(pred.lmtest, wine\_test$qual\_bi)

lm.trerr <- (684+334)/3918

lm.teerr <- (180+81)/980

#4th LDA

lda.fit <- lda(qual\_bi~alcohol+volatile.acidity, wine\_train)

ldatrain.pred <- predict(lda.fit, wine\_train)

ldatrain.predqual <- ldatrain.pred$class

lda.tr.tab <- table(ldatrain.predqual, wine\_train$qual\_bi)

ldatest.pred <- predict(lda.fit, wine\_test)

ldatest.predqual <- ldatest.pred$class

lda.te.tab <- table(ldatest.predqual, wine\_test$qual\_bi)

lda.trerr <- (653+369)/3918

lda.teerr <- (171+91)/980

results <- rbind(c(nB.trerr,trainglm.err,lm.trerr,lda.trerr ), c(nB.teerr,testglm.err,lm.teerr,lda.teerr))

colnames(results) <- c("Naive Bayes", "Logistic Reg.", "Linear Reg", "LDA")

rownames(results) <- c("Training Error", "Test Error")

results

#Show 2 decision boundaries:

plot(wine\_train$volatile.acidity, wine\_train$alcohol, col=factor(wine\_train$qual\_bi), main="Comparison of Linear Classifiers", ylab="Alcohol", xlab="Volatile Acidity")

abline(a=0.00227/0.1908, b=0.1181/0.1908,col="green")

abline(a=-0.189053/1.0844, b=0.63168/1.0844, col="purple", lty=2)

legend("topright", legend=c("Linear", "Logistic"), col=c("Green", "Purple"), lty=1:2)

## FINISH THE GRAPH OF DECISION BOUNDARIES!

plot(wine\_train$volatile.acidity, wine\_train$alcohol, col=factor(wine\_train$qual\_bi))

abline(a=0, b=sqrt(1/log(0.6638591/0.3361409))) #Bayes?

abline(a=sqrt(log(0.6638591/0.3361409)), b=1) #Bayes?

abline(a=-0.1644/0.1908, b=0.1181/0.1908, col="chartreuse3", lty=3)

abline(a=-0.8822/1.0844, b=0.63168/1.0844, col="darkorchid", lty=2)

abline(a=-0.189053/1.0844, b=0.63168/1.0844) #Logistic

abline(a=0.00227/1.0844,0.1181/0.1908) #Linear

legend("topright", legend=c("Bayes", "Logistic", "Linear", "LDA"), col=c("black", "Purple", "Green", "Blue"), lty=1:4)

#First Naive Bayes

nB.fit5 <- naiveBayes(qual\_bi ~ alcohol + volatile.acidity + residual.sugar + fixed.acidity + sulphates, wine\_train)

nB.trainpred5 <- predict(nB.fit5, wine\_train)

nB.testpred5 <- predict(nB.fit5, wine\_test)

tr.table5 <- table(nB.trainpred5, wine\_train$qual\_bi)

te.table5 <- table(nB.testpred5, wine\_test$qual\_bi)

nB.trerr5 <- (652+478)/3918

nB.teerr5 <- (171+124)/980

#2nd Logistic Regression

glm.fit5 <- glm(qual\_bi ~ alcohol + volatile.acidity + residual.sugar + fixed.acidity + sulphates, wine\_train, family="binomial")

glm.probs5 <- predict(glm.fit5, type="response")

glm.pred5 <- rep(0,3918)

glm.pred5[glm.probs5>.5]=1

glm.tr.tab5 <- table(glm.pred5,wine\_train$qual\_bi)

trainglm.err5 <- mean(glm.pred5 != wine\_train$qual\_bi)

glm.testprobs5 <- predict(glm.fit5, wine\_test, type="response")

glm.testpred5 <- rep(0, 980)

glm.testpred5[glm.testprobs5>0.5]=1

glm.te.tab5 <- table(glm.testpred5, wine\_test$qual\_bi)

testglm.err5 <- mean(glm.testpred5 != wine\_test$qual\_bi)

#3rd Linear Regression

lm.fit5 <- lm(qual\_bi ~ alcohol + volatile.acidity + residual.sugar + fixed.acidity + sulphates, wine\_train)

pred.lmtrain5 <- predict(lm.fit5, wine\_train)

pred.lmtrain5 <- ifelse(pred.lmtrain5 <0.5, 0, 1)

lm.tr.tab5 <- table(pred.lmtrain5, wine\_train$qual\_bi)

pred.lmtest5 <- predict(lm.fit5, wine\_test)

pred.lmtest5 <- ifelse(pred.lmtest5 <0.5, 0, 1)

lm.te.tab5 <- table(pred.lmtest5, wine\_test$qual\_bi)

lm.trerr5 <- (687+283)/3918

lm.teerr5 <- (170+66)/980

#4th LDA

lda.fit5 <- lda(qual\_bi ~ alcohol + volatile.acidity + residual.sugar + fixed.acidity + sulphates, wine\_train)

ldatrain.pred5 <- predict(lda.fit5, wine\_train)

ldatrain.predqual5 <- ldatrain.pred5$class

lda.tr.tab5 <- table(ldatrain.predqual5, wine\_train$qual\_bi)

ldatest.pred5 <- predict(lda.fit5, wine\_test)

ldatest.predqual5 <- ldatest.pred5$class

lda.te.tab5 <- table(ldatest.predqual5, wine\_test$qual\_bi)

lda.trerr5 <- (658+311)/3918

lda.teerr5 <- (160+70)/980

results5 <- rbind(c(nB.trerr,trainglm.err,lm.trerr,lda.trerr ), c(nB.teerr5,testglm.err5,lm.teerr5,lda.teerr5))

colnames(results5) <- c("Naive Bayes", "Logistic Reg.", "Linear Reg", "LDA")

rownames(results5) <- c("Test Error - 2", "Test Error - 5")

results5

set.seed(748)

train.qual\_bi <- wine\_train$qual\_bi

train.X <- wine\_train[,1:10]

test.qual\_bi <- wine\_test$qual\_bi

test.X <- wine\_test[,1:10]

k.seq2 <- seq(from=1, to=20, by=1)

knntest.errs2 <- numeric(length(k.seq2))

knntr.errs2 <- numeric(length(k.seq2))

for(i in 1:length(k.seq2)){

knn.pred <- knn(train.X, test.X, train.qual\_bi, k=i)

knntest.errs2[i] <- mean(knn.pred != test.qual\_bi)

knn.predtr <- knn(train.X, train.X, train.qual\_bi, k=i)

knntr.errs2[i] <- mean(knn.predtr != train.qual\_bi)

}

knntest.errs2

k.seq3 <- seq(from=1, to=500, by=2)

knntest.errs3 <- numeric(length(k.seq3))

knntr.errs3 <- numeric(length(k.seq3))

for(i in 1:length(k.seq3)){

knn.pred <- knn(train.X, test.X, train.qual\_bi, k=i)

knntest.errs3[i] <- mean(knn.pred != test.qual\_bi)

knn.predtr <- knn(train.X, train.X, train.qual\_bi, k=i)

knntr.errs3[i] <- mean(knn.predtr != train.qual\_bi)

}

plot(k.seq3, knntest.errs3)

results\_knn <- rbind(c(knntr.errs[1], knntr.errs[2], knntr.errs[3], knntr.errs[50], knntr.errs[100]),c(knntest.errs[1], knntest.errs[10], knntest.errs[25],knntest.errs[50], knntest.errs[100]))

colnames(results\_knn) <- c("1-NN", "10-NN", "25-NN", "50-NN", "100-NN")

rownames(results\_knn) <- c("Training Error", "Test Error")

results\_knn

knn.results <- rbind(c(knntr.errs[1], knntr.errs[2], knntr.errs[3], knntr.errs[4], knntr.errs[5], knntr.errs[6], knntr.errs[7], knntr.errs[8], knntr.errs[9], knntr.errs[10]), c(knntest.errs[1], knntest.errs[2], knntest.errs[3], knntest.errs[4], knntest.errs[5], knntest.errs[6], knntest.errs[7], knntest.errs[8], knntest.errs[9], knntest.errs[10]))

colnames(knn.results) <- c("1-NN", "2-NN", "3-NN", "4-NN", "5-NN", "6-NN", "7-NN", "8-NN", "9-NN", "10-NN")

rownames(knn.results) <- c("Training Error", "Test Error")

knn.results

plot(k.seq3, knntest.errs3, main="K-NN Classification of Wine Quality", xlab="K", ylab="Test Classification Error")

plot(k.seq2, knntest.errs2, main="K-NN Classification of Wine Quality", xlab="K", ylab="Test Classification Error")

test\_results <- c(knntest.errs[1], knntest.errs[8], knntest.errs[9])

set.seed(748)

train.x <- wine\_train[,1:10]

test.x <- wine\_test[,1:10]

train.y <- factor(train.qual\_bi)

test.y <- factor(test.qual\_bi)

tune.out=tune(svm,factor(qual\_bi)~.,data=wine\_train,kernel="linear",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))

summary(tune.out)#error: CV error rate

bestmod=tune.out$best.model

summary(bestmod) #0.1 is best

svmfit=svm(factor(qual\_bi)~., data=wine\_train, kernel="linear", cost=.1,scale=FALSE)

ypred=predict(svmfit,test.X)

table(predict=ypred, truth=test.y)

lin.test.err <- mean(ypred != test.y)

set.seed(748)

tune.out2=tune(svm,factor(qual\_bi)~.,data=wine\_train,kernel="polynomial",ranges=list(cost=c(0.01, 0.1, 1,5,10),gamma=0.1))

summary(tune.out2)#error: CV error rate

bestmod2=tune.out2$best.model

summary(bestmod2) #1 is best

svmfit2=svm(factor(qual\_bi)~., data=wine\_train, kernel="polynomial", cost=1, gamma=0.1, scale=FALSE)

ypred2=predict(svmfit2,test.X)

table(predict=ypred2, truth=test.y)

poly1.test.err <- mean(ypred2 != test.y)

tune.out3=tune(svm,factor(qual\_bi)~.,data=wine\_train,kernel="polynomial", degree=2, ranges=list(cost=c(0.01, 0.1, 1,5,10),gamma=0.1))

summary(tune.out3)#error: CV error rate

bestmod3=tune.out3$best.model

summary(bestmod3)

svmfit3=svm(factor(qual\_bi)~., data=wine\_train, kernel="polynomial", degree=2, cost=1,scale=FALSE)

ypred3=predict(svmfit3,test.X)

table(predict=ypred3, truth=test.y)

poly2.test.err <- mean(ypred3 != test.y)

set.seed(748)

tune.outr=tune(svm, factor(qual\_bi)~., data=wine\_train, kernel="radial", ranges = list(cost=c(0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4)))

summary(tune.outr)

summary(tune.outr$best.model)

table(true=test.y, pred = predict(tune.outr$best.model,newdata=test.X))

rad.test.err <- mean(predict(tune.outr$best.model,newdata=test.X) != test.y)

svm\_results <- rbind(c(lin.test.err, poly2.test.err, poly1.test.err, rad.test.err), c(2298,2511, 2234, 2599), c(0.1,1,1,1))

rownames(svm\_results) <- c("Test Error", "Support Vectors", "Cost")

colnames(svm\_results) <- c("Linear", "Polynomial 2", "Polynomial 3", "Radial")

svm\_results

set.seed(748)

wine.tree <- tree(factor(qual\_bi)~., wine\_train, split="gini")

summary(wine.tree)

plot(wine.tree)

text(wine.tree , pretty = 0)

tree.pred <- predict(wine.tree, wine\_test, type="class")

table(tree.pred, wine\_test$qual\_bi)

tree.err <- (128+110)/980

cv.wine <- cv.tree(wine.tree, FUN=prune.misclass)

min(cv.wine$dev)

par(mfrow=c(1,1))

plot(cv.wine$size, cv.wine$dev, type="b")

plot(cv.wine$k, cv.wine$dev, type="b")

prune.wine <- prune.misclass(wine.tree, best = 14)

plot(prune.wine)

text(prune.wine , pretty = 0)

prune.pred <- predict(prune.wine, wine\_test, type="class")

table(prune.pred, wine\_test$qual\_bi)

prune.err <- (189+70)/980

best\_seq <- seq(from=2, to=150, by=2)

tree.err <- numeric(length(best\_seq))

for(i in 1:length(best\_seq)){

prune.wine <- prune.misclass(wine.tree, best=best\_seq[i])

prune.pred <- predict(prune.wine, wine\_test, type="class")

tree.err[i] <- mean(prune.pred != wine\_test$qual\_bi)

}

plot(best\_seq, tree.err, main="Pruned Classification Trees", xlab="Number of Terminal Nodes", ylab="Test Classification Error",type="l")

trees <- seq(from=10, to=500, by=5)

rf.testerr = rep(NA, length(trees))

for(i in 1:99){

rf.p <- randomForest(train.x, y = train.y, xtest = test.x, ytest = test.y, mtry = ncol(wine\_train) - 1, ntree = trees[i])

rf.testerr[i] <- mean(rf.p$test$err.rate)

}

rf2.testerr = rep(NA, length(trees))

for(i in 1:99){

rf.p2 <- randomForest(train.x, y = train.y, xtest = test.x, ytest = test.y, mtry = (ncol(wine\_train) - 1) / 2, ntree = trees[i])

rf2.testerr[i] <- mean(rf.p2$test$err.rate)

}

rf3.testerr = rep(NA, length(trees))

for(i in 1:99){

rf.sp <- randomForest(train.x, y = train.y, xtest = test.x, ytest = test.y, mtry = sqrt(ncol(wine\_train) - 1), ntree = trees[i])

rf3.testerr[i] <- mean(rf.sp$test$err.rate)

}

plot(trees,rf.testerr, col = "green", type = "l", xlab = "Number of Trees", ylab = "Test Classification Error", main = "Random Forest Classification Error: Wine Quality" ,ylim = c(0.15, 0.25))

lines(trees, rf2.testerr, col = "red", type = "l")

lines(trees, rf3.testerr, col = "blue", type = "l")

legend("topright", c("m = p", "m = p/2", "m = sqrt(p)"), col = c("green", "red", "blue"), cex =1, lty = 1)

abline(h=0.177)

min(rf.testerr); min(rf2.testerr); min(rf3.testerr)

which.min(rf2.testerr) #to determine the index of the smallest error

trees[59] #use the index to get the number of trees for smallest error

#To build the tree for the smallest error and to find the most important vars.

rf.best <- randomForest(train.x, y = train.y, xtest = test.x, ytest = test.y, mtry = (ncol(wine\_train) - 1)/2, ntree = 300, importance=TRUE)

varImpPlot(rf.best, sort=TRUE, n.var = 5)

importance(rf.best)