Predicting Red Wine Quality

Melody C. Brown, MHSA, MPH, CSPO

3/9/2022

Introduction

The world of wine is driven not only by sales but also by prestige, with many wine makers changing their formulations and conditions incrementally in the attempt to achieve the highest possible ratings. Wines are certified based on a combination of physicochemical (include determination of density, alcohol, and ph values) and sensory tests (based on human experts; Cortez et al., 2009). Red wines in particular over-represent ratings over 90 points, with the best red wine ratings being between 95-100 (Siegel, 2017).

Wine makers typically follow a carefully controlled process that involves analytic testing along each step to ensure optimal conditions for fermentation. The most important parameters in the fermentation process that affect quality include the pH, acidity, sugar, sulfur dioxide, and alcohol levels. Additional health benefits of red wines can also be dependent on a complex process of many of these levels as well. Ultimately, careful control of these levels can make or break a wine in terms of ratings.

The purpose of this project is to examine a sample of red wines to determine which physicochemical attributes could be used to predict higher ratings and good or bad quality red wines. The data will first be explored and insights discussed. A regression model will be used first to determine which attributes can best predict ratings, and then a random forest model will be used to classify wines as either good or not good. The results and merits of each model will also be discussed for application within the wine industry to improve their processes.

Methods

The Wine Quality data is a publicly available data set that was donated to the UCI ML repository by Paulo Cortez, from here.Only the red wine data file was used for this analysis.

```
## Read in data file for red wine and indicate separator as ";"
winedb <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-
```

Variables Description

The data set includes ratings and attributes for 1,599 district red wine samples from the Minho region of Portugal, collected between May 2004 and February 2007. The data includes the following 12 physicochemical attributes (Cortez et al., 2009; Thermo Fisher Scientific, 2016):

- **Fixed acidity:** these include tartaric, malic, citric, and succinic acids; acidity levels affect the color, taste, and microbial stability of the wine.

- **Volatile Acidity:** typically indicative of spoiling and associated with higher vinegar tastes; flavors can often be detected at between 0.6-0.9 g/L; can be countered with higher sugar content.

- **Citric Acid:** one of the fixed acid types; ideally ranges between 0.5 and 1 g/L.

- **Residual Sugar:** this is the level of sugar left-over after the fermentation process; lower sugar levels are associated with drier wines.

- **Chlorides:** main contributors of perceived saltiness and affects the terroir; levels that are too high can contribute towards declining quality ratings.

- **Free Sulfur Dioxide:** important for protection from oxidation and spoilage; too much can contribute to bitter or metallic flavors.

- **Total Sulfur Dioxide:** includes free SO2 and bound SO2; important to keep the wine microbially stable
- **Density:** determined by the concentration of alcohol, sugar, and dissolved solids; indicates how well the fermentation process goes; ideally ranges between 1.08 and 1.09.
- **pH:** low levels of pH indicate higher acidity while low levels indicate higher alkali; wines should ideally range between 2.9 and 3.9 on pH levels
- **Sulphites:* produced through fermentation and protects against oxidation or bacterial growth
- **Alcohol:** occurs as a result of the fermentation process; ideally range between 12% and 15% for red wines;
- **Quality:** sensory; median blind taste score across a minimum of 3 human assessors; scale was from 0 (very bad) to 10 (excellent). This column will be later renamed as "Rating" to be more clear.

```
#Glimpse of data set
glimpse(winedb)
```

```
## Rows: 1,599
## Columns: 12
## $ fixed.acidity        <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3, 7.8, 7.5~
## $ volatile.acidity     <dbl> 0.700, 0.880, 0.760, 0.280, 0.700, 0.660, 0.600, ~
## $ citric.acid          <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, 0.00, 0~
## $ residual.sugar       <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2.0, 6.1,~
## $ chlorides            <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.075, 0.069, ~
## $ free.sulfur.dioxide  <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 17, 15, 17, 16~
## $ total.sulfur.dioxide <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102, 65, 102,~
## $ density              <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0.9978, 0~
## $ pH                   <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, 3.39, 3~
## $ sulphates            <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, 0.47, 0~
## $ alcohol              <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, 10.0, 9.5, 10.~
## $ quality              <int> 5, 5, 5, 6, 5, 5, 5, 7, 7, 5, 5, 5, 5, 5, 5, 5, 7~
```

Data Exploration

To dive deeper into the data set, we can start by examining the range and skewness of each of the red wine attributes. We can see strong positive skewness for fixed acidity, residual sugars, sulfur dioxide levels, sulphites, and alcohol levels. Box plots give further visualization of the spread of these attributes. Visual inspection of these indicate that the wines in this data set tend to have lower levels of fixed acidity and sulfur dioxide/sulphites, which can lead to less microbially stable wines and more spoilage. The lower levels of residual sugars could also indicate generally drier wines, which would be consistent with red wines. The

alcohol levels are also lower than typical red wines, and could be due to an insufficient fermentation process across the wines.

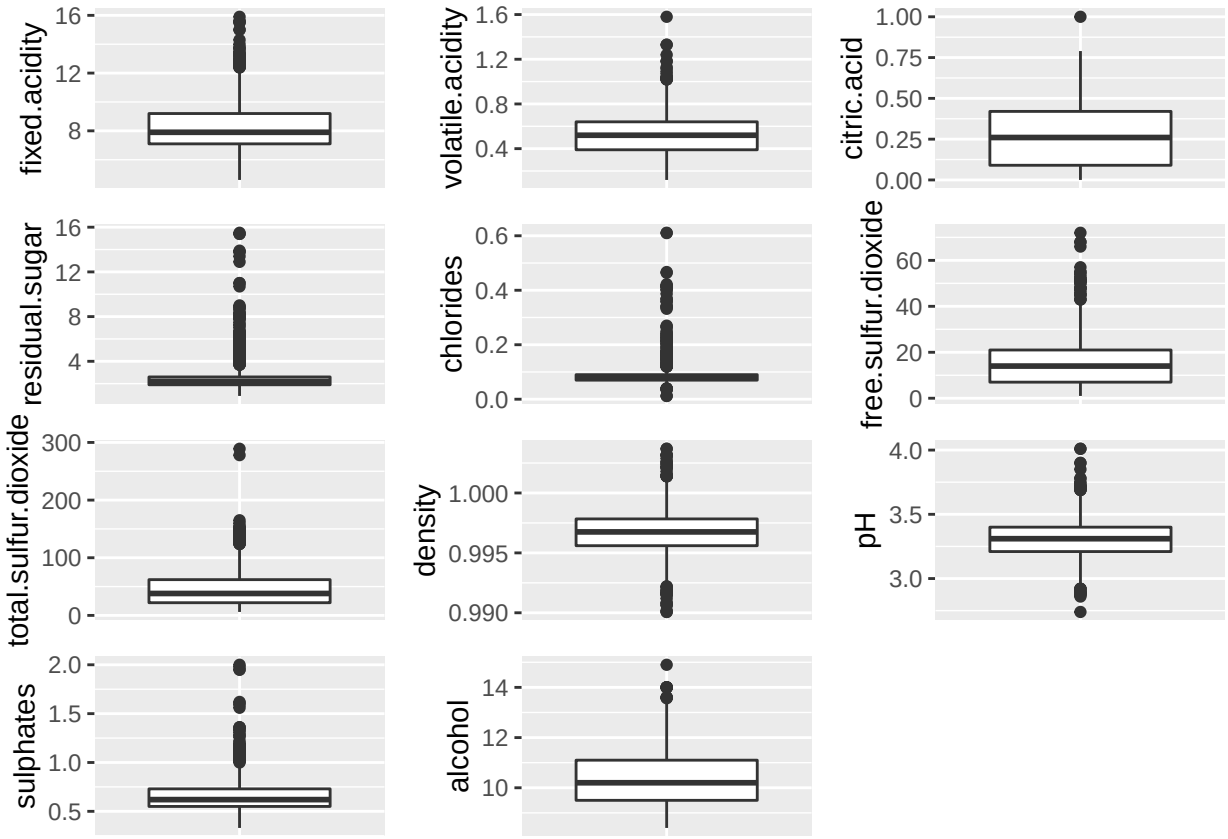
```
#Descriptives Summary Table
descriptives <- describe(winedb)

knitr::kable(descriptives, caption="Descriptives Summary Table")
```

Table 1: Descriptives Summary Table

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
fixed.acidity	1	1599	8.3196	1.7411	7.9000	8.1525	1.4826	4.6000	15.900	11.3000	0.9809	1.1197	0.0435
volatile.acidity	2	1599	0.5278	0.1791	0.5200	0.5181	0.1779	0.1200	1.580	1.4600	0.6703	1.2127	0.0045
citric.acid	3	1599	0.2710	0.1948	0.2600	0.2613	0.2520	0.0000	1.000	1.0000	0.3177	-0.7930	0.0049
residual.sugar	4	1599	2.5388	1.4099	2.2000	2.2584	0.4448	0.9000	15.500	14.6000	4.5321	28.4850	0.0353
chlorides	5	1599	0.0875	0.0471	0.0790	0.0802	0.0148	0.0120	0.611	0.5990	5.6697	41.5260	0.0012
free.sulfur.dioxide	6	1599	15.8749	10.4602	14.0000	14.5773	10.3782	1.0000	72.000	71.0000	1.2482	2.0072	0.2616
total.sulfur.dioxide	7	1599	46.4678	32.8953	38.0000	41.8431	26.6868	6.0000	289.000	283.0000	1.5127	3.7857	0.8226
density	8	1599	0.9967	0.0019	0.9968	0.9967	0.0017	0.9901	1.004	0.0136	0.0712	0.9225	0.0000
pH	9	1599	3.3111	0.1544	3.3100	3.3091	0.1483	2.7400	4.010	1.2700	0.1933	0.7959	0.0039
sulphates	10	1599	0.6581	0.1695	0.6200	0.6374	0.1186	0.3300	2.000	1.6700	2.4241	11.6615	0.0042
alcohol	11	1599	10.4230	1.0657	10.2000	10.3100	1.0378	8.4000	14.900	6.5000	0.8592	0.1917	0.0267
quality	12	1599	5.6360	0.8076	6.0000	5.5886	1.4826	3.0000	8.000	5.0000	0.2174	0.2879	0.0202

```
options(repre.plot.width=8, repr.plot.heigh =7)

#Visually explore attributes with boxplots
#Create function to plot each

FA <- ggplot(winedb, aes(x=factor(0), fixed.acidity)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

VA <- ggplot(winedb, aes(x=factor(0), volatile.acidity)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

CA <- ggplot(winedb, aes(x=factor(0), citric.acid)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

RS <- ggplot(winedb, aes(x=factor(0), residual.sugar)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

Chl <- ggplot(winedb, aes(x=factor(0), chlorides)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

FSO2 <- ggplot(winedb, aes(x=factor(0), free.sulfur.dioxide)) +
  geom_boxplot() +
```

```
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

TSO2 <- ggplot(winedb, aes(x=factor(0), total.sulfur.dioxide)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

Den <- ggplot(winedb, aes(x=factor(0), density)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

pH <- ggplot(winedb, aes(x=factor(0), pH)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

Sul <- ggplot(winedb, aes(x=factor(0), sulphates)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

Alc <- ggplot(winedb, aes(x=factor(0), alcohol)) +
  geom_boxplot() +
  theme(axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())

#Arrange boxplots to be displayed in a group
ggarrange(FA, VA, CA, RS, Chl, FSO2, TSO2, Den, pH, Sul, Alc, ncol= 3, nrow=4)
```



```
rm(FA, VA, CA, RS, Chl, FSO2, TSO2, Den, pH, Sul, Alc, descriptives)
```

When combining these insights with the distribution of ratings, it appears consistent with the majority of ratings falling in the 5-6 range. This indicates that the majority of wines were evaluated as okay wines, but not exceptionally great. A classification model to distinguish between good and not good wines could be useful for this level of wine quality. We can use this range as a breaking point for determining better quality wines over lower quality wines within the sample. These insights also indicate that the attributes with larger skewness could be important indicators of final wine quality ratings. Next, we can explore the data set by looking at which attributes will predict ratings through a linear regression model, and then we will split the ratings between good wines (ratings above 6) and not so good wines (ratings below 6) to maintain enough data points between both ends.

```
## Frequency plot of quality scores
qualitydist <-ggplot(winedb, aes(x=quality))+
  labs(title="Quality Frequency Distribution")+
  geom_bar(color="blue", fill="blue")
```

Procedure

Data and Model Preparation

To begin preparing the data for modeling, we will first rename some of the variables for easier visualization and rename "quality" to "Rating" to distinguish from our planned additional factor of good or not good quality. Then, we will normalize the scales for each attribute (excluding quality) using a Min-Max scaling method. Normalization is important when building algorithms with variables that are on widely different scales so that the model has a common scale to accurately assess the data without distorting differences in the ranges of values and allowing each attribute to contribute equally to the analysis. Next, we will also create an additional output variable of "Quality" that will classify the wines into either "good" or "not good" categories based off rating scores either above or below 6.

```
#Rename column names
winedb <- winedb %>%
  rename(
    FA = fixed.acidity,
    VA = volatile.acidity,
    CA = citric.acid,
    RS = residual.sugar,
    Chl = chlorides,
    FSO2 = free.sulfur.dioxide,
    TSO2 = total.sulfur.dioxide,
    Den = density,
    Sul = sulphates,
    Alc = alcohol,
    Rating = quality
  )

#Normalize the data and combine with quality column from original dataset
process <- preProcess(winedb[, c(1:11)], method= c("range"))
normwinedb <- predict(process, winedb[, c(1:11)])

#Add quality from original data set to normalized data set
normwinedb<- normwinedb %>% cbind(Rating = winedb$Rating)
```

```
#Add classification variable to indicate if wine is good or not good based on a rating of 6 or above
normwinedb <- normwinedb %>%
  mutate(Quality = as.factor(if_else(Rating >=6, "Good", "NotGood")))

#Confirm transformations to the data set
head(normwinedb)
```

```
##       FA     VA   CA      RS    Chl   FSO2    TSO2    Den     pH    Sul    Alc
## 1 0.2478 0.3973 0.00 0.06849 0.1068 0.1408 0.09894 0.5675 0.6063 0.1377 0.1538
## 2 0.2832 0.5205 0.00 0.11644 0.1436 0.3380 0.21555 0.4941 0.3622 0.2096 0.2154
## 3 0.2832 0.4384 0.04 0.09589 0.1336 0.1972 0.16961 0.5088 0.4094 0.1916 0.2154
## 4 0.5841 0.1096 0.56 0.06849 0.1052 0.2254 0.19081 0.5822 0.3307 0.1497 0.2154
## 5 0.2478 0.3973 0.00 0.06849 0.1068 0.1408 0.09894 0.5675 0.6063 0.1377 0.1538
## 6 0.2478 0.3699 0.00 0.06164 0.1052 0.1690 0.12014 0.5675 0.6063 0.1377 0.1538
##   Rating Quality
## 1      5 NotGood
## 2      5 NotGood
## 3      5 NotGood
## 4      6    Good
## 5      5 NotGood
## 6      5 NotGood
```

Partition Data Set

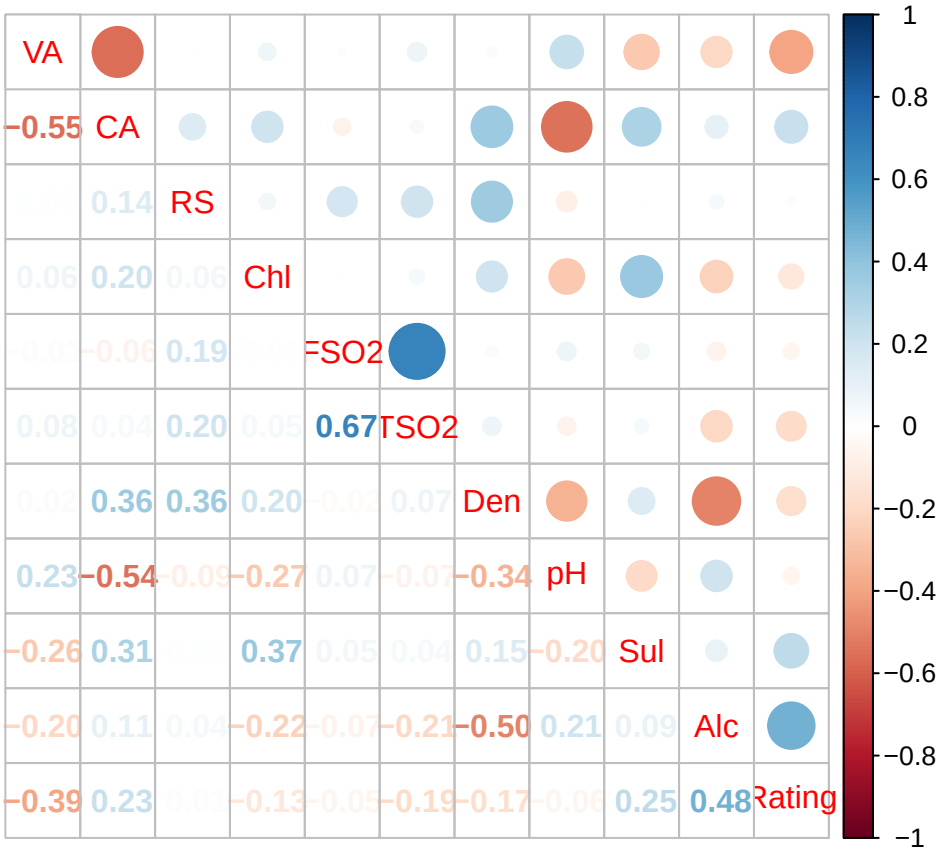
To begin testing our models, we will start by partitioning the data into test and train sets to help avoid over fitting of the models. The ideal split is to section off 10% of the total data set for validation to test the final model on, 10% for testing, and the remaining 80% for training. This works by training the model and testing it on one set of data for developing the model, but then testing it against new data to get a better sense of how well it will work in the future with new data.

```
### Partition Validation Set
#Set seed so that process is reproducible
set.seed (1, sample.kind ="Rounding")

#Set validation set as 10% of dataset
test_index <- createDataPartition(y = normwinedb$Rating, times = 1, p =0.1, list = FALSE)
redwinedb <- normwinedb[-test_index,]
validation <- normwinedb[test_index,]


### Partition Test Set
#Set seed so that process is reproducible
set.seed (1, sample.kind ="Rounding")

#Set test set as 10% of dataset
test_indexb <- createDataPartition (y = redwinedb$Rating, times = 1, p =0.1, list = FALSE)
train_set <- redwinedb[-test_indexb,]
test_set <-redwinedb[test_indexb,]


#Remove unneccessary objects for less memory usage
rm(test_index, test_indexb)
```

Model Development: Visualize Relationships Between Attributes

Before building the models, we will first visualize correlations between variables for better insight on parameter inclusion when building the models to avoid collinearity. We can achieve this by first creating a correlation matrix, where red indicates negative correlations and blue indicates positive correlations.

```
#Perform correlations on attributes in the data set
cors <- cor(normwinedb[,c(2:12)])

#Visualize with a correlation matrix
corrplot.mixed(cors)
```



Within the correlation matrix, there are several insights to note:

Fixed Acidity and Citric Acid Fixed acidity had strong-moderate positive correlations with both citric acid and density, but a strong-moderate negative correlation with pH. Citric acid is one of the fixed acids, so it would be logical for there to be a high correlation with that attribute, and higher acidity are defined by low pH levels and so are also logical. This may indicate that pH may be a better indicator than these attributes in our models.

Volatile Acidity Volatile acidity has a strong-moderate negative correlation with citric acid; weak negative correlations with fixed acidity, sulphites, and alcohol; and a weak positive correlation to pH. With the relationships between the other acidity attributes and pH already explained, that leaves the relationships to sulphites and alcohol left to consider. Since sulphites are meant to prevent spoilage and are a by product of the fermentation process, it would therefore make sense for volatile acidity (where higher levels indicate

spoilage) to have this weak inverse relationship. When also considering the correlations with the other acidity levels, this attribute could be considered in place of the fixed acidity or citric acid attributes.

Residual Sugars The only weak correlation for residual sugars appears to be density, with weak positive correlations with the sulphur dioxide attributes. This could be a good attribute to include in the models that will not interfere much with the others, but it will still need to be determined how well it can be used to predict ratings.

Chlorides and Sulphites Chlorides had a weak positive correlation with sulphites, and weak negative correlations with pH and Alcohol. Sulphites only had additional weak correlations with citric acid and volatile acid. Both attributes can therefore be considered for the models.

Free and Total Sulfur Dioxide These attributes only had positive correlations with each other, and so they should not be used together within the models.

Density and Alcohol Density had a strong-moderate correlation with fixed acidity but only a moderate-weak negative correlation with alcohol. There do not appear to be any logical explanations for these correlations, and so both attributes should still be considered within the model, with an eye on how combining density with fixed acidity may affect the model due to possible undetermined collinearty.

In considering the various relationships discovered, we will therefore take the following approach for the final models: * Try models that choose either fixed acidity or citric acid or volatile acidity, but not all * Include either free or total sulfur dioxide, but not both * Look closely at the effects of combining density and fixed acidity

Building the Multivariate Linear Regression Model

We will first attempt to use a multivariate linear regression model to predict ratings based on each of a combination of the attributes.

Multivariate linear regressions (MLRs) use data from a test set to determine the value of bias parameters for estimating the intercept and slope for each parameter in predicting the outcome variable, Y. It can be written as follows:

$$\hat{Y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ...\beta_n x_n$$

The regression model can be adjusted by examining the R-squared value, which is understood as describing the percent of the data that were fitted by the model, and by the RMSE. Performance will be determined by improvement in the residual mean squared error (RMSE) to measure the size of the error made in predicting the ratings. Since the ratings are integers between 1 and 5, we should hope for an RMSE of less than 1 for the model to be useful in future predictions. RMSE can be generally defined as:

$$RMSE = \sqrt{\frac{1}{N}\Sigma_{i=1}^{n}\Big(\hat{y}_i - y_i\Big)^2}$$

First MLR Model The first model we will try will be with all of the attributes to get a baseline of how well ratings can be predicted with all attributes included. Based on the summary of this model, we can see significant contributions to the model with higher ratings due to higher alcohol and sulphite levels and lower ratings with higher levels of volatile acidity, chlorides, and total sulfur dioxide. This seems consistent with what we expected based on exploratory analyses. The R-squared reported is only 0.362, but the nature of ratings can be considered very subjective in this area. It will be important to also look at the RMSE to determine the size of the errors in predictions.

```r
## First, we can use a baseline model using all attributes
fit1 <- lm(Rating~ FA + VA +CA +RS+Chl+FSO2+TSO2+Den+pH+Sul+Alc, data = train_set)

# Examine summary for additional insights
summary(fit1)
```

```
##
## Call:
## lm(formula = Rating ~ FA + VA + CA + RS + Chl + FSO2 + TSO2 +
##     Den + pH + Sul + Alc, data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6606 -0.3644 -0.0581  0.4518  2.0470
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.761      0.168   34.25  < 2e-16 ***
## FA             0.464      0.323    1.44  0.15139
## VA            -1.621      0.196   -8.26  3.7e-16 ***
## CA            -0.200      0.162   -1.23  0.21749
## RS             0.469      0.252    1.86  0.06279 .
## Chl           -1.179      0.285   -4.14  3.7e-05 ***
## FSO2           0.258      0.170    1.52  0.12978
## TSO2          -0.845      0.226   -3.74  0.00019 ***
## Den           -0.404      0.328   -1.23  0.21856
## pH            -0.508      0.271   -1.88  0.06092 .
## Sul            1.410      0.209    6.74  2.4e-11 ***
## Alc            1.747      0.193    9.04  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.65 on 1281 degrees of freedom
## Multiple R-squared:  0.367,  Adjusted R-squared:  0.362
## F-statistic: 67.6 on 11 and 1281 DF,  p-value: <2e-16
```

When using this model to predict ratings, it provides an RMSE of less than 1 with an error rate of 0.6572. This is less than 1, but we will next see if we can decrease the RMSE by adjusting variables based on our analysis to improve the accuracy of the predicted ratings.

```r
# Calculate predictions
y_hat1 <- predict(fit1, test_set)

#Create RMSE calculation function
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
}

#Calculate RMSE
MLR_M1 <- RMSE(y_hat1, test_set$Rating)

#Put into results table
RMSE_Results <- tibble(Model = "MLR_M1", RMSE = MLR_M1)
```

```
knitr::kable(RMSE_Results, caption = "MLR Model Results Table")
```

Table 2: MLR Model Results Table

Model	RMSE
MLR_M1	0.6572

Second MLR Model For the second MLR model, we will start by removing the attributes that were not significant: fixed acidity, citric acid, free sulfur dioxide, density, and residual sugar. This results in an unchanged R-squared of 0.36, but the t-value of alcohol increases quite a bit.

```
## First, we can use a baseline model using all attributes
fit2 <- update(fit1, .~. -FA -CA -FSO2 -Den -RS)

#Examine summary for additional insights
summary(fit2)
```

```
##
## Call:
## lm(formula = Rating ~ VA + Chl + TSO2 + pH + Sul + Alc, data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5779 -0.3626 -0.0528  0.4565  1.9762
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.754      0.106   54.29  < 2e-16 ***
## VA            -1.556      0.164   -9.51  < 2e-16 ***
## Chl           -1.258      0.271   -4.64  3.8e-06 ***
## TSO2          -0.620      0.157   -3.95  8.1e-05 ***
## pH            -0.659      0.166   -3.97  7.5e-05 ***
## Sul            1.340      0.202    6.63  4.8e-11 ***
## Alc            1.919      0.122   15.68  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.651 on 1286 degrees of freedom
## Multiple R-squared:  0.363,  Adjusted R-squared:  0.36
## F-statistic:  122 on 6 and 1286 DF,  p-value: <2e-16
```

When using this new updated model, we can find that even though the R-squared value did not improve, the RMSE does improve to 0.6553. The lack of change of the R-squared value could indicate that improvements in the model did not come at the cost of over fitting the data.

```
#Calculate predictions
y_hat2 <- predict(fit2, test_set)

#Calculate RMSE
MLR_M2 <- RMSE(y_hat2, test_set$Rating)
```

```r
#Put into results table
RMSE_Results <- tibble(Model = c("MLR_M1", "MLR_M2"), RMSE = c(MLR_M1, MLR_M2))

knitr::kable(RMSE_Results, caption = "MLR Model Results Table")
```

Table 3: MLR Model Results Table

Model	RMSE
MLR_M1	0.6572
MLR_M2	0.6553

Third MLR Model For the third MLR model, we will try to narrow down the attributes a little more to see if we can further improve the model. After removing TSO2, the R-squared decreases to 0.353. This model is therefore not fitting the data as well as the other models.

```r
## First, we can use a baseline model using all attributes
fit3 <- update(fit2, .~. -TSO2)

#Examine summary for additional insights
summary(fit3)
```

```
##
## Call:
## lm(formula = Rating ~ VA + Chl + pH + Sul + Alc, data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5748 -0.3714 -0.0631  0.4578  1.9960
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.641      0.103   54.95  < 2e-16 ***
## VA            -1.600      0.164   -9.75  < 2e-16 ***
## Chl           -1.220      0.272   -4.48  8.1e-06 ***
## pH            -0.622      0.166   -3.74  0.00019 ***
## Sul            1.283      0.203    6.33  3.4e-10 ***
## Alc            2.001      0.121   16.49  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.655 on 1287 degrees of freedom
## Multiple R-squared:  0.355,  Adjusted R-squared:  0.353
## F-statistic:  142 on 5 and 1287 DF,  p-value: <2e-16
```

When using this new version of the model, the RMSE also increases to 0.6599. It is still less than the entire model, but we achieved better results with the second model. We will therefore choose the second model to test against the validation set to compare the RMSE results.

```r
#Calculate predictions
y_hat3 <- predict(fit3, test_set)
```

```
#Calculate RMSE
MLR_M3 <- RMSE(y_hat3, test_set$Rating)

#Put into results table
RMSE_Results <- tibble(Model = c("MLR_M1", "MLR_M2", "MLR_M3"), RMSE = c(MLR_M1, MLR_M2, MLR_M3))

knitr::kable(RMSE_Results, caption = "MLR Model Results Table")
```

Table 4: MLR Model Results Table

Model	RMSE
MLR_M1	0.6572
MLR_M2	0.6553
MLR_M3	0.6599

```
#Remove testing objects for more memory
rm(fit1, fit2, fit3, RMSE_Results, y_hat1, y_hat2, y_hat3, MLR_M1, MLR_M2, MLR_M3)
```

Building the Random Forest Model

Random forest models are another type of supervised learning algorithm frequently used in machine learning, and it is ideal for classification problems, especially when they involve many attributes. This type of model creates many different decision trees and then averages the prediction results across them using bootstrapping to induces randomness. Prediction accuracy is also improved by the reduction of correlation between these decision trees, which can be particularly useful with multiple features.

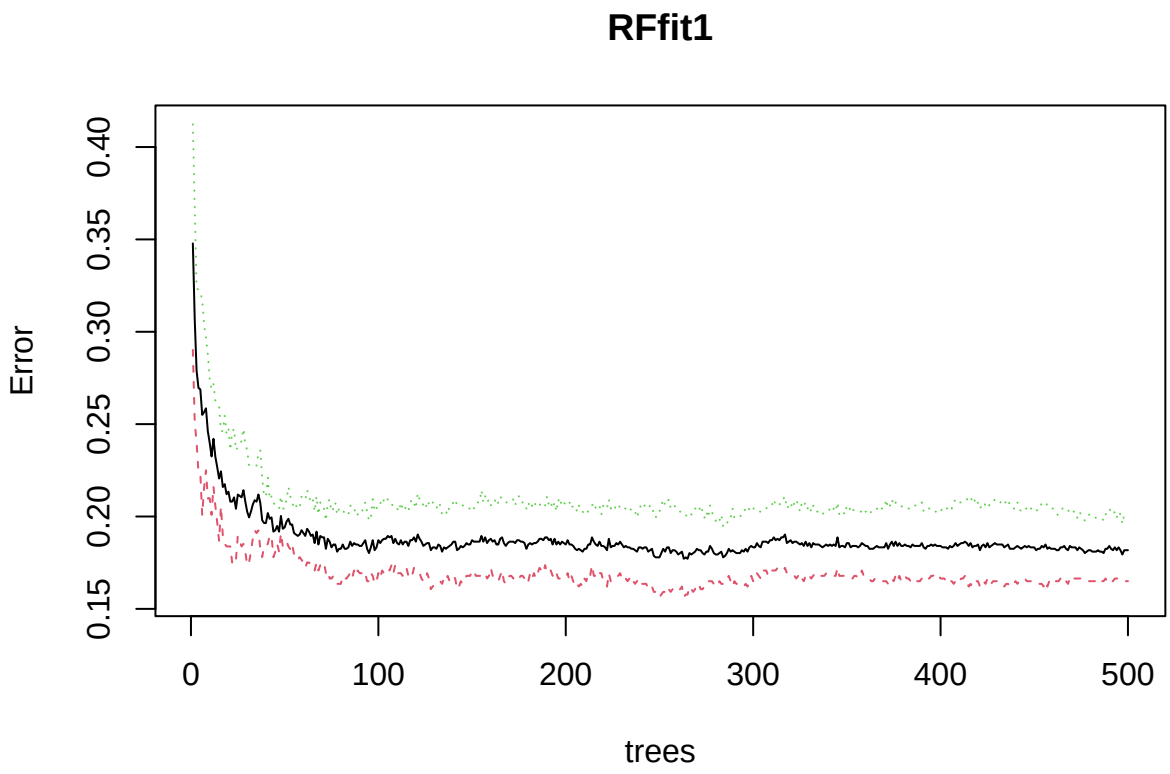
Performance of the random forest model will be based on accuracy in predicting if a wine was of good or not good quality. Accuracy is defined as the overall proportion that is predicted correctly, and can be ascertained using a confusion matrix, which tabulates each combination of prediction and actual values.

First Random Forest Model To begin, we will first build a random forest model without any customization to get a baseline of accuracy. In this first model, we find an accuracy of 76.55% with the error rate dropping substantially with 500 trees; however, we can make further adjustments to the model to help improve accuracy.

```
#Random Forest for all attributes except for Rating
RFfit1 <- randomForest(Quality~ . - Rating, data=train_set)

#Calculate prediction
RFpredict1 <- predict(RFfit1, test_set)

#Create confusion matrix to determine accuracy
confusionMatrix(RFpredict1, test_set$Quality)$overall[["Accuracy"]]
```

```
## [1] 0.7655
```

```
#Plot error rate of number of trees
plot(RFfit1)
```

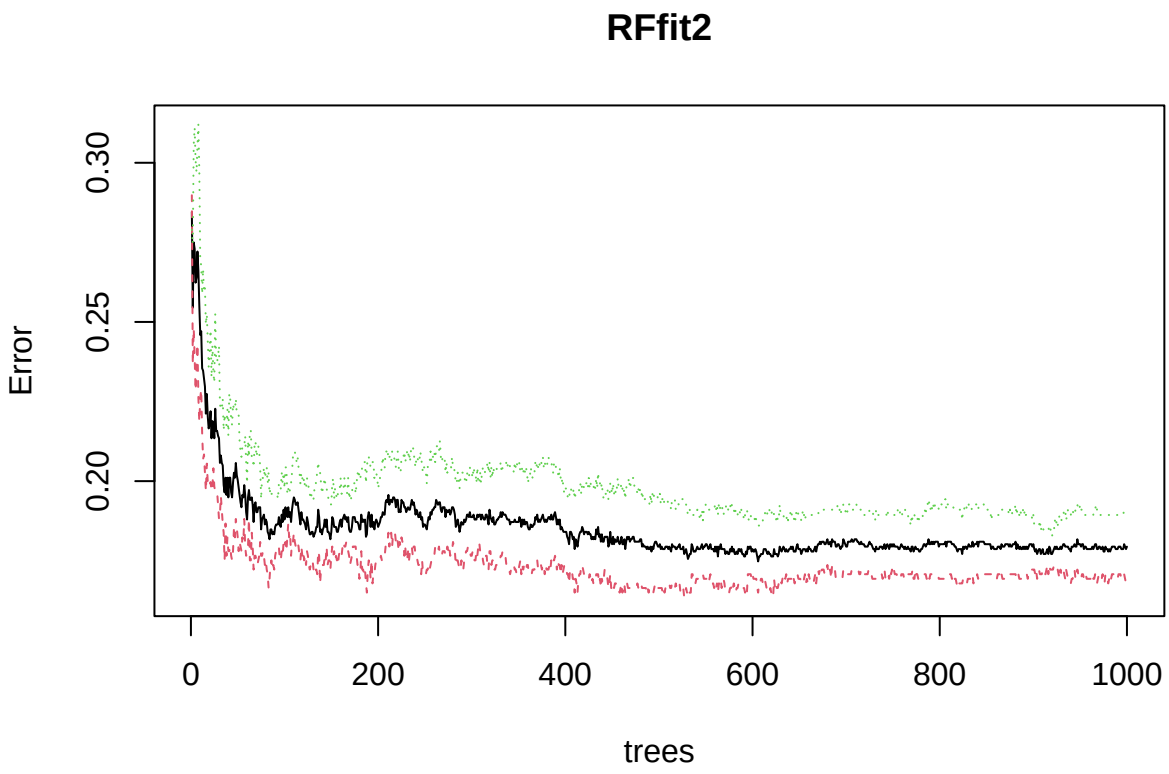


Second Random Forest Model Classification models typically do better with less nodes, and so we will not adjust the number of nodes in this second model. When we plot the error rate of the model, we can also see the error rate get smaller as the number of trees get larger, so we can try to increase the number of trees to 1000 to improve the model further. This results in an improved accuracy rate of 77.93% versus 76.55%. For a final model,

```
#Random Forest model adjusted
RFfit2 <- randomForest(Quality~. - Rating, data= train_set, ntree=1000)

#Calculate prediction
RFpredict2 <- predict(RFfit2, test_set)

#Create confusion matrix to determine accuracy
confusionMatrix(RFpredict2, test_set$Quality)$overall[["Accuracy"]]
```

```
## [1] 0.7793
```

```
#Plot error rate of number of trees
plot(RFfit2)
```

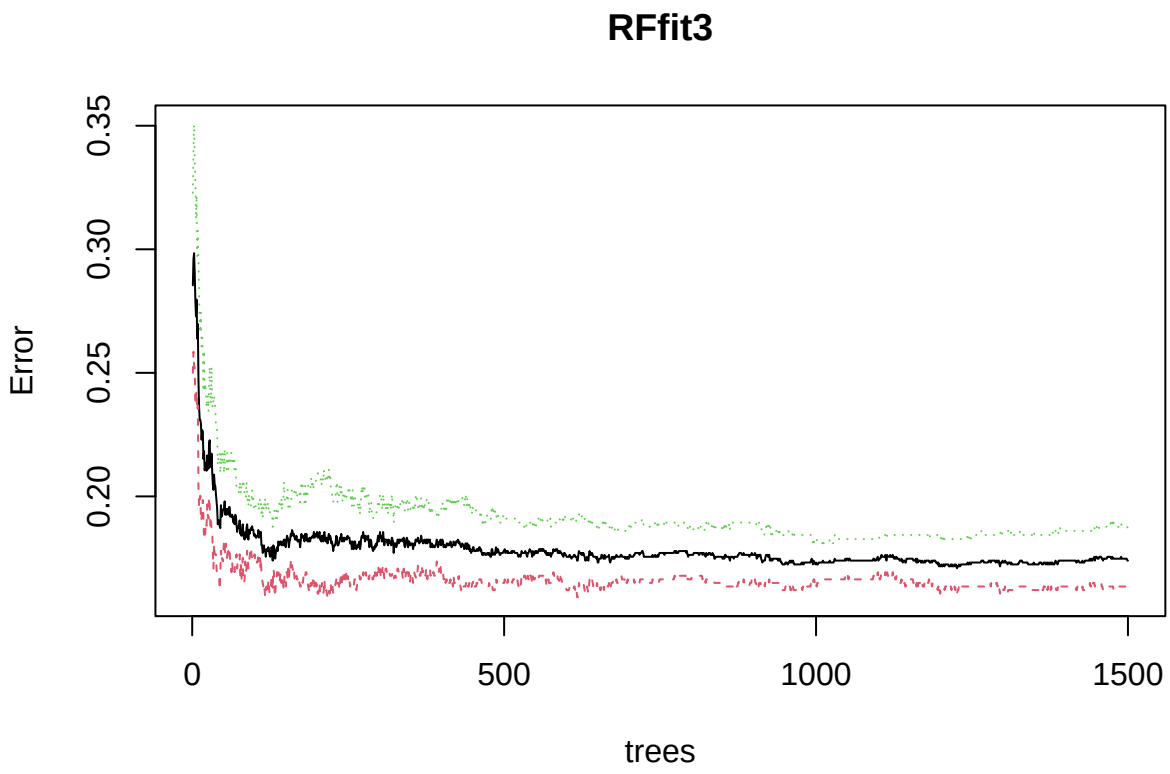


Third Random Forest Model For our third adjustment, we will further increase the number of trees to 1500 to see if additional improvements are made in the accuracy. This results in the same level of accuracy at a rate of 77.93%. Since no additional improvements were made, we will move forward with the model using 1000 trees.

```
#Random Forest model adjusted
RFfit3 <- randomForest(Quality~. - Rating, data= train_set, ntree=1500)

#Calculate prediction
RFpredict3 <- predict(RFfit3, test_set)

#Create confusion matrix to determine accuracy
confusionMatrix(RFpredict3, test_set$Quality)$overall[["Accuracy"]]
```

```
## [1] 0.7793
```

```
#Plot error rate of number of trees
plot(RFfit3)
```



Final Model Results

Multivariate Linear Regression Final Model Results

The model that provided the best RMSE of 0.6553 was the model that included volatile acidity, chlorides, total sulfur dioxide, pH, sulphites, and alcohol. This model was then tested against the validation set to confirm final RMSE results. The R-squared value for this final model was lower than the test models at 0.356, but all attributes remained significant predictors.

```
## First, we can use a baseline model using all attributes
fitfinal <- lm(Rating~ VA + Chl + TSO2 + pH + Sul + Alc, data=redwinedb)

#Examine summary for additional insights
summary(fitfinal)
```

```
##
## Call:
## lm(formula = Rating ~ VA + Chl + TSO2 + pH + Sul + Alc, data = redwinedb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5939 -0.3608 -0.0485  0.4573  1.9577
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.715      0.100   56.95  < 2e-16 ***
## VA            -1.523      0.155   -9.83  < 2e-16 ***
## Chl           -1.208      0.253   -4.77  2.0e-06 ***
## TSO2          -0.636      0.151   -4.21  2.8e-05 ***
## pH            -0.640      0.157   -4.06  5.1e-05 ***
## Sul            1.429      0.192    7.42  1.9e-13 ***
## Alc            1.927      0.117   16.48  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.651 on 1431 degrees of freedom
## Multiple R-squared:  0.359,  Adjusted R-squared:  0.356
## F-statistic:  134 on 6 and 1431 DF,  p-value: <2e-16
```

When using this new updated model, we find that even though the R-squared value worsened over the test set, the RMSE does improves well beyond any of the testing models to 0.6247. This will mean that by using this multivariate linear regression model, we can expect an error of 0.6247 points between predicted and actual ratings of red wines.

```
#Calculate predictions
y_hat <- predict(fitfinal, validation)

#Calculate RMSE
MLR_Final <- RMSE(y_hat, validation$Rating)

#Put into results table
FinalRMSE_Results <- tibble(Model = "Final Model", RMSE = MLR_Final)

knitr::kable(FinalRMSE_Results, caption = "Final MLR Model Results Table")
```

Table 5: Final MLR Model Results Table

Model	RMSE
Final Model	0.6247

Random Forest Final Model Results

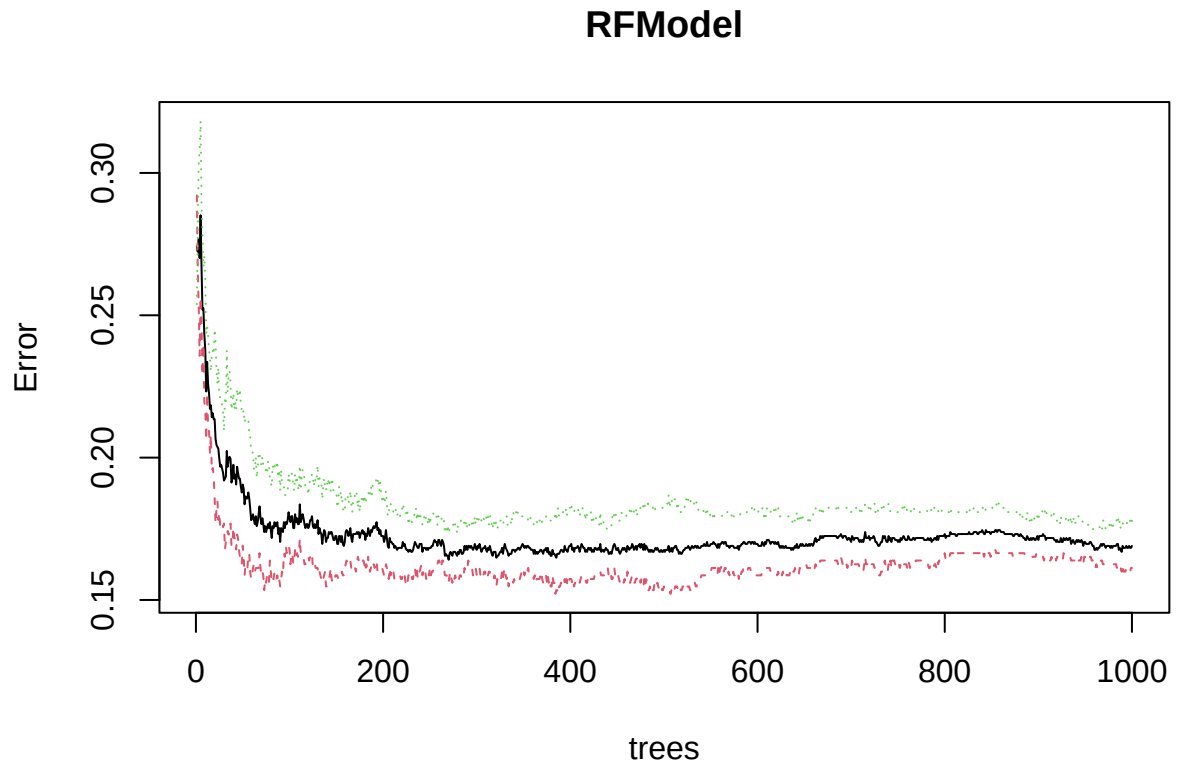
For the final random forest model, we use the initial default model that produces an even more improved accuracy over the test models at 80.12%. Examining the plot of error rates versus number of trees, it appears to be consistent with the test set, but the additional data appears to help improve the accuracy of the model. When looking further at the variables of importance, the model demonstrated consistent results with the linear regression model in finding alcohol levels being one of the most important predictors of wine quality, followed by sulphites and volatile acidity levels. Residual sugars, free sulfur dioxide, and the other measures of acidity were also consistent in being the least important in predicting the quality of red wines.

```
#Random Forest for all attributes except for Rating
RFModel <- randomForest(Quality~ . - Rating, data=redwinedb, ntree= 1000)

#Calculate prediction
RFpredict <- predict(RFModel, validation)
```

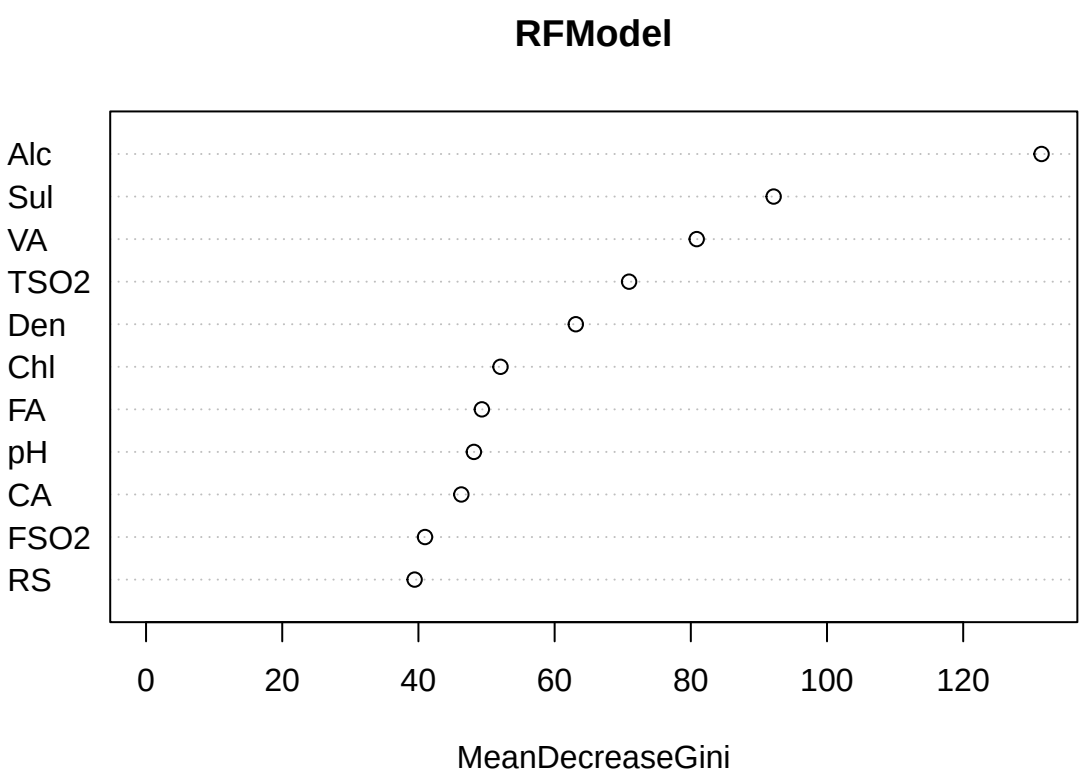
```
#Create confusion matrix to determine accuracy
confusionMatrix(RFpredict, validation$Quality)$overall[["Accuracy"]]
```

```
## [1] 0.8012
```

```
#Plot error rate of number of trees
plot(RFModel)
```




```
#Plot Variable Importance
varImpPlot(RFModel)
```



Conclusions

This report used two different types of models to predict red wine ratings and quality based on the combination of different physicochemical attributes commonly measured in wine. Both models performed reasonably well to predict both rating scores and classification of red wines as either good or not good quality, and both found alcohol, volatile acidity, and sulphite levels to be the best predictors of red wine quality. Classification models, such as the random forest method, could be useful for wine makers to help distinguish between good and not good wines at this medium level of quality, while the multivariate linear regression model could be more useful in predicting more specific ratings.

When looking at these three attributes, volatile acidity and sulphites make sense due to their roles in spoilage. Wines with higher volatile acidity levels are likely due in part to lower sulphite levels, which are meant to protect against oxidation and spoilage. Sulphites are also the product of the fermentation process, so it would be logical that higher levels of sulphites would indicate a more successful fermentation process, and therefore result in higher alcohol levels. These findings seem to indicate that the subtleties of terroir and other notes are less important in medium quality wines (i.e., wines in the range of this data set that were largely between 5-6 and no higher than 8 out of a scale of 1-10). The most distinguishing feature of this level of wine seems to be in the presence of any kind of spoilage that could be tasted with higher levels of volatile acidity. Future modeling using a regression model could focus on data sets with higher qualities of red wine to truly distinguish the physicochemical properties associated ratings with the best quality red wines.

Limitations of this study included a data sample of wines that were largely of medium quality and limited in region. Wines from a different region could potentially produce different results, and so an expanded regional source could be examined for future work. Another limitation was subjective nature of the human quality raters and the lack of additional information about them. While they used a blind tasting method,

there could be other confounding factors, such as objective rating abilities, familiarity with the wines of that region, and many other aspects that could not be controlled for in this sample. Future exploration could increase the number of taste testers and seek to control for some of the potential confounding factors to help make the ratings more consistent and accurate for future predictions.

References

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J.(2009). Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. http://www3.dsi.uminho.pt/pcortez

Siegel, J.(2017). The white wine trap. https://www.wine-business-international.com/wine/styles-regions/white-wine-trap#:~:text=We%20found%20that%2C%20first%2C%20reds,be%20rated%20higher%20than%2090.

Thermo Fisher Scientific. (2016). Wine analysis: From "Grape to Glass." https://tools.thermofisher.com/content/sfs/brochures/XX-72102-Wine-Analysis-XX72102-EN.pdf