COM SCI-X 450.4

Sooihk Ro

8/30/2021

UCLA

Using Regression Modeling for Red Wine Quality Prediction

Table of Contents

Machine Learning Problem Description

The quality of a wine is important for the consumers as well as the producers of wine. In the case of any other product, consumers of wine want to purchase good quality wines for the lowest price possible. Currently, the only way for consumers to get such information on quality of wine is through reviews by wine experts or from the wine community, which can be quite subjective and inaccurate. Many wine producers are interested in amethods to determine the quality of wine easily and accurately, which could increase the profit. Traditional methods of measuring quality, such as wine tasting by experts, are very time consuming. With the world's wine market, currently prcied at around $330 billion, growing at a rapid pace, the demand for an easy, quick, accurate and objective method to determine the wine quality is in demand. The overall wine quality in the market can improve significantly if an accurate and quick prediction through machine learning becomes a reality.

Some researchers have used machine learning techniques to assess wine quality, but there is still room for improvement. Gupta et al. predicted the quality of wine using linear regression, NN and SVM using 11 different physiochemical characteristics using a collection of white and red wine data set (4898 white wine and 1599 red wine samples). Beltran et al. predicted the quality of wine using SVM, neutral network and LDA to classify Chilean wine using wine aroma chromatogram data using 111 wine data set. Even though there are many research papers available online on this topic, no current model provides a universally accepted machine learning model. In this paper, a regression model is is built to quantify the nature of the relationship between the output (wine quality) and multiple input variables (from physiochemical tests including pH, alcohol percentage present in wine and amount of sugar left after fermentation).

Data Source and Data Preprocessing

Our data source is the red wine dataset avaiable from the UCI Machine Learning repository: https://archive.ics.uci.edu/ml/datasets/Wine+Quality. There are 11 input variables which are physicochemical properties with continuous values and a sequential output variable ranging from 1 to 10 based on sensory data(the median of 3 at least 3 wine expert's evaluations). The dataset did not require cleaning and had no missing values. The following table shows the dimensions of the dataset and the classes of its variables:

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

Exploratory Data Analysis

We visually explore the dataset with density plots, QQ plots and boxplots to discern patterns, outliers, and data distribution to aid predicting the quality of red.

The following density/histogram plots (Figure 1) were created to visualize the distribution of data:

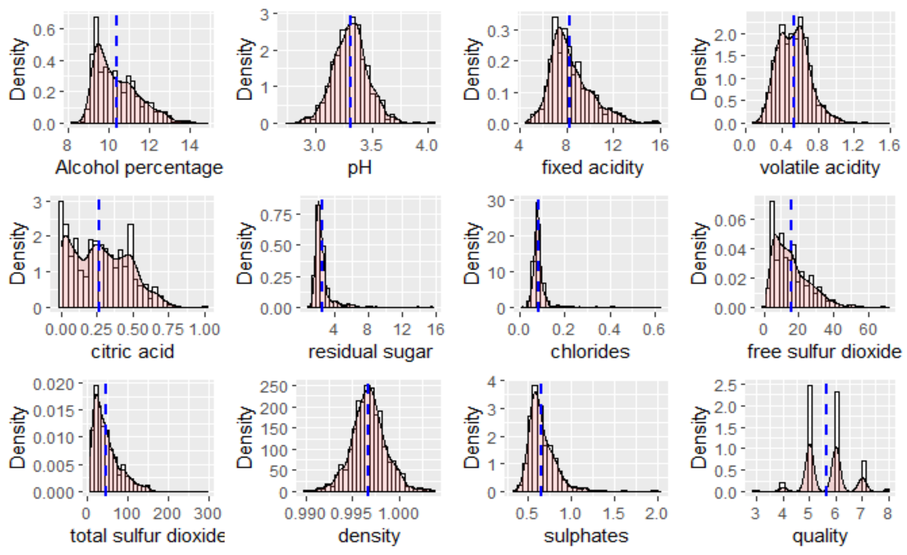

Figure 1: Density/Histogram plots

In Figure 1. pH and density appear normally distributed. The rest of the variables have postively skewed distributions. The pH levels stay between 3 to 4. Variables chlorides and residual.sugar are concentrated around 0.1 and 2 respectively.

QQ plots were created to help visually check if the predictors are normally distributed as models assume they are.

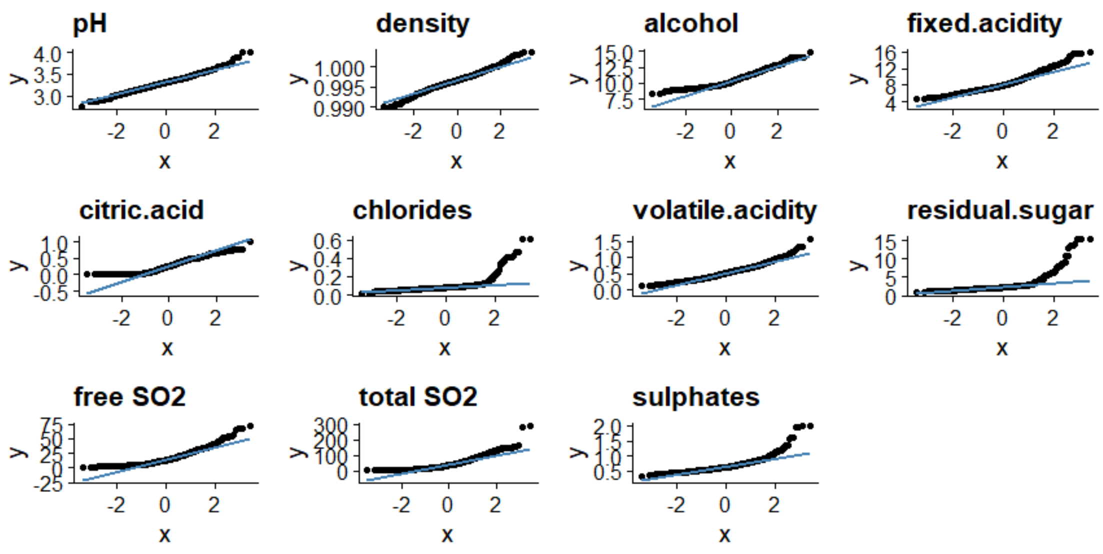

Figure 2: QQ plots, x: Theoretical Quantiles, y: Sample Quantiles

The QQ plots in Figure 2 for pH and density are shown to have near normal distribution. For the other variables, their data is close to normal distribution within 2 standard deviations of the mean but become right skewed. Positive skewed data is not desirable since high levels can cause misleading data. For our case the data is not too positively skewed.

The boxplots in Figure 3 show the distribution of data, mean, outliers, lower and upper limits per quality score for each predictor.

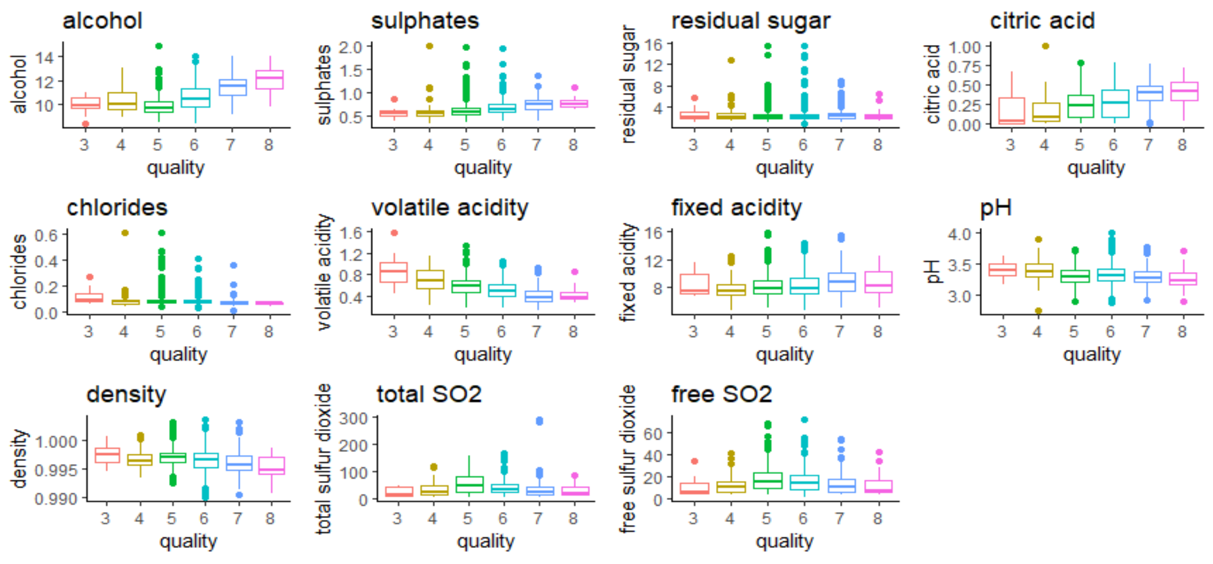

Figure 3: Box plots of predictors

Linear positive relationship appears between quality and alcohol, sulphates, fixed acidity, and citric acid. Higher levels or citric acid and sulphates are preferred since they contribute as preservatives and are antimicrobial which can affect the taste of the wine. Higher percentages of alcohol are also more popular for drinkers. There are also linear negative relationships between quality and density, volatile acidity, pH and chlorides. Visually we can not discern any pattern for fixed.acidity, residual.sugar, the sulfur dioxide variables. There are a number of outliers that will be explored and removed in a section ahead.

Feature Engineering and Selection

From the previous section we saw a number of outliers in the data which can affect out model's predictive power.

```
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
```

Looking at examples of the summary of the dataset, there are outliers in most of the variables where the max value and min values are far from the mean of each column. We will be using Cook's Distance to remove these outliers. Cook's distance considers an observation's leverage and residual and estimates the data point's influence.

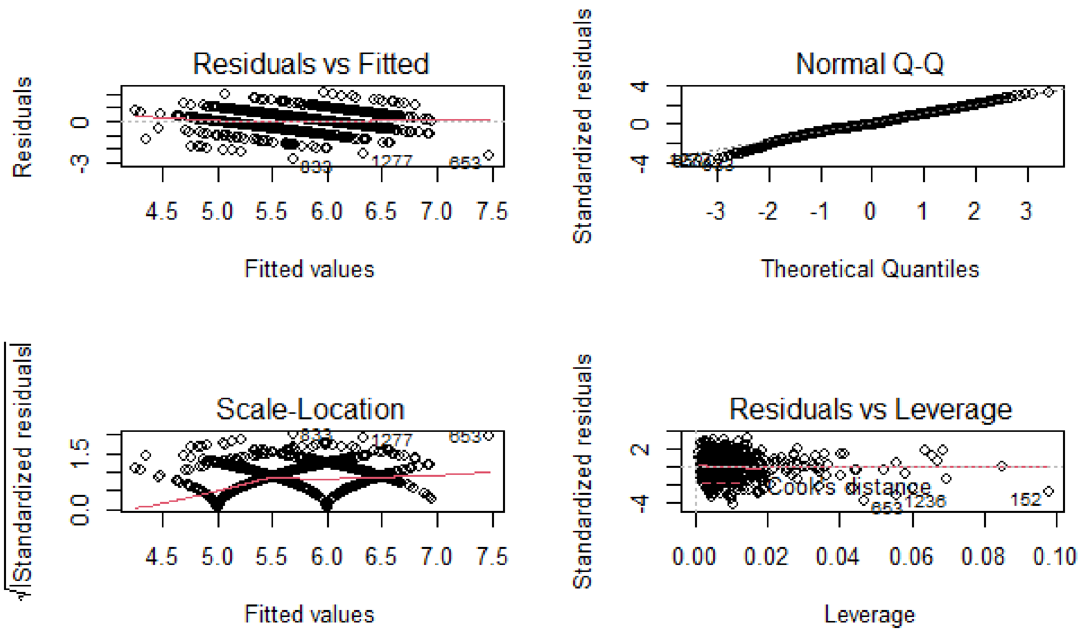

Figure 4: Diagnostic plots of multiple linear regression model using base dataset

In the diagnostic plots in Figure 4, we can see some outliers which we can remove to make our model a better fit. We use cooks.distance on our model to filter out values greater than 4x the mean. Figure 5 visualizes that the points above the blue line are 4x the mean. 67 data outliers were removed from the dataset. The adjusted R-squared had a 12% increase (0.3561 to 0.3989) and the P-values drastically decreased for most variables. The new diagnostic plots shown in Figure 6 have improved as well. The qqplot shows better normal distribution. The Residuals vs Leverage plot do not have any outstanding points with great leverage.

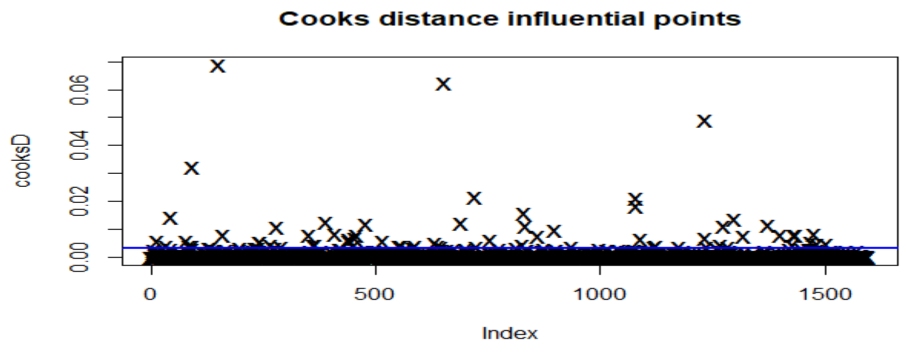

Figure 5: Cooks distance influential points

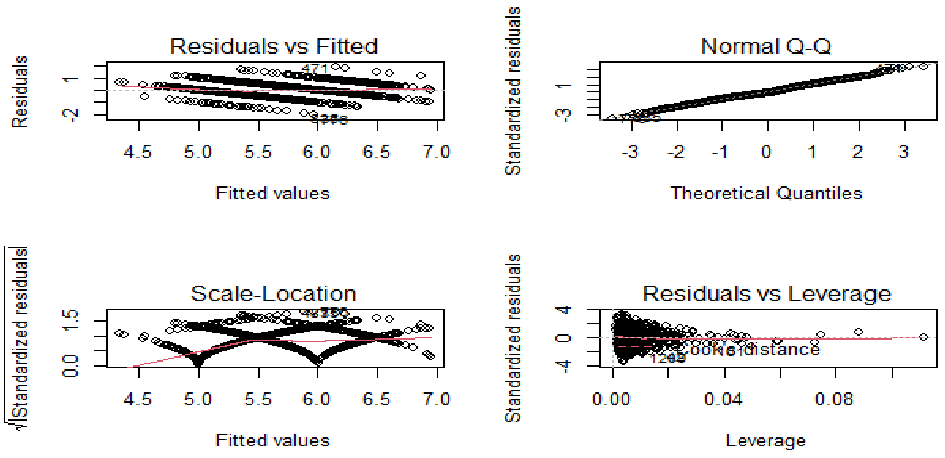

Figure 6: Diagnostic plots of multiple linear regression model using dataset with outliers removed

The table below shows that the skewness values have decreased after the outliers were removed. Skewness values between -0.5 and 0.5 mean the distribution is approximately symmetric. Residual.sugar and chlorides are still highly skewed to the right.

```
       fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
Before     0.9818293        0.6709624   0.3180386       4.536395  5.675017
After      0.8239847        0.5520417   0.2881522       4.245167  5.885137
       free.sulfur.dioxide total.sulfur.dioxide     density        pH sulphates
Before            1.249394             1.514109  0.07122077 0.1935018  2.426393
After             1.245703             1.192397 -0.07431032 0.2765693  1.555431
         alcohol   quality
Before 0.8600211 0.2175972
After  0.8230645 0.3937086
```

A correlation plot Figure (7) was created using the new dataset with the outliers removed to see each predictor’s correlation to quality and one another. We notice there are strong correlations between density and fixed.acidity (0.68), fixed.acidity and citric acid (0.68), fixed.acidity and pH (-0.69), free.sulfur.dioxide and total.sulfur.dioxide (0.68). These relatively high numbers can be explained since the sets of variables measure roughly the same property. Still these values can indicate high multicollinearity which can complicate knowing exactly which variables are truly predictive of the outcome. Thus we use the variance inflation factor(VIF) function on a multiple linear regression model and the outputed values fall below the generally accepted value of 10. Removing fixed.acidity and density lowered the VIF values below 3 for all variables. These results indicate we should run models without fixed.acidity and density as predictors. Alcohol, volatile.acidity and sulphates variables being the strongest correlated factors.

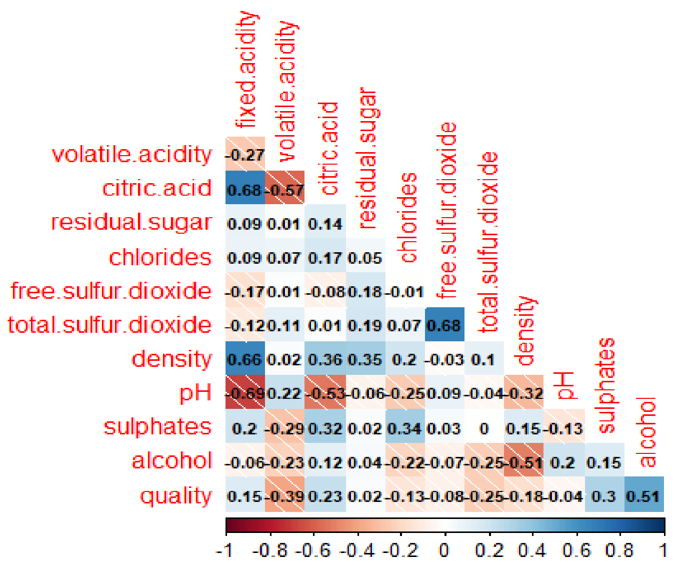

Figure 7: Correlation plot of variables

Utilizing a subset selection cross-validation method, we seek the feature size that will give the model with the least error. We first split our dataset into training and test sets in a 80:20 ratio. Figure 8 shows the for loop that performed cross-validation. The best models with the lowest test errors would be with 6,7 or 11 variables. There wasn't much difference in their R-Squared values so we choose to implement our starting multiple linear regresison model with 7 variables as our format. The 7 variables are: volatile.acidity, chlorides, free.sulfur.dioxide, total.sulfur.dioxide, pH, sulphates and alcohol.

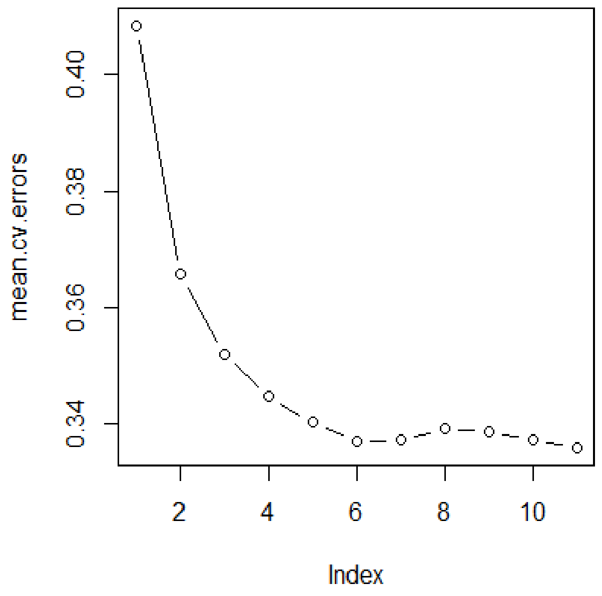

Figure 8: Cross-validation selection model

Modeling

Multiple Linear Regression

From our EDA, feature engineering and selection analysis, seven variables were used for the multiple linear regression model. We start with a multiple linear regression algorithm as a good baseline model to compare other models due to its simplicity. Here is the model summary:

```
## Call:
## lm(formula = quality ~ volatile.acidity + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + pH + sulphates + alcohol, data = redwine_training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.97956 -0.36748 -0.05185  0.43001  1.85093
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.5512746  0.4320576  10.534  < 2e-16 ***
## volatile.acidity     -0.8223278  0.1086646  -7.568 7.47e-14 ***
## chlorides            -1.8994321  0.4538312  -4.185 3.05e-05 ***
## free.sulfur.dioxide   0.0063545  0.0022034   2.884    0.004 **
## total.sulfur.dioxide -0.0045716  0.0007523  -6.077 1.64e-09 ***
## pH                   -0.5332445  0.1259486  -4.234 2.47e-05 ***
## sulphates             0.9822615  0.1252105   7.845 9.42e-15 ***
## alcohol               0.2819271  0.0179255  15.728  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5837 on 1218 degrees of freedom
## Multiple R-squared:  0.392,  Adjusted R-squared:  0.3885
## F-statistic: 112.2 on 7 and 1218 DF,  p-value: < 2.2e-16
```

The low p values show that the variables all have statistical significance. We found a 0.982% increase (± 0.1252) and 0.282% increase (± 0.01793) in the quality of red wine for every 1% increase in sulphates and alcohol respectively. Notably there is a 1.90% decrease (±0.4538), 0.822% decrease (±0.1087), and 0.533% decrease (±0.1259) in quality for every 1% increase in chlorides, volatile.acidity and pH respectively. These are the following performance metrics on the training set: R-squared: 0.3920, Root Mean Squared Error: 0.5817, Mean Absolute Error: 0.4628 and the test dataset resulted in the following metrics: R2: 0.4940, RMSE:0.5395, MAE:0.4346. The following confusion matrix results in the model's accuracy of 65%:

```
##          actual
## predicted  4  5  6  7  8
##         4  0  1  0  0  0
##         5  5 80 14  0  0
##         6  0 36 60 21  1
##         7  0  0  2  7  0
```

Least Absolute Shrinkage and Selection Operator (LASSO) Regression

We use a Lasso regression which regularizes models with penalty factors. Lasso will decreases coefficients of variables deemed not important for prediction even down to 0. Figrue 9 illustrates as the model's complexity increases, the MSE decreases. If the model has at least 6 or 5 models the MSE stays low.

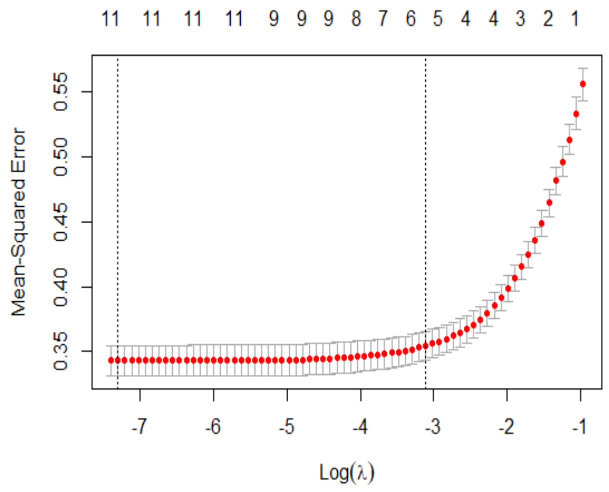

Figure 9: Mean-Squared Error as a function of (<U+03BB>)

Evaluating the model's performance on the test dataset resulted in the following metrics: R2: 0.4923, RMSE:0.4366, MAE:0.5405. The following confusion matrix results in the model's accuracy of 63%.

```
##         actual
## predicted  4  5  6  7  8
##        4  0  1  0  0  0
##        5  5 78 16  0  0
##        6  0 38 57 20  1
##        7  0  0  3  8  0
```

Partial Least Squares Regression

Partial Least Squares has some advantages over a Principle Component Regression since PLS is a supervised alternative to PCR. PLS attempts to find directions that help explain both the response and the predictors. However, while the supervised dimension reduction of PLS can reduce bias, it also has the potential to increase variance. Our PLS Regression summary is below:

```
## Data:    X dimension: 1226 11
##  Y dimension: 1226 1
## Fit method: kernelpls
## Number of components considered: 11
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.7467   0.6026   0.5920   0.5877   0.5855   0.5849   0.5849
## adjCV       0.7467   0.6024   0.5919   0.5875   0.5853   0.5847   0.5846
##        7 comps  8 comps  9 comps  10 comps  11 comps
## CV      0.5848   0.5850   0.5850    0.5851    0.5851
## adjCV   0.5846   0.5847   0.5847    0.5849    0.5849
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X          18.75    40.55    54.75    64.05    71.57    76.40    81.68    89.65
## quality    35.44    37.79    38.83    39.37    39.49    39.53    39.54    39.54
##          9 comps  10 comps  11 comps
## X          92.14     94.07    100.00
## quality    39.54     39.54     39.54
```

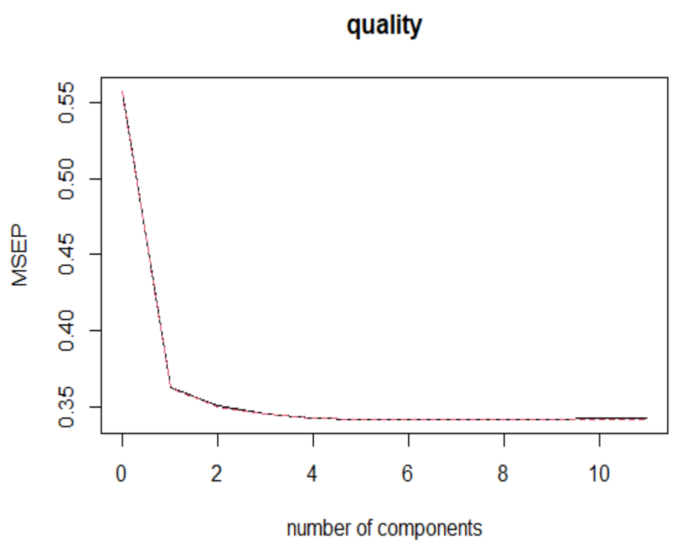

Figure 10: Mean-Squared Error as a function of the # of components

```
##              actual
## fitted.values  4  5  6  7  8
##             4  0  1  0  0  0
##             5  5 78 16  0  0
##             6  0 38 57 20  1
##             7  0  0  3  8  0
```

Random Forests Regression

Random forest combines predictions from multiple algorithms to make a more accurate prediction. It constructs several decision trees and outputs the average of all their predictions to generate one great prediction. Our random forest model summary is below:

```
## Call:
##  randomForest(formula = quality ~ ., data = redwine_training,      mtry = 11/3, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 0.274484
##                     % Var explained: 50.69
```

All variables were included instead of seven variables as it increased the accuracy of the model. The following Figure 11 shows the decreases in accuracy of the model if the values of that variable were randomly permuted. We see how important alcohol, sulphates, volatile.acidity and total.sulfur.dioxide are in predicting red wine quality.

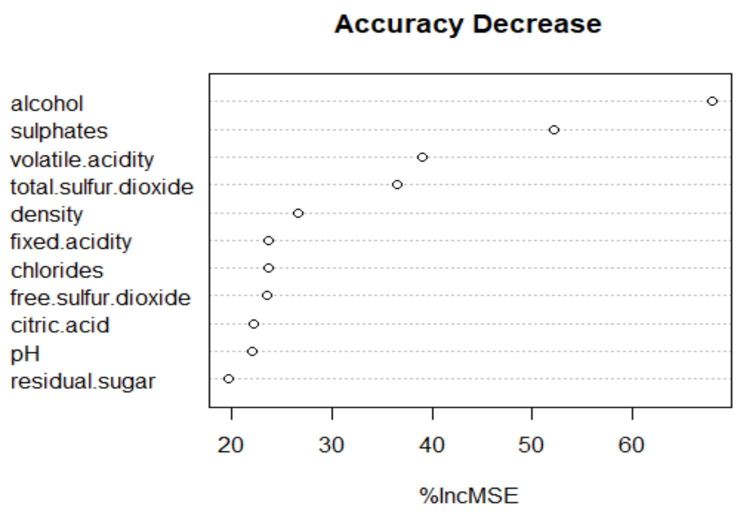

Figure 11: Variable Importance Plot

Evaluating the model's performance on the training set resulted in the following metrics: R-squared: 0.5092, Root Mean Squared Error: 0.5239, Mean Absolute Error: 0.3943 and the test dataset resulted in the following metrics: R2: 0.5292, RMSE:0.5235, MAE:0.4159. The following confusion matrix results in the model's accuracy of 67%:

```
##          actual
## predicted  4  5  6  7  8
##         5  3 82 14  0  0
##         6  2 35 57 15  0
##         7  0  0  5 13  1
```

Conclusion

From our analysis the RandomForest model as our best model based on multiple reasons. First, Random Forest model showed the highest R-squared value on both train and performance datasets. Second, Random Forest model showed the lowest root mean squared error as well as mean absolute error on both train and performance datasets. Third, Random Forest model had the highest accuracy on test data with 67%. Random Forest model was the best model out of 4 models probably because it tends to work well against a large dataset, withstands missing data, is not sensitive to outliers by binning the variables, is nonparametric, and balances the bias-variance trade-off well despite potential risk of overfitting and bias toward variables with more levels.

	R2	RMSE	MAE	Accuracy
Multiple Linear Regression	0.4914275	0.5394768	0.4346109	65%
Lasso Regression	0.4922896	0.5404831	0.4366197	63%
Partial Least Squares Regression	0.4915227	0.5410144	0.4355416	63%
RandomForest Regression	0.5256419	0.5234759	0.4159446	67%

We learned that to build the best machine learning model possible, we need to add as much data as possible, treat missing and/or outlier values, perform feature transformation and creation whenever necessary, go through feature selection process, experiment multiple algorithms to choose an ideal machine learning algorithm, and combine the result of multiple models through bagging and boosting. If we were to continue working on this problem, then we will definitely add more data points (wines from different regions, different types of wines, quality judgement from additional wine experts), run additional machine learning algorithms, and build a classification model to see if we can classify wines good or bad.

Appendix

R Code

```
library(ggplot2) #for visualization

library(lubridate)

library(zoo)

library(dplyr) #data manipulation

library(scales) #map data to aes

library(tidyverse) #data manipulation

library(GGally) #helper of ggplot2

library(corrplot) #

library(MASS) #data functions

library(cowplot) #data visualization

library(caret) #functions for training and plotting regression models

library(car)

library(leaps)

library(moments)

RNGkind(sample.kind = "Rounding")
```

#DATA SOURCE   AND PREPROCESSING

```
#read in data and separate out the semicolons, make 1st row a header, omit any missing values.

redwine_Data <- read.csv(file = "http://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-red.csv", header = TRUE, sep = ";") %>% na.omit()

attach(redwine_Data)

#shows dimensions of dataset

str(redwine_Data)

#summary of each column

summary(redwine_Data)
```

```
#EXPLORATORY DATA ANALYSIS

#making histograms/density plots to visualize spread of data for each predictor

hist1 <- ggplot(redwine_Data, aes(x=alcohol)) + geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="Alcohol percentage", y="Density") +

  geom_vline(aes(xintercept=mean(alcohol)), color="blue", linetype="dashed", size=1)


hist2 <- ggplot(redwine_Data, aes(x=pH)) + geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="pH", y="Density") +

  geom_vline(aes(xintercept=mean(pH)), color="blue", linetype="dashed", size=1)


hist3 <- ggplot(redwine_Data, aes(x=fixed.acidity)) + geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="fixed acidity", y="Density") +

  geom_vline(aes(xintercept=mean(fixed.acidity)), color="blue", linetype="dashed", size=1)


hist4 <- ggplot(redwine_Data, aes(x=volatile.acidity)) + geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="volatile acidity", y="Density") +

  geom_vline(aes(xintercept=mean(volatile.acidity)), color="blue", linetype="dashed", size=1)


hist5 <- ggplot(redwine_Data, aes(x=citric.acid)) + geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="citric acid", y="Density") +

  geom_vline(aes(xintercept=mean(citric.acid)), color="blue", linetype="dashed", size=1)


hist6 <- ggplot(redwine_Data, aes(x=residual.sugar)) + geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="residual sugar", y="Density") +

  geom_vline(aes(xintercept=mean(residual.sugar)), color="blue", linetype="dashed", size=1)


hist7 <- ggplot(redwine_Data, aes(x=chlorides)) + geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="chlorides", y="Density") +
  geom_vline(aes(xintercept=mean(chlorides)), color="blue", linetype="dashed", size=1)


hist8 <- ggplot(redwine_Data, aes(x=free.sulfur.dioxide)) +
geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="free sulfur dioxide", y="Density") +

  geom_vline(aes(xintercept=mean(free.sulfur.dioxide)), color="blue", linetype="dashed", size=1)


hist9 <- ggplot(redwine_Data, aes(x=total.sulfur.dioxide)) +
geom_histogram(aes(y=..density..),color="black", fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="total sulfur dioxide", y="Density") +

  geom_vline(aes(xintercept=mean(total.sulfur.dioxide)), color="blue", linetype="dashed", size=1)


hist10 <- ggplot(redwine_Data, aes(x=density)) + geom_histogram(aes(y=..density..),color="black",
fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="density", y="Density") +

  geom_vline(aes(xintercept=mean(density)), color="blue", linetype="dashed", size=1)


hist11 <- ggplot(redwine_Data, aes(x=sulphates)) + geom_histogram(aes(y=..density..),color="black",
fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="sulphates", y="Density") +

  geom_vline(aes(xintercept=mean(sulphates)), color="blue", linetype="dashed", size=1)


hist12 <- ggplot(redwine_Data, aes(x=quality)) + geom_histogram(aes(y=..density..),color="black",
fill="white") +

  geom_density(alpha=.2, fill="#FF6666") + labs(x="quality", y="Density") +

  geom_vline(aes(xintercept=mean(quality)), color="blue", linetype="dashed", size=1)


plot_grid(hist1, hist2, hist3, hist4)

plot_grid(hist5, hist6, hist7, hist8)

plot_grid(hist9, hist10, hist11, hist12)

#density plots show that most distributions or positively skewed. pH and density are normally distributed.

#------------------------------------------------------------------------------------------------------------------------
```

#QQ plots

```
qqplot_pH <- ggplot(redwine_Data, aes(sample=pH)) + stat_qq() + stat_qq_line(color="steelblue", lwd=1)
+

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="pH") + theme_cowplot()



qqplot_density <- ggplot(redwine_Data, aes(sample=density)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="density") + theme_cowplot()



qqplot_alcohol <- ggplot(redwine_Data, aes(sample=alcohol)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="alcohol") + theme_cowplot()



qqplot_chlorides <- ggplot(redwine_Data, aes(sample=chlorides)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="chlorides") + theme_cowplot()



qqplot_fixed.acidity <- ggplot(redwine_Data, aes(sample=fixed.acidity)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="fixed.acidity") + theme_cowplot()



qqplot_citric.acid <- ggplot(redwine_Data, aes(sample=citric.acid)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="citric.acid") + theme_cowplot()



qqplot_volatile.acidity <- ggplot(redwine_Data, aes(sample=volatile.acidity)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="volatile.acidity") + theme_cowplot()



qqplot_residual.sugar <- ggplot(redwine_Data, aes(sample=residual.sugar)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="residual.sugar") + theme_cowplot()



qqplot_free.sulfur.dioxide <- ggplot(redwine_Data, aes(sample=free.sulfur.dioxide)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="free.sulfur.dioxide") + theme_cowplot()
```

```
qqplot_total.sulfur.dioxide <- ggplot(redwine_Data, aes(sample=total.sulfur.dioxide)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="total.sulfur.dioxide") + theme_cowplot()


qqplot_sulphates <- ggplot(redwine_Data, aes(sample=sulphates)) + stat_qq() +
stat_qq_line(color="steelblue", lwd=1) +

  labs(x="Theoretical Quantiles", y="Sample Quantiles", title="sulphates") + theme_cowplot()


plot_grid(qqplot_pH, qqplot_density, qqplot_alcohol, qqplot_fixed.acidity)

plot_grid(qqplot_citric.acid, qqplot_chlorides, qqplot_volatile.acidity, qqplot_residual.sugar)

plot_grid(qqplot_free.sulfur.dioxide, qqplot_total.sulfur.dioxide, qqplot_sulphates)

#QQ plots of pH and density are shown to have near normal distribution. For the other variables, their
data is close to normal

#distribution within 2 standard deviations of the mean but become right skewed. Extremely positive
skewed data is not desirable

#since high levels can cause misleading data. For our case the data is not too positively skewed.
```

#--

#Create boxplots of each predictor vs quality

```
box1 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=alcohol, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="alcohol", title="alcohol") + theme_classic() + theme(legend.position="none")


box2 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=sulphates, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="sulphates", title="sulphates") + theme_classic() + theme(legend.position="none")


box3 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=residual.sugar, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="residual sugar", title="residual sugar") + theme_classic() +
theme(legend.position="none")


box4 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=citric.acid, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="citric acid", title="citric acid") + theme_classic() + theme(legend.position="none")
```

```
box5 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=chlorides, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="chlorides", title="chlorides") + theme_classic() + theme(legend.position="none")


box6 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=volatile.acidity, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="volatile acidity", title="volatile acidity") + theme_classic() +
theme(legend.position="none")


box7 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=fixed.acidity, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="fixed acidity", title="fixed acidity") + theme_classic() +
theme(legend.position="none")


box8 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=pH, color=as.factor(quality))) + geom_boxplot()
+

  labs(x="quality", y="pH", title="pH") + theme_classic() + theme(legend.position="none")


box9 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=density, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="density", title="density") + theme_classic() + theme(legend.position="none")


box10 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=total.sulfur.dioxide, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="total sulfur dioxide", title="total SO2") + theme_classic() +
theme(legend.position="none")


box11 <- ggplot(redwine_Data, aes(x=as.factor(quality), y=free.sulfur.dioxide, color=as.factor(quality))) +
geom_boxplot() +

  labs(x="quality", y="free sulfur dioxide", title="free SO2") + theme_classic() +
theme(legend.position="none")


plot_grid(box1,box2,box3,box4)

plot_grid(box5,box6,box7,box8)

plot_grid(box9,box10,box11)
```

```
#look for pattern in these boxplots

#We see linear positive relationship with quality from alchohol, sulphates, fixed acidity, and citric acid.

#We also see a linear negative relationship with quality from density, volatile acidity, pH and chlorides


#display table of quality scores. Most the scores are in 5 and 6 with little data for lower and higher scores.

table(redwine_Data$quality)



#---------------------------------------------Feature engineering---------------------------------------------

#Remove outliers, I will add code upon this.

summary(redwine_Data)

#Looking at the summary of each variable, we intially see there are outliers in most of the variables where
the

#max value and min values are very far from the mean of each column.

#We will be using Cook's Distance to remove these outliers as it is an estimate of a data point's influence.

#Cook's distance takes into account an obeservation's leverage and residual. We will investigate points
that

#are more than 4x the mean of all distances.


#First use a multiple linear regression model as a baseline model to compare

model1 <- lm(quality~., data=redwine_Data)

summary(model1)

#baseline model has adjusted R-squared of 0.3561

par(mfrow=c(2,2))

plot(model1)

#-------------------------------------------------------------------------------------------------------------


#From the diagnostic plots, we can see some outliers which we can remove to make our model a better
fit. We use

#cooks.distance function on our model to filter out values greater than 4x the mean. We plot the output of

#cooks.distance function to visualize which points are 4x the mean.

par(mfrow=c(1,1))

cooksD <- cooks.distance(model1)
```

```
plot(cooksD, pch = "x", cex=1.5, main="Cooks distance influential points")

abline(h=4*mean(cooksD, na.rm=TRUE), col = "blue", lwd = 2)


par(mfrow=c(3,2))

plot(lm(quality~alcohol, data=redwine_Data), which=c(5), main="alcohol")

plot(lm(quality~citric.acid, data=redwine_Data), which=c(5), main="ctric.acid")

plot(lm(quality~total.sulfur.dioxide, data=redwine_Data), which=c(5), main="total.sulfur.dioxide")

plot(lm(quality~pH, data=redwine_Data), which=c(5), main="pH")

plot(lm(quality~chlorides, data=redwine_Data), which=c(5), main="chlorides")

plot(lm(quality~fixed.acidity, data=redwine_Data), which=c(5), main="fixed.acidity")

#The figure above shows some examples of the outliers having significant leverage.


influential_points <- cooksD[(cooksD > (4*mean(cooksD, na.rm=TRUE)))]

wine_outliers <- redwine_Data[names(influential_points),]

redwine_Data2 <- anti_join(redwine_Data, wine_outliers)

#we see 67 data outliers were removed in new dataset.


model2 <- lm(redwine_Data2$quality~., data=redwine_Data2)

summary(model2)

#The adjusted R-squared has improved from 0.3561 to 0.3989(a 12% increase). The P-values have
decreased for the majority of the

#features as well.


par(mfrow=c(2,2))

plot(model2)

#The new diagnostic plots have improved as well. The qqplot shows better normal distribution. The
Residuals vs Leverage

#plot does not have any outstanding points with great leverage. Shows we can use linear regression with
4 assumptions.

skew1 <- skewness(redwine_Data$fixed.acidity)

skew1 <- append(skew1, skewness(redwine_Data$volatile.acidity))

skew1 <- append(skew1, skewness(redwine_Data$citric.acid))

skew1 <- append(skew1, skewness(redwine_Data$residual.sugar))
```

```
skew1 <- append(skew1, skewness(redwine_Data$chlorides))

skew1 <- append(skew1, skewness(redwine_Data$free.sulfur.dioxide))

skew1 <- append(skew1, skewness(redwine_Data$total.sulfur.dioxide))

skew1 <- append(skew1, skewness(redwine_Data$density))

skew1 <- append(skew1, skewness(redwine_Data$pH))

skew1 <- append(skew1, skewness(redwine_Data$sulphates))

skew1 <- append(skew1, skewness(redwine_Data$alcohol))

skew1 <- append(skew1, skewness(redwine_Data$quality))


skew2 <- skewness(redwine_Data2$fixed.acidity)

skew2 <- append(skew2, skewness(redwine_Data2$volatile.acidity))

skew2 <- append(skew2, skewness(redwine_Data2$citric.acid))

skew2 <- append(skew2, skewness(redwine_Data2$residual.sugar))

skew2 <- append(skew2, skewness(redwine_Data2$chlorides))

skew2 <- append(skew2, skewness(redwine_Data2$free.sulfur.dioxide))

skew2 <- append(skew2, skewness(redwine_Data2$total.sulfur.dioxide))

skew2 <- append(skew2, skewness(redwine_Data2$density))

skew2 <- append(skew2, skewness(redwine_Data2$pH))

skew2 <- append(skew2, skewness(redwine_Data2$sulphates))

skew2 <- append(skew2, skewness(redwine_Data2$alcohol))

skew2 <- append(skew2, skewness(redwine_Data2$quality))


summary_skew <- rbind(skew1, skew2)

colnames(summary_skew) <- c("fixed.acidity", "volatile.acidity", "citric.acid", "residual.sugar", "chlorides",

                    "free.sulfur.dioxide", "total.sulfur.dioxide", "density", "pH",

                    "sulphates", "alcohol","quality")

rownames(summary_skew) <- c("Before", "After")

summary_skew

#The skewness after the outliers were removed has decreased for most variables. Skewness values between -0.5

#and 0.5 mean the distribution is approximately symmetric. Residual.sugar and chlorides are still highly skewed to the right.
```

```
#------------------------------------------------------------------------------------------------------------

# we could log transform the data to help fix the positively skewed data for several variables. When we
look at a plot of the

# log transformation, it does not improve the normalization or the linearity between quality and the
predictors. Will most likely

#not need to transform


#log_redwine <- log(redwine_Data[,1:11])

#log_redwine$quality <- redwine_Data$quality



#****************************************FEATURE SELECTION*****************************************


#find correlations, use cor() to produce matrix and corrplot to display a plot

par(mfrow=c(1,1))

cor(redwine_Data2)

corrplot(cor(redwine_Data2), method="shade", type="lower",addCoef.col =
"black",diag=FALSE,number.cex=0.7)

#density and fixed.acidity (0.68), fixed.acidity and citric acid (0.68), fixed.acidity and pH (-0.69),
free.sulfur.dioxide

#and total.sulfur.dioxide (0.68) have strong correlation, perhaps high multicollinearity


#display in order the value the predictors correlate to Quality

rankQuality <- cor(redwine_Data2)[-12,12] %>% abs()

head(rankQuality[order(rankQuality, decreasing=TRUE)],12)

#We see that alcohol, volatile.acidity, sulphates, and citric acid have the highest correlation with quality.


#---------------------------------------------------------------------------------------------------------------------

#split dataset into training and testing, 70:30 split. Leaves 1119 obs for training and 381 obs for testing.


set.seed(3)
```

```
redwine_training <- sample_frac(tbl=redwine_Data2, replace = FALSE, size = 0.80)

redwine_test <- anti_join(redwine_Data2, redwine_training)



#We also want to check multicollinearity(variables in a regression are correlated with each other). If we
have

#multicollinearity then we will not know exactly which variables are truly predictive of the outcome.

#Thus we use the variance inflation factor(VIF) function to compute a multicollinearity score. The VIF
measures

#by how much the variance of the coefficient is inflated from multicollinearity.

library(car)

model2 <- lm(quality~., data=redwine_training)

vif(model2)

#All VIF values are below the generally accepted value of 10. Fixed acidity and density values show

#that they are highly correlated We'll try a model without fixed.acidity

model3 <- lm(quality~.-fixed.acidity, data=redwine_training)

vif(model3)

#Without fixed.acidity, VIF values of other variables have decreased, especially density. We can consider

#removing density as well.

#------------------------------------------------------------------------------------------------------------


#choosing model size using k-fold cross validation

library(leaps)

#write predict method to use regsubsets() since there is no predict() method for regsubsets()

predict.regsubsets =function (object , newdata ,id ,...){

  form=as.formula (object$call [[2]])

  mat=model.matrix(form ,newdata )

  coefi=coef(object ,id=id)

  xvars=names(coefi)

  mat[,xvars]%*%coefi

}
```

```
reg.best=regsubsets(quality~.,data=redwine_Data2 , nvmax=11)

k=10

set.seed(3)

folds=sample (1:k,nrow(redwine_Data2),replace=TRUE)

cv.errors=matrix (NA,k,11, dimnames =list(NULL , paste (1:11) ))


#ISLR for loop using predict method above

for(j in 1:k){

  best.fit=regsubsets(quality~.,data=redwine_Data2[folds!=j,], nvmax=11)

  for(i in 1:11){

     pred=predict(best.fit ,redwine_Data2[folds ==j,],id=i)

     cv.errors[j,i]= mean( ( redwine_Data2$quality[ folds==j]-pred)^2)

    }

}

mean.cv.errors=apply(cv.errors ,2, mean)

mean.cv.errors

par(mfrow=c(1,1))

plot(mean.cv.errors ,type="b")


coef(reg.best ,7)

#Based on plot, we will use 7 variables instead of 11. They both have near identical values but 7
variables is simpler than 11.

#These are the variables we will be using for all models. These variables chosen are in argeement with
what we found in our

#correlation ranking and VIF findings.

#1. volatile.acidity

#2. chlorides

#3. free.sulfur.dioxide

#4. total.sulfur.dioxide

#5. pH

#6. sulphates
```

#7. alcohol

#*MODELING***

#Need RMSE, R2, MAE, MSE

library(modelr)

#Multiple Linear Regression

set.seed(3)

multi_winefit <- lm(quality~volatile.acidity + chlorides + free.sulfur.dioxide + total.sulfur.dioxide + pH + sulphates + alcohol,

data=redwine_training)

summary(multi_winefit)

#Combining the results of or EDA, correlation, and feature selection we have chosen to prioritize seven variables out of

#We used the 7 variables chosen by stepwise k-fold cross validation regression for the model of multiple linear regression.

#Multiple linear regression is a simple and good basis model to compare our other models to predict quality. The chosen variables

#are highly correlated to our dependent quality variable and their p-value shows their statistical significance.

#Performance using train dataset

multi_R2_training <- rsquare(multi_winefit, redwine_training) #R-squared 0.391984

multi_RMSE_training <- rmse(multi_winefit, redwine_training) #Root Mean Squared Error 0.581757

multi_MAE_training <- mae(multi_winefit, redwine_training) #Mean Absolute Error 0.462793

#Performance using test dataset

multi_R2_test <- rsquare(multi_winefit, redwine_test) #R-squared 0.491427

multi_RMSE_test <- rmse(multi_winefit, redwine_test) #Root Mean Squared Error 0.539477

multi_MAE_test <- mae(multi_winefit, redwine_test) #Mean Absolute Error 0.434611

#Make a confusion matrix

multi_pred_test <- predict(multi_winefit, redwine_test) #predict with test dataset

```
multi_pred_round <- round(multi_pred_test) #round off values to whole numbers for confusion matrix
multi_confusion_matrix <- table(predicted = multi_pred_round, actual = redwine_test$quality)
multi_confusion_matrix
#accuracy of the model on test data
mean(multi_pred_round==redwine_test$quality) #accuracy is 65%

#-----------------------------------------------------------------------------------------------------------------
#LASSO
library(glmnet)

#prepare model matrix training and test sets
xtrain <- model.matrix(quality~., redwine_training)[,-1]
ytrain <- redwine_training$quality
xtest <- model.matrix(quality~., redwine_test)[,-1]
ytest <- redwine_test$quality

set.seed(3)
cv.lasso.wine <- cv.glmnet(xtrain, ytrain, alpha=1)
plot(cv.lasso.wine)
#obtain best lambda for least MSE
bestlam <- cv.lasso.wine$lambda.min # 0.0006777187

#model with best lambda
lasso.best.wine <- glmnet(xtrain, ytrain, alpha=1, lambda=bestlam)
coef(lasso.best.wine)

#Performance metrics on train dataset
lasso.pred2 <- predict(lasso.best.wine, xtrain)
lasso_R2_winetrain <- cor(ytest, lasso.pred2)^2 #0.0.3953817
lasso_MAE_winetrain <- mean(abs(ytest-lasso.pred2)) #0.4617966
lasso_RMSE_winetrain <- sqrt(mean((ytest-lasso.pred2)^2)) #0.0.5801316
```

```
#Performance metrics on test dataset

lasso.pred <- predict(lasso.best.wine, xtest)

lasso_R2_wine <- cor(ytest, lasso.pred)^2 #0.492290

lasso_MAE_wine <- mean(abs(ytest-lasso.pred)) #0.4366197

lasso_RMSE_wine <- sqrt(mean((ytest-lasso.pred)^2)) #0.5404831


#Make a confusion matrix

lasso_pred_round <- round(lasso.pred) #round off values to whole numbers for confusion matrix

lasso_confusion_matrix <- table(predicted = lasso_pred_round, actual = redwine_test$quality)

lasso_confusion_matrix

#accuracy of the model on test data

mean(lasso_pred_round==redwine_test$quality) #accuracy is 63%
```

#---

#Partial Least Squares

```
library(pls)


set.seed(3)

pls.redwine <- plsr(quality~., data=redwine_training, scale=TRUE, validation="CV")

summary(pls.redwine)

validationplot(pls.redwine, val.type="MSEP")

# we see through the summary and plot that lowest cross-validation error occurs when M=7(ncomp=7) 
partial

#least squares directions are used.


#Performace using train dataset

pls.R2.training <- rsquare(pls.redwine, redwine_training) #0.389671

pls.RMSE.training <- rmse(pls.redwine, redwine_training) #0.583079

pls.MAE.training <- mae(pls.redwine, redwine_training) #0.464885


#Performace using test dataset

pls.R2.test <- rsquare(pls.redwine, redwine_test) #0.491523
pls.RMSE.test <- rmse(pls.redwine, redwine_test) #0.541014

pls.MAE.test <- mae(pls.redwine, redwine_test) #0.435541


pls_pred_test <- predict(pls.redwine, newdata=redwine_test, ncomp=7)

pls_pred_round <- round(pls_pred_test)

pls_confusion_matrix <- table(fitted.values = pls_pred_round, actual = redwine_test$quality)

pls_confusion_matrix

#accuracy of the model on test data

mean(pls_pred_round==redwine_test$quality) #Accuracy is 63%
```

#---

#RandomForest

```
library(randomForest)


set.seed(3)

#rf.redwine <- randomForest(quality~volatile.acidity + chlorides + total.sulfur.dioxide +

#                         pH + sulphates + alcohol, data=redwine_training, mtry = 2, importance=TRUE)

rf.redwine <- randomForest(quality~., data=redwine_training, mtry = 11/3, importance=TRUE)

#summary of model

print(rf.redwine)


#variable importance plot,The former is based upon the mean decrease of accuracy in predictions on the

#out of bag samples when a given variable is excluded from the model. The latter is a measure

#of the total decrease in node impurity that results from splits over that variable, averaged over all trees

par(mfrow=c(1,1))

varImpPlot(rf.redwine, type=1, main="Accuracy Decrease")


#Evaluation

#Performance using train dataset
```

```
rf_R2_training <- rsquare(rf.redwine, redwine_training) #R-squared 0.509190
rf_RMSE_training <- rmse(rf.redwine, redwine_training) #Root Mean Squared Error 0.523912
rf_MAE_training <- mae(rf.redwine, redwine_training) #Mean Absolute Error 0.394347

#Performance using test dataset


rf_R2_test <- rsquare(rf.redwine, redwine_test) #R-squared 0.525642
rf_RMSE_test <- rmse(rf.redwine, redwine_test) #Root Mean Squared Error 0.523476
rf_MAE_test <- mae(rf.redwine, redwine_test) #Mean Absolute Error 0.415944

#Make a confusion matrix
rf_pred_test <- predict(rf.redwine, newdata=redwine_test) #predict with test dataset
rf_pred_round <- round(rf_pred_test) #round off values to whole numbers for confusion matrix
rf_confusion_matrix <- table(predicted = rf_pred_round, actual = redwine_test$quality)
rf_confusion_matrix
#accuracy of the model on test data
mean(rf_pred_round==redwine_test$quality) #accuracy is 67%

#-----------------------------------------------------------------------------------------------------
```