HarvardX Capstone Project 2: Wine Quality

Annie Zhang

INTRODUCTION/OVERVIEW

For some of us stuck at home during quarantine, you may have made the time a little easier with a glass of wine or two... or maybe you even became a wine connoisseur? If you've ever gone wine tasting, you may have experienced the somewhat pretentious subjective evaluation of wine quality. But some studies have shown that actually with blind wine tasting, the judgement of wine quality even by professional wine tasting experts is highly unreliable and inconsistent. So then, is there actually a more objective way to evaluate wine quality?

This is my submission for the second project of the HarvardX Data Science Capstone course. In this project, a Wine quality dataset will be explored and models made for predicting quality score based on the various different objective chemical attributes of the wine.

"Wine improves with age. The older I get, the better I like it." - Anonymous

Project Objective

The goal of this project is to predict wine quality scores given data on objective wine component measures such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, sulfur dioxide, density, pH, sulphates, and alcohol. Various different regression models will be used. The strength of each model will be assessed by comparing the predicted ratings with the validation subset via RMSE.

RMSE, or root mean squared error, is used to measure model accuracy. A low RMSE indicates better model performance. RMSE penalizes for larger errors due to the squaring. The best model for movie prediction will be the one with the lowest RMSE.

RMSE is calculated as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

Exploratory Analysis

The Dataset

The dataset to be used is sourced from the UCI Machine Learning Repository. Only the red wine dataset will be used. It consists of 1599 rows of wine quality observations with 12 columns of variables.

The direct link is as follows: https://archive.ics.uci.edu/ml/datasets/Wine+Quality

I took a good long browse of the UCI Machine Learning Repository before coming across this interesting and amusing gem. I thought it would be fun to see what kinds of models could be made to predict wine quality.

Import and View Dataset

```
#import data from csv file
data <- read.csv2("winequality-red.csv", stringsAsFactors=FALSE, header=TRUE)

#take a look at the data
head(data)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4              0.7           0            1.9     0.076
## 2           7.8             0.88           0            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4              0.7           0            1.9     0.076
## 6           7.4             0.66           0            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968  3.2      0.68     9.8
## 3                  15                   54   0.997 3.26      0.65     9.8
## 4                  17                   60   0.998 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality
## 1       5
## 2       5
## 3       5
## 4       6
## 5       5
## 6       5
```

```
summary(data)
```

```
##  fixed.acidity      volatile.acidity   citric.acid        residual.sugar
##  Length:1599        Length:1599        Length:1599        Length:1599
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##   chlorides         free.sulfur.dioxide total.sulfur.dioxide   density
##  Length:1599        Length:1599         Length:1599           Length:1599
##  Class :character   Class :character    Class :character      Class :character
##  Mode  :character   Mode  :character    Mode  :character      Mode  :character
##
##
##
##       pH             sulphates           alcohol             quality
```

```
##  Length:1599        Length:1599        Length:1599        Min.   :3.000
##  Class :character   Class :character   Class :character   1st Qu.:5.000
##  Mode  :character   Mode  :character   Mode  :character   Median :6.000
##                                                           Mean   :5.636
##                                                           3rd Qu.:6.000
##                                                           Max.   :8.000
```

```
str(data)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : chr  "7.4" "7.8" "7.8" "11.2" ...
##  $ volatile.acidity    : chr  "0.7" "0.88" "0.76" "0.28" ...
##  $ citric.acid         : chr  "0" "0" "0.04" "0.56" ...
##  $ residual.sugar      : chr  "1.9" "2.6" "2.3" "1.9" ...
##  $ chlorides           : chr  "0.076" "0.098" "0.092" "0.075" ...
##  $ free.sulfur.dioxide : chr  "11" "25" "15" "17" ...
##  $ total.sulfur.dioxide: chr  "34" "67" "54" "60" ...
##  $ density             : chr  "0.9978" "0.9968" "0.997" "0.998" ...
##  $ pH                  : chr  "3.51" "3.2" "3.26" "3.16" ...
##  $ sulphates           : chr  "0.56" "0.68" "0.65" "0.58" ...
##  $ alcohol             : chr  "9.4" "9.8" "9.8" "9.8" ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

Data Pre-Processing

```
# convert variables to numeric
data$fixed.acidity <- as.numeric(data$fixed.acidity)
data$volatile.acidity <- as.numeric(data$volatile.acidity)
data$citric.acid <- as.numeric(data$citric.acid)
data$residual.sugar <- as.numeric(data$residual.sugar)
data$chlorides <- as.numeric(data$chlorides)
data$free.sulfur.dioxide <- as.numeric(data$free.sulfur.dioxide)
data$total.sulfur.dioxide <- as.numeric(data$total.sulfur.dioxide)
data$density <- as.numeric(data$density)
data$pH <- as.numeric(data$pH)
data$sulphates <- as.numeric(data$sulphates)
data$alcohol <- as.numeric(data$alcohol)

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

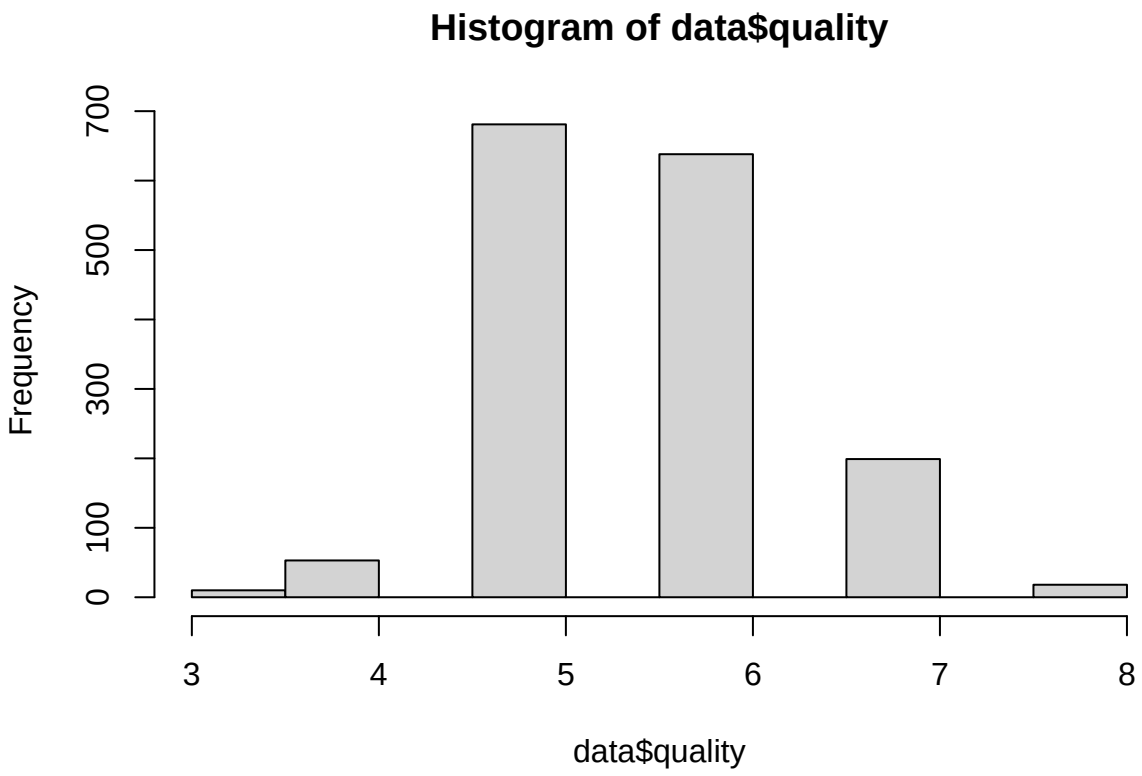
```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
# Validation set will be 10% of MovieLens data
set.seed(1)
test_index <- createDataPartition(y = data$quality, times = 1, p = 0.2, list = FALSE)
training <- data[-test_index,]
test <- data[test_index,]

# can also remove quality score from validation set
test_withoutquality <- test %>% select(-quality)
```

In the data pre-processing stage, I converted all predictor variables from character to numeric so it can be useful for analysis. Then I divided the dataset into the training and validation sets at random, with 80% training and 20% test set. This ratio was chosen because it's a good standard split that balances between variance and bias. If the training set is too large, there will be too much variance in prediction, but if the training set is too small, there may be more bias.

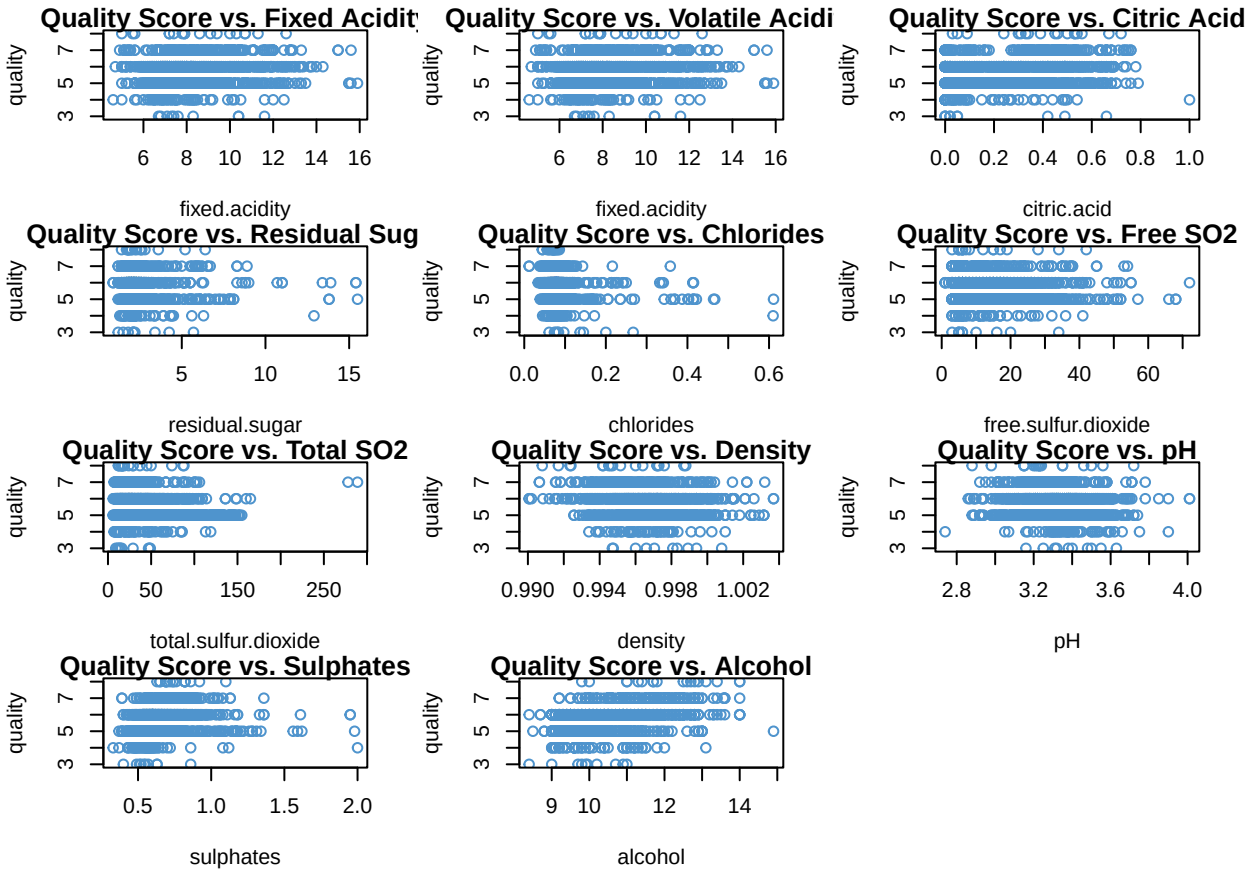
Data Visualization

```
# Plot of distribution of quality scores
hist(data$quality)
```




```
library(graphics)

par(mfrow=c(4,3), mar = c(4, 4, 1, 2))

# Plot of Quality vs. Fixed Acidity
plot(quality~fixed.acidity,data=data,
main="Quality Score vs. Fixed Acidity",
col="steelblue3")

# Plot of Quality vs. Volatile Acidity
plot(quality~fixed.acidity,data=data,
main="Quality Score vs. Volatile Acidity",
col="steelblue3")

# Plot of Quality vs. Citric Acid
plot(quality~citric.acid,data=data,
main="Quality Score vs. Citric Acid",
col="steelblue3")

# Plot of Quality vs. Residual Sugar
plot(quality~residual.sugar,data=data,
main="Quality Score vs. Residual Sugar",
col="steelblue3")

# Plot of Quality vs. Chlorides
plot(quality~chlorides,data=data,
```

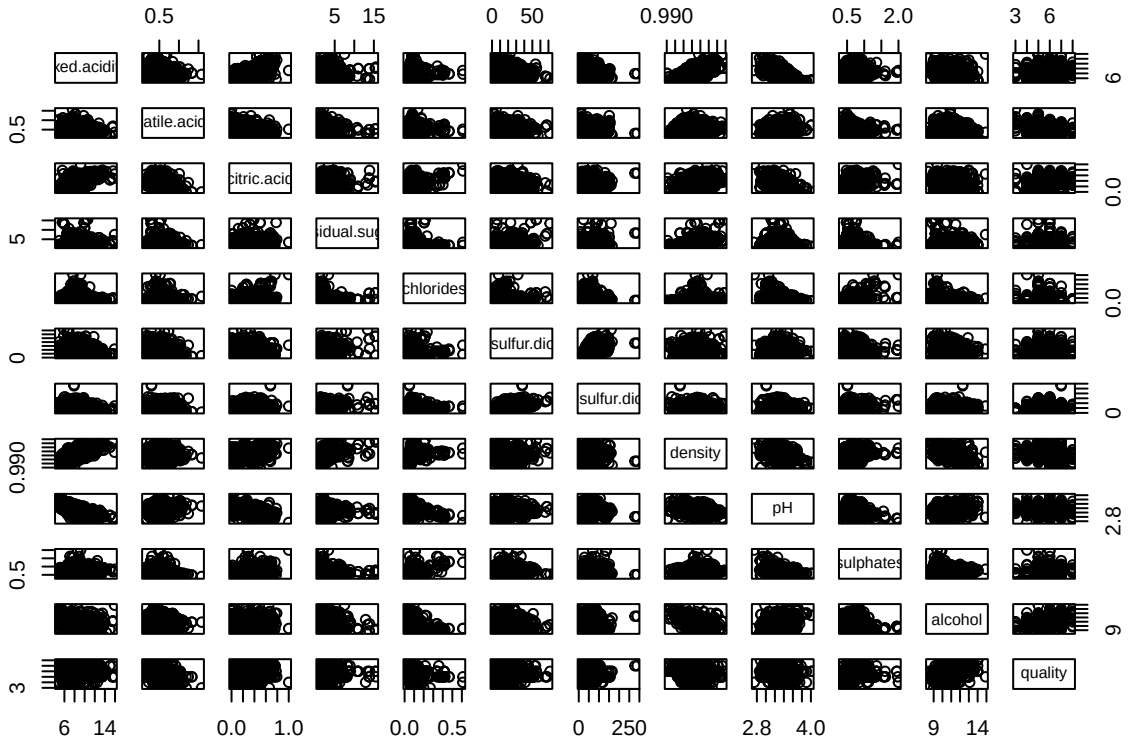
```
main="Quality Score vs. Chlorides",
col="steelblue3")

# Plot of Quality vs. free.sulfur.dioxide
plot(quality~free.sulfur.dioxide,data=data,
main="Quality Score vs. Free SO2",
col="steelblue3")

# Plot of Quality vs. total.sulfur.dioxide
plot(quality~total.sulfur.dioxide,data=data,
main="Quality Score vs. Total SO2",
col="steelblue3")

# Plot of Quality vs. Density
plot(quality~density,data=data,
main="Quality Score vs. Density",
col="steelblue3")

# Plot of Quality vs. pH
plot(quality~pH,data=data,
main="Quality Score vs. pH",
col="steelblue3")

# Plot of Quality vs. Sulphates
plot(quality~sulphates,data=data,
main="Quality Score vs. Sulphates",
col="steelblue3")

# Plot of Quality vs. Alcohol
plot(quality~alcohol,data=data,
main="Quality Score vs. Alcohol",
col="steelblue3")
```



The plots above show every predictor against the dependent variable (quality). The histogram shows the general distribution of the dependent variable. Of note is that the dependent variable is discrete.

Variable Exploration

```
# plot of all variables against each other
plot(data)
```




```
# Calculate correlations table (removed categorical variable)
library(corrplot)
```

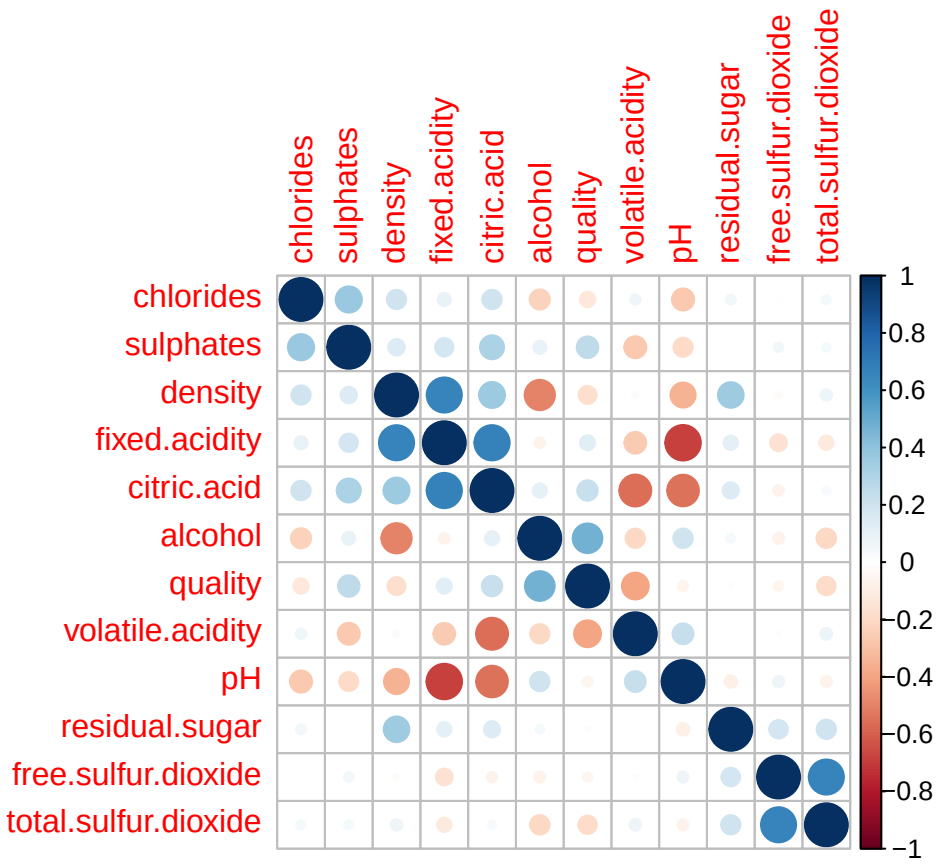
```
## corrplot 0.92 loaded
```

```
correlations <- cor(data)
correlations
```

```
##                       fixed.acidity volatile.acidity citric.acid residual.sugar
## fixed.acidity           1.00000000     -0.256130895  0.67170343    0.114776724
## volatile.acidity       -0.25613089      1.000000000 -0.55249568    0.001917882
## citric.acid             0.67170343     -0.552495685  1.00000000    0.143577162
## residual.sugar          0.11477672      0.001917882  0.14357716    1.000000000
## chlorides               0.09370519      0.061297772  0.20382291    0.055609535
## free.sulfur.dioxide    -0.15379419     -0.010503827 -0.06097813    0.187048995
## total.sulfur.dioxide   -0.11318144      0.076470005  0.03553302    0.203027882
## density                 0.66804729      0.022026232  0.36494718    0.355283371
## pH                     -0.68297819      0.234937294 -0.54190414   -0.085652422
## sulphates               0.18300566     -0.260986685  0.31277004    0.005527121
## alcohol                -0.06166827     -0.202288027  0.10990325    0.042075437
## quality                 0.12405165     -0.390557780  0.22637251    0.013731637
##                        chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity         0.093705186        -0.153794193          -0.11318144
## volatile.acidity      0.061297772        -0.010503827           0.07647000
## citric.acid           0.203822914        -0.060978129           0.03553302
```

```
## residual.sugar         0.055609535          0.187048995           0.20302788
## chlorides              1.000000000          0.005562147           0.04740047
## free.sulfur.dioxide    0.005562147          1.000000000           0.66766645
## total.sulfur.dioxide   0.047400468          0.667666450           1.00000000
## density                0.200632327         -0.021945831           0.07126948
## pH                    -0.265026131          0.070377499          -0.06649456
## sulphates              0.371260481          0.051657572           0.04294684
## alcohol               -0.221140545         -0.069408354          -0.20565394
## quality               -0.128906560         -0.050656057          -0.18510029
##                           density           pH    sulphates     alcohol
## fixed.acidity          0.66804729 -0.68297819  0.183005664 -0.06166827
## volatile.acidity       0.02202623  0.23493729 -0.260986685 -0.20228803
## citric.acid            0.36494718 -0.54190414  0.312770044  0.10990325
## residual.sugar         0.35528337 -0.08565242  0.005527121  0.04207544
## chlorides              0.20063233 -0.26502613  0.371260481 -0.22114054
## free.sulfur.dioxide   -0.02194583  0.07037750  0.051657572 -0.06940835
## total.sulfur.dioxide   0.07126948 -0.06649456  0.042946836 -0.20565394
## density                1.00000000 -0.34169933  0.148506412 -0.49617977
## pH                    -0.34169933  1.00000000 -0.196647602  0.20563251
## sulphates              0.14850641 -0.19664760  1.000000000  0.09359475
## alcohol               -0.49617977  0.20563251  0.093594750  1.00000000
## quality               -0.17491923 -0.05773139  0.251397079  0.47616632
##                           quality
## fixed.acidity          0.12405165
## volatile.acidity      -0.39055778
## citric.acid            0.22637251
## residual.sugar         0.01373164
## chlorides             -0.12890656
## free.sulfur.dioxide   -0.05065606
## total.sulfur.dioxide  -0.18510029
## density               -0.17491923
## pH                    -0.05773139
## sulphates              0.25139708
## alcohol                0.47616632
## quality                1.00000000
```

```
# Display correlation plot
corrplot(correlations, order="hclust")
```



The above plots show the correlations of all the variables with each other. There is moderate levels of correlation between the predictors.

METHODS/ANALYSIS

1. Multivariate Regression Models

A multivariate regression model is appropriate to model this dataset for the purpose of our objective, because it models the relationship between the multiple predictors and the dependent response variable, quality score. As we have 11 predictors, it is called multivariate regression. Regression can be used to predict values of the response variable given values of the predictor variables.

We will first build two multivariate regression models using all the predictors. We will build these models using the training set. Model 2 has the potential outliers removed, as determined using Cook's distance as above.

```
# Build first basic multivariate regression model with all predictors and data
model1 <- lm(quality~., data=training)
summary(model1)
```

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

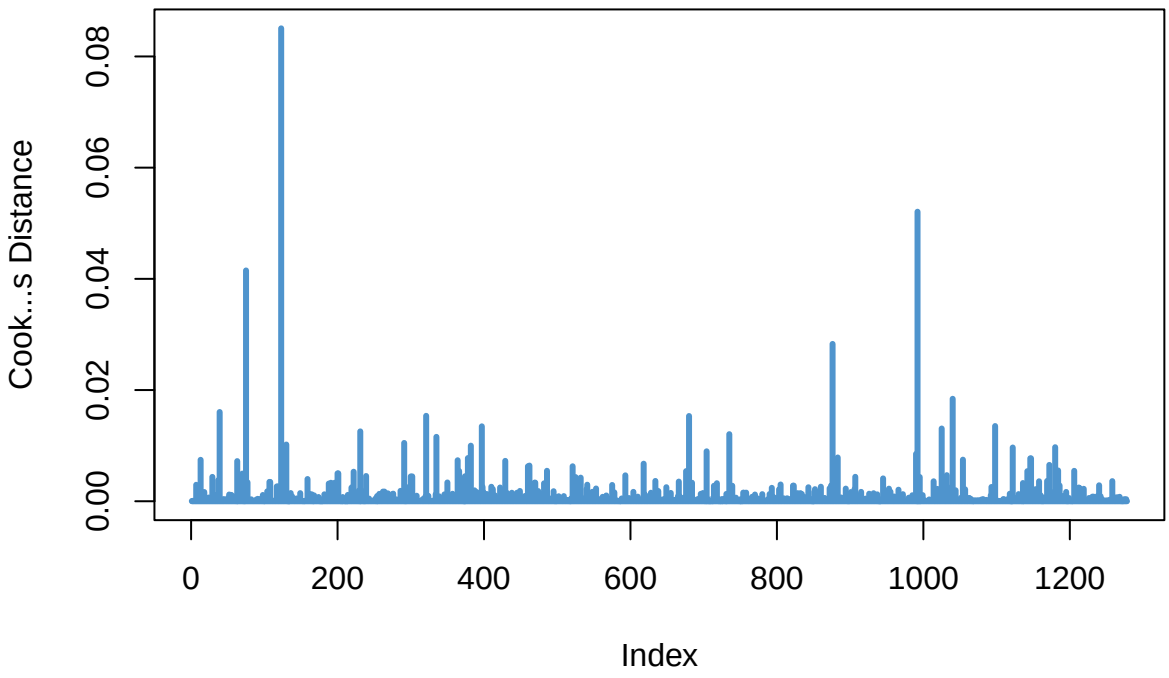
```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3206 -0.3709 -0.0565  0.4522  2.0526
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.762e+01  2.368e+01   1.167   0.2435
## fixed.acidity         4.642e-02  2.928e-02   1.585   0.1132
## volatile.acidity     -9.945e-01  1.325e-01  -7.503 1.17e-13 ***
## citric.acid          -1.785e-01  1.625e-01  -1.099   0.2721
## residual.sugar        1.599e-02  1.622e-02   0.986   0.3243
## chlorides            -1.907e+00  4.581e-01  -4.163 3.35e-05 ***
## free.sulfur.dioxide   5.604e-03  2.475e-03   2.264   0.0237 *
## total.sulfur.dioxide -3.677e-03  8.305e-04  -4.427 1.04e-05 ***
## density              -2.376e+01  2.417e+01  -0.983   0.3259
## pH                   -3.958e-01  2.152e-01  -1.839   0.0661 .
## sulphates             9.391e-01  1.292e-01   7.268 6.38e-13 ***
## alcohol               2.676e-01  2.930e-02   9.134  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6525 on 1266 degrees of freedom
## Multiple R-squared:  0.3536, Adjusted R-squared:  0.3479
## F-statistic: 62.95 on 11 and 1266 DF,  p-value: < 2.2e-16
```

Data Processing

```
# Plot of Cook's Distance
cooks <- cooks.distance(model1)
plot(cooks,type="h", lwd=3, col="steelblue3", ylab="Cook's Distance")
```

```
## Warning in title(...): conversion failure on 'Cook's Distance' in 'mbcsToSbcs':
## dot substituted for <e2>

## Warning in title(...): conversion failure on 'Cook's Distance' in 'mbcsToSbcs':
## dot substituted for <80>

## Warning in title(...): conversion failure on 'Cook's Distance' in 'mbcsToSbcs':
## dot substituted for <99>
```

```
abline(1,0,col="steelblue1")
```




```
# Identify possible outliers
outliers <- as.numeric(names(cooks)[(cooks > 0.04)])
outliers
```

```
## [1]   93  152 1236
```

```
# Remove possible outliers
training2 <- training[-c(152, 653, 1236),]


# Build second multiple linear regression model without possible outliers
model2<-lm(quality~., data=training2)
summary(model2)
```

```
##
## Call:
## lm(formula = quality ~ ., data = training2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.31876 -0.37080 -0.05709  0.45240  2.05296
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.768e+01  2.370e+01   1.168   0.2431
```

```
## fixed.acidity         4.730e-02  2.932e-02   1.613   0.1069
## volatile.acidity     -9.956e-01  1.327e-01  -7.505 1.16e-13 ***
## citric.acid          -1.831e-01  1.628e-01  -1.125   0.2609
## residual.sugar        1.607e-02  1.624e-02   0.989   0.3227
## chlorides            -1.889e+00  4.587e-01  -4.118 4.06e-05 ***
## free.sulfur.dioxide   5.561e-03  2.478e-03   2.245   0.0250 *
## total.sulfur.dioxide -3.647e-03  8.325e-04  -4.381 1.28e-05 ***
## density              -2.385e+01  2.420e+01  -0.985   0.3246
## pH                   -3.870e-01  2.155e-01  -1.795   0.0728 .
## sulphates             9.351e-01  1.294e-01   7.227 8.52e-13 ***
## alcohol               2.673e-01  2.932e-02   9.117  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.653 on 1263 degrees of freedom
## Multiple R-squared:  0.352,  Adjusted R-squared:  0.3464
## F-statistic: 62.38 on 11 and 1263 DF,  p-value: < 2.2e-16
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
# Display VIF
vif(model2)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##             7.763351             1.740910             3.036211
##       residual.sugar            chlorides  free.sulfur.dioxide
##             1.736779             1.521560             1.994665
## total.sulfur.dioxide              density                   pH
##             2.210685             6.400319             3.282185
##            sulphates              alcohol
##             1.444795             2.896054
```

```
# Compute Threshold for VIF
1/(1-summary(model2)$r.squared)
```

```
## [1] 1.543285
```

It appears that there is some multicollinearity between the predictors. This is because the VIF (Variance Inflation Factor) of most predictor variables exceed the threshold of 1.576901.

As such, the next step is to remove redundant or insignicant predictor variables to create a reduced model.

```
# Build third multiple linear regression model without possible outliers and redundant predictor variab
model3<-lm(quality~volatile.acidity+chlorides+total.sulfur.dioxide+sulphates+alcohol, data=training2)
summary(model3)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + chlorides + total.sulfur.dioxide +
##     sulphates + alcohol, data = training2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.33982 -0.39177 -0.05508  0.45413  2.06008
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.0903161  0.2314091  13.354  < 2e-16 ***
## volatile.acidity      -1.0989756  0.1080894 -10.167  < 2e-16 ***
## chlorides             -1.6974815  0.4274599  -3.971 7.56e-05 ***
## total.sulfur.dioxide -0.0025435  0.0005809  -4.379 1.29e-05 ***
## sulphates              0.9402204  0.1249998   7.522 1.02e-13 ***
## alcohol                0.2668657  0.0187413  14.239  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6585 on 1269 degrees of freedom
## Multiple R-squared:  0.3379, Adjusted R-squared:  0.3353
## F-statistic: 129.5 on 5 and 1269 DF,  p-value: < 2.2e-16
```

```
# Display VIF
vif(model3)
```

```
##     volatile.acidity            chlorides total.sulfur.dioxide
##             1.136381             1.299144             1.058504
##            sulphates              alcohol
##             1.325845             1.163236
```

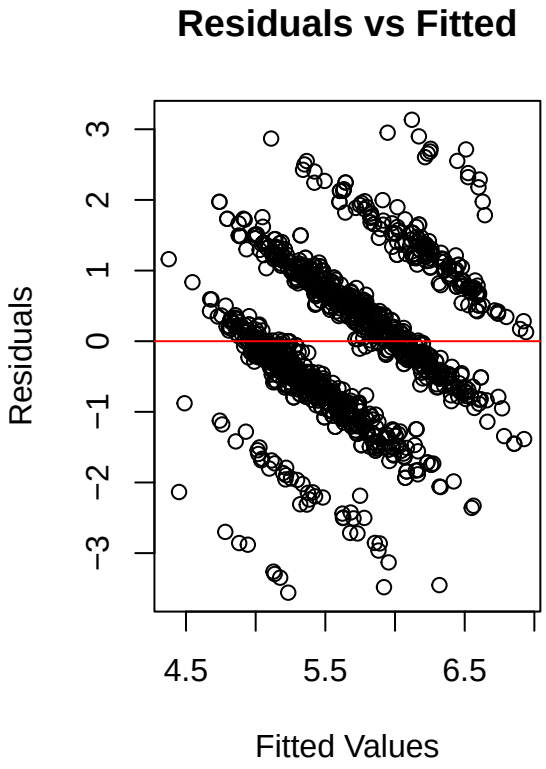
```
# Compute VIF Threshold
1/(1-summary(model3)$r.squared)
```

```
## [1] 1.510424
```

Using only the predictors with p-values less than 0.001 in model 2, I created model 3. Now the VIF of all predictors is less than the threshold, meaning multicollinearity is no longer an issue.

Of note that to better select predictor variables to include in the reduced model, Principal Component Analysis (PCA) can be done (but that in of itself can be a project so I simplified my method). Note that the p-values of the coefficient of each variable is based on a t-test and the result of significance is only such given all other predictors in the model. The most technically correct way to create a reduced model is to do PCA or other method of feature selection and then conduct a partial F-test to determine if the reduced model is better than the full model.

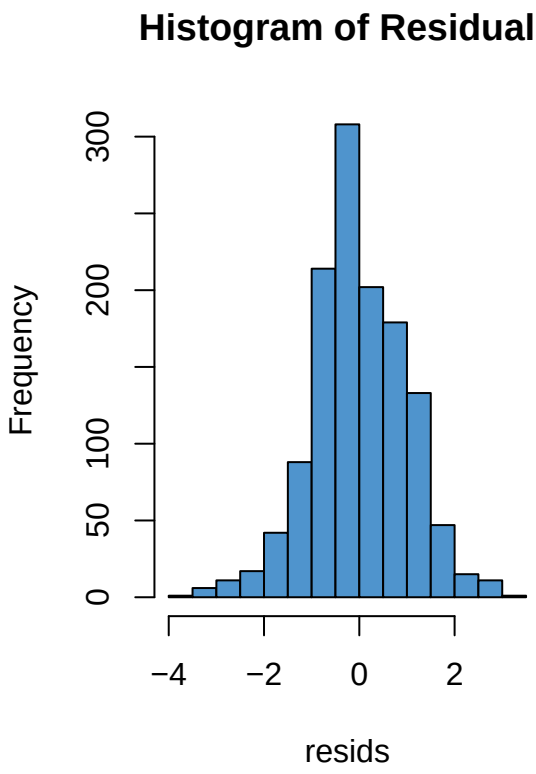
```
# Find standardized residuals
resids <- rstandard(model3)

par(mfrow=c(1,2))

plot(model2$fitted.values, resids, main="Residuals vs Fitted",
xlab="Fitted Values", ylab="Residuals")
abline(h=0, col="red")
```

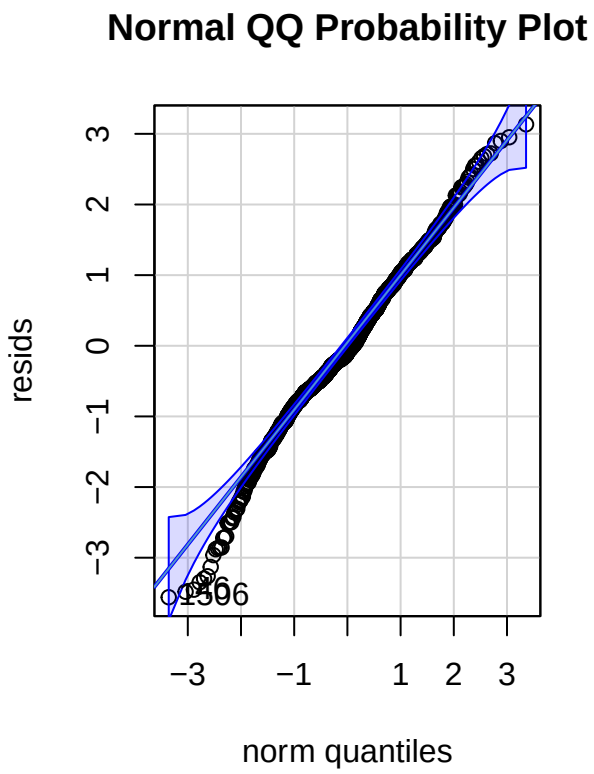



```
# Look at residuals plots
par(mfrow=c(1,2))

hist(resids, col="steelblue3", nclass=15, main="Histogram of Residuals")
qqPlot(resids, main="Normal QQ Probability Plot")
```

```
## 1506   46
## 1204   39
```

```
qqline(resids, col="steelblue3")
```





The distribution of the residuals appears normal, based on the histogram, and looks generally normal with some heavy tails.

Interestingly enough the residual vs fitted plot shows a significant pattern of residuals. This type of diagonal pattern is common when the dependent variable is discrete, as is the case in this dataset.

---

2. K-Nearest Neighbours Models

The classification model that is appropriate for our objective is one that is a multiclass classifier and supervised learning. It is multiclass because there is more than one class given the dependent variable, quality, is discrete from 3 to 8. This is supervised learning because the quality scores are known in the training dataset. As such, an appropriate model to use is the KNN model.

The k Nearest Neighbours Model is a non-parametric algorithm useful when classifying more than 2 classes. It does not result in a classifier equation, but rather classifies each data point based on its nearest neighbours. The variable k refers to the number of neighbours taken into consideration. The value of k is a hyperparameter that can be optimized.

Running The KNN Model

```
library(kknn)
```

```
##
## Attaching package: 'kknn'
```

```
## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```
library(tidyverse)
library(FNN)

# Put response in its own object
quality_response <- data %>% select(quality)

# Define response in each subset
response_train <- quality_response[-test_index, ]
response_test <- quality_response[test_index, ]

# define function to calculate RMSE
rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}

# Check different values of k

# For k=1
modelknn <- knn.reg(training, test=NULL, response_train, k = 1)
predictions <- as.integer(modelknn$pred)
rmse(response_train, predictions)
```

```
## [1] 0.6005996
```

```
# For k=3
modelknn <- knn.reg(training, test=NULL, response_train, k = 3)
predictions <- as.integer(modelknn$pred)
rmse(response_train, predictions)
```

```
## [1] 0.7032231
```

```
# For k=5
modelknn <- knn.reg(training, test=NULL, response_train, k = 5)
predictions <- as.integer(modelknn$pred)
rmse(response_train, predictions)
```

```
## [1] 0.7400884
```

```
# For k=10
modelknn <- knn.reg(training, test=NULL, response_train, k = 10)
predictions <- as.integer(modelknn$pred)
rmse(response_train, predictions)
```

```
## [1] 0.7802331
```

```
# For k=15
modelknn <- knn.reg(training, test=NULL, response_train, k = 15)
predictions <- as.integer(modelknn$pred)
rmse(response_train, predictions)
```

```
## [1] 0.8217505
```

```
# For k=30
modelknn <- knn.reg(training, test=NULL, response_train, k = 30)
predictions <- as.integer(modelknn$pred)
rmse(response_train, predictions)
```

```
## [1] 0.8712052
```

To make this model, I used the knn.reg function in the FNN library to make predictions for each row of observations based on the K Nearest Neighbours approach. K is a hyperparameter to be selected for the model. The value of k chosen indicates the number of neighbours used in the model to predict classification. Since there are 1278 rows of observations in the training set, the maximum possible value for k would be 1278. But that would not be a useful model, so I tested values of k up to 30 since the rule of thumb is the squareroot of the number of rows, which in this case is around 35.7.

In order to determine the best k to use, I created a function to check the RMSE accuracy. In this case the accuracy is in terms of the training data set itself. So comparing the predicted values with the known quality scores in the training set. The k value with the lowest RMSE is k=1. So we can use this value for our model.

At a k value of 1, the RMSE against the training dataset itself was lowest. A smaller k is more computationally cost effective. Larger k values have greater bias, and smaller k values have greater variance or noisiness.

```
# Our final KNN model
KNN_model <- knn.reg(training, test=NULL, response_train, k = 1)
head(KNN_model)
```

```
## $call
## knn.reg(train = training, test = NULL, y = response_train, k = 1)
##
## $k
## [1] 1
##
## $n
## [1] 1278
##
## $pred
##    [1] 5 5 6 5 5 5 6 6 5 5 5 5 5 5 5 6 5 5 5 5 5 5 5 5 6 6 6 5 5 5 6 6 5 5 5 6 5
##   [38] 5 5 5 5 5 5 6 5 5 6 5 5 5 5 6 5 5 5 5 5 6 5 5 6 6 5 5 5 5 5 6 6 5 5 5 6
##   [75] 6 5 5 7 5 5 5 6 6 6 5 5 5 5 6 5 5 6 5 5 5 6 5 5 5 5 5 5 5 6 4 5 5 5 5 5 5
##  [112] 6 6 5 5 5 6 5 6 6 6 6 6 5 5 5 5 5 5 5 5 6 5 5 5 5 5 5 5 6 6 6 5 5 6 5 5 5 5
##  [149] 6 6 5 5 5 5 5 5 5 6 7 7 5 6 6 7 7 4 5 7 6 6 6 5 5 5 5 5 6 5 5 5 6 5 6 5 5
##  [186] 6 5 6 5 5 6 6 6 6 6 6 6 5 6 5 7 7 7 5 5 6 6 6 5 6 5 5 6 6 6 5 5 5 6 5 6 6 7
##  [223] 6 4 6 5 5 6 7 5 6 5 7 5 5 5 7 5 5 7 5 5 5 6 5 5 6 5 6 6 5 6 5 6 5 6 5 6 6
##  [260] 5 6 7 6 7 6 6 6 6 6 7 6 6 6 6 6 6 6 6 5 7 6 6 6 6 6 7 6 6 6 5 6 6 5 7 6 6
##  [297] 7 5 5 7 6 7 6 6 5 5 6 7 8 6 6 5 6 6 6 5 5 6 5 6 7 5 6 6 5 6 6 6 6 6 6 6 6
##  [334] 7 6 5 5 6 6 5 5 6 5 6 5 6 5 7 5 6 7 6 5 6 7 5 7 6 6 5 6 6 6 6 6 7 7 6 5 6
```

```
##  [371] 6 6 6 6 6 7 5 7 5 5 7 5 5 5 6 6 7 6 5 7 6 5 5 6 5 6 7 7 6 6 5 5 6 6 5 7 7
##  [408] 6 5 7 6 5 6 7 7 7 7 7 6 6 7 6 6 7 7 7 5 7 5 7 5 7 5 5 5 5 5 5 6 7 6 6 5 5
##  [445] 7 6 6 7 6 5 5 5 5 7 5 6 6 6 5 5 6 5 6 6 6 5 6 6 6 6 6 5 6 5 5 6 5 5 5 5 6
##  [482] 7 6 7 5 6 6 5 5 5 5 5 6 6 5 6 5 6 6 7 6 5 5 5 6 5 5 6 6 5 5 6 5 5 5 5 5 6
##  [519] 6 6 5 5 5 5 6 5 5 5 5 5 5 5 5 6 6 5 6 6 5 7 4 5 5 6 6 6 6 5 6 6 6 6 5 5 6
##  [556] 5 6 5 6 6 5 5 5 5 5 6 5 5 5 6 6 6 6 6 5 6 5 5 5 5 5 6 7 5 5 5 6 6 6 5 5 5
##  [593] 4 5 7 5 5 6 5 5 5 5 5 5 6 5 5 5 5 5 6 6 6 5 5 5 5 5 6 6 6 6 5 5 5 5 6 5 5 6 5
##  [630] 5 5 6 5 5 6 5 5 5 5 5 5 5 6 6 6 5 5 6 6 5 5 6 5 6 5 6 5 5 7 7 7 5 6 6 6 5 4
##  [667] 5 6 6 5 5 5 5 5 6 6 6 3 6 5 6 5 7 7 6 6 6 7 5 5 5 6 5 5 5 5 6 6 7 6 6 6 5
##  [704] 7 5 5 5 5 7 6 6 6 5 6 7 6 8 5 6 6 5 6 6 5 6 6 6 5 6 5 5 6 7 7 4 5 7 7 7 5
##  [741] 5 5 6 7 5 6 6 6 6 5 6 5 5 5 6 7 5 5 5 5 5 5 6 6 6 6 7 6 6 6 7 7 7 7 7 7 7
##  [778] 6 6 6 5 5 6 7 6 5 5 6 6 5 6 5 6 6 6 6 5 5 5 5 5 5 7 7 6 7 6 7 7 7 7 7 7 5
##  [815] 8 7 5 5 6 7 7 6 6 6 6 5 7 7 6 5 5 7 7 7 5 6 7 8 6 6 6 5 5 5 6 5 5 6 5 5 5
##  [852] 4 6 6 7 5 6 7 6 7 6 6 5 6 6 5 6 5 6 5 5 6 6 5 5 6 6 6 5 6 7 6 7 5 6 6 6 6
##  [889] 5 5 6 6 6 5 5 7 5 5 5 6 6 6 6 6 6 5 8 7 7 6 5 5 6 5 7 7 6 6 6 5 5 6 6 7 5
##  [926] 7 6 7 6 6 5 5 5 5 5 5 6 5 6 6 6 6 5 6 5 7 6 6 6 5 5 6 7 5 5 6 5 6 5 5 5 6
##  [963] 5 5 7 7 7 7 7 6 7 7 6 6 6 6 6 6 6 5 7 6 5 5 7 5 5 5 5 5 6 5 6 6 5 6 5 5 5
## [1000] 5 6 5 6 6 6 5 5 5 5 4 6 6 5 6 6 6 6 6 6 6 5 6 6 6 6 5 6 5 6 5 6 6 6 5 5 3 6
## [1037] 5 5 7 4 6 5 5 4 6 5 5 5 5 6 6 6 6 5 5 6 6 6 6 6 5 6 6 5 6 6 5 6 5 5 5 6 6
## [1074] 6 6 6 7 4 6 5 5 5 5 5 6 5 5 6 5 5 6 5 5 5 6 5 6 6 5 5 5 5 6 6 5 5 5 5 5 6 6
## [1111] 6 6 5 6 5 5 5 6 5 5 5 6 6 6 6 5 6 6 6 5 5 5 5 5 5 4 6 6 6 6 5 6 6 5 6 6 6
## [1148] 5 6 4 5 7 6 5 6 5 6 7 7 7 5 6 5 5 7 6 6 5 7 5 7 5 5 7 5 5 7 5 7 4 5 3 5 5
## [1185] 5 5 5 6 6 5 6 5 6 5 6 7 5 6 5 6 6 6 5 6 7 4 5 5 5 5 6 6 6 5 6 5 6 5 5 6 5
## [1222] 6 6 6 5 5 5 6 7 5 6 6 7 6 6 6 6 5 6 6 5 6 5 6 5 5 5 5 5 6 6 5 6 6 6 6 5 6
## [1259] 6 6 6 6 5 5 5 5 5 7 6 7 6 7 5 6 6 6 6 6
##
## $residuals
##    [1]  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0 -1 -1  1  1  0  0  1  0  0
##   [25]  0 -1  0  0  1  0  0  0 -1  0  0  0  0  0 -1  0  0  0  0  0  1  0  0  0
##   [49]  0  1  0  1  0  0  0  0 -1 -1  0  0  0 -1 -1  0  0  0  0  1 -1  0  0  0
##   [73]  0  0 -1  0 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  1  0  0
##   [97]  1  1  0  0  0  0  0  1  1  0  0  0  1  0  0 -1 -1  0  0  0  0  0  0  0
##  [121]  0  0 -1  0  0  0  0  0  0 -1  0  0  0  0  0 -1  1 -1  0  0  0  0  0  0
##  [145]  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  1  0  0
##  [169]  0  0  0  0  1  0  0  0 -1  0  0  0  0 -1  0  1  0  0  0  1  1  0  0  0
##  [193]  0  0  0  0  0  0  1  0  0 -1  0  0  0  0 -1  0  0  0  0 -1 -1  1  0 -1
##  [217]  0 -1  0  1 -2  1  0  2  0  0  0  0  0  1  1  0  0  0  0  1  0  0  0 -1
##  [241]  1  1  0 -1  0  0  0  1 -1 -1  0  0  0  0  1  0  1  0  0  0  0  0  0  0
##  [265] -1 -1  0  0  0  0 -1  0  0 -1  1  1  0 -1  1 -1  0  0  0  0 -1  0  0  0
##  [289]  0  0 -1  0  0  0  0 -1 -1  0  0  0  0  0 -1 -1  0  0  0  0 -1  0  0  1
##  [313]  0  0 -1  1  1  0  1  1  1  0 -1  1  0  0  0  0  0  0  0 -1  0  0  0 -2  1
##  [337]  0 -1  1  0  0  0  0  0  0  1  0  0  0  1 -1  0  0  0  0  0 -1 -1  0  1
##  [361]  0  0 -1  2  0  0  0  0  0 -1  0  0  0  0  0  0  0  1  0  0  0 -2  0  0
##  [385] -1  0 -1  0  0 -2  0  1  0  0  0  0 -2  1 -1  0  0  0  0  1  1  0  0  0
##  [409]  1  1  0  0  0  0  0  0  0  0  0  0  0 -1  0 -1  0  0  0 -1 -2 -1  0 -1
##  [433]  0  0  0  0  0  0  0 -1 -1  0  0  0 -1  0  0  0 -1  0  1  1  0 -1  1  0
##  [457]  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  1 -1 -1  1  0  0  1  0  0
##  [481]  1  0  0  0  0  2  1  0  1  0  0  1  0  0 -1  0  0  0  0  0  0  0  1  0
##  [505]  0  0  0  0 -1  0  0  0  0  0  0  0  0  0 -1  0 -1  0  0  0  1  1  0  0
##  [529]  0  0  0  2  0  1  0  0  0 -1  0  0  0  1  0  0  0 -1 -1  1  0 -1  0 -1
##  [553]  0  0  0  0  0  0  0 -1  0  0  0  0  0 -1  0  0  0  0  0  0 -1  0  1  0
##  [577]  1 -1  0  0  0  0 -1  0  0  0 -1  0 -1  0  0  0  0  0 -1  0  0 -1  0  0
##  [601]  0  0  0  1 -1  0  0  0  0  1 -1  0  1  0  0  0  0  0  0  0  0  0  1  0
##  [625]  0  1  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0 -1  0  0  0  1 -1  0
```

```
##  [649]  0  0  1  1 -1  0  1  1  1  0  0  0  0  0 -1  0  0  0  0  0 -1  0  0  0
##  [673]  0  0  1  2  0  1  0 -1 -1  0  0  0 -1  1 -1 -1  0  0  0  0  0  0  0  0
##  [697]  0  0  0  1  1  0  0 -1  0  0  0  0 -1  0  0  0  0 -2  0  1 -1 -1  0  0
##  [721]  0  0  0  0  0  0  0  0 -1  0  1  0  0  0 -1  0  0  0  0  0  0  1  0 -1
##  [745]  1  0  0 -1  0  0  0  1  1  0  1 -1 -1  0  0  0  1  0  0  0  1 -1  0  1
##  [769]  1  1  0  0  0  0  0  0  0  0  0  1  0  0  0 -1  0  0  1  0  1  0 -1  0
##  [793] -1  0  1  0  0  0  1  0  0  1  0  0  0 -1  1  0  0  0  0  0  0  0 -1 -1
##  [817]  0  1  0 -1  0  0  0  0  0  0 -1  0  0  1  1  0  0  0  0  0 -1 -1 -1  0
##  [841] -1  1  2  1  0  1  0  0  1  0  0  1  0  0  0  0  1  0  0  1  0  0  1  1
##  [865]  1  0  1  0  0  1  0  1  0  0  0  1  0  0  1  0  0  0  1  1  0  1  0  0
##  [889]  0  0  0  0  0  0  1  0  0  1  1  0  0  0  0  0  0  0  0  0  0 -1  0  0  1
##  [913]  0  0  0  0  0  0  0  0  1  0  0 -1  1  0  0  0  0  0  0  0  2  1  0  0  0
##  [937] -1  1  0  0  0  0  1  0 -1  0  0  0 -1  0  0  0 -2  0 -1  0  0  1  0  1
##  [961]  1  0  1  1  0  1  0  0  0 -1  0  0  0  0  0  0  0  0  0  1 -1  0  0  0
##  [985]  0  0  1  0  0 -1  0 -1  0  0 -1 -1  0  0  1  0 -1  0  0  0  0  0  0  0
## [1009]  0  2  0  0  0 -2 -1  0  0  0  0  1 -1 -1  0  0 -1  0  1  0  1 -1  0 -1
## [1033]  0  1  1  0  0  0 -1 -1 -1  0  0  0 -1  0  1  0  1  0  0  0  0  1  0  0
## [1057] -1  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0 -1  1 -1  0
## [1081]  0  0  1  0  0  0  0  0  1  0  0  1 -1  0  0  0 -1 -2  0  0  1  0 -1  0
## [1105]  0  0  0  0 -1 -1  0 -1  0 -1  1  0  0  1  0  0  1  2  0  1  0  2  0  0
## [1129]  0  0  0  0  0  0  0  0  0  0  0 -1  0 -1 -1  1  0  0  0  0 -1  1  1  0
## [1153]  0  0  0  0 -1  1  0  0  0  0  1  0  0  0  0  0  0 -1  0 -2  0 -1  0  0
## [1177]  0  0  0 -1  0  1  0 -1  0 -1  0 -1 -1  0  0  0 -1  0  1 -1  0  0  1  0
## [1201] -1 -1  0  0 -1 -1  1  0  1  0  0  0  0  0  0  0  0 -1  0 -1  1 -1  0  0
## [1225]  1  1  0  0  0  1 -1  0  0  0  0  1 -1  0 -1  2  0  1  0  0  0  0  0  0
## [1249]  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 -1 -1 -1
## [1273]  1  0  0  0 -1  0
##
## $PRESS
## [1] 461
```

RESULTS

In this section, we will finally determine the accuracy of each model against "new" data, ie. the test dataset. This is meant to determine which model is better suited for our objective of predicting wine quality scores.

Multivariate Regression Model RMSE Evaluation

The rmse function has been defined earlier and we will use it here again.

```
# KNN model used to predict test dataset
predicted_reg <- predict(model3, test)

# Calculate RMSE against test dataset
reg_rmse <- rmse(response_test, predicted_reg)
reg_rmse
```

```
## [1] 0.6252424
```

The RMSE of the chosen multivariate regression model (which indicates true accuracy because it is compared to test data) is 0.625. This indicates moderately good accuracy.

```
#Compare RMSE to other regression models

# Model 1
predicted_reg1 <- predict(model1, test)
reg_rmse1 <- rmse(response_test, predicted_reg1)
reg_rmse1
```

```
## [1] 0.6341841
```

```
# Model 2
predicted_reg2 <- predict(model2, test)
reg_rmse2 <- rmse(response_test, predicted_reg2)
reg_rmse2
```

```
## [1] 0.6341495
```

Compared to the other multivariate regression models, Model 3 is indeed the best regression model in terms of RMSE, although the improvement is quite minor.

K Nearest Neighbours Model RMSE Evaluation

```
# KNN model used to predict test dataset
KNN_model2 <- knn.reg(train=training, test=test, y=response_train, k = 1)
predicted_knn <- as.integer(KNN_model2$pred)

# Calculate RMSE against test dataset
knn_rmse <- rmse(response_test, predicted_knn)
knn_rmse
```

```
## [1] 0.585388
```

The RMSE of the chosen KNN model (true accuracy because compared to test data) is 0.585. As compared to the validation accuracy of 0.60 it is actually lower, although accuracy against a the test dataset is expected to be lower than against the training dataset itself.

```
# Create RMSE table of RMSE results, starting with regression model
rmsetable <- data.frame(Model="Multivariate Regression Model", RMSE = reg_rmse)

# Add KNN model RMSE results
rmsetable <- bind_rows(rmsetable, data_frame(Model="K Nearest Neighbours Model", RMSE = knn_rmse))
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
# Load RMSE table
rmsetable %>% knitr::kable()
```

Model	RMSE
Multivariate Regression Model	0.6252424
K Nearest Neighbours Model	0.5853880

CONCLUSION

Using RMSE for comparison of models, we can see that the better model with the lower RMSE is the K Nearest Neighbours Model. The KNN model had a RMSE of 0.5853880 compared to the multivariate regression model with an RMSE of 0.6252424.

In our preliminary analyses of the red wine dataset, we discovered that there are 11 quantitative continuous predictors for one discrete quantitative response variable, quality. The data was initially processed to be split into 80% training and 20% test data sets. This is in consideration of the bias variance tradeoff, where too small of a training set would mean greater bias but too large would mean greater variance in terms of predicting on new data (represented by the test set). We determined that a multivariate regression model would be appropriate for our objective of predicting response quality scores with given predictors. Another model we explored was the KNN model because it is suitable for multiclass supervised learning.

To reflect on this result, I believe that the RMSE of the multivariate regression model could further be improved, given more time to explore feature selection methods such as principal component analysis to better select which predictors to include. A partial F test could then be done to determine if the reduced model is significantly better. Overall, two models have been built to fulfill our objective of predicting wine quality scores.

So perhaps, instead of relying on subjective judgement of wine quality, it is possible to predict wine quality based on objective quantitative measures in wine chemistry such asfixed acidity, volatile acidity, citric acid, residual sugar, chlorides, sulfur dioxide, density, pH, sulphates, and alcohol. In terms of potential impact, the multivariate regression and KNN models have shown some moderate promise for predicting wine quality, given these predictors. For future work, one can look into exploring other predictors such as age of wine, brand, winery location, and more to see how they affect wine quality and aid in its prediction.

"Life is too short to drink bad wine." - Anonymous
