DATA115 Final Project

Alex Hunt

12/15/2021

Introduction

The wine quality dataset from the UCI Machine Learning Repository was chosen because of how well it would work with a classification model without too much data manipulation. With 11 predictor variables and one outcome variable, quality, this data can be put into a supervised machine learning model such as multiple linear regression or classification. The data can be put into a training set to improve the accuracy of classifying wine quality.

Included in the repository was two datasets, white wine with 4890 observations and red wine with 1599 observations. For this project, I wanted to explore the differences between them such as which ones are usually ranked higher and if they both has the same correlating variables.

The main question for this project is finding out if the wine quality can be accurately predicted using a supervised machine learning model. My goal is to be able to train a model to be at least 70% accurate in determining wine quality. With over 70% accuracy, I would be able answer the main question, that is, there is enough correlation within the data to predict the wine's quality given 11 variables. Depending on how similar the red and white wines are, I plan to only train the model based on one dataset, preferably the white wine as there are over 3 times as many observations than the red wine. If the wines end up being similar through exploratory data analysis, I plan to use the model on the other wine type to find out if red and white wine have the same correlation within the data.

*Functions were re-ran so the classification results seen may not be the ones discussed.

Data processing

```
library(tidyverse)
```

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

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

```
white_df <- read_delim("/Users/alexhunt/Downloads/Data115 Final Project/winequality-white.csv",
    delim = ";", escape_double = FALSE, trim_ws = TRUE)
```

```
## Rows: 4898 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
red_df <- read_delim("/Users/alexhunt/Downloads/Data115 Final Project/winequality-red.csv",
    delim = ";", escape_double = FALSE, trim_ws = TRUE)
```

```
## Rows: 1599 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
view(white_df)
view(red_df)
summary(white_df)
```

```
##  fixed acidity    volatile acidity  citric acid     residual sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free sulfur dioxide total sulfur dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH          sulphates         alcohol         quality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

```
summary(red_df)
```

```
##  fixed acidity   volatile acidity  citric acid    residual sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    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
##        pH          sulphates         alcohol         quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
```

```
sum(is.na(white_df))
```

```
## [1] 0
```

```
sum(is.na(red_df))
```

```
## [1] 0
```

Looking at the summary, there does seem to be a few outliers, however, I will need to conduct more analysis to see if they have any high leverage or influence on the data using residual plots. For data cleaning and processing, I looked for any missing values and found none in both datasets. Without any problems with the data, I can now do some EDA to identify correlations and trends in the data.

The only problem that I encountered when processing the data was that the files had a semi-colon delimiter instead of a comma. This was an easy fix once I found out the problem. Instead of trying to read the files as a csv, I ran the read_delim function.

Exploratory data analysis

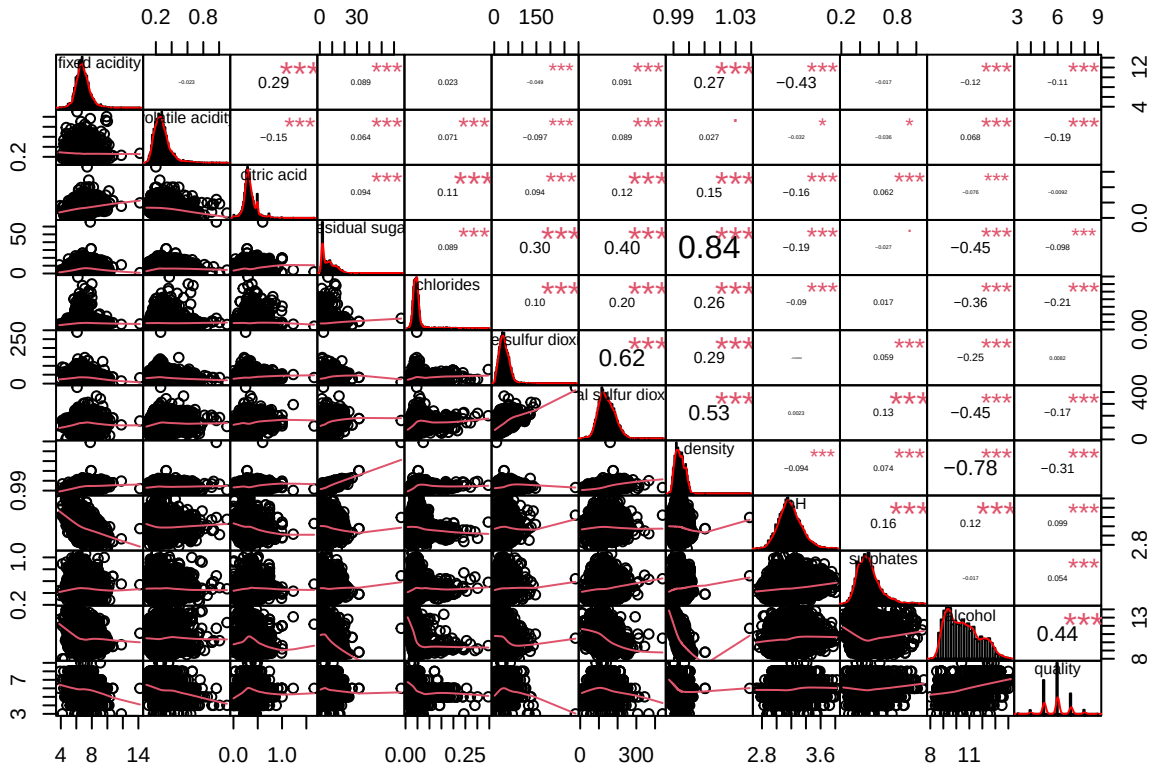
```
library(ggcorrplot)
library(PerformanceAnalytics)
```

```
## Loading required package: xts
```

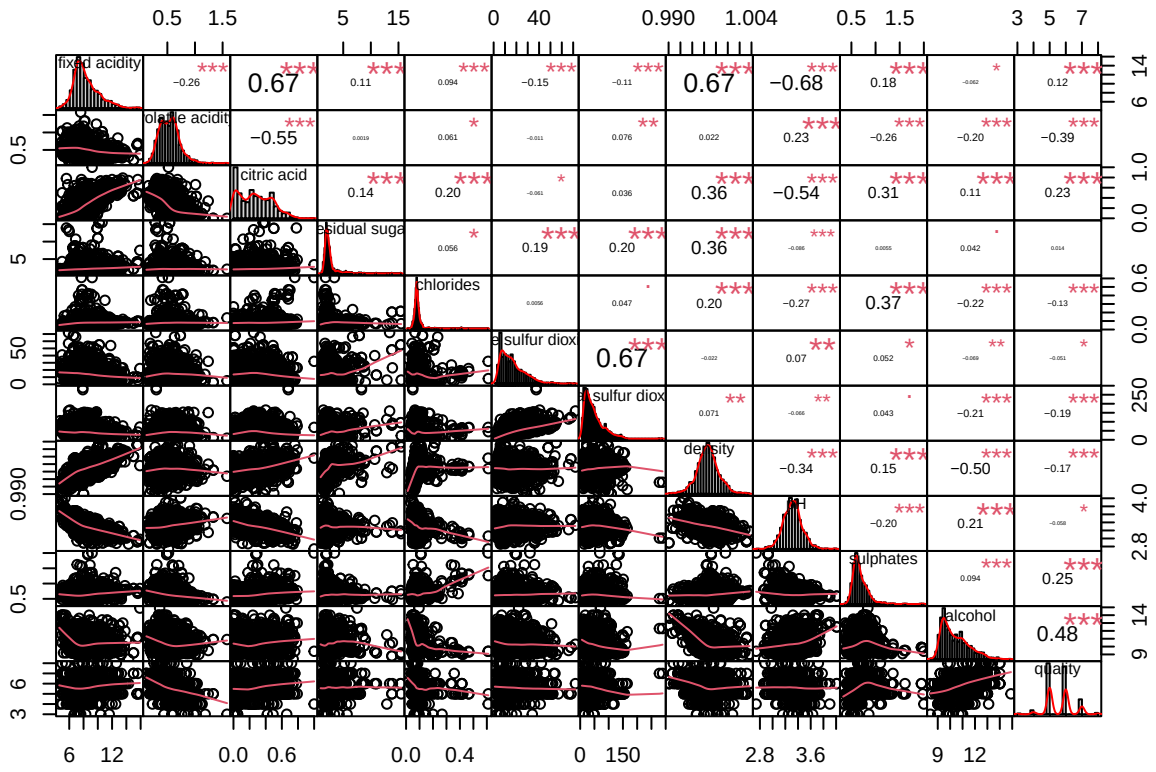
```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

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

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

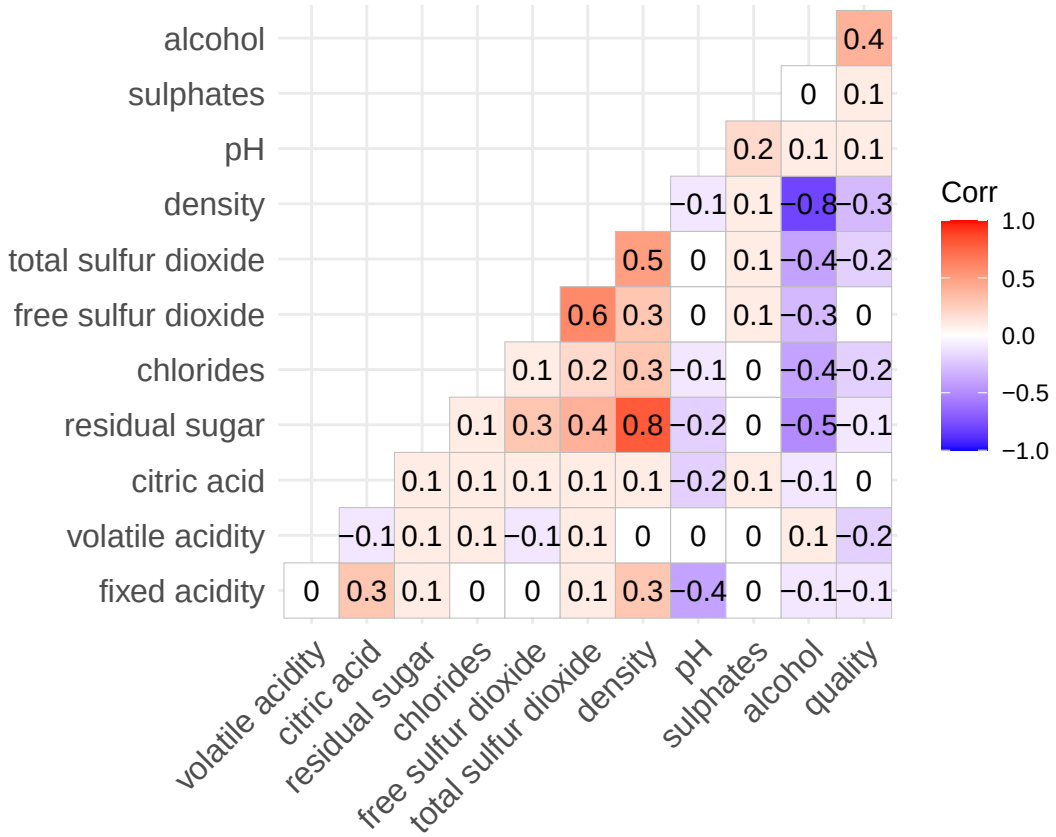
```
chart.Correlation(white_df, hist = TRUE)
```



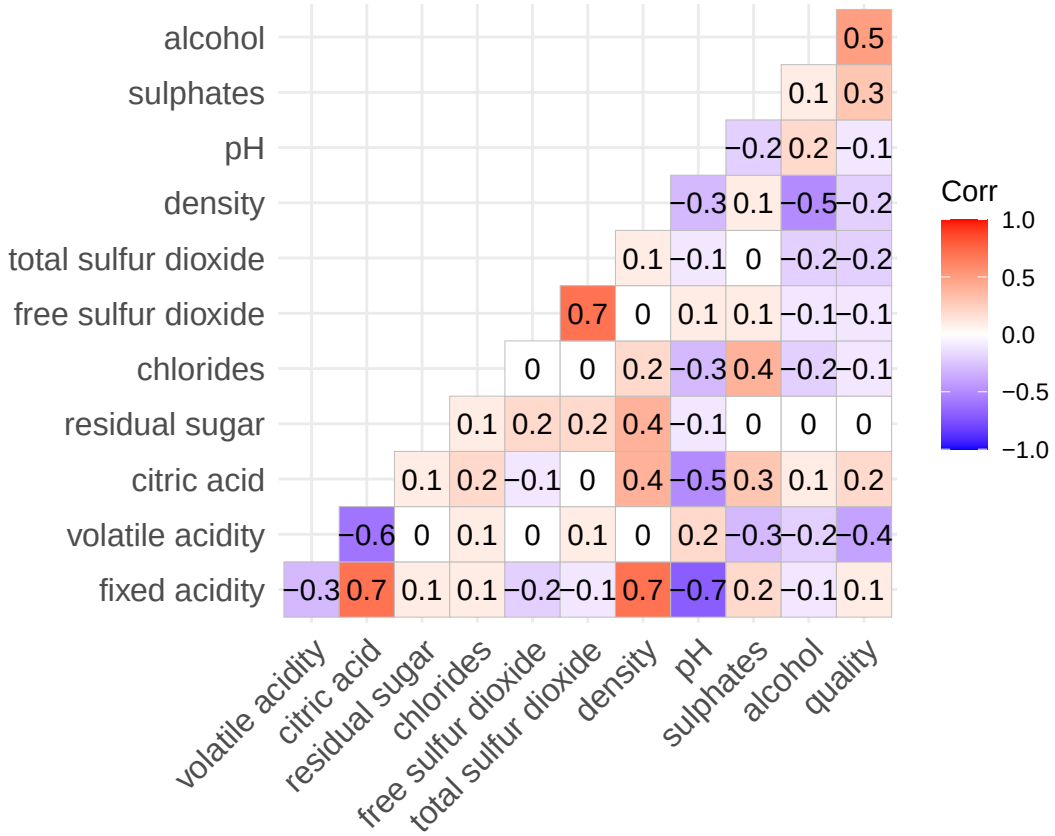

```
chart.Correlation(red_df, hist = TRUE)
```




```
corr <- round(cor(white_df), 1)
ggcorrplot(corr, lab = T, type = "lower")
```



```
corr2 <- round(cor(red_df), 1)
ggcorrplot(corr2, lab = T, type = "lower")
```

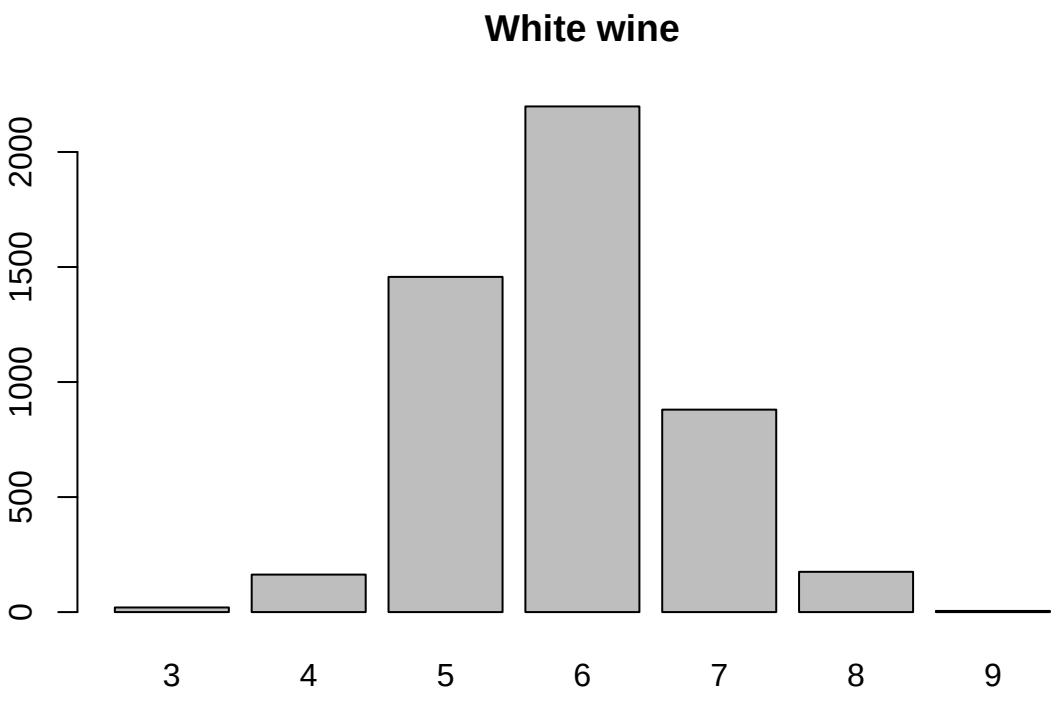


To explore the data, I wanted to know the distribution of the data using histograms, the correlations between variables, and scatterplots to identify any regression lines. I learned that I could use the PerformanceAnalytics library to plot histograms with included density curves, scatterplots, correlations, and the stars represent the p-value in one plot. More stars represent a more significant correlation whereas no stars show that the correlation is not significant. I also included a correlation matrix to get a better view of the correlation between variables.

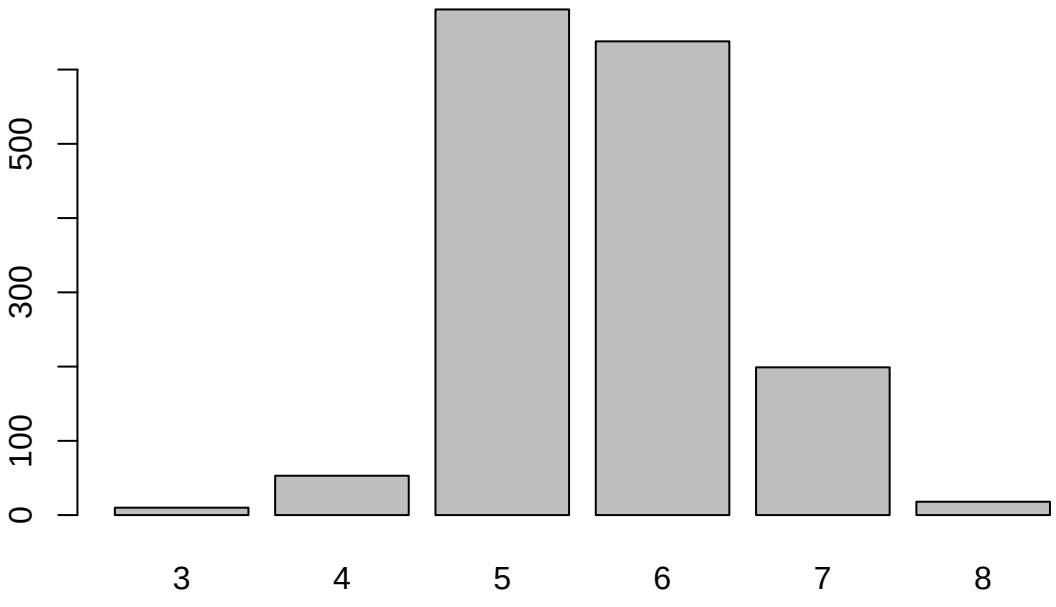
The main takeaway here is that alcohol has the greatest correlation with quality for both red and white wine. However, the second greatest correlation between white wine quality is density which also has a very substantial correlation with alcohol. With red wine, the second greatest correlation is with volatile acidity. because of the vast difference between observations for red and white wines, its hard to get an accurate difference between them.

Other observations about the data include how outliers are almost all on the larger side and with no significant p-values for free sulfur dioxide and citric acid in white wine, and residual sugar in red wine. This indicates that these columns can be removed when creating a regression or classification model and may improve the prediction accuracy.

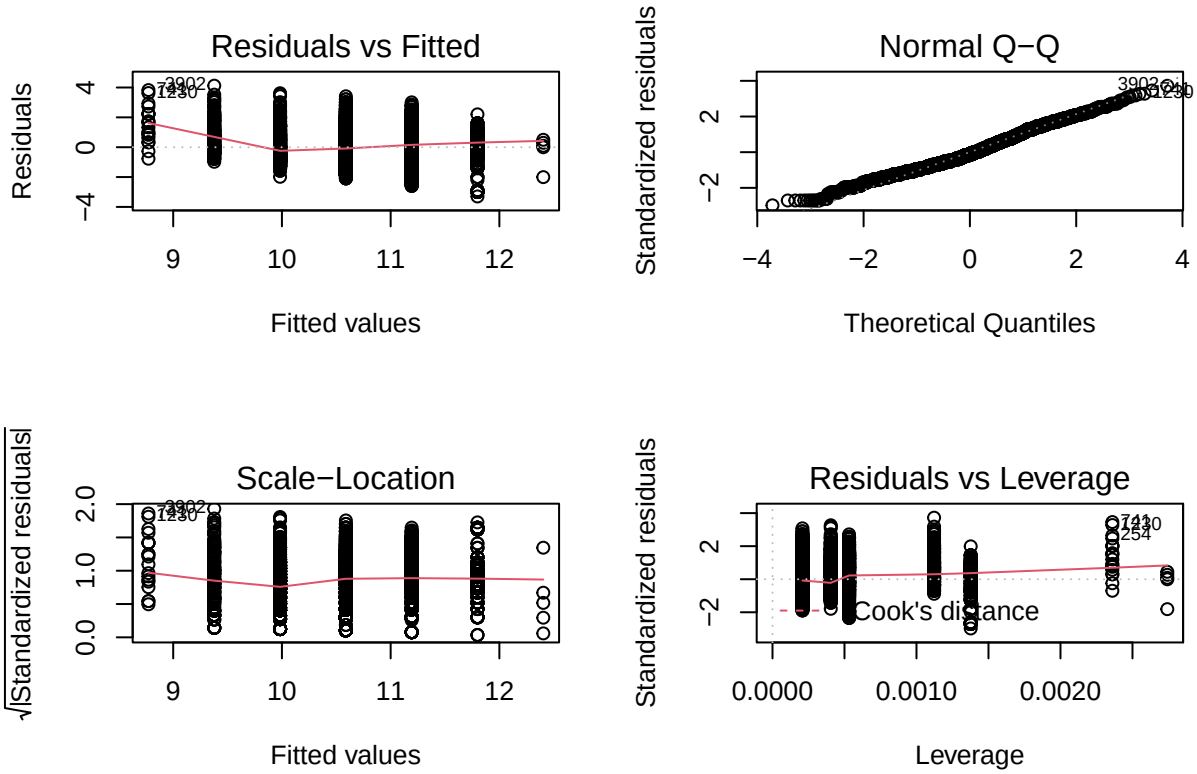
```
barplot(table(white_df$quality), main = "White wine")
```




```
barplot(table(red_df$quality), main = "Red wine")
```




```
par(mfrow=c(2,2))
fit = lm(white_df$alcohol ~ white_df$quality)
plot(fit)
```



With these plots, I wanted to explore the possibility of doing a linear regression model. However, in the previous plots, the quality histogram was not normally distributed, there were multiple peaks. I plotted a bar graph to get a better view and found that quality seems to have class based values, that is, they fall into either 6 or 7 categories. For linear regression, the output variable should have many more values (continuous) and be normally distributed. To make sure regression was out of the question, I plotted the residuals and found that they were not randomly distributed, which would indicate that the data works for linear regression.

The bar graphs answer the question of which wine usually ranks higher in that white wine is more distributed whereas red wine is centered around 5 and 6. Both wines have a low value of 3 but white wine is the only one with a 9 ranked wine.

Because the data is not set up for any significant regression model, the data will have to be updated to work with classification. In order to do this, I would need two or three categories such as good, bad, or normal/mid tier. The red wine graph works well for having two classes, this would work for binary classification. However, because the white wine has more values and is distributed more evenly, 3 classes of mostly equal size can be made.

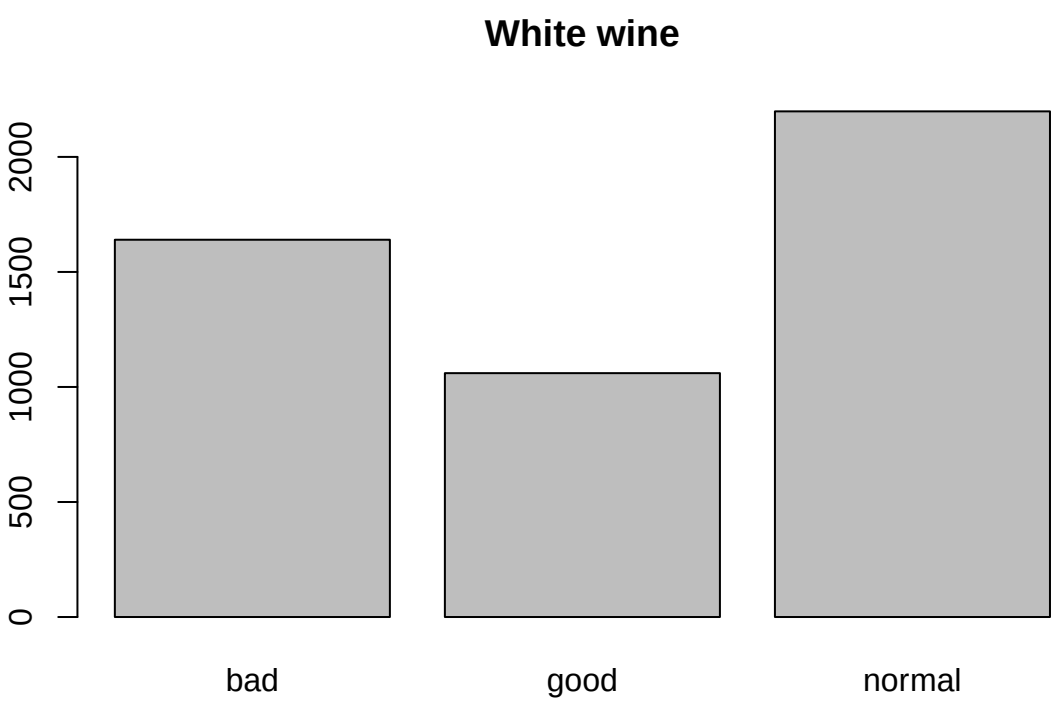
Model creation

After the EDA, it was decided to create 3 classes, good wine, bad wine, and normal wine on the white wine. Values over 6 would be good, below 6 would be bad, and because there were many 6 ranked wines, 6 would be normal. I researched classification algorithms that would work with this type of data (Multi-Class Classification) and decided to implement two of them. The first one being k nearest neighbor (kNN) because it is easy to use and I don't have that large of a dataset, only 4898 observations and 11 dimensions. For the second model, I decided to use the random forest algorithm due to its increased accuracy.

K-Nearest Neighbor (easy to use): Attempts to group data into classes and predicts based on data similarity or how close they are. This algorithm is easy to implement and no assumptions about the data have to be made. It also works well for multi-class data which is what I am working with.

Random Forest (better accuracy): Uses multiple decision trees to predict values based on the predictor variables. This algorithm takes the average of all trees to provide greater accuracy. The benefit of using random forest/decison trees is that it works with categorical data such as 'good' or 'bad' and has some of the best accuracy among classification algorithms.

```
#Creates a column and fills in one of the 3 classes based on if the wine quality is greater than 6, les
#white_df$rank <- ifelse(white_df$quality < 6, 'bad', 'good')
#white_df$rank[white_df$quality == 6] <- 'normal'
#white_df$rank <- as.factor(white_df$rank)

white_df$quality[which(white_df$quality %in% c(3, 4, 5))] = "bad"
white_df$quality[which(white_df$quality %in% c(7, 8, 9))] = "good"
white_df$quality[which(white_df$quality %in% c(6))] = "normal"
barplot(table(white_df$quality), main = "White wine")
```



Initially, I added the classes in a new column. However, I faced problems when creating the kNN model as the class did not have the same length as the training set. I then decided to just replace the quality with the class description. When using the str() function, I found that the rank column and newly updated quality columns had different data structures which was causing the problem. I then plotted the class sizes on a barplot and found that 'good' was about half the size of the 'normal' class. These differences were good to test out the classification model as it is more challenging than having only two classes.

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```r
training_set <- createDataPartition(y = white_df$quality, p = 0.7, list = FALSE)
train <- white_df[training_set ,]
test <- white_df[-training_set,]

nrow(train)
```

```
## [1] 3429
```

```r
nrow(test)
```

```
## [1] 1469
```

```r
#standardization
pre_processed <- preProcess(train, method = c("center", "scale"))
train_standard <- predict(pre_processed, train)
test_standard <- predict(pre_processed, test)
```

Using the caret library, I split the data, 70% in training and 30% in testing. With large datasets, a higher percentage can be allocated to the training set. I found that the 70/30 split is one of the more common ones. Also, according to the caret documentation, the standard data partition method is stratified sampling.

The preProcess function is used to normalize all the predictor variables, that is, make the standard deviation 0 and the mean 1. Scaling the data in this way makes all the values similar to each other. This is necessary so no value is affected by the magnitude of another and remove any bias towards larger numbers.

K-Nearest Neighbor

```r
library(class)
train_wine <- train_standard[ ,-12]
test_wine <- test_standard[ ,-12]

class <- train_standard[, 12, drop = TRUE]
results <- test_standard[, 12, drop = TRUE]

kNN <- knn( train_wine, test_wine, class, k = 25)
confusionMatrix(as.factor(results), kNN)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good normal
##     bad    295   32    165
##     good    14  152    152
##     normal 170   92    397
##
## Overall Statistics
##
##                Accuracy : 0.5745
##                  95% CI : (0.5488, 0.6)
##     No Information Rate : 0.486
```

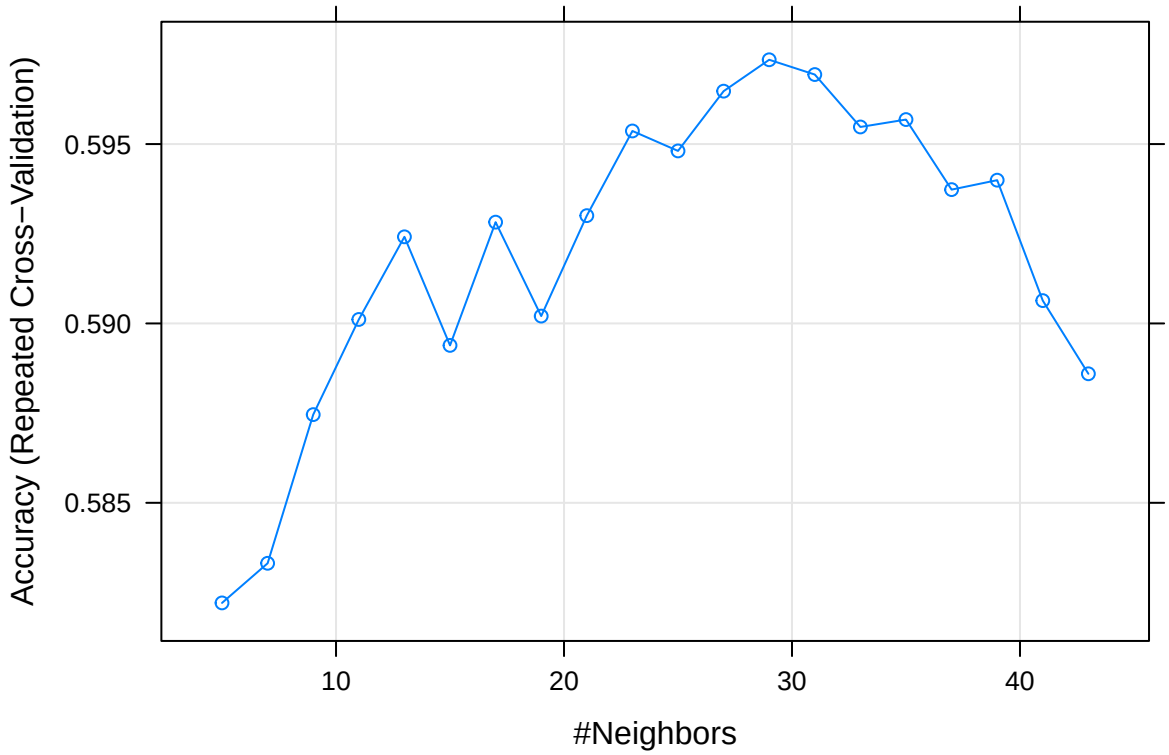
```
##     P-Value [Acc > NIR] : 6.596e-12
##
##                   Kappa : 0.3269
##
##  Mcnemar's Test P-Value : 6.935e-05
##
## Statistics by Class:
##
##                      Class: bad Class: good Class: normal
## Sensitivity              0.6159      0.5507        0.5560
## Specificity              0.8010      0.8609        0.6530
## Pos Pred Value           0.5996      0.4780        0.6024
## Neg Pred Value           0.8117      0.8923        0.6086
## Prevalence               0.3261      0.1879        0.4860
## Detection Rate           0.2008      0.1035        0.2703
## Detection Prevalence     0.3349      0.2165        0.4486
## Balanced Accuracy        0.7084      0.7058        0.6045
```

```
control = trainControl(method = "repeatedcv", repeats = 10, classProbs = TRUE, summaryFunction = defaul
optimalK <- train(quality ~. , data = train_standard, method = "knn", trControl = control, preProcess =
optimalK
```

```
## k-Nearest Neighbors
##
## 3429 samples
##   11 predictor
##    3 classes: 'bad', 'good', 'normal'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 3085, 3086, 3086, 3086, 3086, 3088, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.5822091  0.3455193
##    7  0.5833134  0.3449119
##    9  0.5874585  0.3502584
##   11  0.5901103  0.3530203
##   13  0.5924144  0.3555831
##   15  0.5893881  0.3497374
##   17  0.5928229  0.3547534
##   19  0.5902047  0.3498270
##   21  0.5930049  0.3530975
##   23  0.5953646  0.3570260
##   25  0.5948086  0.3552109
##   27  0.5964746  0.3572024
##   29  0.5973501  0.3582767
##   31  0.5969380  0.3569954
##   33  0.5954765  0.3541349
##   35  0.5956831  0.3544417
##   37  0.5937301  0.3511022
##   39  0.5939931  0.3507788
##   41  0.5906364  0.3449075
##   43  0.5885963  0.3412785
```

```
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 29.
```

```
plot(optimalK)
```



Using the control and train functions from the caret library, I was able to find the optimal neighbor size that would give me the greatest accuracy, 25, which provided a 60% accuracy in classifying wine quality. The repeated k-fold cross validation is a way to improve the estimated performance of a machine learning model. I found that 10 is a good default for the number of repeats and a greater tune length provided a larger array of tested k values. Because of the processing time, I only went up to 20.

Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

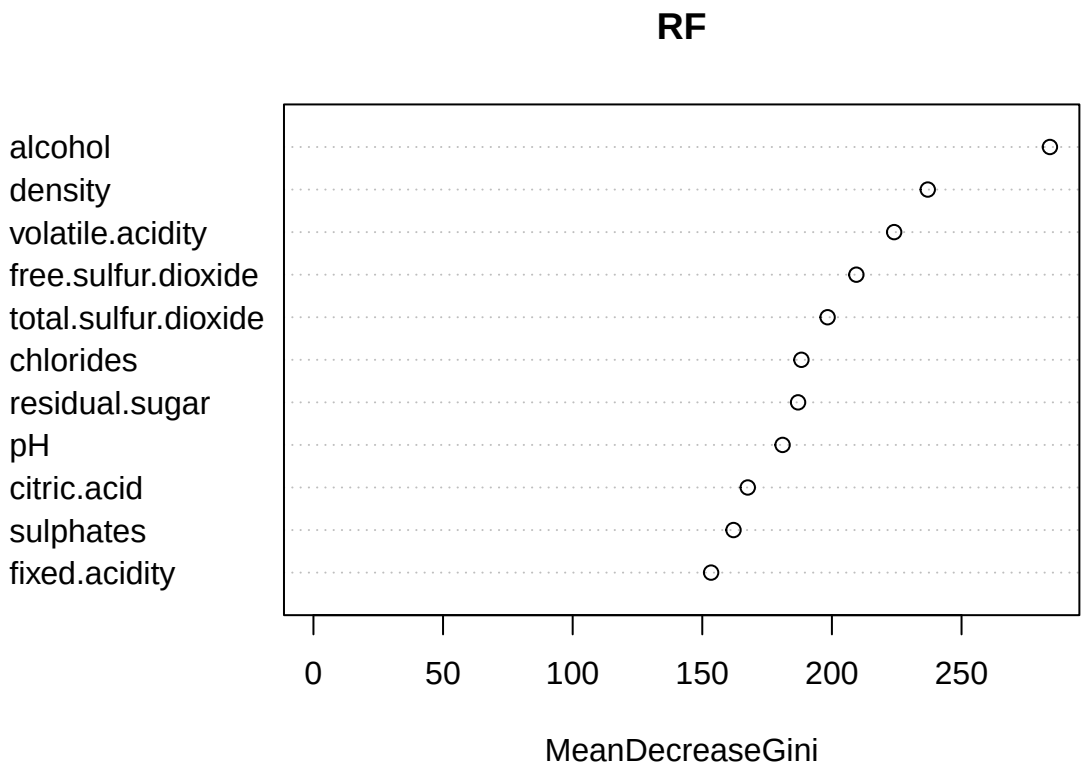
```
wineRF <- white_df
names(wineRF) <- make.names(names(wineRF))
wineRF$quality <- factor(wineRF$quality)

trainRF <- wineRF[training_set, ]
testRF <- wineRF[-training_set, ]

RF <- randomForest(quality ~ ., trainRF)
RF
```

```
##
## Call:
##  randomForest(formula = quality ~ ., data = trainRF)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 28.2%
## Confusion matrix:
##        bad good normal class.error
## bad    830   14    304   0.2770035
## good    15  455    272   0.3867925
## normal 237  125   1177   0.2352177
```

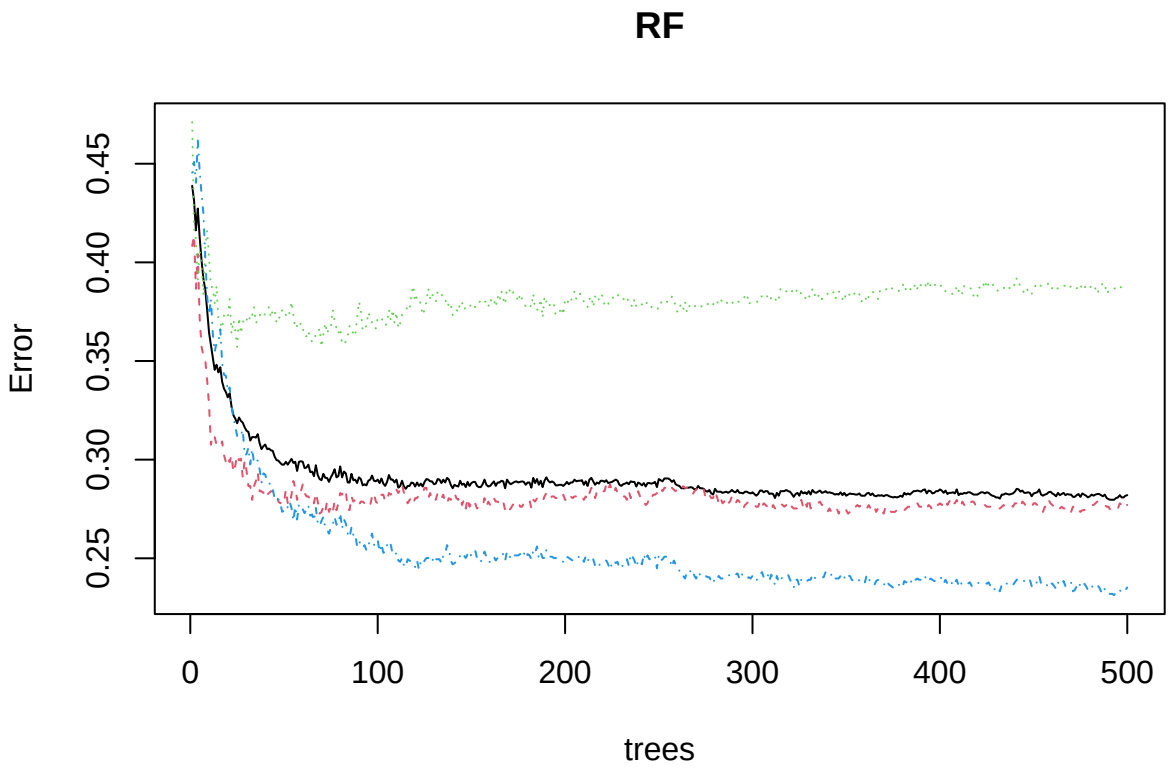
```
RFPrediction <- predict(RF, testRF)
table(RFPrediction, testRF$quality)
```

```
##
## RFPrediction bad good normal
##       bad    346    9    108
##       good     8  195     54
##       normal 138  114    497
```

```
varImpPlot(RF)
```




```
plot(RF)
```



To make the data work with the randomForest library, I had to remove spaces in the column names. At first, I used clean_names() which included underscores but that didn't work. I found out that the standard method is to add dots which I used on a copied data frame. I then had to create the training and testing

data and plugged it into the model. I used the predict function to test out the accuracy on the test data and found that the accuracy was 72.8%. I found that by adding the correct results divided by all of the test data - (341 + 210 + 519) / 1469 = 72.8%. This is a 12.8% increase over the kNN algorithm.

I also plotted out the variable importance and found that alcohol was the main factor for classifying wine quality. This plot shows the mean decrease in node impurity, it represents how important that variable is for estimation across all trees in the forest. In this case, there were 500 trees. When plotting the error chart, we can see that the average estimated error rate flattens out at the 200 tree mark.

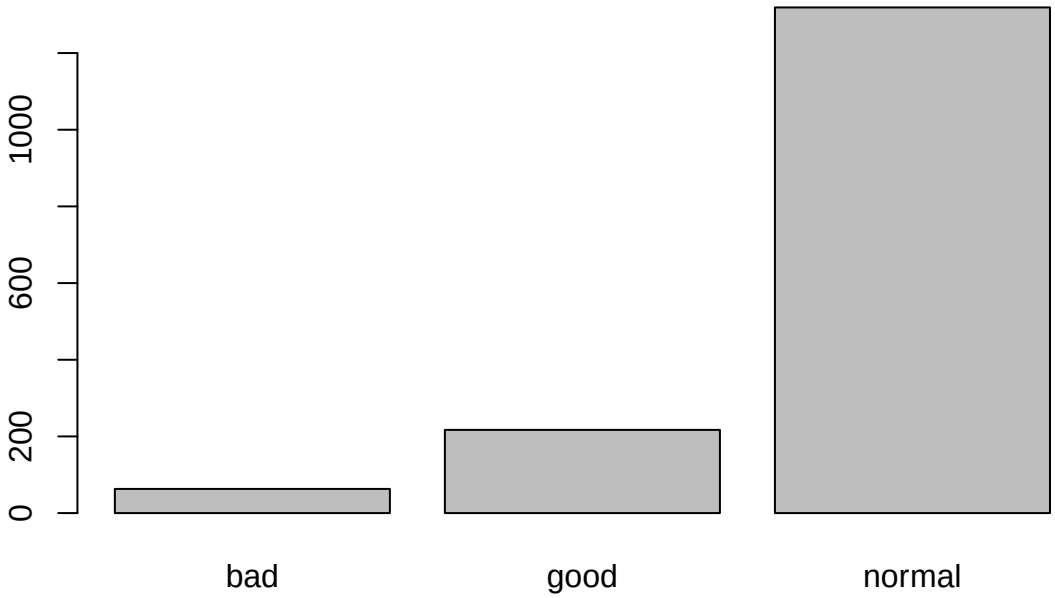
Testing the red wine

This next section will test the random forest model, which provided a 72.8% accuracy on the white wine data, using the entire red wine dataset. If the prediction accuracy is close to 72.8 then both wines are similar in how their predictor variables correlate with quality. If the accuracy is off by a substantial amount, in this case, a difference of 10%, then red and white wines have different variables that determine quality.

```
red_df$quality[which(red_df$quality %in% c(3, 4))] = "bad"
red_df$quality[which(red_df$quality %in% c(7, 8))] = "good"
red_df$quality[which(red_df$quality %in% c(5, 6))] = "normal"

#Provides the same prediction accuracy
#red_df$quality[which(red_df$quality %in% c(3, 4, 5))] = "bad"
#red_df$quality[which(red_df$quality %in% c(7, 8, 9))] = "good"
#red_df$quality[which(red_df$quality %in% c(6))] = "normal"

names(red_df) <- make.names(names(red_df))
red_df$quality <- factor(red_df$quality)

RedRF_prediction <- predict(RF, red_df)
table(RedRF_prediction, red_df$quality)
```

```
##
## RedRF_prediction  bad good normal
##           bad      61  136   1174
##           good      0    3      1
##           normal    2   78    144
```

```
barplot(table(red_df$quality), main = "Red wine")
```



(62 + 7 + 138)/1599 = 12.9% accuracy in predicting red wine quality. The red wine had an accuracy rating of 59.9% less than the white wine which proves that red and white wines are not similar in terms of their chemical properties and how they correlate to quality. However, one problem with this test is the distribution of red wine qualities. Because most wines are graded either 5 or 6, its hard to get an even distribution of good, bad, and normal wines. In my test, I did two different rankings, the one based on the white wine (commented out) and one based on the first two, middle, and last two. Both tests generated the same accuracy. With a distribution more like the white wine, I would expect a better prediction percentage but probably not anywhere close to 72.8%.

Conclusion

With a 72.8% prediction accuracy, 2.8% over the threshold I set at the beginning, the big question is answered in that wine quality can be accurately predicted by using the random forest algorithm. This data shows that, for white wine, the chemical properties do have some sort of correlation towards wine quality.

I also concluded that despite the differences in quality distribution, red wine does not have the same correlation variables as white wine. However, in both cases, the primary variable determined by the correlation matrix and the variable importance plot, is alcohol.

To extend and improve upon the accuracy of this analysis, I would need more data for red wine, close to the amount the white wine had. This would provide an accurate comparison between red and white wine quality. With more data, I would be able to allocate more to the training set and hopefully improve accuracy for the kNN algorithm. In these datasets, there is only chemical properties of the wine, to extend the analysis and improve accuracy of the wine quality prediction algorithms, more variables are needed. This could include things like color and even branding aspects like the name, font, and shape of the bottle or cost. This would show what aspects of the wine are perceived as better quality.