Predicting Wine Quality

Jake O'Donnell and Prateek Sai Mechineni

Introduction

Wine is an alcoholic beverage that has been associated with celebrations, leisure, and religious ceremonies dating back as far as 6000 BCE. The process of creating wine requires a chemical reaction involving yeast and the fermentation of grape juice. During the fermentation process the sugars found in grape juice are converted into ethanol, a type of alcohol. As wine making has progressed, the process has become increasingly advanced and is viewed widely as an art form. Vinters, those who make wine, use selective breeding processes when growing grapes to control which traits will carry on and which traits will die, creating their own unique grape strain. Each strain of grape created has its own unique taste, and when fermented with diverse types of yeast, produces a distinctive wine taste and is the reason each brand of wine tastes different than another.

Currently, there are over 10,000 different varieties of wine grapes and an estimated 300,000 wineries worldwide. With such a vast product selection, how does a consumer make the decision of what type of wine to purchase? To remove the need for a sommelier, we have set out on a mission to predict wine quality using analytical data. In our study we have used a red wine dataset from the UCI Machine Learning Repository to answer the all-important question. We believe that data analytics can be useful in predicting wine quality and will have benefits for both consumers and producers. Potential benefits include improved wine quality, a more competitive wine market, and increased production efficiencies.

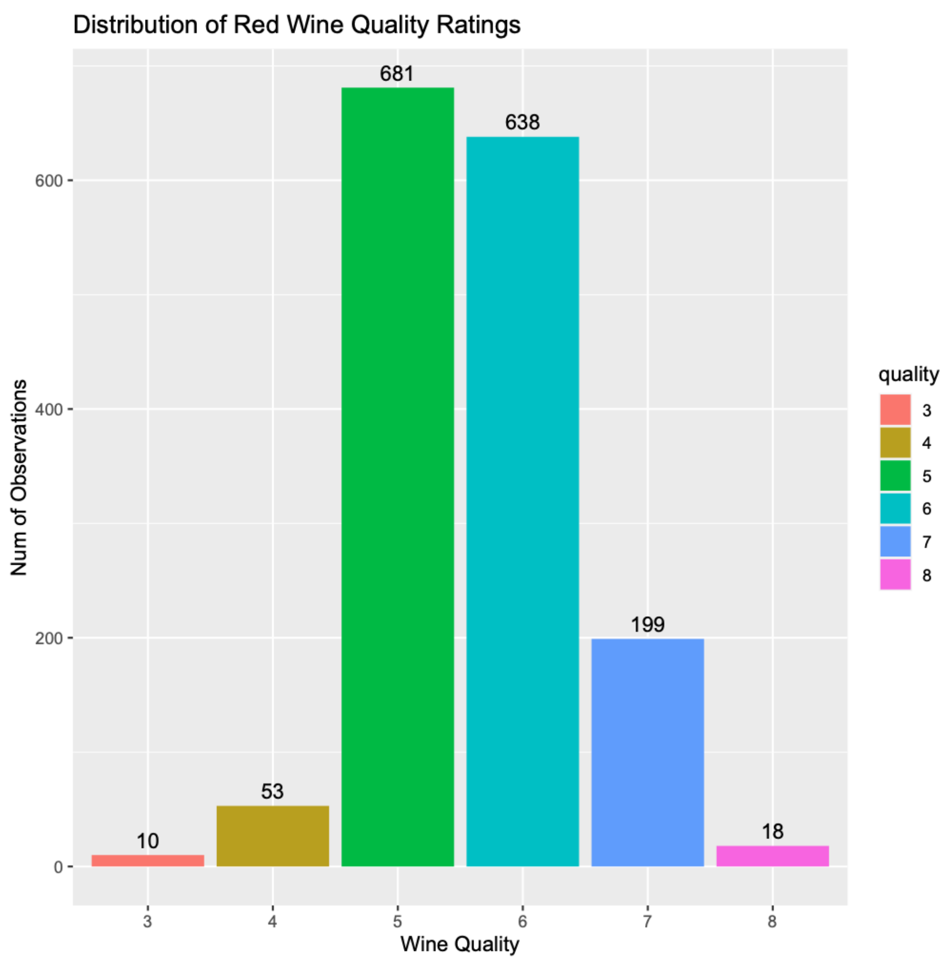
Dataset Description

The red wine dataset from the UCI Machine Learning Repository is comprised of 12 input variables including Fixed Acidity, Volatile Acidity, Citric Acid, Residual Sugar, Chlorides, Free Sulfur Dioxide, Total Sulfur Dioxide, Density, pH, Sulphates and Alcohol. The input attributes include objective tests, and the output is wine grade. The definitions of the input variables are as follows:

- Fixed acidity: Most acids involved with wine or fixed or nonvolatile (do not evaporate readily)
- Volatile acidity: The amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste
- ´Citric acid: Found in small quantities, citric acid can add 'freshness' and flavor to wines
- Residual sugar: the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter and
- ´Chlorides: the amount of salt in the wine

- Free Sulfur Dioxide: the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion; it prevents microbial growth and the oxidation of wine
- ´Total Sulfur Dioxide: amount of free and bound forms of S02; in low concentrations, SO2 is mostly undetectable in wine, but at free SO2 concentrations over 50 ppm, SO2 becomes evident in the nose and taste of wine
- Density: the density of water is close to that of water depending on the percent alcohol and sugar content
- pH describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4 on the pH scale
- Sulphates: a wine additive which can contribute to sulfur dioxide gas (S02) levels, which acts as an antimicrobial and antioxidant
- Alcohol: The percent alcohol of the wine

The dataset scores wine quality on a scale of 1-10, 1 being 'very bad', and 10 being 'very excellent'. The dataset is comprised of 1,599 instances. We ran a distribution to visualize the make-up of the dataset as shown below. What becomes evident is the number of wines that fell within the 5-6 range of wine quality.



As seen in the chart above, most wine fell within the 5-6 quality score, with only a handful scoring as low as three and as high as eight. This tells us that most wines are of similar quality, and most wines are in the middle of the quality scale. Additionally, no wine fell below a quality score of three, or greater than a score of eight. As we completed further analyses, the concertation of quality yielded itself more telling.

Data Preprocessing

To understand what input variables were considered when testing wine quality, we found it important to understand each variable's definition. You can find the definitions below.

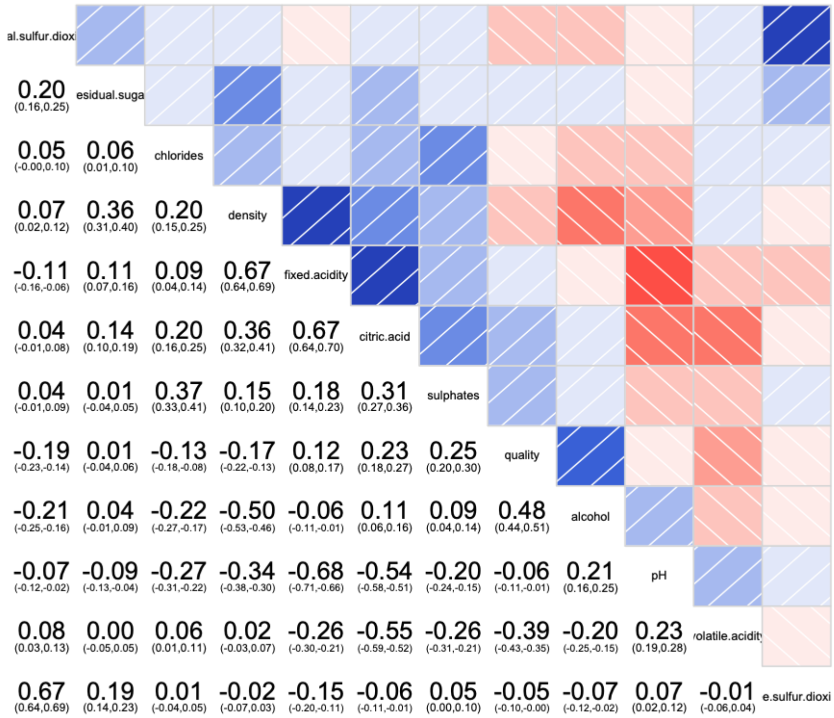
Data Cleaning

After a basic assessment of our dataset and the corresponding variables, we began to clean the data in preparation for a deeper analysis. Initially, we viewed the data as clean and opted not to do any cleansing. However, upon running our first decision tree model, we realized the model's accuracy was low and needed to revisit the data structure. Since there were no observations within the 1-2 and 9-10 quality ranges, we removed this from our dataset and grouped the rest into three categories, Bad (3-4), Medium (5-6), and Good (7-8). This lowered the decision making into three variables as opposed to six, which improved accuracy drastically. Since wine quality is similar, we determined it was unnecessary to have so many categories. Additionally, we scaled the data to make it more useable and get the data points closer together, which led to improvements in our modeling accuracy.

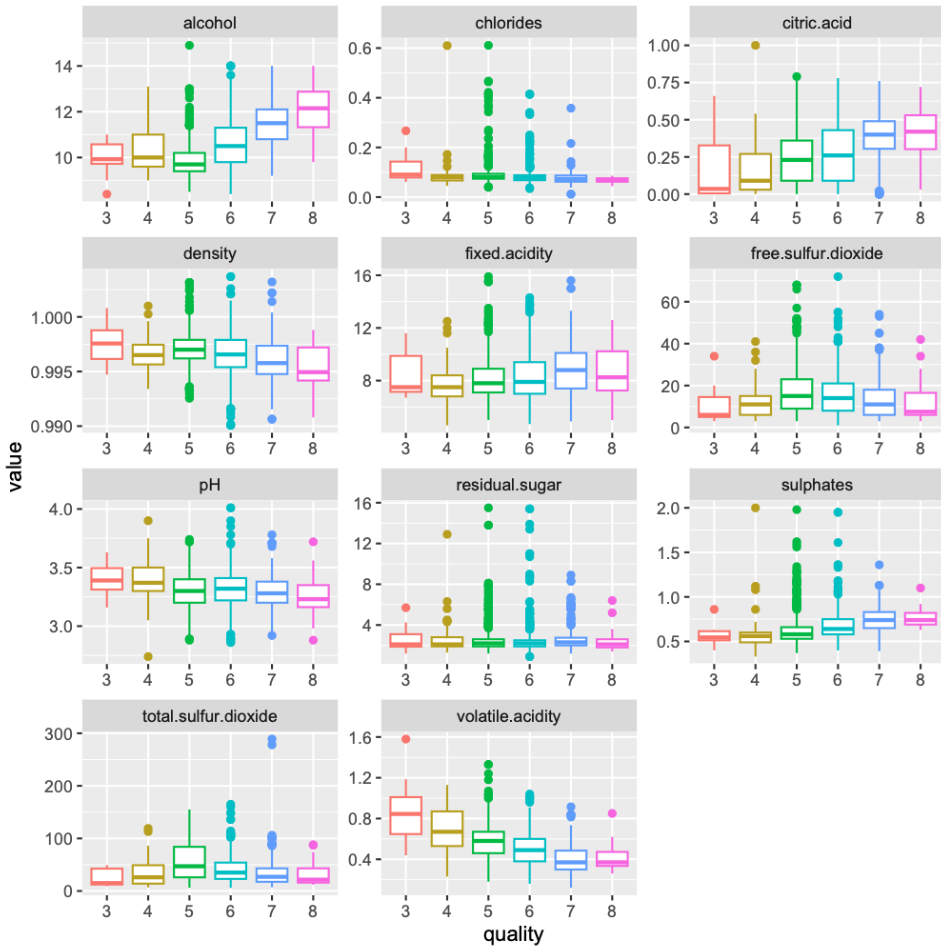
Correlation Analysis

One of the first tests we ran was a correlation analysis between the input variables to get a better understanding of which variables impacted quality the most, and which variables had high or low correlations to each other. We ran the following code and our correlation analysis looked as follows. We were a bit surprised to learn that alcohol and quality were the highest correlated variables. From our own personal experience, sometimes higher alcohol content overpowers the fine taste of an excellent wine.

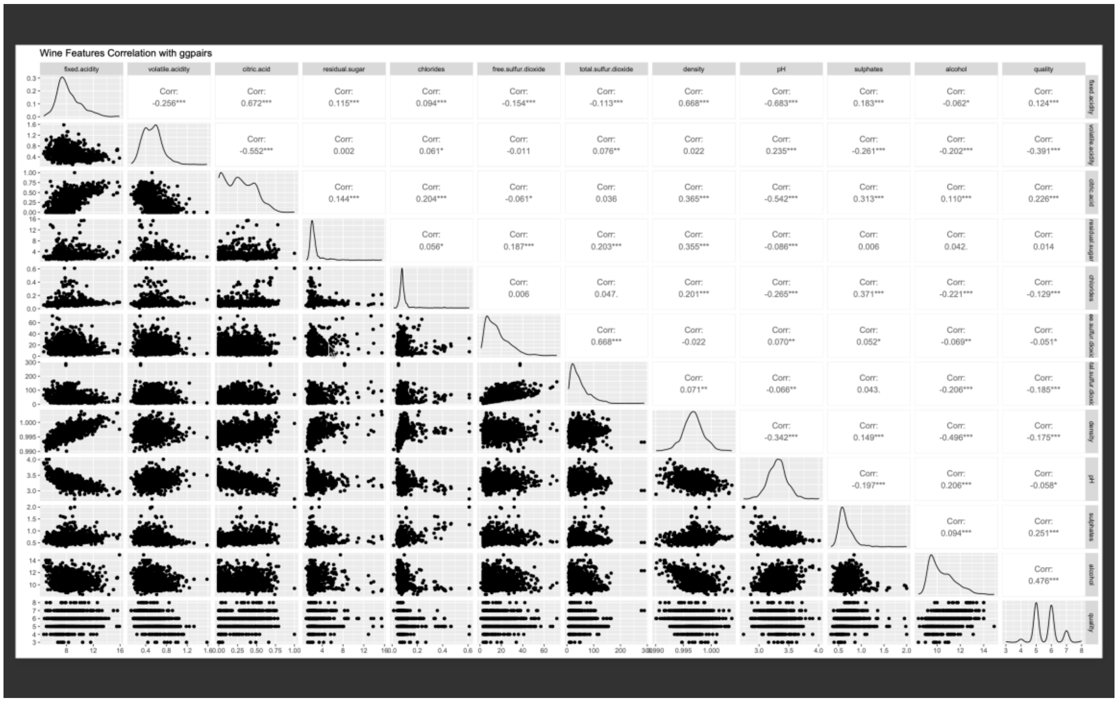
```
Redwinecorr <- cor(wine) %>%  as.data.frame() %>%  mutate(var1 = rownames(.)) %>%
gather(var2, value, -var1) %>%  arrange(desc(value)) %>%  group_by(value) %>%
filter(row_number()==1)

Redwinecorr
```



What we were able to determine is that Quality and Alcohol have the highest correlation, suggesting that wines with higher quality scores also have a higher alcohol content than those with a lower quality score. It also suggests that high volatile acidity and quality are inversely related. This is also exemplified in our boxplot chart, comparing quality score to each input variable.



We then used ggpairs to form another correlation analysis to get a better understanding of what variables were most correlated to each other, again finding the strongest correlation to be between quality and alcohol.

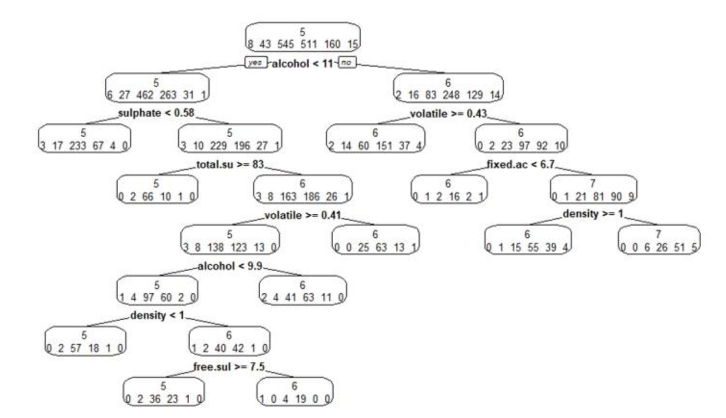


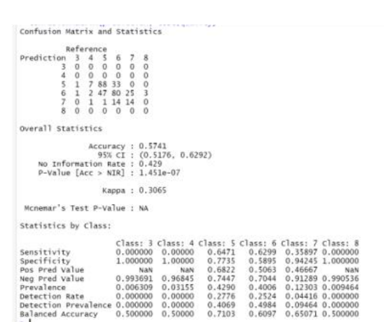
Data Analysis

After cleaning and processing our data, we began to apply various modeling techniques in our attempt to predict wine quality based on the wine's individual attributes. The modeling methods used include Decision Tree, Random Forest, Naïve Bayes, SVM and KNN.

Decision Tree

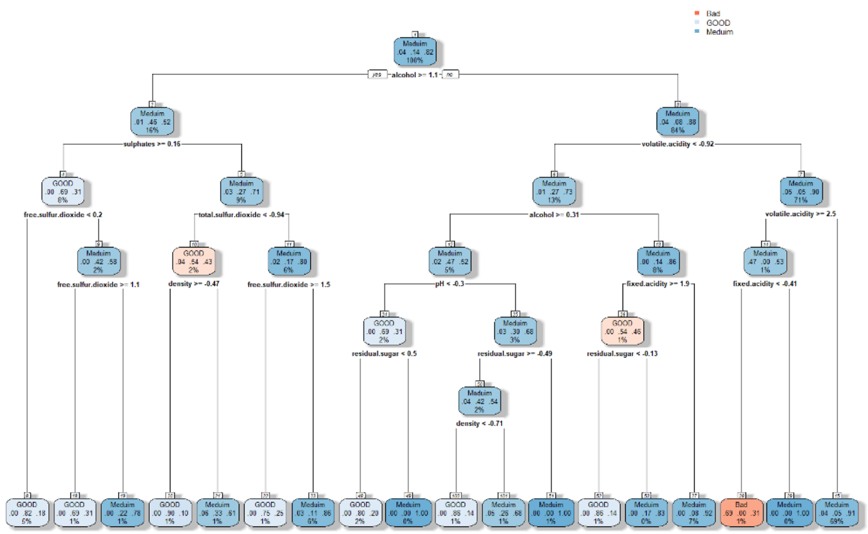
We started our analysis by running a decision tree as seen below. Our first tree resulted in a lackluster 57.4% accuracy, which led us to conclude that the data needed to be cleaned further to give us a more accurate representation. Originally, the decision tree was pointing to the individual wines rather than wine quality, so we decided to change that to reflect what we were looking for, that being quality.





```
Confusion Matrix and Statistics

          Reference
Prediction  3  4  5  6  7  8
         3  0  0  0  0  0  0
         4  0  0  0  0  0  0
         5  1  7 88 33  0  0
         6  1  2 47 80 25  3
         7  0  1  1 14 14  0
         8  0  0  0  0  0  0

Overall Statistics

               Accuracy : 0.5741
                 95% CI : (0.5176, 0.6292)
    No Information Rate : 0.429
    P-Value [Acc > NIR] : 1.451e-07

                  Kappa : 0.3065

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity          0.000000  0.00000   0.6471   0.6299  0.35897 0.000000
Specificity          1.000000  1.00000   0.7735   0.5895  0.94245 1.000000
Pos Pred Value            NaN      NaN   0.6822   0.5063  0.46667      NaN
Neg Pred Value       0.993691  0.96845   0.7447   0.7044  0.91289 0.990536
Prevalence           0.006309  0.03155   0.4290   0.4006  0.12303 0.009464
Detection Rate       0.000000  0.00000   0.2776   0.2524  0.04416 0.000000
Detection Prevalence 0.000000  0.00000   0.4069   0.4984  0.09464 0.000000
Balanced Accuracy    0.500000  0.50000   0.7103   0.6097  0.65071 0.500000
```

To improve our accuracy, we decided to scale the data to pull the data points together, as well as remove quality scores 1,2,9 and 10 since none of the wines scored in those categories. For ease of analysis, we then grouped the data into three categories, “Bad” (3-4), “Medium” (5-6) and “Good” (7-8). We then ran our decision tree again and received much more favorable results.



```
Confusion Matrix and Statistics

          Reference
Prediction Bad GOOD Meduim
    Bad      1    0      1
    GOOD     0   15     13
    Meduim  15   18    257

Overall Statistics

               Accuracy : 0.8531
                 95% CI : (0.8095, 0.89)
    No Information Rate : 0.8469
    P-Value [Acc > NIR] : 0.4146

                  Kappa : 0.3419

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Bad Class: GOOD Class: Meduim
Sensitivity            0.062500     0.45455        0.9483
Specificity            0.996711     0.95470        0.3265
Pos Pred Value         0.500000     0.53571        0.8862
Neg Pred Value         0.952830     0.93836        0.5333
Prevalence             0.050000     0.10312        0.8469
Detection Rate         0.003125     0.04688        0.8031
Detection Prevalence   0.006250     0.08750        0.9062
Balanced Accuracy      0.529605     0.70462        0.6374
```

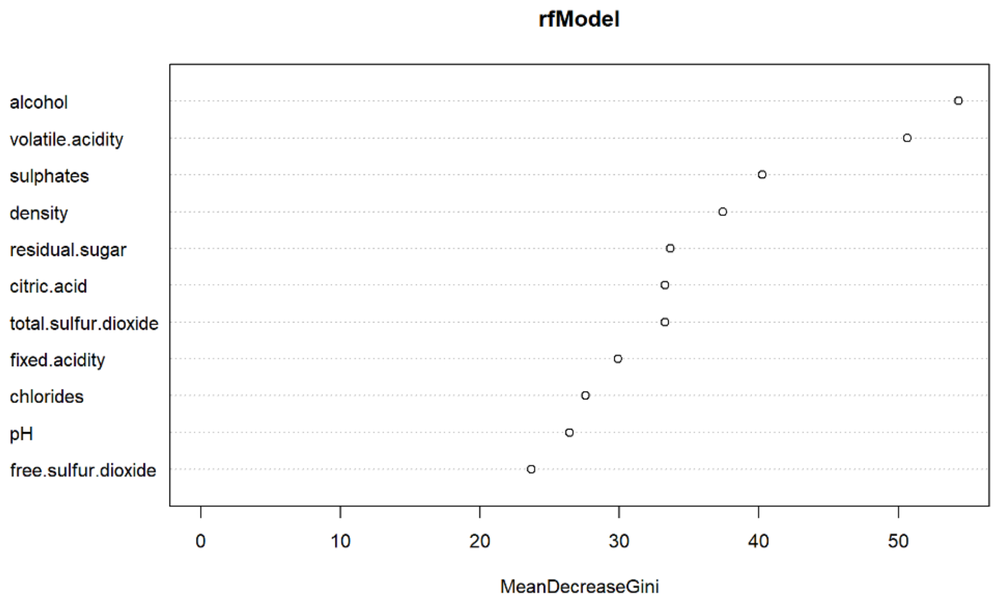
By scaling and splitting the data into more generalized ranges, we were able to improve accuracy, from 57.4% to 85.21%. One major takeaway was the impact of fixed acidity and wine quality. The decision tree revealed that when fixed acidity was lower than .41, it always resulted in poor wine quality.

Random Forest

We ran a Random Forest Model and again concluded that alcohol and volatile acidity had the most impact on wine quality. We used tuneRF to find the optimal value for mtry and that was determined to be 2. We tried different mtree sizes and minsplit sizes but that did not change the accuracy significantly. Once again, we assume that, since the data was scaled and already concentrated in the medium range, the data is already processed and changing the variables does not affect the prediction. The RF model also confirmed our EDA that alcohol and volatile acidity were the most key factors in predicting the quality of the wine. Overall, we tried multiple attempts but, in the end, we were able to get our accuracy as high as 89.06%.

```
rfModel <- randomForest(quality ~., data=train_data,
                        ntree=500,  # Use 500 trees in the forest
                        mtry=2,     # Consider 3 variables at each split
                        minsplit=10, # Require at least 10 samples at a node before splitting
                        minbucket=5  # Require at least 5 samples at a leaf node
)
```

```
> confusionMatrix(predictions3, test_data$quality)
Confusion Matrix and Statistics

          Reference
Prediction Bad GOOD Meduim
    Bad      1    0      0
    GOOD     0   19      6
    Meduim  15   14    265

Overall Statistics

               Accuracy : 0.8906
                 95% CI : (0.8512, 0.9226)
    No Information Rate : 0.8469
    P-Value [Acc > NIR] : 0.01513

                  Kappa : 0.4882

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Bad Class: GOOD Class: Meduim
Sensitivity            0.062500     0.57576        0.9779
Specificity            1.000000     0.97909        0.4082
Pos Pred Value         1.000000     0.76000        0.9014
Neg Pred Value         0.952978     0.95254        0.7692
Prevalence             0.050000     0.10312        0.8469
Detection Rate         0.003125     0.05937        0.8281
Detection Prevalence   0.003125     0.07812        0.9187
Balanced Accuracy      0.531250     0.77743        0.6930
> 
```



SVM

Support Vector Machines is an algorithm that seeks the optimum separating hyperplane between the variables by maximizing the margin between the classes' closest points. When running this model, we kept all the factors the same but changed the kernel each time to get the most accurate result. What we found was that linear and radial produced the most accurate predictions, while Poly was the least accurate.

```r
svmModel <- svm(quality ~., data=train_data,
                kernel="radial",   # Use a radial kernel
                degree=3,              # Use a degree of 2 for the radial
                cost=10,               # Use a cost of 10 for the SVM
                epsilon=0.1            # Use a tolerance of 0.1 for the SVM
)

svmModel2 <- svm(quality ~., data=train_data,
                kernel="polynomial",   # Use a polynomial kernel
                degree=3,              # Use a degree of 3 for the polynomial
                cost=10,               # Use a cost of 10 for the SVM
                epsilon=0.1            # Use a tolerance of 0.1 for the SVM
)

svmModel3 <- svm(quality ~., data=train_data,
                kernel="linear",   # Use a linear kernel
                degree=3,              # Use a degree of 3 for linear
                cost=10,               # Use a cost of 10 for the SVM
                epsilon=0.1            # Use a tolerance of 0.1 for the SVM
)
```

```
> confusionMatrix(predictionspoly, test_data$quality)
Confusion Matrix and Statistics

          Reference
Prediction Bad GOOD Meduim
    Bad      1    0      3
    GOOD     1   10     12
    Meduim  14   23    256

Overall Statistics

               Accuracy : 0.8344
                 95% CI : (0.789, 0.8734)
    No Information Rate : 0.8469
    P-Value [Acc > NIR] : 0.760449

                  Kappa : 0.2351

 Mcnemar's Test P-Value : 0.008991

Statistics by Class:

                     Class: Bad Class: GOOD Class: Meduim
Sensitivity            0.062500     0.30303        0.9446
Specificity            0.990132     0.95470        0.2449
Pos Pred Value         0.250000     0.43478        0.8737
Neg Pred Value         0.952532     0.92256        0.4444
Prevalence             0.050000     0.10312        0.8469
Detection Rate         0.003125     0.03125        0.8000
Detection Prevalence   0.012500     0.07187        0.9156
Balanced Accuracy      0.526316     0.62887        0.5948
```

```
> confusionMatrix(predictionsradial, test_data$quality)
Confusion Matrix and Statistics

          Reference
Prediction Bad GOOD Meduim
    Bad      0    0      0
    GOOD     0    0      0
    Meduim  16   33    271

Overall Statistics

               Accuracy : 0.8469
                 95% CI : (0.8027, 0.8845)
    No Information Rate : 0.8469
    P-Value [Acc > NIR] : 0.538

                  Kappa : 0

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Bad Class: GOOD Class: Meduim
Sensitivity                0.00      0.0000        1.0000
Specificity                1.00      1.0000        0.0000
Pos Pred Value              NaN         NaN        0.8469
Neg Pred Value             0.95      0.8969           NaN
Prevalence                 0.05      0.1031        0.8469
Detection Rate             0.00      0.0000        0.8469
Detection Prevalence       0.00      0.0000        1.0000
Balanced Accuracy          0.50      0.5000        0.5000
```

```
Confusion Matrix and Statistics

          Reference
Prediction Bad GOOD Meduim
    Bad      0    0      0
    GOOD     0    0      0
    Meduim  16   33    271

Overall Statistics

               Accuracy : 0.8469
                 95% CI : (0.8027, 0.8845)
    No Information Rate : 0.8469
    P-Value [Acc > NIR] : 0.538

                  Kappa : 0

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Bad Class: GOOD Class: Meduim
Sensitivity                0.00      0.0000        1.0000
Specificity                1.00      1.0000        0.0000
Pos Pred Value              NaN         NaN        0.8469
Neg Pred Value             0.95      0.8969           NaN
Prevalence                 0.05      0.1031        0.8469
Detection Rate             0.00      0.0000        0.8469
Detection Prevalence       0.00      0.0000        1.0000
Balanced Accuracy          0.50      0.5000        0.5000
```

Naïve Bayes

```r
nbModel <- naiveBayes(quality ~., data=train_data,
                     laplace=2,           # Use Laplace smoothing with a value of 2
                     usekernel=TRUE,     # Use a kernel density estimate for continuous variables
                     adjust=1            # Use a bandwidth adjustment factor of 1
)

nbModel2 <- naiveBayes(quality ~., data=train_data)
```

```
> confusionMatrix(predictionsnb1, test_data$quality)
Confusion Matrix and Statistics

          Reference
Prediction Bad GOOD Meduim
    Bad      5    3     14
    GOOD     0   23     38
    Meduim  11    7    219

Overall Statistics

               Accuracy : 0.7719
                 95% CI : (0.7219, 0.8167)
    No Information Rate : 0.8469
    P-Value [Acc > NIR] : 0.9998

                  Kappa : 0.3476

 Mcnemar's Test P-Value : 1.771e-05

Statistics by Class:

                     Class: Bad Class: GOOD Class: Meduim
Sensitivity             0.31250     0.69697        0.8081
Specificity             0.94408     0.86760        0.6327
Pos Pred Value          0.22727     0.37705        0.9241
Neg Pred Value          0.96309     0.96139        0.3735
Prevalence              0.05000     0.10312        0.8469
Detection Rate          0.01562     0.07187        0.6844
Detection Prevalence    0.06875     0.19062        0.7406
Balanced Accuracy       0.62829     0.78228        0.7204
>
```

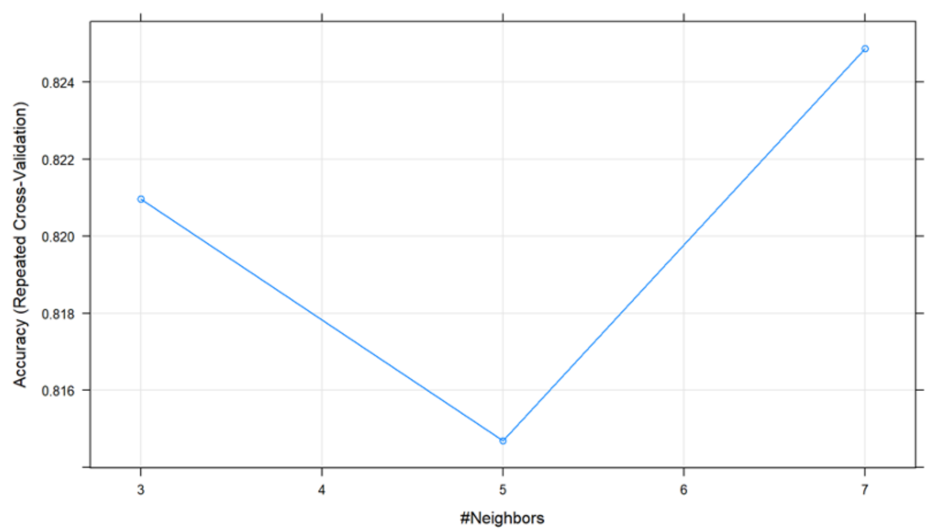
```
> confusionMatrix(predictionsnb2, test_data$quality)
Confusion Matrix and Statistics

          Reference
Prediction Bad GOOD Meduim
    Bad      5    3     14
    GOOD     0   23     38
    Meduim  11    7    219

Overall Statistics

               Accuracy : 0.7719
                 95% CI : (0.7219, 0.8167)
    No Information Rate : 0.8469
    P-Value [Acc > NIR] : 0.9998

                  Kappa : 0.3476

 Mcnemar's Test P-Value : 1.771e-05

Statistics by Class:

                     Class: Bad Class: GOOD Class: Meduim
Sensitivity             0.31250     0.69697        0.8081
Specificity             0.94408     0.86760        0.6327
Pos Pred Value          0.22727     0.37705        0.9241
Neg Pred Value          0.96309     0.96139        0.3735
Prevalence              0.05000     0.10312        0.8469
Detection Rate          0.01562     0.07187        0.6844
Detection Prevalence    0.06875     0.19062        0.7406
Balanced Accuracy       0.62829     0.78228        0.7204
```

After running the Naïve Bayes model, we were able to get an accuracy of 77.19%. Despite changing the model, we could not improve our accuracy despite running it multiple times. We believe the reason for this is because we had already scaled our data and that already reduces over/under fitting. Due to this, the smoothing variable could not improve our model.

KNN

```
# Train the KNN classifier with modified parameters
k_values <- data.frame(k=c(3, 5, 7))
ctrl <- trainControl(method="repeatedcv", number=3, verboseIter=TRUE)
knnfit <- train(quality~., data = train_data, method = "knn", trControl = ctrl, tuneGrid = k_values)
# Use the trained model to make predictions on the test data
knnpred <- predict(knnfit, newdata = test_data)
```

```
> confusionMatrix(data = as.factor(knnpred), as.factor(test_data$quality))
Confusion Matrix and Statistics

          Reference
Prediction Bad GOOD Meduim
    Bad      2    0      2
    GOOD     0   13     13
    Meduim  14   20    256

Overall Statistics

               Accuracy : 0.8469
                 95% CI : (0.8027, 0.8845)
    No Information Rate : 0.8469
    P-Value [Acc > NIR] : 0.538

                  Kappa : 0.3149

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Bad Class: GOOD Class: Meduim
Sensitivity             0.12500     0.39394        0.9446
Specificity             0.99342     0.95470        0.3061
Pos Pred Value          0.50000     0.50000        0.8828
Neg Pred Value          0.95570     0.93197        0.5000
Prevalence              0.05000     0.10312        0.8469
Detection Rate          0.00625     0.04063        0.8000
Detection Prevalence    0.01250     0.08125        0.9062
Balanced Accuracy       0.55921     0.67432        0.6254
```



We also conducted a KNN prediction on the dataset. With KNN, we can specify how many neighbors/clusters we want the algorithm to use while trying to predict. We used 3,5, and 7 as good starting points and found that they provide the best accuracy. We thought that maybe as the number of neighbors goes up, the accuracy would go up as well, but that was proven incorrect. 3 had a higher accuracy than 5 but 7 had a higher accuracy than both. It was not what we expected but, in the end, we were able to get an accuracy of 84.69% that was on par with our other predictive methods.

Conclusion

In conclusion, we were able to determine that alcohol and quality have a strong positive relationship, and acidity and wine quality are inversely related. The results of our analysis showed that Random Forest had the highest accuracy of all the models we analyzed. What we were able to determine is that alcohol and quality have a strong positive relationship, and acidity and wine quality are inversely related. While all the models provided us with valuable insight, there were some limitations to our dataset. Given there were only 1,599 observations, and 82.4% of those were in the 5-6 quality range, we were limited in the conclusions we were able to draw. Going forward, we would like to analyze a more robust and balanced dataset in which we can train our data against wine with scores greater than eight as well as wine with scores less than three. Also, we believe that there are over 12 input variables that impact wine quality, and it would be important to explore that further to understand what variables impact wine quality.