DATA 440 Project: Wine Quality Classification

Max Schemitsch

May 15, 2019

Dataset Citation

This dataset is public available for research. The details are described in [Cortez et al., 2009].

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

Available at: [@Elsevier] http://dx.doi.org/10.1016/j.dss.2009.05.016 [Pre-press (pdf)] http://www3.dsi.uminho.pt/pcortez/winequality09.pdf [bib] http://www3.dsi.uminho.pt/pcortez/dss09.bib

---

Dataset Notes

These two datasets, which I have downloaded from Kaggle, use red and white wine samples. The metrics used are objective levels like acidity, pH, chlorides, and sugars. Using these types of levels, a score, or quality, is output on a scale of 0 to 10, with 0 being very poor and 10 being outstanding.

The wine used in this dataset are variants of the Portuguese "Vinho Verde" wine. This wine comes from the Minho province in northwest Portugal.[1] This region is one of the largest wine regions on the planet.

Attribute Information

Before diving into the dataset, there are a few bits of information that can be extracted at face value.

Our dataframe has 1599 instances of red wines, and 4898 instances of white wines. Although more than two-thirds of the data is white-wine related, the methods that we use will regress them together.

We have a total of 11 input attributes:

Input variables (based on physicochemical tests):

```
1 - fixed acidity (tartaric acid - g / dm^3)

2 - volatile acidity (acetic acid - g / dm^3)

3 - citric acid (g / dm^3)

4 - residual sugar (g / dm^3)

5 - chlorides (sodium chloride - g / dm^3

6 - free sulfur dioxide (mg / dm^3)

7 - total sulfur dioxide (mg / dm^3)
```

```
   8 - density (g / cm^3)

   9 - pH

   10 - sulphates (potassium sulphate - g / dm3)

   11 - alcohol (% by volume)
```

We also have our singular output attribute:

```
Output variable (based on sensory data):
   12 - quality (score between 0 and 10)
```

The Kaggle site this data is from includes what each attribute means[2]:

```
   1 - fixed acidity: most acids involved with wine or fixed or nonvolatile (do not
   evaporate readily)

   2 - volatile acidity: the amount of acetic acid in wine, which at too high of levels
   can lead to an unpleasant, vinegar taste

   3 - citric acid: found in small quantities, citric acid can add 'freshness' and flavor
   to wines

   4 - residual sugar: the amount of sugar remaining after fermentation stops, it's rare
   to find wines with less than 1 gram/liter and wines with greater than 45 grams/liter
   are considered sweet

   5 - chlorides: the amount of salt in the wine

   6 - 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

   7 - total sulfur dioxide: amount of free and bound forms of SO2; 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

   8 - density: the density of water is close to that of water depending on the percent
   alcohol and sugar content

   9 - 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

   10 - sulphates: a wine additive which can contribute to sulfur dioxide gas (SO2) levels,
   which acts as an antimicrobial and antioxidant

   11 - alcohol: the percent alcohol content of the wine
```

Project Goals & Methods

For this project, there will be two parts of interest.

The first part of this project will look in-depth at the characteristics and correlations of variables.

The primary purpose of this section is to first gain an understanding of the data. Additionally, it will gives us an idea of what variables will be useful when moving to the second part of the project.

The second part of this project will be using different regression methods to create models that determine wine scores.

Load Data & Libraries

```
library(ggplot2)
library(corrplot)
library(randomForest)
library(caret)
library(class)
library(boot)
library(tree)
library(leaps)
red = read.csv("https://www.dropbox.com/s/jtfubj8tfqpqsa4/wineReds.csv?dl=1")
white = read.csv("https://www.dropbox.com/s/n1pbwl5fkne2i3k/wineWhites.csv?dl=1")
red["color"]="red"
white["color"]="white"
df = rbind(red, white)
attach(df)
```

Dataframe Characteristics

```
head(df)
```

```
##   X fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1 1           7.4             0.70        0.00            1.9     0.076
## 2 2           7.8             0.88        0.00            2.6     0.098
## 3 3           7.8             0.76        0.04            2.3     0.092
## 4 4          11.2             0.28        0.56            1.9     0.075
## 5 5           7.4             0.70        0.00            1.9     0.076
## 6 6           7.4             0.66        0.00            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 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 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 color
## 1       5   red
```

```
## 2       5   red
## 3       5   red
## 4       6   red
## 5       5   red
## 6       5   red
```

```
names(df)
```

```
##  [1] "X"                    "fixed.acidity"        "volatile.acidity"
##  [4] "citric.acid"          "residual.sugar"       "chlorides"
##  [7] "free.sulfur.dioxide"  "total.sulfur.dioxide" "density"
## [10] "pH"                   "sulphates"            "alcohol"
## [13] "quality"              "color"
```

Looking at the top of our dataset, tells us our attribute variable names, and a general idea of what their values are like.

(We can also see that the integer attribute that increments the wines is called X.)

```
summary(df)
```

```
##        X        fixed.acidity    volatile.acidity  citric.acid
##  Min.   :   1   Min.   : 3.800   Min.   :0.0800   Min.   :0.0000
##  1st Qu.: 813   1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2500
##  Median :1650   Median : 7.000   Median :0.2900   Median :0.3100
##  Mean   :2044   Mean   : 7.215   Mean   :0.3397   Mean   :0.3186
##  3rd Qu.:3274   3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.3900
##  Max.   :4898   Max.   :15.900   Max.   :1.5800   Max.   :1.6600
##  residual.sugar     chlorides       free.sulfur.dioxide
##  Min.   : 0.600   Min.   :0.00900   Min.   :  1.00
##  1st Qu.: 1.800   1st Qu.:0.03800   1st Qu.: 17.00
##  Median : 3.000   Median :0.04700   Median : 29.00
##  Mean   : 5.443   Mean   :0.05603   Mean   : 30.53
##  3rd Qu.: 8.100   3rd Qu.:0.06500   3rd Qu.: 41.00
##  Max.   :65.800   Max.   :0.61100   Max.   :289.00
##  total.sulfur.dioxide    density             pH          sulphates
##  Min.   :  6.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200
##  1st Qu.: 77.0        1st Qu.:0.9923   1st Qu.:3.110   1st Qu.:0.4300
##  Median :118.0        Median :0.9949   Median :3.210   Median :0.5100
##  Mean   :115.7        Mean   :0.9947   Mean   :3.219   Mean   :0.5313
##  3rd Qu.:156.0        3rd Qu.:0.9970   3rd Qu.:3.320   3rd Qu.:0.6000
##  Max.   :440.0        Max.   :1.0390   Max.   :4.010   Max.   :2.0000
##     alcohol         quality         color
##  Min.   : 8.00   Min.   :3.000   Length:6497
##  1st Qu.: 9.50   1st Qu.:5.000   Class :character
##  Median :10.30   Median :6.000   Mode  :character
##  Mean   :10.49   Mean   :5.818
##  3rd Qu.:11.30   3rd Qu.:6.000
##  Max.   :14.90   Max.   :9.000
```

The summary of our dataframe gives important values like averages, minimums, and maximums.

Looking at these values, there are a few important notes to make:

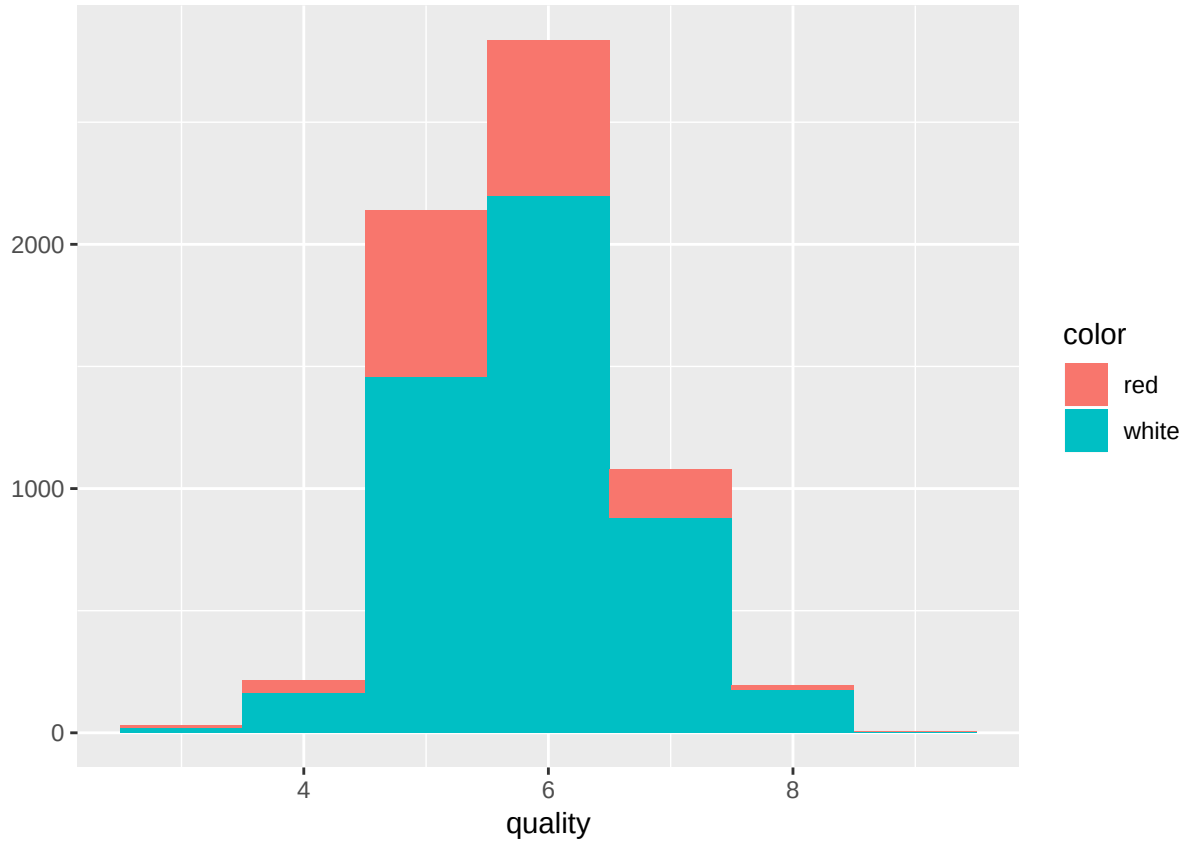
```
1 - Minimum quality is 3 and maximum is 9

2 - The average residual sugar is skewed left: the mean is 5.443, but has a maximum of 65.8.

3 - Similarly, chlorides is skewed. It's range is from 0.009 to 0.611, but has a mean of 0.056.

4 - In the same vein, both free form sulfur dioxide and total sulfur dioxide averages are skewed.
```

First, we can take a look at the distribution of scores and wines:

```
table(quality)
```

```
## quality
##    3    4    5    6    7    8    9
##   30  216 2138 2836 1079  193    5
```

```
qplot(quality, data = df, fill = color, binwidth = 1)
```

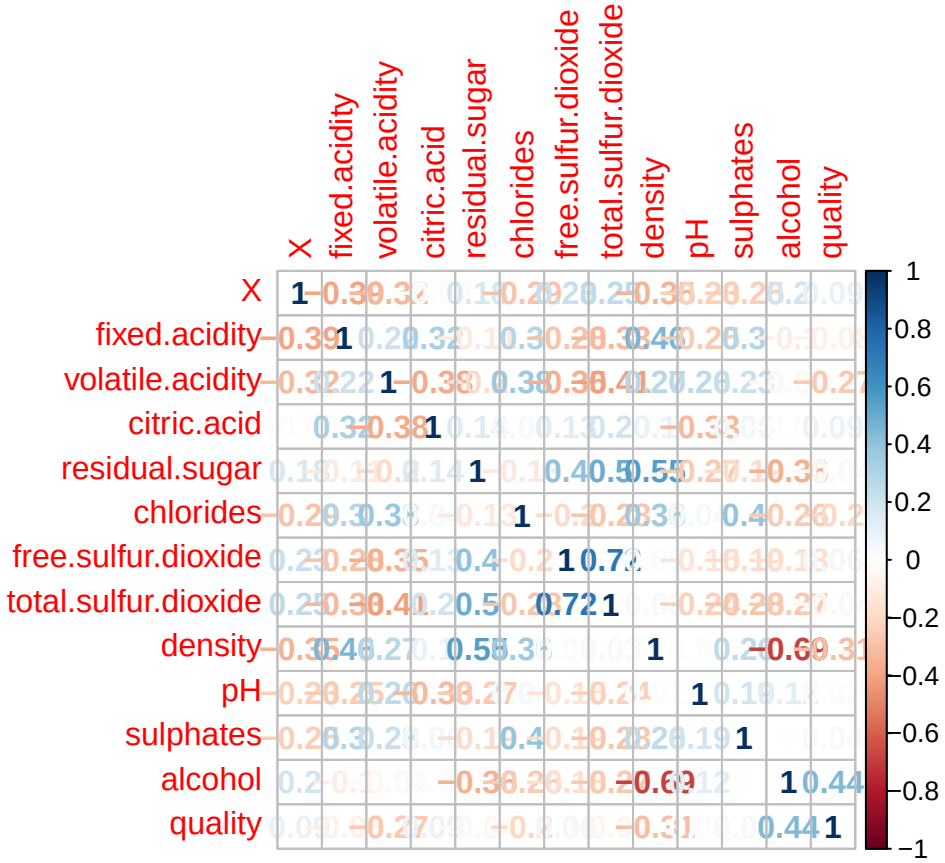


This shows us that a majority (~6,000) of score values lie between the 5-7 range. There are roughly 200 values for both scores of 4 and scores of 8. Finally, we only have 30 scores of 3 and a miniscule five scores of 9.

The histogram of wine qualities also shows us the the scores are normally distribued for both red and white wines.

Next what we can do, before anything else, is check the correlations of variables. This will help us better understand which characteristics are and aren't related.

```
dfcorr=cor(df[,-14])  # our correlation coefficients
corrplot(dfcorr, method="number")
```



Since we want to predict quality score, we can first look there for variables. A surprising (or unsurprising) seven of the eleven characteristics have correlation coefficients of between -0.1 and 0.1. This means these variables most likely have very little to do with score output.

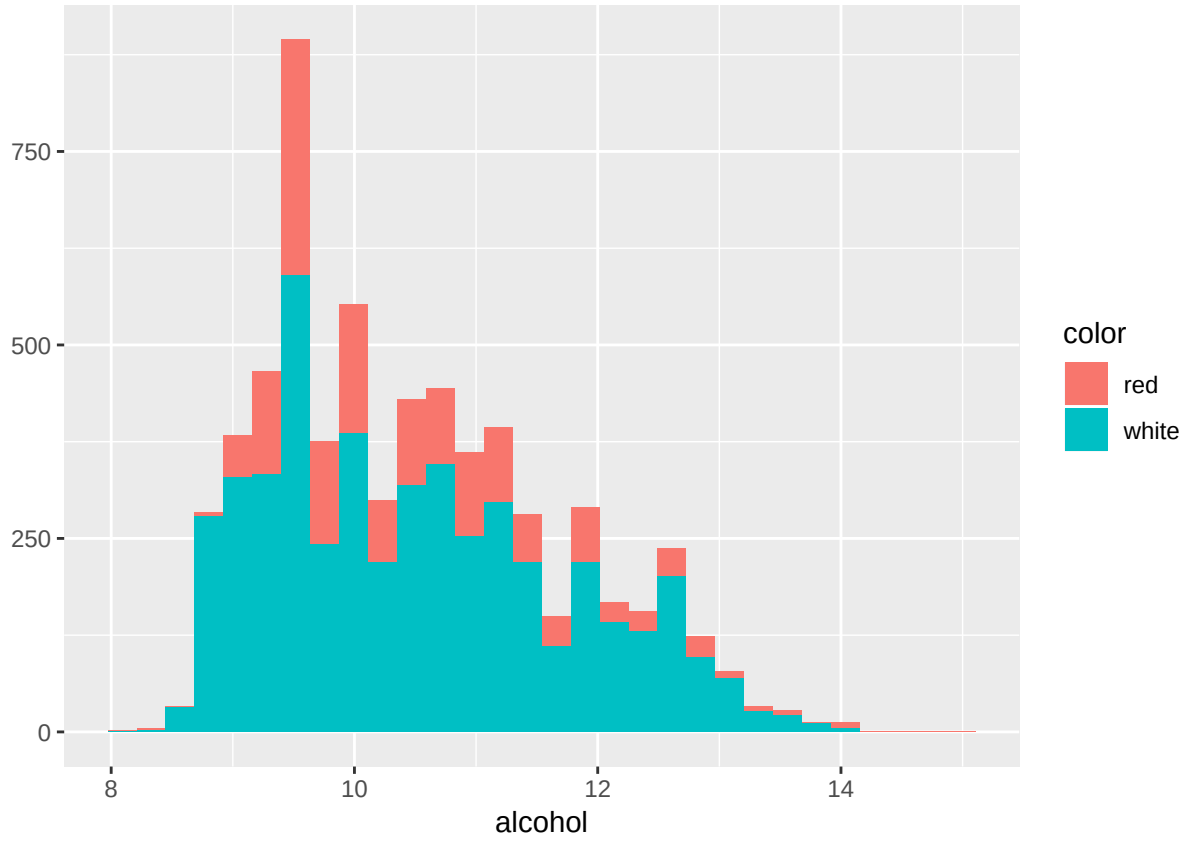
Does this mean they won't be included in our modeling process? Not necessarily. Seemingly irrelevant data points can actually improve model accuracy if included.

The other four variables with at least a ± 2 correlation are volatile acidity (-0.27), chlorides (-0.2), density (0.31), and alcohol level (0.44). Before we do any analysis of these variables, we can tell that these will most likely be important to our modelling process later on.

Additional Analysis

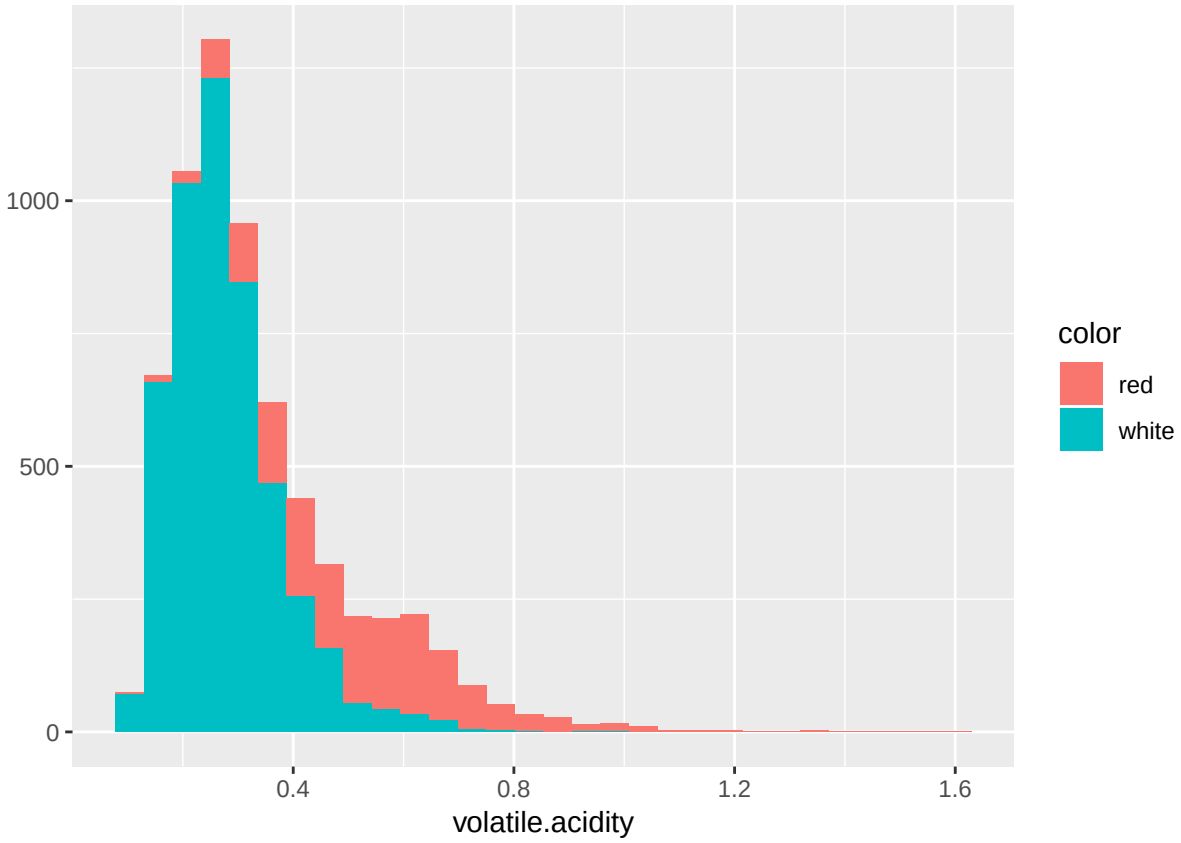
```
qplot(alcohol, data=df, fill=color)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qplot(volatile.acidity, data=df, fill=color)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
tapply(alcohol, quality, mean)
```

```
##        3        4        5        6        7        8        9
## 10.215000 10.180093  9.837783 10.587553 11.386006 11.678756 12.180000
```

```
tapply(density, quality, mean)
```

```
##        3        4        5        6        7        8        9
## 0.9957440 0.9948326 0.9958490 0.9945583 0.9931259 0.9925135 0.9914600
```

```
tapply(volatile.acidity, quality, mean)
```

```
##        3        4        5        6        7        8        9
## 0.5170000 0.4579630 0.3896141 0.3138628 0.2887998 0.2910104 0.2980000
```

```
tapply(chlorides, quality, mean)
```

```
##          3          4          5          6          7          8
## 0.07703333 0.06005556 0.06466604 0.05415726 0.04527247 0.04112435
##          9
## 0.02740000
```

We can see that quality score generally rises as alcohol percent increases. We can also see that one third of wines have a score of 5.

It appears that density stays relatively the same throughout all quality scores, ever so slightly decreasing.

Additionally, volatile acidity and chloride levels generally decrease as quality goes up.

Predicting Quality Score

For the purposes of this project, the methods we will use to predict quality score are Multiple Linear Regression, k-Nearest Neighbors, and Random Forest. We will first start with Multiple Linear Regression.

Linear Regression

Our first model to check will include every variable:

```
mod.all=lm(quality~fixed.acidity+volatile.acidity+citric.acid+residual.sugar
           +chlorides+free.sulfur.dioxide+total.sulfur.dioxide+density+pH
           +sulphates+alcohol, data=df)
summary(mod.all)$adj.r.squared
```

```
## [1] 0.2909362
```

We can see that with this model, the adjusted R-squared value is only 0.2909. This is rather poor. Our next goal is to use the attribute analysis from before to trim unnecessary parameters. First, we can try just using the alcohol attribute, as it has the highest correlation value with quality.

```
mod1=lm(quality~alcohol, data=df)
summary(mod1)$adj.r.squared
```

```
## [1] 0.1972954
```

While the adjusted R-squared is worse than before, it still achieves a value of 0.1973 with just one variable. We can check one more model with the variables that have higher correlations with quality.

```
mod2=lm(quality~alcohol+volatile.acidity+density, data=df)
summary(mod2)$adj.r.squared
```

```
## [1] 0.2669701
```

We can see an immediate increase in our R-squared value after using just three variables. We can do some further model testing to find the best model with only four variables.

```
mod3=lm(quality~alcohol+volatile.acidity+density+sulphates, data=df)
summary(mod3)$adj.r.squared
```

```
## [1] 0.2725567
```

```
mod4=lm(quality~alcohol+volatile.acidity+density+pH, data=df)
summary(mod4)$adj.r.squared
```

```
## [1] 0.2677253
```

```
mod5=lm(quality~alcohol+volatile.acidity+density+total.sulfur.dioxide, data=df)
summary(mod5)$adj.r.squared
```

```
## [1] 0.2675827
```

```
mod6=lm(quality~alcohol+volatile.acidity+density+chlorides, data=df)
summary(mod6)$adj.r.squared
```

```
## [1] 0.2669253
```

```
mod7=lm(quality~alcohol+volatile.acidity+density+residual.sugar, data=df)
summary(mod7)$adj.r.squared
```

```
## [1] 0.267925
```

```
mod8=lm(quality~alcohol+volatile.acidity+density+citric.acid, data=df)
summary(mod8)$adj.r.squared
```

```
## [1] 0.267831
```

From these various four-attribute models, our best overall adjusted R-square value is 0.2725567 with model 3. Notice how this model includes sulphates which only had a 0.04 correlation with quality.

Linear Regression Results

We can see that model 3 is indeed the most effective four-variable method to predict quality. Despite this, our R-square values are still relatively poor. This can be explained by a number of things.

Primarily, this is due to the fact that wine scoring is inherently subjective. Different people have different tastebuds, and there is no formula for calculating taste. Although the attributes in this dataset have qualities that people associate with good or bad tastes (like citric acid being sour or residual sugar being sweet), the overall quality of a wine will vary from person to person.

The second important idea to understand is that many of the variables were not correlated at all with quality. The few that were slightly correlated still only have values less than 0.5 or greater than -0.5. Level of alcohol was the only true predictor, and this is seen in our model only using that as a predictor. While adding the other three variables did contribute to better results, it only bumped up the R-square value by nine percent.

---

k-Nearest Neighbors

With the work in linear regression leaving us unsatisfied, what can we do to improve accuracy? In this section, k-Nearest Neighbors will be used to predict quality. In order to do this, we will have to simplify the quality characteristic. Instead of using each score (from 3 to 9), we can separate these scores in three categories. For this instance, scores 3 and 4 will be Poor, 5 and 6 will be Average, and 7 through 9 will be Good.

```
df$qualityCat=df$quality
df$qualityCat[which(df$quality %in% c(3,4))] = "Poor"
df$qualityCat[which(df$quality %in% c(5,6))] = "Average"
df$qualityCat[which(df$quality %in% c(7,8,9))] = "Good"
```

This first bit allows us to create a new attribute, *qualityCat*, that separates each quality score into the three types we planned before.

```
set.seed(1)
n=nrow(df)
index=sample(1:n, size=trunc(0.7*n))
train=df[index,]
test=df[-index,]
x_train=train[2:12]
y_train=train$qualityCat
x_test=test[2:12]
y_test=test$qualityCat
```

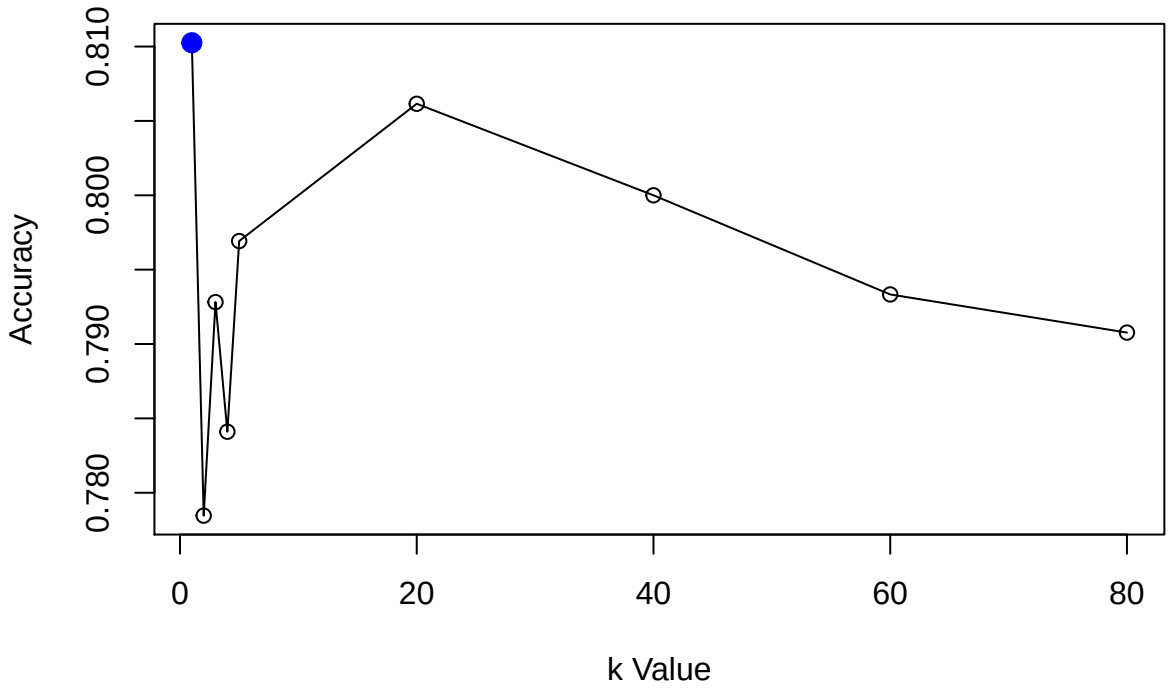
Here we separate the data into training and testing sets. Our data will utilize all variables to predict *qualityCat*. Now we can begin modeling.

```
knn1=knn(train=scale(x_train), test=scale(x_test), y_train, k=1, prob=TRUE)
knn2=knn(train=scale(x_train), test=scale(x_test), y_train, k=2, prob=TRUE)
knn3=knn(train=scale(x_train), test=scale(x_test), y_train, k=3, prob=TRUE)
knn4=knn(train=scale(x_train), test=scale(x_test), y_train, k=4, prob=TRUE)
knn5=knn(train=scale(x_train), test=scale(x_test), y_train, k=5, prob=TRUE)
knn10=knn(train=scale(x_train), test=scale(x_test), y_train, k=10, prob=TRUE)
knn20=knn(train=scale(x_train), test=scale(x_test), y_train, k=20, prob=TRUE)
knn40=knn(train=scale(x_train), test=scale(x_test), y_train, k=40, prob=TRUE)
knn60=knn(train=scale(x_train), test=scale(x_test), y_train, k=60, prob=TRUE)
knn80=knn(train=scale(x_train), test=scale(x_test), y_train, k=80, prob=TRUE)
```

Our models for k-Nearest Neighbors will use $k = 1, 2, 3, 4, 5, 10, 20, 40, 60, 80$. This gives us a wide range of possible model fits. Now we can check the accuracy of these models and chart the accuracy incrementally.

```
u1=mean(knn1==y_test)
u2=mean(knn2==y_test)
u3=mean(knn3==y_test)
u4=mean(knn4==y_test)
u5=mean(knn5==y_test)
u10=mean(knn10==y_test)
u20=mean(knn20==y_test)
u40=mean(knn40==y_test)
u60=mean(knn60==y_test)
u80=mean(knn80==y_test)
```

```
u=c(u1, u2, u3, u4, u5, u20, u40, u60, u80)
incre=c(1, 2, 3, 4, 5, 20, 40, 60, 80)
plot(incre, u, xlab="k Value", ylab="Accuracy")
lines(incre,u)
points(max(u), col="blue", cex=2, pch=20)
```



It can be seen that $k = 1$ is our best bet, as it yields the best accuracy. We can check the accuracy for $k = 1$.

```
mean(knn1==y_test)
```

```
## [1] 0.8102564
```

```
table(knn1, y_test)
```

```
##          y_test
## knn1      Average Good Poor
##   Average    1319  126   56
##   Good        145  245    5
##   Poor         35    3   16
```

Our accuracy is roughly 81%. We also see that our prediction table has some rough areas. Specifically, predicting the Average qualities yields very good results, but predicting Good and Poor qualities are a bit lacking. This is probably due to the fact that both the mean and median are in the Average range.

```r
mean(df$quality)
```

```
## [1] 5.818378
```

```r
median(df$quality)
```

```
## [1] 6
```

Tree Method / Random Forest

So far, we've seen the lackluster performance of linear regression and the relative success of k-Nearest Neighbors. The last method I would like to try is a combination of the Tree method and Random Forest. This method utilizes decision trees that determine specific categories based on variable values. For this particular method, we will again use our *qualityCat* attribute. To do this, we must make sure *qualityCat* has levels:

```r
df$qualityCat=as.factor(df$qualityCat)
head(df$qualityCat)
```

```
## [1] Average Average Average Average Average Average
## Levels: Average Good Poor
```

```r
detach(df)
attach(df)
```

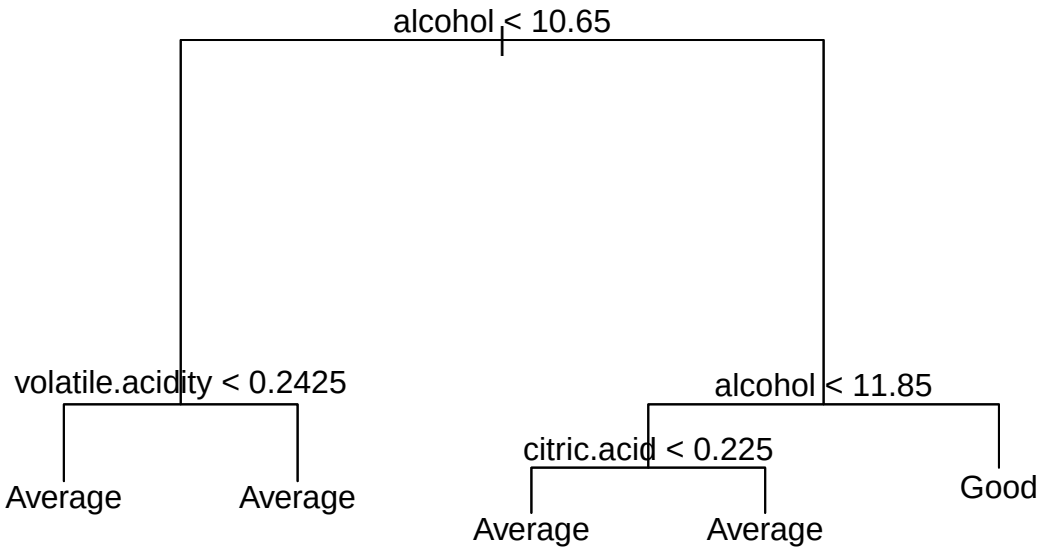
First we can make a tree model with every variable, and then find the optimal number of attributes.

```r
set.seed(2)
index=sample(1:nrow(df), trunc(0.7*nrow(df)))
train=df[index,]
test=df[-index,]
tree1=tree(qualityCat~alcohol+sulphates+pH+density+total.sulfur.dioxide
           +free.sulfur.dioxide+chlorides+residual.sugar+citric.acid
           +volatile.acidity+fixed.acidity, data=train)
summary(tree1)
```

```
##
## Classification tree:
## tree(formula = qualityCat ~ alcohol + sulphates + pH + density +
##     total.sulfur.dioxide + free.sulfur.dioxide + chlorides +
##     residual.sugar + citric.acid + volatile.acidity + fixed.acidity,
##     data = train)
## Variables actually used in tree construction:
## [1] "alcohol"          "volatile.acidity" "citric.acid"
## Number of terminal nodes:  5
## Residual mean deviance:  1.095 = 4974 / 4542
## Misclassification error rate: 0.2281 = 1037 / 4547
```

First, it can be seen that the only variables used for this tree were alcohol, volatile acidity, citric acid. We also have an error rate of 23%. Next we can plot the tree:

```
plot(tree1)
text(tree1, pretty=0)
```



As it turns out, at this particular step in the process is mostly useless. The tree shows that a good portion of the wines are infact average rating, and that an alcohol percent of almost 12% or above results in a good rating.

```
p=predict(tree1, test, type="class")
mean(p==test$qualityCat)
```

```
## [1] 0.76
```

Using the testing data, our prediction rate is actually reasonable. Roughly three out of every four predictions turned out to be right.

```
cv.prunce=cv.tree(tree1, FUN=prune.misclass)
best.size=cv.prunce$size[which(cv.prunce$dev==min(cv.prunce$dev))]
best.size
```

```
## [1] 5 3
```

With these lines of code, we can now see that the best size of model is either 3, or 5. Next we can check which variables are the best for models of size 3 and 5.

```
reg=regsubsets(qualityCat~fixed.acidity+volatile.acidity+citric.acid+residual.sugar
               +chlorides+free.sulfur.dioxide+total.sulfur.dioxide
               +density+pH+sulphates+alcohol, df)
summary(reg)
```

```
## Subset selection object
## Call: regsubsets.formula(qualityCat ~ fixed.acidity + volatile.acidity +
##     citric.acid + residual.sugar + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + density + pH + sulphates + alcohol,
##     df)
## 11 Variables  (and intercept)
##                       Forced in Forced out
## fixed.acidity             FALSE      FALSE
## volatile.acidity          FALSE      FALSE
## citric.acid               FALSE      FALSE
## residual.sugar            FALSE      FALSE
## chlorides                 FALSE      FALSE
## free.sulfur.dioxide       FALSE      FALSE
## total.sulfur.dioxide      FALSE      FALSE
## density                   FALSE      FALSE
## pH                        FALSE      FALSE
## sulphates                 FALSE      FALSE
## alcohol                   FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          fixed.acidity volatile.acidity citric.acid residual.sugar
## 1  ( 1 ) " "           " "              " "         " "
## 2  ( 1 ) " "           " "              " "         " "
## 3  ( 1 ) " "           " "              " "         " "
## 4  ( 1 ) "*"           " "              " "         "*"
## 5  ( 1 ) "*"           " "              " "         "*"
## 6  ( 1 ) "*"           "*"              " "         "*"
## 7  ( 1 ) "*"           "*"              " "         "*"
## 8  ( 1 ) "*"           "*"              " "         "*"
##          chlorides free.sulfur.dioxide total.sulfur.dioxide density pH
## 1  ( 1 ) " "       " "                 " "                   " "     " "
## 2  ( 1 ) "*"       " "                 " "                   " "     " "
## 3  ( 1 ) "*"       " "                 " "                   "*"     " "
## 4  ( 1 ) " "       " "                 " "                   "*"     " "
## 5  ( 1 ) " "       " "                 " "                   "*"     "*"
## 6  ( 1 ) " "       " "                 " "                   "*"     "*"
## 7  ( 1 ) " "       " "                 " "                   "*"     "*"
## 8  ( 1 ) " "       " "                 "*"                   "*"     "*"
##          sulphates alcohol
## 1  ( 1 ) " "       "*"
## 2  ( 1 ) " "       "*"
## 3  ( 1 ) " "       "*"
## 4  ( 1 ) " "       "*"
## 5  ( 1 ) " "       "*"
## 6  ( 1 ) "*"       " "
## 7  ( 1 ) "*"       "*"
```

```
## 8  ( 1 ) "*"       "*"
```

From this we can see that for size 3, the best variables are chlorides, density, and alcohol. For size 5, the best attributes are fixed acidity, residual sugar, density, pH, and alcohol. We can now model again accounting for the two different sizes.

```
tree3=tree(qualityCat~chlorides+density+alcohol, data=train)
tree5=tree(qualityCat~fixed.acidity+residual.sugar+density+pH+alcohol, data=train)
p3=predict(tree3, test, type="class")
mean(p3==test$qualityCat)
```

```
## [1] 0.76
```

```
p5=predict(tree5, test, type="class")
mean(p5==test$qualityCat)
```

```
## [1] 0.76
```

As it turns out, both models give us the same accuracy score. Why is this? We can check out summaries of the models to find out.

```
summary(tree3)
```

```
##
## Classification tree:
## tree(formula = qualityCat ~ chlorides + density + alcohol, data = train)
## Variables actually used in tree construction:
## [1] "alcohol"
## Number of terminal nodes:  3
## Residual mean deviance:  1.138 = 5170 / 4544
## Misclassification error rate: 0.2281 = 1037 / 4547
```

```
summary(tree5)
```

```
##
## Classification tree:
## tree(formula = qualityCat ~ fixed.acidity + residual.sugar +
##     density + pH + alcohol, data = train)
## Variables actually used in tree construction:
## [1] "alcohol"
## Number of terminal nodes:  3
## Residual mean deviance:  1.138 = 5170 / 4544
## Misclassification error rate: 0.2281 = 1037 / 4547
```

Unexpectedly, both models only ended up using alcohol as a predictor. Surprisingly, the accuracies here are equal to the accuracy from the all-inclusive model.

We can try one more method, the bagging method, just as an alternative to this tree method.

```
mod.bag=randomForest(qualityCat~fixed.acidity+volatile.acidity+citric.acid
                     +residual.sugar+chlorides+free.sulfur.dioxide
                     +total.sulfur.dioxide+density+pH+sulphates+alcohol,
                     data=train, mtry=11, importance=T)
mod.bag
```

```
##
## Call:
##  randomForest(formula = qualityCat ~ fixed.acidity + volatile.acidity +      citric.acid + residual.s
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##         OOB estimate of  error rate: 15.86%
## Confusion matrix:
##         Average Good Poor class.error
## Average    3332  152   14  0.04745569
## Good        399  475    0  0.45652174
## Poor        154    2   19  0.89142857
```

Once again, we see that this type of method, using 500 trees, results in a very good "Average" prediction, an okay "Good" prediction, and a very underwhelming "Poor" prediction.

```
p.bag=predict(mod.bag, test)
mean(p.bag==test$qualityCat)
```

```
## [1] 0.8410256
```

```
table(p.bag, test$qualityCat)
```

```
##
## p.bag     Average Good Poor
##   Average    1394  168   59
##   Good         78  235    1
##   Poor          4    0   11
```

However, looking at the mean accuracy of predictions, our accuracy is roughly 84%. This is 3% higher than our kNN classification accuracy!

Results & Conclusion

During this project, methods of linear regression, k-Nearest Neighbors, and Random Forest were used.

It has been seen that linear regression is not successful when regressing on this particular data set. This is primarily due to the subjective nature of wine review and a lack of correlation between variables; someone scoring a wine doesn't look at a particular sulphate count and decide "this is a 7 out of 10!"

When switching to the kNN method, splitting the data into score categories greatly enhancing our accuracy of prediction. We ended up with an 80% prediction rate that is relatively successful given that an overwhelming majority of the data were either rated 5 or 6.

When looking at Tree methodology, our prediction rate hovered around 77%. This is primarily because of how the Tree method works; it siphoned out perceived "useless" variables and only kept in the alcohol attribute. Using the bagging method with Random Forest, we achieved a success rate of roughly 84%. This is our highest prediction rate, and a pretty commendable one at that. Most of these variables had little to no correlation, so getting a rate that high is interesting to say the least.

To address some hang ups with this project, my only real concerns are the average performance for "Good" wines and the poor performance for "Poor" wines. I'm not entirely sure how I could've addressed this, but it seems to be a problem stemming from the dataset used. If there were more examples of "Good" or "Poor" wines, perhaps the confusion matrices would've been different. Otherwise, it is satisfactory to know that certain methods can be useful to predict mostly uncorrelated attributes.

Works Cited

1. "Vinho Verde, Comissao De Viticultura Da Regiao Dos Vinhos Verdes, 2019, www.vinhoverde.pt/en/history-of-vinho-verde

2. Panizzo, Daniel S. "Wine Quality." Kaggle, Kaggle Inc., 29 Oct. 2017, www.kaggle.com/danielpanizzo/wine-quality