Selection of Premium Red Wines

David List

04/10/2023

1 Introduction

The Wine Quality Data Set available from the UCI (University of California, Irvine) Machine Learning Repository provides an interesting opportunity to explore both classification and regression machine learning techniques on the same data set.

Two separate sets of data are available, one for red wines and one for white. All of the wines come from the northwest region of Portugal and are designated as *vihno verde* wines which appears to be a protected name that may only be used to refer to wines coming from that specific region of Portugal, perhaps similar to "Scotch" and "Champagne" which may only be used to refer to whiskies from Scotland and sparkling wines from the Champagne region of France respectively.

The red and white data sets contain 1599 and 4898 records respectively, each with the following attributes:

Fixed Acidity - Amount of tartaric acid in g/L, although more generally fixed acids in wine include malic, citric, and succinic acid as well.

Volatile Acidity - Amount of acetic acid (vinegar) in g/L. More generally volatile acids may also include lactic, formic, butiric, and propionic acids. These acids are associated with spoilage in wine.

Citric Acid - Amount of citric acid in g/L. Citric acid is normally present in small amounts in grapes but may be added to wine to increase acidity.

Residual Sugar - Typically the amount of natural sugar in g/L remaining in the wine after fermentation completes. Some countries allow additional sugar to be added, but this practice has fallen out of favor with critics.

Chlorides - Amount of sodium chloride in g/L.

Free Sulfur Dioxide - Amount of sulfites available to react in mg/L. Sulfites (sulfur dioxide or $S0_2$) are often added to wine as a preservative, but some also occurs naturally.

Total Sulfur Dioxide - Total amount of free and already reacted (bound) sulfites in mg/L.

Density - Measured in g/ml.

pH - Measurement of the acidity of the wine (lower pH is more acidic.)

Sulphates - A different form of naturally occurring sulfur (SO_4) that depends on the composition of the soil in which the grapes are grown.

Alcohol - Percent alcohol by volume.

Quality - The quality of the wine from 0 to 10 (10 being the best) as measured by assessors using blind tastes. This is the value the algorithms in this paper attempt to predict.

This paper will focus specifically on the red wines part of the data set and attempt to identify a high precision algorithm in the R programming language for selecting premium wines. To do this, the paper will explore the classification and regression capabilities of the following machine learning algorithms before selecting the one with the best performance:

- Linear Regression
- Classification with k-Nearest Neighbors
- Regression with k-Nearest Neighbors
- Classification with Random Forests (Rborist)
- Regression with Random Forests (Rborist)

Initially, the paper will investigate the overall relative performance of each algorithm against the others, but in the second half of the analysis section, the paper will switch to focusing on the main goal of premium wine selection with high precision. Here precision is defined with the following equation:

$$precision = \frac{TP}{TP+FP}$$

where TP and FP stand for *true positives* and *false positives* respectively.

If not obvious, the motivation for this paper relates to the author's preference for drinking more better tasting red wines and fewer poor tasting ones out of the many, many red wines available to consumers.

2 Analysis

2.1 Data Cleaning

After loading the data into an R data.frame, the column names were first updated to replace spaces with underscores to make the coding effort simpler. See the following code illustrating these steps.

```r
# Load the red wines.
wines_full <- read_delim('winequality-red.csv',
                    delim = ';',
                    col_types = cols(
                      `fixed acidity` = col_double(),
                      `volatile acidity` = col_double(),
                      `citric acid` = col_double(),
                      `residual sugar` = col_double(),
                      chlorides = col_double(),
                      `free sulfur dioxide` = col_number(),
                      `total sulfur dioxide` = col_number(),
                      density = col_double(),
                      pH = col_double(),
                      sulphates = col_double(),
                      alcohol = col_double(),
                      quality = col_number()))

# Get rid of the spaces from the column names.
colnames(wines_full) <- c('fixed_acidity', 'volatile_acidity', 'citric_acid',
                          'residual_sugar', 'chlorides', 'free_sulfur_dioxide',
                          'total_sulfur_dioxide', 'density', 'pH', 'sulphates',
                          'alcohol', 'quality')
```

Next, the values for each column or factor of data were inspected to make sure the ranges made sense for the given units and that no values were missing (represented by NAs in the first column or by out of range 9999.99 values, etc.) Note this procedure was greatly simplified by comparing the calculated values against the similar Table 1 of Cortez et. al. containing similar values for the data set and which was also the source used for identifying the units for each factor in the data set.

The results of this analysis are shown in the following table.

Description	NAs	Min	Max	Mean	Standard Deviation
Fixed Acidity (g/L)	0	4.60000	15.90000	8.31964	1.74110
Volatile Acidity (g/L)	0	0.12000	1.58000	0.52782	0.17906
Citric Acid (g/L)	0	0.00000	1.00000	0.27098	0.19480
Residual Sugar (g/L)	0	0.90000	15.50000	2.53881	1.40993
Chlorides (g/L)	0	0.01200	0.61100	0.08747	0.04707
Free Sulfur Dioxide (mg/L)	0	1.00000	72.00000	15.87492	10.46016
Total Sulfur Dioxide (mg/L)	0	6.00000	289.00000	46.46779	32.89532
Density (g/mL)	0	0.99007	1.00369	0.99675	0.00189
pH	0	2.74000	4.01000	3.31111	0.15439
Sulphates (g/L)	0	0.33000	2.00000	0.65815	0.16951
Alcohol (% volume)	0	8.40000	14.90000	10.42298	1.06567
Quality	0	3.00000	8.00000	5.63602	0.80757

It is interesting the results are not exactly the same as Cortez et. al., but the differences are not significant.

2.1.1 Duplicate values

In the course of performing the k-Nearest Neighbors algorithms below in Sections 2.6 - 2.8, it became clear the algorithms performed best with k equal to one which is exactly what might be expected with a data set that has duplicate values. Further investigation revealed the data had 240 duplicate rows representing about 15% of the total number of rows. The question then became whether or not these duplicates represent legitimate values and whether or not to remove them from the data set. The following from Cortez et. al. describes the process for creating the database.

> During the preprocessing stage, the database was transformed in order to include a distinct wine sample (with all tests) per row. To avoid discarding examples, only the most common physicochemical tests were selected. Since the red and white tastes are quite different, the analysis will be performed separately, thus two datasets were built with 1599 red and 4898 white examples. Table 1 presents the physicochemical statistics per dataset. Regarding the preferences, each sample was evaluated by a minimum of three sensory assessors (using blind tastes), which graded the wine in a scale that ranges from 0 (very bad) to 10 (excellent). The final sensory score is given by the median of these evaluations.

Given that description, and that only the median sensory scores (quality) were included, there should only ever be one record for a given wine. Therefore, to be legitimate the duplicate records would need to be for different wines but with coincidentally identical readings for all 12 factors. If that were true and they are in fact different wines, then there should be a substantial number of records that are identical except for the quality rating. Presumably different wines should at least occasionally receive different ratings even with identical measured values. However, removing quality from the search for unique rows returned the exact same 240 records suggesting the duplicate records are actually just duplicate records for the same wines so they were removed before proceeding.

The following code was used to perform the analysis.

```r
# The data has 240 duplicate rows.  The question is whether these duplicates
# are legitimate data points or actual duplicates.  To help answer, count the
# number of unique rows including the quality (wine rating) column and not.

initial_count <- nrow(wines_full)
initial_count
unique_count <- nrow(unique(wines_full))
unique_count
```

```
unique_count_minus_quality <- nrow(unique((wines_full %>%
                                          select('fixed_acidity', 'volatile_acidity',
                                                 'citric_acid', 'residual_sugar',
                                                 'chlorides', 'free_sulfur_dioxide',
                                                 'total_sulfur_dioxide', 'density',
                                                 'pH', 'sulphates', 'alcohol'))))
unique_count_minus_quality
```

With the following results:

Description	Value
Initial Count	1599
Unique Count	1359
Unique Count w/out Quality	1359

2.2 Qf and Grade

To assist in the analysis below, two additional attributes were added to the dataset. *Qf* was added as a factorized version of quality to aid classification, and *grade* was added with the values *Regular* and *Premium* to signify wines with a quality value less than 7 vs. 7 or more respectively.

```
# Add qf as a factor version of quality
wines_full <- wines_full %>% mutate(qf = factor(quality))

# Add grade as a factor with quality < 7 Regular and >= 7 Premium
wines_full <- wines_full %>%
  mutate(grade = factor(ifelse(quality < 7, 'Regular', 'Premium')))
```

2.3 Cross Validation

The original red wines data set was broken by random sampling into 3 different groups in order to implement what was effectively a double holdout strategy. First, the full data set, referred to in the code as *wines_full*, was split into a *wines* training set, and a *final_test* holdout set. The latter was set aside to be used exclusively to evaluate the final model. Next, the *wines* training set was split again into a *wines_train* training set and a *wines_test* validation set so that different models and model parameters could be evaluated without using final_test. Per the course material from Prof. Irizarry, a typical split is to use 10-20% for the test set and 80-90% for the training set. Because the red wines data set is relatively small, 20% was chosen at each step for the test/holdout sets to maximize their usefulness while only minimally impacting the training sets.

The following code was used to create the three sets.

```
# Create the final training and test sets.
# Variables:
#     wines_full -  The full set of loaded wines w/duplicates removed.  Not modified
#                   below.
#     final_test -  The final test set used only for calculating the final results.
#     wines -       The final training set.  This is split again into wines_train and
#                   wines_test to select the model and model parameters.
#     wines_train - Cross validation training set.
#     wines_test -  Cross validation test set.
#
set.seed(1, sample.kind = 'Rounding')
test_index <- createDataPartition(wines_full$quality, times = 1, p = 0.2, list = FALSE)
final_test <- wines_full[test_index,]
```

```r
wines <- wines_full[-test_index,]

set.seed(1, sample.kind = 'Rounding')
test_index <- createDataPartition(wines$quality, times = 1, p = 0.2, list = FALSE)
wines_test <- wines[test_index,]
wines_train <- wines[-test_index,]
```

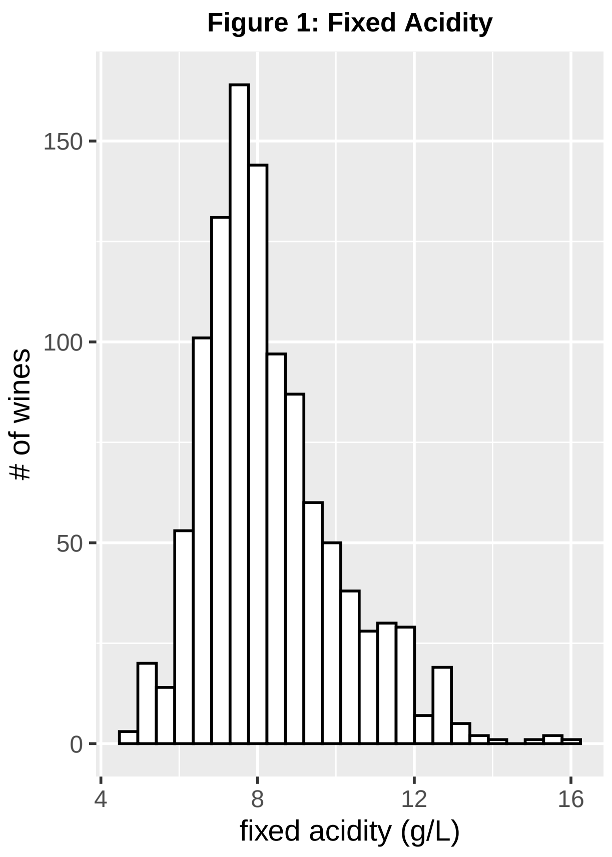
2.4 Data Exploration

The following sections explore the *wines* training set utilizing a variety of different strategies to help provide a better understanding of the data.

2.4.1 Feature Histograms

This section contains histograms of each of the 12 features contained in the data set. Each figure was created with code similar to the following:

```r
wines %>%
  ggplot(aes(fixed_acidity)) +
  geom_histogram(bins = 25, color = 'black', fill = 'white') +
  labs(title = 'Figure 1: Fixed Acidity', x = 'fixed acidity (g/L)', y = '# of wines') +
  theme(plot.title = element_text(face = 'bold', size = 10, hjust = 0.5))
```



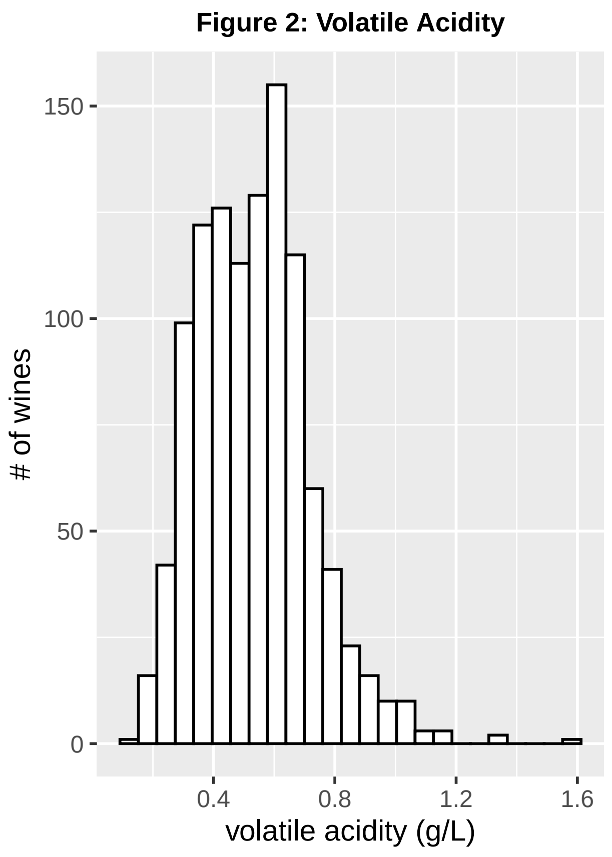


Figure 3: Citric Acid

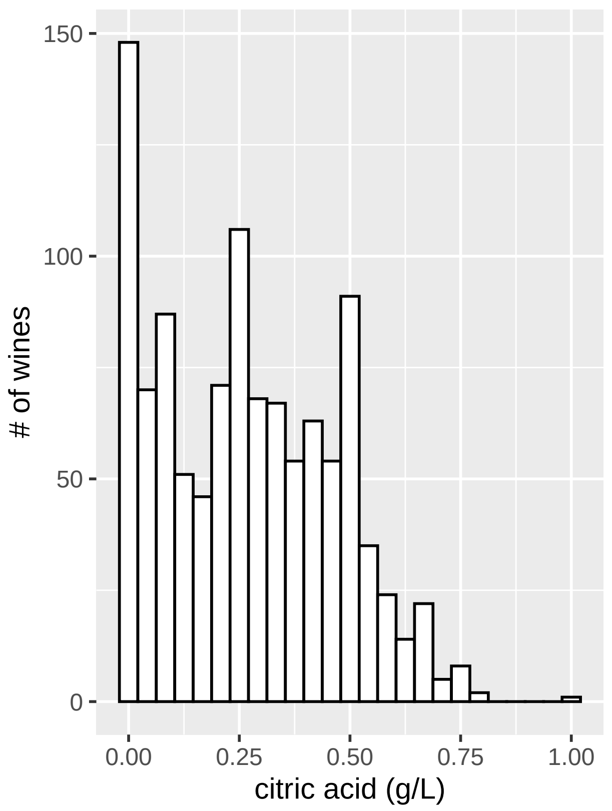


Figure 4: Residual Sugar

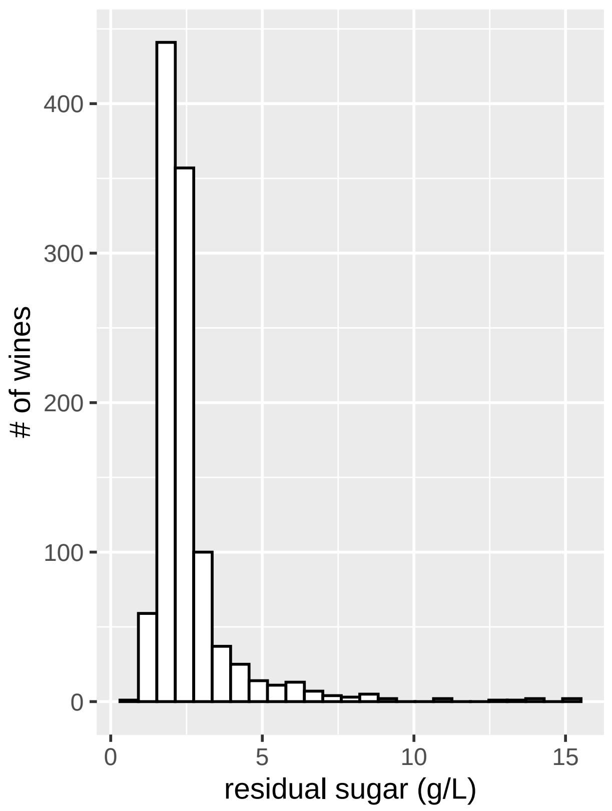


Figure 5: Chlorides

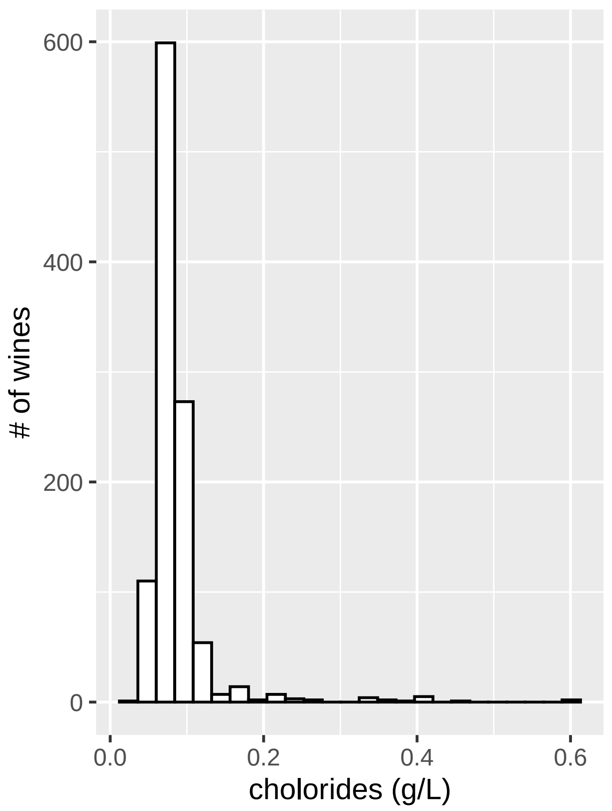


Figure 6: Free Sulfur Dioxide

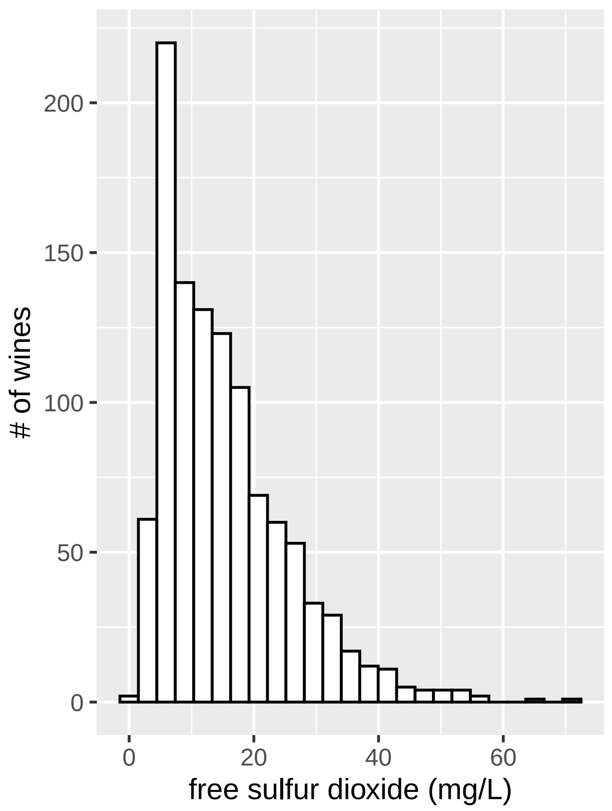


Figure 7: Total Sulfur Dioxide

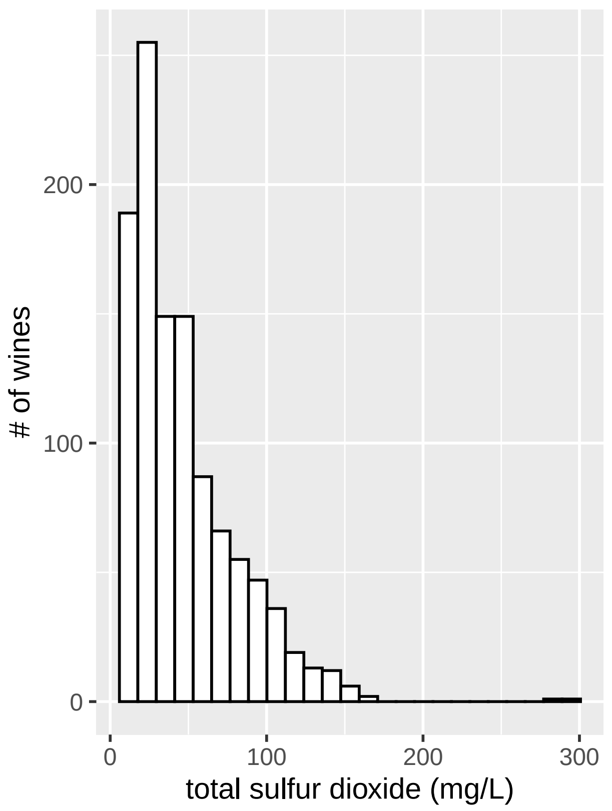


Figure 8: Density

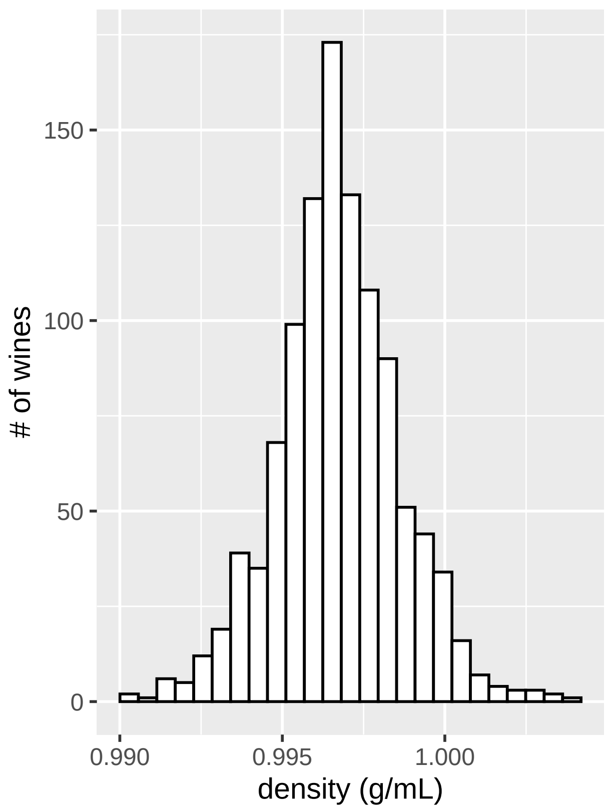


Figure 9: pH

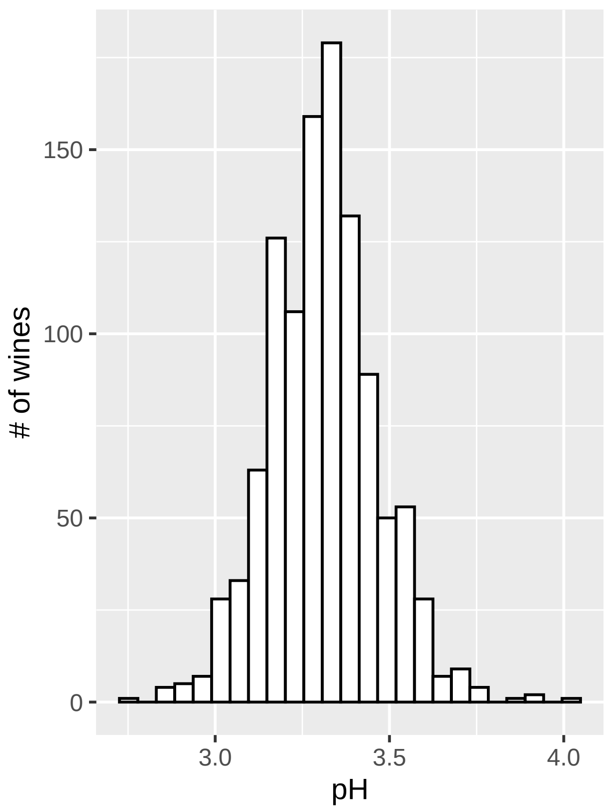
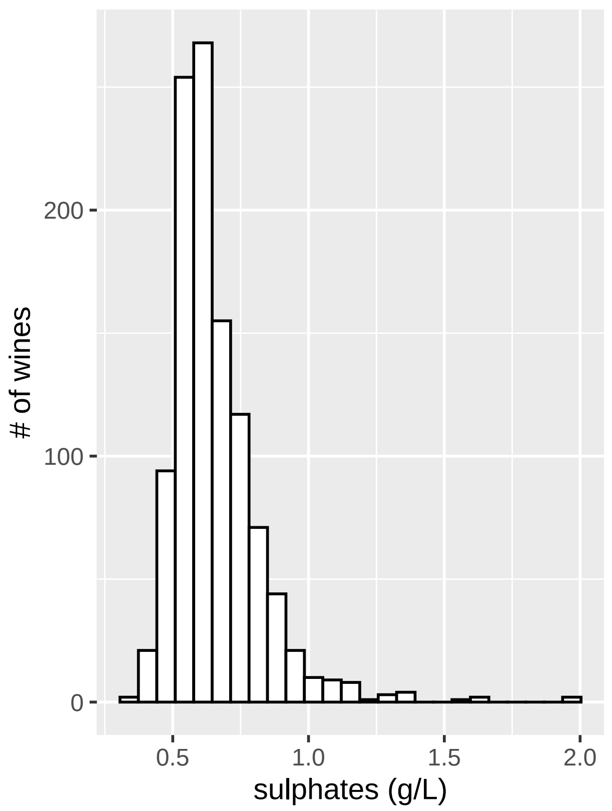
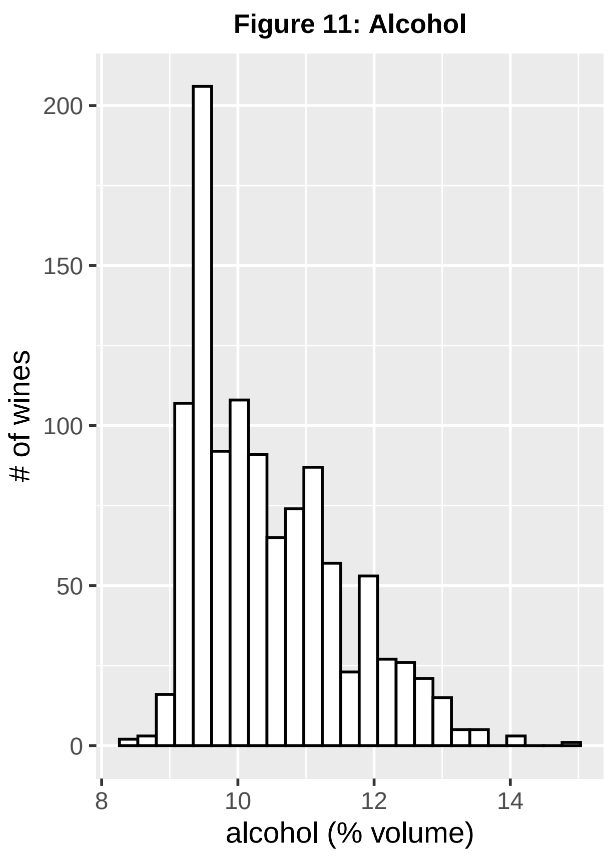
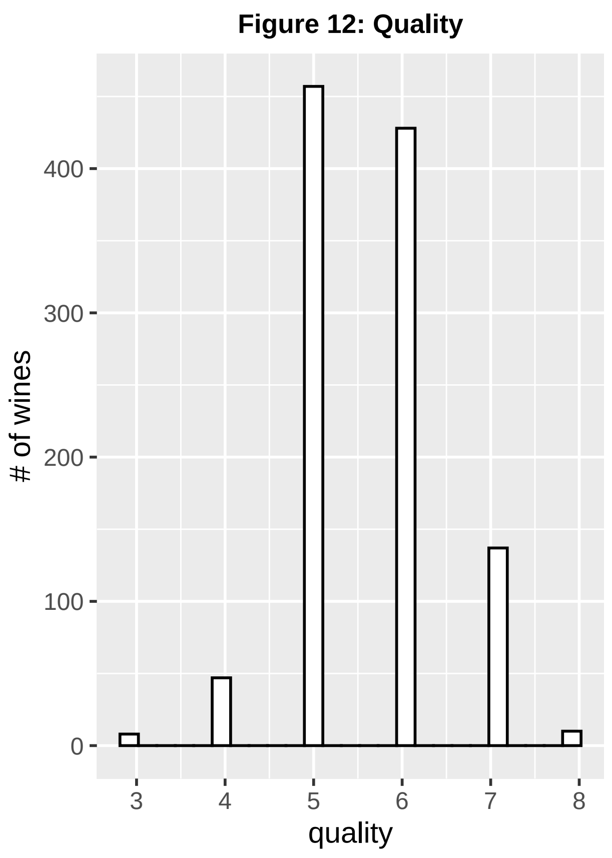


Figure 10: Sulphates







Note that among the 12, only density, pH, and quality appear to follow a normal distribution, although some of the others, like fixed acidity are close. Also note that for quality there are no values for 0, 1, 2, 9, or 10 even though each is a valid rating in the ratings scale.

2.4.2 Feature vs. Quality Box Plots

In this section box plots are shown for each of the 11 non-quality features against quality. Each figure was created with code similar to the following:

```
wines %>%
  ggplot(aes(qf, fixed_acidity)) + geom_boxplot() +
  labs(title = 'Figure 13: Fixed Acidity vs. Quality', x = 'quality',
       y = 'fixed acidity (g/L)') +
  theme(plot.title = element_text(face = 'bold', size = 10, hjust = 0.5))
```

Figure 13: Fixed Acidity vs. Quality

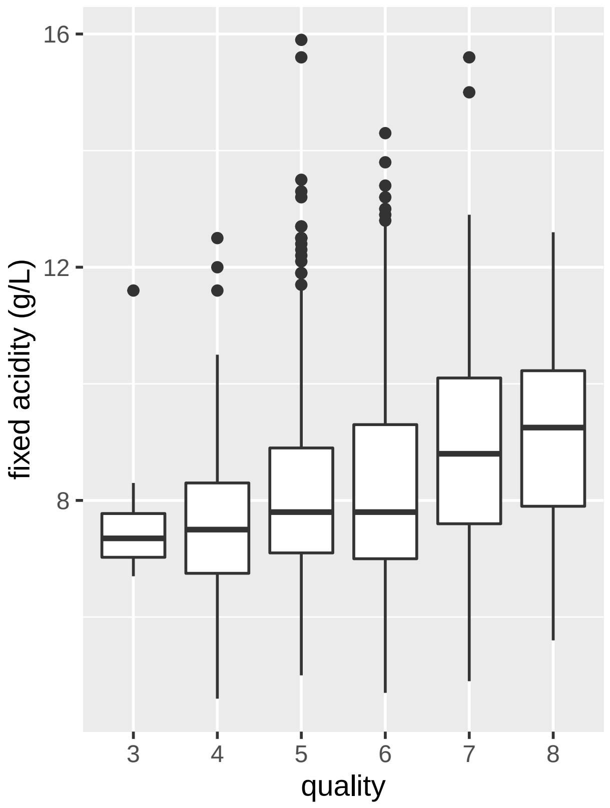


Figure 14: Volatile Acidity vs. Quality

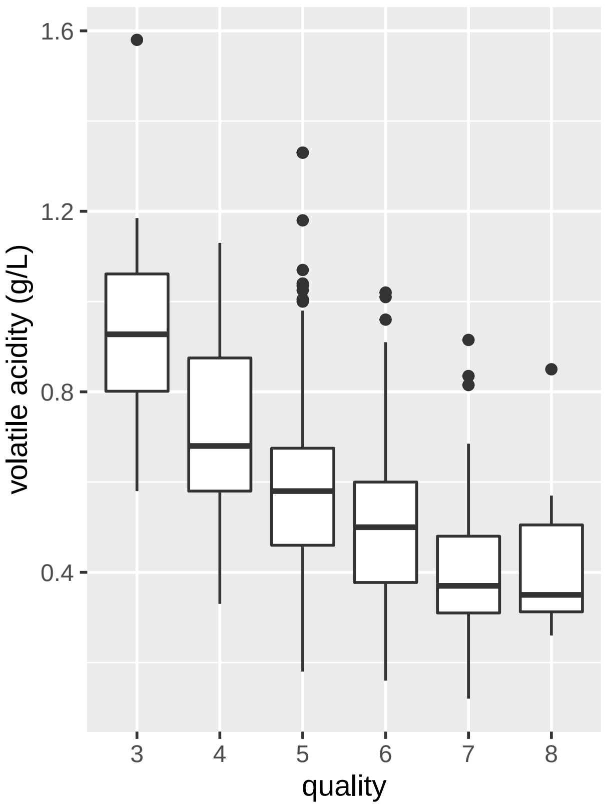


Figure 15: Citric Acid vs. Quality

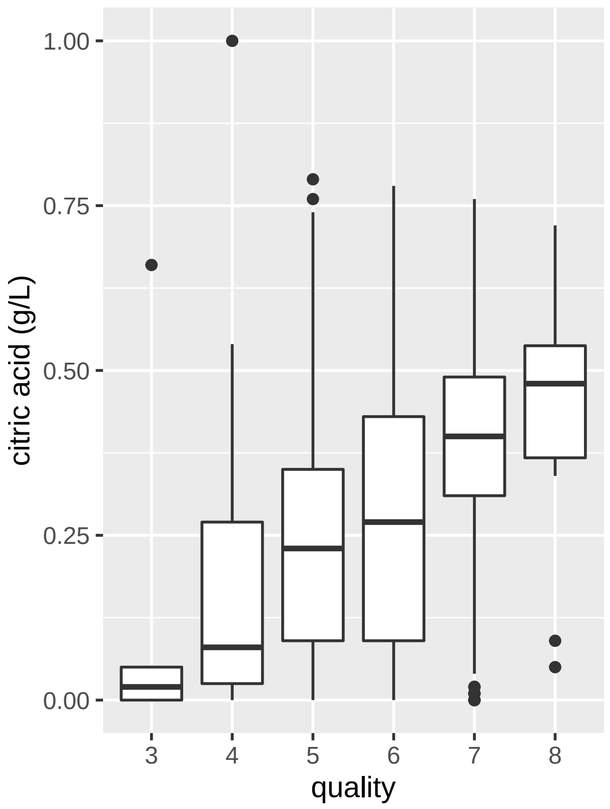


Figure 16: Residual Sugar vs. Quality

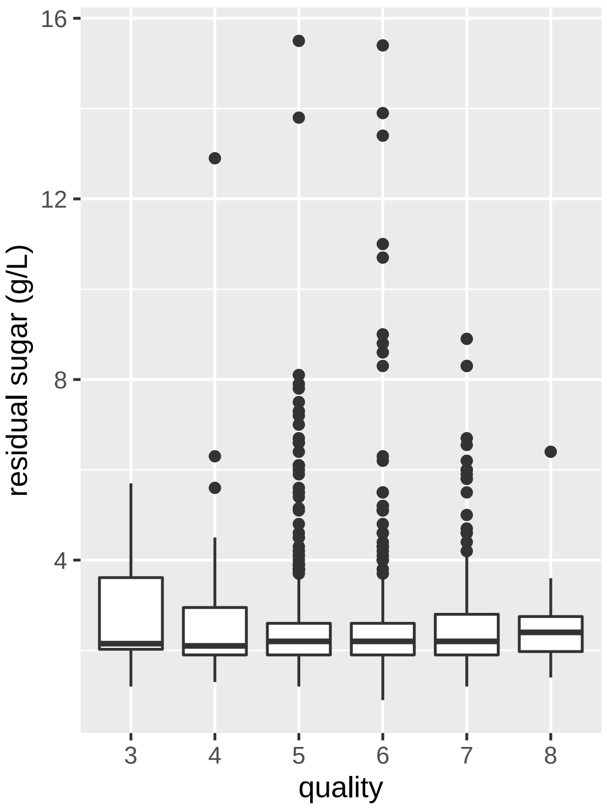


Figure 17: Chlorides vs. Quality

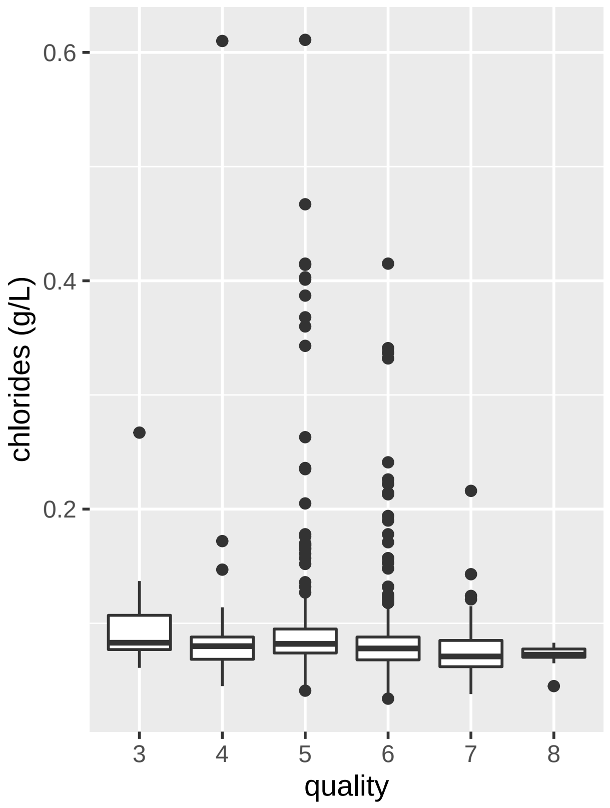


Figure 18: Free Sulfur Dioxide vs. Quality

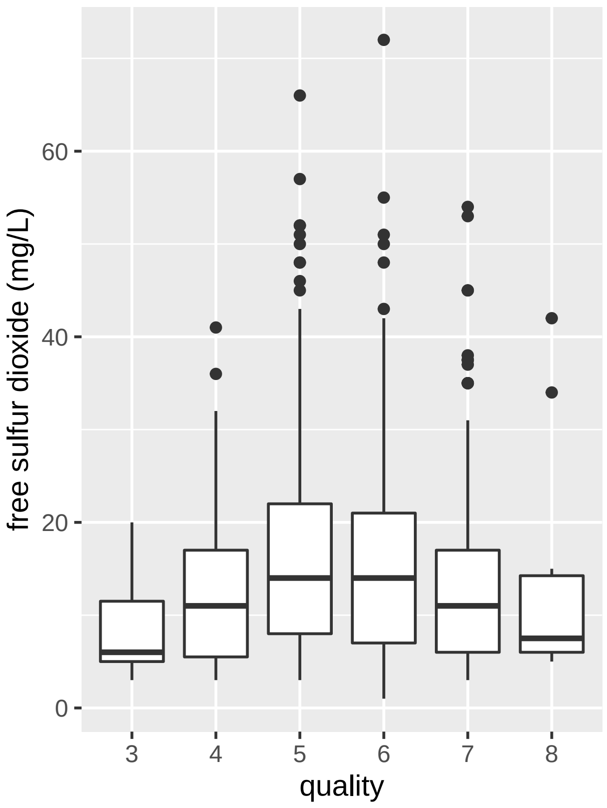


Figure 19: Total Sulfur Dioxide vs. Quality

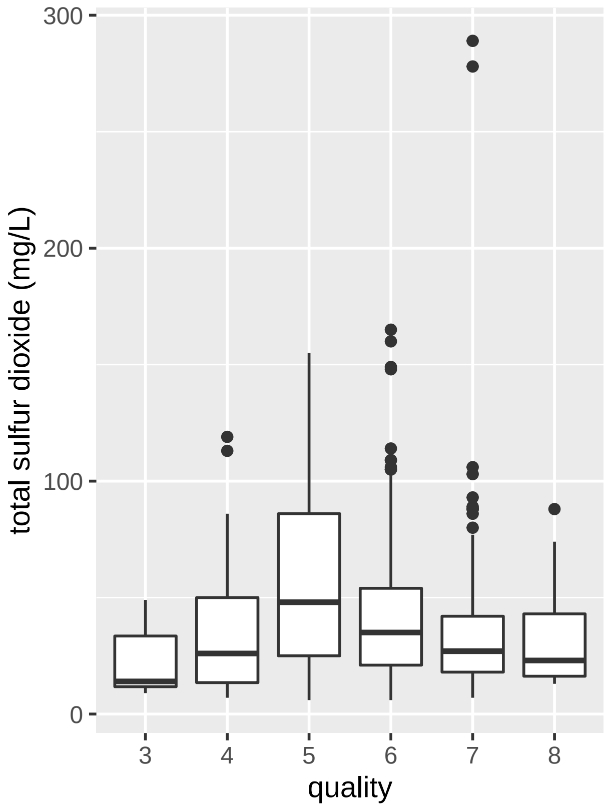


Figure 20: Density vs. Quality

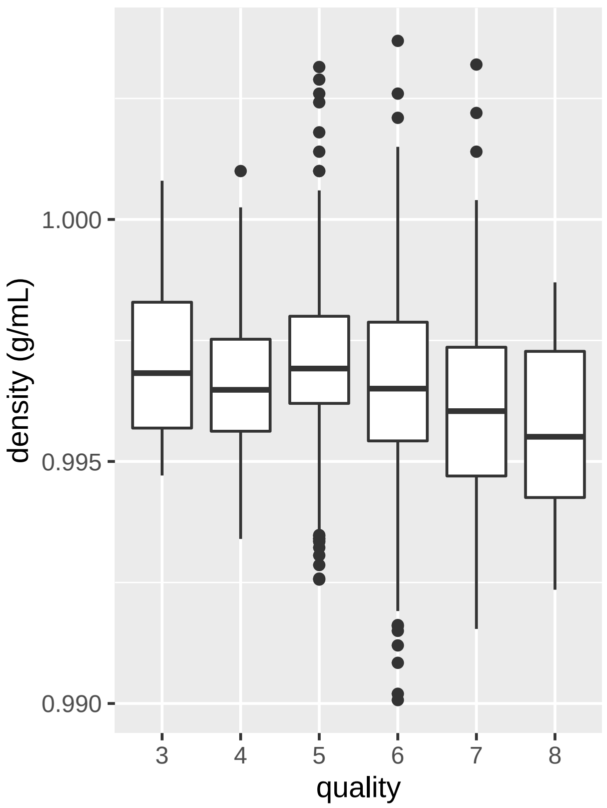


Figure 21: pH vs. Quality

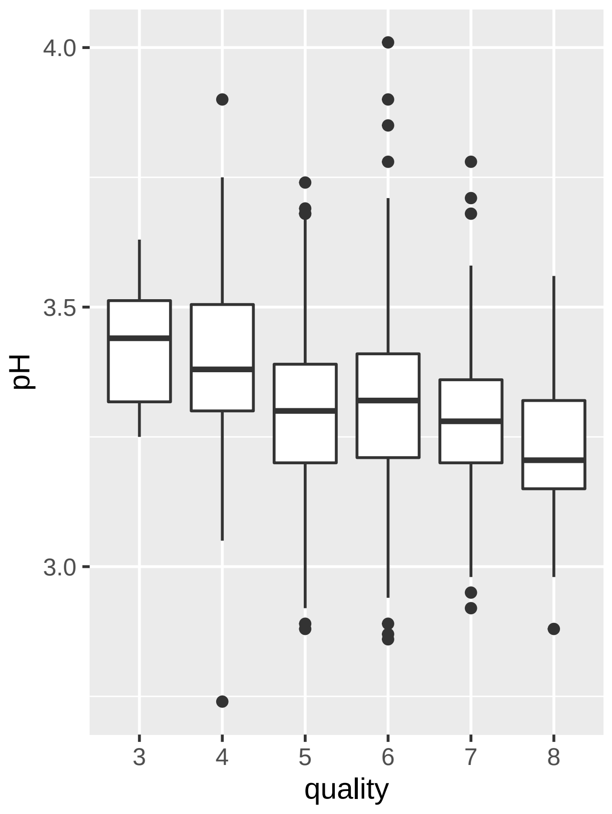


Figure 22: Sulphates vs. Quality

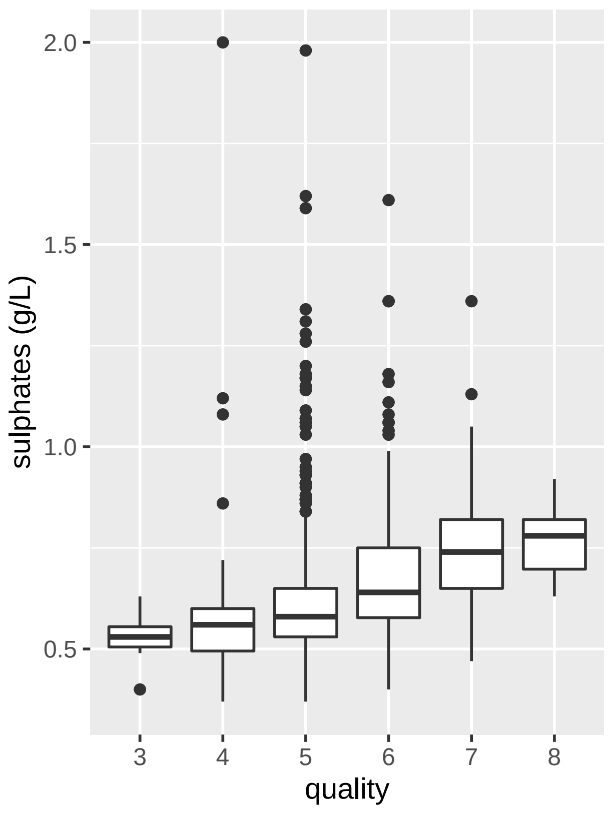
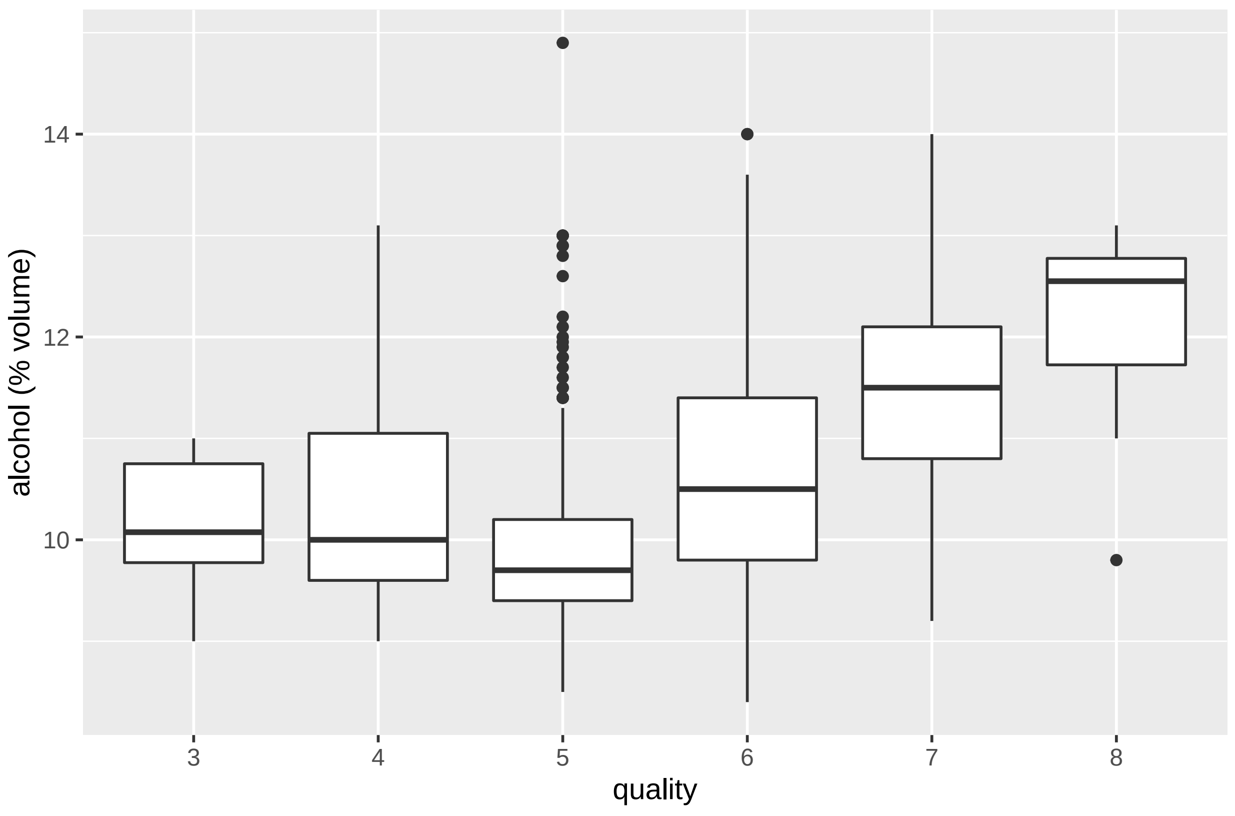


Figure 23: Alcohol vs. Quality



Note how many of the factors, such as fixed acidity, volatile acidity, citric acid, density, pH, sulphates, and alcohol appear to have a linear or close to linear relationship to quality.

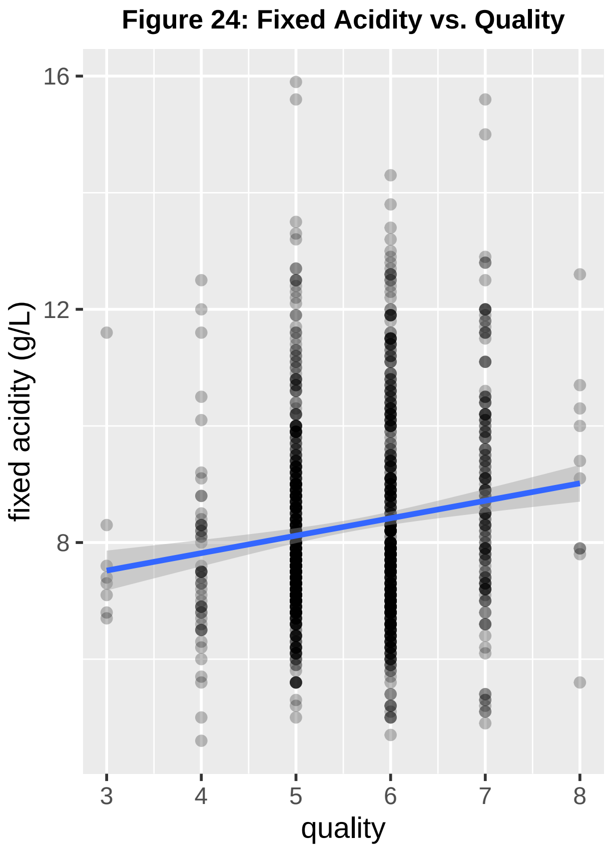
Also note how volatile acidity (vinegar) which is associated with wine spoilage has an inverse relationship with quality which is what might be intuitively expected.

2.4.3 Feature vs. Quality Scatter Plots

Here the same figures from the previous section are instead shown as scatter plots in combination with a best fit linear regression line. Each figure was created with code similar to the following:

```r
wines %>%
  ggplot(aes(quality, fixed_acidity)) + geom_point(alpha = 0.25) +
  geom_smooth(method = 'lm', formula = 'y ~ x') +
  labs(title = 'Figure 13: Fixed Acidity vs. Quality', x = 'quality',
       y = 'fixed acidity (g/L)') +
  theme(plot.title = element_text(face = 'bold', size = 10, hjust = 0.5))
```

One interesting thing to note is that a few of the features, such as alcohol and volatile acidity, appear more believably linear and closer to following a bivariate distribution when plotted as scatter plots versus box plots. Others, such as, citric acid, however, are less believable.



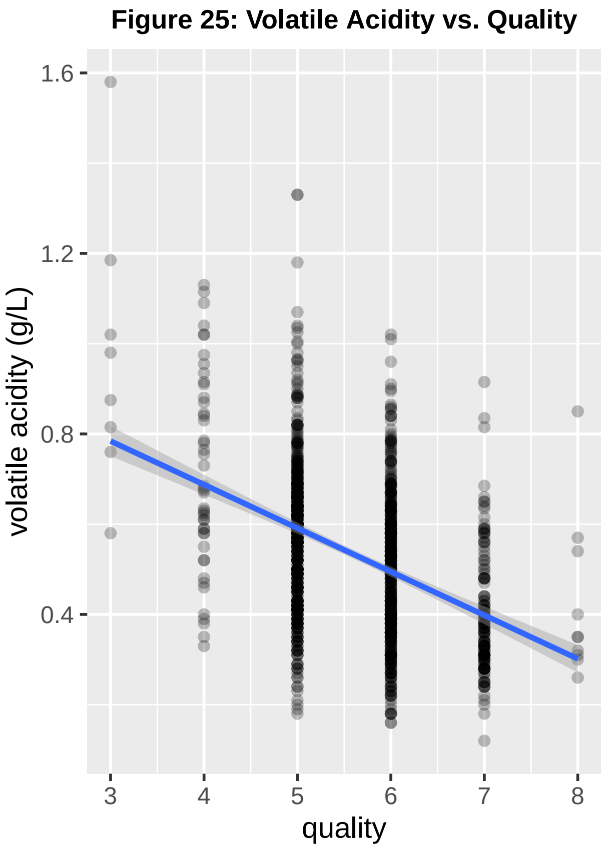


Figure 26: Citric Acid vs. Quality

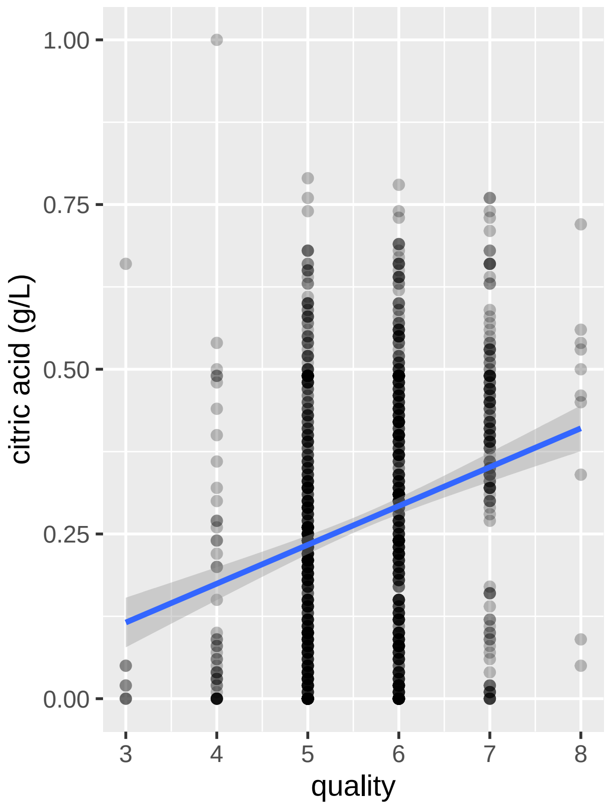


Figure 27: Residual Sugar vs. Quality

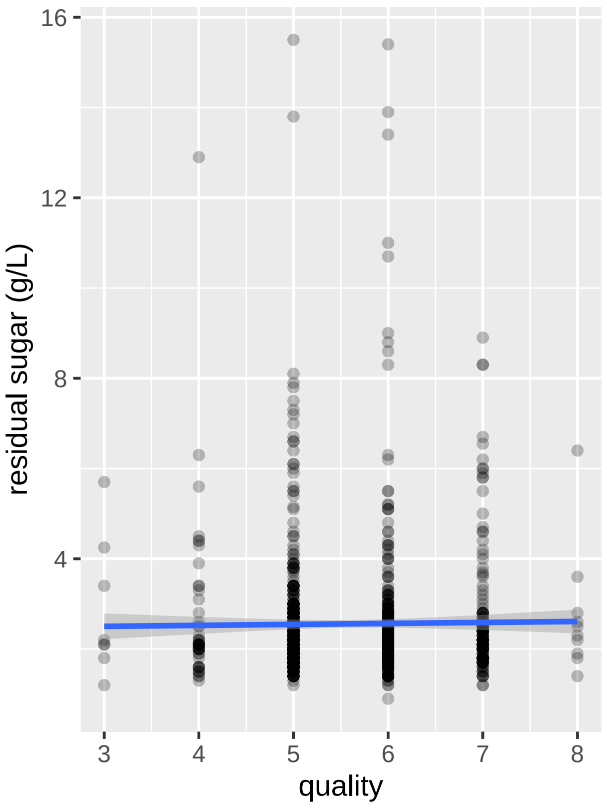


Figure 28: Chlorides vs. Quality

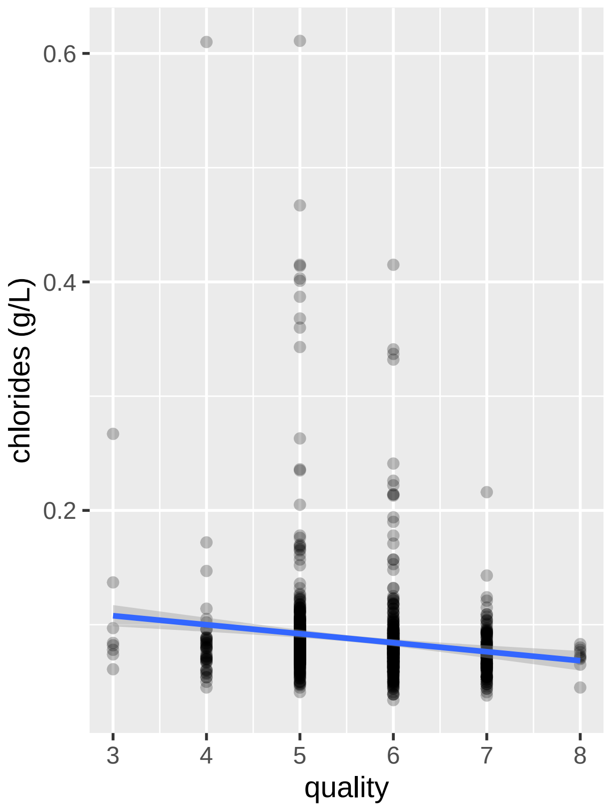


Figure 29: Free Sulfur Dioxide vs. Quality

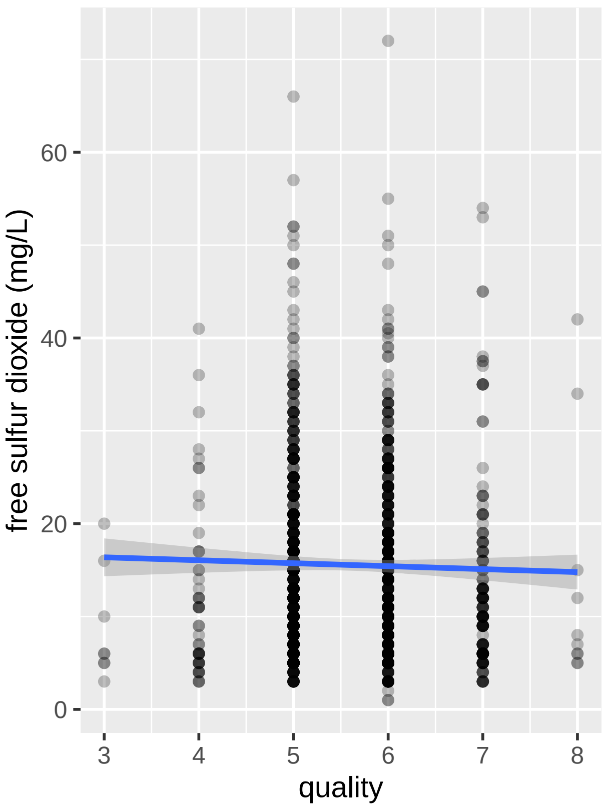


Figure 30: Total Sulfur Dioxide vs. Quality

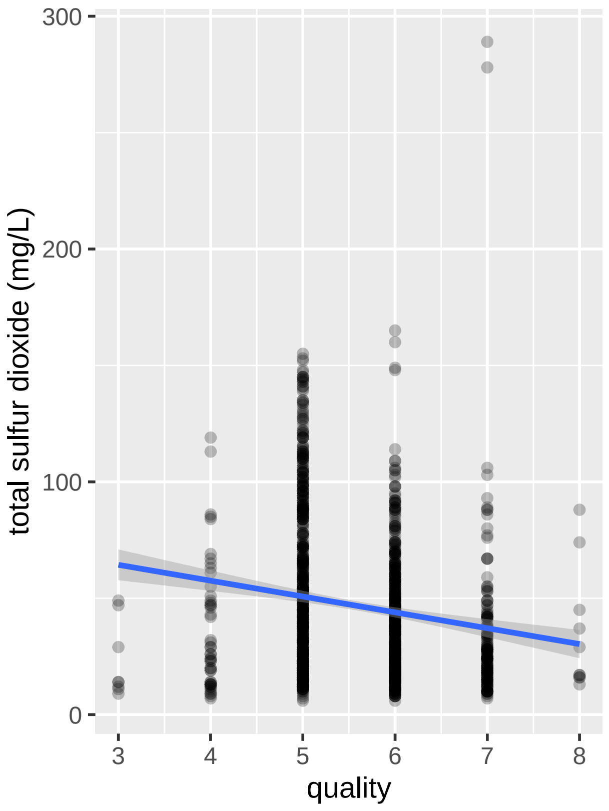


Figure 31: Density vs. Quality

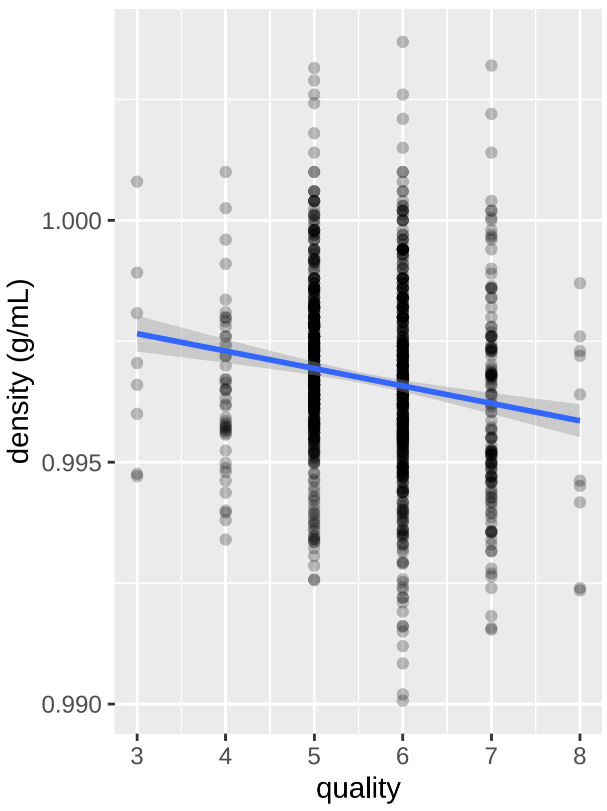


Figure 32: pH vs. Quality

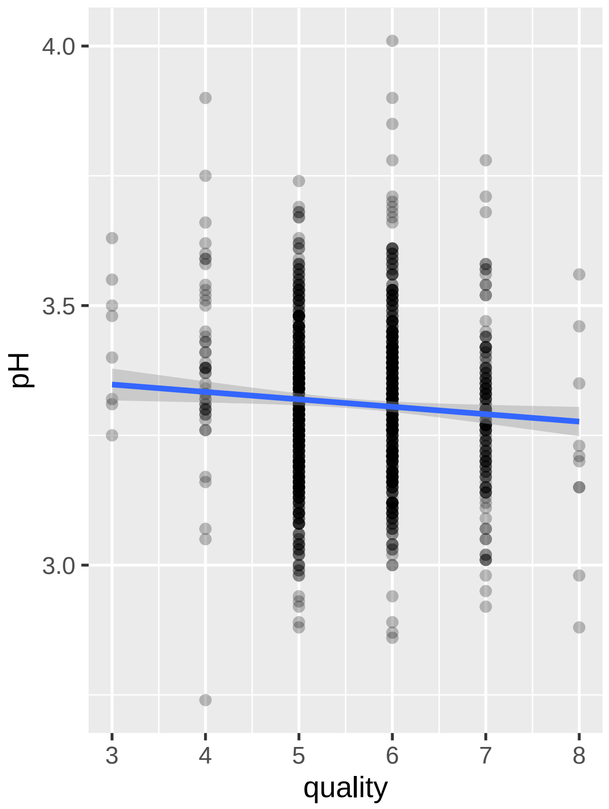


Figure 33: Sulphates vs. Quality

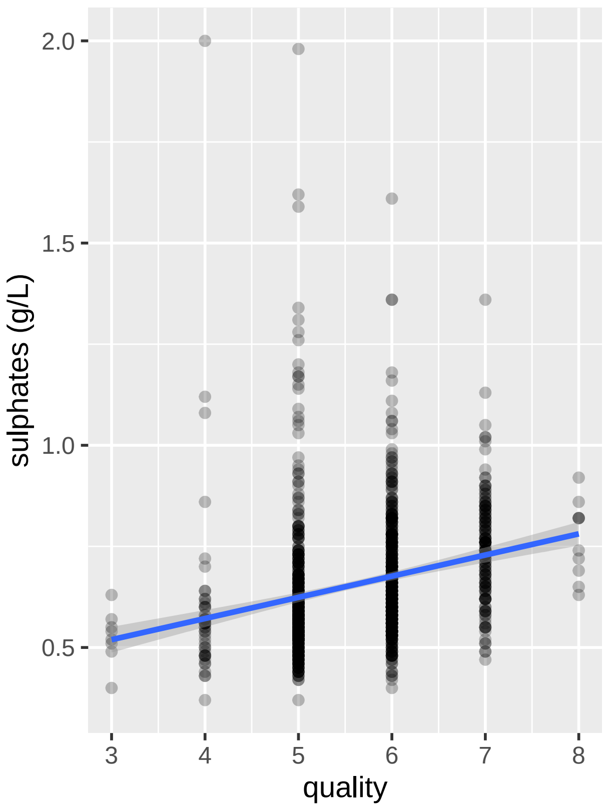
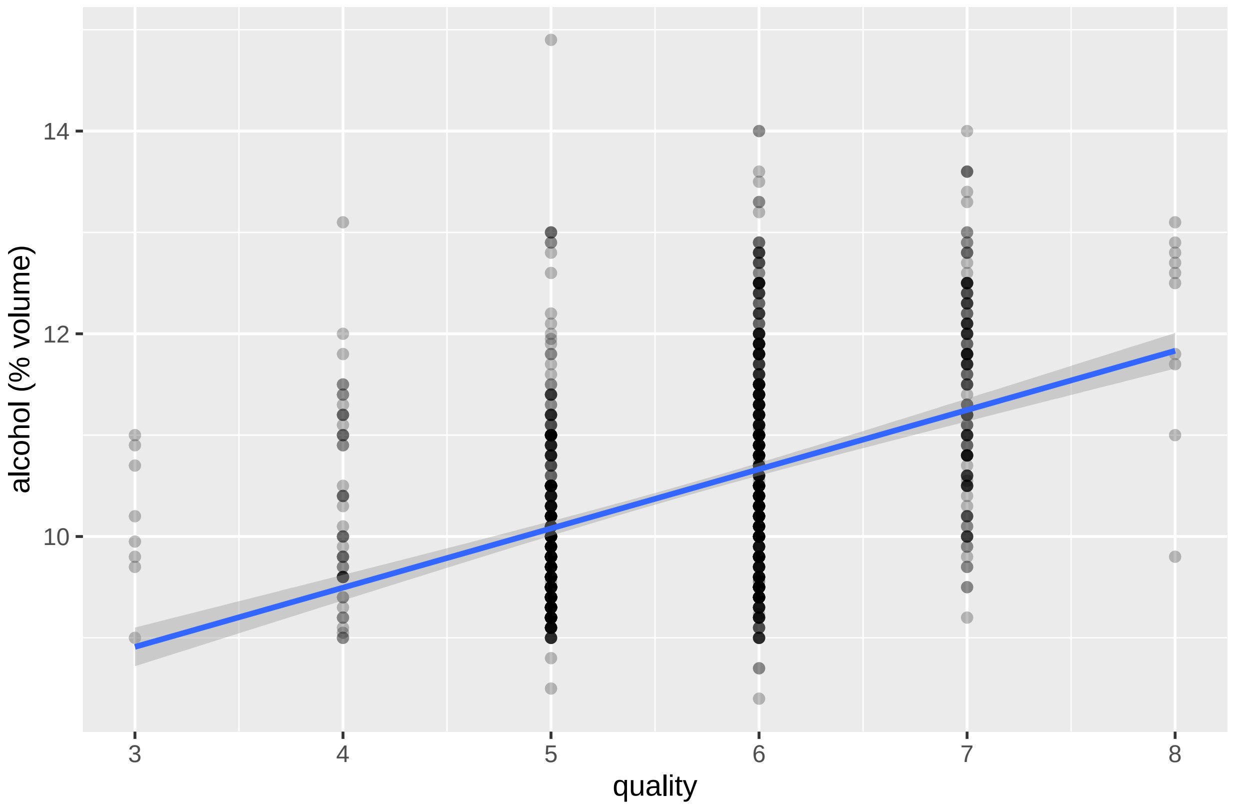


Figure 34: Alcohol vs. Quality



2.4.4 Feature/Quality Correlation

Remaining on the subject of linear regression, the following table lists the correlation coefficients of each feature vs. quality sorted by absolute value.

Description	Correlation Coefficient
Alcohol	0.44701
Volatile Acidity	-0.42798
Sulphates	0.25973
Citric Acid	0.24740
Total Sulfur Dioxide	-0.16578
Density	-0.15689
Fixed Acidity	0.14142
Chlorides	-0.13696
pH	-0.07568
Free Sulfur Dioxide	-0.02545
Residual Sugar	0.01234

The following graphs plot the top three factors in term of correlation coefficient against each other in order to illustrate the separation provided by each in terms of wine quality.

Figure 35: Volatile Acidity vs. Alcohol

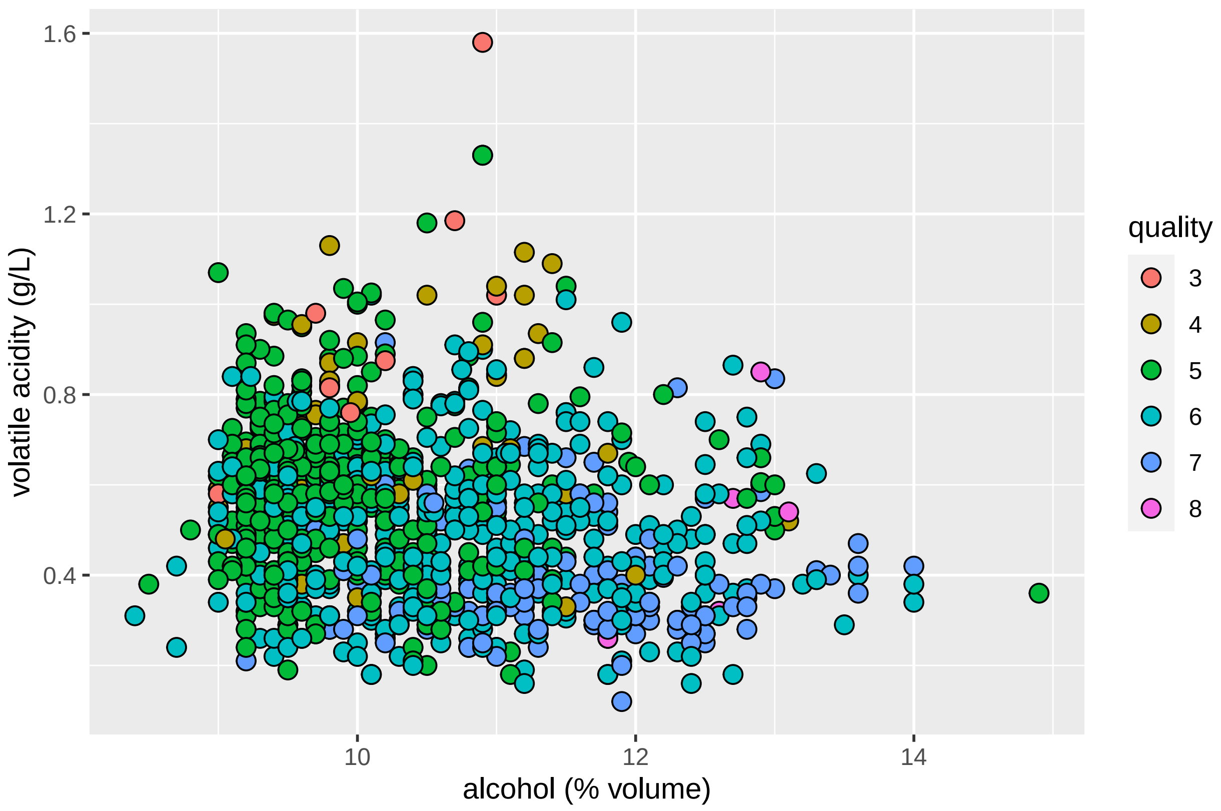


Figure 36: Sulphates vs. Alcohol

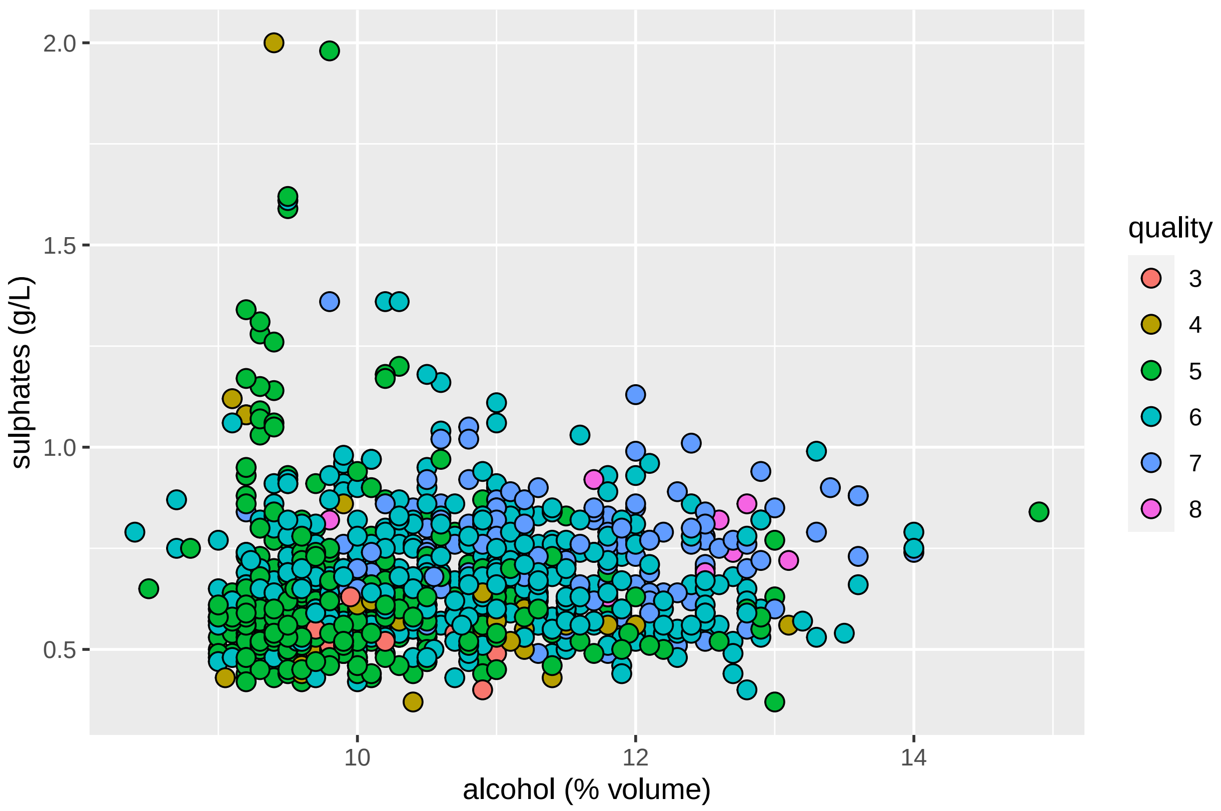
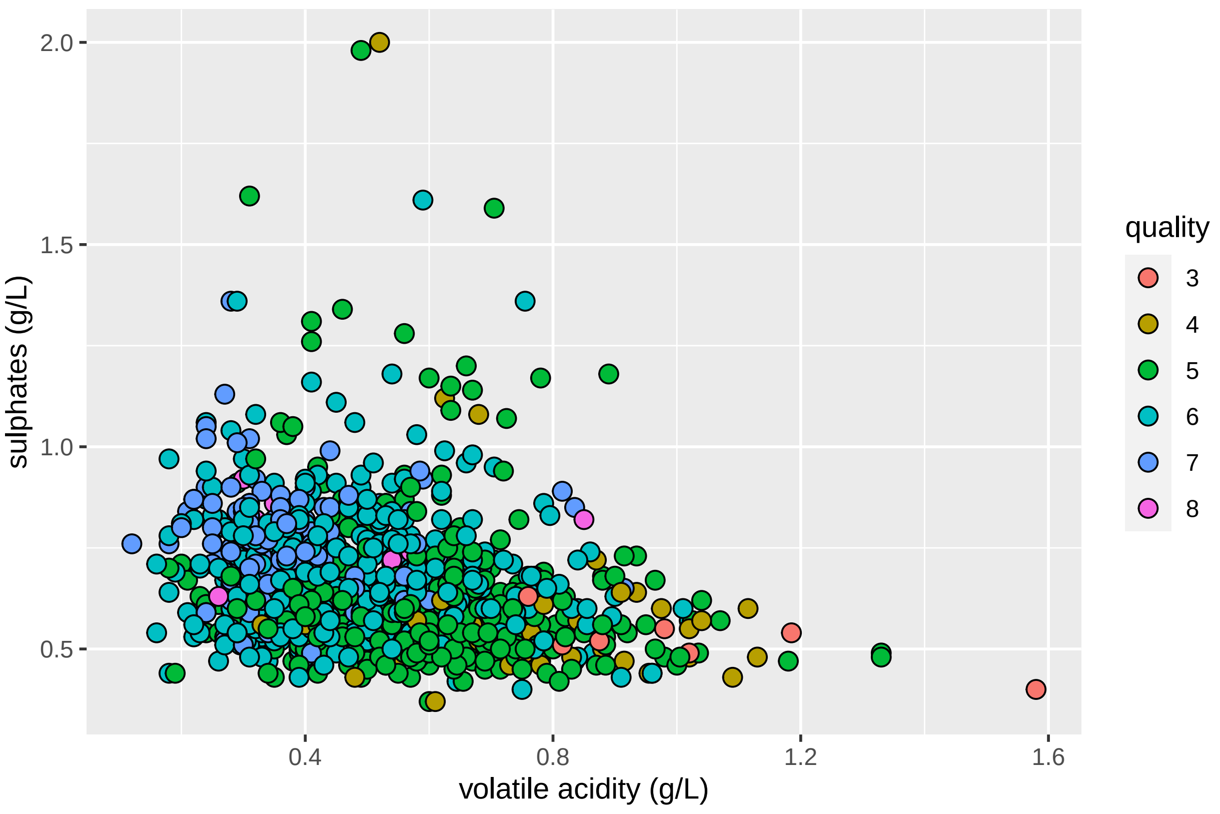


Figure 37: Sulphates vs. Volatile Acidity



While in each plot there are pretty clear regions of higher and lower wine quality the separation is pretty fuzzy with no clear dividing lines.

2.5 Linear Regression

The first algorithm run against the training set was linear regression. Several iterations were run starting with the factor with the highest correlation coefficient, alcohol, and adding the factor with the next highest correlation coefficient each iteration.

The following shows the code used for the first iteration which also records values for RMSE, accuracy, and accuracy +/-1 quality value:

```r
fit_lm1 <- lm(quality ~ alcohol, data = wines_train)
y_hat <- predict(fit_lm1, wines_test)
rmse_lm1 <- rmse(y_hat, wines_test$quality)
acc_lm1 <- mean(round(y_hat) == wines_test$quality)
accp1_lm1 <- mean(abs(round(y_hat) - wines_test$quality) <= 1)
```

The following table shows the results of the different model runs with the first row showing the results of always guessing the average value of 5.6082949 which, of course, rounds to 6. Linear regression doesn't naturally perform classification so the values for accuracy are determined by rounding the predicted value to the nearest integer and comparing that value with the value for quality. The final column displays how often the predicted value is within one of the value for quality as mention above. Note that based on the first row, 94% of the wines in the data set have quality values of 5, 6, or 7.

Description	RMSE	Accuracy	Accuracy +/-1
Guess Average	0.81228	0.39269	0.94064
Top Factor	0.74398	0.54795	0.94977
Top Two	0.69921	0.56164	0.97260
Top Three	0.67774	0.58447	0.97260
Top Four	0.67866	0.57534	0.97260
Top Five	0.67751	0.57991	0.97260
Top Six	0.67465	0.58904	0.97260
Top Seven	0.66762	0.57534	0.97717
Top Eight	0.66261	0.57078	0.97260
Top Nine	0.66128	0.57078	0.97260
Top Ten	0.65864	0.57534	0.97260
All Eleven	0.66095	0.56621	0.97260

With the exception of the entries for "Top 4" and "All Eleven", the RMSE values decrease with the addition of each new factor. The factors added at those entries are citric acid and residual sugar respectively. Looking at the scatter plot for citric acid, its distribution appears to be less of a bivariate normal distribution than many of the others so perhaps that's the reason the RMSE goes up as linear regression works best with a bivariate normal distribution. For residual sugar, the correlation coefficient is almost zero so it's unclear what benefit it would add even with a perfect distribution.

This same trend is not followed for the accuracy column which maxes out at the sixth factor. It is interesting that better accuracy does not automatically follow from a lower RMSE.

Although not shown in the table, an obvious potential improvement for RMSE would be to remove citric acid from the linear model, however, doing so actually increased the RMSE after the other factors were added.

2.6 Classification with k-Nearest Neighbors

The next algorithm run against the dataset was k-Nearest Neighbors which looks at the k closest points in a training set to predict the value of an unknown point. Unlike linear regression, k-Nearest Neighbors supports both classification and regression. Which algorithm is used is determined automatically and depends on whether the result to be predicted is a factor or a numeric value respectively.

Because k-Nearest Neighbors uses Euclidean distance to determine which values are closest, a naive approach would be to run the algorithm without consideration of the different ranges and units for each factor. As the author was naive at the time, this approach was the first attempt and was performed with code similar to the following:

```r
train(qf ~ alcohol, method = 'knn', data = wines_train,
             tuneGrid = data.frame(k = seq(1, 100, 1)))
y_hat <- predict(fit, wines_test, type = 'raw')
acc_knnc1 <- mean(y_hat == wines_test$qf)
accp1_knnc1 <- mean(abs(as.numeric(as.character(y_hat)) - wines_test$quality) <= 1)
k_knnc1 <- fit$bestTune$k
```

As with linear regression several iterations were run starting with alcohol and adding the factor with the next highest correlation coefficient each iteration. The results are summarized in the following table:

Description	Accuracy	Accuracy +/-1	k
Guess Average	0.39269	0.94064	NA
Top Factor	0.54795	0.94977	87
Top Two	0.57991	0.94977	99
Top Three	0.54795	0.95434	91
Top Four	0.55251	0.94977	85
Top Five	0.55251	0.94977	97
Top Six	0.47032	0.94064	18
Top Seven	0.47032	0.94521	13
Top Eight	0.49772	0.93151	9
Top Nine	0.44749	0.91781	16
Top Ten	0.51142	0.94521	62
All Eleven	0.50685	0.94521	60

Note that the best accuracy obtained of 58.0% is obtained with only two factors and is not as good as the 58.9% achieved with linear regression.

2.7 Classification with k-Nearest Neighbors and Scaled Factors

Normalization, scaling values to between 0 an 1, is apparently a common approach for scaling factors with kNN. Because several of the factors have outliers, as seen in Section 2.4.1, each was instead first standardized by subtracting its mean and dividing by its standard deviation. Then as an added step each factor was divided by its corresponding correlation coefficient when measured against quality. This second step had the effect of shortening distances based on factors with higher correlation. Note that each value in this process was calculated entirely from the *wines_train* set so the values used vary slightly from those calculated in Section 2.4.4 which used the larger *wines* training set.

The following table includes the values used for scaling the factors.

Description	Correlation Coefficient	Mean	Standard Deviation
Alcohol	0.45754	10.42801	1.09180
Volatile Acidity	-0.44424	0.53488	0.18634
Citric Acid	0.23548	0.26843	0.19552
Sulphates	0.22673	0.65674	0.17149
Density	-0.18157	0.99675	0.00193
Total Sulfur Dioxide	-0.16846	47.70853	33.66617
Chlorides	-0.15575	0.08811	0.04936
Fixed Acidity	0.11043	8.31429	1.75094
pH	-0.04431	3.30787	0.15460
Free Sulfur Dioxide	-0.02161	15.91014	10.43204
Residual Sugar	0.01386	2.55023	1.40179

The same process as above was used, and the results are summarized below:

Description	Accuracy	Accuracy +/-1	k
Guess Average	0.39269	0.94064	NA
Top Factor	0.54795	0.94977	93
Top Two	0.53881	0.96347	75
Top Three	0.56621	0.96804	32
Top Four	0.57991	0.96804	56
Top Five	0.57534	0.94977	59
Top Six	0.58447	0.95434	46
Top Seven	0.57534	0.95890	70
Top Eight	0.52511	0.94977	59
Top Nine	0.50228	0.95890	99
Top Ten	0.44749	0.91781	18
All Eleven	0.50228	0.91324	10

Here the results were better at 58.4% with six factors but still slightly worse than linear regression.

Figure 38: Optimization of k for kNN Classification

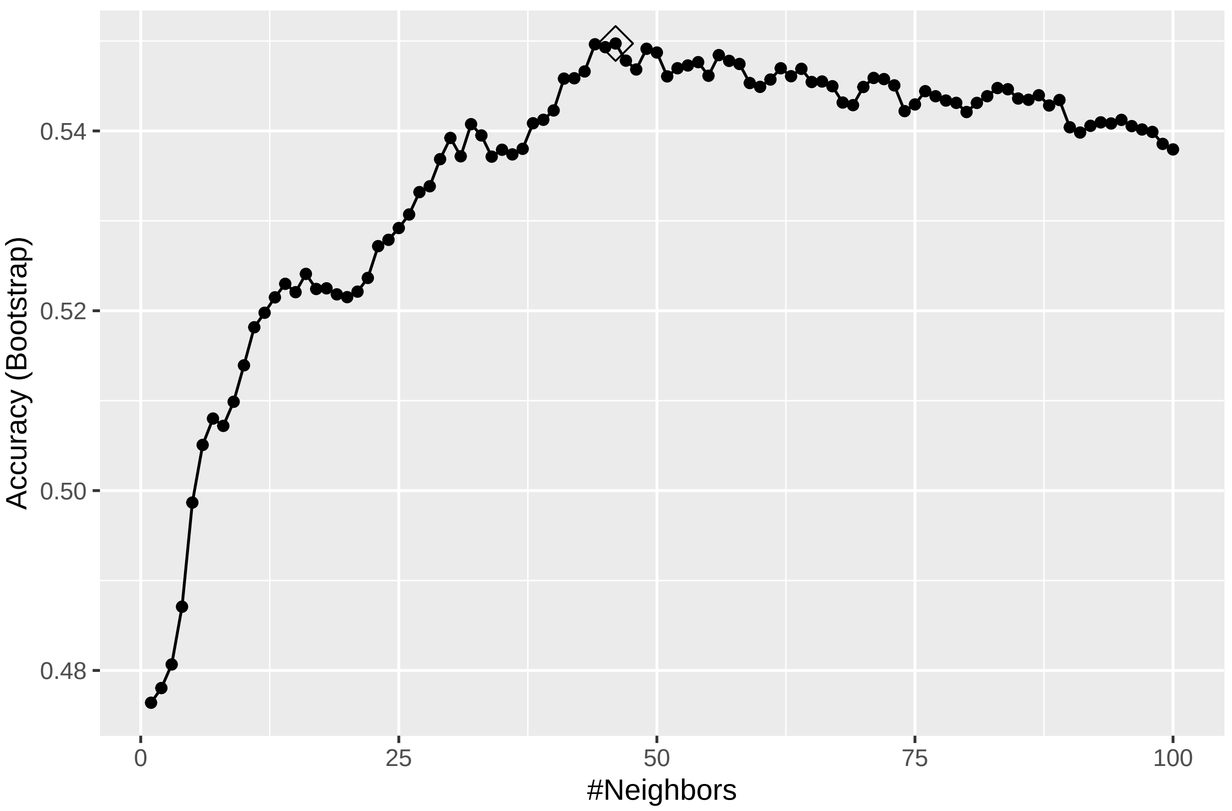


Figure 38 displays a plot of the various accuracies achieved with different values of k. Note that for the tuneGrid parameter from the example code above a sequence from 1 to 100 was specified. Those values correspond to the x axis in the plot.

2.8 Regression with k-Nearest Neighbors and Scaled Factors

To perform regression, the qf factorized value was replaced with the numeric quality value as in the following code:

```
fit_knnr1 <- train(quality ~ alcohol_std, method = 'knn', data = wines_train,
             tuneGrid = data.frame(k = seq(1, 100, 1)))
y_hat <- predict(fit_knnr1, wines_test, type = 'raw')
rmse_knnr1 <- rmse(y_hat, wines_test$quality)
acc_knnr1 <- mean(round(y_hat) == wines_test$quality)
accp1_knnr1 <- mean(abs(round(y_hat) - wines_test$quality) <= 1)
k_knnr1 <- fit_knnr1$bestTune$k
```

The same process as above was used, and the results are summarized below:

Description	RMSE	Accuracy	Accuracy +/-1	k
Guess Average	0.81228	0.39269	0.94064	NA
Top Factor	0.74236	0.54795	0.94977	98
Top Two	0.70392	0.55251	0.96347	67
Top Three	0.66844	0.50685	0.96347	48
Top Four	0.65539	0.54795	0.97260	44
Top Five	0.66779	0.56164	0.96347	37
Top Six	0.66979	0.57078	0.95890	35
Top Seven	0.67201	0.55708	0.95890	39
Top Eight	0.68253	0.56621	0.95434	29
Top Nine	0.70843	0.48858	0.96347	33
Top Ten	0.76331	0.44292	0.94521	26
All Eleven	0.78355	0.47032	0.94064	17

The RMSE results beat linear regression slightly with four factors at 0.655. The best accuracy of 57.1% with six factors again trails linear regression.

Figure 39: Optimization of k for kNN Regression

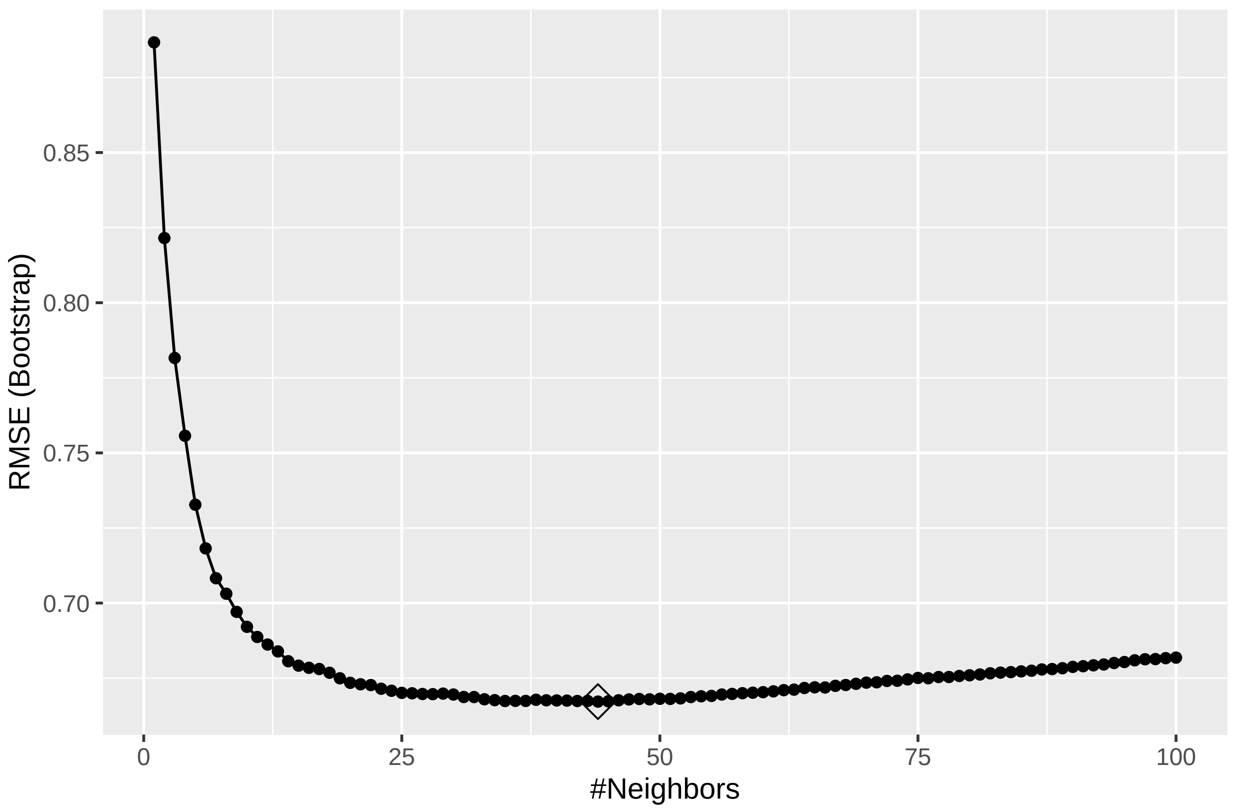


Figure 39 displays a similar plot to Figure 38 but with RMSE on the y axis instead of accuracy.

2.9 Classification with Random Forests (Rborist)

The final algorithm run against the data set was the RBorist implementation of Random Forests which works by consolidating a series of randomly generated decision trees. Like k-Nearest Neighbors, Random Forests supports both classification and regression. Instead of incrementally adding new factors at each step, however, the PredFixed (a.k.a. mtry) tuning parameter which controls the number of variables that are randomly sampled at candidates at each split was adjusted from one to eleven.

This was done with code similar to the following:

```r
fit_rfr1 <- train(quality ~ alcohol + volatile_acidity + sulphates + citric_acid
                  + density + total_sulfur_dioxide + chlorides + fixed_acidity
                  + pH + free_sulfur_dioxide + residual_sugar, method = 'Rborist',
                  data = wines_train, tuneGrid = data.frame(predFixed = 1,
                                                            minNode = seq(1, 20)))
y_hat <- predict(fit_rfr1, wines_test, type = 'raw')
rmse_rfr1 <- rmse(y_hat, wines_test$quality)
acc_rfr1 <- mean(abs(round(y_hat) == wines_test$quality))
accp1_rfr1 <- mean(abs(round(y_hat) - wines_test$quality) <= 1)
mn_rfr1 <- fit_rfr1$bestTune$minNode
```

The results for classification are shown in the following table:

Description	Accuracy	Accuracy +/-1	minNode
Guess Average	0.39269	0.94064	NA
PredFixed = 1	0.59361	0.94977	3
PredFixed = 2	0.59817	0.94521	2
PredFixed = 3	0.59817	0.95434	8
PredFixed = 4	0.60731	0.96804	19
PredFixed = 5	0.59361	0.95890	4
PredFixed = 6	0.59817	0.95434	2
PredFixed = 7	0.58904	0.94977	4
PredFixed = 8	0.58904	0.95890	5
PredFixed = 9	0.58447	0.94977	2
PredFixed = 10	0.56621	0.95434	4
PredFixed = 11	0.56621	0.96347	4

Here the best accuracy so far of 60.7% was achieved with a PredFixed value of 4.

Figure 40: Optimization of minNode for Random Forest Classification

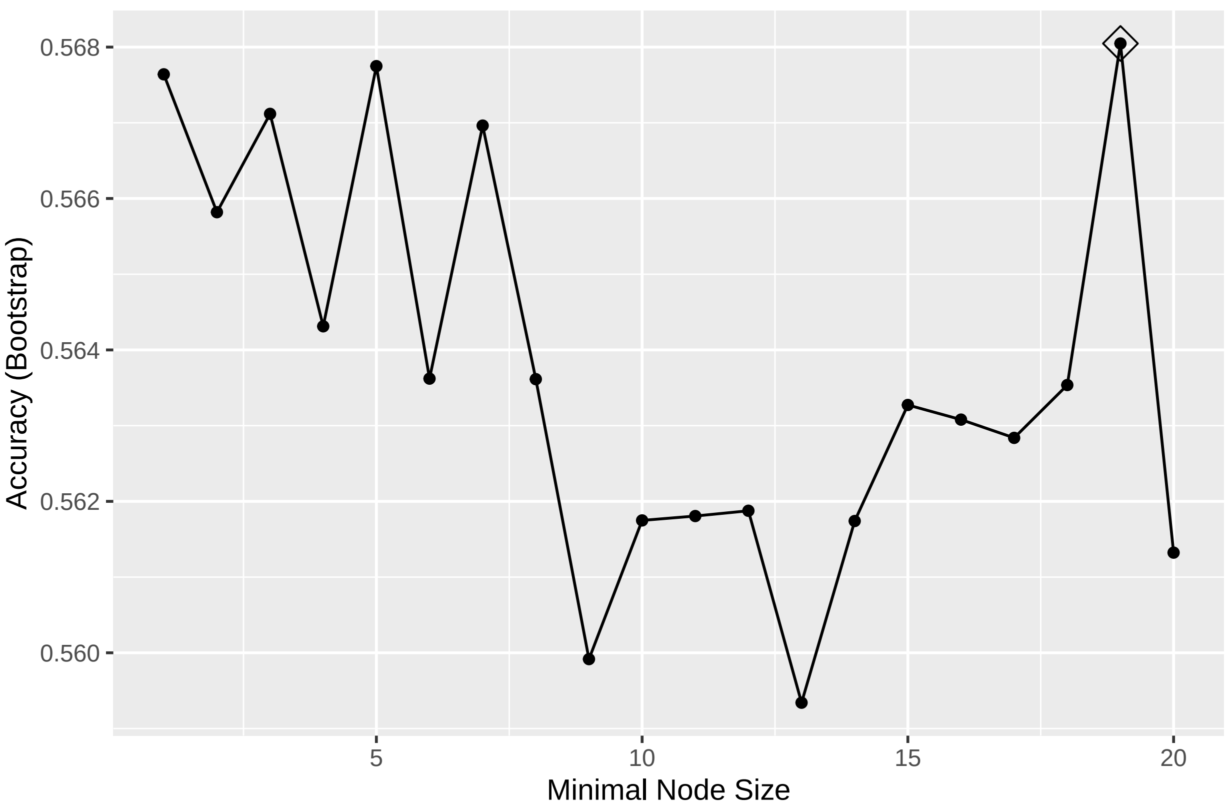


Figure 40 shows a plot similar to the previous two except that in this case the minNode parameter was optimized. The RBorist documentation describes this parameter as the “minimum number of distinct row references to split a node” and it should be somewhat analogous to k for k-Nearest Neighbors.

At first glance, the plot appears substantially more random than the previous two, however, the y axis scale is substantially smaller and the full range represents less that 0.01 in accuracy.

2.10 Regression with Random Forests (Rborist)

Swapping out qf for quality, the same process was followed for regression. The results are shown below:

Description	RMSE	Accuracy	Accuracy +/-1	minNode
Guess Average	0.81228	0.39269	0.94064	NA
PredFixed = 1	0.66168	0.58447	0.96804	2
PredFixed = 2	0.65170	0.57991	0.97260	4
PredFixed = 3	0.64622	0.58447	0.97260	5
PredFixed = 4	0.64745	0.58447	0.97260	20
PredFixed = 5	0.64487	0.56621	0.97260	16
PredFixed = 6	0.64214	0.59817	0.97260	5
PredFixed = 7	0.64483	0.58447	0.97260	1
PredFixed = 8	0.64328	0.58447	0.97260	20
PredFixed = 9	0.66168	0.57534	0.97260	18
PredFixed = 10	0.64503	0.57534	0.97260	20
PredFixed = 11	0.64526	0.56164	0.97260	20

This had the best value for RMSE with 0.642 at a PredFixed value of 6 and except for Random Forest classification also the best accuracy with 59.8%.

Figure 41: Optimization of minNode for Random Forest Regression

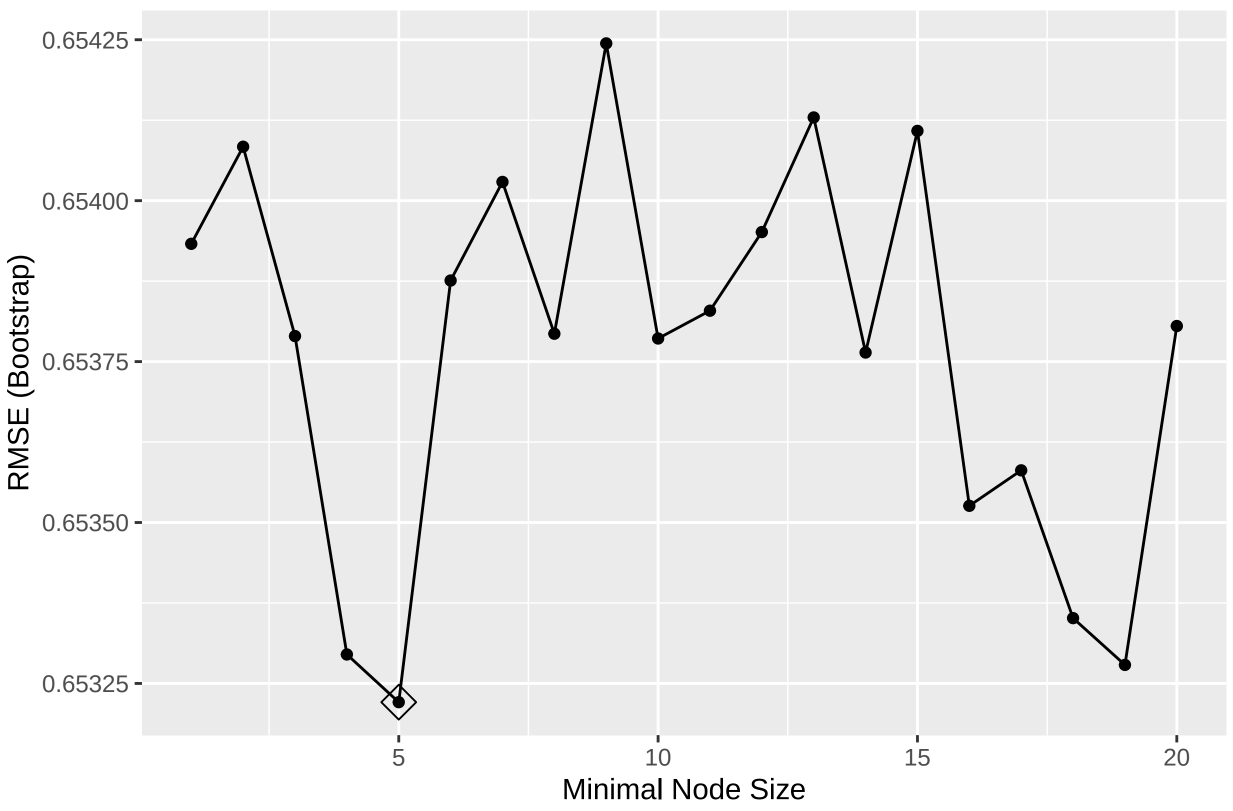


Figure 41 displays a similar plot to Figure 40 but again with RMSE on the y axis instead of accuracy since this was for regression. Note also that here the full range of the y axis is about 0.001.

2.11 Variable Importance and Comparison to Cortez et. al.

As mentioned by Prof. Irizarry in section 6.1 of his edX course on Machine Learning, the RBorist package doesn't implement variable importance so here the more standard randomForest algorithm was called with similar parameters as in the following code:

```
fit_randomf <- randomForest(quality ~ alcohol + volatile_acidity + sulphates
                            + citric_acid + density + total_sulfur_dioxide
                            + chlorides + fixed_acidity + pH
                            + free_sulfur_dioxide + residual_sugar,
                            data = wines_train, mtry = 6, minsize = 5)
```

Using this algorithm, the importance information is available by calling the *importance()* function or via the *$importance* value in the result. The importance results are listed in the following table. Note that the ordering, at least, is mostly consistent with an ordering based on correlation coefficients.

Description	Importance
Alcohol	124.59289
Volatile Acidity	93.99300
Sulphates	78.95893
Total Sulfur Dioxide	48.41749
Density	37.11762
Chlorides	36.43458
Citric Acid	31.34906
Residual Sugar	30.41065
pH	30.06225
Fixed Acidity	27.16572
Free Sulfur Dioxide	24.66480

Getting importance information from k-Nearest Neighbors was impractical as the factors themselves were scaled by the corresponding correlation coefficients in an attempt to improve performance. For linear regression it's possible to look at the regression coefficients, but each coefficient is scaled by different units.

A possible solution, however, comes from the equation for the slope of a regression line which is the correlation coefficient ρ multiplied by the ratio of the standard deviations of the correlated values.

$$m = \rho \frac{\sigma_y}{\sigma_x}$$

Reordering to solve for ρ and assuming y, the predicted value, is quality gives:

$$\rho = m \frac{\sigma_f}{\sigma_q}$$

Linear regression for multiple factors is more complicated than for a single factor so it's unclear how well the above equation survives more rigorous mathematical scrutiny, but it seems to at least provide a decent approximation. The results for each factor are shown in the following table sorted by magnitude.

Description	Importance
Alcohol	0.33315
Volatile Acidity	-0.31104
Sulphates	0.17730
Total Sulfur Dioxide	-0.10341
pH	-0.06165
Citric Acid	-0.05543
Fixed Acidity	0.04651
Free Sulfur Dioxide	0.04466
Density	-0.03246
Residual Sugar	0.03060
Chlorides	0.00108

It should be noted the Figure 4 of Cortez et. al. shows the variable importance they calculated using an algorithm based on support vector machines (SVM.) Those results are reproduced here.

Order	Description
1	Sulphates
2	pH
3	Total Sulfur Dioxide
4	Alcohol
5	Volatile Acidity
6	Free Sulfur Dioxide
7	Fixed Acidity
8	Residual Sugar
9	Chlorides
10	Density
11	Citric Acid

While the order is different, 4 out of the top 5 factors are the same in all three cases. Also, Cortez et. al. achieved an overall accuracy of 59.1% for both Linear Regression and Neural Networks algorithms which is largely consistent with the accuracies seen so far in this paper. They achieved 62.4% using SVM but speculate one possible reason for the difference is the SVM cost function which is more tolerant of outliers.

2.12 Premium vs. Regular Wines

From here the focus of this paper switches to identifying the best algorithm for selecting premium wines with high precision. As mentioned in Section 2.2, here a premium wine is defined as one with a quality value of 7 or more and a regular wine is one with a quality below 7.

The following figures recreate those from Section 2.4.4 this time with colors to indicate regular vs. premium red wines.

Figure 42: Volatile Acidity vs. Alcohol by Grade

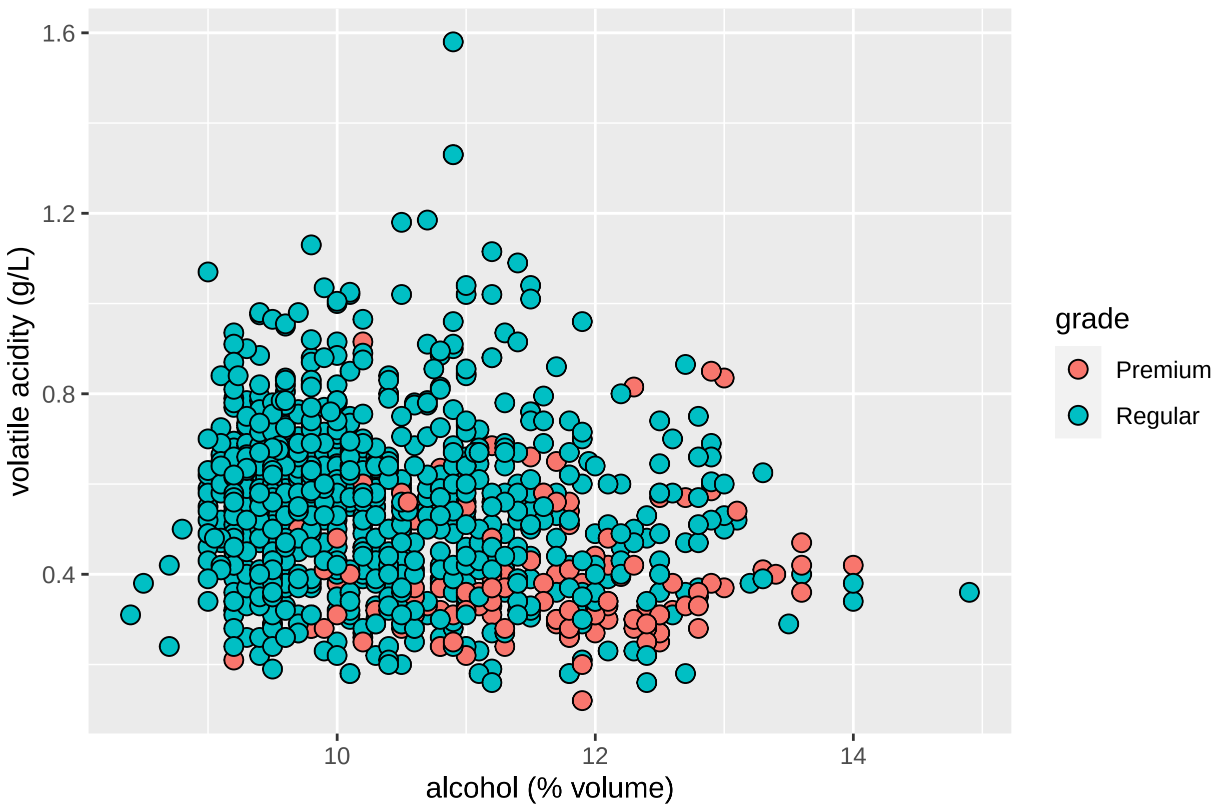


Figure 43: Sulphates vs. Alcohol by Grade

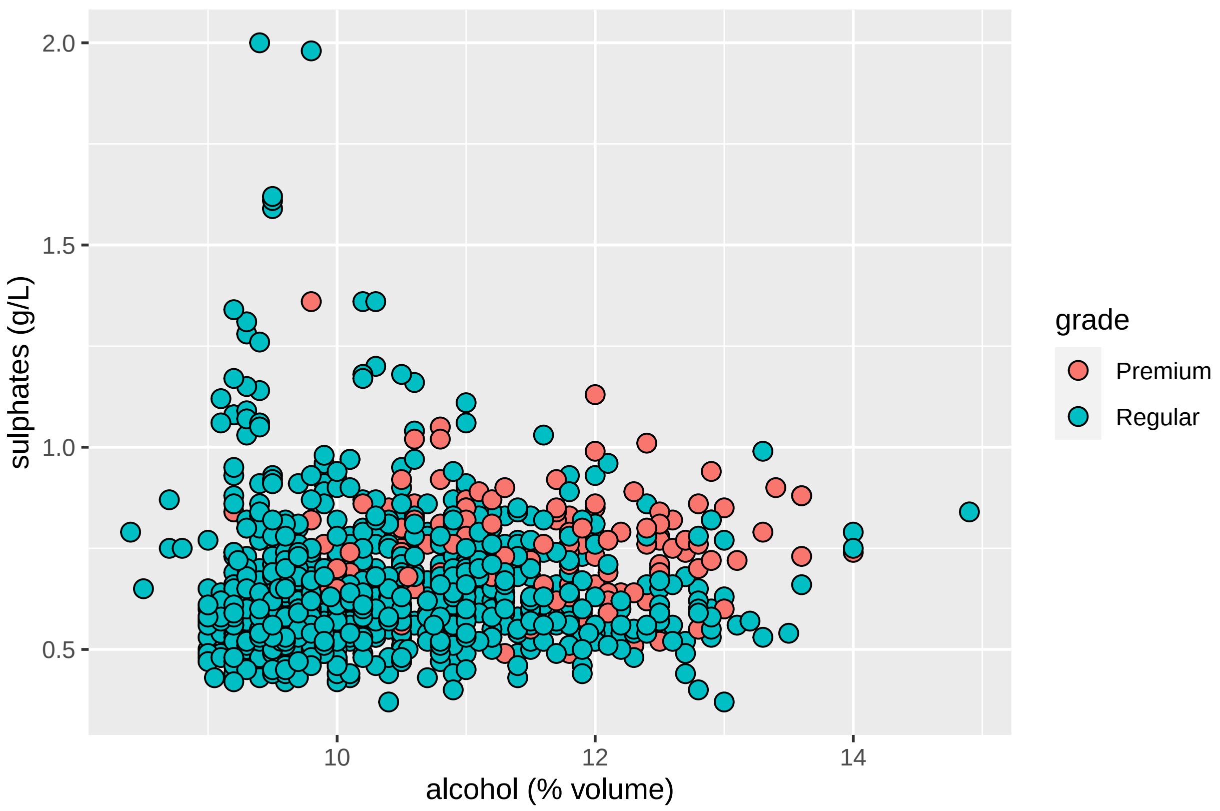
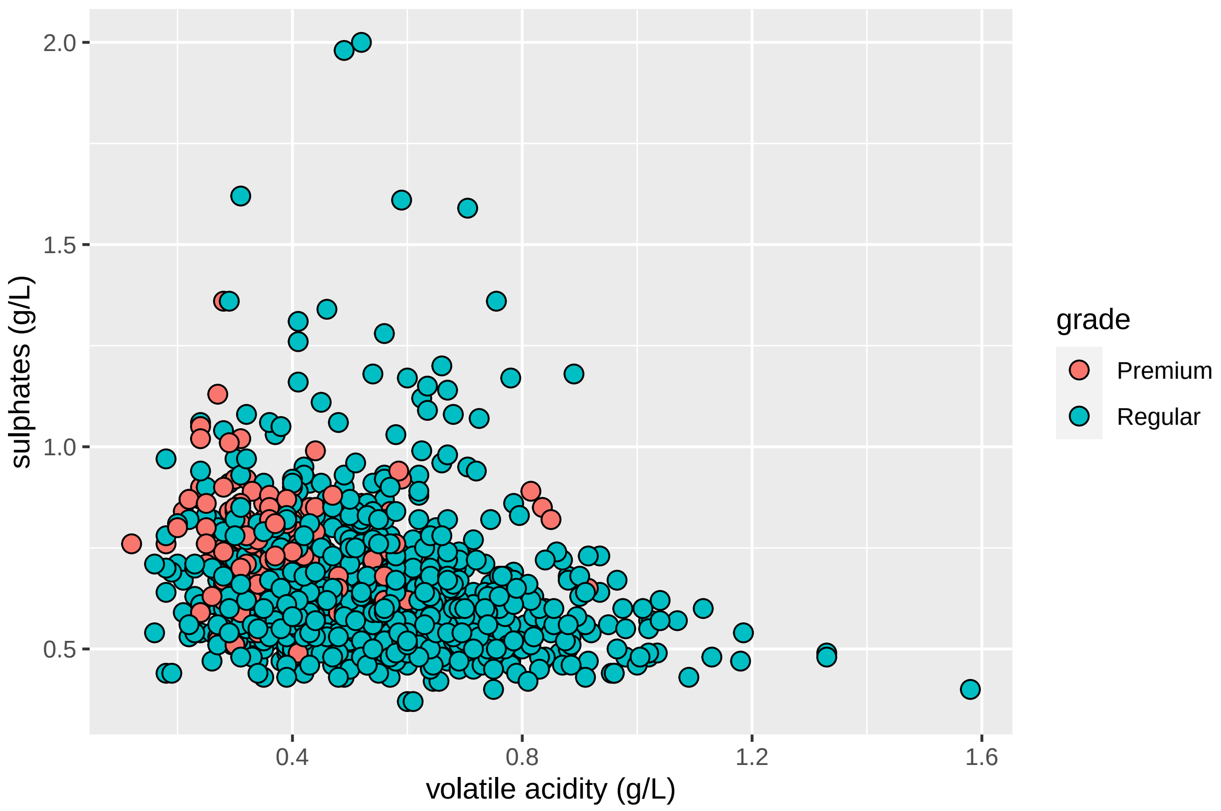


Figure 44: Sulphates vs. Volatile Acidity by Grade



2.12.1 Classification Models

The best performing classification algorithms for both k-Nearest Neighbors and Random Forests from the previous sections were run again this time predicting the *Premium* vs. *Regular* grade factor created in Section 2.2.

```
# Use the best knn model from above for classification of premium wines.
fit_knncg <- train(grade ~ alcohol_std + volatile_acidity_std + sulphates_std
                   + citric_acid_std + density_std + total_sulfur_dioxide_std,
                   method = 'knn', data = wines_train,
             tuneGrid = data.frame(k = seq(1, 100, 1)))

# Do the same for random forests.
fit_rfg <- train(grade ~ alcohol + volatile_acidity + sulphates + citric_acid
                 + density + total_sulfur_dioxide + chlorides + fixed_acidity
                 + pH + free_sulfur_dioxide + residual_sugar, method = 'Rborist',
                  data = wines_train, tuneGrid = data.frame(predFixed = 4,
                                                            minNode = seq(1, 20)))
```

The following table is the confusion matrix for k-Nearest Neighbors.

	Premium	Regular
Premium	10	4
Regular	20	185

And the following is for Random Forests.

	Premium	Regular
Premium	13	5
Regular	17	184

For clarification of each of the fields, *Premium* is considered the positive value. Predictions are specified by the row labels to the left and actual values are specified by the column headings across the top so the values *TP*, *FP*, *FN*, and *TN* are *true positives*, *false positives*, *false negatives*, and *true negatives* respectively.

	Premium	Regular
Premium	TP	FP
Regular	FN	TN

The following table gives various statistics that can each be calculated from the values in the confusion matrices for both algorithms.

Description	Accuracy	Sensitivity	Specificity	Precision	P-Value
k-Nearest Neighbor	0.89041	0.33333	0.97884	0.71429	0.13869
Random Forest	0.89954	0.43333	0.97354	0.72222	0.06593

At first glance the accuracies of 89.0% and 90.0% appear quite impressive, however, because of the high prevalence of *Regular* wines in the dataset, simply guessing *Regular* every time results in an accuracy of 86.3%. In fact, looking at the p-values, the accuracies don't even pass the usual standard of 5% to be considered statistically significant.

This paper is less concerned with accuracy than with precision which in this case is the likelihood that a wine identified as *Premium* is actually a premium wine. The equation from the introduction for precision is repeated for convenience:

$$precision = \frac{TP}{TP+FP}$$

Here, the precisions are 71.4% and 72.2% for k-Nearest Neighbors and Random Forests respectively. However, if the accuracy isn't considered significant, it's reasonable to ask whether the precisions suffer from the same problem. Calculating a p-value like value for the precisions is outside of the scope of this paper, however, the values can easily be estimated with a monte carlo simulation as illustrated by the following code for each algorithm.

```r
mean(replicate(10000, {
  sum(sample(wines_test$grade, 14) == 'Premium')
}) >= 10)

mean(replicate(10000, {
  sum(sample(wines_test$grade, 18) == 'Premium')
}) >= 13)
```

Translating to English, the first three lines of R code is asking, what percentage of time does picking 14 random wines from the *wines_test* set result in choosing 10 or more premium wines. The second three lines of code asks a similar question for the Random Forests confusion matrix. The results of the monte carlo simulations are 0.000% and 0.000% respectively which both are obviously well below the 5% threshold.

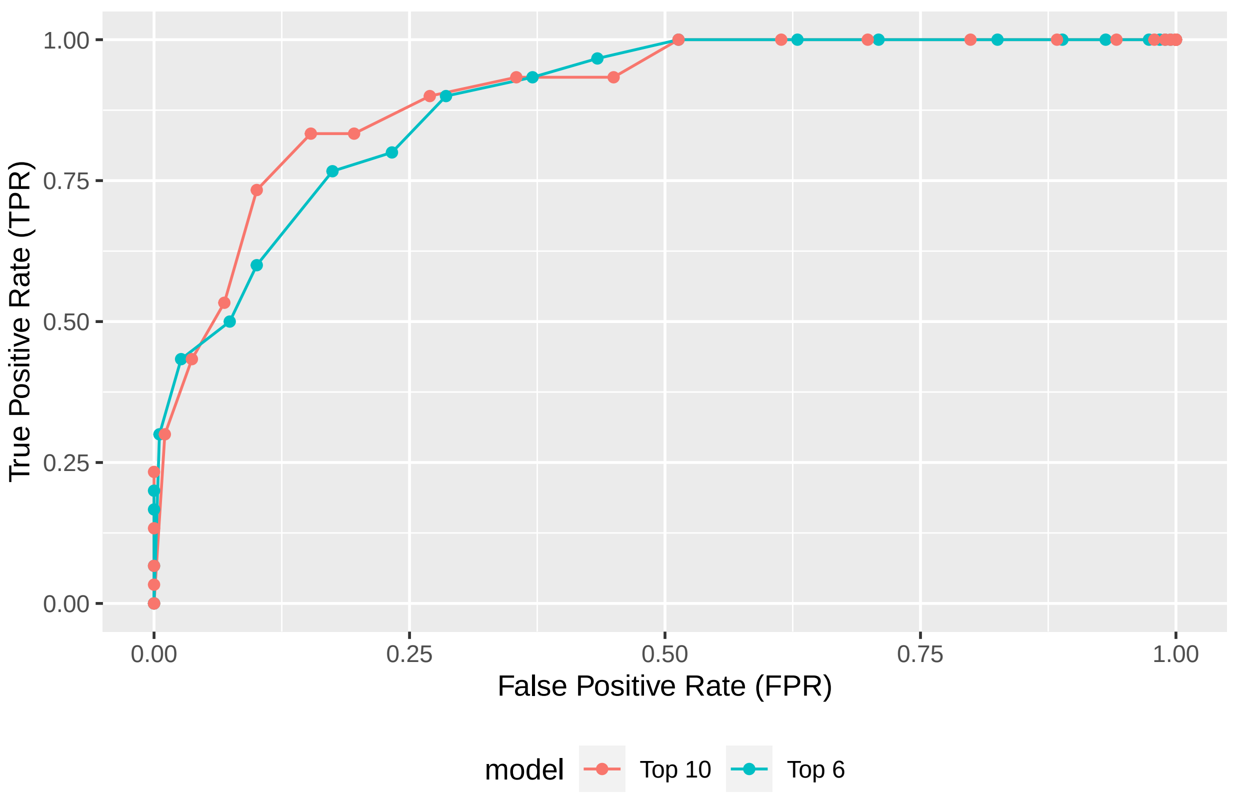
These values for precision are good, but why settle for drinking poorer wines 25% of the time? The next section explores a strategy that can be used with regression algorithms that should, at least in theory, lead to even higher precision.

2.12.2 Regression Model Selection

The first step in looking at higher precision from the regression models was to choose the best model for each algorithm. Recall from the previous sections that the model with the best RMSE value was not typically the one with the best accuracy. The following figures display ROC plots for the models with the best RMSE and accuracy for each type of algorithm.

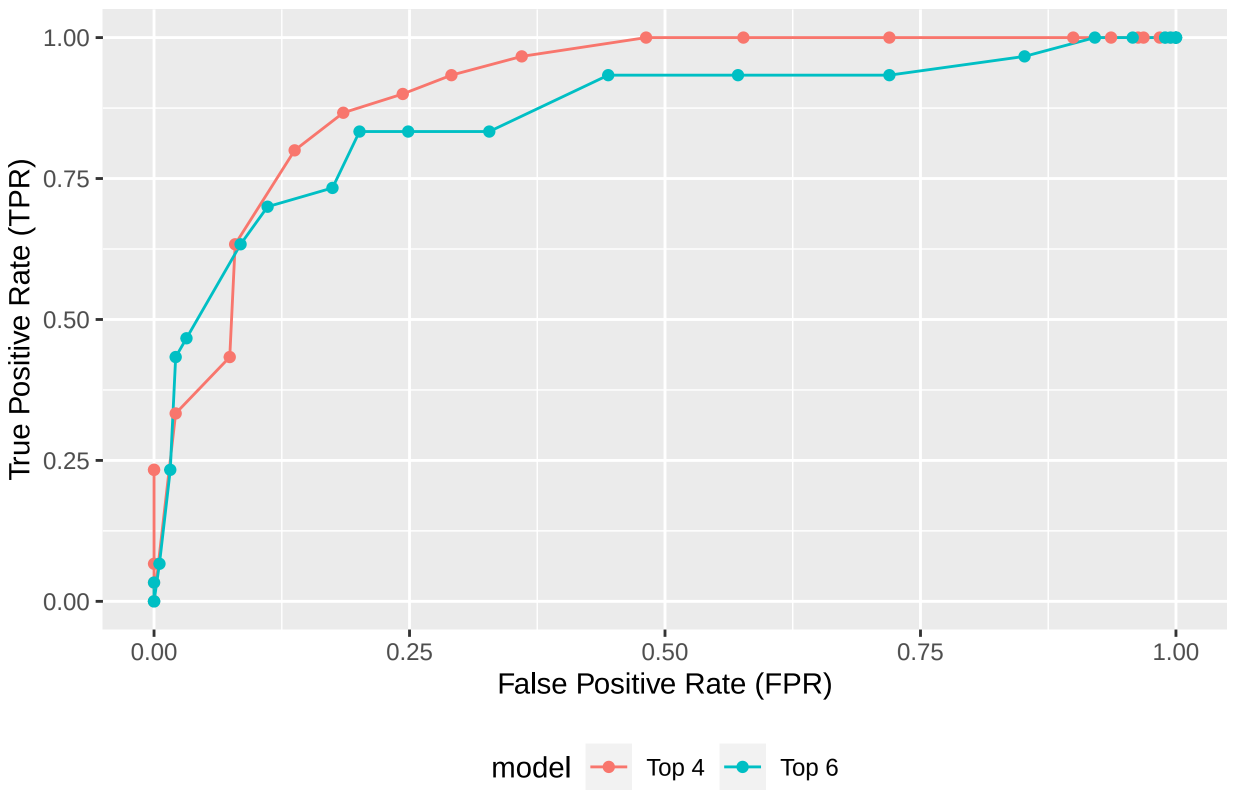
For an ROC plot, sensitivity and specificity both peak in the upper left corner so the better model will be the one furthest outside the other in that direction. Note that TPR is just another name for sensitivity, and FPR is one minus specificity.

Figure 45: Linear Regression ROC Plots



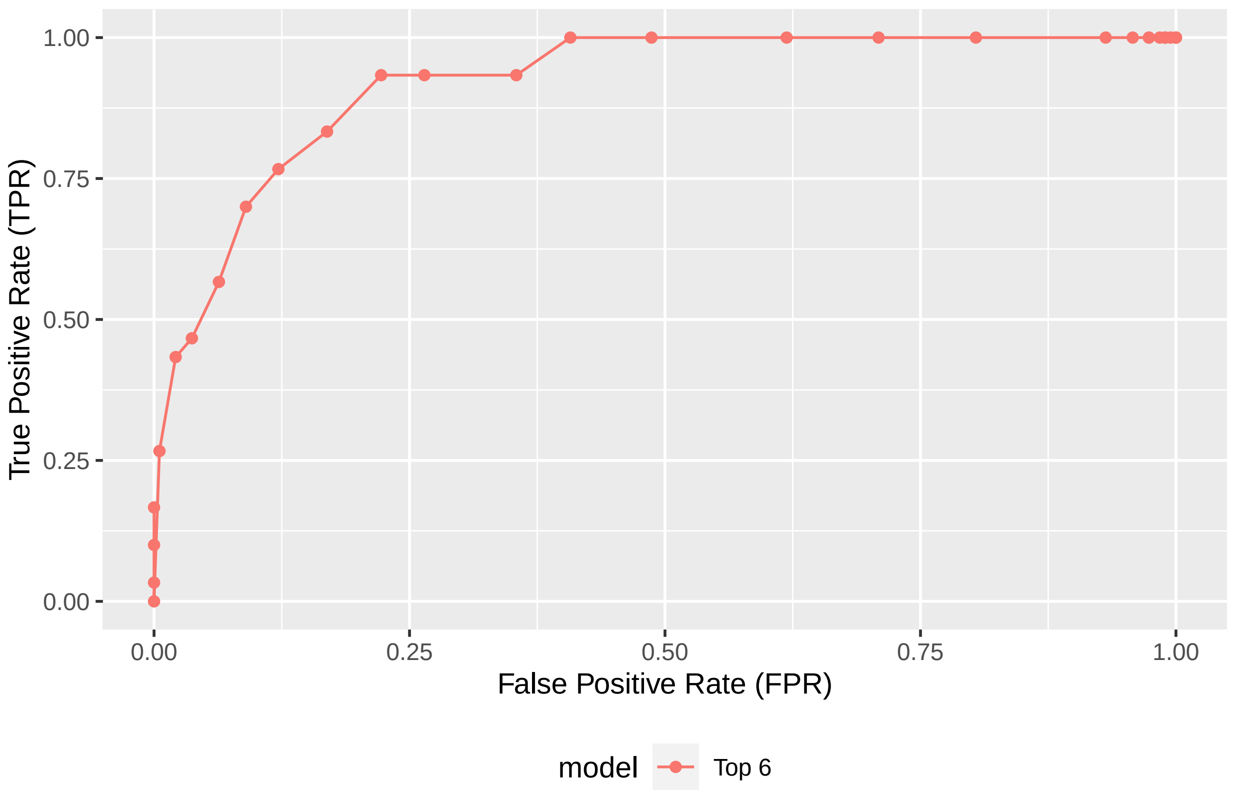
Although the two switch a few times, for linear regression, the model outside the other the most is the one that uses ten factors and had the best RMSE value.

Figure 46: k−Nearest Neighbors ROC Plots



For k-Nearest Neighbors, the clear winner is again the model with the highest RMSE and the one that in this case includes only four factors.

Figure 47: Random Forest ROC Plots



For Random Forests, the same model had both the best RMSE and the best accuracy so it's obviously the winner.

2.12.3 Regression Model Ranges

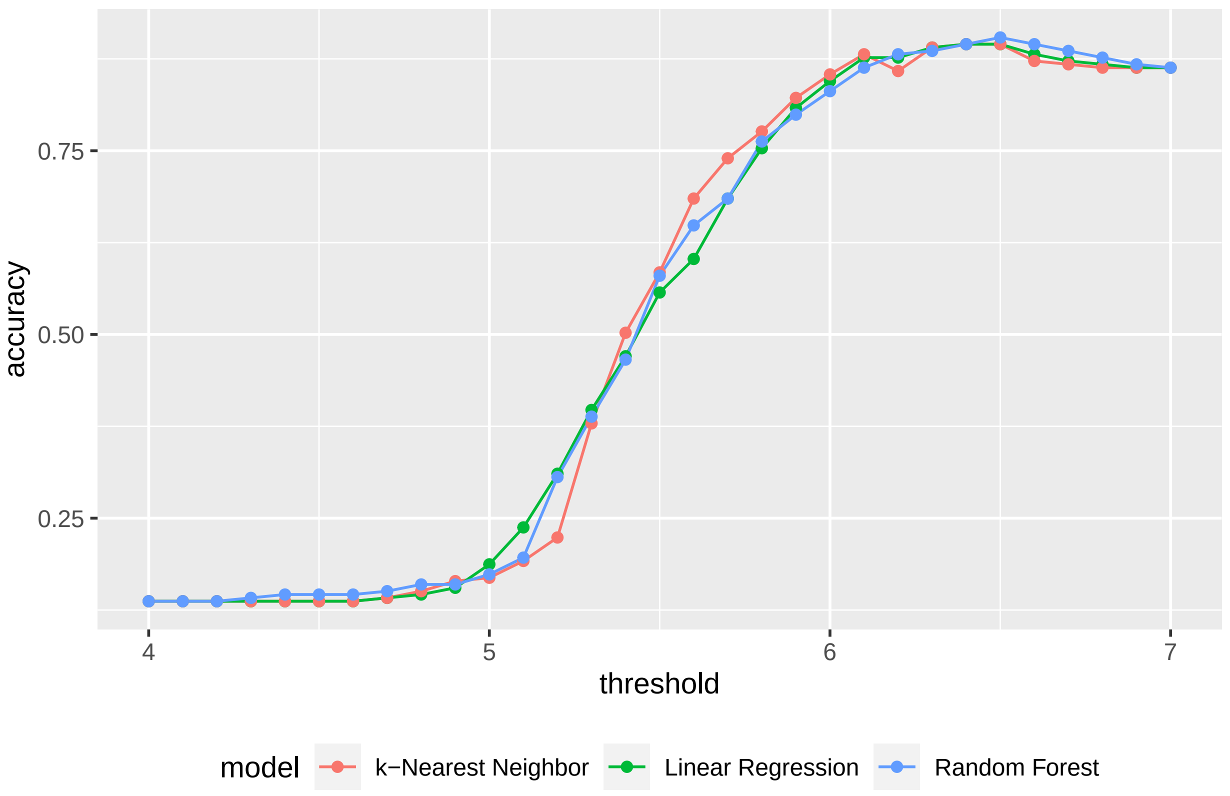
The following table displays the output ranges for each algorithm for the wines in the *wines_test* set. This is shown here, because it explains the x-axis range for the next few sections. Note that none of the algorithms ever predicted an 8 and that only Random Forests ever predicted a 4 even though both values are in the set.

Description	Min	Max
Linear Regression	4.64545	6.84178
k-Nearest Neighbor	4.63636	6.70455
Random Forest	4.28709	6.92001

2.12.4 Regression Accuracy

Because the output of a regression algorithm is continuous, it's necessary to specify a threshold in order to use it for classification. In this case an obvious choice would be to use 6.5 as the threshold between regular and premium wines, however it's possible to select different combinations of sensitivity, specificity, precision, and even accuracy by adjusting that threshold. Figure 48 shows the impact different thresholds have on accuracy.

Figure 48: Regression Model Accuracy vs Threshold



The following table shows various statistics of each algorithm at the threshold corresponding to maximum accuracy.

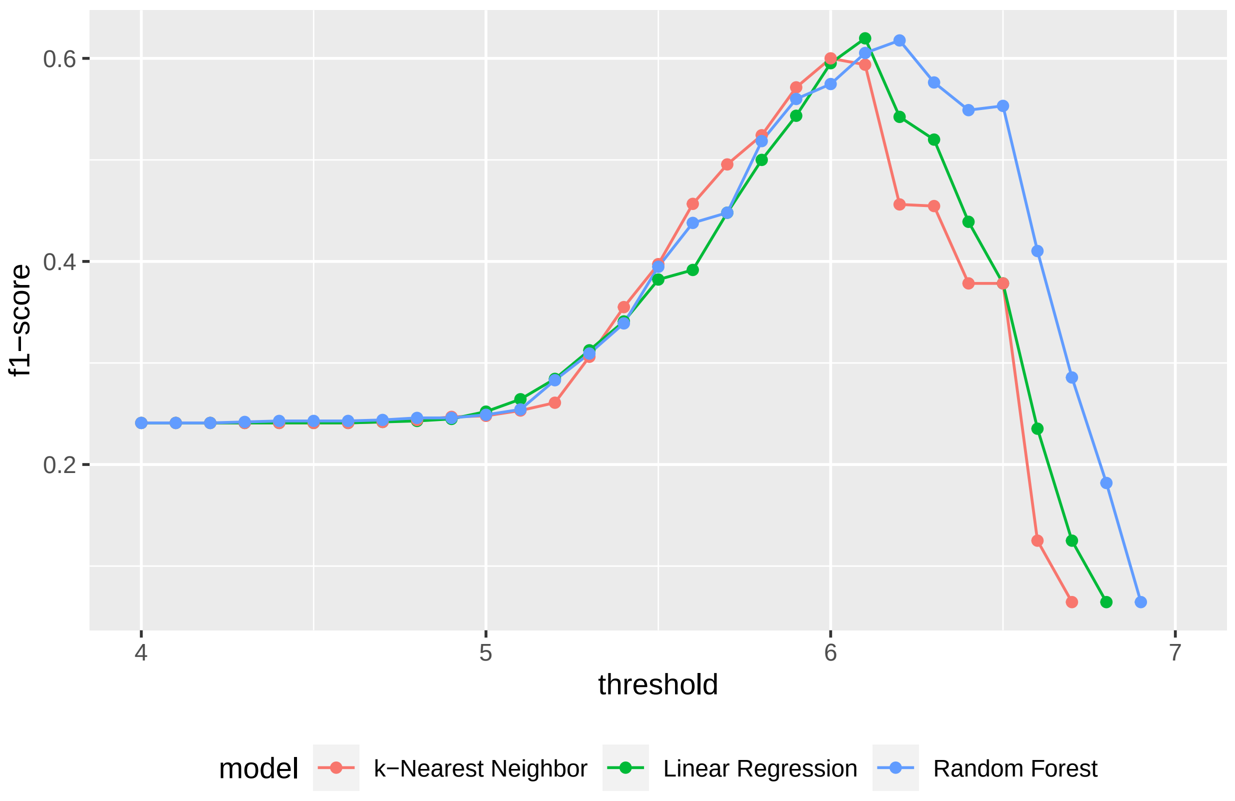
Description	Threshold	Accuracy	Sensitivity	Specificity	Precision	P-Value
Linear Regression	6.4	0.89498	0.30000	0.98942	0.81818	0.09762
k-Nearest Neighbor	6.4	0.89498	0.23333	1.00000	1.00000	0.09762
Random Forest	6.5	0.90411	0.43333	0.97884	0.76471	0.04262

Note that maximum accuracy for Linear Regression and k-Nearest Neighbors actually occurs at 6.4. Also note in this case, the best regression accuracies actually slightly exceed the corresponding classification accuracies for k-Nearest Neighbors and Random Forests though probably not by enough to be statistically significant.

2.12.5 F1-Score

The F1-Score is the harmonic average of precision and recall (a.k.a sensitivity.) Figure 49 shows the effect threshold selection has on the F1-Score for each algorithm.

Figure 49: F1−Score vs. Threshold

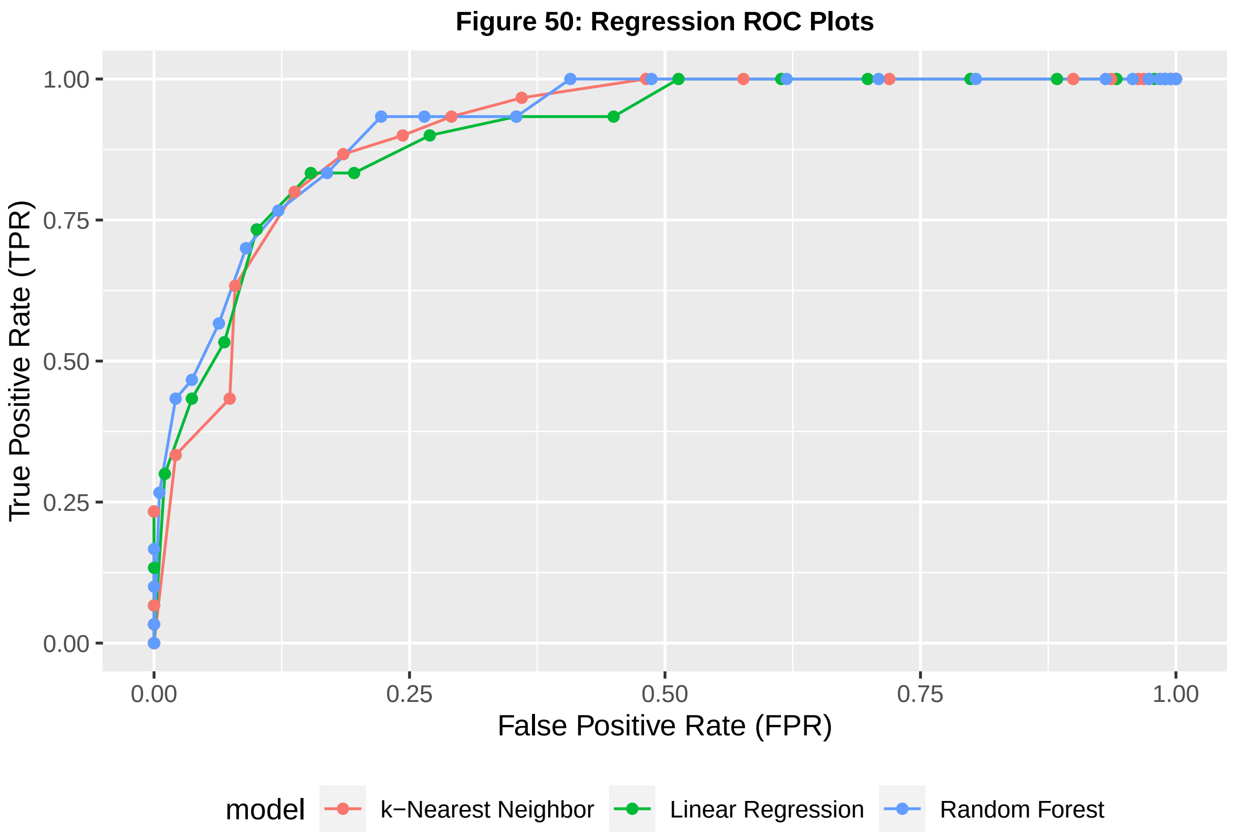


The following table shows various statistics of each algorithm at the thresholds corresponding to maximum f1-score.

Description	Threshold	F1	Accuracy	Sensitivity	Specificity	Precision	P-Value
Linear Regression	6.1	0.61972	0.87671	0.73333	0.89947	0.53659	0.31814
k-Nearest Neighbor	6.0	0.60000	0.85388	0.80000	0.86243	0.48000	0.69470
Random Forest	6.2	0.61765	0.88128	0.70000	0.91005	0.55263	0.24979

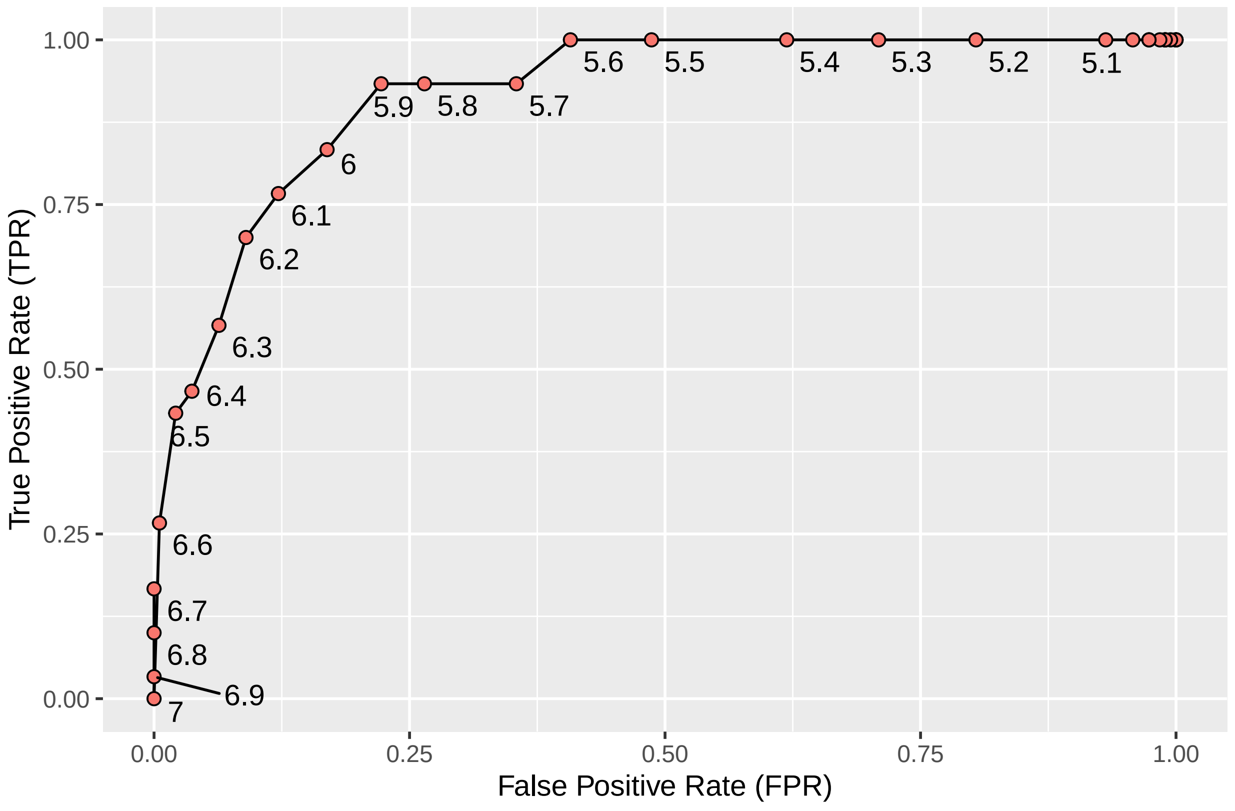
2.12.6 ROC Plots

Figure 50 shows an ROC Plot, introduced earlier, of the three algorithms.



Note that Random Forests appears to have a slight edge over the others. Figure 51 shows just Random Forests with the thresholds associated with each point.

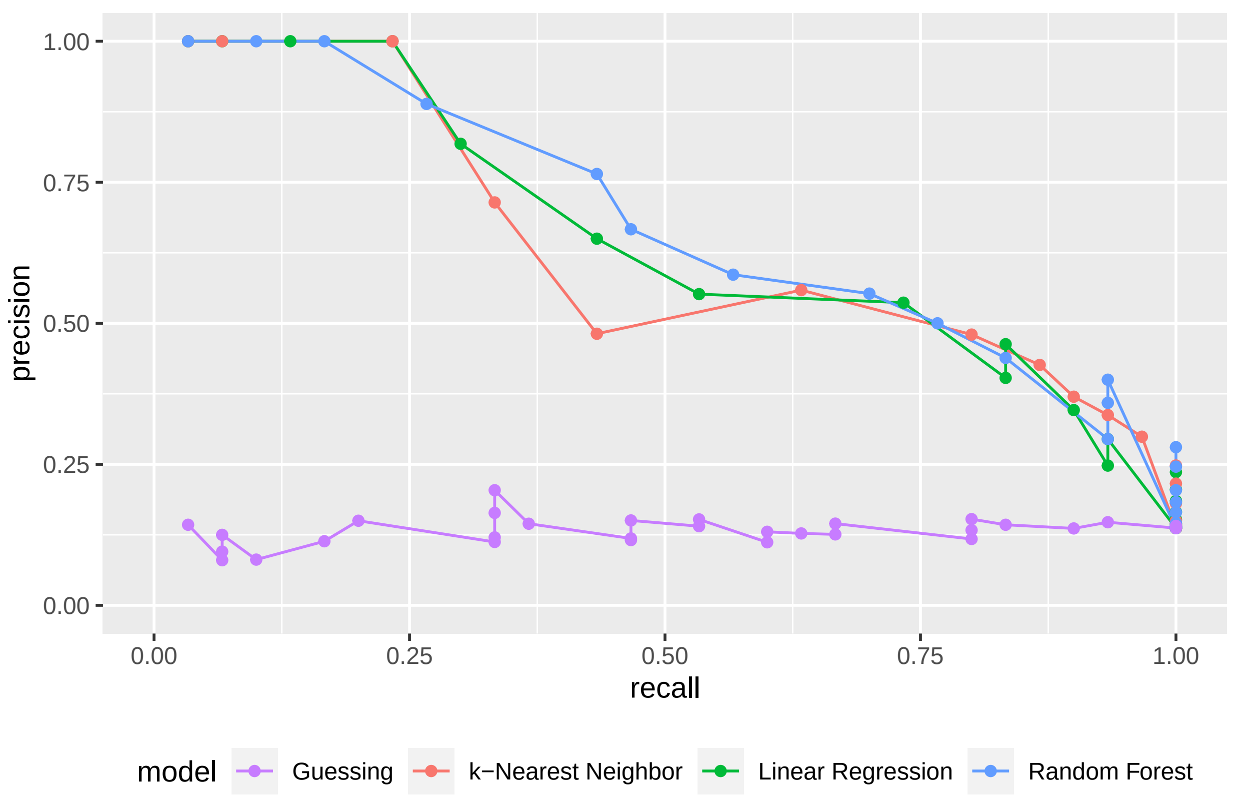
Figure 51: Random Forest ROC Plots



2.12.7 Precision-Recall Plots

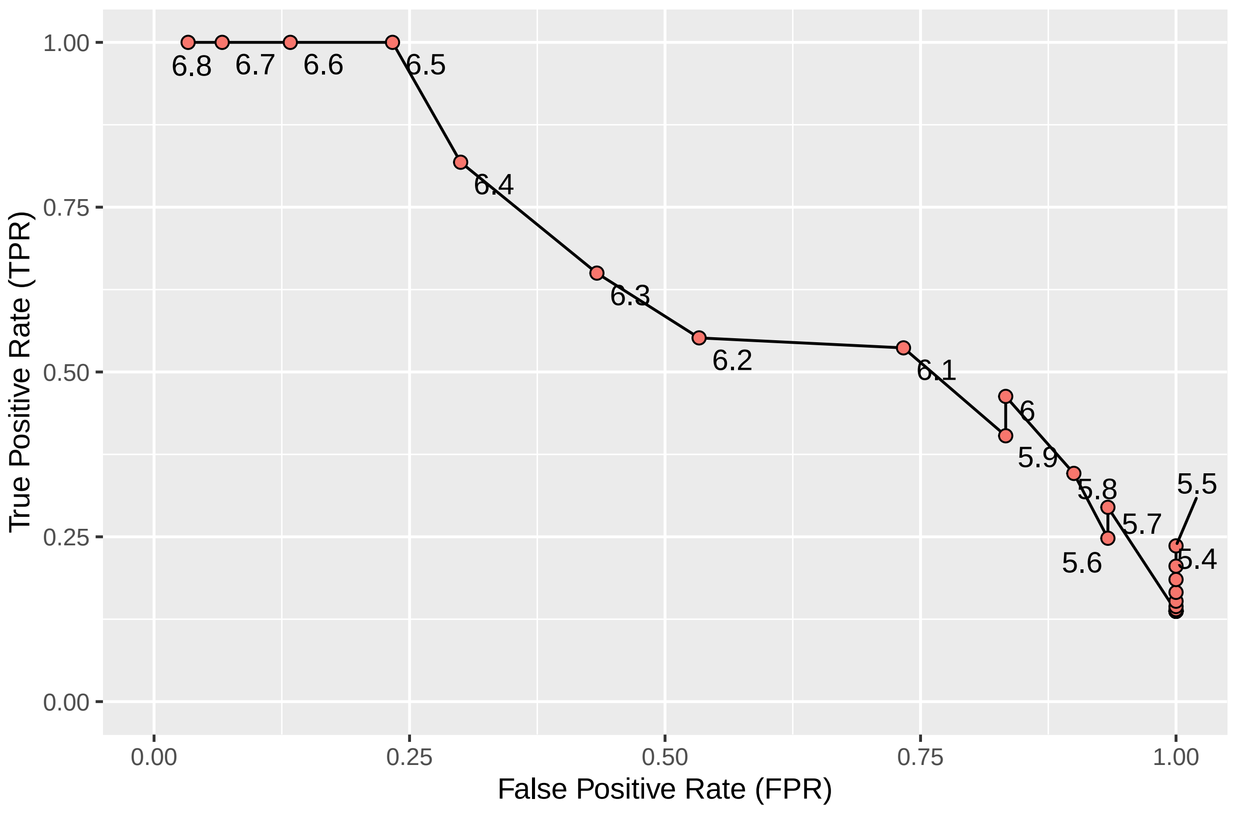
Because the goal of this paper is to select an algorithm with the highest precision, Precision-Recall plots are more suited to making a selection than ROC plots. Figure 52 shows a Precision-Recall Plot of the three algorithms.

Figure 52: Regression Precision−Recall Plots



Surprisingly, all three algorithms show thresholds with 100% precision. Linear Regression and k-Nearest Neighbors both show 100% precision with higher sensitivity (recall) with Linear Regression showing a slight edge once precision drops below 100%. Figure 53 shows just Linear Regression with the thresholds associated with each point. Here a threshold of 6.5 shows the greatest sensitivity with 100% precision.

Figure 53: Random Forest ROC Plots



2.12.8 Selected Model Training Test Set Results

The following table shows the confusion matrix associated with applying the Linear Regression model against the wines_test set with a threshold of 6.5.

	Premium	Regular
Premium	7	0
Regular	23	189

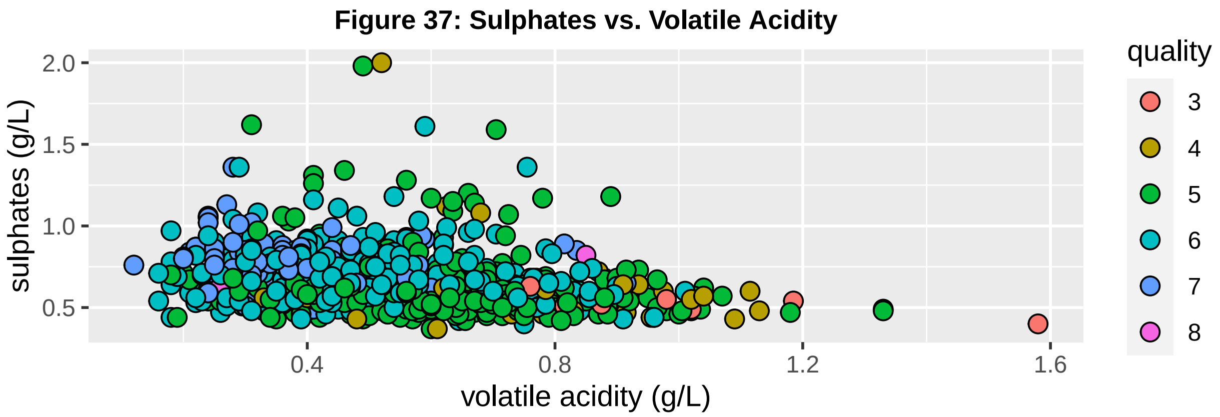
The next table shows the associated statistics.

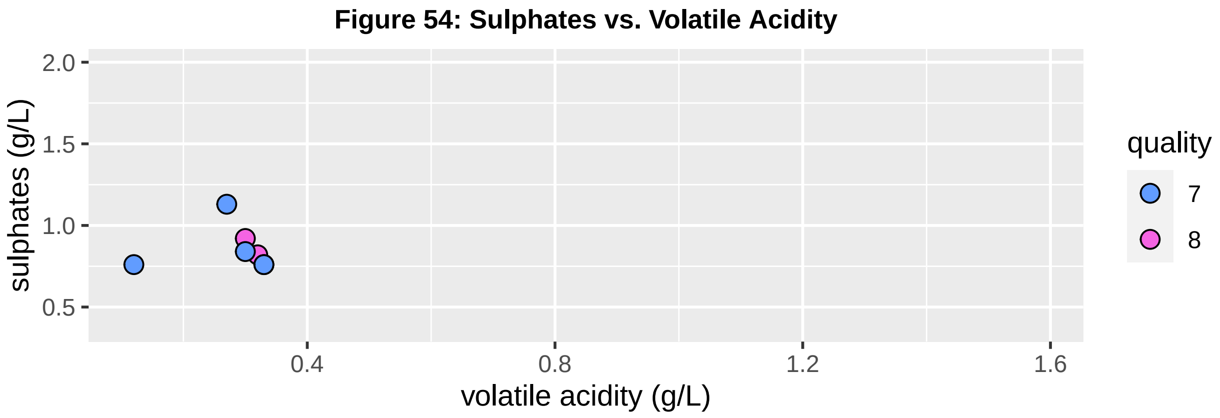
Description	Accuracy	Sensitivity	Specificity	Precision	P-Value
Linear Regression	0.89498	0.23333	1	1	0.09762

And finally, the following table shows statistics from the wines that were selected. Two have quality values of 8, and 5 have values of 7.

	alcohol	volatile_acidity	sulphates	quality
1	12.6	0.32	0.82	8
2	11.7	0.30	0.92	8
3	12.0	0.27	1.13	7
4	12.4	0.33	0.76	7
5	11.9	0.12	0.76	7
6	11.7	0.30	0.84	7
7	12.8	0.33	0.76	7

Figure 54 shows the selected wines plotted as sulphates vs. volatile acidity. Figure 37 is repeated for context.





3 Results

To get the final results, the Linear Regression model from the previous section was retrained against the full training set, *wines*, and then used to predict the values in the final holdout set, *final_test* as shown in the following code:

```
fit_final <- lm(quality ~ alcohol + volatile_acidity + sulphates + citric_acid
                + density + total_sulfur_dioxide + chlorides + fixed_acidity + pH
                + free_sulfur_dioxide, data = wines)
cm_final <- confusionMatrix(factor(ifelse(predict(fit_final, final_test) >= 6.5,
                                          'Premium', 'Regular'),
                                   levels = c('Premium', 'Regular')), final_test$grade)
```

The following table shows the confusion matrix for the final result.

	Premium	Regular
Premium	6	5
Regular	31	230

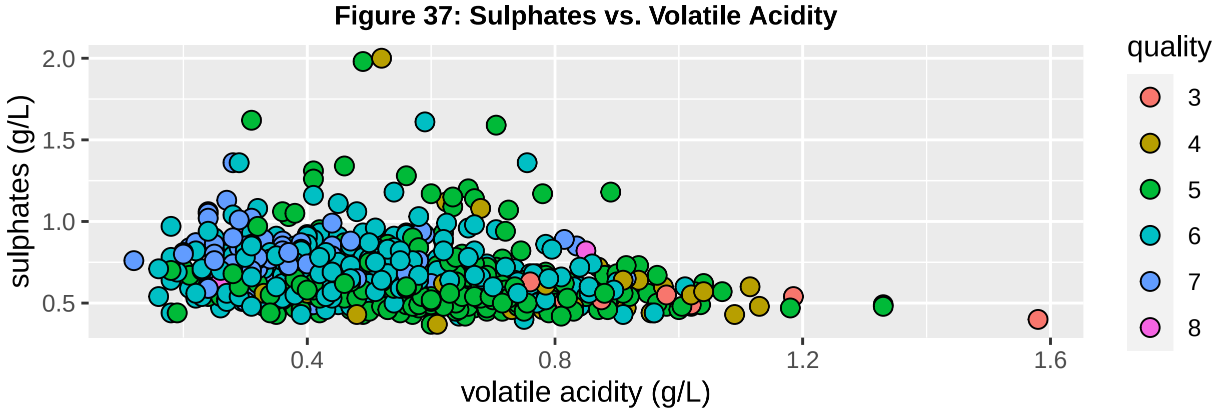
The next table shows the associated statistics.

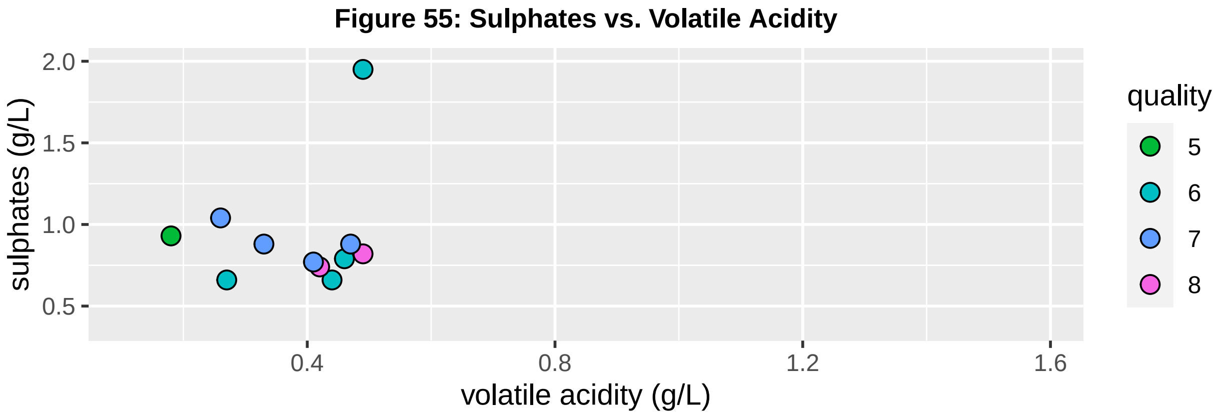
Description	Accuracy	Sensitivity	Specificity	Precision	P-Value
Linear Regression	0.86765	0.16216	0.97872	0.54545	0.47329

And finally, the following table shows statistics from the wines that were selected. Two have quality values of 8, 4 have values of 7, 4 have values of 6, and one is a 5.

	alcohol	volatile_acidity	sulphates	quality
1	14.00000	0.42	0.74	8
2	14.00000	0.49	0.82	8
3	10.90000	0.26	1.04	7
4	12.60000	0.41	0.77	7
5	12.40000	0.33	0.88	7
6	13.56667	0.47	0.88	7
7	9.90000	0.49	1.95	6
8	14.00000	0.46	0.79	6
9	13.40000	0.44	0.66	6
10	12.80000	0.27	0.66	6
11	11.30000	0.18	0.93	5

Figure 54 shows the selected wines plotted as sulphates vs. volatile acidity. Figure 37 is again repeated for context.



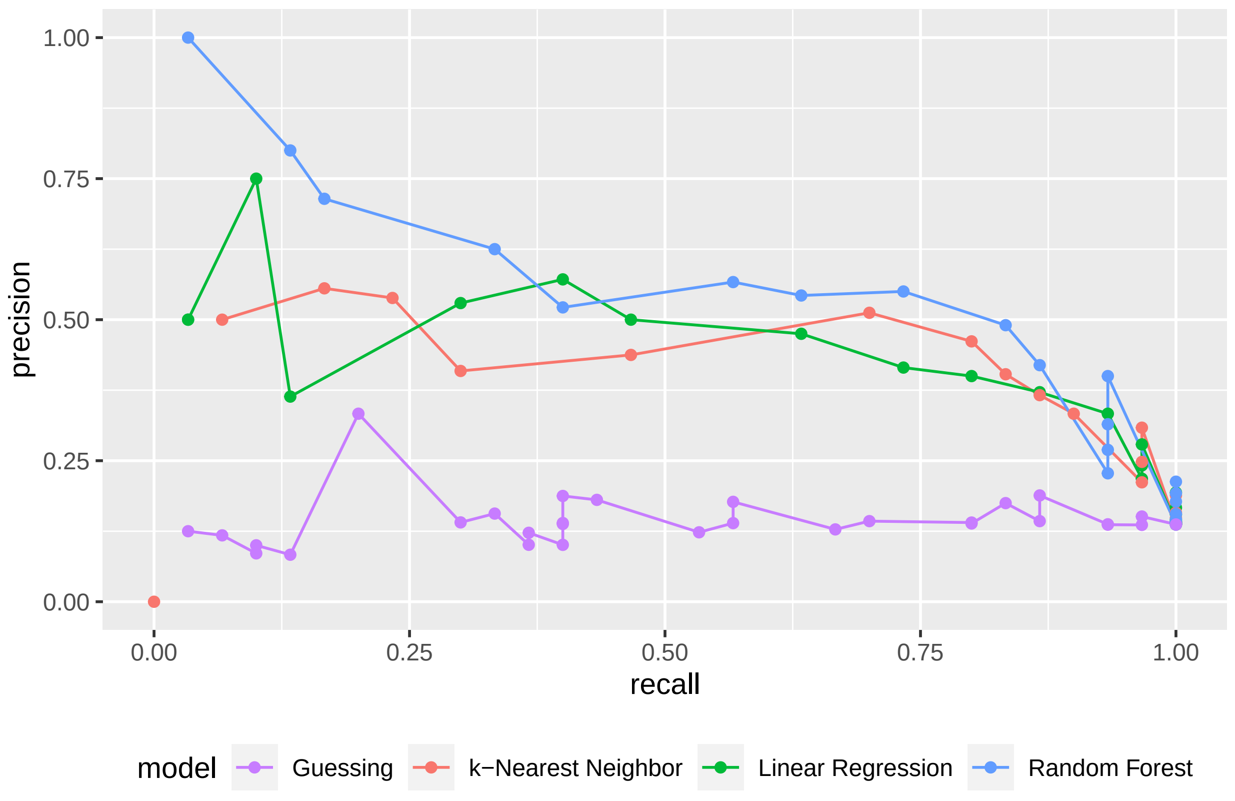


3.1 Results Discussion

The goal of this paper was to come up with an algorithm for selecting premium wines with high precision, and that goal was achieved albeit at almost half the precision that was seen in the analysis section. Performing the monte carlo simulation from 2.12.1 shows that randomly selecting 6 or more premium wines in a group of 11 occurs randomly only 0.1% - 0.2% of the time so the algorithm is a substantial improvement over random guessing at least by a factor of about 4. The unanswered question, however, is what accounts for the difference in results from the analysis section to the final results.

Changing the random seed from 1 to 2 at the point where *wines_train* and *wines_test* are split provides a likely explanation. Figure 56 shows the new Precision-Recall Plot that results from changing the seed value but leaving all of the other algorithm details the same.

Figure 56: Regression Precision−Recall Plots



Here the plot is much more consistent with what was seen in the final results suggesting that the *wine_test* set generated with the initial seed produced unusually good results in terms of precision across the board for all three algorithms.

Rerunning the various linear regression models from section 2.5 with the new seed results in the same general progression of continual RMSE reductions as seen there albeit with a bit lower RMSE. This also suggests the issue is likely related to variability in the data sets instead of a problem with the overall model selection process. The alternate linear regression results are shown in the following table.

Description	RMSE	Accuracy	Accuracy +/-1
Guess Average	0.81228	0.39269	0.94064
Top Factor	0.71515	0.52055	0.96347
Top Two	0.67388	0.52511	0.98174
Top Three	0.65301	0.52055	0.98174
Top Four	0.65621	0.52055	0.98174
Top Five	0.65495	0.52511	0.98174
Top Six	0.65216	0.53881	0.98174
Top Seven	0.64372	0.56621	0.98174
Top Eight	0.64091	0.58447	0.97717
Top Nine	0.63836	0.57078	0.97717
Top Ten	0.64204	0.55251	0.97717
All Eleven	0.64204	0.55251	0.97717

A potential way to solve this issue with data set variability would be to perform more of a k-fold cross validation strategy at the model selection step with k sets of precision recall plots used to select the final model instead of just one. However, a likely outcome of that process might be a reduced expectation of model performance instead of an actual improvement, although there is some evidence that Random Forests might be selected and show slightly better performance in that scenario. The reason the k-fold strategy was not used here was because multiplying the already multi-hour processing time by 5 or 10 was impractical.

4 Conclusion

This paper demonstrates a technique for utilizing regression algorithms for binary classification in a way that is more flexible than regular classification algorithms in terms of selecting certain characteristics, such as precision, that may be more desirable for a given purpose than accuracy.

The techniques used is this paper for selecting between different regression algorithms resulted in a significant overestimate of expected accuracy, but the paper also suggests a path forward for better accuracy estimation, and possibly better performance, in future work using a k-fold cross validation approach in the model selection phase instead of the holdout strategy used here to reduce computation time. Some of the findings from this paper might also be useful in reducing computation time as well. For instance, optimizing the tuning parameters of Random Forest regression, which was one of the biggest culprits, resulted in very little reduction of RMSE. Also, a future paper could focus exclusively on regression which would eliminate the computation time associated with Random Forest classification, another large culprit.

The motivation for this paper, selection of premium wines for personal consumption, is obviously a bit contrived, but techniques that maximize precision are useful in any number of real world applications. Specifically in terms of wine, for instance, a company or an investor with limited resources might use such techniques to select a smaller group of wines to prioritize from a much larger pool.

5 References

UCI Machine Learning Repository - Wine Quality Data Set

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.

P. Cortez et. al. - Paper available for free at this link

UC Davis, Waterhouse Lab - Fixed Acidity

UC Davis, Waterhouse Lab - Volatile Acidity

UC Davis, Waterhouse Lab - Sulfites

UC Davis, Viticulture & Enology - Citric Acid

Wine Folly - Residual Sugar

Australian Wine Research Institute - Salt in Grapes and Wine a Common Issue

Monroe et. al. - Sensing Free Sulfur Dioxide in Wine

Wine Spectator - What's the difference between sulfate and sulfites?

Smith et. al. - Sources of volatile sulfur compounds in wine

Data Science: Machine Learning

RBorist Documentation

Random Forest Documentation

Introduction to Data Science, Rafael A. Irizarry - Cross Validation

Introduction to Data Science, Rafael A. Irizarry - The Regression Line

Introduction to Data Science, Rafael A. Irizarry - Introduction to Machine Learning

Why do you need to scale data in KNN