Predicting Wine Quality Using Machine Learning

Introduction

Vinho Verde (from the latin words “green wine”) is a specific type of wine that originates from the Minho region in Northern Portugal. Vinho Verde can be distilled into several wine colors and varieties. This study derives from a dataset which was collected by five Portuguese researchers from the University of Minho, and includes the red and white colors of this wine. This data (both white and red wine datasets), for the purpose of this project, was found on the UCI Machine Learning Repository. The data is comprised of variables which are physicochemical properties of the wine; the wine is then rated on a scale from 0-10 by select wine experts. The goal of this project is to take this dataset and predict the quality of wine on a numeric scale of 0-10 using several learning algorithms. The subsequent tests and conclusions will best serve the wine producers* in this region who may wish to increase their wine sales by consistently exporting higher qualities of wine rather than low ones.

**Note: “Producers” of wine in this context directly refer to the vineyards/breweries who have the ability to change the physicochemical make-up of the wine*

The methodology for this project represents this problem as one of numeric prediction and one of classification. For numeric prediction, Linear Regression and Random Forest with Regression Trees will be implemented. Those models will be validated using the Hold-Out method, where 80% of the data is allocated to the training set, 20% of the data is allocated to the test set, and 20% of the training data will serve as the validation set. For classification, the *quality* variable will be determined to be “High” or “Low” based on a certain threshold, and K-Nearest Neighbor and Logistic Regression will be carried out. Both of these models are to be validated using K-Fold Cross Validation on the training set (80% training 20% test split). Additionally, feature engineering techniques will be utilized to determine what alternative factors/predictors could potentially influence wine quality.

Overview and Description of the Data

There were two datasets provided with identical variables; one for white wine with 4898 instances, and one for red with 1599. The separate red and white wine datasets were combined into a single dataset of 6,497 instances. In this dataset, roughly 75% of the dataset consists of white wine instances, and the remaining 25% is red. To retain the differentiation of color, a new column/variable *color* was introduced. This variable is a factor type and has levels “w” for white and “r” for red. This dataset was used to create all the initial models. Data cleaning was not necessary as the data was compiled by data scientists, and careful measures were taken to avoid inaccurate data. This was demonstrated by the lack of missing values, duplicate values, and lack of string values which did not leave opportunities for misspelled entries (*see* ***Exhibit 1*** *for a comprehensive explanation of each variable*).

```
> levels(df$color)
[1] "r" "w"
> prop.table(summary(df$color))
        r         w
0.2461136 0.7538864
```

After the data was analyzed on a high level and the variables were understood, the data was summarized using basic Exploratory Data Analysis (*see* ***Exhibit 2*** *for visualizations*). Taking a look at the *quality* variable a bit deeper, it is apparent that the central tendency is around 6. This is also evidenced by the median of the variable. The next variables to be analyzed were *alcohol* and *residual.sugar* to see how they would affect wine quality. Through visualizing both variables using boxplots, one can see there is a direct relationship between wine quality and alcohol content; higher wine qualities generally have higher alcohol content (median amount). Residual sugar was a bit more ambiguous, as there was not a direct trend between residual sugar and wine quality.

Model Building

Linear Regression

Linear regression is a scalar model which deals with one (simple) or more (multiple) explanatory variables which are used to predict the response variable. It is used when there is a linear relationship between the explanatory and response variables, and is relatively simple to interpret. The resulting line is theoretically the line of best fit through the data points.

At glance, the variables seem like they easily fall into a linear regression model. Independent and response variables all seem to easily identify themselves. Since the response variable is a numeric value, the problem seemed as though it can be solved using linear regression. In order to do that, however, the assumptions of linear regression must be tested.

The assumptions for linear regression are: *Linearity, Lack of Multicollinearity, Normal DIstribution of Errors, and Homoscedasticity*; a full linear model was initially built on the training data to analyze the previous assumptions. First, linearity was tested between the explanatory and response variables by plotting the actual data points and the model's residuals with a Residuals vs. Fit Plot (see ***Exhibit 3***). After thorough analysis, it became apparent that the plot fails to show a random scattering of points across the line $y = 0$, indicating a lack of a linear relationship between predictor and response variables. The Normal QQ plot passes the eye test, indicating a normal distribution of errors (see ***Exhibit 3***). The scale location plot had an unorthodox pattern heavily indicating lack of homoscedasticity. Furthermore, a vif test (see ***Exhibit 3***) shows that the *density* variable was heavily dependent on the *alcohol* and *residual sugar* variables. This is demonstrated when the model is run again without the *density* variable, and passes the vif test (see ***Exhibit 3***). Since three assumptions of linear regression were violated, it was discovered that linear regression was not possible to implement with this data.

Random Forest (with Regression Trees)

As Linear Regression could not be applied to this data due to the violation of certain assumptions, Random Forest was implemented to make numeric predictions for wine quality. Unlike Decision Trees, which suffer from instability, Random Forest uses *bagging* (bootstrap sampling aggregating) to create multiple trees. By using bootstrap sampling on the training data, variance is significantly reduced. At each split, Random Forest chooses a random subset of m predictors from a total of M possible inputs to decorrelate the trees. In this instance, as this was a numeric problem, the algorithm chooses m to be $p/3$ predictors when splitting the trees.

After the data was split into training, validation, and test sets, Random Forest was implemented on the training set with *quality* being the outcome variable. By default, the algorithm yields 500 trees, but after analyzing the Number of Trees vs. Error Plot (*see* ***Exhibit 4***), it was determined that 100 trees would be optimal because there is marginal improvement in error for more trees. Additionally, it is a bit simpler to interpret.

As the validation method of choice is *Hold-Out*, both models, the full and reduced, were used to make predictions on the validation set. The full model performed marginally better on the validation set, as it delivered a Mean Squared Error measure of about 0.069 while the Mean Squared Error for the reduced model was 0.07. The full model was then applied to the test set, and this yielded a Mean Squared Error value of about 0.34. According to the Variance Importance Plots (*see* ***Exhibit 4***), the variables *alcohol* and *volatile.acidity* were the most relevant when improving the Mean Squared Error metric and reducing Residual Sum of Squares at each split.

Logistic Regression

Logistic Regression can essentially be thought of as regression when the desired outcome variable is categorical, not numeric. This algorithm is useful because it only returns values between 0 and 1, known as the log-odds. Coefficients work similarly to Linear Regression, except they are found by using maximum likelihood estimators. To get a better understanding of the relationship between the coefficients and the log odds, the coefficients can be exponentiated. Once the coefficients have been estimated, a probability threshold can be decided for what each predicted outcome should be classified as. Akaike's Information Criterion (AIC) is a popular metric when deciding between which logistic regression models to be used; typically, the lower the better.

Before proceeding to applying logistic regression, the *quality* variable had to be changed from a continuous to categorical type. As the 3rd quartile of the *quality* variable was 6, it was determined that any wine quality that was greater than or equal to 6 be classified as 'High' while the rest be classified as 'Low'. 66.5% of all qualities ended up being classified as 'High' while the rest were classified as 'Low'; this number served as a baseline for the accuracy metric.

Two models were built using logistic regression, one using all of the predictors, then one using only the significant ones (*See* ***Exhibit 5***). As both models were built on the training data, they were both validated through K-Fold Cross Validation with K = 5 to avoid overfitting and using a probability threshold of .3 (more detailed explanation in next paragraph). The reduced model ended up having better metrics overall, with accuracy being 69%, sensitivity of high being 81%, and precision of high being 55% (*See* ***Exhibit 5*** *for comparison between the models*).

To determine the appropriate probability threshold for both models, the Receiver Operating Characteristic (ROC) Curve was analyzed (*See* ***Exhibit 5***). As this models the trade-off between sensitivity and specificity, a threshold of 0.3 was chosen to optimize sensitivity. This is the metric to emphasize for this specific business situation because False Negatives are costly. In this context, this would mean a producer of wine would actually be high quality, but incorrectly identified as low quality.

Interestingly, when analyzing how each coefficient affected the log-odds, it was determined that *volatile.acidity* was the most significant, as it increased the odds of classifying the wine as low by a factor of almost 61. Now that the reduced model was determined to be optimal in this situation, it was applied to the test set to serve as a proxy for performance on future, unseen data. The

application yielded metrics of 70% (beats baseline), 84%, and 56% for accuracy, sensitivity of high, and precision of high, respectively (*See **Exhibit 5** for detailed comparisons and relationships between quality and coefficients*). The test accuracy beats the baseline, and sensitivity is optimized while sacrificing precision.

K-Nearest Neighbor

Another algorithm typically used for classification is K-Nearest Neighbor (KNN). First, the distance between the new observation and all other observations are calculated, then, for the K closest observations, the algorithm determines majority classification and assigns that classification to the new observation.

For KNN particularly, all continuous variables should be normalized since Euclidean distances between predictors of different scales will have uneven influence. The categorical values were then converted to 0 and 1 for white and red wines, respectively. The same classification rules for *quality* as Logistic Regression were used, with high quality being greater than or equal to 6.

After the data was split between training and test sets, K-Fold Cross Validation was used on the training data to determine the optimal hyperparameter K and also to avoid overfitting on the test set. After implementing 5-fold cross validation, the graph comparing the accuracy, sensitivity of high, and precision of high of each K was analyzed *(see **Exhibit 6**)*. As K increases, accuracy and precision share the same behavior of initially decreasing but exhibits constant fluctuation. Sensitivity, however, increases as K increases. According to the business goal, as producers, false negatives are costly. The KNN model should ideally try to minimize them to maximize sensitivity; this can be controlled by determining the optimal K.

As a result, K = 17 was chosen to be optimal because it delivered the highest sensitivity but also very decent accuracy and precision. The model was then applied to the test set, and test accuracy happened to be 75%. Of the total 901 high quality wines, the model was able to capture 77.2% of them (precision). Moreover, when predicting wine with high quality, the model was correct 85.5% of the time (sensitivity) *(see **Exhibit 6**)*. This model returns a better result as compared to the baseline performance of 66.5% for predicting all “high”. It is also better than the performance on the training set when all accuracy, sensitivity and precision all increase by around 1%.

Performance Evaluation/Model Comparison

As this methodology was carried out using algorithms from both a numeric prediction standpoint and classification standpoint, it would be fair to discuss them in such categories. Starting with algorithms used for numeric prediction, Random Forest (with regression trees) was implemented on the test set. Ultimately, the model performed decently, yielding a mean squared error of .34.

Continuing on, after the quality of the wines was classified accordingly as “High” or “Low”, classification algorithms such as KNN and Logistic Regression were implemented. Through K-Fold Cross Validation (K = 5) on the training data, it was determined that the optimal hyperparameter K in this context would be K = 17. After applying KNN to the test data and analyzing the confusion matrix, accuracy was 75%, precision of high was 77.2%, and sensitivity of high was 85.5%. The logistic regression model was also validated using K-Fold Cross Validation, and the reduced model was deemed optimal. When that model was applied to the training set, accuracy was 70%, sensitivity of high was 84%, and precision of high was 56%.

Overall, out of the two classification models, it is apparent that the KNN model performed better than the Logistic Regression Model in this business context.

Feature Engineering/Dataset Manipulation

The dataset used for the main part of the procedure was comprised of the white wine data and the red wine data; *color* then became a new variable (0 is white, 1 is red). After further creating models using this data and analyzing the problem further, *flavor* and *total.acidity* were determined to be desired variables in the data. These variable were to be used to see if they were meaningful in predicting *quality* or not. The *flavor* variable was then added to the dataset, and had been given values such as 'Dry', 'Off-Dry', 'Semi-Sweet', or 'Sweet' (levels of the variable, as it is a factor type). These classifications of *flavor* were determined by the *residual.sugar* variable (measure of sweetness). If residual sugar the wine was between 0 and 9, it is determined to be 'Dry', if it fell between 9 and 18, it is 'Off Dry', if it was between 18 and 50, it is 'Semi Sweet', and if the value was greater than 50, it was 'Sweet.' This scale of measurement was derived from a general spectrum of flavor on wine. The variable *total.acidity* was simply determined to be the sum of the *fixed.acidity* and *volatile.acidity* variables.

Random Forest (For Classification) and KNN

For this portion of the project, the newly created dataset was used. The algorithms of choice are KNN and Random Forest (for classification this time). The objective here was to see if the new variables, *flavor* and *total.acidity*, were important for predictions. The *quality* variable is classified the same way, with the threshold of greater than or equal to 6 being "High" and the rest being of "Low" quality.

Regarding KNN, the model building process stays the same as mentioned previously, however, the validation on the training set had a slight decrease of roughly 1% regarding the three metrics. When coming to the test error, the exact same sensitivity of high as the previous model built above was shown, of 85.5%. Nevertheless, accuracy and precision of high were lower, with only 74% and 76% respectively - less than around 1% comparing to the previous model *(see **Exhibit 7**)*. As a result, this KNN model was not an improvement compared to the previous KNN model; thus Random Forest with classification was tested next.

Random Forest was then implemented on the training data and created 500 trees by default. As opposed to Random Forest with Regression Trees, 2($\sqrt{p}$ predictors, now with 13 variables) were considered at each split using the classification variant of the algorithm. The out-of-bag (OOB) estimated test error rate is relatively small, being 17.03%; this entails nearly 83% accuracy. The training confusion matrix is also shown *(see **Exhibit 8**)*.

Afterward, predictions were made on the test set using this model *(see **Exhibit 8**)*. The prediction yielded lower false negatives (91) as well as lower false positives (148), which led to tremendous improvements on all three metrics: 82% accuracy, 89% sensitivity of high and 83% precision of high. This actually turned out to be the best performing model. For the purpose of testing the importance of new variables however, the Variable Importance Plot *(see **Exhibit 8**)* shows that *total.acidity* has a moderate importance when improving accuracy and decreasing Gini Index, which is reasonable since it is the sum of fixed and volatile acidity. *Flavor* seems to rank as one of the least important predictors, which indicates that *flavor* is not a relevant variable to the predictions.

Conclusions/Takeaways

If the producer of wine was concerned with predicting the actual numeric wine quality, the Random Forest (with Regression Trees) model would be useful to them. When implemented on the test data, the Mean Squared Error happened to be .34. This entails that, on average, a producer of wine can predict the numeric value of the wine quality, with a slight variance of .34 either above or below. This is beneficial because there is not much difference between a wine that is, for example 7.34 and 7 in quality for a certain consumer. The numbers are relatively close to one another, so consumers should theoretically portray the same amount of satisfaction. It is also useful to note that if a producer wanted to improve wine quality, the producer can alter the alcohol content and volatile acidity as these were the most important variables when predicting quality based on the Variable Importance Plot. This could be because the experts generally liked their wines to be strong, as evidenced in the initial Exploratory Data Analysis.

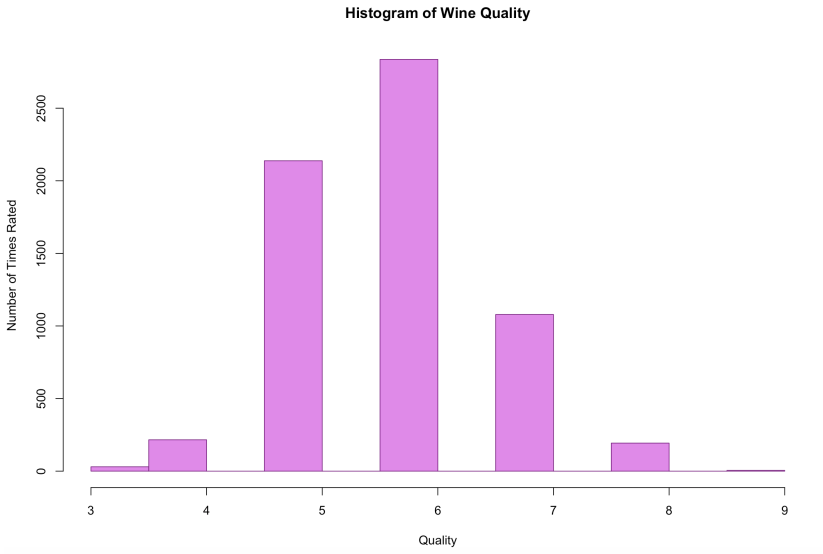
If the producer was concerned with generally predicting “High” or “Low” quality (with the original data), the KNN model with K = 17 would be the most useful to them. When applied to the test data, the model yielded 75%, 77.2%, and 85.5% for accuracy, precision of high, and sensitivity of high, respectively. If the new dataset was used to predict “High” or “Low”, Random Forest would be optimal as it delivered 82%, 89%, and 83% for accuracy, precision of high, and sensitivity of high, respectively when it was applied to the test data. In this context, it is more costly for a producer classified a wine quality as “Low” when it was actually “High” (seek to minimize False Negatives) so the model should optimize sensitivity of high. Intuitively, if the wine was classified as “Low” when it was actually “High”, a consumer or distributor may not opt to buy that particular bottle of wine, which translates to lost sales for the producer.

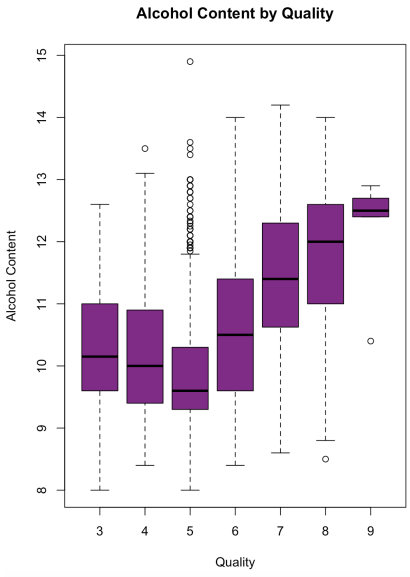
Appendix

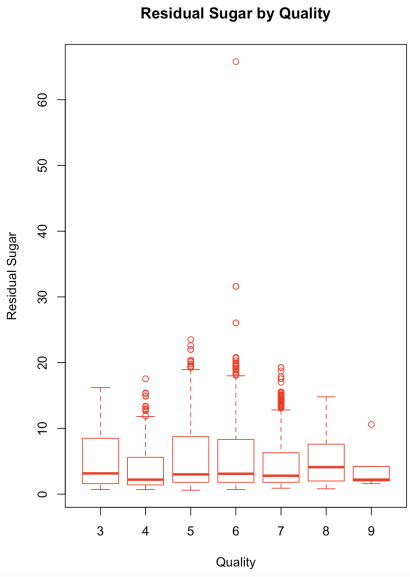
Exhibit 1: Overview of Variables

Variable Name	**Description**	**Units**	**Data Type**
Quality	Outcome variable, based on sensory data, **target variable**	Scale of 0-10 (entirely sensory)	int
Citric acid	Natural preservative, adds sour taste	g/dm^3	num
Residual Sugar	Measure of sweetness	g/dm^3	num
Chlorides	Sodium Chloride content	g/dm^3	num
Free Sulfur Dioxide	Free SO_2	mg/dm^3	num
Total Sulfur Dioxide	Free and Bound SO_2 used as preservative	mg/dm^3	num
Density	Weight per volume of wine, determined by alcohol and sugar	g/dm^3	num
pH	Measure of strength of acids	pH higher than 7 is acidic, while pH less than 7 is basic	num
Sulphates	Potassium sulphate, used to prevent discoloration	g/dm^3	num
Alcohol	Alcohol	Percent by metric volume	num
Fixed Acidity	Tartaric acid	g/dm^3	num
Volatile Acidity	Gaseous Acidic element in wine (scent)	g/dm^3	num
Color	Color	NA	factor

Exhibit 2: Exploratory Data Analysis/Visualizations





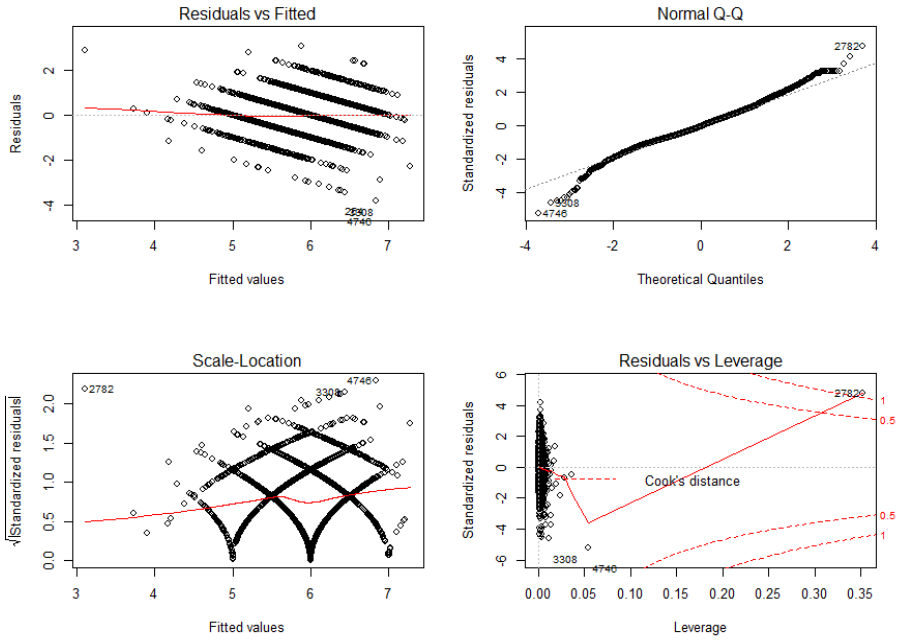


Descriptive Statistics for *quality* and *color*

```
   quality         color
Min.   :3.000   r:1599
1st Qu.:5.000   w:4898
Median :6.000
Mean   :5.818
3rd Qu.:6.000
Max.   :9.000
```

Exhibit 3: Linear Regression

Linear Regression Initial Plots



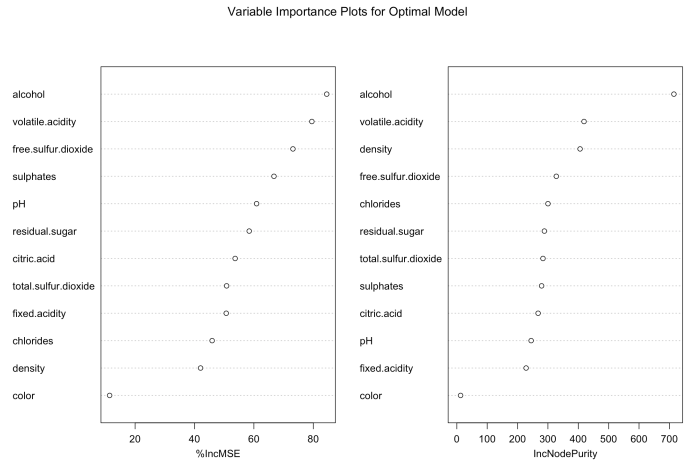
VIF test one

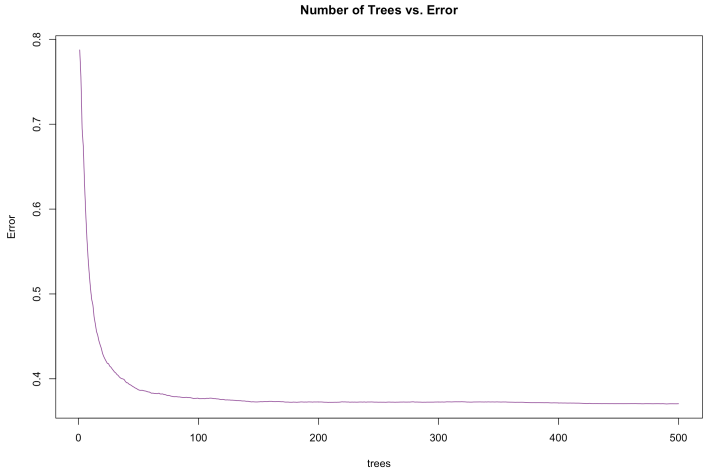
```
> vif(qpred)
    ï..fixed.acidity     volatile.acidity          citric.acid
            2.612817             1.129962             1.151897
      residual.sugar  free.sulfur.dioxide total.sulfur.dioxide
           12.152487             1.786346             2.239028
             density                   pH            sulphates
           27.475237             2.137241             1.136952
             alcohol
            7.706337
>
```

VIF test without density

```
> vif(qpredadj2)
   ï..fixed.acidity    volatile.acidity      residual.sugar
           1.242182            1.030666            1.375945
free.sulfur.dioxide                  pH           sulphates
           1.147581            1.295197            1.033553
             alcohol
           1.303358
>
```

Exhibit 4: Random Forest with Regression Trees





Full Model

```
Call:
 randomForest(formula = quality ~ ., data = train, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 0.3705526
                    % Var explained: 51.29
```

Reduced Model

```
Call:
 randomForest(formula = quality ~ ., data = train, ntree = 100,      importance = TRUE)
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 4

          Mean of squared residuals: 0.3814941
                    % Var explained: 49.85
```

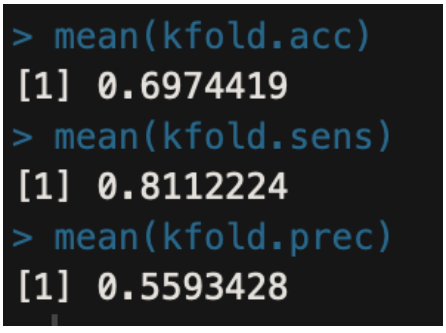
Exhibit 5: Logistic Regression

Full Model

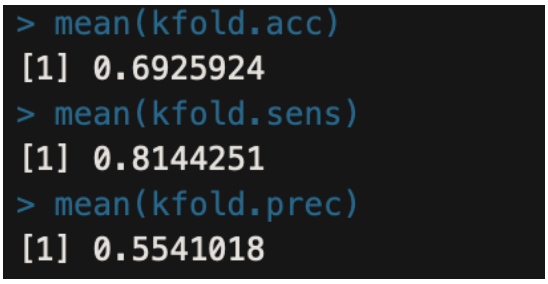
```
Call:
glm(formula = as.factor(quality) ~ ., family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5575  -0.8184  -0.4459   0.9082   3.4052

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.335e+02  4.938e+01  -2.703  0.00686 **
fixed.acidity       -1.144e-01  5.635e-02  -2.030  0.04237 *
volatile.acidity     4.631e+00  3.261e-01  14.202  < 2e-16 ***
citric.acid          5.496e-01  2.840e-01   1.935  0.05297 .
residual.sugar      -1.263e-01  2.119e-02  -5.961 2.51e-09 ***
chlorides            8.432e-01  1.168e+00   0.722  0.47019
free.sulfur.dioxide -1.363e-02  2.876e-03  -4.741 2.12e-06 ***
total.sulfur.dioxide 5.500e-03  1.175e-03   4.682 2.84e-06 ***
density              1.447e+02  5.012e+01   2.888  0.00388 **
pH                  -8.848e-01  3.312e-01  -2.671  0.00756 **
sulphates           -1.925e+00  2.945e-01  -6.536 6.34e-11 ***
alcohol             -7.953e-01  6.674e-02 -11.917  < 2e-16 ***
colorw               6.663e-01  2.110e-01   3.158  0.00159 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6822.2  on 5196  degrees of freedom
Residual deviance: 5388.5  on 5184  degrees of freedom
AIC: 5414.5

Number of Fisher Scoring iterations: 4
```

Reduced Model

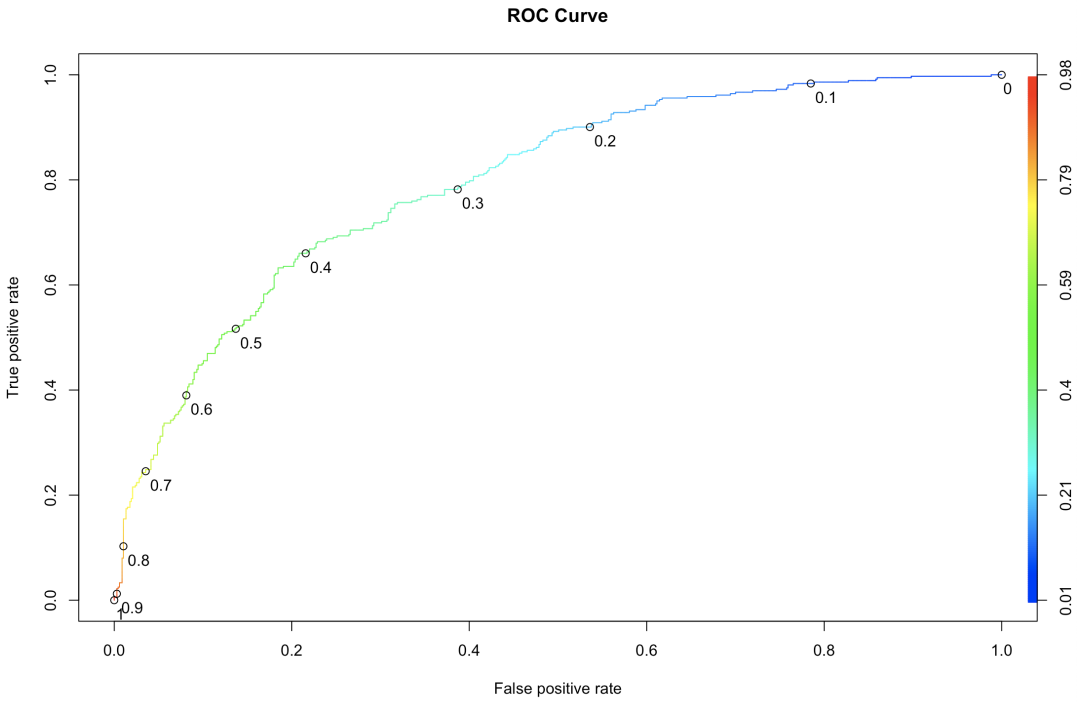
```
Call:
glm(formula = as.factor(quality) ~ volatile.acidity + residual.sugar +
    free.sulfur.dioxide + total.sulfur.dioxide + sulphates +
    alcohol, family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5955  -0.8210  -0.4434   0.9228   3.3838

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)          8.6877445  0.4695820  18.501  < 2e-16 ***
volatile.acidity     4.1187160  0.2445806  16.840  < 2e-16 ***
residual.sugar      -0.0665916  0.0083189  -8.005 1.20e-15 ***
free.sulfur.dioxide -0.0158986  0.0027738  -5.732 9.95e-09 ***
total.sulfur.dioxide 0.0075452  0.0009347   8.072 6.89e-16 ***
sulphates           -1.8310502  0.2506923  -7.304 2.79e-13 ***
alcohol             -0.9423230  0.0379221 -24.849  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6822.2  on 5196  degrees of freedom
Residual deviance: 5414.3  on 5190  degrees of freedom
AIC: 5428.3

Number of Fisher Scoring iterations: 4
```



Validation for Full Model



Validation for Reduced Model



Test Confusion Matrix/Metrics

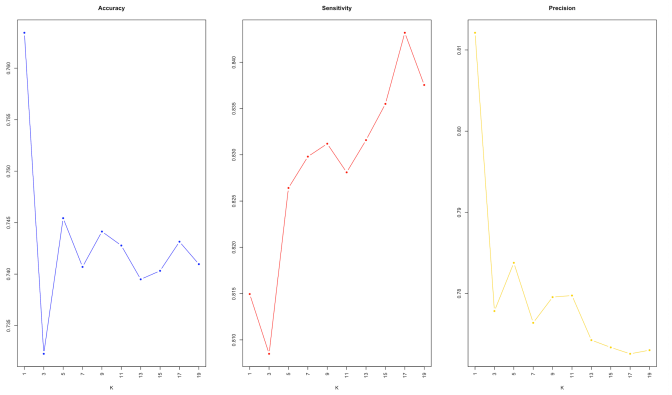
	pred.class.test	
actual	High	Low
High	499	315
Low	75	411

```
> data.frame(acc.log.test, sens.log.high.test, prec.log.high.test)
  acc.log.test sens.log.high.test prec.log.high.test
1          0.7           0.845679          0.5661157
```

Exhibit 6: K-Nearest Neighbor

Metrics Comparison for each K

	K	accuracy	sensitivity	precision
1	1	0.7634409	0.8149515	0.8121191
2	3	0.7322569	0.8084944	0.7778431
3	5	0.7454308	0.8264142	0.7837856
4	7	0.7406983	0.8298022	0.7763941
5	9	0.7441322	0.8311948	0.7795592
6	11	0.7427756	0.8281016	0.7797560
7	13	0.7394931	0.8315853	0.7742658
8	15	0.7403180	0.8355004	0.7733672
9	17	0.7431466	0.8431835	0.7725779
10	19	0.7409641	0.8375421	0.7730241



Test Error on the Optimal Model of K = 17

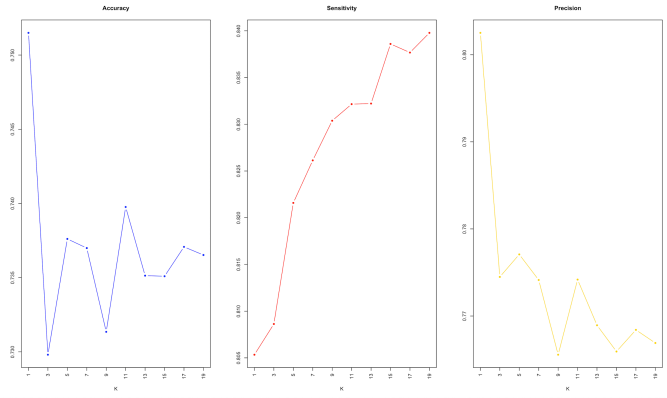
	pred.class	
	high	low
high	696	118
low	205	281

	acc	sens.high	prec.high
1	0.7515385	0.8550369	0.772475

Exhibit 7: KNN (using new dataset)

Metrics Comparison for each K

	K	accuracy	sensitivity	precision
1	1	0.7514931	0.8053422	0.8025152
2	3	0.7298080	0.8086283	0.7744904
3	5	0.7376085	0.8215846	0.7770754
4	7	0.7369928	0.8261345	0.7741381
5	9	0.7313483	0.8303793	0.7655587
6	11	0.7397605	0.8321536	0.7741833
7	13	0.7351337	0.8322177	0.7689388
8	15	0.7350956	0.8385950	0.7659259
9	17	0.7370739	0.8376739	0.7684095
10	19	0.7365254	0.8397812	0.7669026



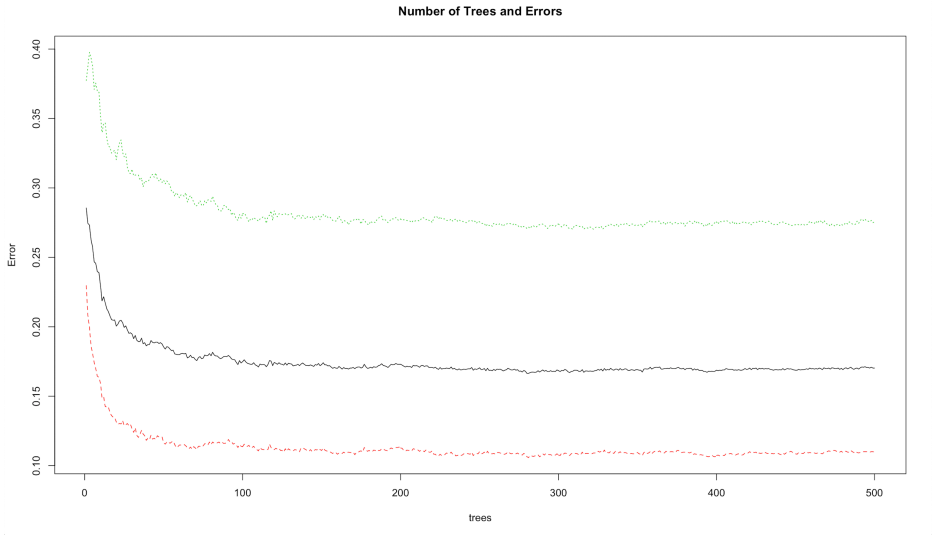
Test Error for K = 19

	pred.class	
	high	low
high	696	118
low	222	264

	acc	sens.high	prec.high
1	0.7384615	0.8550369	0.7581699

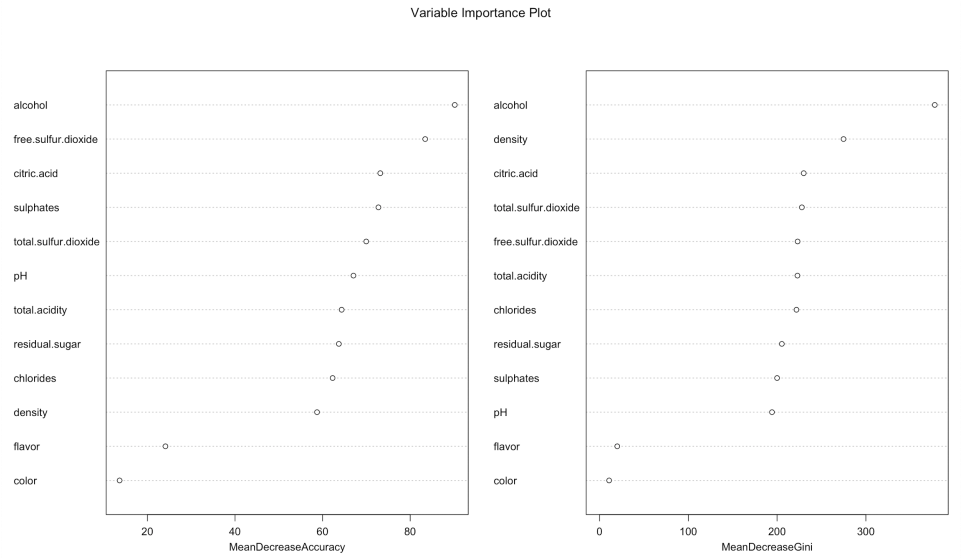
Exhibit 8: Random Forest with Classification (using new dataset)

```
Call:
 randomForest(formula = as.factor(quality) ~ ., data = train,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 17.03%
Confusion matrix:
     high  low class.error
high 2937  362   0.1097302
low   523 1375   0.2755532
```



Test Error

	pred.class				
	high	low	acc	sens	prec
high	723	91			
low	148	338	1 0.8161538	0.8882064	0.8300804



Additional Resources (used mainly to understand some variables)

Admin. “Understanding Wine Acidity.” *Winemaker's Academy*, 25 June 2018, winemakersacademy.com/understanding-wine-acidity/.

“The Taste of Wine: Acid, Sweetness, and Tannin.” *GuildSomm*, www.guildsomm.com/public_content/features/articles/b/jamie_goode/posts/the-taste-of-wine-acid-sweetness-and-tannin.

“Wines From Dry to Sweet (Chart).” *Wine Folly*, 14 Apr. 2016, winefolly.com/tutorial/wines-from-dry-to-sweet-chart/.