Determining Red Wine Quality Based on Physiochemical Attributes Using Logistic Regression

Nicholas Abad

January 21, 2018

Abstract

Taking inspiration from studies conducted by Brochet (2001) and Cortez et. al (2009), an experiment was conducted to determine whether eleven objective, physiochemical attributes can determine whether the overall quality of a certain red wine could be categorized as being "good" or "bad". Through the use of a self-created ensemble method that incorporates the voting method of a number of logistic regression models, parameters, namely the learning rate, percentage of training observations used, and the number of models within a single ensemble method, were altered to study the impact these had on the accuracy, precision, and recall values. After experimentation, it was found that the self-created methods had a peak accuracy of 0.78125, a peak precision rate of 1.0, and a peak recall rate of 1.0, each of which occurred when using completely different parameters. Similarly, when put into individual linear models, none of the changeable parameters were statistically significant, which was also not ideal. Due to computational limitations, additional studies could be conducted in the future to further iterate through these parameters and train these models on larger datasets in order to better predict whether the quality of wine should be categorized as being good or bad.

1 Introduction

In a famous experiment conducted in 2001 [1], Frederic Brochet, a then PhD candidate at the University of Bordeaux II in France, set out to study how university-level wine tasting students tasted and perceived red wine. In this specific experiment, Brochet had two bottles of wine, one of which was described as being a very expensive wine while the other was described as being very cheap. After these students tasted the wines coming from the two separate bottles, they were then asked to describe the differences between the two. The wine within the expensive bottle was met with high praise and was described as being "complex" and "rounded" while, on the other hand, the wine from the cheap bottle was looked down upon and was described as being "weak" and "flat". However, unbeknownst to these students, Brochet previously emptied out the expensive bottle, proceeded to fill this expensive bottle with the wine from the cheap bottle, and then served it to the wine tasting students. In reality, these subjects were ultimately tricked and seemingly based their descriptions of each wine on the price tag moreso than anything.

If these students could be fooled into thinking that one wine is better than the other simply from of the bottle it is poured from, the average person will more than likely fall subject to that same exact fault, granted that they have not studied the art of wine tasting. Because of this, Cortez et. al in 2009 [2] conducted a comprehensive study in order to determine whether studying absolute, unchangeable physiochemical attributes, such as pH levels and acidity rates, could better predict the quality of wine, rather than relying on values that could theoretically fluctuate such as the price of wine in the aforementioned example. By using these these two studies as motivation, I thus wanted to experiment whether I could use a machine learning method, such as logistic regression, in order to determine the quality of wine based on 11 physiochemical attributes. In order to complete my objective of wine quality classification, I decided to create my own logistic regression model that implements gradient descent when training the model, create an ensemble method that incorporates a simple voting method, and produce measures for accuracy, precision, and recall. In order to analyze my results, I then used R in order to produce linear models and determine whether changing the parameters plays a part in determining these three measures.

2 Methods

2.1 Dataset Attributes and Description

In order to develop a logistic regression model to determine the quality of a red wine based on its physiochemical attributes, Cortez et. al 2009 [2] originally gathered a dataset that was made publicly available within the University of California Irvine Machine Learning Repository website [3]. The red wine dataset that was used within this experiment contains a total of 1599 observations with 12 unique variables. Of these 1599 observations, there were luckily no missing values or alarming extreme values to account for.

Of those 12 variables, the attribute labeled *quality*, which will be considered to be the dependent variable, was graded on a scale between 0 and 10 by 3 expert judges, with a score of 0 meaning that the wine has poor quality and while a score of 10 would mean that the wine has excellent quality. With each judge independently grading these 1599 different red wines, the average of each wine was taken between the three judges in order to come up with a final grade that was inputted into the dataset. Although this grade may be considered to be subjective, using experts within the wine field as well as aggregating their results should alleviate this issue.

In regards to the remaining 11 explanatory variables, specific scientific measurements were taken by Cortez et. al within a lab setting in a uniform way to ensure that these values remained as objective as possible. The 11 explanatory variables that will be used to determine the quality of a certain red wine are as follows: *fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.*

2.2 Data Preprocessing

With an objective to determine the quality of red wine, it was first necessary to determine what it would mean within our experiment for a certain red wine to be considered of *good* or *bad* quality. Due to the response variable being a continuous number ranging from 0 to 10 and also due to only wanting a binary response variable, the median of the *quality* column was taken and this value was used as the threshold to determine whether a particular red wine were to be considered *good* or *bad*. By using the median rather than the mean of the dataset as our threshold, this would ensure that the dataset is thus balanced meaning that over-sampling and/or under-sampling techniques would not need to be used such as SMOTE or Tomek Links. In this particular case, the median was found to be 6.0, which implies that any *quality* value greater than or equal to 6.0 would be categorized as *good* while any wine less than 6.0 would be categorized as *bad*. To further put this into a numerical binary format, those wines categorized as *good* were coded to be a 1 within our new dataset while all other wines were coded to be a 0.

With the binary response variable now in place, the next logical step was to conduct exploratory analysis to both visualize the data at hand and try to determine if there are any clusters that determine whether a wine would be categorized as a 0 or a 1. Unfortunately, and as exemplified below in *Figure 1*, there were no telling clusters that accurately depicted the quality.

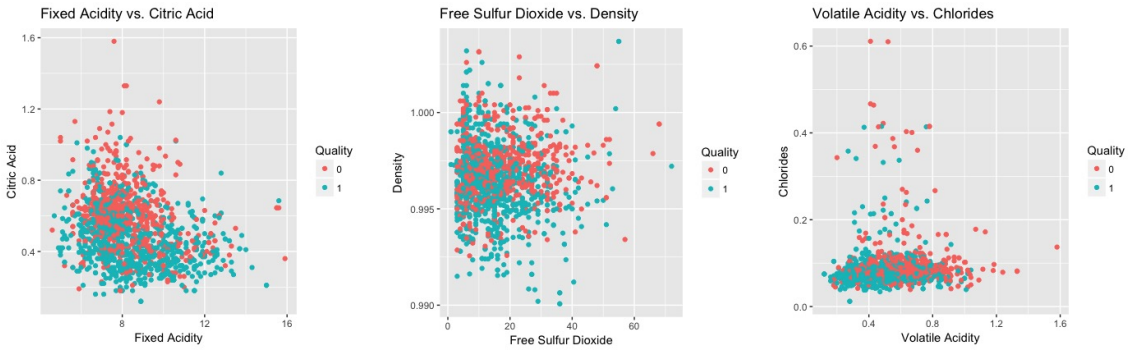


Figure 1: *Examples of exploratory plots that show that good wine (labeled as 0) and bad wine (labeled as 1) are not very distinguishable from each other. Due to this, reliable clusters could not be generated.*

In terms of the explanatory variables, it was found that by scaling each explanatory variable value to be between -1 and 1, the accuracy generally improved and in order to implement this

within the Python script, Scikit-Learn's MinMax scaler was used. This experimentation regarding the accuracies between scaled and non-scaled explanatory variable values is not covered in this paper but it should be noted that the accuracy improved up to 11% for some observations.

With each explanatory variable now being scaled and with the response variable, or wine quality, being categorized as either a 1 or 0, it was then necessary to split our entire dataset into a training and testing set. Before doing so, however, the dataset was first randomized in order to eliminate any potential sampling bias within the model. Once the randomization was completed, our test set was then chosen to have roughly 20%, or 320, of the 1599 observations while our training set consisted of the remaining 80% or 1279 observations.

2.3 Logistic Regression

Because our response variable can only take one of two possible values, one of the most logical statistical methods to use when training and testing our dataset would be a logistic regression model. Through the use of logistic regression, probabilities are produced for a given testing observation and are given in the form:

$$p(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+...+\beta_n X_n)}} \tag{1}$$

where $p(Y = 1|X)$ denotes the probability of the response variable resulting in 1 and where β_i denotes each variables' weight. In this specific case, β_i is calculated using gradient descent, which incorporates a maximum likelihood estimation and is additionally an iterative process. As you can see in *Equation 1*, the amount of β_i's is exactly the amount of features within your training and/or testing set plus an additional β_0 that represents the intercept value. Therefore, if your dataset has a total of N features, this would result in a total of $N + 1$ different βs.

By being given a training set and calculating the experimental β values, it is then possible to calculate the probability of $Y = 1$. In order to do so, plug in testing observation values that correspond to the particular X_i value in *Equation 1*. As an example, say that a logistic model is trained in which $\beta_0 = 0.1$, $\beta_1 = 0.2$, and $\beta_2 = 0.3$. Additionally, say that a test observation has values of 10 and 0.5 respectively. *Equation 1* would then turn into the following:

$$p(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2)}} \implies \frac{1}{1 + e^{-(0.1+(0.2\times 10)+(0.3\times 0.5))}} = 0.9046505 \tag{2}$$

In our specific model, however, there is much more than just two explanatory variables as is discussed previously in *Section 2.1*. Because the dataset consists of 11 variables that are attempting to predict the quality of wine, 12 individual β's are needed. In order to get updated β values, each β is initialized at 0. From there, the gradient is then computed and we move in the negative direction in order to get further down the graph. With this new location on the graph, we update the model and keep repeating this process until the amount of iterations is reached. Once this is done for all β's, the final weights will be computed and these weights will be used for testing our dataset.

2.4 Ensemble Method

An ensemble method can roughly be defined as a technique used within a classification problem that first creates a multitude of different machine learning methods and comes up with a classification decision based on the results of each of these individual methods. Through the aggregation of each of the results in each of these individual models, the aggregated result is generally improved when being compared to any individual result. In order to come up with a joint decision, there are several methods that many people use but one of the easiest to understand is a simple voting method, which we will implement and can be exemplified below.

As an example of a simple ensemble method that implements multiple machine learning methods as well as the voting method, say that one wants to implement an ensemble and decides to use a decision tree, a logistic regression model, and a neural network in order to determine whether an observation is classified as "good" or "bad" in some context. In this example, each of these models are trained on a random subset taken **with replacement** from the original training set. Once these models are trained, a test observation (or test observations) is then put through each of these models and, as a result, say that the decision tree as well as the neural network categorizes

this test observation as "good" while only the logistic regression model categorizes the observation as "bad". For this particular test observation, the joint decision would then be to predict that the individual test observation is "good" due to the fact that 2 out of the 3 individual models labeled it as "good".

In a similar fashion, the experiment detailed in this paper also uses an ensemble method with voting in order to tackle the classification problem of determining whether red wine could be categorized as *good* or *bad* based on its scientific attributes. However, rather than using the three different machine learning methods exemplified above, this experiment incorporates a number of logistic regression models and attempts to come up with a decision based on the results of these logistic regression models. By implementing an ensemble method, the joint decision should ideally be improved.

In order to create the ensemble method described above, the first step taken was to create a Python script that loads in the entire training dataset. As previously noted, the entire training dataset consists of 1279 observations, which accounts for roughly 80% of the original dataset. With this entire training set in hand, a random sample **with replacement** is thus taken from this 80% in differing levels of proportion. The exact amount within each of these random samples is further specified within the *Results* section.

With this random sample in hand, a single logistic regression model is then trained using this random sample. Using the original test set that consists of the remaining 20% of the original dataset, probabilities and predictions are then made in order to determine whether this specific logistic regression model categorizes each test observation as a 1 or a 0. This process of taking a random sample with replacement from the training set, training a logistic regression model on this said random sample, and testing the exact same testing observations on this model is repeated a number of times, using a threshold of 0.5 meaning that for a test observation, if the probability exceeds 0.5, the predicted outcome will be a 1. Similarly, if the probability for a test observation is less than or equal to 0.5, the predicted outcome will then be a 0.

With the results of each of these logistic regression models now available, a simple voting method is then implemented in which the ensemble's final prediction is determined by which category has more votes. By comparing the ensemble's final prediction to the true value of each of the testing observations, measurements such as accuracy, precision, and recall could then be calculated. To be specific in how each of these measurements were calculated, the formulas can be seen below:

$$Accuracy = \frac{|TruePositive| + |TrueNegative|}{|TruePositive| + |TrueNegative| + |FalsePositive| + |FalseNegative|} \tag{3}$$

$$Precision = \frac{|TruePositive|}{|TruePositive| + |FalsePositive|} \tag{4}$$

$$Recall = \frac{|TruePositive|}{|TruePositive| + |FalseNegative|} \tag{5}$$

3 Results

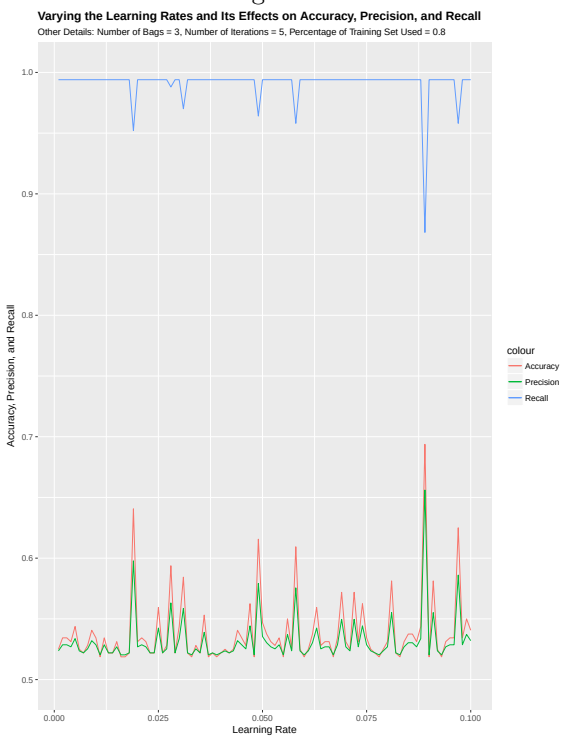
In an attempt to classify the quality of a wine, there were originally and precisely four parameters that needed to be specified before coming to a conclusion regarding the experiment. These four parameters were the learning rate, the number of logistic regression models used in an ensemble method, the percentage of the original training set that was used to train each logistic regression model, and the number of iterations used to update the β values. Due to computational limitations, it was decided to train each logistic model using 5 iterations, which ultimately negated this as being a changeable parameter.

3.1 Varying the Learning Rates

Through experimentation, implementation, and visualization in Python and R, the first step taken was to vary the learning rates to see how this effects the accuracy, precision, and recall rates within each of these models. By holding all other variables constant and in particular making the number of logistic regression models used within our ensemble equal to 3, the number of iterations used to update the β values be equal to 5, and training each logistic regression model on 80% of the original training data with replacement, it was found that varying the learning rates did not quite

have a positive linear effect on the accuracy, recall, or precision, which can be exemplified in the figure below. However, when using accuracy as the response variable in a simple linear model and by using the learning rate as the only explanatory variable, the learning rate was found to be statistically significant with a p-value of $5.28e^{-8}$ but this resulted in only an adjusted R^2 value of 0.1349 meaning that the model is not good and does not properly explain the variation.

Figure 2:



As exemplified in Figure 2, it was interesting to note, however, that the accuracy and precision values seemed to be quite consistent with each other. Although having very similar formulas, the correlation coefficient between the two was found to be 0.9778866 while the correlation coefficient between accuracy and recall was found to be -0.798205. Because of these values being relatively similar to 1 and -1 respectively, this makes sense since a peak in accuracy or precision almost certainly results in a valley in recall, as exemplified. Specifically when the learning rate was 0.089, this brought forth a global maximum in the accuracy and precision graphs but meanwhile resulted in a global minimum for the recall graph.

3.2 Varying the Amount of Logistic Regression Models in an Ensemble

Additionally, I wanted to vary the amount of logistic regression models chosen to take part in the ensemble next. Once again, by keeping the remaining parameters constant, specifically by having the number of iterations being equal to 5, the percentage of training data used being equal to 0.8, and the learning rate being equal to 0.089, which was found to give the highest accuracy when previously varying through the learning rates, I created a Python script that uses up to 99 different logistic models within a single ensemble method starting at 1 and incrementing by 2. In doing so, I gathered the resulting accuracy, precision, and recall values for each of these different ensemble methods when being compared to the original testing set. Similarly to when varying the learning rate, varying the amount of bags did not seem to have a large effect on predicting these three measures as well. Despite this, however, and while varying the amount of logistic models that each ensemble used, data was also gathered regarding the amount of time it took to run each ensemble method on a MacBook Pro. As exemplified in *Figure 3*, this resulted in a nearly linear relationship, which is intuitive since increasing the amount of logistic regression models in each ensemble method would increase the runtime.

3.3 Varying the Percentage of Training Observations Used

Lastly, I decided to vary the percentage of training observations used. As one may recall, from the original dataset, a training set and a testing set were constructed using an 80/20 split. From this 80%, a certain percentage of this set was used to train each logistic regression model within a given ensemble. With each of these models, each were then tested on the aforementioned testing set. As exemplified in *Figure 4*, this is very similar to graph *Figure2* in that it does not show that big of a difference when varying this certain parameter. This was supported when once again trying to fit a linear model with the response variable being accuracy and the percentage being the singular explanatory variable. In this specific case, the p-value was highly insignificant at 0.792.

3.4 Varying All Three Parameters Iteratively

By creating a function within Python that changes each of the three parameters while outputting the accuracy, precision, and recall measures onto a .csv file, a script within R was then created in order to analyze these results to see what conclusions can be made regarding the data. *It should*

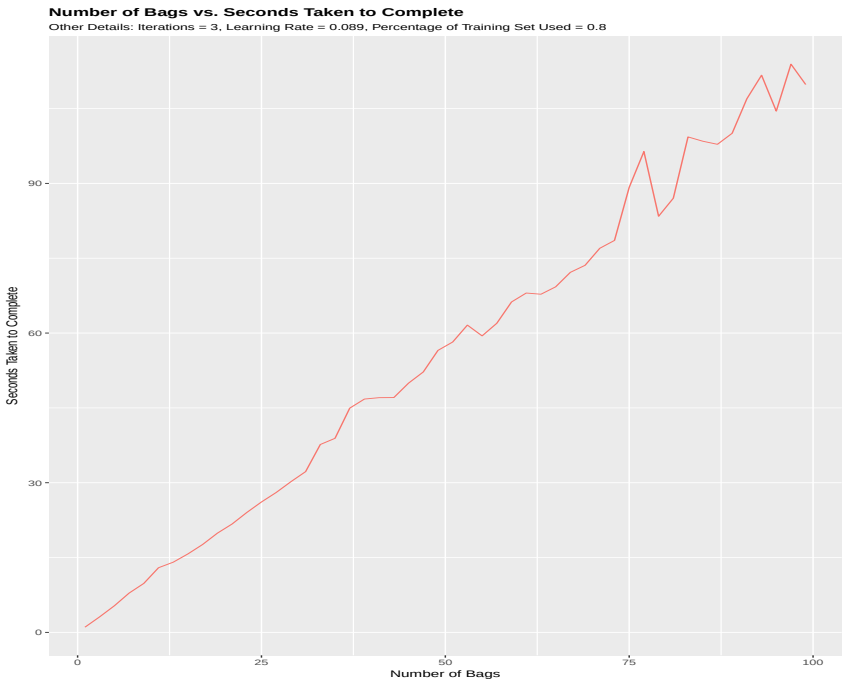


Figure 3: *This graph shows the runtime taken (in seconds) to train the model while increasing the amount of logistic regression models trained within the ensemble method. All other parameters are kept constant through these iterations.*

be noted, however, that the number of logistic regression models varies from 1 to 7 in steps of 2, the percentage of the training set used to train a single logistic regression model varies from 0.1 to 0.9 in steps of 0.1, and lastly, the learning rate varies from 0.001 to 0.991 in steps of 0.01. In an attempt to see which of these parameters can be useful when trying to predict accuracy, precision, and recall, three linear models were created. The three linear models had a response variable of accuracy, recall, and precision, respectively, with each of these response variables trying to be explained by the three aforementioned parameters that have been changing.

For each of the three linear models, the three parameters, percentage of training observations used, learning rate, and the number of logistic regression models in a single ensemble, were all statistically significant, each of which had p-values less than $2e^{-16}$. This came as a surprise to me because on an individual level, none of these variables seemed to be significant but when varying these three all together, this changed.

Additionally, the highest accuracy was found to be 0.7815 and this corresponded to when the number of logistic regression models trained within an ensemble was 7, the percentage of training observations used was 20% and the learning rate was 0.2. Likewise, the highest precision rate was found to be 1.0, which occurred when only a single logistic regression model was trained, the percentage of training observations used was 10% and the learning rate was 0.001. Lastly, the highest recall rate was at 1.0 in which the number of logistic regression models used was 1, the percentage of training observations used was 10% and the learning rate was 0.041.

4 Limitations and Further Work

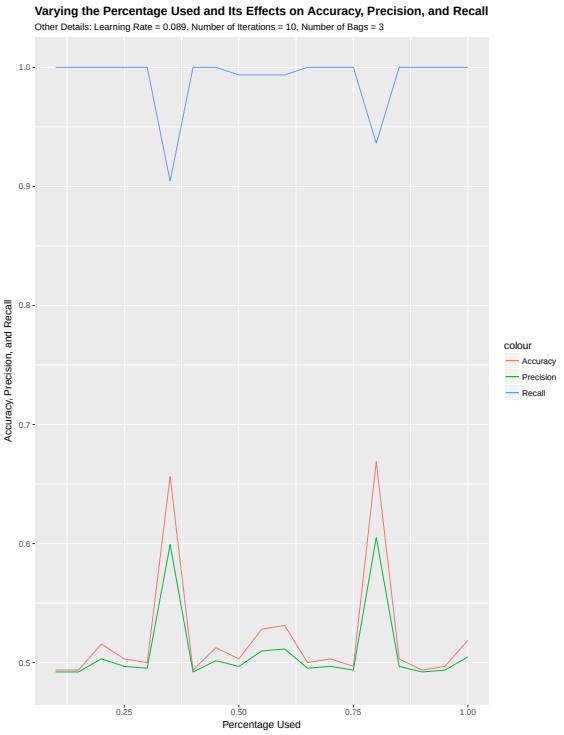
Throughout this project, I thought that the biggest limitation came down to the computational power and speed of my laptop. At first, I pre-processed a much larger dataset in an attempt to use logistic regression to determine the probability of a person making over $50,000 a year in the United States. This dataset consisted of over 48,000 observations and after using one-hot encoding on several of the 14 explanatory variables, the amount of features for each observation expanded to 29. Despite doing all of this preprocessing work, however, my laptop was unable to handle this load and I had to choose another dataset instead.

Additionally, when working on the red wine dataset, gathering accuracy, precision, and recall

measures took longer than expected, specifically when iterating through the four changeable parameters. In an attempt to gather these three measures at the same time, my laptop struggled to do this in a timely manner, which limited my analysis a bit.

In terms of potential future work, I think it would be useful to compare my logistic regression method to an already known logistic regression method, such as Python's Scikit-Learn, to see how it compares on an individual level. Additionally, by using the same ensembling and voting method on both my logistic regression model and the aforementioned Scikit-Learn's logistic regression model, we would better be able to see how similar or different these results are. If given more time, I would have also liked to implement neural networks and/or decision trees within my ensemble method so that the type of classifier is different for every vote.

Figure 4:



5 Conclusion and Discussion

In conclusion, I found that using my own logistic regression method as well as implementing my own ensemble method did not quite produce the results that I was hoping for. With nearly 0 correlation between increasing the number of logistic regression models used in an ensemble to increasing the accuracy of my model, this came as a disappointment simply because the ensemble method, in theory, is supposed to improve accuracy scores quite significantly. However, despite this being the case, I was able to produce good precision and recall scores for the most part during this experiment.

When trying to come up with reasoning behind my results, one possible explanation came forth when I looked at my original dataset before the preprocessing took place. As one may recall, the original dataset had qualities that should have ranged from 0 to 10 but in actuality, nearly all of the quality scores ranged from 4 to 8. Because I preprocessed this dataset in a way such that any score lower than 5 would be considered to be a 0, or considered to be "bad", and any quality score greater than or equal to 5 would be considered to be a 1, or be considered as "good" wine, there was not much distinction between what values of the explanatory variables made wine quality good or not. If these were to be more separated, I believe that my results would have changed quite significantly.Overall, however, this project taught me a lot of about the ins and outs of logistic regression and how an ensemble method worked using a simple voting method.

References

[1] Frederic Brochet *Cheimcal Object Representation in the Field of Consciousness* 2001.

[2] P. Cortez and A. Cerdeira and F. Almeida and T. Matos and J. Reis *Decision Support Systems: Modeling wine preferences by data mining from physicochemical properties* 2009.

[3] UCI Machine Learning Repository *Wine Quality Dataset: https://archive.ics.uci.edu/ml/datasets/Wine+Quality* 2009.

Appendices

The actual R script is also included within my .zip file for easier access.

```r
#### EXPLORATORY WORK ####
dataset <- read.csv("redwine_nolabels.csv")

colnames(dataset) <- c('fixed_acidity', 'volatile_acidity', 'citric_acid',
                       'residual_sugar', 'chlorides', 'free_sulfur_dioxide',
                       'total_sulfur_dioxide', 'density', 'pH', 'sulphates',
                       'alcohol', 'quality')
range(dataset$quality)
pairs(dataset)
dataset$quality <- as.factor(dataset$quality)
ggpairs(dataset, columns = 1:11, ggplot2::aes(colour = quality))

exploratory_image1 <- x[2,1] + labs(title = "Fixed Acidity vs. Citric Acid",
                                    x = "Fixed Acidity", y = "Citric Acid",
                                    legend = "Quality") + guides(color=guide_legend("Quality"))
exploratory_image2 <- x[5,2] + labs(title = "Volatile Acidity vs. Chlorides",
                                    x = "Volatile Acidity",
                                    y = "Chlorides",
                                    legend = "Quality") + guides(color=guide_legend("Quality"))
exploratory_image3 <- x[8,6] + labs(title = "Free Sulfur Dioxide vs. Density",
                                    x = "Free Sulfur Dioxide",
                                    y = "Density",
                                    legend = "Quality") + guides(color=guide_legend("Quality"))
#########################
### RESULTS #####
scale_1_to_10 <- read.csv("final_csv_minMax_1_to_10.csv")
scale_neg1_to_1 <- read.csv("final_csv_minmax_neg1_to_1.csv")
noScale <- read.csv("final_csv_noScale.csv")

range(noScale$accuracy)
range(scale_1_to_10$accuracy)
range(scale_neg1_to_1$accuracy)
# Ranges are pretty much the same

median(noScale$accuracy)
median(scale_1_to_10$accuracy)
median(scale_neg1_to_1$accuracy)
# Median for scale_neg1_to_1 is 0.575, which is 0.03 greater than the other

mean(noScale$accuracy)
mean(scale_1_to_10$accuracy)
mean(scale_neg1_to_1$accuracy)
# Mean is slightly worse
#### Decided to use the scaled from -1 to 1
whole_data <- scale_neg1_to_1
attach(whole_data)
whole_data_lm_acc <- lm(accuracy ~. -precision -recall, data = whole_data)
  # learning_rate has p_value of 0.0227
  # numIterations has p_value of 0.1002
whole_data_lm_pre <- lm(precision ~. -accuracy -recall, data = whole_data)
  # No significant p_values
  # Closest was the learning rate which had a p_value of 0.110
whole_data_lm_rec <- lm(recall ~. -precision -accuracy, data = whole_data)
  # numBags has a p_value of 0.0478
### VARYING LEARNING RATE
varyLR_data <- read.csv("varyLR.csv")
attach(varyLR_data)
lm <- lm(accuracy ~ learning_rate)
plot(x = learning_rate, y = accuracy)

varyLR_data <-ggplot(varyLR) +
  labs(x = "Learning Rate",
       y = "Accuracy, Precision, and Recall",
       title = "Varying the Learning Rates and Its Effects on Accuracy, Precision, and Recall",
       subtitle = "Other Details: Number of Bags = 3, Number of Iterations = 5, Percentage of Training Set Used = 0.8") +
  theme(plot.title = element_text(face="bold")) +
  geom_line(data = varyLR, aes(x = learning_rate, y = accuracy, color = "Accuracy")) +
  geom_line(data = varyLR, aes(x = learning_rate, y = recall, color = "Recall")) +
  geom_line(data = varyLR, aes(x = learning_rate, y = precision, color = "Precision")) +
  scale_y_continuous(limits = c(0.5, 1))
#### VARYING BAGS
varyBags_data <- read.csv("varyBags_report.csv")
varyBags <-ggplot(varyBags_data) +
  labs(x = "Learning Rate",
       y = "Accuracy, Precision, and Recall",
       title = "Varying the Amount of Bags and Its Effects on Accuracy, Precision, and Recall",
       subtitle = "Other Details: Learning Rate = 0.089, Number of Iterations = 3, Percentage of Training Set Used = 0.8") +
  theme(plot.title = element_text(face="bold")) +
  geom_line(data = varyBags_data, aes(x = numBags, y = accuracy, color = "Accuracy")) +
  geom_line(data = varyBags_data, aes(x = numBags, y = recall, color = "Recall")) +
  geom_line(data = varyBags_data, aes(x = numBags, y = precision, color = "Precision")) +
  scale_y_continuous(limits = c(0.5, 1))
varyBags


bags_seconds <- ggplot(varyBags_data, aes(x = varyBags_data$numBags, y = varyBags_data$time_in_seconds)) +
  geom_line(aes(color = "red")) + theme(plot.title = element_text(face="bold")) +
  labs(x = "Number of Bags",
       y = "Seconds Taken to Complete",
       title = "Number of Bags vs. Seconds Taken to Complete",
       subtitle = "Other Details: Iterations = 3, Learning Rate = 0.089, Percentage of Training Set Used = 0.8") +
  theme(legend.position="none")
```

```r
#### VARYING PERC
varyPerc_data <- read.csv("varyPerc.csv")
varyPerc <-ggplot(varyPerc_data) +
  labs(x = "Percentage Used",
       y = "Accuracy, Precision, and Recall",
       title = "Varying the Percentage Used and Its Effects on Accuracy, Precision, and Recall",
       subtitle = "Other Details: Learning Rate = 0.089, Number of Iterations = 10, Number of Bags = 3") +
  theme(plot.title = element_text(face="bold")) +
  geom_line(data = varyPerc_data, aes(x = percentage_of_training_used, y = accuracy, color = "Accuracy")) +
  geom_line(data = varyPerc_data, aes(x = percentage_of_training_used, y = recall, color = "Recall")) +
  geom_line(data = varyPerc_data, aes(x = percentage_of_training_used, y = precision, color = "Precision")) +
  scale_y_continuous(limits = c(0.49, 1))
varyPerc

attach(varyPerc_data)
perc_lm <- lm(accuracy ~ percentage_of_training_used)
summary(perc_lm)
### VARYING PERC, BAGS AND LEARNING RATE
allData <- read.csv("varyBagsPercLR.csv")
attach(allData)
all_acc <- lm(accuracy ~ percentage_of_training_used + numBags + learning_rate)
summary(all_acc)

all_pre <- lm(precision ~ percentage_of_training_used + numBags + learning_rate)
summary(all_pre)

all_rec <- lm(recall ~ percentage_of_training_used + numBags + learning_rate)
summary(all_rec)

allData_acc <- order(allData$accuracy, decreasing = TRUE)[1]
allData_pre <- order(allData$precision, decreasing = TRUE)[1]
allData_rec <- order(allData$recall, decreasing = TRUE)[1]

# Highest accuracy
allData[allData_acc,]
# Highest precision
allData[allData_pre,]
# Highest recall
allData[allData_rec,]
```