Comparing Machine Learning Approaches to Predict Wine Quality

Luke M.

6/23/2020

Introduction

Summary. This project compares several methods for using physiochemical properties of wines to predict their quality. The goal of this project was to compare various machine learning (ML) and non-ML modeling approaches and determine which single approach or multi-model ensemble approach could yield the best predictions with the lowest root mean squared error (RMSE). Four ML methods were compared: stepwise regression, stepwise regression with k-means cluster centroids, k-nearest neighbors, and random forests. The random forest model performed the best, with a final *RMSE of 0.60105*. The RMSEs for all of the ML models were better than the RMSEs generated by the baseline non-ML models examine here. Overall, this project demonstrates that (1) even something as seemingly subjective as wine quality can be grounded in and related to measurable data and that (2) ML methods can help identify these complex relationships.

Motivation. Wines have been an important part of human culture for millennia. Many think of wine preferences and quality to be a subjective experience, but others (e.g., see Cortez et al. 2009) have used measurable physical and chemical (together, physiochemical) properties to predict wine quality with some accuracy, merging the subjective and the objective experience. The goal of this project was to examine if ML methods can be applied to analyze physiochemical wine data to accurately predict wine quality ratings.

Data and Exploratory Analysis

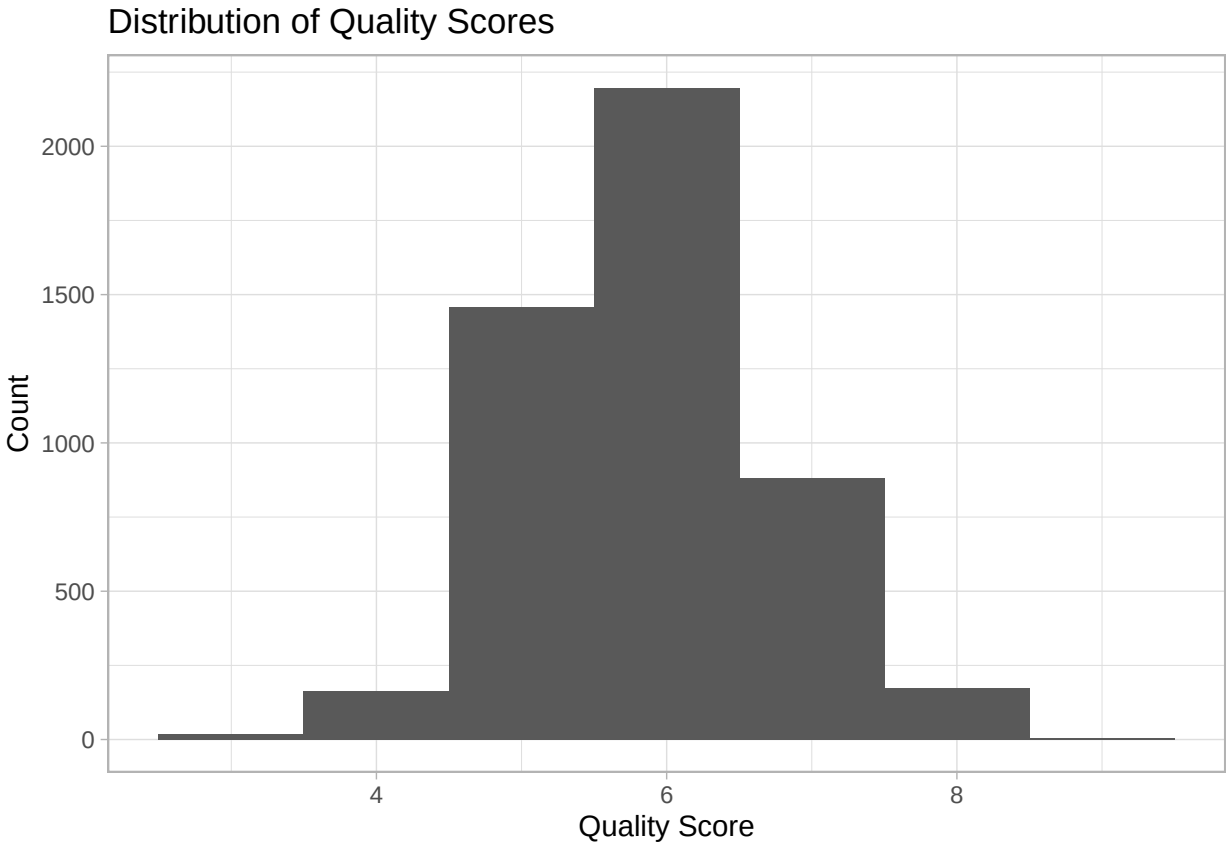
*Descriptive Statistics.** The data for this project originate from Cortez et al (2009)'s Wine Quality Data Set for white wines from the *vinho verde* region of Portugal. This data set includes 4898 observations, with one integer variable for wine quality rating which ranges from the worst rating of 3 to the best rating of 9. The average rating is 5.8779094 with a standard deviation of 0.8856386. In addition, the dataset includes 11 the following continuous variables of the physiochemical properties for each observation: fixed.acidity, volatile.acidity, citric.acid, residual.sugar, chlorides, free.sulfur.dioxide, total.sulfur.dioxide, density, pH, sulphates, alcohol. The table below summarizes the values of these 11 variables, which will be used as the predictors.

fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide
Min. : 3.800	Min. :0.0800	Min. :0.0000	Min. : 0.600	Min. :0.00900	Min. : 2.00
1st Qu.: 6.300	1st Qu.:0.2100	1st Qu.:0.2700	1st Qu.: 1.700	1st Qu.:0.03600	1st Qu.: 23.00
Median : 6.800	Median :0.2600	Median :0.3200	Median : 5.200	Median :0.04300	Median : 34.00
Mean : 6.855	Mean :0.2782	Mean :0.3342	Mean : 6.391	Mean :0.04577	Mean : 35.31
3rd Qu.: 7.300	3rd Qu.:0.3200	3rd Qu.:0.3900	3rd Qu.: 9.900	3rd Qu.:0.05000	3rd Qu.: 46.00
Max. :14.200	Max. :1.1000	Max. :1.6600	Max. :65.800	Max. :0.34600	Max. :289.00

total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
Min. : 9.0	Min. :0.9871	Min. :2.720	Min. :0.2200	Min. : 8.00	Min. :3.000
1st Qu.:108.0	1st Qu.:0.9917	1st Qu.:3.090	1st Qu.:0.4100	1st Qu.: 9.50	1st Qu.:5.000

total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
Median :134.0	Median :0.9937	Median :3.180	Median :0.4700	Median :10.40	Median :6.000
Mean :138.4	Mean :0.9940	Mean :3.188	Mean :0.4898	Mean :10.51	Mean :5.878
3rd Qu.:167.0	3rd Qu.:0.9961	3rd Qu.:3.280	3rd Qu.:0.5500	3rd Qu.:11.40	3rd Qu.:6.000
Max. :440.0	Max. :1.0390	Max. :3.820	Max. :1.0800	Max. :14.20	Max. :9.000

The mode of the wine quality ratings was 6, and there are very few observations of very fine wines and very bad wines. As seen later in the analysis, the very small number of fine wines made it difficult to identify features that accurately predict these rarities.



Correlations Between Ratings and Physiochemical Properties. A useful method for preliminary exploratory analysis to estimate the covariance of the outcome of interest (in this case, the wine quality rating) and possible predictor variables. Covariance alone is insufficient to estimate or demonstrate a causal link between independent and dependent variables, but exploring covariance correlations is useful for exploring possible causal links that warrant further analysis. It is also useful for identifying possible instances of homogeneity among possible predictor variables that might be redundant measurements of the same underlying phenomenon.

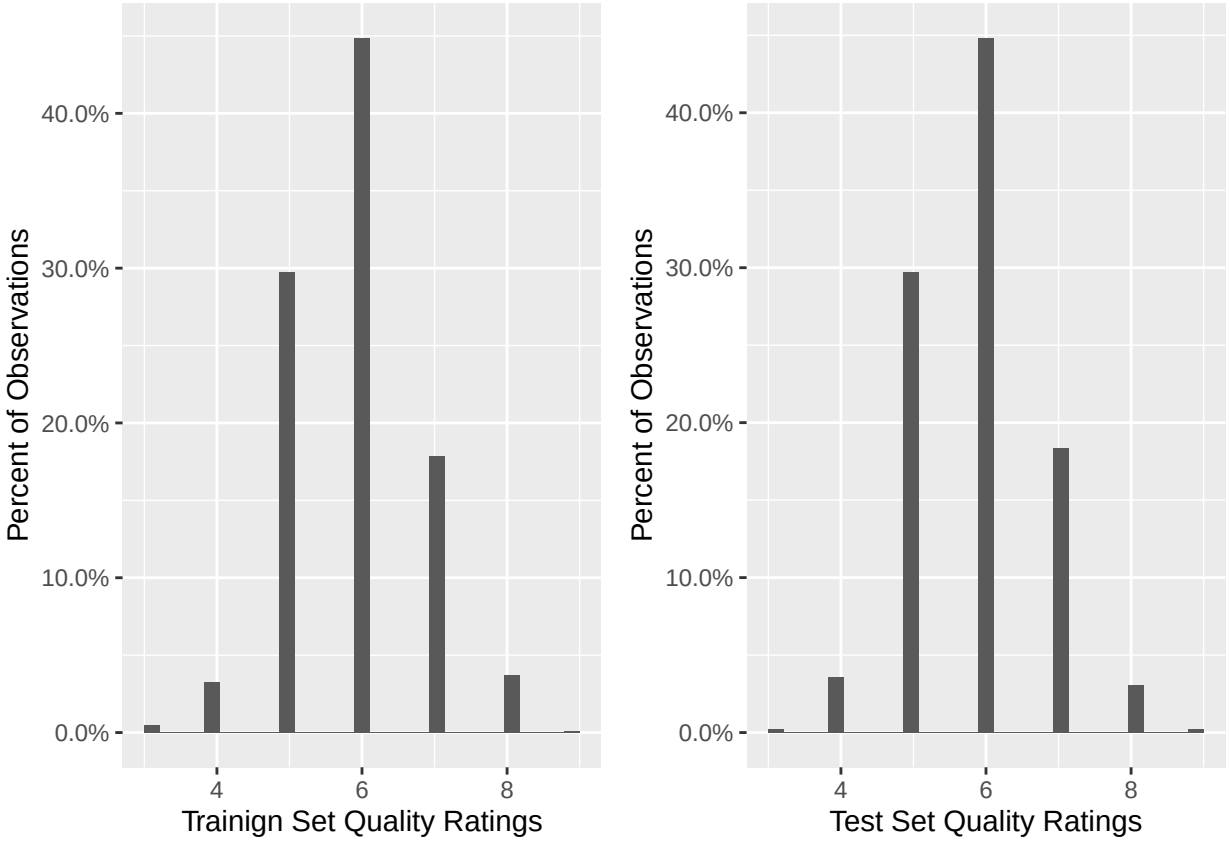
A common method for examining covariance between possible variables of interest is Pearson's pairwise correlation coeffcient estimates. Pearson's pairwise correlation coefficient is a common, easily interpreted method for estimating the extent to which two variables co-vary. The Pearson correlation coefficient can be thought of as the percent of two variables' variance that is correlated (i.e., the percent of variance that co-varies). The table below lists the correlation coefficients, which indicate that alcohol, chlorides, and density have the highest covariance with quality.

	correlation_coeff
alcohol	0.4355747
pH	0.0994272
sulphates	0.0536779
free.sulfur.dioxide	0.0081581
citric.acid	-0.0092091
residual.sugar	-0.0975768
fixed.acidity	-0.1136628
total.sulfur.dioxide	-0.1747372
volatile.acidity	-0.1947230
chlorides	-0.2099344
density	-0.3071233

Methods

Data Partitioning and Setup

The dataset was split into a training set with 80% of observations and a test set with 20% of observations. This 80/20 ration of training/test observations was selected because the analysis methods will use cross-validation within the training set, so increasing the size of the training set to 80% of the observations allowed a greater proportion of the total observations to be sampled in the cross-validation partitions generated as the model was being trained. The histograms below of the quality rating distributions for the training and test set data indicate both datasets have similar distributions, suggesting the test set likely has an accurate approximation of the prior probability distribution of quality ratings in the training set.



The following code was used to create tables to store and compare the RMSEs for each modeling method and to store the predictions from each method.

```
#create empty tables where rmse results for each model will be stored
train_rmse_table<-data.frame(model=character(0),rmse=double(0))
test_rmse_table<-data.frame(model=character(0),rmse=double(0))

#create dfs to store the y-hats from each model will be stored
train_results<-data.frame(rating=train$quality)
test_results<-data.frame(rating=test$quality)

#rename the quality column "rating" for easy reference in subsequent code
names(train)[which(names(train)=="quality")] <- "rating"
names(test)[which(names(train)=="quality")] <- "rating"
names(wine_data)[which(names(wine_data)=="quality")] <- "rating"
```

Baseline Models

Before applying various ML techniques, the performance of several simple models was estimated to establish a baseline of RMSEs to use to evaluate whether applying ML techniques meaningfully improved rating predictions.

Random Guesses. The first baseline, non-ML model examined was a model that randomly guesses wine ratings. This approach assumed all possible rating had the same probability of occurring. This random guessing method was expected to perform the worst and have the highest RMSE because this method did not base its predictions on any information gleaned from the training data.

```
#create list of possible ratings
poss_ratings<-seq(3,9,1)

#use random sample of possible ratings to generate predicted ratings and RMSEs
train_results[["y_hat_random_guesses"]]<-
  sample(poss_ratings, nrow(train),replace = T)

train_rmse_table[nrow(train_rmse_table)+1,]<-
  c("random_guesses",
    RMSE(train_results$y_hat_random_guesses,
         train_results$rating))

test_results[["y_hat_random_guesses"]]<-
  sample(poss_ratings, nrow(test),replace = T)

test_rmse_table[nrow(test_rmse_table)+1,]<-
  c("random_guesses",
    RMSE(test_results$y_hat_random_guesses,
         test_results$rating))
```

Weighted Guesses. This method also randomly assigns predicted wine quality ratings, but this method allowed each particular rating level to have different probabilities of being predicted for the test set. The training data was used to estimated the prior probability of each rating level. Then, the test set predictions were generated based on taking samples of ratings weighted to occur in proportion to how often they occurred in the training set.

```
#estimate the proportion of observations in the training data that have each rating level
probs<-train %>% group_by(rating) %>%
  summarize(prob_per_rating=as.double(n()/nrow(train)))
```

```
#assign predictions at each rating level according their priors and find RMSEs
train_results[["y_hat_weighted_guesses"]]<-
  sample(poss_ratings, nrow(train), replace = T, prob=c(probs$prob_per_rating))

train_rmse_table[nrow(train_rmse_table)+1,]<-
  c("weighted_guesses",
    RMSE(train_results$y_hat_weighted_guesses,
         train_results$rating))

test_results[["y_hat_weighted_guesses"]]<-
  sample(poss_ratings, nrow(test), replace = T, prob=c(probs$prob_per_rating))

test_rmse_table[nrow(test_rmse_table)+1,]<-
  c("weighted_guesses",
    RMSE(test_results$y_hat_weighted_guesses,
         test_results$rating))
```

Global Average. The next baseline model used the overall average rating observed in the training data to predict all ratings in the test set. The average rating of all observations in the training set was 5.8782853, which in this baseline method was the predicted rating for all observations in the test set.

```
#make predictions solely from overall average rating in the training dataset
train_results[["y_hat_global_mean"]]<-mean(train$rating)
train_rmse_table<-rbind(train_rmse_table,c("global_mean",
                                           RMSE(train_results$y_hat_global_mean,
                                                train_results$rating)))

test_results[["y_hat_global_mean"]]<-mean(train$rating)
test_rmse_table<-rbind(test_rmse_table,c("global_mean",
                                         RMSE(test_results$y_hat_global_mean,
                                              test_results$rating)))
```

Machine Learning Approaches

Stepwise Regression. Stepwise regression is a ML technique that takes a list of possible parameters and builds a linear regression model that contains the combination of parameters that achieves the optimal model fit (i.e., achieves the lowest RMSE). This allows the model to determine which predictors have a statistically significant effect when modeled together.

```
#estimate the best stepwise regression model based on 25 bootstraps
best_stepwise_fit<-train(rating ~ .,
                         method = "leapSeq",
                         data = wine_data[train_index,],
                         tuneGrid=data.frame(nvmax=seq(1,ncol(train)-1,1)))
```

K-means Clustering and Stepwise Regression. A popular clustering method is known as k-means clustering, which is an unsupervised learning algorithm that sorts observations into clusters of observations with similar attributes. Clustering can be used to identify the average expected rating for members of a cluster. This average expected outcome is often referred to as the centroid of the group. In this approach, the optimum number of clusters and centroids were estimated, and then the stepwise regression method outlined in the last section was applied to estimate the effect of centroids on ratings, except this re-run of the stepwise algorithm included a predictor with the centroids of the clusters assigned to each observation.

In the code below, k specifies the number of clusters created. The optimal k is somewhat subjective, but it is generally considered to occur in the "elbow" in the graph of k vs *SSD* (the sum of squared differences among

the observations in each cluster). This elbow occurs indicates the general value of k at which increasing the number of clusters would not significantly improve cohesion within clusters. In this context, cohesion is defined as the sum of within-cluster differences, which is a way of measuring overall similarity of observations in any given cluster. The theory is that clusters with highly similar observations generally will yield better cluster-based predictions.

```
set.seed(1)

#create list of possible values of k
k=seq(2,50,1)

#calculate the within-cluster sum of squares for each k
wss<-sapply(k,function(k_){
  kmeans_fit<-kmeans(wine_data[1:ncol(wine_data)-1],
                     k_, iter.max = 20, nstart=10)

  #return the total within-cluster sum of squares
  kmeans_fit$tot.withinss
})

#this plot shows us that the "elbow" occurs somewhere between 10-15, so
ggplot(NULL, aes(x=k,y=wss)) +
  geom_point() +
  theme_grey() +
  labs(x="Cluster Size k",
       y="With-in Cluster Sum of Squares",
       title="Cluster Size K vs. Within-Cluster Sum of Squares")
```


```
#identify the best k that minimizes SSD for observations within clusters
best_k<-30

#using the optimal k to cluster the wines
clusters<-kmeans(wine_data[1:ncol(wine_data)-1],
                 best_k, iter.max = 20, nstart=10)$cluster

#add the cluster numbers for each observation to their respective rows
wine_clusters <- cbind(wine_data, clusters)

#add columns for the wine's average rating for each cluster
wine_clusters <- wine_clusters %>% group_by(clusters) %>%
  mutate(cluster_rating_avg=mean(rating))

#remove the cluster number because we don't need it anymore
wine_clusters <- select(wine_clusters,-c(clusters))
```

The best model generated by the stepwise algorithm included 9 out of the 11 possible predictors.

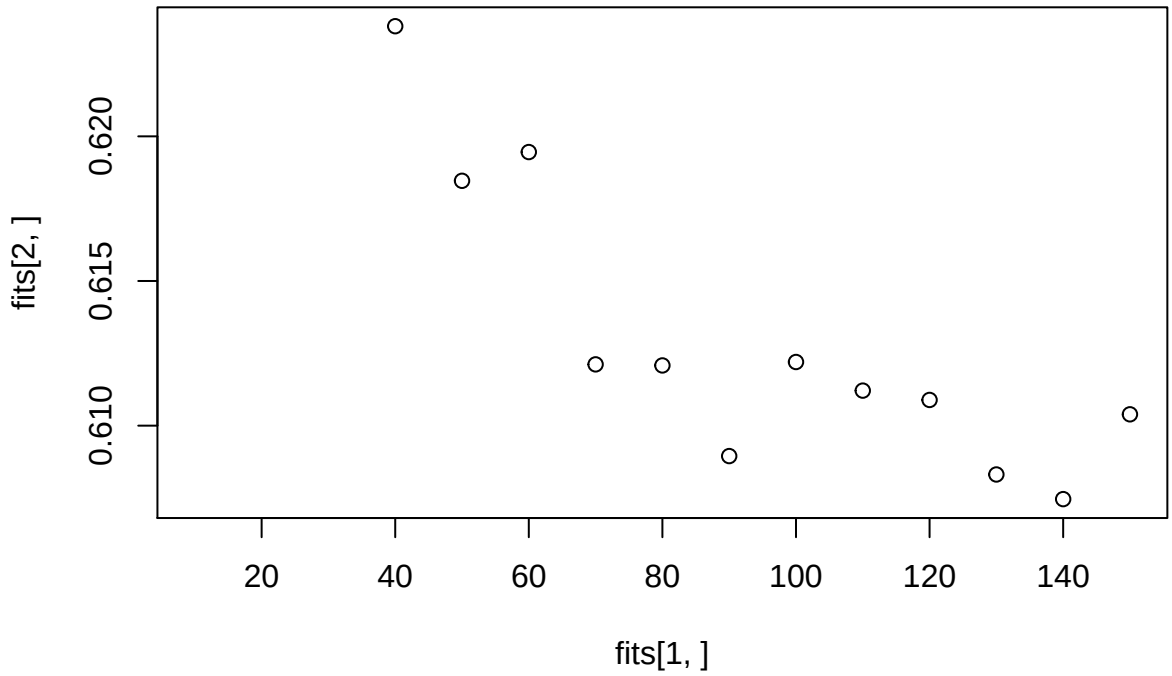
K-Nearest Neighbor. K-nearest neighbor clustering (knn) is a non-linear method that makes prediction an observation's outcome (in this case, wine quality) based on known-outcomes for k number of other observations that have similar attributes as the observation for which we are trying to make a prediction. In this manner, this method predicts an outcome based on its "neighbors"–other observations that are similar to it.

Random Forest. The Random Forest method is a ML method that creates and tests a range of decision trees to develop a classification algorithm to sort observations into outcomes based on their values. Random Forest methods excel with classification problems like the task at hand, which is seeking to sort wines into

rating levels based on attributes.

```
trees<-seq(10,150,10)

#calculate fits for each number of trees
fits<-sapply(trees, function(tree){
  fit<-randomForest(as.matrix(train[1:ncol(train)-1]),
                    as.matrix(train$rating), ntree=tree)
  list(tree,RMSE(fit$predicted,train_results$rating),fit,fit$pred)
})
```




```
## [1] "The random forest with the lowest RMSE has 140 trees with RMSE of 0.607460318772068"
```

Ensemble Model. An ensemble model was estimated by testing whether averaging the predictions of one or more of the aforementioned methods would reduce the overall RMSE. A key assumption underpinning ensemble modeling methods is that any given method will yield predictions with errors that *are normally distributed* around the true value of the predicted variable. Therefore, averaging the predictions of multiple models should in theory reduce overall RMSE by shifting the predictions toward the true values of the predicted parameter.

```
#rank the modeling methods according to their RMSEs
test_rmse_table[["rank"]]<-as.integer(rank(test_rmse_table$rmse))

#create a list of possible number of models
poss_num_models <- seq(2,nrow(test_rmse_table),1)

#estimate the RMSEs for ensemble models with various numbers of models included
rmses<- sapply(poss_num_models,function(poss_models){
```

```
  models_to_keep <- test_rmse_table %>% filter(rank<poss_models)
  results_to_keep <- lapply(list(models_to_keep$model), function(k){
    paste("y_hat",k,sep="_")
  })

  #average the models kept and calculate RMSE
  y_hat_meta_model_test <- rowMeans(select(test_results,
                                           c(unlist(results_to_keep))))
  RMSE(y_hat_meta_model_test, test_results$rating)
})
```

The one or more models listed in the table below were determined to be either the single model, when taken alone, or the *combination* of models that achieved the lowest RMSE.

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

```
##   model              rmse rank
## 1    rf 0.601051267643038    1
```

Results

The table below lists the final RMSEs for each individual modeling method, as well as the best ensemble method. The random forest model performed the best, with an *RMSE of 0.60105*. As show in the ensemble model analysis in the previous section, averaging the predictions of the random forest model with predictions from one or more of the other models did not beat the RMSE of the random forest model alone.

model	rmse	rank
random_guesses	2.16386896716229	7
weighted_guesses	1.30802504560054	6
global_mean	0.876458892867988	5
stepwise_regression	0.744699648031199	3
stepwise_regression_clusters	0.726904538704833	2
knn	0.808721949047543	4
rf	0.601051267643038	1
meta_model	0.601051267643038	meta_model

Conclusion

This project achieved its overall goal by designing a ML model that predicted wine ratings, and the final RMSE of the best method was *RMSE=0.60105*. After comparing several non-ML and ML methods, it was determined that the best method for predicting ratings was the random forest method. This model achieved a substantially lower RMSE than the next-best model, which was the stepwise regression that included the k-means cluster averages (RMSE=0.72690). This demonstrated that using ML methods can help us to achieve better predictions that those generated by non-ML approaches. It also provides suggestive evidence that even subjective experiences like wine ratings can be grounded in predictable, measurable data, in this case, the physiochemical properties of Portugese white wines.

References

Paulo Cortez, University of Minho, Guimarães, Portugal, http://www3.dsi.uminho.pt/pcortez / https://archive.ics.uci.edu/ml/datasets/Wine+Quality A. Cerdeira, F. Almeida, T. Matos and J. Reis, Viticulture Commission of the Vinho Verde Region(CVRVV), Porto, Portugal