Madeline Hayes
ECE532 Final Project Proposal
12 December 2020

Investigating Classification of Wine Color and Quality from Physiochemical Properties

Background: Advanced characterization of wine can be achieved by combining traditional sensory assessments with physiochemical tests. These data are useful for assuring quality and safety, targeting growing consumer markets, and pricing. Though assessment of wine by sensory classification is a time-honored tradition (and an important cultural marker in many countries), the connection between taste and chemical properties is still not well understood. Accumulating large datasets of both the quality assessment (human sensory judgement) and physiochemical properties (e.g. alcohol content, density, sugar content) provides an opportunity to construct predictive models that will allow producers to rapidly characterize their product and target distribution to appropriate markets.

Dataset of Choice: Dataset: Wine Quality Dataset (https://archive.ics.uci.edu/ml/datasets/wine+quality)

This dataset is composed of 6497 samples with 13 attributes. Each sample is a variant of the Protugese Vinho Verde, with 1599 reds and 4898 whites. These data have 12 chemical attributes (fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol content) and a quality score (human taste rating from 1 to 10). This wide variety of data makes this dataset ideal for a variety of machine learning applications. This dataset was collected with the goal of quality prediction, however, due to the suitable sample size of both reds and whites, this dataset is also well suited for classification tasks.

Github Repository: https://github.com/mmhayes/ECE532_Final

Proposed Research Questions: Can we determine whether a wine is red or white from its chemical properties? Can we predict a drinker's quality rating based on a wine's chemical properties?

Initial Dataset Characterization

Code: exploratory_stats.ipynb
Figures not included here: exploratory_stats.pdf

Each attribute of the samples was visually assessed by histogram and violin plot to visualize the shape and distribution of each attribute (see **exploratory_stats.pdf**). Assessment of the data in this way can help determine if data values need to be scaled. No statistical analysis was completed beyond visual assessment.

Approach #1: K-means clustering

Code: kmeans.ipynb

From Original Proposal: In this approach I am to classify a wine's color based on its chemical properties. This is an unsupervised problem where the algorithm will use the distances between points to determine cluster centers. In this problem I propose to combine the red and white datasets and compare the distributions of red and whites over a k=2 clustering problem. I hypothesize that excluding some features of the data will lead to better clustering, as the distribution of certain properties (e.g. alcohol content) is less likely to vary between reds and whites.

Methods and Results: To determine the best approach for kmeans, I first ran a k=2 clustering problem over the whole dataset without any scaling or transformation. This was to verify that scaling the data would be appropriate for this problem. With no scaling, the error rate (misclassification) was moderately high, with the error rate for reds and whites being 5.2% and 26.7% respectively, and a total error rate of 21.4%. This data took 18 iterations to converge and had a sum of squares distance of 8594875.94 (**Table 1**). After using a minmax scaler to scale each feature between 0 and 1, the error rate increased dramatically, with the error rate for reds and whites being 62.6% and 42.1% respectively, and a total error rate of 47.2%. Interestingly, with scaling the algorithm took only 8 iterations to converge and had a sum of squares distance of 699.5 (**Table 1**). I hypothesized that some features which have very similar distributions between red and white samples may be contributing noise, which I visually assessed via histogram and violin plot (**see exploratory_stats.pdf**). From this visual assessment, I identified two features, alcohol content and quality, which had largely overlapping distributions (**Figure 1**). To assess whether these features were predictive or noisy, I removed these features from the analysis and repeated (both scaled and unscaled data) kmeans with k=2 clusters. The unscaled data error rate was identical to the full dataset, but the scaled data with trimmed dataset gave the best performance (**Table 1**).

Iteration ID	Features Used	Scaling?	Red Errors	White Errors	Total Errors	Iterations/SSD
1	All	None	5.2%	26.7%	21.4%	18/8594876
2	All	0 to 1	62.6%	42.1%	47.2%	8/699.5
3	11/13 (removed alcohol content and quality)	None	5.2%	26.7%	21.44%	18/8581133.7
4	**11/13 (removed alcohol content and quality)**	**0 to 1**	**2.3%**	**1.9%**	**2.0%**	**5/410.3**

Table 1: Summary of kmeans clustering results. Iterations are numbered 1-4, as in jupyter notebooks, as data was transformed and some features removed. Scaling was performed using minmax scaler, which forces values between 0 and 1. Error rates were calculated by dividing number of misclassifications by totals. Iterations and sum of squares distances were returns from algorithm.

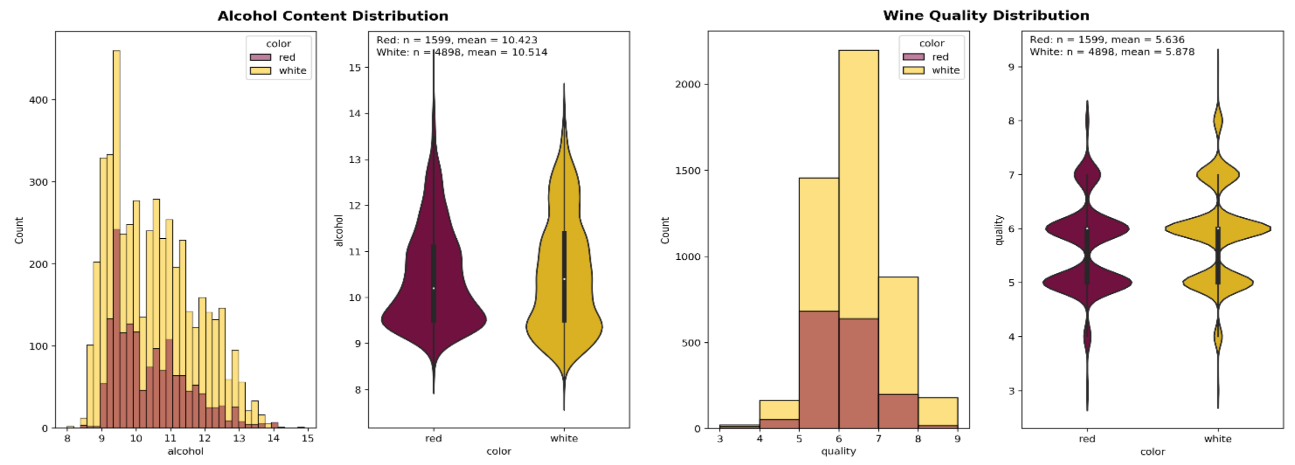


Figure 1: Distributions of alcohol content (left) and quality rating (right). Means of each value are listed in the violin plot. Because these distributions were largely similar between red and white (contrast with other features as plotted in exploratory_stats.pdf), these features were excluded during iterations attempting to reduce noise.

Kmeans clustering is a useful tool for determining the similarity of datapoints with no *a priori* knowledge of how those data are related. However, kmeans can still be used in instances where the ground truth (labels) is known for each sample, and there are evaluation methods for the performance of such a model apart from the error rate. For this particular problem, I chose homogeneity, completeness, and v-measures to assess the performance of the model. Homogeneity scores whether a cluster contains only members of a single class, completeness scores whether all members of a given class are assigned to the same cluster. The v-measure score is a function of homogeneity and completeness (**equation 1**):

$$v = \frac{(1+\beta) \times \text{homogeneity} \times \text{completeness}}{(\beta \times \text{homogeneity} + \text{completeness})}$$

Where beta allows for differential weighting of either homogeneity or completeness, whichever is more 'important' for the clustering problem. Scores range from 0 (worst) to 1 (best). In this case, I did not specify beta. Results of the v-score analysis are pictured in **Table 2**. As highlighted, iteration 4 (selected data, scaled) which had the lowest error rate also has the highest homogeneity, completeness, and V-score.

	Iteration	Homogeneity	Completeness	V-Score
0	1	0.352393	0.287212	0.316481
1	2	0.001556	0.001283	0.001406
2	3	0.352393	0.287212	0.316481
3	4	0.842913	0.829359	0.836081

Table 2: Summary of V-scores results. Iterations are numbered 1-4, as in jupyter notebook and Table 1.

Potential pitfalls and future directions: As mentioned, Kmeans is useful when there is not prior knowledge about the ground truth of the data. However, in simple classification tasks, it can be appropriate. One downside is that to reduce noise, one must have some intuition about which features may be most relevant, which is not always possible. The V-score measure of performance is appropriate for a dataset of this size with <8 clusters, however only two clusters could be driving down the V-score artificially. For future approaches to this problem, I would more rigorously analyze features contributing noise (perhaps by PCA) and implement other performance evaluations for comparison.

Approach #2: K-nearest neighbor (KNN)

Code: knearestneighbors.ipynb

From proposal: In this approach I aim to predict the approximate rating of a wine from its chemical properties by choosing target ratings and classifying each wine based on its proximity to that rating. KNN is a supervised learning problem where the labels are provided. KNN can be optimized by k-folds cross validation. I hypothesize that this method may be more accurate for predicting labels than regression.

Methods and Results: For my initial approach, I chose to use KNeighborsClassifier from sklearn since the quality ratings are discrete values in this dataset. This operates as a discrete classifier, which may not intuitively seem appropriate for this dataset; referring back to **Figure 1** we see that the quality ratings (right) are not truly continuous and can be treated as classes. Unlike Kmeans, KNN is a method that is amenable to cross-validation. For this approach, I can split the data into train and test sets and optimize the number of neighbors which get a vote to classify a new datapoint. I split the data into 90% train / 10% test and ran the model 30 times, each time evaluating accuracy at each value of k. From this approach I determined that the highest accuracy over 30 validations 65%, with k=1 (**Figure 2**). Note: I also used a different split size for cross validation (80/20 and 70/30) with results in agreement, k=1 remained highest average accuracy (see **Supplementary Figure 1**).

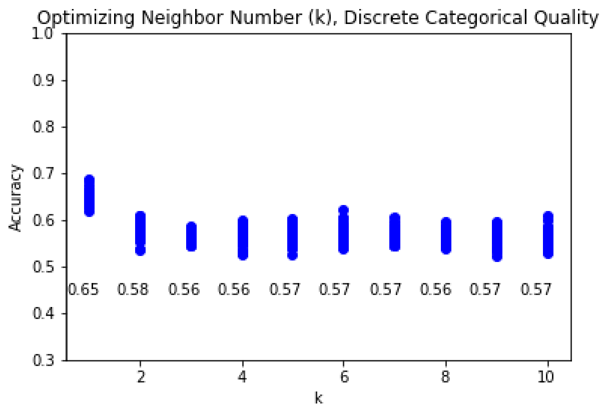


Figure 2: Cross validation to optimize k. For every value of k between 1 and 10, the dataset was split 90/10 for training and test sets. The model was trained and then asked to predict the test set 30 times for each value of k. Average accuracy of the predicted label is plotted as a percentage of correctly labelled points. K with the highest average accuracy was chosen as optimal k.

After optimizing k, I split the data into a final training and evaluation set to test the performance of the model. The final split was 80% randomly sampled training data, 20% evaluation data. I trained the KNClassifier model with k=1 on the training set and then predicted the quality of the evaluation data with 65% accuracy.

As in the previous kmeans approach, I wanted to use intuition to remove noisy features. Since the kmeans approach demonstrated that alcohol content was noise for a color prediction, I wanted to see if this was also true for a quality prediction. I performed KNN again without alcohol, and also once leaving out color. Cross validation confirmed that k=1 is the optimum neighbor number regardless of feature number (**Supplementary Figure 2**). I repeated final evaluation with an 80/20 split; leaving out alcohol, accuracy was 64%, and leaving out alcohol and color, slightly higher at 67%.

Although this accuracy seems low, the mode of classification only takes into account perfect prediction. For the purposes of this dataset, it may be beneficial to consider the quality ratings as continuous, and evaluate accuracy as distance from true rating. For this approach we can use KNearestRegressor. For this process I used k-folds optimization as above to optimize k, and model evaluation on a houldout set also as above. The optimum k took several iterations and split sizes to optimize (data not shown, see jupyter notebooks for extensive figures), but settled on k=10 (**Figure 3**). This k also seems to have the tightest distribution of distances.

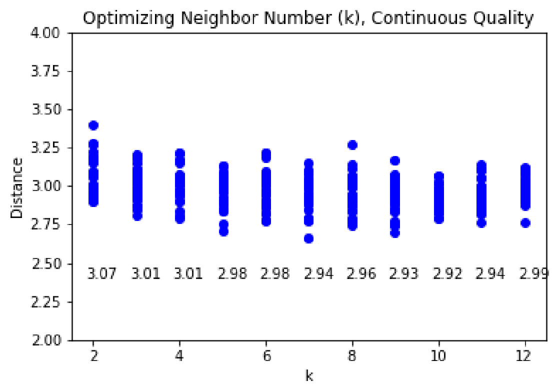


Figure 3: Cross validation to optimize k. For every value of k between 1 and 10, the dataset was split 90/10 for training and test sets. The model was trained and then asked to predict the test set 30 times for each value of k. Average 2-norm (distance) of the predicted label from the true label is plotted. K with the lowest average distance was chosen as optimal k.

Final holdout evaluation reveals a model that is not terribly correlated, the actual linear regression r^2 value is only 0.599 (**Figure 4**).

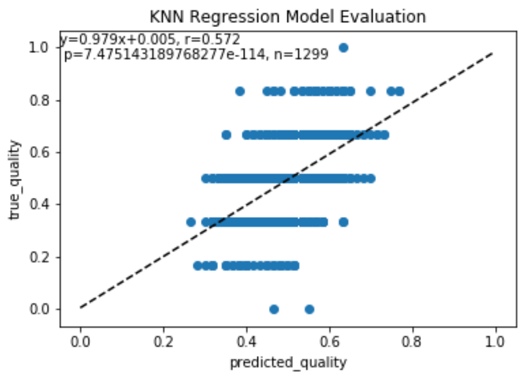


Figure 4: Regression of final evaluation for KNN with continuous quality values. After optimization, evaluation of KNN was performed on a holdout set of data. Here, predicted_quality is the scaled quality value predicted from the model and true_quality is the actual scaled label associated with that point. We can see that even though the model treats the variables as continuous, a discrete grouping appears. The model only has a weak fit with r^2=0.599.

For the manual feature selection I assumed optimum k=10 and performed holdout evaluation. This model was less predictive than the full dataset (**Supplementary Figure 3**).

Potential pitfalls and future directions: This approach is probably not best suited to predicting quality. KNN is by nature a clustering algorithm and would probably be more appropriately compared to kmeans for color prediction of this dataset. It would appear that this is true regardless of whether or not the data being predicted is discrete or continuous. As with kmeans, a more rigorous evaluation of noisy features would probably decrease error/distance.

Approach #3: Regression

Code: linearregression.ipynb

From proposal: In this approach I will aim to predict a drinker's rating of a wine from its chemical composition. This approach utilizes a weighted sum of each feature (physiochemical property) to approximate the label (rating) for a sample. This is a supervised learning problem and can be optimized using cross validation. I hypothesize that the models for red and white wines will be different in their most significantly contributing features.

Methods and Results: For this approach, I used a simple linear regression with LASSO regularization, because I wanted an automated variable selection (a large enough lambda will force some features to 0). Optimal lambda was chosen by cross validation (as optimal k was previously chosen in KNN approach). Unexpectedly, even though there are many more samples than features, a perfect fit can be achieved with a relatively high lambda (**Figure 5**). I therefore chose a lambda which reflected some error to avoid overfitting (**Supplementary Figure 4** shows the goodness of fit for this model, though trained on the whole dataset and then predicting the whole dataset).

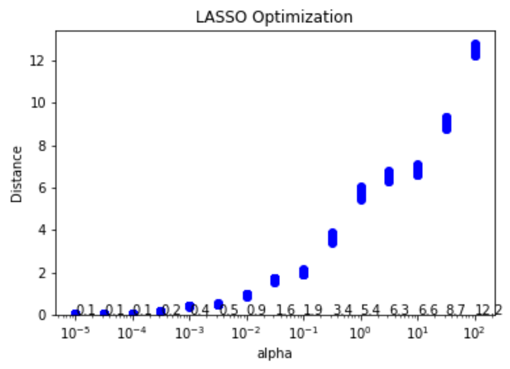
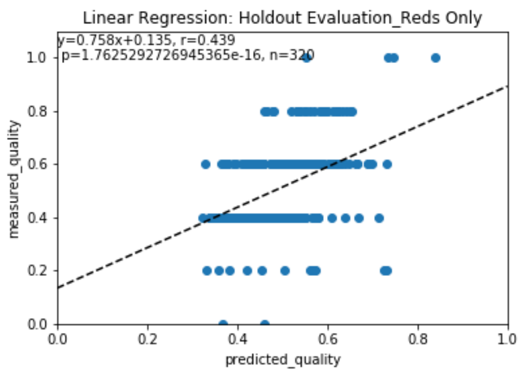


Figure 5: Optimization of lambda for linear regression. Cross validation using 30 splits of 90/10, lambda was optimized over a range of values. I would expect a parabolic shape from this process but it appears that small lambdas can converge to a perfect fit for this dataset.

In order to evaluate the performance of the linear regression, I performed holdout evaluation as in KNN. This was a poorly predictive model (**Supplementary Figure 5**). I hypothesized that each color of wine may be better predicted separately, so I repeated the above process with reds and whites separately (see **Supplementary Figure 6** for lambda optimization). These were equally poorly predictive **(Figure 6)**.



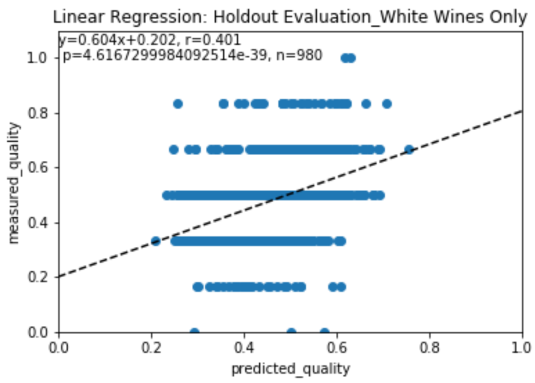
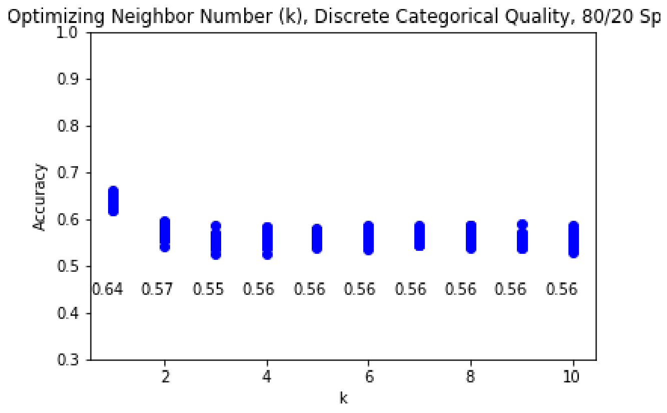


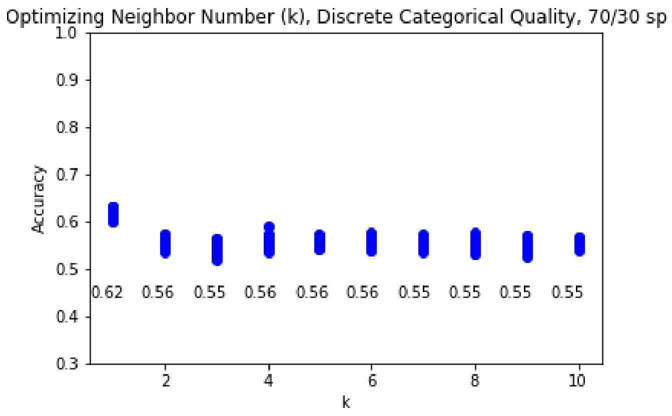
Figure 6: Red and White Separated Holdout Evaluations (lambda=0.01). LASSO optimization was completed for reds and whites separately, though both final models used lambda=0.01. Holdout split was 80/20, as with KNN.

Potential pitfalls and future directions: This approach was by far the weakest for this project on this dataset. It seems possible that feature selection needs to be applied very rigorously for these data, if prediction is possible. For future directions, I would first use a Leave-One-Out approach for feature selection as compared to PCA to see which features are contributing to the model, and rather than LASSO regularization I would opt to try ridge regression. I started the project with this section, so the coding for these models is definitely the weakest. If I had more time I would definitely go back and incorporate some best practices that I learned during the Kmeans and KNN sections.

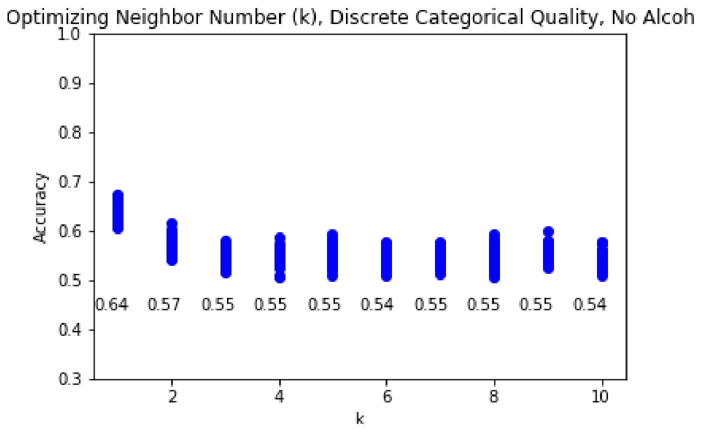
Discussion: While the ability to predict a wine's quality from easily testable chemical properties would undoubtedly allow producers to better target their markets and price their products, the relationship between taste and physiochemical properties is poorly understood at best. In addition, even though sommelier rating standards are rigorous and prestigious, quality is ultimately a subjective judgement call. This dataset is an excellent opportunity to learn general modelling approaches and machine learning techniques, however, it seems that a more advanced approach is probably necessary to make accurate predictions from the properties in this dataset. It may be that more features and more rigorous feature selection than those employed here could increase the accuracy of a quality predictive model. Despite the shortcomings in quality prediction, the ability to predict a wine's color from its properties is very straightforward. I would be very interested in performing the same kind of analysis on other colors as well, including blends and roses.

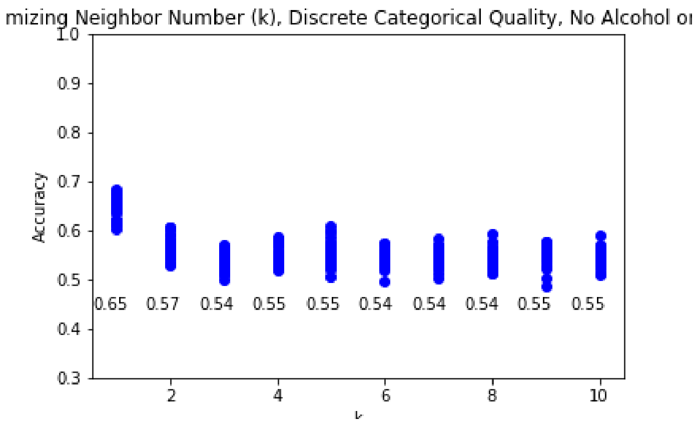
Supplementary Figures



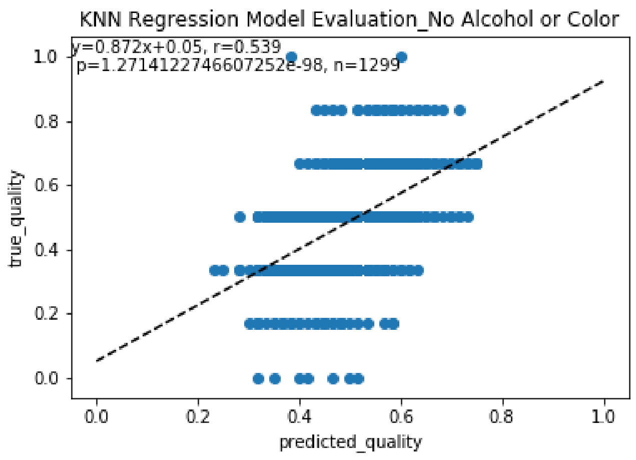


Supplementary Figure 1: Differential splits for KNN k optimization. For every value of k between 1 and 10, the dataset was split 80/20(left) or 70/30(right) for training and test sets respectively. The model was trained and then asked to predict the test set 30 times for each value of k. Average accuracy of the predicted label is plotted as a percentage of correctly labelled points. K with the highest average accuracy was chosen as optimal k.

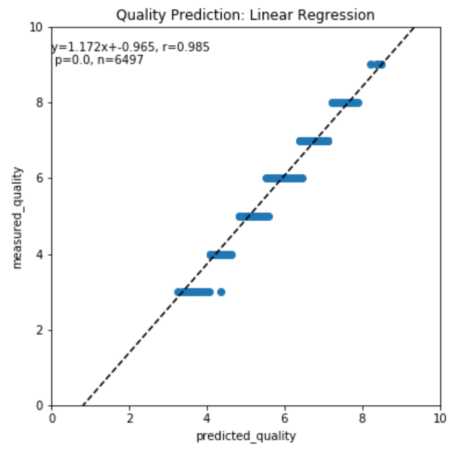




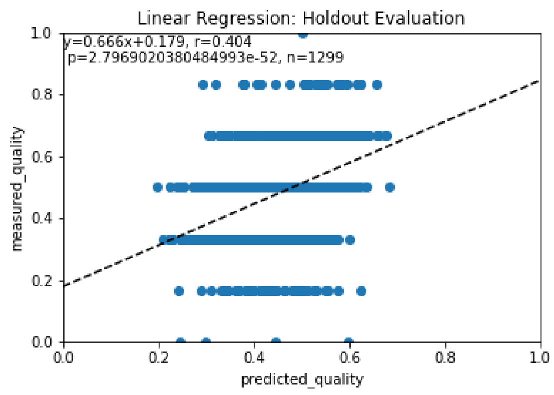
Supplementary Figure 2: Manual feature selection for KNN. KNN optimization for data with no alcohol content feature (left) or no alcohol or color feature (right). For every value of k between 1 and 10, the dataset was split 90/10 for training and test sets respectively. The model was trained and then asked to predict the test set 30 times for each value of k. Average accuracy of the predicted label is plotted as a percentage of correctly labelled points. K with the highest average accuracy was chosen as optimal k.



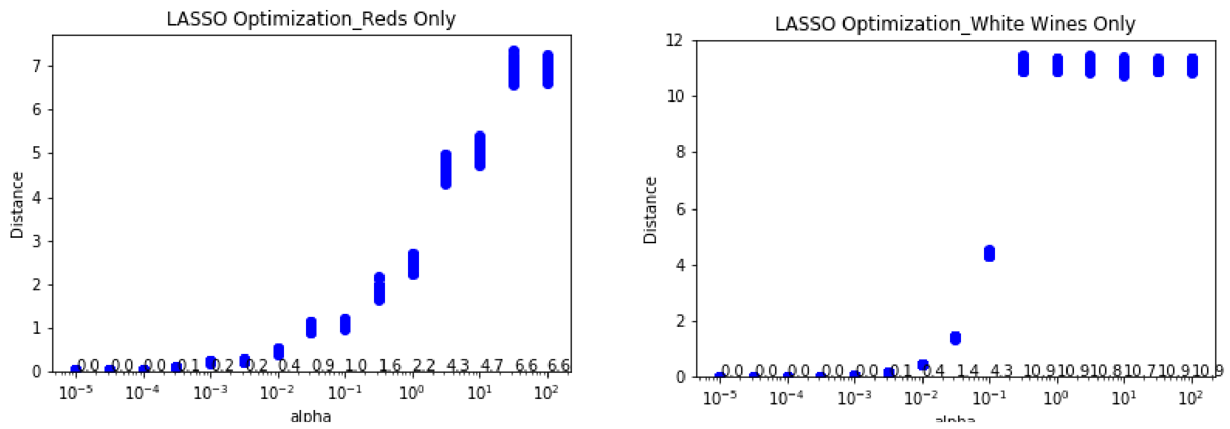
Supplementary Figure 3: Manual feature selection for KNN Regression. For k=10 neighbors, KNregression was performed with no alcohol content or color features. This reduced predictiveness for this model.



Supplementary Figure 4: Linear regression model for quality. This is not a true model, but demonstrates that prediction is not perfect with a lambda of 0.01, chosen to avoid overfitting.



Supplementary Figure 5: Linear regression model for quality, holdout evaluation with all features. The poorness of fit for this model spurred the hypothesis that feature selection may be required.



Supplementary Figure 6: Lambda optimization for red (left) and white wines (right) performed separately. For a range of possible lambdas, the dataset was split 90/10 for training and test sets respectively. The model was trained and then asked to predict the test set 30 times for each value of lambda. Average distance from actual to predicted label is plotted.