Two Approaches to Predicting Wine Quality

Joseph Despres

April 21, 2022

Contents

Introduction

Anyone looking to purchase a bottle of wine is facing a large and complex search space. Wine varies significantly in price, quality, and chemical properties with no obvious relationship; therefore, predicting the quality of a wine based on its features is a problem well suited for machine learning methods. In particular, this project aims to specify a function that maps the chemical and non-chemical properties of wine to a quality rating. To accomplish this, I will be using the Sklearn package (Pedregosa et al., 2011) in the Python Programming Language (Van Rossum & Drake, 2009). This report begins with a formal description of the problem followed by a detailed description of the data available. I explain two different datasets that will be used to model quality. After that, I compare different feature scaling on logistic regression performance and show a significant difference in runtime and accuracy. After that, I discuss the ensemble of models I am using Elastic net Multinomial Regression, XGBoost, and Support Vector Machines. After that, I discuss additional feature engineering that improved the ensemble's accuracy by roughly 2%. These models together reach a testing set accuracy of 67%.

Problem Description

This project aims to discover the relationship between wine quality chemical properties, and factors commonly associated with quality wine. The motivation for this project is that one purchasing wine seriously must consider different regions, grapes, growing climate, seasons, storage time, and substantial variations in quality and price. This is a vast and complex search space therefore, the problem is well suited to Machine Learning. This project aims to accurately predict the quality of wine given its chemical properties and determine if chemical properties are more informative of quality than common factors such as year, region, and price. The benefits of an accurate wine model are that one could select excellent wine at a low price, discover undervalued wines at markets and auctions, avoid low-quality wine, and purchase wine confidently without expertise.

Data

This study is predicting wine quality using different datasets. The first is based on the distinct chemical properties of a wine. Another is using more common factors such as price, year, and region. The chemical data is hosted by the University of California Irvine's Machine Learning Repository (Lichman, 2013). These data are from a study conducted by the Portuguese Government researching a Data Mining approach to predicting wine taste (Cortez, Cerdeira, Almeida, Matos, & Reis, 2009). In their study, they achieve an out-of-sample accuracy of 64% using Support Vector Machines. First, I do not have the domain knowledge to assess the validity of these data[1]. I have no idea what is a reasonable level of acidity, for instance. However, given that this is a published dataset, I assume all the observations are valid observations and not measurement errors.

The chemical Dataset is a structured, curated, tabular dataset made of numerical features conveniently in First Normal Form. The outcome we wish to predict is the quality as rated by a professional wine judge. It is not obvious from the UCI repository, but there are duplicated rows. This means multiple judges rated the same wine as the same quality. Judges are anonymous in these data, so there are no features indicating which judge is doing the scoring. There are 11 additional features, which are listed in Table 1.

The target is wine quality as scored by a professional judge. Wine quality is an ordinal categorical variable with a significantly unbalanced number in each category. The reality of this problem is that very few wines will be of very high quality and many will be average. See Figure 2 for a plot of the class distributions. Note that only 5 of 6,497 of these

[1] I abandoned Bayesian methods outlined in my proposal because I do not have the requisite expertise to set informative priors. Without informative priors, the model performance is similar to Machine Learning methods with a significantly longer runtime.

Table 1: Descriptive Statistics

Feature	Mean	Std
Fixed Acidity	7.215	1.296
Volatile Acidity	0.340	0.165
Citric Acid	0.319	0.145
Residual Sugar	5.443	4.758
Chlorides	0.056	0.035
Free Sulfur Dioxide	30.525	17.749
Total Sulfur Dioxide	115.745	56.522
Density	0.995	0.003
Ph	3.219	0.161
Sulphates	0.531	0.149
Alcohol	10.492	1.193
Quality	5.818	0.873
Is Red	0.246	0.431

Note: 6,497 observations

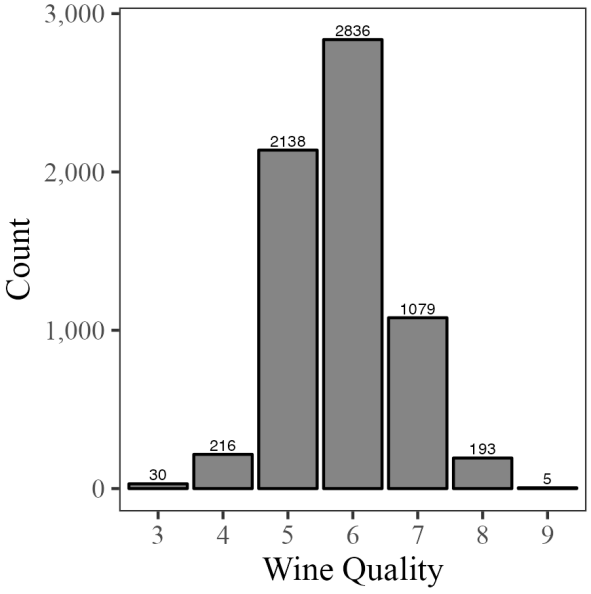

Figure 1: Wine Quality Determined by Professional Judge

wines are rated 9/10. This may necessitate some re-sampling methods because the most valuable model will be able to detect high-quality wine.

The second dataset used is provided by a Kaggle(Kaggle, 2021) competition. The host scrapped data from ratings published by a popular wine selecting company, Vivino. Unlike the first dataset, these are a mix of casual and sophisticated consumers. Some of the features are numeric however others are nested categories. The winery, region, and country are in a hierarchical structure, meaning a winery is within a region, within a country. To appropriately model this would require some multilevel hierarchical models that are, as I understand, beyond the scope of Scikit-Learn.[2]

To make these data compatible with Sklearn's modeling format I encoded these categorical variables into a matrix using 0 and 1 encoding resulting in a highly sparse matrix. I recognize this is not the most appropriate way to model this dataset, however, I want to compare the predictive power of the two datasets. Since these are in two very different formats, I am willing to make such a compromise. In the next section, we will set up methods to explore the predictive power of each dataset.

[2]I fit a hierarchical model using specialized statistical software (Stan) and found substantial random effects of winery independent of price, region, and country.

Table 2: Vivino Features table

Feature	Distinct
Wine Type	4
Country	33
Region	861
Winery	3505

Table 3: Vivino Numerical Features table

Feature	Mean	Std
Rating	3.866	0.296
Number of Ratings	428.322	1838.414
Price	33.025	70.900

Note: 13,834 observations

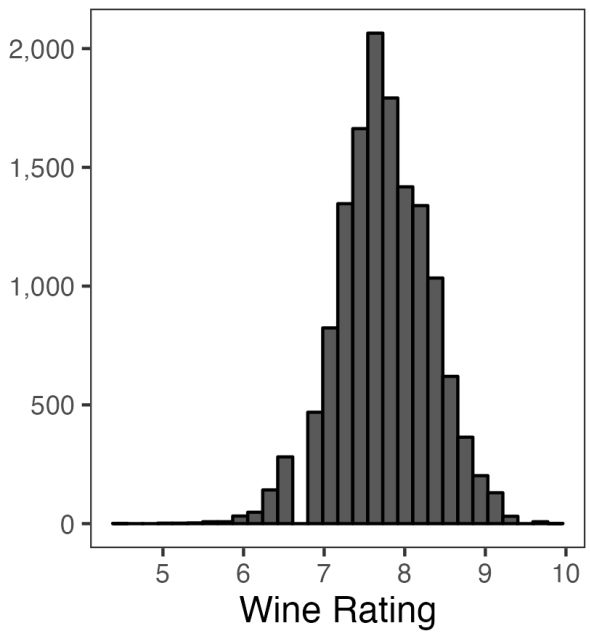

Figure 2: Vivino Wine Quality Scores

Methodology

I split the data into training and testing sets and reserved 20% of the data for testing. I will not touch these data until it is time to report a final result. This paper will consider two classification performance metrics and one prediction metric. The first is accuracy. Accuracy is measured as the ratio of correct predictions to predictions made. Note, this is an unbalanced assignment problem, accuracy should not be compared to the number of classes but rather the frequency of the most frequent class. The most frequent class is a quality score of 6. Which occurs in about 43% of the observations. This metric is intuitive but does not capture our true intentions. I will also use a balance between precision and recall AUC. AUC is simply the area under the ROC curve. To compare the performance of both datasets, I will use Root Mean Squared Error. This, metric is well suited for regression, not classification. However, the target class is numeric 1-10 so RMSE can be used by changing the data type from categorical to numeric and applying the formula.

After some difficulties fitting initial models to the chemical dataset, I test and document the results using several different scaling strategies.[3] I ran 2,500 different partitions of training and validation data with a 50% split. As shown in Table 1 and Figure 3, there is a significant difference in predictive power. This is due to the design matrices having difficulties inverting with inappropriate scaling. Notice that the mean AUC increases from 0.59 to 0.733 depending on the scaling method. Also, the standard deviation of the AUC is reduced from 0.024 to 0.015. The model is faster, more accurate, and more consistent using appropriate feature scaling methods.

Table 4: AUC by feature scaling method

Scaling Method	Mean	Std	Median
No Transformation	0.590	0.024	0.592
Dropped Outliers	0.645	0.026	0.641
Min Max Scale	0.690	0.018	0.691
Rescaled to Uniform	0.707	0.015	0.707
Standard Scale	0.726	0.017	0.725
Yao Johnson Transform	0.733	0.015	0.732

The Yao-Johnson outperforms the Standard scaling methods, however, not by much and is a far more complicated transformation process. Essentially, it is a Box-Cox transformation method modified to fit feature values less than 0. The Box-Cox Method is for linearizing vectors by finding the exponent that will convert the variable to a close to a linear scale. I conclude that is the best scaling strategy because is more accurate and has less standard deviation, and had a shorter runtime.

I plan to use an ensemble of models to predict wine quality. The intuition behind this is that a collection of classifiers will be more accurate than a single because one model could be wrong in many ways, however, it would be correct in only one.

[3] This is included because I was rather shocked by how much of a difference it made.

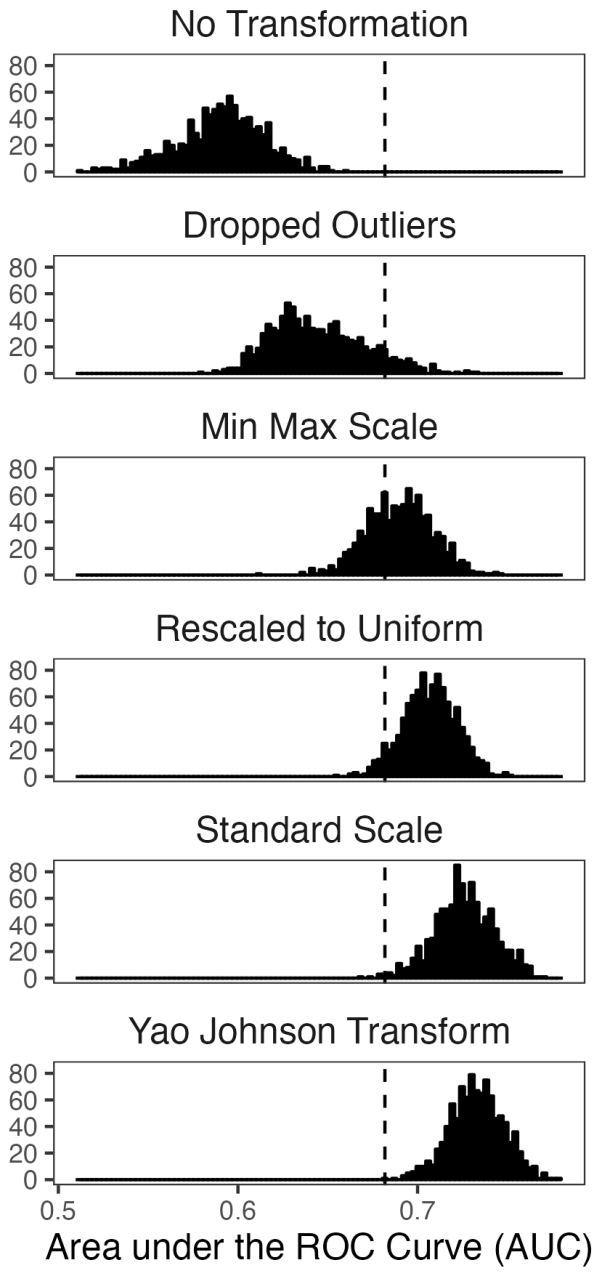

Figure 3: Comparing Resampled Prediction Accuracy by Transformationf

The first model is the Multinomial Logistic Regression to the model. This model is based on estimating the cumulative log odds of belonging to a given category based on the features. I add some regularization parameters L^1 and L^2 norms. commonly referred to as elastic net regression. A grid-search over a five-fold cross-validation shows that an optimal $L^1 = 0.0315$ and $L^2 = 0.013$. This achieves a training set accuracy of 51.3% and an AUC of 0.775. These are not impressive results, so we will fit a Support Vector Machines.

A Support Vector Machine aims to find the maximum margin hyperplane between features that separate points. I use the radial basis function kernel and tune only the regularization parameter squared L^2 norm. After cross-validating I find the best regularization parameter to be 6.58. This reaches an in-sample accuracy of 52.2% and an AUC of 0.781. A very slight improvement.[4]

After that, I use XGBoost (Chen & Guestrin, 2016) an efficient implementation of gradient boosted trees. There are many parameters to search through, however, I will only tune max tree depth, column sample by tree, and learning rate. After tuning, I found the column sample to be 50% to be the best a learning rate of about 0.034 and a max depth of 13. This yields an astonishing sample accuracy of 64.5% and an AUC of 0.848. This is a substantial improvement over the past two models. From here, I will wrap all three in an ensemble and test the accuracy. Given these metrics, it may be better to not even ensemble and just use XGBoost. I compromise by weighting the XGBoost model three times heavier.

After fitting this ensemble, I do some more feature engineering to improve the model as much as possible. First, I address the unbalanced assignment by bootstrap resampling until there are 10,000 rows of each of the quality categories. This strategy is crude and expensive, however, it improves the model by roughly 2% depending on the seed. After that, I do some imputation of outliers. To do this, I replaced any data point greater than 99.5% percentile

[4]Cortez was able to get to 62% by altering the decision boundaries having a model make more mid-range-quality predictions.

or less than the 0.5% percentile with an NA. After that, I used a simple KNN imputer to repopulate the data frame with an estimate based on the datapoint's nearest neighbors. This did not improve the out-of-sample accuracy. However, since It did not reduce it I will keep it, considering this will likely make the model more robust in the case of extreme outliers in the testing set.

I will do similar modeling with the second dataset, only changing classification for regression. I do the same Yao-Johnson feature scaling and grid search through parameters for an ensemble.

Results

There is a significant level of noise in both of these datasets. Additionally, there are important and informative features missing from both datasets. which explains the accurate but unimpressive predictions from both models. In this case the data with Chemical properties of wine. An ensemble of Elastic Net Multinomial Regression, Support Vector Machines, and XGBoost reach a training set accuracy of 63%. Individual performance is listed in Table 5. I was unable to significantly outperform the results in the paper accompanying the data without adjusting the classification boundaries. However, the factors more commonly associated with quality such as the region, year, price, and winery, are all missing from these data. Therefore, we will use the price dataset and determine if we can specify a more informative model. Expertise in this area is required to determine the limitations of this chemical model. I would like to study a combination of chemical properties from the UCI data and the Vivino data together. I believe that wine quality could be modeled far more accurately given those features combined.

Table 5: Classification Model Performance With Standard Metrics

	Accuracy	AUC
XGBoost	0.6377	0.8361
Elastic Net	0.5146	0.7604
SVM	0.5146	0.7584
Ensemble	0.6731	0.8885

Table 6: Model Performance

	RMSE Classifier	RMSE Regressor
XGBoost	0.7190	0.9265
Elastic Net	0.8053	0.9054
SVM	0.8053	0.9160
Ensemble	0.7458	0.9108

Conclusions

This study finds that wine quality can be predicted with an accuracy of 67% using standard classification boundaries. Chemical features to predict a professional's appraisal seems to be more informative than using year and price predicting. Wine quality can be modeled with machine learning methods, however, without more privileged access to data, this may not be very useful in practice as they are wrong a large portion of the time.

Further study in this area could be done with a comprehensive dataset combining chemical and non-chemical features of wine in the same dataset. Also, the discrepancy between a professional judge vs a casual consumer could be different in only location or actual preference. I would like a better understanding of the differences in ratings. Additionally, the weather could also be combined into the data and determine when the best climate for a given grape making up a wine.

References

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2939672.2939785` doi: 10.1145/2939672.2939785

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, *47*(4), 547-553. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0167923609001377` (Smart Business Networks: Concepts and Empirical Evidence) doi: https://doi.org/10.1016/j.dss.2009.05.016

Kaggle. (2021). *Wine rating and price.*

Lichman, M. (2013). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual.* Scotts Valley, CA: CreateSpace.