The Efficacy of ML Models on White Wine Quality Prediction

Zachary Harris

Dept. of Computer Science
University of Wyoming
Laramie, Wyoming

***Abstract*—This paper showcases a few different ML models using standard model hyperparameters on the prediction of white wine qualities. It also shows the improvement in model performance as the dataset size increases, showing that there is still room to generalize with more data.**

On the white wine dataset from the Wine Quality dataset, I achieved a test prediction accuracy of 66.0 ± 0.7% using random forest regression with standard hyperparameters from the scikit-learn Python library [1][3]. This outperforms all other tested methods on a similar time budget.

I. Introduction

A. Motivation

In a simple applied ML setting, many end-users may want to simply copy a simple command from somewhere online, or have an easily implementable interface to use that will guarantee a good solution for their problem. In the wine quality dataset, we can imagine a situation where a tech-savvy sommelier wants to train an ML model on this dataset to predict if certain wines will be worth their time and money to taste.

The goal of this paper is to showcase the outcomes of using a few different ML models in the task of wine quality prediction. This should enable some sommelier to easily pick a single model and create their quality predictor.

B. Proposed Solution

Since I am analyzing simple ML models on this white wine quality dataset, I chose 4 commonly used models with 2 models having 2 different setups. Specifically, I chose random forest regression, support vector machine regression, XGBoost with standard hyperparameters and with increased boosted decision tree count, multiple linear regression, and multiple linear regression using the Huber loss function.

Since we know the goal of this paper is to analyze the efficacy of the simple models in predicting white wine quality, I broke up the problem into 3 steps: analyzing the data, predicting the wine quality, and verifying the predictions.

The first part was done in R to graph and easily visualize the data.
Once pre-processing on the data was finished, the models were built in python and the performance was tracked as they were trained, giving us Figures 3-5.
The final step of prediction verification came by running 30 randomly seeded trials of model training and prediction to give us good estimates of the model MSEs.

II. Background

A. Model Overview

XGBoost regression is a gradient boosted decision tree algorithm developed by Nvidia as a fast, accurate, and scalable algorithm for use on their GPUs. This algorithm uses 100 gradient boosted decision trees ensembled together to predict the response.
Random forest regression is a simple random forest regressor using 100 decision trees that ensemble 100 trees using a random subset of features for each tree to predict the response.
Multiple linear regression is a simple method where multiple linear relationships between the features and the response are ensembled together to predict the response. The Huber loss function is also used in this experiment as a test to see if the MLR model was overly restrained by outliers in the data.
Support vector machine regression uses support vector machines to approximate the linear relationships between the features and the response variable. Yet again, the individual relationships are ensembled to predict the response.

B. Dataset Description

The dataset I used for this project was a subset of the wine quality dataset focused solely on white wine [1]. This dataset has 11 features to describe the wine, and one response variable, being the quality of the wine. The features are as follows: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulfates, and alcohol.

For this project, I began by graphing pairs of every feature and response variable to see if there were any interesting relationships. There were three main things I got out of this. The first, is that I got an idea as to how the data is distributed. The second is the strong correlations between density and alcohol, and density and sugar are particularly interesting. Finally, the third thing I got out of this was that I could easily identify the need to remove some outliers

from the data.

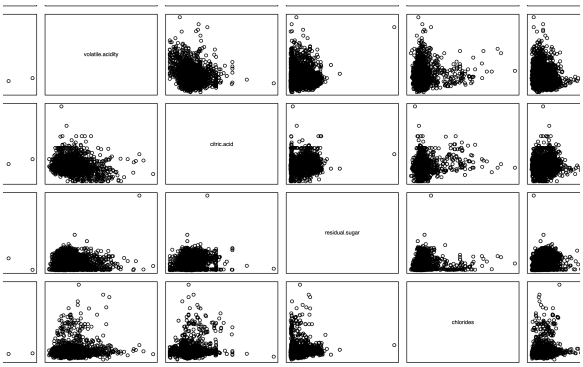

Figure 1: A picture showing the data before outlier removal.

As shown in Figure 1, we can see that the outliers in the data force much of the distribution to be bound to a corner of the graph. This is bad because if we normalize the data, then we are left with the majority of the data seeming much closer together than it should be. To fix this, I removed all points that lied greater than five standard deviations away from the mean. Figure 2 shows the updated graphs, which have a much better distribution of the data across the feature space.

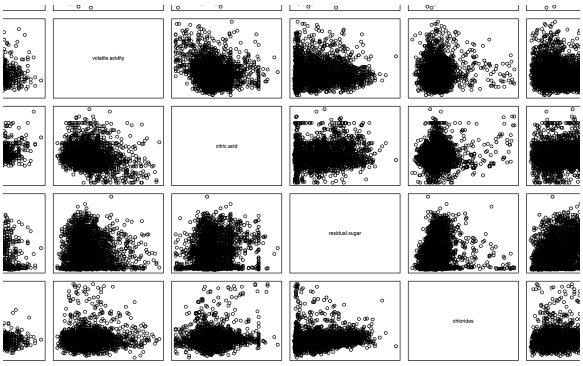

Figure 2: A picture showing the data after outlier removal.

III. Project Design

A. Methodology

This problem for which we are describing has 11 features and a single response. The interesting thing about this response variable is that while it can contain 11 possible outputs (0 through 10), there are only 7 in the data (3 through 9). This leaves the potential for a classification problem with 7 outputs, and also a regression problem as we can format these distinct 'classes' of quality into a soft rating that we can then exploit with a regression algorithm.

For this experiment, I used a multitude of regression algorithms to estimate the quality of the final quality. I prefer this over a distinct classification problem because when encapsulating this problem as a regression problem, we preserve the spatial dynamic that the different qualities have (i.e. the numbers closest to the actual response should be predicted more often). Whereas in a classification setting, we would essentially be learning the dynamics between the qualities in an unsupervised manner.

The models I used in this experiment include multiple linear regression, Huber regression, random forest regression, support vector regression, and a gradient boosted decision tree algorithm called XGBoost through the scikit-learn python library [2][3].
The idea behind this experiment is to show the performance of the different libraries and how they are able to generalize throughout the training process.

Within this experiment, I used Mean Squared Error (MSE) to showcase the overall distance off of the actual quality for the wine because a drastic difference in quality prediction should be penalized much more heavily in my opinion than a basic l1 distance might normally. The idea here is that we want to get the closest possible to the actual predictions even if we can't guess the right quality.
It should also be noted that I chose not to round the predictions, unless specified otherwise, to integers to match the qualities shown in the data because I felt that the reduced information from the rounding process would make the comparisons between the different models less meaningful. A rounded prediction might hide information about a model's actual predictions that the standard MSE might better showcase.

IV. Experimental Outcome

A. Results

Within this project, I have identified that the best ML method to use for this dataset is Random Forest algorithm.

The random forest algorithm performed best with an average MSE of 0.385 ± 0.003, and an average MSE with rounded predictions of 0.439 ± 0.010. The support vector regression performed with an average MSE of 1.364 ± 1.264, and an average MSE with rounded predictions of 1.442 ± 1.264. The XGBoost regression model performed with an average MSE of 0.424 ± 0.0, and an average MSE with rounded predictions of 0.472 ± 0.0. The XGBoost regression with 1500 boosted trees performed with an average MSE of 0.413 ± 0.0, and an average MSE with rounded predictions of 0.453 ± 0.0. The multiple linear regression performed with an average MSE of 0.538 ± 0.0, and an average MSE with rounded predictions of 0.630 ± 0.0. The Huber regression performed with an average MSE of 0.552 ± 0.0, and an average MSE with rounded predictions of 0.673 ± 0.0.

These results show that the random forest regression, in a similar amount of time to the nearest runner-up, XGBoost with 1500 boosted trees (824 seconds vs 796 seconds), outperformed it to a significant degree.

Figure 3 shows a training scheme for support vector machine (SVM) linear regression compared to multiple linear

regression (MLR). Here, we can see that the SVM significantly underperforms MLR, with a wide variance in its training outcome. In the following graphs, we do not compare SVM against the other methods.

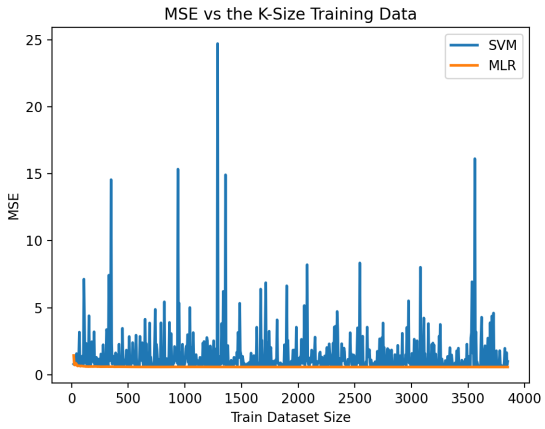

Figure 3: A plot showing test MSE against the training dataset size, comparing support vector regression vs multiple linear regression.

Figure 4 shows the overall comparison between the 4 other methods, with random forest regression using 100 decision trees coming in with the lowest MSE.

Figure 4: A plot showing test MSE against the training dataset size for 4 methods.

Figure 5 shows the overall comparison between the two XGBoost algorithms. This comparison highlights how small the performance gain is between using 100 and 1500 boosted decision trees.

Figure 5: A plot showing test MSE against the training dataset size, comparing multiple linear regression vs XGBoost with a standard 100 boosted trees vs XGBoost with 1500 boosted trees.

Through the two plots showing the XGBoost regression algorithm, and the single plot showing the random forest regression, we can see that as the model training data grew, the model continued to learn. This shows that there is still some room for improvement in these models. This contrasts the multiple linear regression and Huber regression models that appeared to stop learning after the training dataset size reached 1000 data points. This shows a limit to the linear complexity of the model over the 11 features it is modelling that the other algorithms are able to outperform.

B. Conclusion

We can see that the model best suited to be used for white wine quality prediction using un-optimized model hyperparameters, is random forest regression. While the expected accuracy is only around 66%, the error is that is a very good starting point for further optimization and giving an MAE of 0.371 away from the actual quality is an astoundingly accurate score. This means that on average, each predicted quality should be very close to the actual quality.

While XGBoost regression gives close performance using 1500 boosted trees and a similar time budget, without further optimizing the model hyperparameters, it is still recommended to use random forest regression instead.

In a future work, it would be worth exploring the hyperparameter optimization task for this problem as it could yield interesting performance gains for different models. I personally expect that an optimized XGBoost regressor would outperform the RF regressor every time.

REFERENCES

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009. A. Cerdeira, F. Almeida, T. Matos and J. Reis, Viticulture Commission of the Vinho Verde Region(CVRVV), Porto, Portugal, 2009.

[2] T. Chen and C. Guestrin, "XGBoost," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[3] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Mach. Learn. Res., 2011, doi: https://doi.org/10.5555/1953048.2078195.

APPENDICES

C. Non-Default Model Hyperparameters

Other changed model hyperparameters include increasing the max iteration count from 1e4 to 1e8 for the multiple linear regression and huber regression models so they would not terminate early.

D. Links

This document was formatted using IEEE standards found at the Purdue Owl website.

All code and plots are available at my github.

Other citations are included in the code comments as they were used.

E. Other Metrics

Model Accuracy with Rounded Predictions: SVM: 0.351 ± 0.134, XGBoost: 0.647 ± 0.0, XGBoost-1500: 0.655 ± 0.0, RF: 0.660 ± 0.007, MLR: 0.519 ± 0.0, HR: 0.516 ± 0.0.

Model Mean Absolute Error: SVM: 0.908 ± 0.391, XGBoost: 0.467 ± 0.0, XGBoost-1500: 0.422 ± 0.0, RF: 0.445 ± 0.002, MLR: 0.574 ± 0.0, HR: 0.582 ± 0.0.

Model Mean Absolute Error with Rounded Predictions: SVM: 0.876 ± 0.404, XGBoost: 0.389 ± 0.000, XGBoost-1500: 0.378 ± 0.0, RF: 0.371 ± 0.007, MLR: 0.529 ± 0.0, HR: 0.544 ± 0.0.