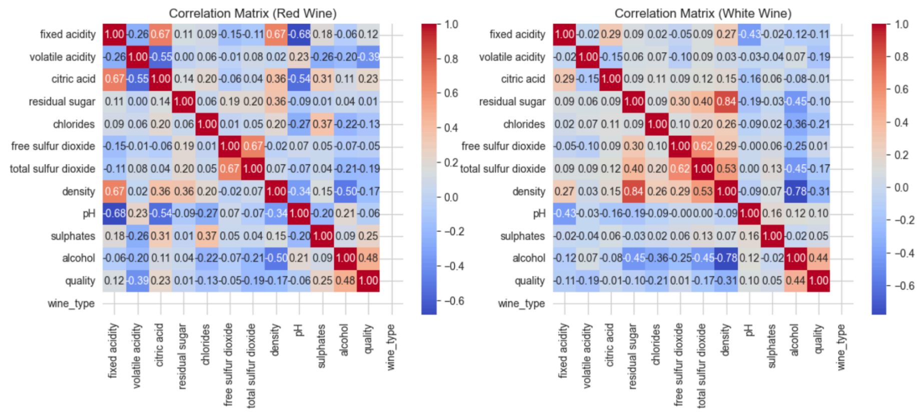
Predicting Wine Quality Based on its Chemical Properties

By:
Ingrid Morales, Viveka Agrawal, Huizi Xu

For our project, we decided to tackle the problem of predicting wine quality based on its chemical properties. We obtained our data from UCI's Machine Learning Repository. This dataset contains two different datasets for red and white wine. For our project, we decided to combine these two datasets and introduce a new feature 'wine_type' that distinguishes red and white wine. Essentially we are solving a multiclass classification problem. Our goal is to build a model that predicts a rating from 0 to 10, where 0 is the worst rating and 10 is the best possible rating. The dataset has 11 features to help our model predict wine quality. We explored gradient boosting and random forest to help solve this multiclass classification or regression problem, as we noticed that the underlying data was nonlinear. Since all 11 features are continuous, we can also treat this problem as a regression problem that predicts a continuous value from 0 to 10. We explored the gradient boosting and random forest algorithms to solve our problem and compared the results at the end to conclude what the best model is. In this paper, we will discuss in more detail our data exploration, model exploration and selection, data preprocessing and feature design, and lastly, performance validation to decide on our final model.

The first step in our project is data exploration. We used the Pandas and Seaborn library to understand and visualize the features underlying distribution. By the histograms, it was very apparent that red and white wine chemical properties are quite different. Therefore, it was imperative that we introduce a new feature so that our model can distinguish between red and white wine. We computed the correlation matrix (Figure 1) separately for red and white wine and concluded that in both matrices, the features 'free sulfur dioxide' and 'total sulfur dioxide' were highly correlated. Therefore, we decided to remove 'total sulfur dioxide' since we only need one of those two features. We further visualized the features against the quality to see if there were any linear relationships, but the scatter plots proved otherwise. The scatter plot made it clear that our features had a nonlinear relationship with quality.

Figure 1: Correlation matrix for the wine dataset



To address the problem at hand, we first built a model that implements the gradient boosting framework. Ingrid was responsible for writing the code for the gradient boosting algorithm. For this reason and for gradient boosting predictive power, we decided to give this algorithm a try. First, I trained my dataset using the Gradient Boosting Regression model from the scikit-learn library. In order to find the best parameters to finetune my model, I used Grid Search CV to exhaustively search for the best specified hyperparameter combination. I also used the method of cross validation from the scikit-learn library to ensure my mean squared error (MSE) was consistent across different subsets of the data. My cross validation MSE came out to be 0.4845 with a standard deviation of ±0.0391. After finding the best parameters for the learning rate, n estimators, maximum depth, minimum sample split, and minimum sample leaf node, the MSE for my test data was 0.3456. The root MSE was 0.5879 and the R squared was 0.5169. Figure 2 displays the MSE loss over 200 boosting stages. It is notable to mention that more boosting stages would overfit our model. I also attempted to include the following interaction terms, in hopes that it would improve the model: *wine_type* * *chlorides*, *wine_type* * *density*, *wine_type* * *volatile acidity*. Due to the different correlations among the features dependent on the wine type, I created three interactions that would account for that dependence. However, the model did not perform better with the presence of interaction terms. The test data had an MSE of 0.3667 and an R-squared of 0.4875. Therefore, we will continue with the model without any interaction terms.

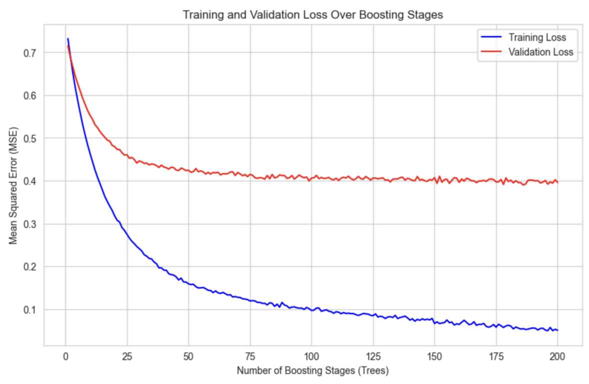


Figure 2: Loss graph for Gradient Boosting

Next, we built a model that implements the Random Forest classifier, and Viveka was responsible for this task. I trained my dataset using the Random Forest Classifier (for calculating accuracy) and the Random Forest Regression (for calculating MSE) models from the scikit-learn library, then proceeded to find the best hyperparameters for maximum depth, minimum sample split, minimum sample leaf node, maximum features, and bootstrap using the Randomized Search CV to work with the best possible model, and made sure to include the Stratified K-fold cross validation method (also from the scikit-learn library). After running the model a number of times, the best result I achieved with the least difference in training vs. validation was: Train MSE = 0.4449, Val MSE = 0.5354, Train Accuracy = 0.6961, Val Accuracy = 0.6000 (Figure 3). From the results, the model was not performing well and there was definitely some overfitting

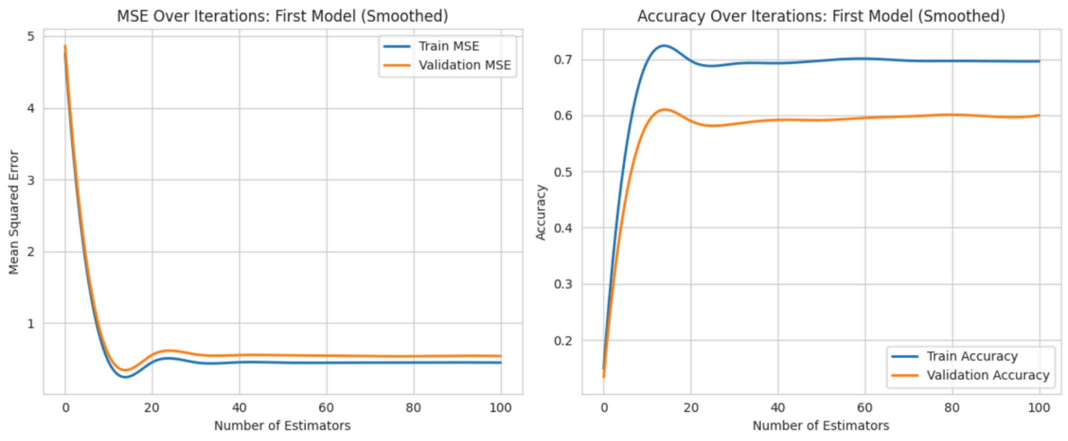


Figure 3: MSE and Accuracy graphs for the initial Random Forest Model

happening, so I modified my model by applying the SMOTE method to address the class imbalance since the Wine Quality Distribution is as follows: {3: 30, 4: 216, 5: 2138, 6: 2836, 7: 1079, 8: 193, 9: 5}, showing that most wines are predicted as having a quality of 5 or 6. After applying SMOTE, the balanced Wine Quality Distribution becomes: {3: 2836, 4: 2836, 5: 2836, 6: 2836, 7: 2836, 8: 2836, 9: 2836} and the best result I achieved with the least difference in training vs. validation using the exact same hyperparameters as the first model is: Train MSE = 0.3650, Val MSE = 0.4370, Train Accuracy = 0.8123, Val Accuracy = 0.7705 (Figure 4). There still is some overfitting happening, but the model performs much better and makes more accurate

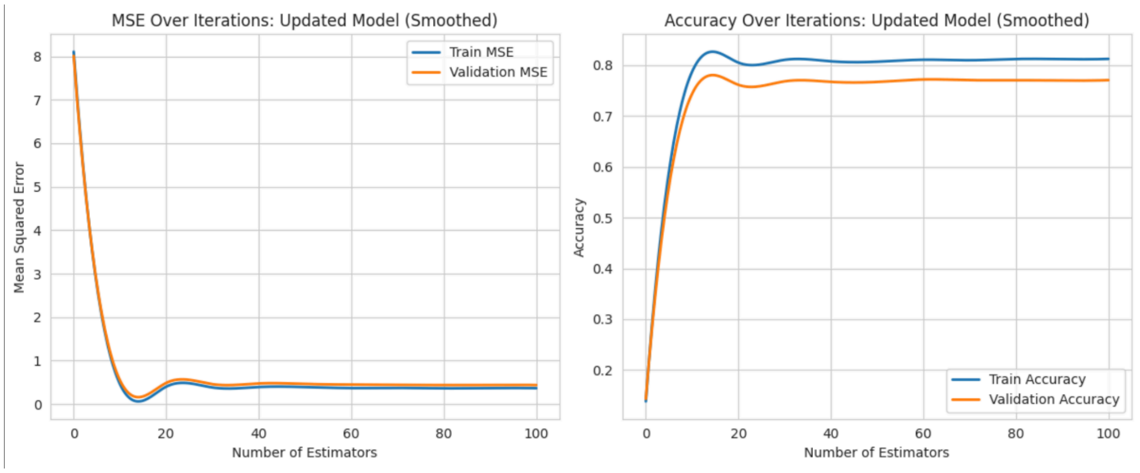


Figure 4: MSE and Accuracy graphs for the best Random Forest Model

predictions on the quality of wine in comparison to the model before. I tested out more than 2 models, but I thought it was best to compare the first model with the best model I was able to test in the report. In addition, I attempted to introduce interaction terms to the model in hopes of better performance but noticed that the MSE and accuracy values were worse than without them.

Finally, we need to compare different models to determine which one is the most accurate predictor of wine quality, and Huizi is responsible for this task. Although from the scatter plot of data exploration, I know that our features have a nonlinear relationship with quality. I trained a Linear Regression model as a baseline for predicting wine quality (MSE: 0.5408, RMSE: 0.7354, R²: 0.2672). The experimental results demonstrate that the Gradient Boosting Regressor outperforms both Random Forest and Linear Regression in predicting wine quality, achieving the lowest MSE (0.3456), RMSE (0.5879), and highest R² score (0.5169). This superior performance, evident in the residual plots (Figure 5), underscores the effectiveness of ensemble methods in modeling nonlinear relationships between chemical properties and wine quality. The plots reveal Gradient Boosting Model's tightly clustered errors near zero, contrasting with the wider spreads of Random Forest and Linear Regression. While all models performed better on white wines (narrower residuals) than red wines, Linear Regression exhibited slight skew in its residuals, reflecting systematic biases and reinforcing its inadequacy for nonlinear problems (R²: 0.2672). Random Forest, though competitive (MSE: 0.3650, R²: 0.4380), fell short of Gradient Boosting's precision, likely due to the latter's iterative error-correction mechanism. The inclusion of the *wine_type* feature improved model discrimination, as seen in the distinct error distributions between wine types. Despite Gradient Boosting's dominance, moderate R² scores suggest room for improvement through hyperparameter tuning, feature engineering, or reframing the task as classification to align with wine quality's discrete scale. These results validate the utility of combining datasets and leveraging ensemble methods, with Gradient Boosting emerging as the optimal model for practical deployment.

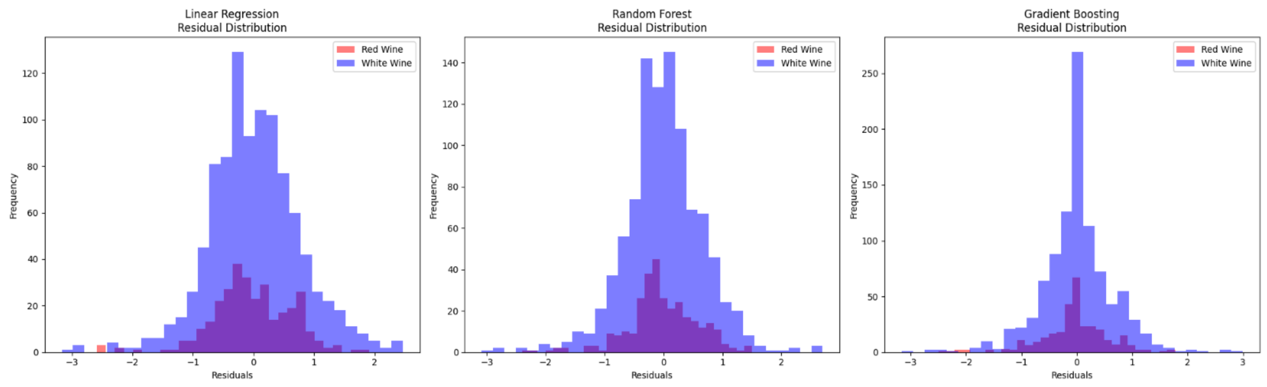


Figure 5: Residual Distributions by Model and Wine Type