Wine Quality Predictions

By:

Jordan Hayes

**Introduction:**

The Wine Quality dataset was used to build machine learning models using Python in Google Colab. The important libraries used in this study were sci-kit learn, pandas, numpy, and plotly. Use of large language model, ChatGPT, was used to assist in creating code to aggregate, visualize, and generate models. The models created were aimed to generate models, like Random Forest Regressor and XGBoost to predict wine quality based on 12 different dimensions. The features of the wine in the dataset were: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. Through pre-processing, feature reduction, and hyperparameter tuning, the model with the lowest mean squared error was used.

**Literature Review/Background**

There was one article that was used to assist in creating machine learning models to best predict the quality of wine in this study, *Machine learning-based predictive modeling for the enhancement of wine quality.* The main areas of that were taken away from the literature were the machine learning methods used to predict wine quality, pre-processing steps, dimension reduction methods, and visualization techniques. The key difference between the wine quality being predicted in the reviewed literature and this study is that in the reviewed literature, quality was split into classes ‘*good quality’* and ‘*bad* quality’ based on the record’s *quality*. Records with *quality ≥* 7 were put in the ‘*good quality’* class and records < 7 in ‘*bad quality’.*

The preprocessing steps taken in literature study were dropping duplicate values and normalizing the data. Their data was normalized by making the mean of each dimension 0 and the standard deviation 1, a very common approach to normalizing data. The next step in this study was to run dimension reduction methods to find which features were the most important to predicting wine quality. They used Feature Significance analysis based on the Random Forest and XGBoost model. The top 3 features that were found in the Feature Significance analysis were alcohol, sulfates, and volatile acidity. It is important to note that their most important features will be different than the ones found in this study because they are predicting 2 classes, *‘good quality’* or *‘bad quality’*, where in this study, the actual quality of wine is to predict. The difference in classification will lead to different results in feature significance which will be shown later in this report.

The next steps the article took to further analysis the importance of dimensions was looking at descriptive statistics and cluster analysis. The methods used in the cluster analysis were creating a dendrogram, to see the relationship between features, and spearman’s correlation to compute variance in the dataset. The 8 selected features were: alcohol, fixed acidity, volatile acidity, residual sugar, chlorides, free sulfur dioxide, pH, and sulfates. The removed features were citric acid, total sulfur dioxide, and density.

After the most important features were decided, the Random Forest and XGBoost models went through hyperparameter tuning. Hyperparameter tuning is when the parameters used to create estimators are given different options and models are run with each option. For example, when tuning Random Forest parameters, the model will be run with the *criterion* parameter as *‘gini’* and ‘*entropy’,* the parameter that produces the model with the best results is the parameter that should be chosen.

After preprocessing, normalization, feature analysis and dimension reduction, and hyperparameter tuning, the final Random Forest and XGBoost models were run. What this study found was that the classifiers that were “trained and evaluated utilizing key factors” performed better than models that did not.

**Methods**

Pre-processing

The first step in preprocessing was to get rid of duplicate rows. There were \_ rows deleted after checking for duplicates and getting rid of them. The next step in preprocessing was to get rid of outliers. The method used in this study was to get rid of any records that were 4 z-scores away from the mean of the respective dimension. After calculating z-scores for each dimension **­\_** rows were deleted. After running and optimizing the models, the dataset with the outliers present actually performed better.A diagram of a graph

Description automatically generated

Fig. 1

The next step of pre-processing the wine data was to normalize the data. There were two different methods used to normalize the data, min-max normalization, and mean/standard deviation. Min-max normalization normalizes the data by reducing each dimension to a scale of [0, 1] with the minimum value as 0 and the maximum value of 1. Mean/standard deviation normalization normalizes the data by scaling the data so that each dimension has a mean of 0 and a standard deviation of 1. Mean/standard deviation saw the best models results, so that set of normalized data was the one that was used for all of feature selection, hyperparameter tuning, and modeling.

Feature Selection

Feature selection was the hardest, but the most important part of this study was on predicting wine quality. The first step of feature selection was to look at the correlation matrix in Fig. \_. Here you can see that *alcohol, density,* and *volatile acidity* have the highest correlation with quality. *Alcohol* has the highest positive correlation at .46, and *density* and *volatile acidity* have negative correlations at -.3 and -.21 respectively. None of these features have extremely strong correlations, but they are worth noting and will come up later for feature selection.

A screenshot of a graph

Description automatically generated

Fig. 2

A graph with a line

Description automatically generated The next step was using recursive feature elimination using Random Forest as the model and weighted f-1 score as the statistic to decide which feature is more important. It is important to note that weighted f-1 was used to account for models trying to predict multiple classes that were also imbalanced. After running RFE, the optimal features decided by the methodology was 11, but looking at Fig. 3, after 8 features, the mean f-1 score does not increase by a significant amount.

Fig. 3

Next, the most important features were determined. In Fig. 4, it shows the features listed by importance, which was measured running a Random Forest model and determined importance based on weighted f-1 statistic again. After looking at the optimal feature amount and the importance of each feature, the following 8 features were determined to be the best for the model: *chlorides, free sulfur dioxide, pH, residual sugar, density, alcohol, citric acid,* and *volatile acidity.* *Chlorides, free sulfur dioxide, residual sugar, ph* and *citric acid* were decided based on feature importance. *Chlorides*, *density,* and *volatile acidity* were decided based on the correlation matrix.

|  |  |
| --- | --- |
| **Feature** | **Importance** |
| fixed.acidity | 0.06533268 |
| volatile.acidity | 0.06930648 |
| citric.acid | 0.07406018 |
| residual.sugar | 0.07984929 |
| chlorides | 0.09584467 |
| free.sulfur.dioxide | 0.09090006 |
| total.sulfur.dioxide | 0.07283097 |
| density | 0.13847651 |
| pH | 0.08214645 |
| sulphates | 0.07685891 |
| alcohol | 0.1543938 |

Fig. 4

Hyperparameter Tuning

For hyperparameter tuning, sci-kit’s library GridSearchCV was used to create a grid of hyperparameters so that the models could be run with the parameter options, and the model with the best score had its parameters output. The metric used to evaluate models to create the best parameters was mean squared error. In all hyperparameter tuning scenarios, the parameters that were being tuned were: criterion, max\_depth, n\_estimators, min\_samples\_leaf, and min\_samples\_split. From the f1-weighted hyperparameter tuning, the best parameters were that generated a mean squared error of 0.254:

{'n\_estimators': 100, 'min\_samples\_split': 2, 'min\_samples\_leaf': 1, 'max\_features': 'sqrt', 'max\_depth': None, 'criterion': 'squared\_error'}

Next, in a hyperparameter method that uses PCA to reduce dimension and then determine the best model based on weighted f-1, mean squared error and Manhattan distance, the following parameters were chosen, with the following statistics.

*Estimators*

The main estimator that was used in this study was the Random Forest Regressor. The Random Forest Classifier was used but the best results came from the Random Forest Regressor. It was interesting to see that the regressor model performed very well because the quality was ordinal data, so regression worked very well. The Gradient Boost Regressor and XGBoost classifier were attempting to leverage Random Forest Regressor and Random Forest Classifier as the weak learner, but neither of the model results were better than Random Forest Regressor. Therefore, when optimizing the models, the Random Forest Regressor model was the only estimator focused on and optimized.

*Hyperparameter tuning*

This code in Fig. 5 was used to tune the hyperparameters of the Random Forest Regressor using all the features. These results were used to create the model with the lowest mean squared error, 0.255. The parameters chosen were used to create the model in Fig. 6, the final model used in this study that generated the lowest mean squared error.

A screen shot of a computer program

Description automatically generatedFig. 5

**Results**

TheA screen shot of a computer program

Description automatically generated Random Forest Regressor create with the below code generated a model that predicted wine quality with a mean squared error of .254. This mean squared error is quantifying the difference between the predicted value and the actual value, and then squaring the distance and finding the mean over those values. In the case where the numbers are so close like in the wine quality predictions, the maximum at 9 and the minimum at 3, this mean squared is good but again, not great.

In the study, the biggest reason the mean squared error wasn’t lower was due to the hyperparameters not being tuned to the optimal point to generate a better model. There were more parameters in the model that could have been changed. There also could have been a better feature method, whether it be reduction or change.

The last shortcoming that I noticed in my model was that it could not account for the class imbalance, which seemed to be the biggest struggle in the models’ performance. Wine quality with a value of *‘9’* only has seven occurrences, whereas records with a quality of *‘6’* had 2,049 occurrences. Undersampling and oversampling techniques like SMOTE were attempted, but with no success.

**Conclusion**

This study was not as successful as it could have been. The shortcomings of this study into machine learning methods to predict wine quality most likely came from not tuning the hyperparameters, feature selection, and imbalanced classes. There were changes made, but the changes could have been too excessive or too little or just in the wrong area. Using stacking methods like Gradient Boost and XGBoost did not generate better results, even though using an estimator’s predictions to build another estimator would lead to the idea that it would generate better predictions. More analysis must be done and starting the learning from scratch and looking at it from a different angle will most likely generate better results.