ISyE 6740 - Spring 2024
Final Project Report

Team Member Names: Regina Kang (rkang47, Term Project 003)

Project Title: Predicting White Wine Quality Based on Physicochemical Tests

Contents

1 Problem Statement

Wine is one of Portugal's most popular products and it is an essential part of Portuguese culture. Portuguese wines originated in around 2000 BC. Wines produced during these times served as a type of currency, making it a vital part of everyday life in Portugal. In the 14th century and again in the 18th century, the exportation and development of Portuguese wine skyrocketed [3]. Portugal is one of the top ten countries in terms of wine exports [2] and vinho verde is a unique wine from the Minho region of Portugal that is appreciated for its freshness and accounts for 15% of the total Portuguese wine production [1].

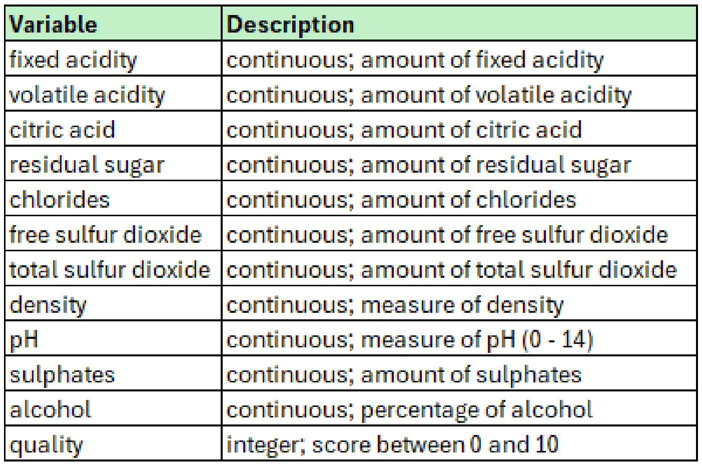
As vinho verde is one of the most exported products in Portugal, it is imperative for the vinho verde wine producers to make data-driven decisions in terms of the features to focus on when creating wines and how to rate the quality of their wines. In this project, I aim to test multiple models to predict vinho verde white wine quality from the Minho region of Portugal based on physicochemical tests. These models will serve as a powerful tool for white wine producers in assessing what data-driven decisions they should make. This analysis and modeling can have a tremendous impact on the wine industry in Portugal and worldwide if scaled as wine producers will be able to make better data-driven decisions, sell more wines, and improve the economy.

In a previous analysis, P. Cortez et al. explored three regression techniques in modeling wine quality [1]. The techniques used were multiple linear regression, support vector machine (SVM), and neural networks. They found that the support vector machine model achieved the best results. I will expand on their analysis by testing these models as well as additional predictive models. In order to make more novel findings through this project, I will also explore ensemble methods as well as combinations of ensemble methods (using stacking and blending models) to see if I can achieve better predictive accuracy.

2 Data Source

The data set for this project was downloaded from the UC Irvine Machine Learning Repository. I was able to access it by downloading it from the link below. There are two separate .csv files, one with red wine data, and one with white wine data. I have chosen to analyze only the white wine data. The white wine data consists of 4898 observations and 12 variables. The variables consist of 11 predictor variables and 1 response variable. The data set is made up of data on white vinho verde white samples from the north of Portugal. Please see Figure 1 below for descriptions of all the variables.

$$https://archive.ics.uci.edu/dataset/186/wine+quality$$



Variable	Description
fixed acidity	continuous; amount of fixed acidity
volatile acidity	continuous; amount of volatile acidity
citric acid	continuous; amount of citric acid
residual sugar	continuous; amount of residual sugar
chlorides	continuous; amount of chlorides
free sulfur dioxide	continuous; amount of free sulfur dioxide
total sulfur dioxide	continuous; amount of total sulfur dioxide
density	continuous; measure of density
pH	continuous; measure of pH (0 - 14)
sulphates	continuous; amount of sulphates
alcohol	continuous; percentage of alcohol
quality	integer; score between 0 and 10

Figure 1: Descriptions of All Variables

3 Exploratory Data Analysis and Data Preparation

As you can see from the variable descriptions, the data consists of 11 continuous feature variables and 1 integer response variable. There are no missing values. The feature variables measure the following physicochemical properties of each wine: 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', and 'alcohol'. The response variable is an integer quality rating between 0 and 10. The histogram of the response variable in Figure 2 shows that the most common quality ratings are 6 and 5. There are very few ratings at the extremes of 3 and 9, and there are no wines rated 0, 1, 2, or 10.

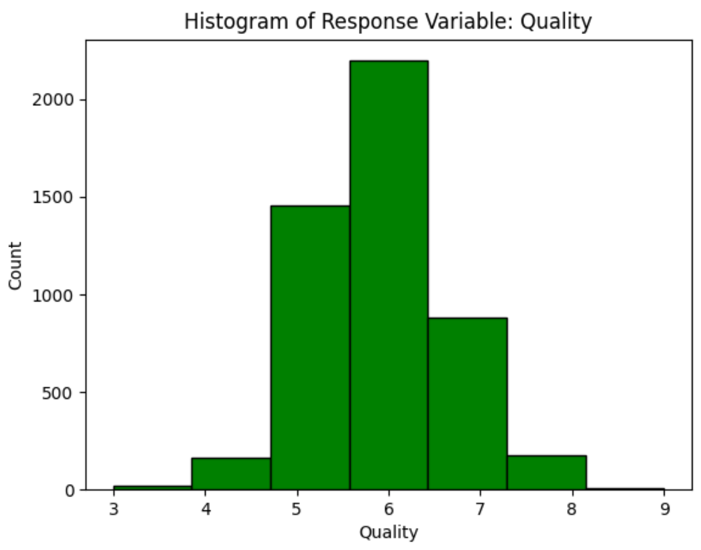
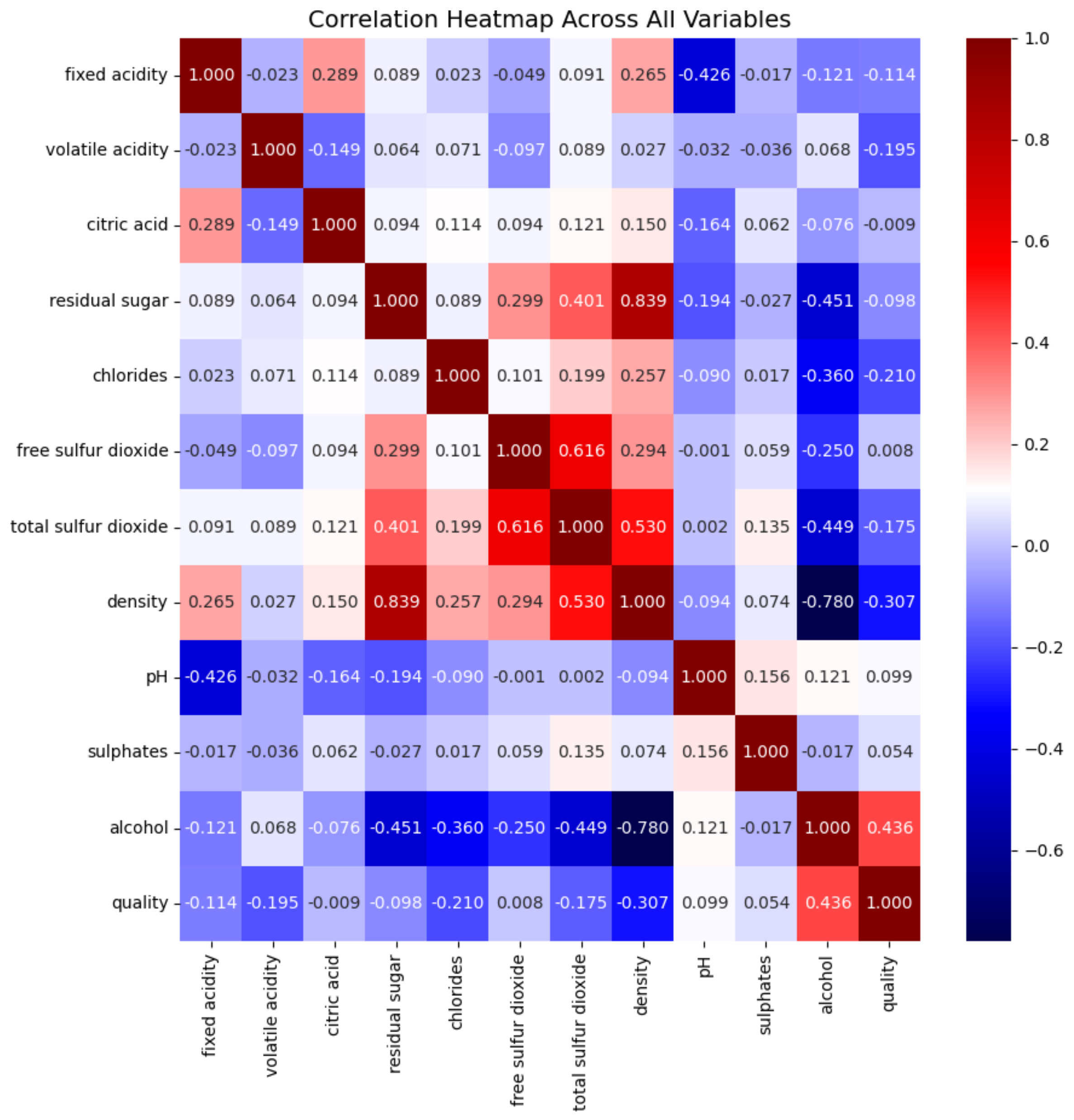


Figure 2: Histogram of Response Variable: Quality

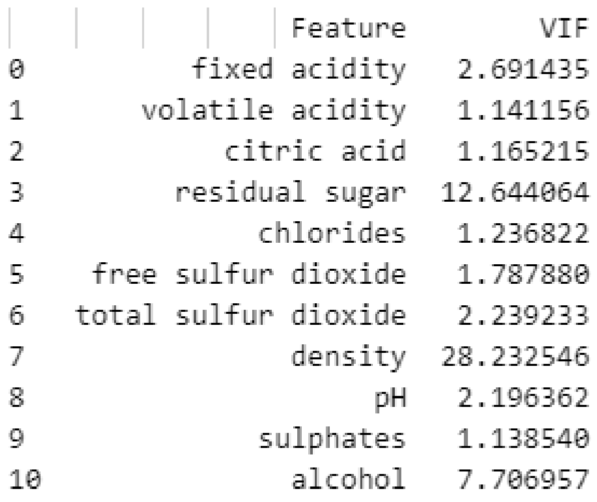
I also created a correlation heatmap to explore the correlation between all variables. As you can see in Figure 3 below, certain variables have strong correlations between each other. Some examples of variable pairs with strong correlations are 'alcohol' and 'density', 'residual sugar' and 'density', and 'free sulfur dioxide' and 'total sulfur dioxide'.



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000	-0.023	0.289	0.089	0.023	-0.049	0.091	0.265	-0.426	-0.017	-0.121	-0.114
volatile acidity	-0.023	1.000	-0.149	0.064	0.071	-0.097	0.089	0.027	-0.032	-0.036	0.068	-0.195
citric acid	0.289	-0.149	1.000	0.094	0.114	0.094	0.121	0.150	-0.164	0.062	-0.076	-0.009
residual sugar	0.089	0.064	0.094	1.000	0.089	0.299	0.401	0.839	-0.194	-0.027	-0.451	-0.098
chlorides	0.023	0.071	0.114	0.089	1.000	0.101	0.199	0.257	-0.090	0.017	-0.360	-0.210
free sulfur dioxide	-0.049	-0.097	0.094	0.299	0.101	1.000	0.616	0.294	-0.001	0.059	-0.250	0.008
total sulfur dioxide	0.091	0.089	0.121	0.401	0.199	0.616	1.000	0.530	0.002	0.135	-0.449	-0.175
density	0.265	0.027	0.150	0.839	0.257	0.294	0.530	1.000	-0.094	0.074	-0.780	-0.307
pH	-0.426	-0.032	-0.164	-0.194	-0.090	-0.001	0.002	-0.094	1.000	0.156	0.121	0.099
sulphates	-0.017	-0.036	0.062	-0.027	0.017	0.059	0.135	0.074	0.156	1.000	-0.017	0.054
alcohol	-0.121	0.068	-0.076	-0.451	-0.360	-0.250	-0.449	-0.780	0.121	-0.017	1.000	0.436
quality	-0.114	-0.195	-0.009	-0.098	-0.210	0.008	-0.175	-0.307	0.099	0.054	0.436	1.000

Figure 3: Correlation Heatmap of All Variables

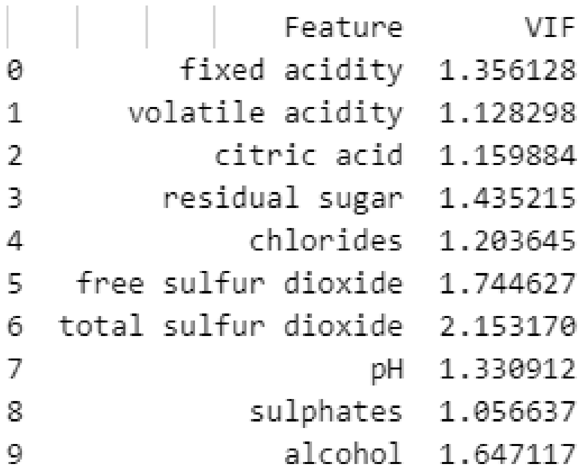
In order to explore if there was any multicollinearity between the predictor variables, I decided to do a VIF analysis. Please see Figure 4 below for the VIF results. As you can see 'density', 'residual sugar', and 'alcohol' had the highest VIF values respectively.



```
                  Feature        VIF
0           fixed acidity   2.691435
1        volatile acidity   1.141156
2             citric acid   1.165215
3          residual sugar  12.644064
4               chlorides   1.236822
5     free sulfur dioxide   1.787880
6    total sulfur dioxide   2.239233
7                 density  28.232546
8                      pH   2.196362
9               sulphates   1.138540
10                alcohol   7.706957
```

Figure 4: VIF Results

In exploring the correlation plot in Figure 3, I noticed that both 'residual sugar' and 'alcohol' were highly correlated with 'density'. Therefore, I hypothesized that removing 'density' would remove multicollinearity (generally, a VIF value less than 5 is considered okay). This did exactly what I expected and removed the multicollinearity issues in the data. Please see Figure 5 below for the updated VIF values once 'density' was removed from the data set.

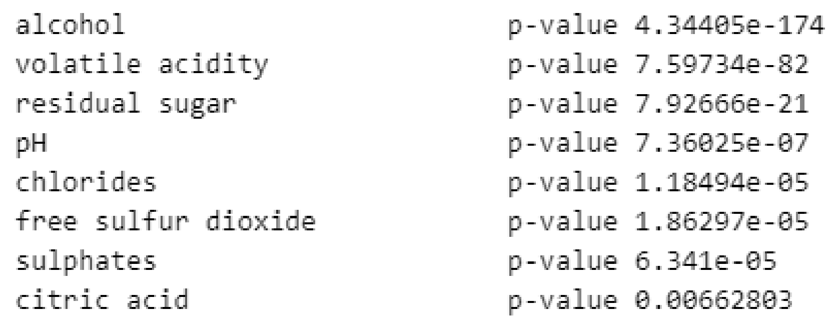


```
                 Feature       VIF
0          fixed acidity  1.356128
1       volatile acidity  1.128298
2            citric acid  1.159884
3         residual sugar  1.435215
4              chlorides  1.203645
5    free sulfur dioxide  1.744627
6   total sulfur dioxide  2.153170
7                     pH  1.330912
8              sulphates  1.056637
9                alcohol  1.647117
```

Figure 5: VIF Results After Removing 'Density'

Then, I removed outliers from the data. I looped through all of the predictor variables (other than 'density'), and removed the observation if the z-score was greater than 3 (meaning the value was more than 3 standard deviations from the mean). Then, I standardized all of the feature variables. In order to create effective models using this data, I decided I needed to balance the data. As you can see from the histogram of the original data set in Figure 2, the data set is very unbalanced. There are no or very few observations at the extremes, and many observations in the middle. While I attempted to use oversampling at first, certain classes had so few observations (for example, the rating 9 only had 5 observations before removing outliers) that I determined this was not an effective approach as it would cause overfitting. Therefore, I decided to group the ratings into two classes: 0 for bad, and 1 for good. I re-classed any observations rated 5 and below as bad, and any observations rated 6 and above as good. After splitting my data into training and test sets using an 80-20 split, I was left with a training set with 1200 observations in class 0 and 2404 observations in class 1. From there, I decided it would be appropriate to oversample using SMOTE (synthetic minority over-sampling technique) to balance the training data set. SMOTE is different from regular oversampling in that it creates synthetic data points instead of duplicates of the existing data. This helps to prevent overfitting. The final training set had 2404 observations in both classes. I did not oversample the test set as this can lead to overfitting and artificially strong performance metrics.

I also decided to use forward and backward stepwise variable selection to choose the features that are most predictive of the response variable 'quality' using p-values of the coefficients as the decision criteria. The most predictive features, in order, were 'alcohol', 'volatile acidity', 'residual sugar', 'pH', 'chlorides', 'free sulphur dioxide', 'sulphates', and 'citric acid'. Please see Figure 6 for the full results.

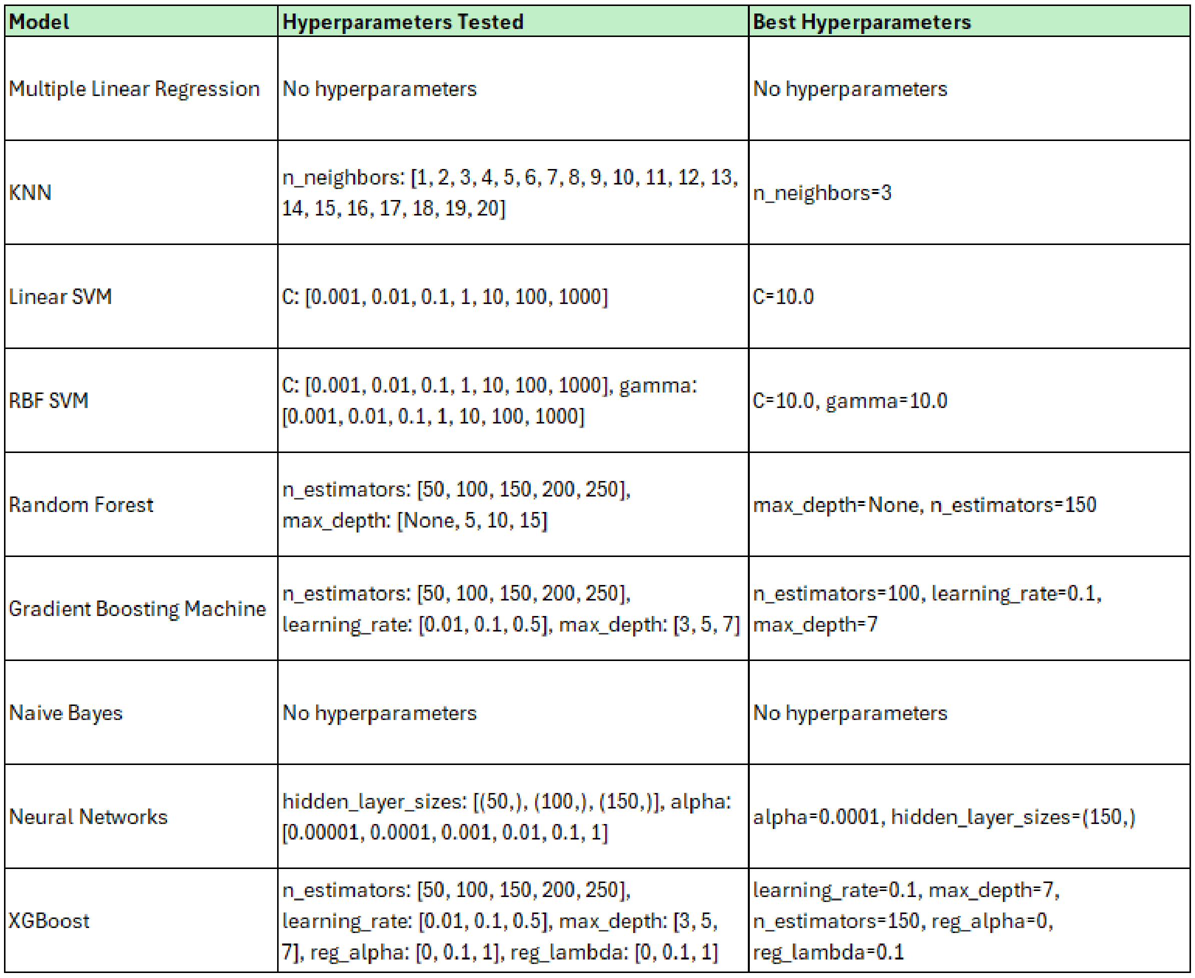


```
alcohol                         p-value 4.34405e-174
volatile acidity                p-value 7.59734e-82
residual sugar                  p-value 7.92666e-21
pH                              p-value 7.36025e-07
chlorides                       p-value 1.18494e-05
free sulfur dioxide             p-value 1.86297e-05
sulphates                       p-value 6.341e-05
citric acid                     p-value 0.00662803
```

Figure 6: Stepwise Variable Selection Results

4 Methodology

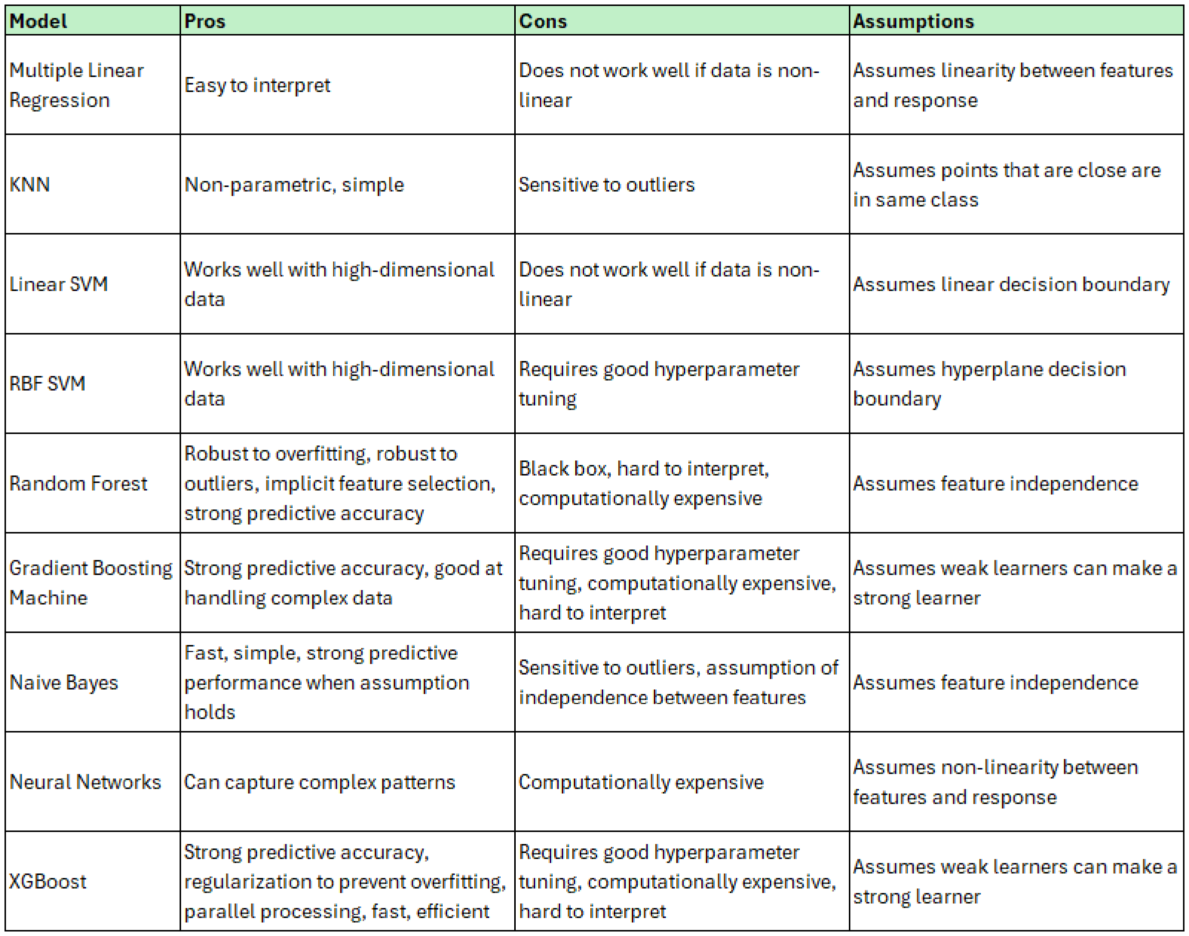
As mentioned previously, my goal for this project was to find the features that most affect white wine quality and come up with the best model to predict white wine quality. The coding language I used for all of my analysis was Python. For this project, I decided to test the following models: multiple linear regression, KNN, Linear SVM, RBF SVM, random forest, gradient boosting machine, naive bayes, neural networks, and XGBoost. First, I used 5-fold cross validation to perform hyperparameter tuning. Please see Figure 7 below for the hyperparameters tested for each model, as well as the hyperparameters I found to be best.



Model	Hyperparameters Tested	Best Hyperparameters
Multiple Linear Regression	No hyperparameters	No hyperparameters
KNN	n_neighbors: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]	n_neighbors=3
Linear SVM	C: [0.001, 0.01, 0.1, 1, 10, 100, 1000]	C=10.0
RBF SVM	C: [0.001, 0.01, 0.1, 1, 10, 100, 1000], gamma: [0.001, 0.01, 0.1, 1, 10, 100, 1000]	C=10.0, gamma=10.0
Random Forest	n_estimators: [50, 100, 150, 200, 250], max_depth: [None, 5, 10, 15]	max_depth=None, n_estimators=150
Gradient Boosting Machine	n_estimators: [50, 100, 150, 200, 250], learning_rate: [0.01, 0.1, 0.5], max_depth: [3, 5, 7]	n_estimators=100, learning_rate=0.1, max_depth=7
Naive Bayes	No hyperparameters	No hyperparameters
Neural Networks	hidden_layer_sizes: [(50,), (100,), (150,)], alpha: [0.00001, 0.0001, 0.001, 0.01, 0.1, 1]	alpha=0.0001, hidden_layer_sizes=(150,)
XGBoost	n_estimators: [50, 100, 150, 200, 250], learning_rate: [0.01, 0.1, 0.5], max_depth: [3, 5, 7], reg_alpha: [0, 0.1, 1], reg_lambda: [0, 0.1, 1]	learning_rate=0.1, max_depth=7, n_estimators=150, reg_alpha=0, reg_lambda=0.1

Figure 7: Hyperparameters Tested Within Each Model

In Figure 8 below, I have listed the pros, cons, and assumptions for each model. Before doing my analysis, I hypothesized that the best performing models would be gradient boosting machine, XGBoost, and random forest as they are all ensemble methods known to have better predictive accuracy than baseline models. Gradient boosting machine and XGBoost are boosting methods which means we iteratively improve on the weak learners by adjusting their weights [4]. Random forest is a bagging method meaning we perform an average of the weak learners [5].

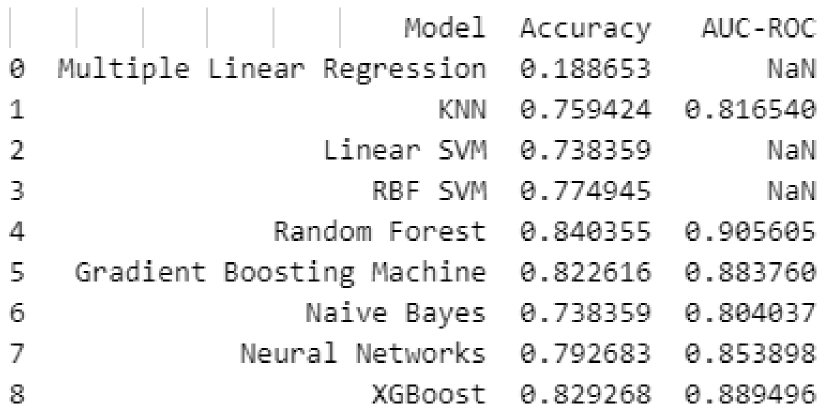


Model	Pros	Cons	Assumptions
Multiple Linear Regression	Easy to interpret	Does not work well if data is non-linear	Assumes linearity between features and response
KNN	Non-parametric, simple	Sensitive to outliers	Assumes points that are close are in same class
Linear SVM	Works well with high-dimensional data	Does not work well if data is non-linear	Assumes linear decision boundary
RBF SVM	Works well with high-dimensional data	Requires good hyperparameter tuning	Assumes hyperplane decision boundary
Random Forest	Robust to overfitting, robust to outliers, implicit feature selection, strong predictive accuracy	Black box, hard to interpret, computationally expensive	Assumes feature independence
Gradient Boosting Machine	Strong predictive accuracy, good at handling complex data	Requires good hyperparameter tuning, computationally expensive, hard to interpret	Assumes weak learners can make a strong learner
Naive Bayes	Fast, simple, strong predictive performance when assumption holds	Sensitive to outliers, assumption of independence between features	Assumes feature independence
Neural Networks	Can capture complex patterns	Computationally expensive	Assumes non-linearity between features and response
XGBoost	Strong predictive accuracy, regularization to prevent overfitting, parallel processing, fast, efficient	Requires good hyperparameter tuning, computationally expensive, hard to interpret	Assumes weak learners can make a strong learner

Figure 8: Pros, Cons, and Assumptions by Model

5 Evaluation and Final Results

Once I found the optimal hyperparameters for each model, I used Monte Carlo cross validation across 50 iterations to make a robust decision as to which model performed best. For the models that do not implicitly perform feature selection (multiple linear regression, KNN, naive bayes, and neural networks), I used only the features chosen by the stepwise variable selection I performed earlier. Please see Figure 9 below for the average test accuracies across 50 Monte Carlo iterations as well as the AUC-ROC values. After evaluating my models, I also attempted to perform PCA for dimensionality reduction and re-evaluated my models to see if this improved model quality. However, PCA actually reduced predictive accuracy, so I removed PCA from my process.



```
                           Model  Accuracy   AUC-ROC
0  Multiple Linear Regression  0.188653       NaN
1                         KNN  0.759424  0.816540
2                  Linear SVM  0.738359       NaN
3                     RBF SVM  0.774945       NaN
4               Random Forest  0.840355  0.905605
5   Gradient Boosting Machine  0.822616  0.883760
6                 Naive Bayes  0.738359  0.804037
7             Neural Networks  0.792683  0.853898
8                     XGBoost  0.829268  0.889496
```

Figure 9: Average Test Accuracies and AUC-ROC Values by Model

As I hypothesized, the best performing models were the ensemble methods: random forest, XGBoost, and gradient boosting machine respectively. While these models had strong performance, I did not stop here. In order to test if I could make improvements to these models, I experimented with stacking using combinations of many models, keeping in mind that using a diverse set of models usually results in better performance. Stacking is an ensemble method that combines predictions from multiple models (that can include other ensemble methods) by using their results as features for a meta model. The meta model then makes the final predictions. After thorough trial and error, I found that the best results came from stacking the random forest and XGBoost models, with the final meta model itself being an XGBoost model as well. Some other combinations of models I tried are random forest, XGBoost, RBF SVM, and neural networks, as well RBF SVM, neural networks, and KNN, experimenting with different meta models like logistic regression, but none of these combinations performed as well as my final stacking model. Please see Figure 10 below for a simple visual representation of my stacking model.

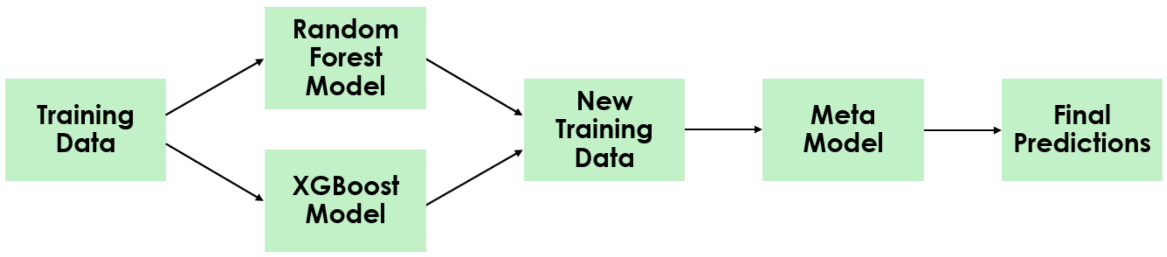


Figure 10: Diagram of Stacking Model

I performed another round of hyperparameter tuning for the stacking model and found the best parameters for random forest within the stacking model to be max_depth=20 and n_estimators=250. The best parameters for XGBoost within the stacking model were learning_rate=0.1, max_depth=7, n_estimators=250, reg_alpha=0, and reg_lambda=0.1. The best parameters for the meta model were learning_rate=0.1, max_depth=3, reg_alpha=0.1, and reg_lambda=0.1. Please see Figures 11, 12, and 13 below for the confusion matrices for the random forest model, the XGBoost model, and the stacking model. In Figure 14, I have listed the average test accuracies and AUC-ROC values for the random forest, XGBoost, and stacking models in order to summarize the performance differences between the stacking model and its base models.

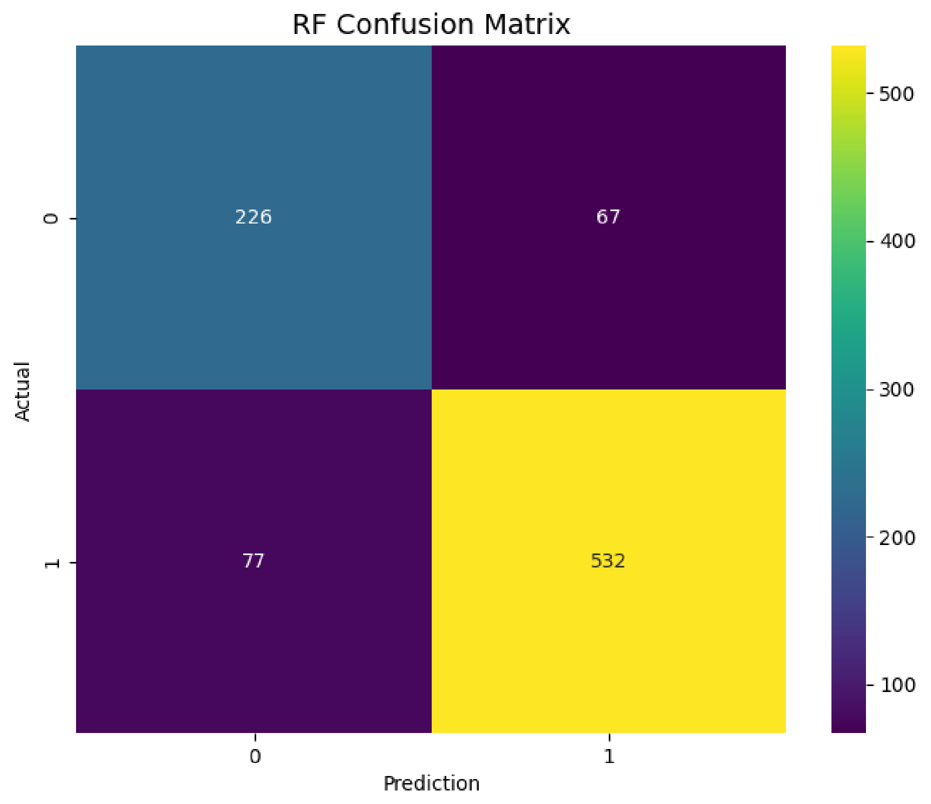


Figure 11: Random Forest Confusion Matrix

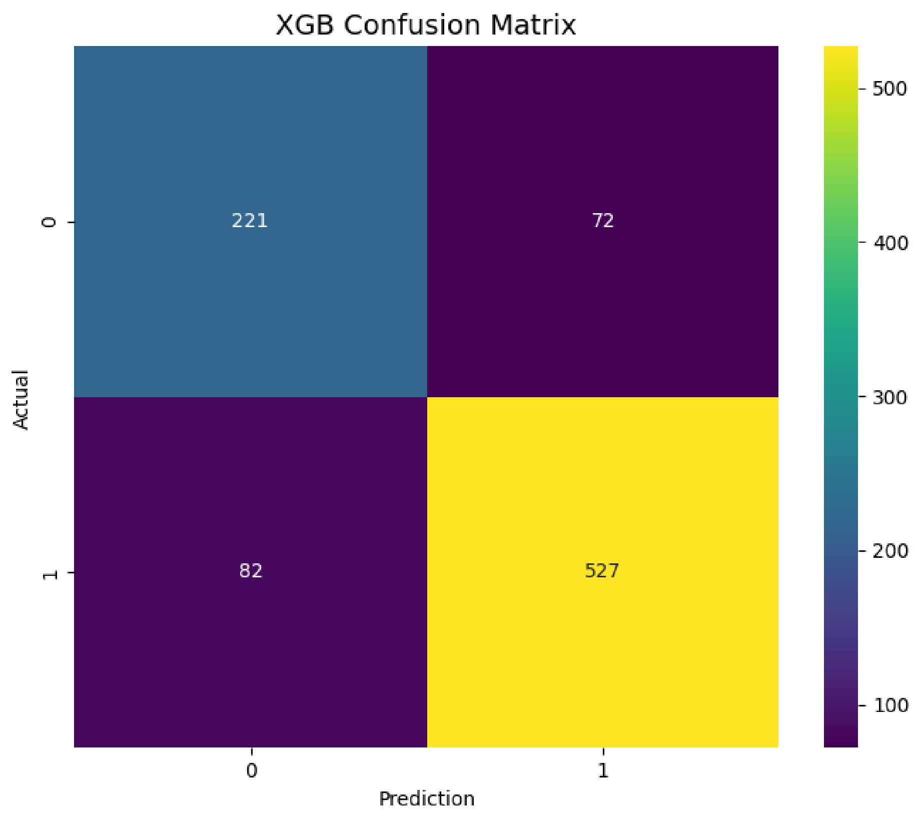


Figure 12: XGBoost Confusion Matrix

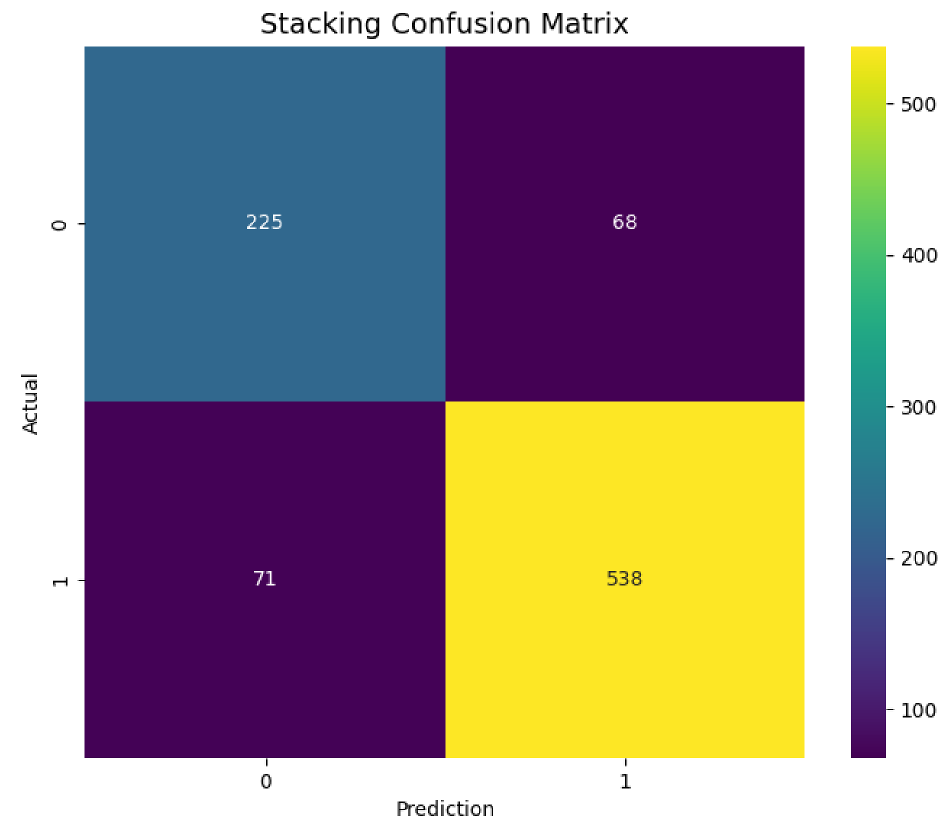
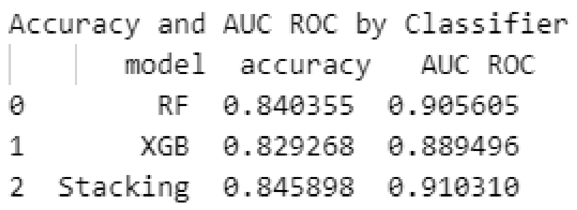


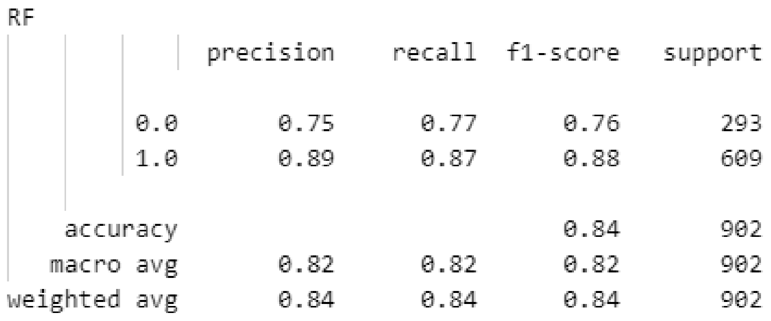
Figure 13: Stacking Confusion Matrix



```
Accuracy and AUC ROC by Classifier
         model  accuracy   AUC ROC
0           RF  0.840355  0.905605
1          XGB  0.829268  0.889496
2     Stacking  0.845898  0.910310
```

Figure 14: Accuracy and AUC-ROC by Classifier

As you can see, stacking outperformed both of its base models that were used to create it, random forest and XGBoost, which were the best models from the initial set of models. I have also included the precision, recall, and f1-scores for the random forest, XGBoost, and stacking models in Figures 15, 16, and 17 below. Stacking had the best performance when looking at these measures as well, but the performance improvement compared to the base ensemble models was very modest.

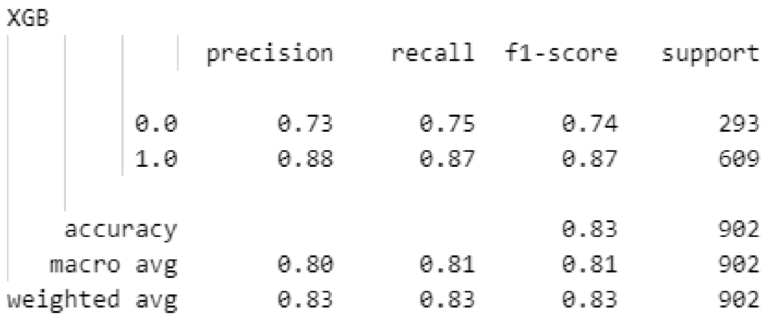


```
RF
              precision    recall  f1-score   support

         0.0       0.75      0.77      0.76       293
         1.0       0.89      0.87      0.88       609

    accuracy                           0.84       902
   macro avg       0.82      0.82      0.82       902
weighted avg       0.84      0.84      0.84       902
```

Figure 15: Random Forest Precision, Recall, and F-1 Score

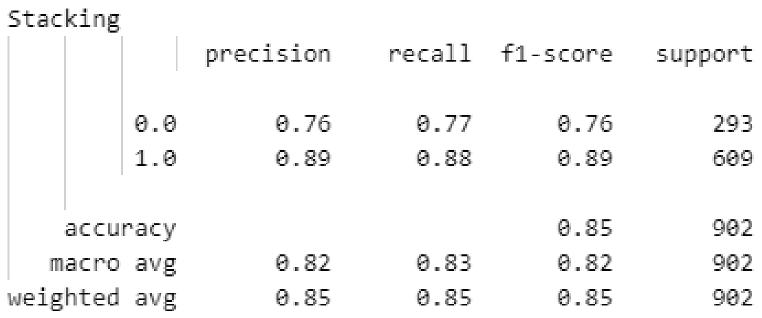


```
XGB
              precision    recall  f1-score   support

         0.0       0.73      0.75      0.74       293
         1.0       0.88      0.87      0.87       609

    accuracy                           0.83       902
   macro avg       0.80      0.81      0.81       902
weighted avg       0.83      0.83      0.83       902
```

Figure 16: XGBoost Precision, Recall, and F-1 Score



```
Stacking
              precision    recall  f1-score   support

         0.0       0.76      0.77      0.76       293
         1.0       0.89      0.88      0.89       609

    accuracy                           0.85       902
   macro avg       0.82      0.83      0.82       902
weighted avg       0.85      0.85      0.85       902
```

Figure 17: Stacking Precision, Recall, and F-1 Score

In addition to stacking, I also experimented with another method of combining models: blending. Blending is very similar to stacking but it has a key difference. In stacking, out-of-fold predictions, often from k-fold cross validation, are used to train the meta model. In blending, validation set predictions are used to train the meta model. However, as I experimented with different model combinations and hyperparameters, I was not able to find a blending model that outperformed my stacking model. From there, I attempted to stack my stacking model and blending model. However, once again, it did not outperform my original stacking model. This analysis has exemplified how ensemble methods generally perform better than baseline methods, and that their performance can be optimized with good hyperparameter tuning. It has also exemplified that it is possible to improve performance even further by combining some of the best models into stacking or blending models. While I tested a fixed number of model combinations, there are endless possibilities when it comes to finding creative ways to combine models, and it is very possible that there is a better stacking or blending model that could optimize performance even further.

6 Conclusion

There were two main objectives in this project: create a strong predictive model and find the features most predictive of wine quality. I believe the first objective was met because my stacking model effectively predicts if a wine is good or bad using its physicochemical properties. However, I believe that there are two main reasons why it is difficult to create a model with better predictive accuracy than around 85%. The first reason is that wine quality is a subjective measure that can vary depending on who is testing it, and other factors such as the tester's mood at the time of testing. The second reason is that in the original data set, there were very few observations with 'quality' values at the extremes of the range. If we could balance the data set through adding more real observations (rather than just oversampling), we may be able to improve on predictive accuracy and even predict using the 0 through 10 scale rather than the binary classes. With an updated data set, it is important to note that we would need to re-tune the hyperparameters and re-evaluate which models to include in the stacking model. We could also re-evaluate different combinations for blending models. If any of these models were to be implemented for real-world use, it might also make sense to add more features to the data set as well.

I believe the second objective was met as well because I was successfully able to identify the features that are most predictive of wine quality. According to the stepwise variable selection I performed earlier, the features that are most predictive of wine quality are 'alcohol', 'volatile acidity', 'residual sugar', 'pH', 'chlorides', 'free sulphur dioxide', 'sulphates', and 'citric acid' respectively. According to my random forest model, the most important features are 'alcohol', 'volatile acidity', 'chlorides', 'free sulphur dioxide', 'citric acid', 'residual sugar', 'total sulfur dioxide', and 'pH'. While these results do not align exactly, they do generally align, and it is interesting to see that both results indicate that 'alcohol' and 'volatile acidity' are the most important features. Knowing this can help wine producers focus on a few elements to increase the quality of their wines.

In summary, in order for wine producers to best predict the quality of their wines, the best approach would be to run a physicochemical test and input those numbers into my stacking model. This gives them the ability to easily assess the quality of their existing and future wines and help them develop pricing strategies. Since physicochemical properties of wines are not readily available for consumers, I believe that this analysis will be more beneficial for wine producers than wine consumers. For those who want to use data to produce better quality wines, they can focus on the physicochemical features most predictive of wine quality. If wine makers use this analysis to their full advantage, I believe that they will be able to drive significantly higher sales and improve the economy of Portugal. Additionally, since the features of this data set can be extracted from all wines, I believe that this analysis can be scaled for wines all around the world, improving the overall quality of wines worldwide and improving the sales of wines globally. There is also potential for the analysis to be scaled to different types of alcoholic drinks such as whiskey, tequila, bourbon, beer, and more which has the potential to have a significant positive impact the global economy.

References

[1] Et Al. *Discovery science : 12th international conference, DS 2009, Porto, Portugal, October 3-5, 2009 : proceedings.* Springer, Cop, Berlin ; New York, 2009.

[2] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, Nov 2009.

[3] Wines of Portugal. History - winesofportugal.com, 2010.

[4] Yao Xie Pd.D. Isye 6740 - lecture 14: Boosting, 2020. Accessed: 4/5/2024.

[5] Yao Xie Pd.D. Isye 6740 - lecture 15: Decision tree and random forest, 2020. Accessed: 4/5/2024.