Report of Linear Regression and Classification

Introduction

Using a linear regression model as a basis to predict the quality of red wine yielded some results that proved a better type of model is needed for this complex data. So the next step was to move to a classification model and try to accurately predict the quality of the wine that way.

Dataset Description

The dataset used is a red wine dataset, that has eleven distinct features of wine. Those eleven features are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. There are no missing values in this dataset, and during training, the quality category is removed and used to compare the predicted results to the actual results. There are 1599 different wines in this dataset, and none are missing any values in any of the categories.

Experimental Setup

I first set up a way to simplify the training of the model and improve the speed at which training occurs. I did this by using a correlation matrix of the eleven features to determine the five most important ones for determining quality. Those five features I used to create a new dataset to train the model with and compared the resulting quality predictions to the actual predictions. I split the dataset into two separate sets, and training (80% of the data) and testing set (remaining 20% of the data). Misunderstanding k-fold cross validation, I incorporated a 5-fold version and then increased it to a 10-fold version. Learning afterward that k-fold validation results in better estimates of performance and not better performance. I kept the k-fold in for the increase estimates. Dropping back to 5-fold cross validation for the classification due to getting a warning "UserWarning: The least populated class in y has only 9 members, which is less than n_splits=10," when it was set to 10-fold.

Results

Running the linear regression program, the model produced a model that was relatively poor in my opinion. Attempting to increase performance I tried using the scikit-learn library and its preprocessing package to us Polynomial Features for the model fit more complex relationships between the top five selected features. The results improved, though very little. Figure 1 shows the Mean Squared Error (MSE), and the R2 Scores for the two models (MSE lower the better, R2 larger the better). Figure 3 on the next page has the graphs of how the model predicted the quality (red dots) versus what the quality actually was (blue line).

Figure 1

Basic Model:

- MSE: 0.45000046789225695
- R2 Score: 0.3114056110795572

Model using Polynomial Features:

- MSE: 0.43618220200679547
- R2 Score: 0.3325504343236473

With such a little increase in performance, I moved on to a classification model here, and arbitrarily chose Random Forest Classifier. The random state of the Random Forest is

set to 42, because I wanted an integer to make the results reproducible and “The Answer to the Ultimate Question of Life, the Universe, and Everything is 42.” (The Hitchhiker’s Guide to the Galaxy by Douglas Adams). The results from this run were better, and I originally tried to improve the results even further by using some hyperparameter optimizations (HPO). I removed those from the code here so I could focus on them more in the hyperparameter optimization exercise. Figure 2 contains the accuracy of the classifier model. On the next page, Figure 4 shows the graph of the results of the classifier (Same graph style as Figure 2), and Figure 5 is the Confusion Matrix for the results.

Figure 2
Random Forest Classifier Model:
Accuracy: 0.625

Figure 3

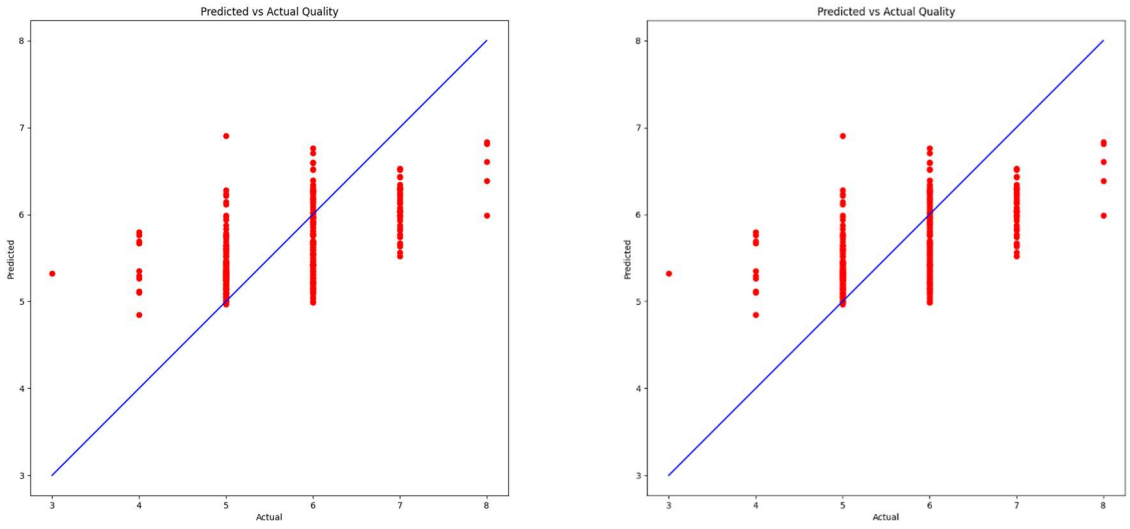

Figure 4

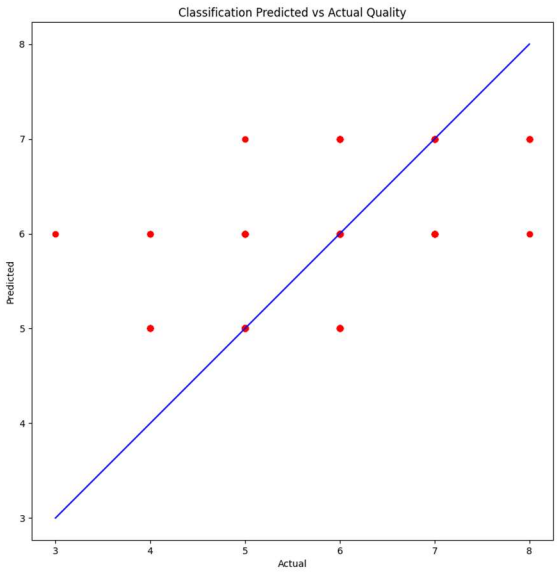

Figure 5