Report of ML Algorithm Selection

Introduction

The problem we have as a society is being able to accurately tell if a wine is of good quality or not. The normal method to fix this problem would be enduring the rigorous and tedious process of learning how to tell wines apart and how to determine their quality based on their labels. Instead of trying to do this, I created a machine learning algorithm to aid in predicting the quality of a wine.

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 three separate datasets, a training set, a validation set, and a test set, with a 60/20/20 split respectively. There exist many types of machine learning models and selecting one proved a difficult decision. Instead, I had a model selection process occur. This was performed by creating a dictionary of pipelines, that each uses a preprocessing step to standardize the features of the data. There are a few reasons this was done, the first is because some algorithms assume that all features are centered around zero. Another important reason is because this scaling improves gradient descent making the features converge more quickly. There are four pipelines in the dictionary, each to produce a different type of model: Random Forest Classifier, Support Vector Classifier (SVC), a Neural Network specifically a Multilayer Perceptron Classifier, and K-Nearest Neighbors (KNN). The models were trained on the training set, and the validation set was used to compare how well they could predict the wine quality. Finally, the best model was run with the test set to get an unbiased accuracy estimate.

Results

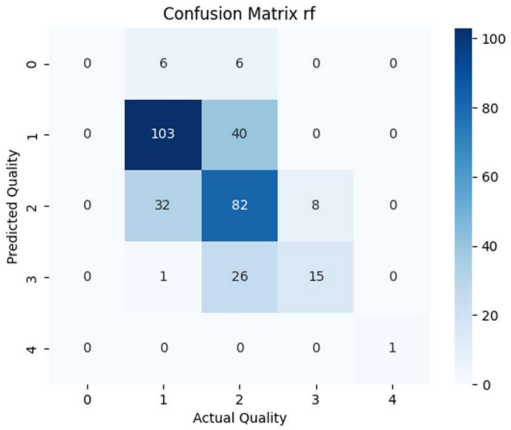
The results from running the pipelines of models to compare results returned Random Forest as the one with the highest accuracy. However, there was a potential issue. When the Random Forest model was then run on the test set to get an unbiased estimate, it had a significantly lower estimate. Due to that over 5% drop, potentially means that there is some overfitting going on. This poses two problems, 1) Random Forest may not actually be the most accurate, and 2) This means that the previous exercise where the Random Forest is the only model, I ran also likely has some overfitting occurring. I ran the program several more times

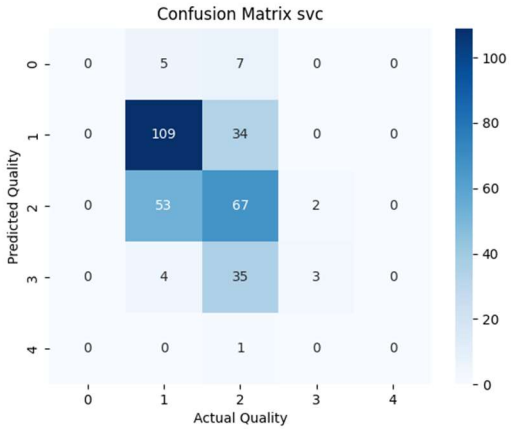
Accuracy Estimates:	
Random Forest:	0.6375
SVC:	0.559375
Neural Network:	0.571875
KNN:	0.58125
Test Set Accuracy:	
Random Forest:	0.58125

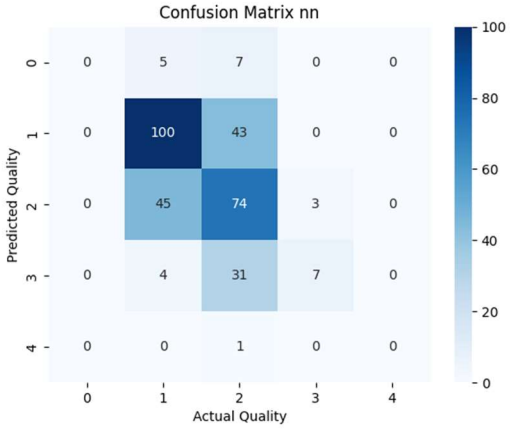
and every time it was the similar results. Random Forest was the highest and its accuracy estimate ranged from 0.615625 to 0. 0.646875while its accuracy on the test set ranged from 0.575 to 0.609375, with an average difference of 0.053125. My first action attempting to fix this was to set some parameters on the Random Forest model. I set the max number of trees to 100, the max depth it would search to 10, and set its random state to 42, and ran a few more tests with this. Both accuracies dropped on the Random Forest. The Accuracy of the Random Forest dropped a few percentages but never dipped below 0.6. The test set accuracy dropped more so, to on average dropping approximately 3.7%. Not wanting to adjust the hyperparameters too much I removed the changes I made to Random Forest. I found that increasing the size of the training set could lower the risk of overfitting. With this information, I adjusted my data split to be 80/10/10, training, validation, test. The results produced from these changes proved to fix any overfitting problem I was having. Random Forest was still beating out the other models in accuracy, and now the test set accuracy was very near to the validation accuracy. I could not have asked for a better fix, because the split used in the previous exercises was also 80% for the training data, lowering the likelihood I had previously been overfitting.

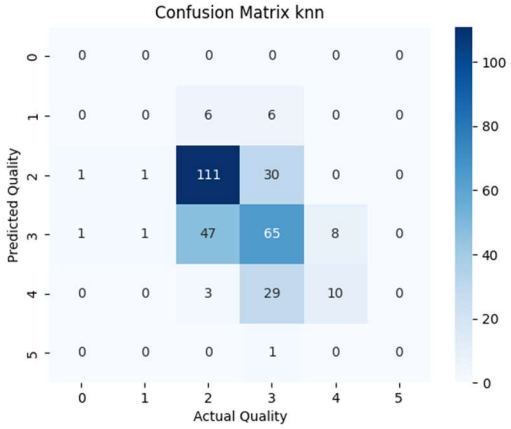
Updated Accuracy Estimates:	
Random Forest:	0.6125
SVC:	0.54375
Neural Network:	0.54375
KNN:	0.55625
Test Set Accuracy:	
Random Forest:	0.625

On the next page are the confusion matrices for the models when the dataset was split 60/20/20 with the “Best” matrix being the Random Forest on the test set in purple coloring. The page following that one contains the confusion matrices for the models when the dataset was adjusted to be split 80/10/10 and again the “Best” matrix is the Random Forest on the test set in purple coloring. Finally the last page is a graph that shows the models in a scatter plot. The closer to the blue line the more accurate the prediction is, and the colors reflect accuracy. The lighter the blue the more accurate, and as the blue darkens and turns purple it shows a less accurate prediction, with pink being very inaccurate. The graph labeled “Alg Selection Predicted vs Actual Quality 2” is the graph with the adjusted 80/10/10 split.

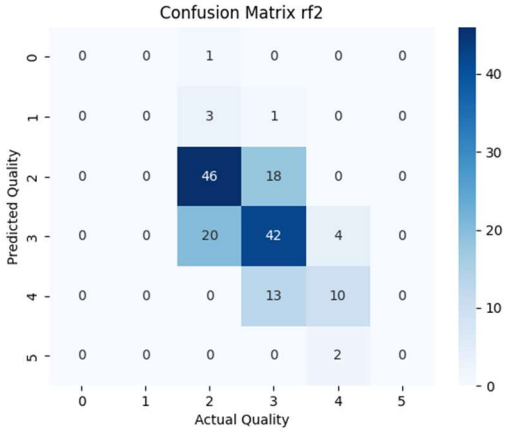

Confusion Matrix rf
Predicted Quality
Actual Quality
0 6 6 0 0
0 103 40 0 0
0 32 82 8 0
0 1 26 15 0
0 0 0 0 1
0 1 2 3 4
100 80 60 40 20 0

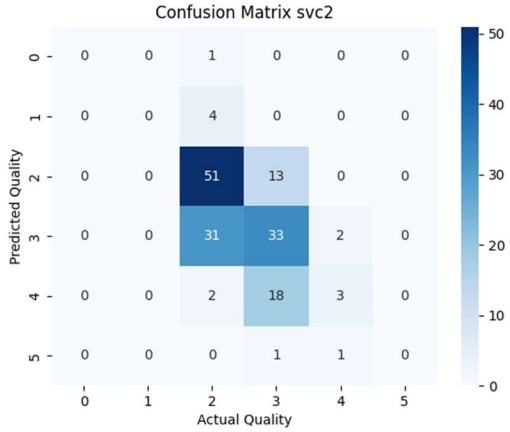

Confusion Matrix svc
Predicted Quality
Actual Quality
0 5 7 0 0
0 109 34 0 0
0 53 67 2 0
0 4 35 3 0
0 0 1 0 0
0 1 2 3 4
100 80 60 40 20 0

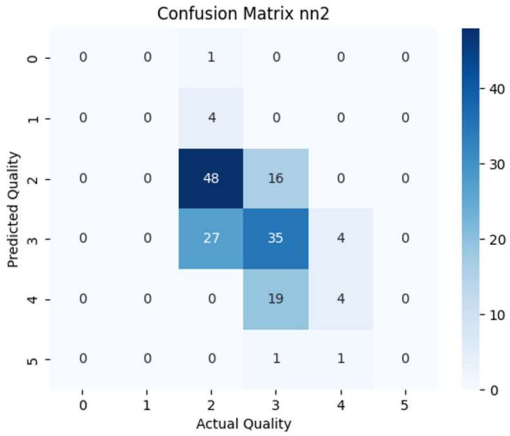

Confusion Matrix nn
Predicted Quality
Actual Quality
0 5 7 0 0
0 100 43 0 0
0 45 74 3 0
0 4 31 7 0
0 0 1 0 0
0 1 2 3 4
100 80 60 40 20 0

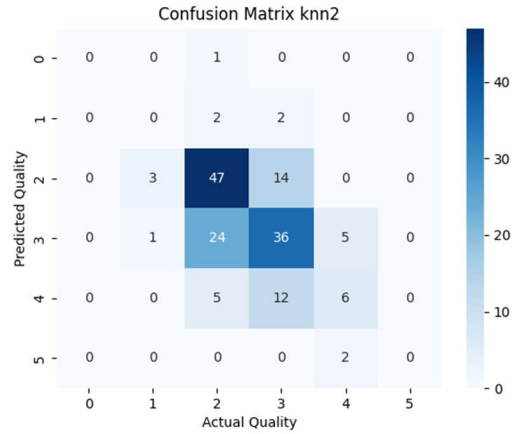

Confusion Matrix knn
Predicted Quality
Actual Quality
0 0 0 0 0 0
0 0 6 6 0 0
1 1 111 30 0 0
1 1 47 65 8 0
0 0 3 29 10 0
0 0 0 1 0 0
0 1 2 3 4 5
100 80 60 40 20 0


Confusion Matrix Best
Predicted Quality
Actual Quality
0 0 1 1 0 0
0 1 5 5 0 0
0 0 99 34 2 0
0 0 41 72 12 1
0 0 0 20 19 0
0 0 0 2 5 0
0 1 2 3 4 5
80 60 40 20 0

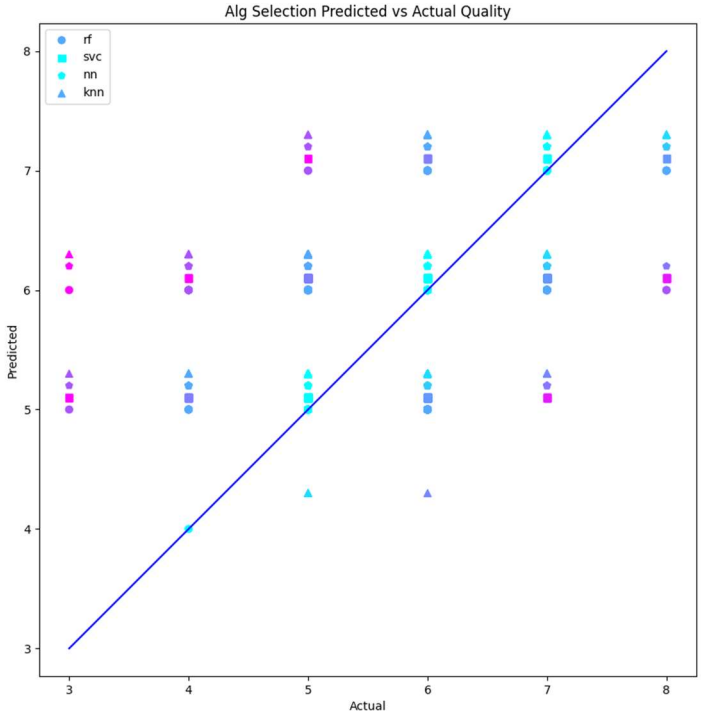

Confusion Matrix rf2
Predicted Quality
Actual Quality
0 0 1 0 0 0
0 0 3 1 0 0
0 0 46 18 0 0
0 0 20 42 4 0
0 0 0 13 10 0
0 0 0 0 2 0
0 1 2 3 4 5
0 10 20 30 40

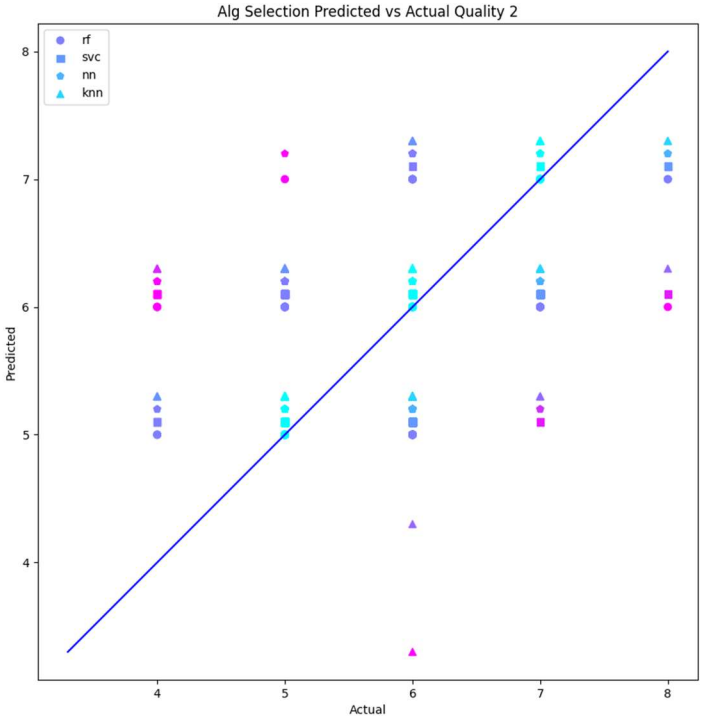

Confusion Matrix svc2
Predicted Quality
Actual Quality
0 0 1 0 0 0
0 0 4 0 0 0
0 0 51 13 0 0
0 0 31 33 2 0
0 0 2 18 3 0
0 0 0 1 1 0
0 1 2 3 4 5
0 10 20 30 40 50


Confusion Matrix nn2
Predicted Quality
Actual Quality
0 0 1 0 0 0
0 0 4 0 0 0
0 0 48 16 0 0
0 0 27 35 4 0
0 0 0 19 4 0
0 0 0 1 1 0
0 1 2 3 4 5
0 10 20 30 40


Confusion Matrix knn2
Predicted Quality
Actual Quality
0 0 1 0 0 0
0 0 2 2 0 0
0 3 47 14 0 0
0 1 24 36 5 0
0 0 5 12 6 0
0 0 0 0 2 0
0 1 2 3 4 5
0 10 20 30 40


Confusion Matrix Best 2
Predicted Quality
Actual Quality
0 3 3 0 0
0 53 12 1 0
0 23 36 7 0
0 0 8 11 0
0 0 1 2 0
0 1 2 3 4
0 10 20 30 40 50


Alg Selection Predicted vs Actual Quality
rf
svc
nn
knn
Predicted
Actual
3
4
5
6
7
8


Alg Selection Predicted vs Actual Quality 2
rf
svc
nn
knn
Predicted
Actual
3
4
5
6
7
8