Blaise Holloway
bhollowa
1583872

CMPUT 466 Final Project – ML Algorithm Analysis on Wine Quality

Public GitHub Repo: https://github.com/blaiseh33/CMPUT466-Final-Project

Introduction and Formulation:

We will attempt to predict the quality of wines based on 11 attributes: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. The output and final attribute is quality on a scale of 0-10. We have 2 datasets found here, one with red wines and the other with white wines. These wines are variants of the Portuguese "Vinho Verde" wine, and the datasets are not ordered or balances meaning that there are many more average scored wines compared to low or high scored ones. There are 1599 data entries for the red wine and 4898 entries for the white wine. We will use linear regression, polynomial regression, and neural net regression to test and analyze the performance of each algorithm for this application. We will use a training-validation-test infrastructure with hyperparameter optimization/tuning. This experiment will also be used to infer about what the most important features of the wine are in getting a better-quality score. I am also interested to see how one can predict quality scores of a wine based on simple chemical and physical data of the wine as this may lead to a conclusion that the quality score is arbitrary compared to this data, and it is all up to personal preference regarding what makes a wine higher quality than others.

Approaches and Baselines:

For linear regression using the sklearn.linear_model we do not have any hyperparameters to manually optimize or tune as it is already done by the model. We will use the default model parameters of {'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'positive': True}. For our polynomial regression, we are once again using the sklearn.linear_model library, however this time we will be including a preprocessing step through the same library called PolynomialFeatures. With this model, we can easily manipulate the degree of the polynomial before passing these new features to our linear model for training, validation, and testing. For the neural net regression model, we can manipulate the hyperparameters {alpha, hidden_layer_sizes}. We will test the accuracy, mean squared error, and mean absolute error for all of the algorithms and determine the best hyperparameters for each model. We will also consider overfitting based on our training error and accuracy compared to our testing error and accuracy. Because the regression models will result in real number prediction values, we will simply round these values to the nearest integer and ensure that these predictions are within the quality range [0-10].

Evaluation:

To evaluate the performance of each algorithm we will use accuracy (percentage of correct quality scores), mean squared error, and mean absolute error. Mean squared error and mean absolute error can be calculated through the sklearn metrics library, whereas I will use my accuracy code from the previous coding assignments. We use the train-validate-test infrastructure to identify the best models for each algorithm through the mean squared error over multiple epochs. The main goal of this experiment is to determine how well these regression algorithms can predict the quality scores of the wines, and these quality scores are given through professional taste testing. With that being said, the quality score may not have a perfect correlation between all or any of the features for the wine, and we can try to determine how much the feature data influences the quality score. To do this, we will also test our predictions for close accuracy where we allow our wine score prediction to be within 1 to the correct quality score. I believe this is a reasonable statistic to observe due to the arbitrary nature of the quality score being given strictly by taste – we cannot know how much each of the features have a direct effect on the taste.

Results:

The results of the experiment are quite interesting to me, and I believe that these results are due to the potential lack of correlation between the feature data and the somewhat arbitrary quality score. We can see in the 2 tables below for each wine type that the accuracy (exact correct quality score predictions) are somewhat poor – all algorithms on both data sets are around 50%. Polynomial regression slightly outperforms the other 2 algorithms for accuracy. Close accuracy gives a much better prediction on the quality as giving a buffer of plus or minus 1 score leads to a greater than 90% accuracy for most algorithms on average. The neural network performs very well with the smaller data set of red wine for close accuracy, whereas the polynomial regression doesn't perform quite as well. Overall, I believe that all 3 algorithms are good potential candidates for predicting the quality of wines based on the 11 given features. With that said however, I do believe that the poor performance of the exact accuracy is likely due to the wine quality score not having a direct correlation to all of the features and data given – the score is influenced by personal preference and taste. A rough estimate of the score using close accuracy can however be used to interpret the quality of these wines. Based on the coefficients/weights of the final model for each algorithm, we can also conclude that the features with the greatest influence on the quality score are volatile acidity, density, citric acid, pH, and alcohol.

White Wine:

	Linear Reg.	Polynomial Reg.	NN Reg.
Mean Squared Error	0.6347	0.6146	0.6971
Mean Absolute Error	0.5256	0.4922	0.5545
Accuracy	0.5267	0.5579	0.5133
Close Accuracy (+-1)	0.9498	0.9565	0.9354

Red Wine:

	Linear Reg.	Polynomial Reg.	NN Reg.
Mean Squared Error	0.7257	0.9096	0.6086
Mean Absolute Error	0.5652	0.5819	0.5083
Accuracy	0.4949	0.5451	0.5384
Close Accuracy (+-1)	0.9565	0.8996	0.9665

References:

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

https://datascience.stackexchange.com/questions/15135/train-test-validation-set-splitting-in-sklearn