Machine Learning Algorithm Selection

Introduction:

For this exercise I am going to test several machine learning algorithms on the white wine quality dataset, I will be attempting to solve this as a regression problem using a number of algorithms using their default parameters and evaluating then using a few different metrics that I will go over in the experimental setup.

Dataset Description:

I opted to use the wine quality dataset[1] for this and picked the white wine dataset because it was the larger of the two with 4898 data entries. The data includes 11 features and a target value called "quality" that that can be assigned 0 to 10.

[1]: https://archive-beta.ics.uci.edu/dataset/186/wine+quality

Experimental Setup:

For the setup I used scikit-learn's[2] machine learning libraries in python to make and fit models. To evaluate performance I have a number of metrics including: accuracy[3], coefficient of determination (r2 score), and absolute error score. I picked these because I was familiar with r2 score already and when looking at ways to score regression models I saw absolute error score and decided that would be a simple way to compare models given that the best score is 1, and anything lower shows inaccuracies.

For coefficient of determination and absolute error score I used scikit-learn's libraries[4] to determine them, for accuracy I ended up looking to see how others were evaluating performance and saw a student named "Michael Elgin" use a formula to determine accuracy[3] by rounding the predicted quality values to the nearest whole number, comparing the values to what they should be and determining accuracy based on the number of matches. I liked this approach and decided to include it as it helped visualize better how well the model was performing.

For the models chosen I went with the ones recommended in the exercise's github repository being linear regression, random forest, decision tree, support vector machine, gradient boosting, and then I decided to also pick nearest neighbor regression. I used default values for all models.

I then split the test data to be 20% of the overall data and began the experiments, starting with seeing how accurate random guesses were using values 4-9. Even though the min and max of the values in quality are 3-9, I ran random guesses with multiple ranges and found 4-9 to give the best results. Afterwards I ran the machine learning algorithms against the data and recorded the performance.

[2]: https://scikit-learn.org/stable/supervised_learning.html

[3]: https://github.com/COSC5557/ml-algorithm-selection-Mikey-E/blob/main/Alg_Selection.ipynb

[4]: https://scikit-learn.org/stable/modules/model_evaluation.html

Results:

The last run of the program produced this output:

Percent accuracy for Randomly Guessing:20.82%

Coefficient of determination for Linear Regression: 0.2829507536603465
Percent accuracy for Linear Regression: 53.06%
Absolute error score for Linear Regression: 0.0868720087726057

Coefficient of determination for Random Forest: 0.5153704381436331
Percent accuracy for Random Forest: 68.78%
Absolute error score for Random Forest: 0.31256535947712416

Coefficient of determination for Decision Tree: 0.0570046219963779
Percent accuracy for Decision Tree: 57.76%
Absolute error score for Decision Tree: 0.1797385620915033

Coefficient of determination for Nearest Neighbor Regression: 0.15591318214990402
Percent accuracy for Nearest Neighbor Regression: 48.27%
Absolute error score for Nearest Neighbor Regression: 0.004248366013071991

Coefficient of determination for Support Vector Machine: 0.0027029218585289394
Percent accuracy for Support Vector Machine: 46.43%
Absolute error score for Support Vector Machine: 0.021241830065359513

Coefficient of determination for Gradient Boosting Regression: 0.38981935230004705
Percent accuracy for Gradient Boosting Regression: 58.67%
Absolute error score for Gradient Boosting Regression: 0.15496079712684707

Random forest performed the best here with the highest accuracy, highest coefficient of determination and highest absolute error score. The worst performance came from the support vector regression, this may be due to not tuning the model properly and only using default values.

Nearest neighbor regression also seemed to perform better when I limited the neighbor count to 6, but to keep consistency I decided for this exercise to keep it at default values.

All of these are better than randomly guessing, but random forest performed much better than other regression models that I tested all with default parameters.

Code:

Is in the repository under main.py