ML Algorithm Selection Report

Soudabeh Bolouri

Introduction:

In this exercise, we present the simplest version of Automated Machine Learning aimed at choose the best type of machine learning model. Our goal is to predict the quality of white wine based on different features. To accomplish this, we use several ML algorithms and utilize a Wine Quality dataset. The primary objective is to find the most appropriate algorithm for wine quality(target) prediction.

Dataset Description:

The dataset that we utilized in this exercise is the "Wine Quality" dataset, precisely the "white" wine version. It contains data on different features of white wines, including fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, PH, sulphates, and alcohol. The dataset includes a totality of 4,898 rows and 12 features. The "quality" of the wine is the target variable, which we desire to predict. Also, we did not find any missing values in the dataset.

Experimental Setup:

We use the Python programming language for this experiment. First we split the dataset using the "split_data" function into training and testing sets with an 80-20 split ratio and employed a fixed random seed (random_state=42) for reproducibility. In order to design machine learning models or conduct experiments, consistency and reproducibility are crucial. The result should be the same if we rerun the same code with the same data, ensuring that we don't have any random factors influencing our results. Then, we separated the features and the target variable (quality) for both the training and testing sets using the "separate_features_target" function. This ensures machine learning algorithms can be trained on the features to predict the target variable.
Then we assume a list of machine learning algorithms, each initialized with their default hyperparameters. Each element in this list is a tuple containing the algorithm's name and the model. Algorithms that we considered in this exercise are as follows:

- Random Forest; with a fixed random seed of 42
- Support Vector Machine (SVM)
- Linear Regression
- Logistic Regression; with a fixed maximum iteration of 10,000

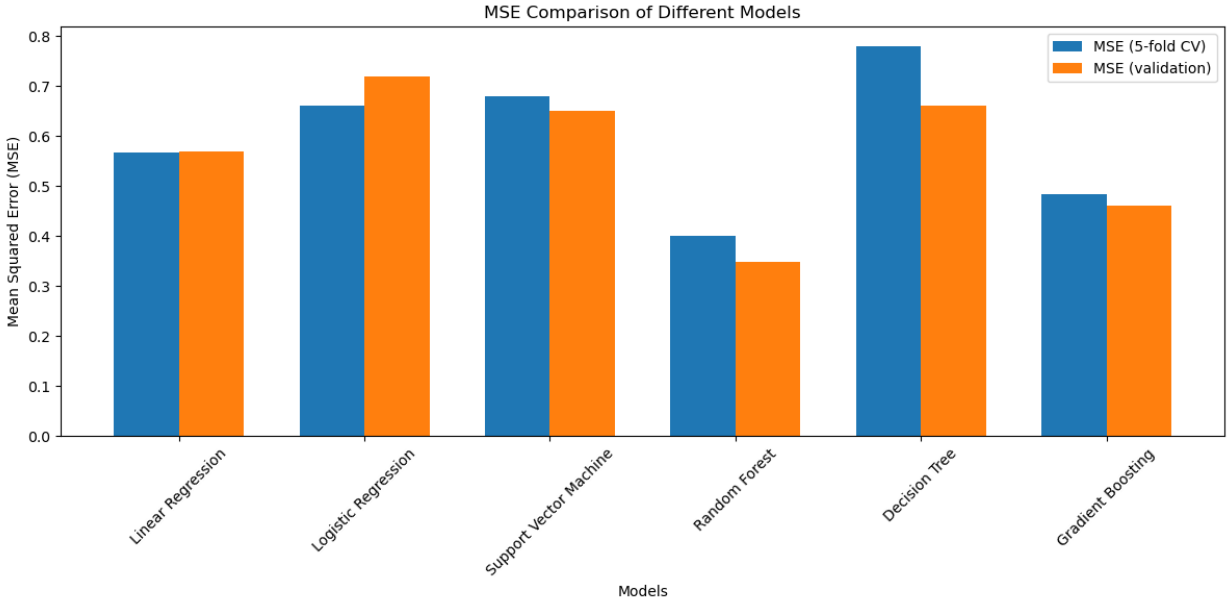
- Decision Tree; with a fixed random seed of 42
- Gradient Boosting; with a fixed random seed of 42

To assess the performance of each algorithm, we utilize 5-fold cross-validation using scikit-learn website. We used the negative mean squared error (neg_mean_squared_error) as the scoring metric, a common choice for regression problems. This metric evaluates how sufficiently the expected wine quality values align with the actual values.

In the next step we fit the machine learning model to the entire training dataset. It trains the model on all available training data. Finally, we predict on the test data and compute MSE for the validation set. We calculated the MSE between the predicted values and the actual target values on the test dataset to quantify the prediction accuracy.

Results:

In the chart below, we can see that Random Forest is the most efficient algorithm with CV MSE of **0.40** and validation MSE of **0.34**. Below the chart, we can see the exact results for each algorithm.



```
Algorithm: Linear Regression
MSE for 5-fold CV: 0.5675668500262956
MSE for validation: 0.5690247717229276

Algorithm: Logistic Regression
MSE for 5-fold CV: 0.6620790653426122
MSE for validation: 0.7193877551020408

Algorithm: Support Vector Machine
MSE for 5-fold CV: 0.679407807157072
MSE for validation: 0.6499807765344245

Algorithm: Random Forest
MSE for 5-fold CV: 0.40170756157635 47
MSE for validation: 0.34775581632653063

Algorithm: Decision Tree
MSE for 5-fold CV: 0.7797413793103448
MSE for validation: 0.6602040816326531

Algorithm: Gradient Boosting
MSE for 5-fold CV: 0.4830130465627553
MSE for validation: 0.46181061595210393
```

References:

[1] https://stackoverflow.com/questions/39064684/mean-squared-error-in-python

[2] Cross-validation: evaluating estimator performance - https://scikit-learn.org/stable/modules/cross_validation.html

[3] https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter

[4] https://matplotlib.org/stable/gallery/lines_bars_and_markers/barchart.html#sphx-glr-gallery-lines-bars-and-markers-barchart-py