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 set up the experiment as follows using the Python programming language:

1. **Data Preprocessing:** 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.

2. **Machine Learning Algorithms:** 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

3. **Cross-Validation:** 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 (using this link). This metric evaluates how sufficiently the expected wine quality values align with the actual values.

4. **Statistical Testing:** Statistical testing is a crucial step in model selection to determine which algorithm among a set of candidates is the best performer. It involves applying statistical tests to the evaluation metrics (we used mean squared error in this practice) of different algorithms to determine if one significantly surpasses the others. We used a paired t-test (ttest_rel from scipy.stats) to compare the mean squared error scores of each algorithm pair. The algorithm pair with the lowest p-value will be chosen as the most satisfactory.

Results:

The best algorithm, "best_alg," is chosen based on its performance in the paired t-tests and is a combination of two algorithms; Support Vector Machine vs. Gradient Boosting. Finally, we need to train and evaluate the best algorithm on the test set. We fitted one of the best algorithms (best_model) on the training data (train_features and train_target). After training, the best model is used to make predictions on the test dataset. Then, we calculated the Mean Squared Error (MSE) between the predicted values and the actual target values on the test dataset to quantify the prediction accuracy.

The result of the code is as follows:

```
The best algorithms are: Support Vector Machine vs. Gradient Boosting
Mean Squared Error for the Best Algorithm on Test Data: 0.6499807765344245
```