COSC 4557/5557 Practical Machine Learning Spring 2024
Hyper Parameter Optimization

Submitted by: Iqbal khatoon

Introduction

Hyperparameter optimization is crucial in machine learning to fine-tune models for optimal performance. This study explores various hyperparameter optimization techniques applied to predictive models using the "wine quality" dataset. The objective is to identify the model with the highest predictive accuracy and to evaluate the impact of hyperparameter tuning across different machine learning algorithms.

Wine Quality Dataset

ML algorithm selection exercise employs the "winequality-red" dataset, containing information on different attributes of red wines. These attributes encompass fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol content. The dataset consists of eleven features. The target variable, denoted as "quality," assesses the wine's quality on a scale ranging from zero to ten (0-10). Based on our data analysis on the provided data set, we can ascertain that the dataset contains a total of 1599 entries, each with non-null values across all features and the label. This implies that there are no missing values present in the dataset, which is a positive aspect for our analysis. Furthermore, upon inspecting the data types assigned to each column, we observe that all features have been appropriately assigned the 'float64' data type, indicating numerical values. Similarly, the label class 'quality' comprises integer values exclusively, aligning with its assigned 'integer' data type.

Methodology

Preprocessing Steps

The preprocessing involved two main steps:

- Log Transformation: Applied to reduce skewness in feature distributions.
- Standard Scaling: Used to normalize features, thus ensuring that they contribute equally to the analysis.

Model Training

Four models were trained on the "winequality-red" dataset:

- Random Forest
- Logistic Regression
- Support Vector Machine (SVM)
- AdaBoost

Each model was trained on both raw and preprocessed data. The data was split into training (80%) and testing (20%) sets, with model performance evaluated on the testing set.
1. Logistic Regression:

Algorithms and Hyperparameters

Detailed hyperparameter settings are designed to finely tune each model to achieve the best possible performance on the wine quality dataset while carefully managing the trade-off between fitting the training data and generalizing well to unseen data.

1. Logistic Regression

- C (Inverse of Regularization Strength): This hyperparameter manages the regularization strength, which is crucial in preventing overfitting. The regularization strength is inversely proportional to

C. A lower value of C leads to stronger regularization, helping to reduce model complexity and overfitting risks. The search space for C is log-uniformly distributed between 0.1 and 10, allowing for a wide exploration from very strong to very weak regularization.

2. Random Forest Classifier

- n_estimators (Number of Trees): Specifies the number of trees in the forest. More trees generally improve the model's performance but also increase computational cost. The search space ranges from 100 to 1000, enabling extensive experimentation with both moderate and large forest sizes to find the optimal balance between performance and computational efficiency.
- max_depth (Maximum Depth of Trees): Controls the depth of each tree. A deeper tree can model more complex relationships by allowing more splits in the data. The range for max_depth is set from 10 to 100, facilitating investigations into the effects of tree complexity on the model's accuracy and potential overfitting.

3. Support Vector Machine (SVM)

- C (Regularization Parameter): Influences the trade-off between achieving a low error on the training data and minimizing the model complexity for better generalization to new data. The parameter C is explored within a log-uniform distribution from 0.1 to 10, covering a broad spectrum of values from strong to weak regularization to optimize the margin width and error.
- Gamma (Kernel Coefficient): Affects the reach of a single training example, with higher values resulting in a more complex model that may fit the training data well but generalize poorly. The choices for gamma are 'scale' and 'auto', where 'scale' adjusts gamma based on the number of features and their variance, and 'auto' sets it inversely proportional to the number of features. This allows for adaptive behavior based on the dataset's characteristics.

4. AdaBoost Classifier

- n_estimators (Number of Weak Learners): Determines the maximum number of weak learners that are used in boosting. Increasing this number generally improves the ensemble's correction for misclassified instances in earlier rounds, at the cost of increased computation and potential overfitting. The range for n_estimators is set from 50 to 500, offering a spectrum for evaluating the trade-offs in model complexity and learning capacity.
- learning_rate: Modulates the contribution of each classifier at each iteration. A higher learning rate can strongly impact each learner but might cause rapid convergence to suboptimal solutions. It varies from 0.01 to 2, allowing the model to explore a range of rates from slow to fast learning speeds, optimizing how the model adjusts to errors in previous iterations.

Each model was encapsulated within a pipeline including preprocessing and the classifier, subjected to Bayesian optimization over ten iterations within a 5-fold cross-validation framework targeting maximized accuracy.

Results

Table 1 illustrates the mean cross-validation accuracies obtained from the ML classifiers before and after optimizing the hyperparameters, demonstrating significant improvements, particularly for SVM and Random Forest and AdaBoost.

Classifier	**Default Train**	**Tuned Train**	**Default Test**	**Tuned Test**	**Improvement**
Logistic Regression	0.6037	0.6083	0.5719	0.5625	-0.0094
Random Forest	0.6779	0.6889	0.6469	0.6531	0.0062
SVM	0.5943	0.6216	0.5750	0.6094	0.0344
AdaBoost	0.5505	0.5677	0.5281	0.5594	0.0312

Table 1 - Results of the ML Classifiers Before and After Optimizing the Hyperparameters

Figures 1 and 2 showcase comparative visualizations of the training and test data accuracies before and after tuning, highlighting the efficacy of hyperparameter optimization.

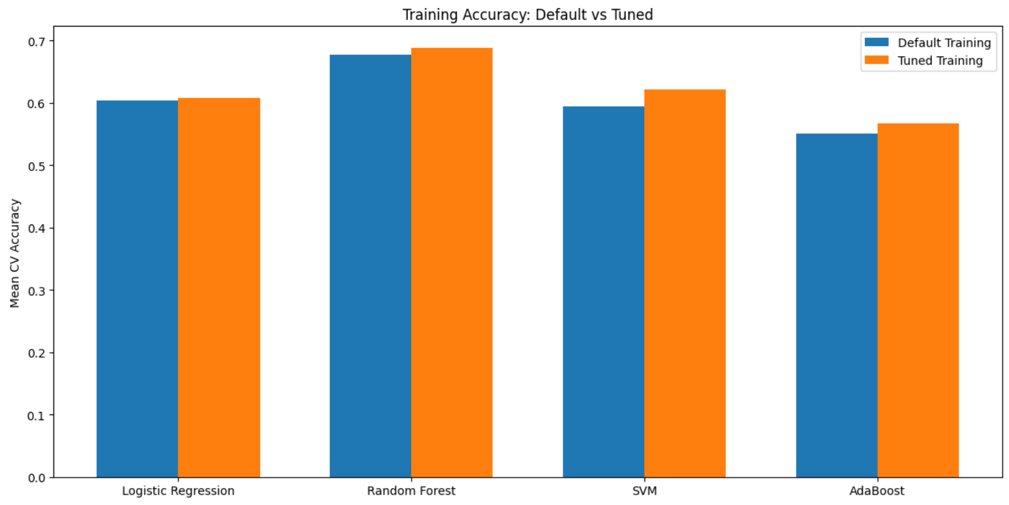

Figure 1: Training-data mean CV accuracies of various classifiers before and after the hyperparameter optimization

Figure 1 shows that the mean cross-validation accuracies during training have consistently increased for all models following hyperparameter tuning, with the most notable improvement seen in the SVM model. This result highlights the improved learning capabilities of the models from the training data when using optimized hyperparameters.

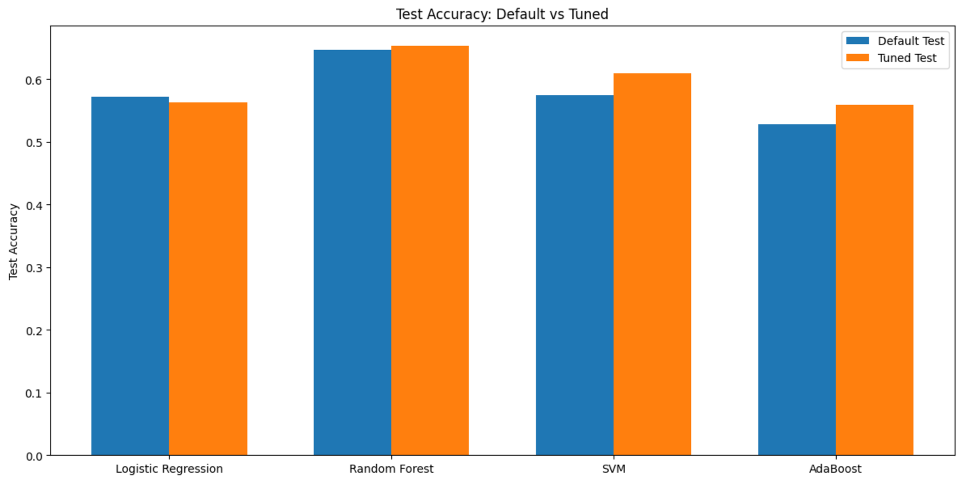

Figure 2: Test-data mean CV accuracies of various classifiers before and after the hyperparameter optimization

Figure 2 demonstrates that the performance of the Random Forest and SVM models on test data has significantly improved due to hyperparameter optimization during training. The mean test CV accuracies for these models increased by 0.0062 and 0.0344, respectively. Initially, the SVM model exhibited the lowest performance; however, post-tuning, it surpassed the Logistic Regression in both training and testing phases. These outcomes affirm the effectiveness of Bayesian optimization in refining model configurations to more accurately discern the underlying patterns within the wine quality dataset. For the Logistic Regression model, despite an increase in training accuracy through hyperparameter optimization, there was a slight decrease in test accuracy compared to its default configuration. This slight reduction in post-tuning accuracy indicates that the chosen hyperparameter space may not have been ideal, or that the model might inherently be less suited to this dataset. Notably, increasing the iterations from 10 to 100 did not result in any significant improvement in accuracy. The AdaBoost model also saw an improvement after tuning, with its test accuracy increasing by 0.0312. This enhancement further underscores the utility of careful hyperparameter adjustments in boosting model performance on unseen data.

The hyperparameter tuning for Logistic Regression indicated a negative effect on test data accuracy, prompting an investigation into data characteristics and preprocessing. Alterations to preprocessing stages did not yield positive improvements, suggesting inherent limitations of the model for this dataset. Changing the random state provided varied results, emphasizing the sensitivity of outcomes to initial conditions.

More on Hyperparameter Optimization:

Logistic regression can be sensitive to how data is scaled and distributed. Initially, the preprocessing involved applying both logarithmic transformation and standard scaling to the features, which may have added complexities that adversely affected logistic regression. Although these preprocessing steps are beneficial for more complex models like random forests and SVMs, they might not be as favorable for logistic regression. To explore this, two additional scenarios were tested on the logistic regression model, one without any preprocessing and another with only scaling. Neither scenario led to improvements in test data performance, leading to the decision to retain the original preprocessing approach.

Random state = 12

Regarding the influence of the random state, changing it can significantly affect the outcomes, particularly in the context of performance gains post-hyperparameter tuning. In this instance, the random state was altered from 42 to 12. The outcomes of this adjustment in hyperparameter tuning are detailed in Figure 3.

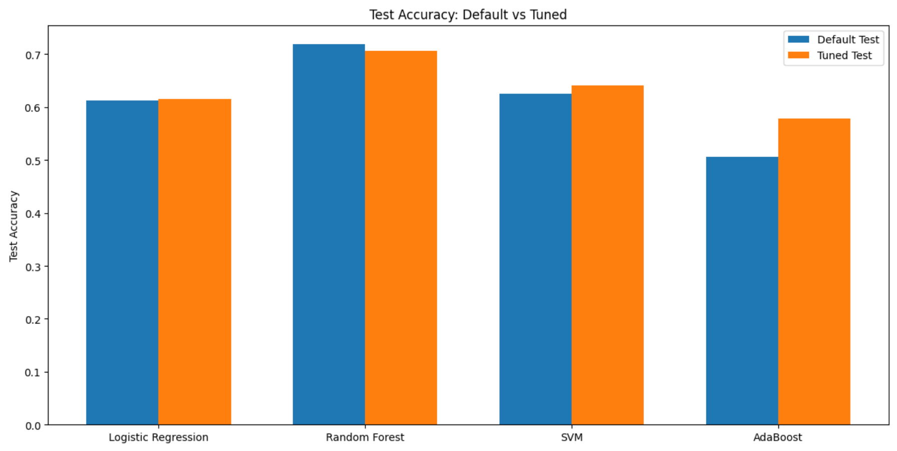

Figure 3: Test-data mean CV accuracies of various classifiers before and after the hyperparameter optimization with random state = 12.

We can see in figure 3 that with change in random state = 12, test accuracy of logistic regression was improved a bit but then Random Forest accuracy decreased.

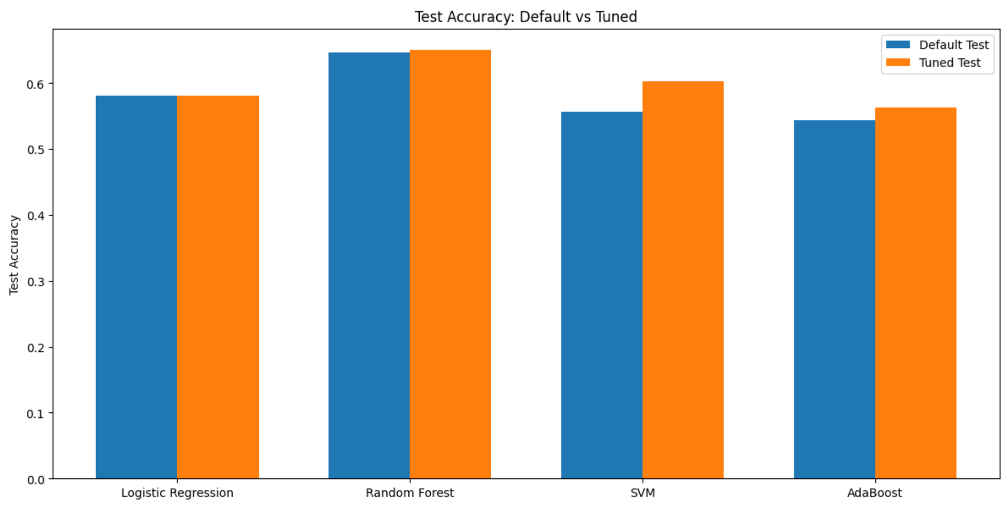

Figure 4: Test-data mean CV accuracies of various classifiers before and after the hyperparameter optimization with random state = 24.

Random sate =24

In Figure 4 we can see that the results after hyperparameter tuning with a random state of 24 showed varied improvements across different models. Logistic Regression saw a slight increase in training accuracy from 0.6044 to 0.6052, with no change in testing accuracy, remaining steady at 0.5813, indicating no improvement. Random Forest's training accuracy rose from 0.6802 to 0.6888, with a modest increase in testing accuracy from 0.6469 to 0.6500, reflecting a small improvement of 0.0031. The SVM model demonstrated significant enhancements, with training accuracy improving notably from 0.5934 to 0.6177 and testing accuracy jumping from 0.5563 to 0.6031, marking the highest improvement among the models at 0.0469. AdaBoost also showed positive changes, with training accuracy moving from 0.5465 to 0.5684 and testing accuracy increasing from 0.5437 to 0.5625, an improvement of 0.0188. There was no any decrement in accuracy with this random state.

Conclusion

The findings from this study emphasize the importance of careful and methodical hyperparameter tuning in enhancing the predictive performance of machine learning models. By systematically exploring and optimizing the hyperparameter space, significant improvements in model accuracy can be achieved, thereby increasing the reliability and applicability of these models in practical scenarios. This study also highlights the necessity of understanding model sensitivities and dataset characteristics to tailor hyperparameter settings effectively, ensuring that the models not only fit the training data well but also generalize efficiently to new, unseen data.

References:

[1] Wine Quality - UCI Machine Learning Repository
[2] **https://scikit-learn.org/stable/**
[3] Iran’s Github repository for Hyperparameter Optimization