Pipeline Optimization Report

Soudabeh Bolouri

Introduction:

This exercise involves a comprehensive approach to predicting wine quality utilizing the Support Vector Machine (SVM) algorithm, a robust classifier used extensively in classification problems. The workflow consists of several key steps, including data loading, preprocessing, model creation, evaluation, hyperparameter tuning, and performance comparison visualization.

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:

First, we need to import our dataset "winequality-white.csv" using Panda. The dataset is then split into training and testing sets with an 80-20 ratio using the **train_test_split** function from Scikit-Learn. In the next step, the dataset is preprocessed to ensure its suitability for model training. Missing values are replaced with the mean of the column via an imputer (this is a general case for implementing pipelines even though there are no missing values for this specific dataset). As a way of standardizing features, it uses the Standard Scaler to remove the mean and scale to unit variance. Then feature scaling and one-hot encoding are performed using **StandardScaler** and **OneHotEncoder** respectively.

The preprocessing pipeline is integrated with an SVM classifier using **Pipeline**. This unified pipeline facilitates seamless data transformation and model training. The initial SVM model, known as the "simple pipeline," is evaluated using the test set to measure its predictive accuracy. Additionally, the time taken for fitting this simple pipeline to the training data is recorded, offering insights into its computational efficiency.

As a comparison, another method in hyperparameter optimization is applied to the pipeline to see if there is an improvement in accuracy. A randomized hyperparameter tuning process is initiated using **RandomizedSearchCV** to fine-tune the SVM model. The hyperparameters and their values to be searched are as follows:

Hyperparameter	Values	Description
classifier__C	[0.001, 0.01, 10, 100]	This parameter controls the trade-off between maximizing the margin and minimizing classification error.
classifier__gamma	[0.1, 0.01, 0.001, 0.0001]	This parameter defines the influence of a single training example's distance, affecting the reach of the kernel
classifier__kernel	['linear', 'rbf', 'poly']	Kernel specifies the type of kernel used in the SVM algorithm, determining the decision boundary shape in the feature space.Three kernel options are considered here: **'linear'**, **'rbf'** (Gaussian kernel), and **'poly'** (polynomial kernel).

Results:

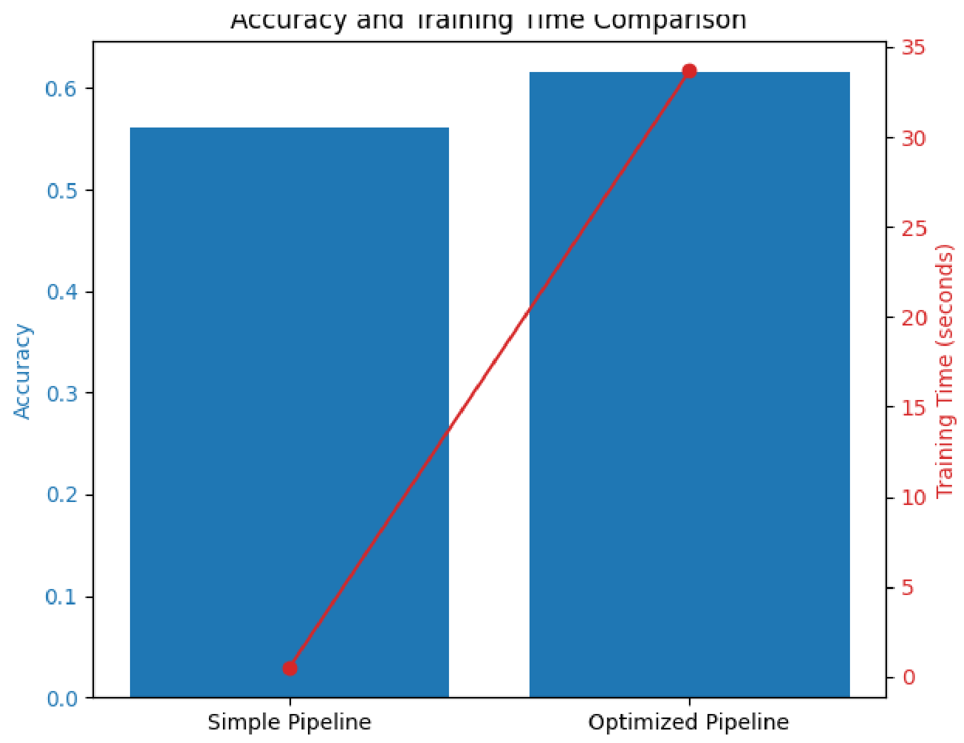
After going through each of these hyperparameters using Random Search, the best hyperparameters have been printed as follows:

- classifier__kernel: 'rbf '
- classifier__gamma: 0.1
- classifier__C: 100

Model	Accuracy (Percent)	Time (Sec)
SVM without HPO	0.5612	0.5039
SVM with HPO	0.6153	33.6680

The analysis highlights the improvement in predictive accuracy achieved by fine-tuning the model parameters while also emphasizing the increased computational time required for the optimized model. We generated informative visualizations to present the findings (The simple pipeline means the one without hyperparameter optimization, while the optimized pipeline uses Random Search as HPO):

- The bar chart visually compares the accuracy scores achieved by simple and optimized pipelines, offering a clear view of their performance differences.
- The line plot illustrates the training times for both pipelines, highlighting the trade-off between model accuracy and computational cost.



References:

[1] https://towardsdatascience.com/step-by-step-tutorial-of-sci-kit-learn-pipeline-62402d5629b6

[2] https://medium.com/mlearning-ai/how-to-use-sklearns-pipelines-to-optimize-your-analysis-b6cd91999be

[3] https://scikit-learn.org/stable/modules/grid_search.html

[4] https://towardsdatascience.com/hyper-parameter-tuning-and-model-selection-like-a-movie-star-a884b8ee8d68