Pipeline Optimization

Ali Torabi

1. Introduction

Machine learning is changing our lives swiftly. While a simple task like image recognition seems so trivial for humans, there is a lot of work to do in the machine learning pipeline, such as data cleaning, feature engineering, finding which models best fit for the specific problem, and hyperparameters tuning, among many other tasks. A lot of these tasks are still not automated and need an expert to do some of these trials and errors. Automated Machine Learning (AutoML) provides a process to automatically discover the best machine learning model for a given task according to the dataset with very little expert need. In this experiment, instead of optimizing a single component, we want to optimize the entire ML approach, including any preprocessing. For this experiment, again the White Wine dataset has been used to predict the quality of wines. The classifier that has been used for this experiment is the Gradient Boosting Classifier. Also, as a different approach to be used for hyperparameter optimization, in this experiment the Randomized Search method is used.

2. Dataset Description

The dataset chosen for this experiment is the Wine Quality dataset [1]. It is related to white wine samples from the north of Portugal. It comprises 1599 instances and 11 different features. The label is the quality of the wine that makes the data suitable for Classification and Regression tasks. Each of these algorithms would detect the quality of wine ranges from poor to excellent. As mentioned in the description of the dataset itself, it has no missing value. The features are: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, and sulphates. The output variable is quality which is scored between 0 and 10. We can use the Pandas describe method to show some of the main properties of the dataset.

	fixed acidity	volatile acidity	citric acid	residual sugar	pH	sulphates	alcohol	quality
count	4898	4898	4898	4898	4898	4898	4898	4898
mean	6.854788	0.278241	0.334192	6.391415	3.188267	0.489847	10.51427	5.877909
std	0.843868	0.100795	0.12102	5.072058	0.151001	0.114126	1.230621	0.885639
min	3.8	0.08	0	0.6	2.72	0.22	8	3
25%	6.3	0.21	0.27	1.7	3.09	0.41	9.5	5
50%	6.8	0.26	0.32	5.2	3.18	0.47	10.4	6
75%	7.3	0.32	0.39	9.9	3.28	0.55	11.4	6
max	14.2	1.1	1.66	65.8	3.82	1.08	14.2	9

3. Experimental Setup

First, we need a library like Pandas to import and manipulate the dataset. We import CSV data into the Dataframe using pandas. To understand the structure of the data, we can use some of the descriptor methods used in Pandas, like Shape, Info, and Describe. Then, the dataset also split into Train, Validation and Test sets. After splitting data, the preprocessing pipeline has been constructed. The first part of the pipeline uses imputer as a tool for the imputation of missing values if any exist. The imputer replaces nan values (missing values) with the mean of the column (imputer is used as a general case for implementing pipeline even though for this specific dataset there is no missing values). Then it uses the Standard Scaler as a way to standardize features by removing the mean and scaling to unit variance.

$$Z = \frac{x - \mu}{\sigma}$$

Where μ is mean and σ is standard deviation. As a categorial transformer, the imputer and on-hot-encoding have been used. Then after training the model, the accuracy and time is reported albeit without using any hyperparameter optimization techniques. As a comparison, another method in hyperparameter optimization is applied to the pipeline to see if there is an improvement in accuracy. The hyperparameters and their values to be searched are as follows:

Hyperparameter	Values	Desc.
n_estimators	[50, 100, 200, 300 ,…, 600]	Number of Trees
learning_rate	0.01, 0.1, 0.2, 0.3	It determines the step size at each iteration while moving toward a minimum of a loss function
max_depth	[3, 4, 5, …, 15]	Depth of the tree. stopping criteria that restrict the growth of the tree
min_samples_split	[2, 4, .., 20]	It controls the minimum number of samples required to split a node and the minimum number of samples at the leaf node
min_samples_leaf	[1,2,3,4,…, 20]	It sets the number of samples in the leaf node

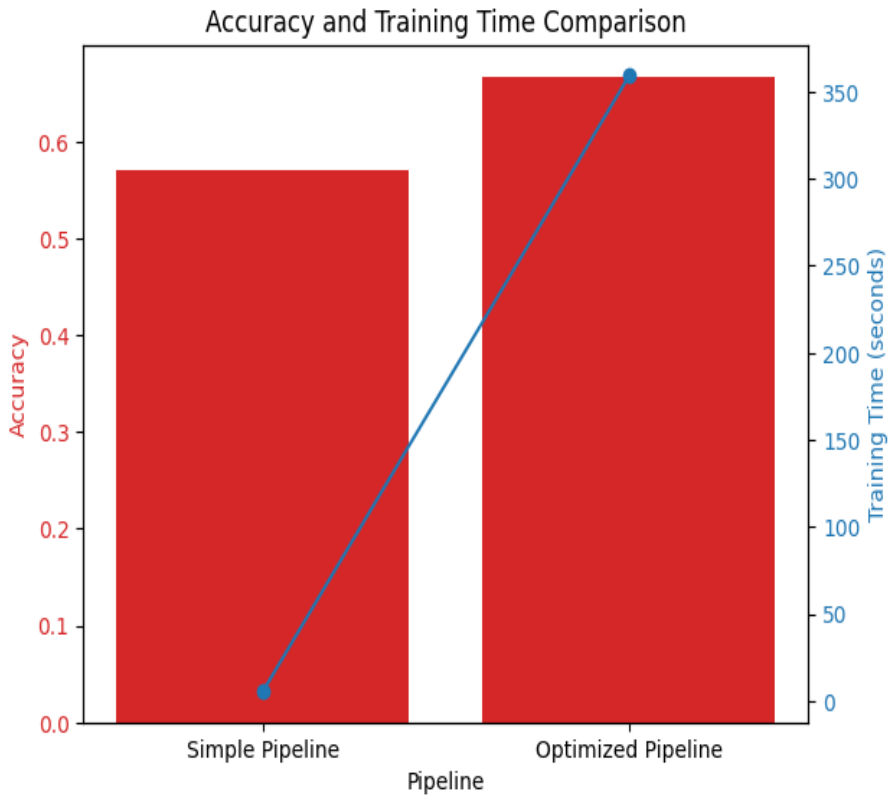
After going through each of these hyperparameters using Random Search, the best hyperparameters have been printed as follows:

- Classifier_n_estimators: 200
- Classifier_min_samples_split: 10
- Classifier_min_samples_leaf: 1
- Classifier_max_depth: 5
- Classifier_learning_rate: 0.2

In this experiment, both time and accuracy are computed as comparison metrics.

Model	Accuracy (percent)	Time (seconds)
Gradient Boosting Classifier without HPO	57.46%	5.399
Gradient Boosting Classifier with HPO (Random Search)	67.28%	358.9692

As you can see, hyperparameter optimization over the entire pipeline can lead to better results. First, you take into account the interdependencies and interactions between the preprocessing and modeling stages when you optimize the hyperparameters for the entire pipeline. This lets you identify a set of hyperparameters that function well together. This holistic approach is more likely to result in a well-tuned model that achieves better generalization and predictive accuracy. However, the particular situation, the resources at hand, and the trade-offs between computational cost and performance will determine whether to optimize the pipeline as a whole or its individual components.
In the diagram below, the comparison between the two aforementioned paradigms has been depicted while indicating both accuracy and time.



The simple pipeline means the one without hyperparameter optimization, while the optimized pipeline uses Random Search as HPO. The left y-axis shows the accuracy scores and the right y-axis show the time takes in seconds. Also, the blue line shows the comparison between the training times that the two approaches take.

References:

1 - https://machinelearningmastery.com/modeling-pipeline-optimization-with-scikit-learn/

2 - https://www.youtube.com/watch?v=TLjMibCN6v4

3 - https://www.youtube.com/watch?v=w9IGkBfOoic

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