Hyperparameter 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 project, I want to predict the quality of wine based on its features. This experiment uses Random Forest as a default algorithm. Then it will compares the accuracy with hyperparameter optimization on Random Search and Bayesian Optimization methods.

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 be to 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.

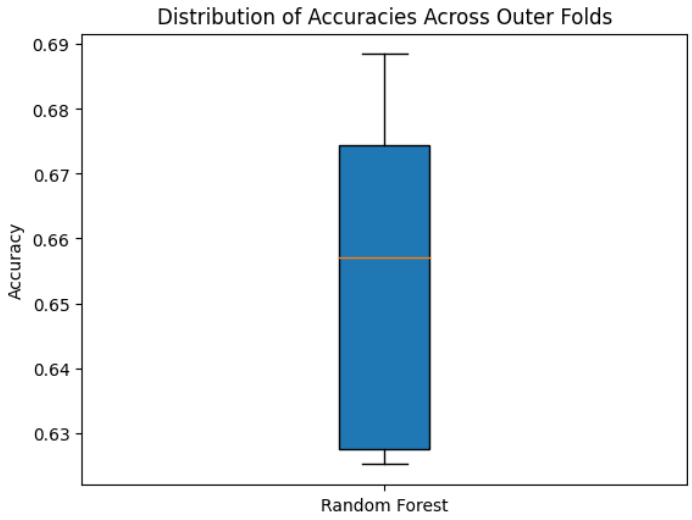
	fixed acidity	volatile acidity	citric acid	residual sugar	pH	sulphates	alcohol	quality
count	4898	4898	4898	4898	4898	4898	4898	4898
mean	6.854788	0.278241	0.334192	6.391415	3.188267	0.489847	10.51427	5.877909
std	0.843868	0.100795	0.12102	5.072058	0.151001	0.114126	1.230621	0.885639
min	3.8	0.08	0	0.6	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 and foremost, 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. Thereafter, some sort of standardization method has been deployed to make the data normalized, in order to make our model more effective. Then, I chose Random Forest as the algorithm to predict the quality of wines. By choosing Random Search and Bayes Optimization as methods of hyperparameter optimization, I try to compare the Random Forest algorithm in different situations. As a baseline, we use Random Forest without any hyperparameter optimization techniques. In Random Search we loop over some of the hyperparameters to find which combination of hyperparameters has the best result in terms of accuracy. Then, the Bayesian Optimization method is used for hyperparameters tuning in Random Forest and at the end the results has been compared in terms of accuracy.

In this experiment I use SK-Learn library for Hyperparameter optimization. I define an exhaustive number of hyperparameters and use cross-validation to find the best combination for our Random Forest model. To make the model more efficient, I also setup the nested resampling. To do this, the Random Forest classifier has been used and the hyperparameters will be tuned using Random Search and Bayes Optimization. Nested resampling uses an additional layer of resampling that separates the tuning activities from the process used to estimate the efficacy of the model. An *outer* resampling scheme is used and, for every split in the outer resample, another full set of resampling splits are created on the original analysis set. For example, if 10-fold cross-validation is used on the outside and 5-fold cross-validation on the inside, a total of 500 models will be fit. The parameter tuning will be conducted 10 times and the best parameters are determined from the

average of the 5 assessment sets. This process occurs 10 times [4]. The outer cross-validation as Stratified K-Fold with number of splits equal to 5 has been used. In each iteration in outer cross-validation's split, the Bayes Optimization will be applied on each inner_cv and the computer accuracy will be appended to the outer_fold_accuracies. After executing the nested resampling, the distribution of accuracies across the folds can be shown in a plot like blow box-plot for Bayes Optimization:



The values for Outer Fold Accuracies in Bayes Optimization are as follows:

Fold	Accuracy (%)
1	65.71
2	67.44
3	62.75
4	68.84
5	62.51

As a general rule, by using Bayesian optimization, we have a more efficient search through the hyperparameter space compared to Random Search. Albeit, worth noting that, by expanding the hyperparameter and search space to search through more exhaustively, you allow the model to explore a broader range of possibilities, potentially leading to better performance at the cost of increased computational demand. The Random Forest parameters that feeding to the Random Search and Bayesian Optimization with possible values has been shown in Table below:

Hyperparameter	Values	Desc.
n_estimators	[100, 200, …, 1000]	Number of Trees in Random Forest
max_features	None, sqrt, log2	The methods to use randomly select the number of features at each node.
max_depth	[5, 10, 15, …, 30]	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

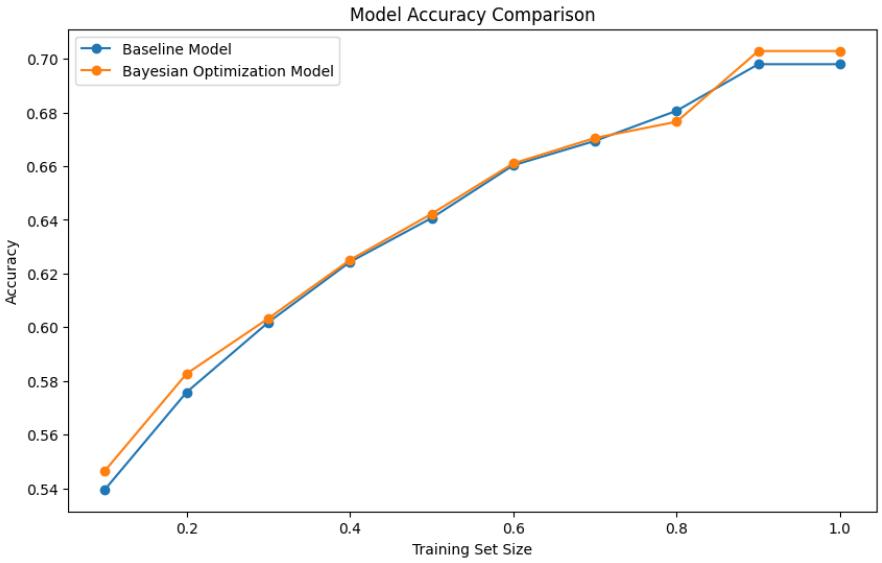
The values of the same hyperparameters have been used for both Random Search and Bayesian Optimization, To make the comparison fair. After tuning the hyperparameters, I evaluate the models by accuracy. The optimized model is then tested on unseen data (test set) to assess its generalization capability. For Baseline, Random Search, and Bayesian the same settings have been used, like using the same K-Fold validation (with K = 5) and, the same metric to compare, among other settings. Nested resampling is also used for optimization. As budget for this experiment, would be the maximum number of iterations for evaluation for both Bayesian Optimization and Random Search. The number of iterations for both algorithms is 100. The best hyperparameters that has been found for Random Search method are : Max_Depth = 20; Max_Features = None; Min_Samples_Leaf = 1; Mean_Samples_Split = 2; N-Estimators = 300. But the same hyperparameters for Bayesian method have the values to give the best value in terms of accuracy like this: Max_depth = 25; Max_Features = sqrt; Min_Samples_Leaf = 6; Mean_Samples_Split = 4; N_Estimators = 900. The results has been shown in the table below:

Model	Accuracy
Random Forest Baseline	66.85%
Random Forest with Random Search	68.75%
Random Forest with Bayesian Optimization	68.84%

When comparing these models in terms of accuracy, there would be several important points to consider: First, Both Random and Bayesian methods have led to improvements in accuracy over the baseline. This indicates that hyperparameter tuning has been effective in enhancing model

performance. The improvement from the baseline to Random Search and Bayesian Optimization is obvious. Also, the more sophisticated and exhaustive search strategy of Bayesian Optimization may be more effective for this specific dataset and problem. Sometimes even minor advancements can have a significant impact, particularly in fields like financial forecasting or medical diagnostics. The real improvement in accuracy must be evaluated against the increased computing cost and complexity involved with hyperparameter tuning, particularly with Bayesian Optimization.

Evaluating the result for this experiment, where we have the model tuned with Bayesian Optimization, involves analyzing how the performance of the model changes as the amount of training data increases. We compare the baseline model and Bayesian method in the graph in terms of accuracy. As the graph shows, the accuracy gets better for Bayesian optimization over time and got better accuracy after all compared to the baseline method. The reason for this difference accuracy between Bayesian and Random Search Optimization is that Bayesian is more efficient due to the fact that it uses past evaluations in order to select the next hyperparameter values to evaluate. So, in this way, it tries to find the best hyperparameters in fewer iterations. It also means Bayesian Optimization even can explore hyperparameters in which they are outside of a predefined hyperparameters space. Let's discuss how to interpret these curves in the context of the problem. The comparison is shown in the diagram bellow:



The x-axis represents the size of the training data .The y-axis represents the model's performance (e.g., accuracy). Typically, two lines are plotted: one for the baseline accuracy and another for the Bayesian score. This graph shows good performance for Random Forest with hyperparameter optimization using Bayesian method.

References:

1 - https://medium.com/grabngoinfo/support-vector-machine-svm-hyperparameter-tuning-in-python-a65586289bcb

2 - https://www.geeksforgeeks.org/svm-hyperparameter-tuning-using-gridsearchcv-ml/

3 - https://medium.com/grabngoinfo/support-vector-machine-svm-hyperparameter-tuning-in-python-a65586289bcb#:~:text=You can check out the,to make it linearly separable.

4 - https://www.tidymodels.org/learn/work/nested-resampling/#:~:text=Nested%20resampling uses an additional,on the original analysis set.