Hyperparameter Optimization

COSC 5010 – 03

Isaac Baah

1.0 Introduction

Building machine learning models is very important, however, building robust machine learning supersedes just building basic machine learning models. In this project, we will delve into the world of hyperparameter tuning. For this project, two hyperparameter tuning processes were adopted, the Randomized SearchCV and the Bayesian Optimization. In the Randomized SearchCV process, an initial hyperparameters were chosen, when I realized that the model was overfitting, I increased the search space for the hyperparameters and later got a model that is generalized and performs well in predicted the unseen data (testing data).

2.0 Data

The data used is from a machine learning tutorial I did on Udemy, the data consists of nine independent variables. All the variables are numeric. The dependent variable is the "Class". The data is trying to predict the whether the plant will grow "big or tall" or otherwise considering the characteristics of the plant. Class "2" means the plant will grow "big or tall" and Class "4" means "Otherwise".

Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
5	1	1	1	2	1	3	1	1	2
5	4	4	5	7	10	3	2	1	2
3	1	1	1	2	2	3	1	1	2
6	8	8	1	3	4	3	7	1	2
4	1	1	3	2	1	3	1	1	2
8	10	10	8	7	10	9	7	1	4
1	1	1	1	2	10	3	1	1	2
2	1	2	1	2	1	3	1	1	2
2	1	1	1	2	1	1	1	5	2
4	2	1	1	2	1	2	1	1	2
1	1	1	1	1	1	3	1	1	2
2	1	1	1	2	1	2	1	1	2
5	3	3	3	2	3	4	4	1	4
1	1	1	1	2	3	3	1	1	2
8	7	5	10	7	9	5	5	4	4
7	4	6	4	6	1	4	3	1	4

Figure 1. A snapshot of the data used in the project.

3.0 Experimental Setup

To predict based on the characteristics of the plant whether a plant will grow very well "big or tall" or grow otherwise, several models were developed. In this project, the following models were built, Logistic Regression, Support Vector Machines, Decision Trees, and Random Forest Classifier. This project used the scikit-learn library available in python to conduct this project. Using the decision tree as an example for fitting the data. The first thing was to split the data into

training and testing datasets. The data was split into 80% training data and 20% testing data. The training data was then used to train the machine learning models. The model utilized a constant partitioning of 10-fold cross-validation and two hyperparameter optimization methods utilizing accuracy as the main scoring criteria.

For the Randomized SearchCV, we tried to select the best hyperparameters that will prevent our model from overfitting. First, a range of values and categories were defined for the hyperparameters and then we ran the selected hyperparameters on the data. Again, we also tried to increase the range for the hyperparameters. That is, we increased the range of hyperparameters that take numeric values and the categories were also added to the hyperparameters that take on categorical values. At the end, we selected the model that performed well on both the training and testing datasets.

4.0 Results

In this section, I present the results of the hyperparameter optimization project. What I did in this project was to select the best hyperparameters for the dataset for every algorithm. Again, I also compared the performance of the result of using Randomized SearchCV and Bayesian Optimization.

Table 1. Selected Hyperparameters for the final models

ML Models	**Hyperparameters**
Logistic Regression	solver = 'liblinear', max_iter = 50, C = 0.1
Support Vector Machines	kernel = 'linear', gamma = 0.5, C = 0.8
Decision Trees	min_samples_split = 10, min_samples_leaf = 15, criterion = ‘entropy’
Random Forest	n_estimators = 300, criterion = ‘gini’, min_samples_split = 200, min_samples_leaf = 8

The Figures 2,3,4,5 below shows the confusion matrix for the machine learning algorithms used in this project.

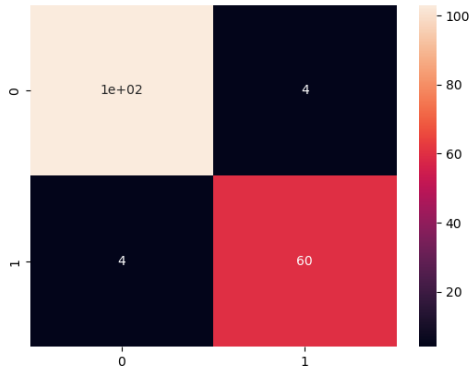

Figure 2. Confusion Matrix of test data for the Logistic Regression Model.

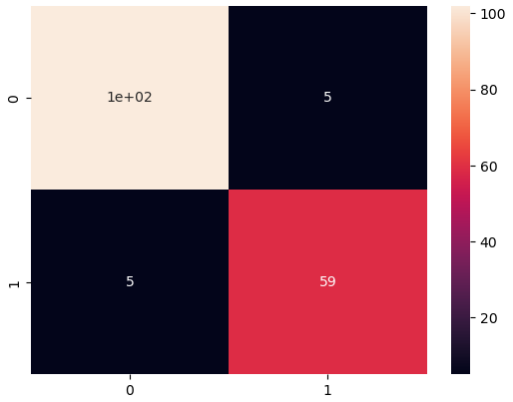

Figure 3. Confusion Matrix of test data for the Support Vector Machines Model.

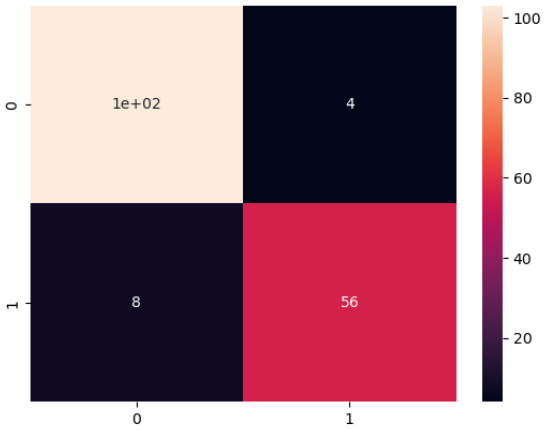

Figure 4. Confusion Matrix of test data for the Decision Tree Model.

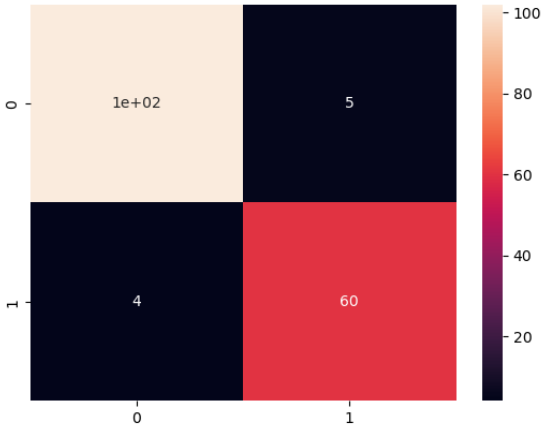

Figure 5. Confusion Matrix of test data for the Random Forest Model.

From the four-confusion matrix above, we can see that logistic regression was able to classify well on the test data compared to the other models.

Comparison of Randomized SearchCV to Bayesian Optimization

This part of the project will compare the performance of the two optimization methods that were used. From the two figures below, we can observe that the two hyperparameter optimization methods performed almost the same except for the Decision Tree model in which the Bayesian Optimization method outperformed the Randomized SearchCV on both the training and testing datasets.

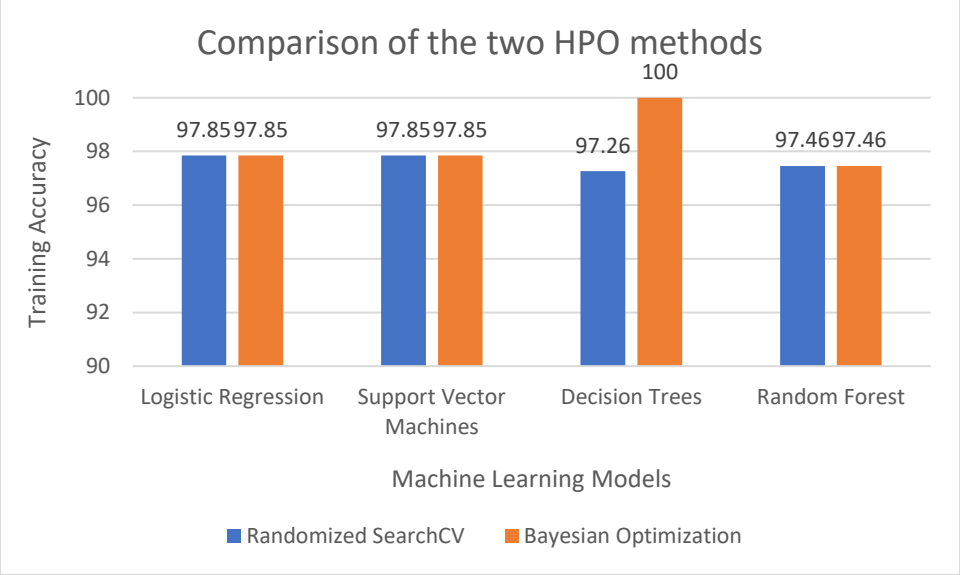

Figure 6. Comparison of the two HPO methods based on Accuracy of the Training data.

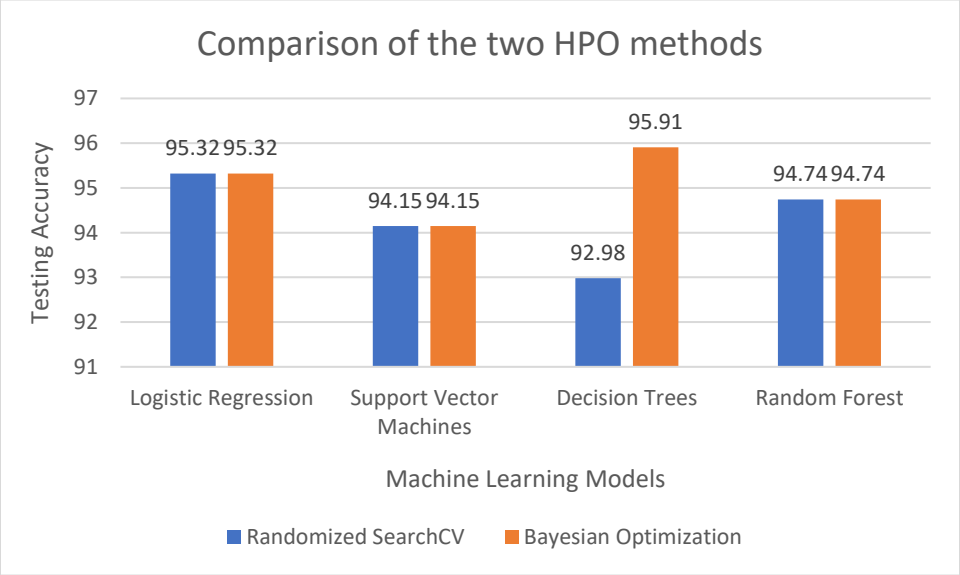

Figure 7. Comparison of the two HPO methods based on Accuracy of the Testing data.

5.0 References

1. Hands-On Machine Learning with Python and R. https://www.udemy.com/course/machinelearning/. [Accessed, October 2023].

2. Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061.