Hyperparameter Tuning Using Optimization Strategies Project

March 28, 2025

Zarin Karanikolov & Stoyan Valchev

---

1 Introduction

Hyperparameter tuning is a a step in optimizing machine learning models, as it directly impacts both training efficiency and predictive performance. This project explores three popular hyperparameter optimization strategies—grid search, random search, and Bayesian optimization—to fine-tune parameters such as the number of decision trees in the forest, maximum depth of each individual decision tree and minimum number of samples that are required to split an internal node. By comparing these methods, we analyze their effects on training time and model accuracy(overfitting potential), highlighting the trade-offs between computational cost and performance improvement.

The code's repository, related to this project is in the following link in GitHub: https://github.com/zarkobeats/Hyperparameter-Tuning-Using-Optimization-Strategies.git

2 Methodology

This section describes the three hyperparameter optimization techniques used to tune a decision tree-based model. The parameters optimized include the number of decision trees in the forest, the maximum depth of each tree, and the minimum number of samples required to split an internal node. Each method is evaluated based on its impact on training time and overfitting potential.

2.1 Grid search optimization

Grid search systematically evaluates all possible combinations of hyperparameter values within predefined ranges. This exhaustive approach ensures the best combination is found but can be computationally expensive, especially when optimizing multiple parameters with large value ranges. [1]

2.2 Random search optimization

Random search selects hyperparameter values randomly from specified distributions. Unlike grid search, it does not explore all possible combinations, making it more efficient while still capable of finding near-optimal configurations, particularly when some hyperparameters have less influence on performance. [2]

2.3 Bayesian search optimization

Bayesian optimization builds a probabilistic model to guide the search for optimal hyperparameters. By leveraging past evaluations, it intelligently selects new parameter values, balancing exploration and exploitation. This approach reduces computational cost while often achieving superior performance compared to grid and random search. [3]

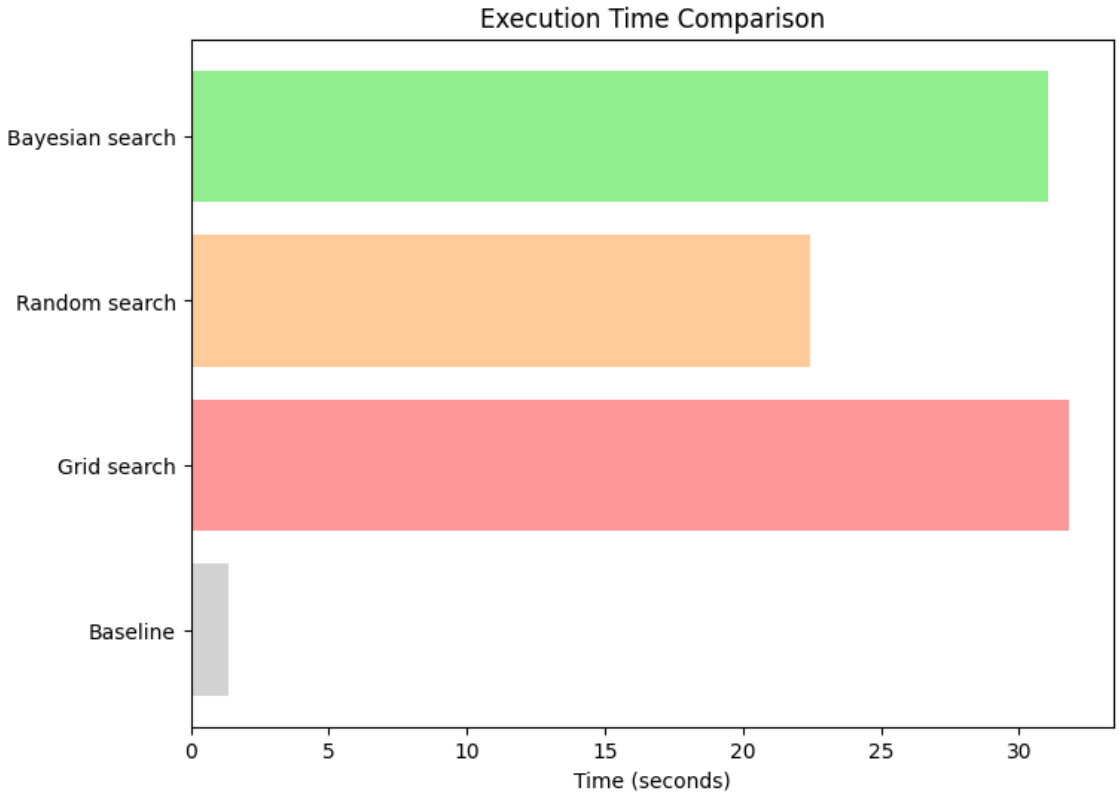
3 Experiment and results

The following results show the performance of different hyperparameter tuning strategies applied to a decision tree-based model. In this experiment we are making 100 iterations.

***Disclaimer: Results may slightly vary based on different computers/runs, because we couldn't achieve reproducibility and the results aren't strictly the same. No matter that, the main point of out study still applies.**

3.1 Testing Runtime

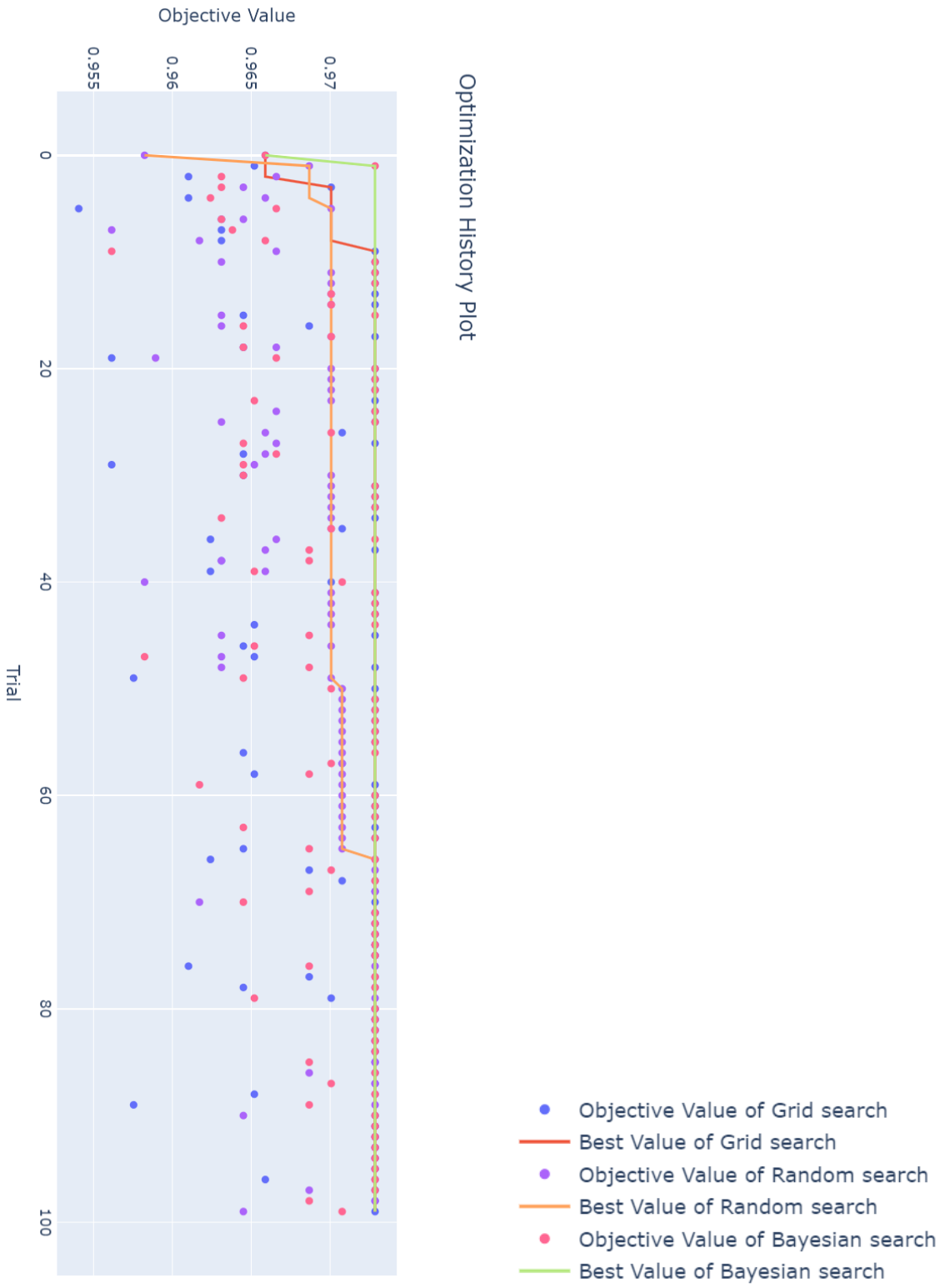
The Baseline Model Performance has an accuracy score of 0.9701 and training time of 1.35s. This serves as a reference point for evaluating the impact of hyperparameter tuning.
The Grid search optimization achieved accuracy score of 0.9729 and training time of 31.82s.
The Random search optimization achieved accuracy score of 0.9729 and training time of 22.43s.
The Random search optimization achieved accuracy score of 0.9729 and training time of 31.06s.
The following graph shows the runtime for every of the optimizers compared to each other and also the performance os the baseline model:



Almost every time Random search was the fastest optimization, while Bayesian and grid search had similar run times, still Bayesian had faster run time.

3.2 Accuracy

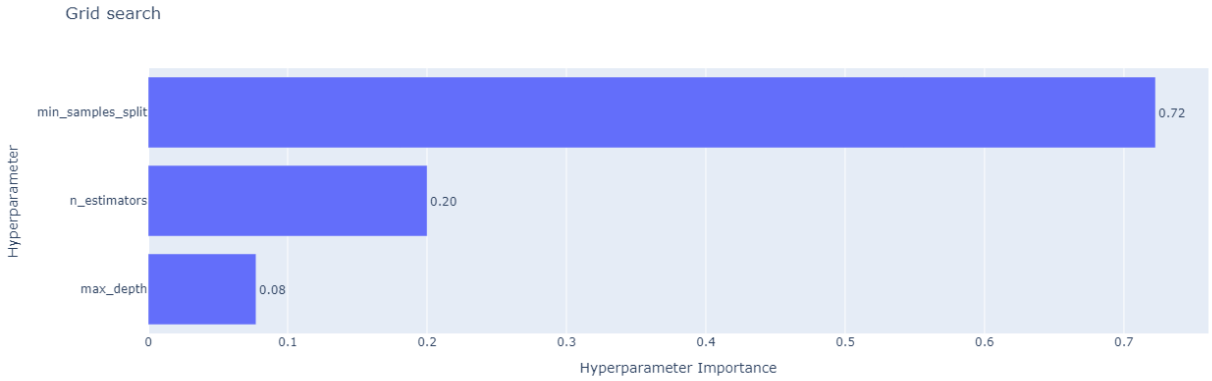
In order to see which optimizer is the best, we iterated 100 times and found out which achieves earliest the best high score, which is the same for the 3 optimization models - 0.9729. The following plot shows how the accuracy changes throughout the study.

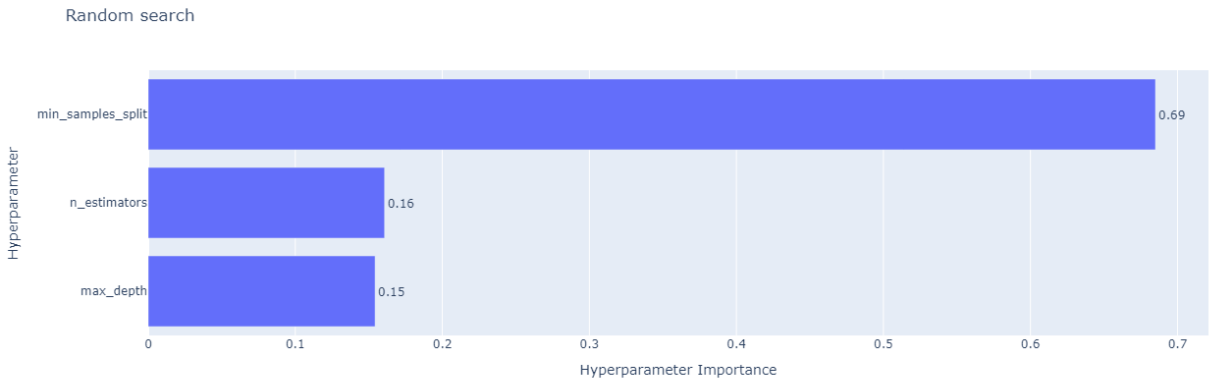

Optimization History Plot
Objective Value
0.955
0.96
0.965
0.97
Trial
0
20
40
60
80
100
Objective Value of Grid search
Best Value of Grid search
Objective Value of Random search
Best Value of Random search
Objective Value of Bayesian search
Best Value of Bayesian search

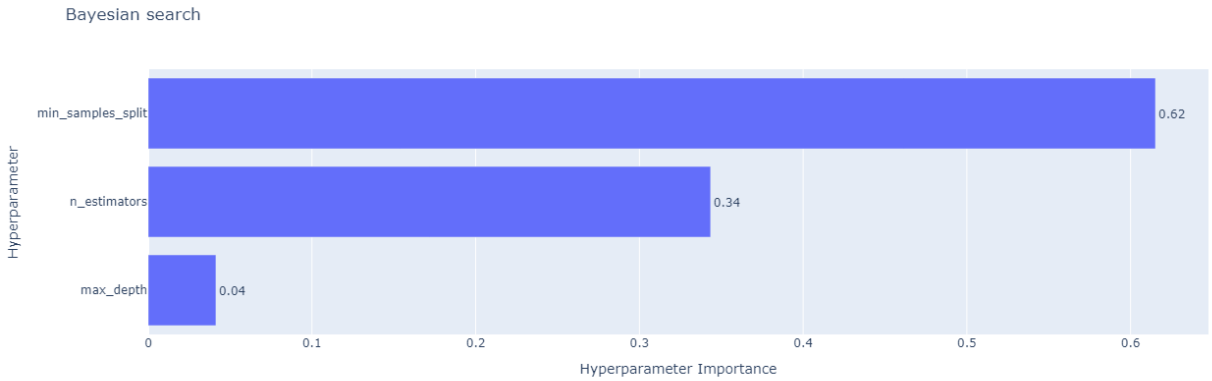
It can be seen, that every optimization achieves the same best value. Then the other parameter that is relevant is how fast was it achieved: The fastest one was Bayesian, which reached the global maximum in only 2 iterations.
Second comes, which converged in the optimum in 10 iterations.
Last is the random search, which reached peak accuracy in 67 iterations.

3.3 Hyperparameter Importance

This section is about which hyperparameter from the mentioned in the introduction has bigger or less of an impact on the type of search optimization. The following graphs show exactly this:



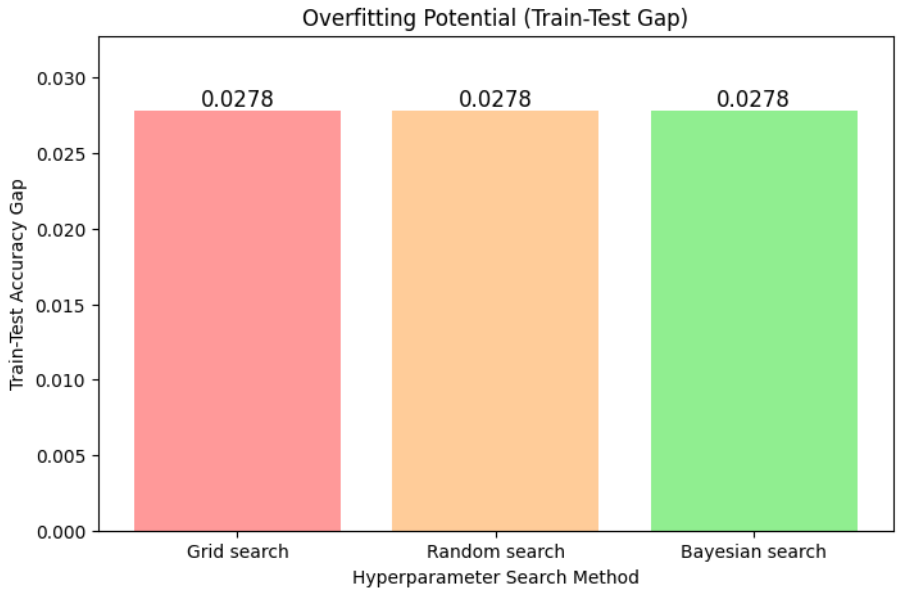




Seeing the results, it can be concluded that the most important hyperparameter is the minimum number of samples that are required to split an internal node. In other hand the maximum depth of each decision tree seems to be more irrelevant.

3.4 Overfitting potential

This section is to calculate the overfitting potential for each optimizer in this study. Because every of the optimizers reached the same accuracy, sooner or later, the overfitting potential for them is the same. The formula that we 'created' is just subtracting the test accuracy from the training accuracy, to see if the data is doing well to unseen data, relatively to the one, used for training:



4 Conclusion

Based on the experiments, all three optimization strategies—Grid Search, Random Search, and Bayesian Optimization—achieved the same best accuracy score of 0.9729. However, their efficiency varied significantly:
Bayesian Optimization was the most efficient, reaching peak accuracy in just 2 iterations. Grid Search required 10 iterations to reach the same accuracy.
Random Search was the least efficient, taking 67 iterations to converge to the optimal accuracy.
In terms of runtime, Random Search was generally the fastest optimizer, but Bayesian Optimization provided the best trade-off between speed and accuracy. Grid Search, while complete, took significantly more time due to evaluating all possible parameter combinations.
From the hyperparameter importance analysis, the minimum number of samples required to split an internal node was the most influential parameter, while maximum depth had a relatively minor impact.
Regarding overfitting potential, since all optimizers reached the same accuracy eventually, their overfitting risks were similar. A simple calculation comparing test and training accuracy showed no significant overfitting across the models.

Bayesian Optimization emerged as the best overall strategy, offering a balance of speed, accuracy, and computational efficiency. Grid Search, while effective, is computationally expensive, and Random Search, though faster, is less reliable in quickly finding the optimal solution.

References

[1] Petro Liashchynskyi, Pavlo Liashchynskyi, 2019. *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*, Department of Computer Engineering, Ternopil National Economic University, Ternopil, 46003, Ukraine

[2] James Bergstra, Yoshua Bengio, 2012. *Random Search for Hyper-Parameter Optimization*, Departement d'Informatique et de recherche operationnelle, Universite de Montreal, Montreal, QC, H3C 3J7, Canada

[3] Alma Rahat, Michael Woo, 2020. *On Bayesian Search for the Feasible Space Under Computationally Expensive Constraints*, Department of Computer Science, Swansea University, Swansea, UK