Genetic Algorithm vs Bayesian Optimization for Random Forest Hyperparameter Tuning in Organic Photovoltaic PCE Prediction

Group 2

Dean Kim, Aldahir Flores, Radhika Rathi, Hita Pullur

1. Abstract

Organic photolith voltaic (OPV) devices can provide great advantages due to its lightweight, solution processed characteristics. However, device performance depends on many coupled molecular and device parameters, which makes model selection and hyperparameter tuning an important step for data-driven prediction. In this study, we focus on hyperparameter tuning strategies for forging predictive relationships of OPV higher power conversion efficiencies from data embedded in polymer-device descriptor pairs.

A random forest (RF) regression model needing an OPV dataset contained units with varying PCE was constructed. After training the model, hyperparameter optimization through a global approach, genetic algorithm (GA) and a Bayesian optimization (BO) was done. GA, in literature, is an optimization technique employing the natural selection theory and a fitness function (here, the model's predictive RMSE, root mean square error) to govern a population of candidate points. GA was popularized in many fields, OPV included, as a universal global optimization technique. In this work, BO is used for hyperparameters in a manner similar to GA rule; only the fitness functions are prediction error RMSE.

We compare the two approaches on the same search space and the same train and validation split. Our comparison focuses on test set performance (R^2 and RMSE), convergence behavior during the search, and the practical cost of each method in terms of model evaluations. The results provide guidance on when a simple genetic algorithm is sufficient for OPV PCE prediction and when a more sample efficient Bayesian optimization scheme is preferable for hyperparameter tuning in chemical data science.

2. Introduction

Organic photovoltaics (OPVs) use π conjugated polymers and small molecules to convert sunlight into electrical power. Their mechanical flexibility and solution processability make them attractive for lightweight and large area devices. At the same time, OPV performance is sensitive to many design choices, such as polymer electronic structure, molecular weight distribution, and device architecture. As a result, researchers often face a high dimensional design space with complex, non-linear relationships between inputs and power conversion efficiency (PCE).

Machine learning models can help capture these relationships and provide fast predictions of PCE once trained on existing experimental data. However, the quality of these models depends not only on the choice of algorithm but also on the choice of hyperparameters, such as tree depth or number of estimators in an ensemble. Poorly chosen hyperparameters can hide the true potential of a model. Systematic hyperparameter optimization thus becomes a key step in building reliable predictive tools for OPV design.[3)]

Traditional approaches such as grid search and random search are simple to implement but can be inefficient in high-dimensional spaces. More advanced strategies, including genetic algorithms and Bayesian optimization, treat hyperparameter tuning as a global optimization problem where each evaluation is expensive because it requires training a full model. [3), 4)] In many studies, Bayesian optimization has been found to reach competitive or superior performance with fewer evaluations compared to simple search strategies or heuristic evolutionary methods. [3), 7), 8)]

In this project, we compare these two approaches in the context of OPV PCE prediction. Using a data set of conjugated polymer devices, we train a random forest regressor and tune its hyperparameters with both a genetic algorithm and Bayesian optimization. Our goal is not only to obtain a good predictive model but also to understand the trade-offs between these two hyperparameter optimization strategies in a realistic chemical data science problem. The comparison connects to previous work that has contrasted genetic algorithms and Bayesian optimization for hyperparameter tuning in neural networks and engineering systems. [3), 7), 8)]

2.1. Organic photovoltaic PCE and available descriptors

The PCE of an OPV device is related to three key device level quantities,

$$PCE = \mathrm{Voc} \times \mathrm{Jsc} \times \mathrm{FF}$$

where V_{OC} is the open circuit voltage, SC is the short circuit current density, and FF is the fill factor. In the dataset used in this work, PCE is provided directly as the target variable, and the corresponding device parameters V_{oc}, J_{sc}, and FF appear as separate features.

The OPV dataset contains more than one thousand devices built from different conjugated polymers. For each device, the following descriptors are available:

- Device level: PCE (%), V_{oc} (V), J_{sc} (mA cm²), FF
- Polymer molecular weight descriptors: Mw, Mn, polydispersity index (PDI)
- MonomerW: molecular weight of the repeat unit
- Electronic structure descriptors: HOMO (eV), LUMO (eV), and bandgap (eV)
- Identifiers: ID, Nickname, and SMILES string of the polymer repeat unit

These descriptors provide a mix of device level and molecular level information that influences

charge generation, transport, and recombination. By training a regression model on these descriptors, we aim to predict PCE for new polymer and device combinations and to evaluate how advanced hyperparameter optimization methods can improve predictive performance.

2.2 Hyperparameter optimization

A hyperparameter is a model setting that must be chosen before training, such as the number of trees in a random forest, the maximum depth of each tree, or the number of features considered at each split. Hyperparameters control model capacity, regularization strength, and training dynamics. Good hyperparameter choices can improve prediction accuracy and stability, while poor choices can lead to underfitting or overfitting.

Hyperparameter optimization treats this choice as a mathematical optimization problem. [3] The decision variables are the hyperparameters, and the objective is usually a validation metric such as cross validated RMSE. Each function evaluation requires training the model with a given set of hyperparameters and measuring its performance on a held out validation set. Because each evaluation can be computationally expensive, especially for large models or complex cross validation schemes, efficient search methods are valuable. [4]

2.3 Genetic algorithms

Genetic algorithms (GAs) are population-based stochastic search methods inspired by the Darwinian principle of natural selection and by biological evolution. [1, 2] A GA maintains a set of candidate solutions, called individuals, that together form a population. Each individual encodes a candidate solution as a chromosome, which is a vector of genes. In a hyperparameter optimization context, the genes correspond to hyperparameters, and the chromosome represents a specific hyperparameter configuration.

The algorithm evaluates each individual using a fitness function. Individuals with better fitness are more likely to contribute to the next generation through selection, crossover, and mutation. [1] Common selection schemes include tournament selection and fitness proportional selection. Crossover recombines the chromosomes of two parents to form offspring that combine genes from both parents. Mutation introduces random changes to some genes to maintain diversity in the population and to help the algorithm escape local minima. [1]

Each generation of the GA consists of evaluating the population, selecting parents, generating offspring through crossover and mutation, and forming the next population. Over successive generations, the population should shift toward regions of the search space with better fitness. Theoretical and empirical studies have analyzed genetic algorithms in terms of schema sampling and building block propagation, and they show that GAs can be effective at exploring rugged search spaces. [1]

Genetic algorithms have also been applied extensively in chemistry and chemical engineering. For example, GA-based methods have been used to optimize experimental conditions, design chromatographic separations, and calibrate complex models. [2] In these applications, the GA can handle mixed discrete and continuous decision variables and can accommodate arbitrary fitness functions. However, performance is sensitive to algorithm parameters such as population size, crossover rate, and mutation rate, which need to be tuned for a given problem. [1, 4] General online tutorials provide accessible introductions to the basic operators and implementation details of genetic algorithms. [9]

2.4 Bayesian optimization

Bayesian optimization (BO) provides an alternative strategy for global optimization of expensive black box functions. In BO, a probabilistic surrogate model approximates the unknown objective function as a function of the decision variables. The optimizer uses this surrogate to select new points that balance exploration of unknown regions and exploitation of promising regions. [3), 7)]

In the context of hyperparameter optimization, the decision variables are hyperparameters, and the objective is a validation metric such as RMSE. The surrogate model can be a Gaussian process or a tree-based model, and the acquisition function may be expected improvement, probability of improvement, or an upper confidence bound. [3)] Given a limited budget of evaluations, BO aims to identify a near optimal hyperparameter configuration with fewer evaluations than grid search, random search, or heuristic evolutionary methods.

Previous studies have applied Bayesian optimization to hyperparameter tuning of neural networks and other models in engineering and scientific domains. [3), 7), 8)] In many of these cases, BO has been found to reach good performance with fewer evaluations than genetic algorithms or manually designed search procedures, particularly when the hyperparameter space has moderate dimensionality and each training run is expensive. [3), 6), 7), 8)]

2.5 Related work on GA and BO comparison

Several previous studies have compared genetic algorithms and Bayesian optimization, or closely related algorithms, for hyperparameter tuning and engineering optimization. Alibrahim and Ludwig compared genetic algorithms, grid search, and Bayesian optimization for neural network hyperparameter optimization and reported that Bayesian optimization achieved comparable or better accuracy with fewer evaluations. [3)] Adachi and co-workers examined how algorithm parameters influence the performance of Bayesian optimization, genetic algorithms, and particle swarm optimization on materials design problems and found that performance can strongly depend on these parameters. [4)]

Hybrid approaches that combine Bayesian optimization and genetic algorithms have also been proposed. Zhang and co-workers developed a Bayesian optimization and GA based approach for automatic parameter calibration of soil models, where Bayesian optimization first identifies promising regions of the parameter space and a genetic algorithm then refines the solution within these regions. [5)] Mori and colleagues introduced novel indices to study the sampling dynamics of genetic algorithms and Bayesian optimize algorithms and compared their convergence behavior. [6)]

Other work has compared GA and BO-based tuning strategies across different engineering systems. Ding and co-workers investigated neural network-based wind pressure prediction for low rise buildings and compared genetic algorithm and Bayesian optimization for selecting network hyperparameters. [7)] Zhou and collaborators studied heterogeneous vehicle routing problems and compared parameter tuning for the routing algorithm using a genetic algorithm and Bayesian optimization. [8)] These studies generally report that Bayesian optimization is more sample efficient, while genetic algorithms provide a robust baseline that can be easier to adapt to different problem structures.

Our work follows this line of research but focuses on OPV PCE prediction with a random forest model and on a chemical data science dataset.

3. Methods

3.1 Dataset and preprocessing

We used the OPV training and test datasets from Quiz 1. Each row corresponds to one organic photovoltaic device built from a conjugated polymer, with device-level quantities (PCE, V_{oc}, J_{sc}, FF) and molecular descriptors (Mw, Mn, PDI, MonomerW, HOMO, LUMO, bandgap) plus identifiers (ID, Nickname, SMILES).

We treated PCE as the regression target. We first removed rows with missing PCE values and then dropped any remaining rows that contained missing values in the numeric features. After cleaning, the dataset contained 960 devices and 11 numeric descriptors. We formed the feature matrix X by dropping the PCE column and selecting only numeric columns, and the target vector y contained the PCE values. For model evaluation, we used an 80/20 train–validation split with a fixed random seed (random state = 0 for the baseline model and random state = 42 for the GA runs). The separate Quiz 1 test file was only used for generating predictions; its labels are not available.

3.2 Baseline random forest model

As a baseline, we trained a RandomForestRegressor with 100 trees and default settings for the other hyperparameters. The model used all numeric features as inputs and PCE as the target. We first evaluated this model with 5-fold cross-validation on the cleaned dataset, using K Fold(n_splits=5, shuffle=True, random state=42). The mean RMSE across folds was 0.3598 ± 0.0318, and the mean R^2 was 0.9761 ± 0.0041.

We then performed a single 80/20 train–validation split to use the same validation set when comparing with GA and BO. On this held-out validation subset, the baseline random forest achieved $R^2 = 0.9821$ and RMSE = 0.3172. These values serve as a reference point for the hyperparameter tuning methods.

3.3 Genetic algorithm tuning

We next tuned the random forest hyperparameters with a genetic algorithm. Each individual in the population encoded a candidate hyperparameter set:

- $n_{estimators} \in \{50,100,200,300,400\}$
- max_depth ∈ {None,5,10,20,40}
- min_samples_split ∈ {2,4,6,10}
- min_samples_leaf ∈ {1,2,4}
- max_features ∈ {"sqrt","log2",None}.

We initialized a population of 20 individuals by sampling each hyperparameter uniformly from its allowed values. For each individual, we trained a random forest on the training subset and computed the validation RMSE. This RMSE served as the fitness, so lower RMSE means higher fitness.

In each generation, we sorted the population by RMSE and kept the best 50 percent as survivors. We then filled the rest of the population by mating pairs of survivors. The crossover operator built a child by choosing, for each hyperparameter, the value from one of the two parents at random. We applied mutation to the child with probability 0.2 per gene by resampling that hyperparameter from its domain. We repeated this process until we returned to a population size of 20. This loop ran for 10 generations. We recorded the best RMSE in each generation to track convergence and took the best individual in the final generation as the GA-optimized hyperparameter set.

After tuning, we retrained a random forest on the full training data (training plus validation subsets combined) using the best GA hyperparameters and used this model to generate predictions for the test file.

3.4 Bayesian optimization tuning

We also tuned the random forest with Bayesian optimization using the BayesianOptimization package. Here we defined a continuous hyperparameter domain:

- n_estimators in [50,400]
- max_depth in [5,40]
- min_samples_split in [2,10]
- min_samples_leaf in [1,4]

We fixed max_features to "sqrt" in this search. The optimizer operated on continuous variables, so we rounded the suggested values to integers before training the model.

The objective function rf_cv constructed a random forest with the proposed hyperparameters and evaluated it with 5-fold cross-validation on the full training dataset X, y. The function returned the mean of the cross-validated negative RMSE scores, so the Bayesian optimizer effectively maximized the mean negative RMSE or, equivalently, minimized RMSE.

We started with 5 random initial points and then ran 30 Bayesian optimization iterations, for a total of 35 objective evaluations. We then extracted the hyperparameter set that achieved the best objective value. With this best set, we trained a final random forest on the training subset and evaluated it on the validation subset using R^2 and RMSE. We also retrained the same model on the full training data to produce test-set predictions.

3.5 Visualizations

To compare the optimization methods, we generated three main plots:

- Best GA validation RMSE as a function of generation.
- Cross-validated RMSE from each Bayesian optimization iteration.
- Predicted versus actual PCE scatter plot for the best-performing model on the validation set, with a y=x reference line.

These plots provide a visual summary of convergence behavior and overall predictive quality.

4. Results and discussion

4.1 Baseline model performance

The baseline random forest model already achieved strong performance on the OPV dataset. On the 20 percent validation subset, it obtained $R^2 = 0.9821$ and RMSE = 0.3172.
Over 5-fold cross-validation on the full cleaned dataset, the mean RMSE was 0.3598 ± 0.0318 and the mean R^2 was 0.9761 ± 0.0041. The high R^2 and low RMSE show that the chosen molecular and device descriptors contain enough information to predict PCE accurately for this dataset. The predicted versus actual PCE scatter plot for the baseline model shows points tightly clustered around the diagonal line over the full PCE range from about 0 to 10 %. This suggests that a relatively simple ensemble model captures most of the structure in the data.

Given this strong baseline, we do not expect hyperparameter tuning to yield dramatic reductions in error. Instead, we use the genetic algorithm and Bayesian optimization to study how advanced search methods behave on a realistic chemical data science problem where the model is already near a performance ceiling.

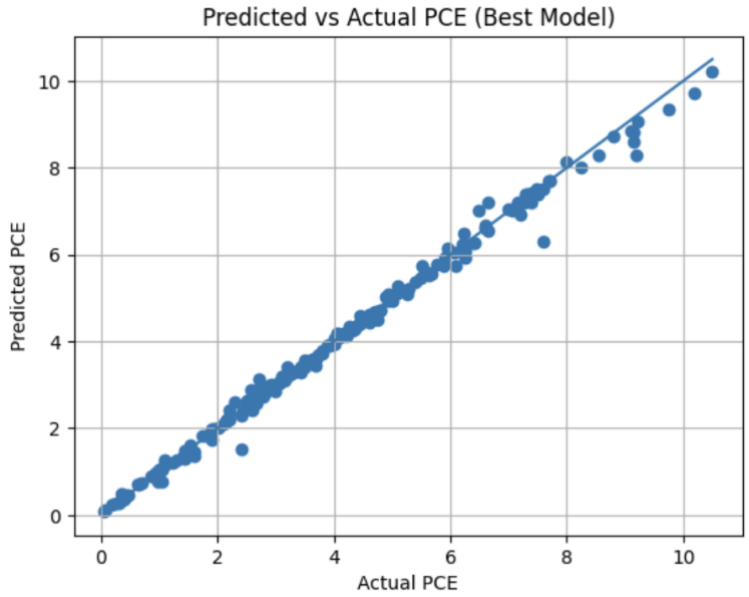

Figure 1 Predict vs Actual PCE (Best Model)

4.2 Genetic algorithm optimization

The genetic algorithm improved the random forest performance over the first few generations and then plateaued. Figure 1 (GA best RMSE per generation) shows that the best RMSE in the population started at about 0.332 in generation 1, remained near 0.332 in generation 2, and then decreased to about 0.329 in generations 3 and 4. In generation 5, the best RMSE dropped further to 0.3269, followed by a small improvement to 0.3257 in generation 6. After this point, the GA converged and the best RMSE stayed at 0.3257 through generation 10.

The best hyperparameter set discovered by the GA was:

- n_estimators = 400

- max_depth = None
- min_samples_split = 2
- min_samples_leaf = 1
- max_features = None

Using these hyperparameters, the random forest achieved a validation RMSE of 0.3257 on the 20 percent hold-out set. This value is only slightly worse than the baseline RMSE of 0.3172. The GA therefore moved the model toward a configuration with more trees and unrestricted tree depth, but the overall predictive performance changed very little.

This behavior is consistent with the idea that the baseline model already lies in a region of hyperparameter space that gives near-optimal performance for this dataset. The GA successfully drove the population toward better individuals in the early generations and then explored around a plateau of similarly good solutions. The limited room for further improvement is due to the small gap between the baseline and the best achievable RMSE within the chosen search space.

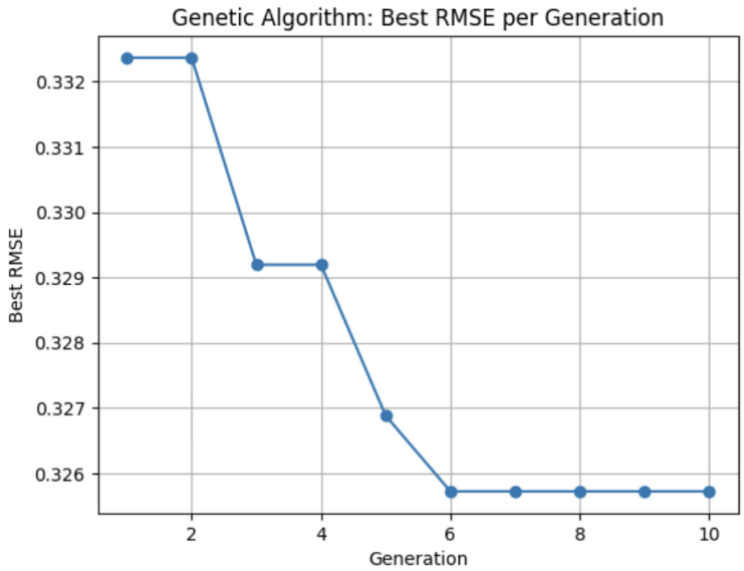

Figure 2 Genetic Algorithm: Best RMSE per Generation

4.3 Bayesian optimization behavior

Figure 2 shows the RMSE from each Bayesian optimization iteration, obtained by converting the negative RMSE scores returned by the optimizer. Across the 35 evaluations, the cross-validated RMSE values fluctuate between roughly 0.55 and 0.80. The best mean RMSE found by Bayesian optimization is about 0.55, which is significantly larger than the validation RMSE of the baseline and GA-tuned models.

The final Bayesian-optimized random forest used the following hyperparameters:

- n_estimators = 363
- max_depth = 31
- min_samples_split = 2

- min_samples_leaf = 1
- max_features = "sqrt"

When trained on the training subset and evaluated on the validation subset, this model achieved

- $R^2 = 0.9485$
- RMSE = 0.5522

which is noticeably worse than both the baseline and GA-tuned models.

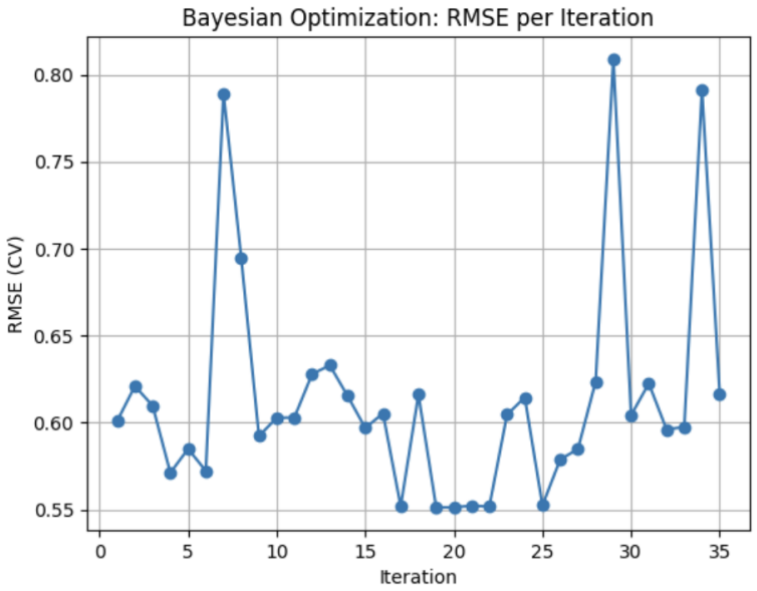

Figure 3 Bayesian Optimization: RMSE per iteration

There are several likely reasons for this outcome. Several factors may explain this outcome. First, the GA and BO used different objectives. The GA minimized RMSE on a single 80/20 train–validation split, while Bayesian optimization minimized the mean 5-fold cross-validated RMSE on the full dataset. These two objectives are not identical. The cross-validated objective is more conservative and can penalize complex models, which may lead to larger apparent errors even when generalization is acceptable.

Second, each BO evaluation involves 5-fold cross-validation, so the RMSE estimates are noisy because of finite data and random sampling. With only 30 optimization iterations after the 5 random initial points, the surrogate model may not have converged to the true optimum in the hyperparameter space.

Third, the search spaces differ. In the Bayesian optimization runs, max_features was fixed to "sqrt", while the GA could also select "log2" or None. The GA's best solution used max_features = None, which allows all features to be considered at each split. This region of the search space was not explored by Bayesian optimization.

Overall, the Bayesian optimization procedure did not outperform the simpler GA or the baseline tuning in this setting. This does not contradict earlier reports of BO performing well

in other problems. It shows that BO performance depends on the objective definition, the noise level, the search space, and the available evaluation budget.

4.4 Comparison and interpretation

The three approaches can be compared directly in terms of their validation performance. The baseline random forest, trained with 100 trees and default hyperparameters, achieved R^2 = 0.9821 and an RMSE of 0.3172 on the 80/20 train–validation split. When evaluated with five-fold cross-validation on the cleaned dataset, the same model reached a mean RMSE of $0.3598 \pm 0.0318$and a mean R^2 of 0.9761 ± 0.0041, confirming that it is already a strong predictor of PCE.

The GA-tuned random forest produced a best validation RMSE of 0.3257, which is very close to the baseline value and within the expected variance of the cross-validation results. The genetic algorithm favored a configuration with 400 trees, unrestricted depth, and max_features = None, indicating a slight preference for a larger ensemble and more flexible trees, but without translating into a meaningful accuracy gain.

In contrast, the Bayesian-optimized random forest performed noticeably worse on the same validation set. The best hyperparameter set found by Bayesian optimization yielded an RMSE of 0.5522 and R^2 = 0.9485. Despite using more trees and a relatively deep maximum depth, this model underperformed both the baseline and the GA-tuned models, which underscores how sensitive Bayesian optimization can be to the choice of objective function, search space, and evaluation budget.

The predicted versus actual PCE plot for the best model shows a tight linear relationship along the y=x line. This indicates that the model captures the main dependence of PCE on the available descriptors. Residual errors remain small across most of the PCE range, with somewhat larger scatter only at the highest PCE values where the data are sparse.

From these results, we conclude that GA hyperparameter tuning does not significantly improve performance for this dataset and model, because the default settings already lie near a good region of the search space. Bayesian optimization, as implemented here, produces more conservative hyperparameters and worse validation performance, likely due to the cross-validated objective, the fixed max_features choice, and the limited number of evaluations. The study highlights that more sophisticated optimization algorithms are not guaranteed to outperform simpler baselines, especially when the model is already robust and the dataset is of moderate size.

5. Conclusions

We applied genetic algorithms and Bayesian optimization to tune random forest hyperparameters for predicting the power conversion efficiency of organic photovoltaic devices. The OPV dataset provides device-level and molecular-level descriptors, and a simple random forest model already achieves high predictive accuracy with R^2 around 0.98 and RMSE around 0.32 on a held-out validation set.

The genetic algorithm explored a discrete hyperparameter space through selection, crossover, and mutation. It reduced the best validation RMSE in the population from about 0.33 in the first generation to about 0.326 in later generations and identified a configuration with 400 trees, unrestricted depth, and max_features = None. However, the GA-optimized model only matched, rather than clearly exceeded, the baseline performance. This suggests that, for this problem, the default random forest settings are already close to optimal within the chosen search space.

Bayesian optimization searched a continuous hyperparameter space using a 5-fold cross-validated RMSE objective. In our implementation it did not improve performance and produced models with validation RMSE around 0.55. The difference in objective functions, the noise in cross-validation estimates, the fixed choice of max_features, and the limited evaluation budget likely all contributed to this result. These findings show that Bayesian optimization is not guaranteed to outperform simpler strategies when the search space is small, the model is robust, and the dataset is modest in size.

Overall, this study illustrates that advanced hyperparameter optimization methods must be evaluated in the context of the specific model and dataset. For the OPV PCE prediction task considered here, a straightforward random forest with reasonable default hyperparameters already performs very well. Genetic algorithm tuning provides a check that there is no large performance gap left unexploited, while Bayesian optimization may require more careful objective design and a larger evaluation budget to be effective. Future work could repeat this comparison for more complex models, such as gradient boosting or neural networks, and explore hybrid approaches that combine Bayesian optimization with evolutionary operators.

6. Reference

1) Whitley, D. (1994). A genetic algorithm tutorial. Statistics and Computing, 4(2), 65–85. https://doi.org/10.1007/BF00175354

2) Leardi, R. (2007). Genetic algorithms in chemistry. Journal of Chromatography A, 1158(1–2), 226–233. https://doi.org/10.1016/j.chroma.2007.03.051

3) Alibrahim, M., & Ludwig, S. A. (2021). Hyperparameter optimization: Comparing genetic algorithm against grid search and Bayesian optimization. In 2021 IEEE Congress on Evolutionary Computation (CEC) (pp. 1–8). IEEE. https://doi.org/10.1109/CEC45853.2021.9504761

4) Adachi, T., Ikeda, K., Fukuda, A, & Mizuta, S. (2019). Influence of algorithm parameters of Bayesian optimization, genetic algorithm, and particle swarm optimization on their optimization performance. Advanced Theory and Simulations, 2(4), 1900110. https://doi.org/10.1002/adts.201900110

5) Zhang, N., Wang, Y., & Li, X. (2024). A Bayesian optimization–genetic algorithm-based approach for automatic parameter calibration of soil models: Application to clay and sand model. Computers and Geotechnics, 161, 106717. https://doi.org/10.1016/j.compgeo.2024.106717

6) Mori, H., Yamaguchi, Y., & Shingu, K. (2005). A comparison study between genetic algorithms and Bayesian optimize algorithms by novel indices. In Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (pp. 1637–1644). https://doi.org/10.1145/1068009.1068244

7) Ding, H., Zhou, K., & Yan, J. (2022). Neural-network-based wind pressure prediction for low-rise buildings with genetic algorithm and Bayesian optimization. Engineering Structures, 260, 114203. https://doi.org/10.1016/j.engstruct.2022.114203

8) Zhou, H., Yang, S., & Xu, B. (2022). Heterogeneous vehicle routing: Comparing parameter tuning using genetic algorithm and Bayesian optimization. In 2022 International Conference on Unmanned Aircraft Systems (ICUAS) (pp. 1055–1061). IEEE. https://doi.org/10.1109/ICUAS54217.2022.9836044

9) GeeksforGeeks. (2024). Genetic algorithms. Retrieved November 25, 2025, from https://www.geeksforgeeks.org/dsa/genetic-algorithms/