AUTOML WITH BAYESIAN HYPERPARAMETER OPTIMIZATION

Table of Contents

1. Overview of Hyperparameter Optimization

Hyperparameter optimization aims to find the best set of hyperparameters for a machine learning model to maximize its performance metrics, such as accuracy or AUC-ROC score. Unlike model parameters, which are learned during training, hyperparameters are set prior to the training process. Proper tuning of these hyperparameters can significantly enhance the model's performance.

Bayesian Optimization

Bayesian optimization is an advanced method for hyperparameter tuning that uses probabilistic models to predict the performance of different hyperparameter settings. It balances exploration (trying new hyperparameters) and exploitation (refining known good hyperparameters) efficiently. Key components include:

- **Surrogate Model:** Typically a Gaussian Process (GP), which models the objective function.
- **Acquisition Function:** Determines the next set of hyperparameters to evaluate, balancing exploration and exploitation.

2. Expected Improvement and Propose Location Functions

In Bayesian optimization, two critical components are the Expected Improvement (EI) acquisition function and the Propose Location function. These components guide the optimization process by balancing exploration and exploitation, helping us efficiently search the hyperparameter space.

- **Expected Improvement Function**
  The Expected Improvement function quantifies how much improvement we can expect from sampling a new point compared to the best point sampled so far. The idea is to focus on areas where the model predicts high uncertainty and potential for improvement.
- **Propose Location Function**
  The Propose Location function identifies the next point in the hyperparameter space to evaluate by maximizing the Expected Improvement.
  The differential_evolution function is used to find the candidate point that maximizes the Expected Improvement. This optimization technique is suitable for complex, non-convex functions and provides a global optimization approach.

3. Custom Optimization Implementation

Process Description

The custom implementation of Bayesian hyperparameter optimization involves the following steps:

1. **Define the Bounds:** Specify the range for each hyperparameter.
2. **Initialize Samples:** Generate initial samples within the bounds.
3. **Gaussian Process Regression:** Use GPR to model the relationship between hyperparameters and the objective function (e.g., accuracy).
4. **Iterative Optimization:**
   - Fit the GPR model with current samples.
   - Use the acquisition function to propose the next sample.
   - Evaluate the objective function at the new sample.
   - Update the GPR model with the new sample.
   - Repeat until the maximum number of iterations is reached.

Results

The process is applied to different models, such as XGBoost, GradientBoosting, and AdaBoost. The custom optimization aims to find the best hyperparameters to maximize model accuracy.

4. Hyperopt-Based Optimization Implementation

Process Description

Hyperopt is a powerful Python library for hyperparameter optimization, which provides a convenient and efficient way to search through the hyperparameter space of machine

learning models. Hyperopt uses tree-structured Parzen estimators (TPE) as a surrogate model for Bayesian optimization, making it well-suited for complex search spaces.

Here is a step-by-step description of the optimization process using Hyperopt:

1. **Define the Search Space:** The search space is specified using Hyperopt's hp module, which allows for different types of distributions (e.g., uniform, normal) and integer constraints. This search space includes the hyperparameters of the models we want to optimize.
2. **Define the Objective Function:** This function trains the model with the given hyperparameters and evaluates its performance using cross-validation. The objective function returns the negative score (since Hyperopt minimizes the objective).
3. **Run Optimization:** Hyperopt's fmin function is used to perform the optimization. The tpe.suggest algorithm guides the search through the hyperparameter space. The optimization process runs for a specified number of evaluations (e.g., 20).
4. **Train the Best Model:** Using the best hyperparameters, the final model is trained and evaluated to obtain the performance metrics.

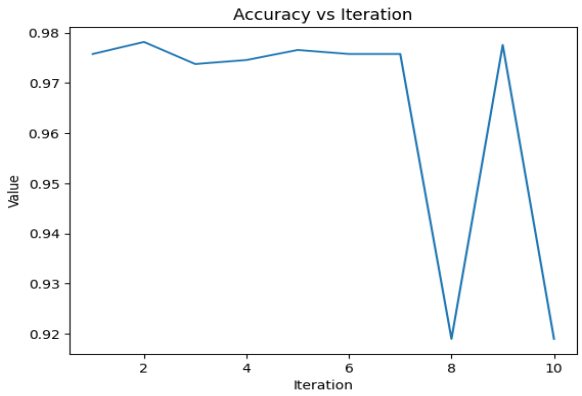
Results

The results of the Hyperopt-based optimization include:

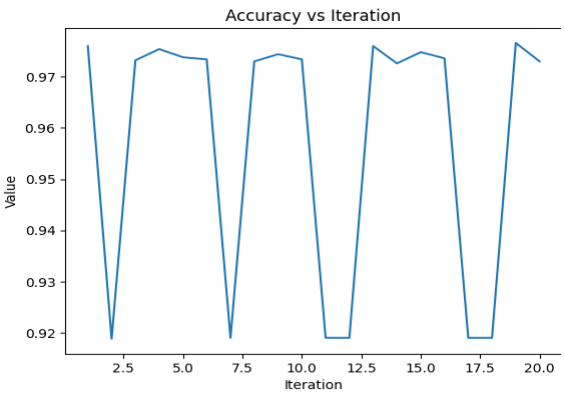
- **Best Hyperparameters:** The optimal set of hyperparameters found by Hyperopt.
- **Accuracy:** The mean cross-validated accuracy of the model with the best hyperparameters.
- **ROC AUC Score:** The ROC AUC score, which evaluates the model's ability to distinguish between classes.

Advantages of Using Hyperopt:

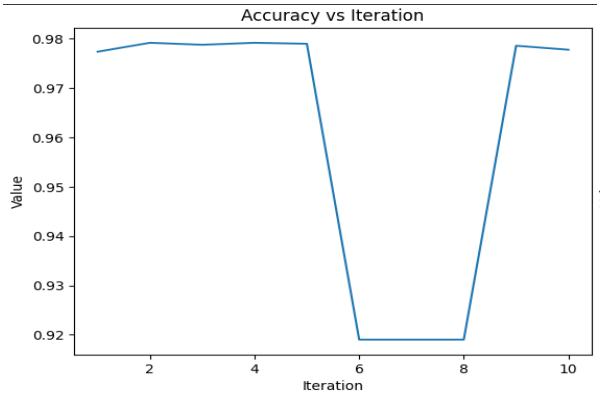
- **Efficiency:** Hyperopt efficiently explores the hyperparameter space using probabilistic models, which can be more effective than grid search or random search, especially for high-dimensional spaces.
- **Ease of Use:** The hp module and fmin function provide a straightforward way to set up and run optimization tasks.
- **Versatility:** Hyperopt can be used for a wide range of optimization tasks beyond hyperparameter tuning, including algorithm selection and feature selection.



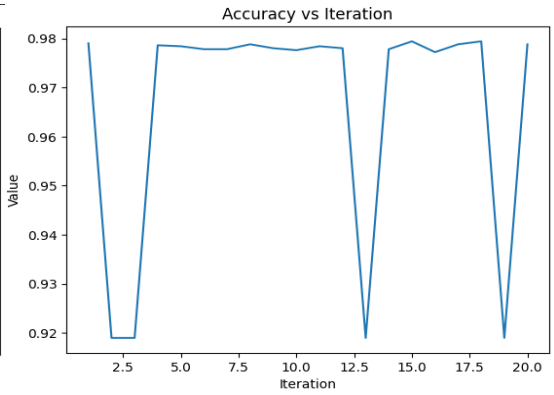
Gradient Boosting Accuracy= 0.92



Gradient Boosting Accuracy with Hyperopt= 0.97



XGBoosting Accuracy= 0.97



XGBoosting Accuracy with Hyperopt= 0.98

5. Summary and Conclusion

In this report, we explored two approaches to Bayesian hyperparameter optimization for AutoML models: a custom implementation and an approach using Hyperopt. Both methods demonstrated the capability to efficiently search the hyperparameter space and improve model performance.

Key Points:

- **Custom Implementation:** Provides fine-grained control over the optimization process but requires more effort to implement and tune.
- **Hyperopt Implementation:** Easier to use with built-in functionalities for hyperparameter optimization, making it a practical choice for most users.

Conclusion

Bayesian optimization, whether custom-implemented or via libraries like Hyperopt, is a powerful method for tuning hyperparameters in AutoML models. It systematically balances exploration and exploitation, leading to more efficient and effective optimization compared

to traditional methods. By using these techniques, we can achieve improved model performance, crucial for developing robust machine learning solutions.