INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE

VLG Open Project 2024

AutoML

Automated Hyperparameter Optimisation

Name : RADHIKA MAHESHWARI
Enrollment Number: 22114075
Email-id : radhika_m@cs.iitr.ac.in

INTRODUCTION

Machine learning (ML) models are integral to various applications, such as image recognition and natural language processing. The performance of these models significantly depends on the selection of hyperparameters, including settings like learning rate, regularization parameters, and neural network architecture. Traditionally, choosing these hyperparameters is a manual, time-consuming process that requires substantial expertise. However, this process can be automated using AutoML techniques.

The goal of this project is to develop an automated hyperparameter optimization (HPO) system leveraging AutoML techniques. This system aims to efficiently identify the optimal hyperparameter configurations for given ML models and datasets, enhancing model performance while minimizing human intervention. This report outlines the design and implementation of the automated hyperparameter optimization system. Additionally, the system aims to integrate with various ML models, accommodate different data types, and employ efficient HPO techniques to determine the best hyperparameter configurations.

PROBLEM STATEMENT

Given the crucial impact of hyperparameter settings on the performance of machine learning models, the objective is to develop an automated HPO system capable of efficiently identifying optimal hyperparameters for various models and datasets. The system should:

- Integrate with multiple machine learning models and handle diverse data types.
- Employ efficient AutoML techniques such as Bayesian optimization, random forests, or TPE.
- Provide evaluation metrics including ROC AUC, cross-validation scores, and learning rate distribution curves.
- Compare learning rate distribution curves across different HPO methods, such as random search, submitted model, and TPE.
- Exclude pre-existing HPO models like Hyperopt to encourage custom implementation.

SYSTEM DESIGN AND IMPLEMENTATION

The system architecture is designed to accommodate different data types and seamlessly integrate with various machine learning models. The core components of the architecture include:

- Data Preprocessing Module: This module prepares datasets for training and evaluation. It handles tasks such as managing missing values, encoding categorical features, and scaling numerical features to ensure the data is suitable for model training.

- Model Selection Module: This module includes various machine learning models such as SVM, Random Forest, etc. It offers a flexible interface to select and train different models based on the given dataset.Models Used:
    - Support Vector Machine (SVM)
    - Random Forest
    - Gradient Boosting Classifier
    - LightGBM

- Bayesian Optimization Module: This module efficiently searches for optimal hyperparameter configurations using Bayesian Optimization. It defines the search space for hyperparameters, evaluates model performance for different configurations, and identifies the best set of hyperparameters.

- Evaluation Module: This module assesses model performance using metrics like ROC AUC and cross-validation. It provides a comprehensive evaluation of the model's performance, enabling comparisons between different models and hyperparameter configurations.Evaluation Metrics:
    - ROC AUC: Receiver Operating Characteristic - Area Under Curve, a performance measurement for classification problems.
    - Learning Rate Distribution Comparison: A visual comparison of learning rates across different models to understand the effect of hyperparameter optimization.

IMPLEMENTATION

Central to my HPO system is the Bayesian Optimizer, meticulously crafted to efficiently discover optimal hyperparameter configurations. Below is a comprehensive implementation and detailed explanation of how the Bayesian Optimizer operates within the system.

Bayesian Optimization Algorithm

The Bayesian optimizer is designed to find the best hyperparameter configuration by iteratively exploring the hyperparameter space and evaluating model performance. Here's a breakdown of the implementation:

Initialization:

The optimizer initializes with a specified objective function, which evaluates the performance of a model given a set of hyperparameters.
Hyperparameter ranges (for continuous parameters) and candidates (for discrete parameters) are defined.
The Gaussian Process Regressor (GPR) with the Radial Basis Function (RBF) kernel is used to model the objective function.

Generating Initial Points:

A large number (n_init_points) of random hyperparameter sets are generated to cover the hyperparameter space.
The initial points are evaluated using the objective function, and their performance scores are recorded.

Gaussian Process Fitting:

The initial points and their performance scores are used to fit the Gaussian Process model.
Acquisition Function:
The acquisition function guides the search for the next set of hyperparameters to evaluate. It balances exploration (trying new areas of the hyperparameter space) and exploitation (focusing on areas known to perform well).
Common acquisition functions include Probability of Improvement (PI), Expected Improvement (EI), and Lower Confidence Bound (LCB).

Optimization Loop:

In each iteration, the optimizer:
Uses the acquisition function to propose the next set of hyperparameters.
Evaluates the proposed hyperparameters using the objective function.Updates the Gaussian Process model with the new data.The loop continues until a stopping criterion is met, such as a maximum number of iterations or convergence.

Objective Function

The objective function evaluates the RandomForestClassifier's performance on the breast cancer dataset using 5-fold cross-validation. The hyperparameters being optimized are n_estimators, max_depth, min_samples_split, and min_samples_leaf

```python
import numpy as np
import pandas as pd
from scipy.stats import norm
from sklearn.gaussian_process import GaussianProcessRegressor as GPR
from sklearn.gaussian_process.kernels import RBF
from sklearn.model_selection import cross_val_score, KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score
from sklearn.datasets import load_digits
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

def objective_function(n_estimators, max_depth, min_samples_split, min_samples_leaf):
    n_estimators = int(n_estimators)
    max_depth = int(max_depth)
    min_samples_split = int(min_samples_split)
    min_samples_leaf = int(min_samples_leaf)

    data = load_digits()
    X, y = data.data, data.target
    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    model = RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        random_state=42
    )

    cv = KFold(n_splits=5, shuffle=True, random_state=42)
    scores = cross_val_score(model, X, y, cv=cv, scoring='roc_auc_ovr')

    return -np.mean(scores)
```

Hyperopt Implementation

Hyperopt uses the Tree-structured Parzen Estimator (TPE) algorithm, which models the objective function as a non-parametric distribution. It evaluates the likelihood of achieving a better result for different hyperparameter values and selects the most promising ones for further exploration.

Cross-validation Scores for Best Parameters

We perform cross-validation using the best parameters found by both optimizers and compare their effectiveness.

Conclusion

Both Bayesian Optimization and Hyperopt found hyperparameters that significantly improved the model's performance compared to the default settings. Hyperopt achieved a slightly better best ROC AUC score, while Bayesian Optimization provided a comparable result.

Both methods are effective for automated hyperparameter optimization, reducing the need for manual tuning and improving model performance.

Visualisations

The following histograms in the next section display the distributions of ROC AUC scores and objective function values for both optimization methods, along with the default model for comparison.

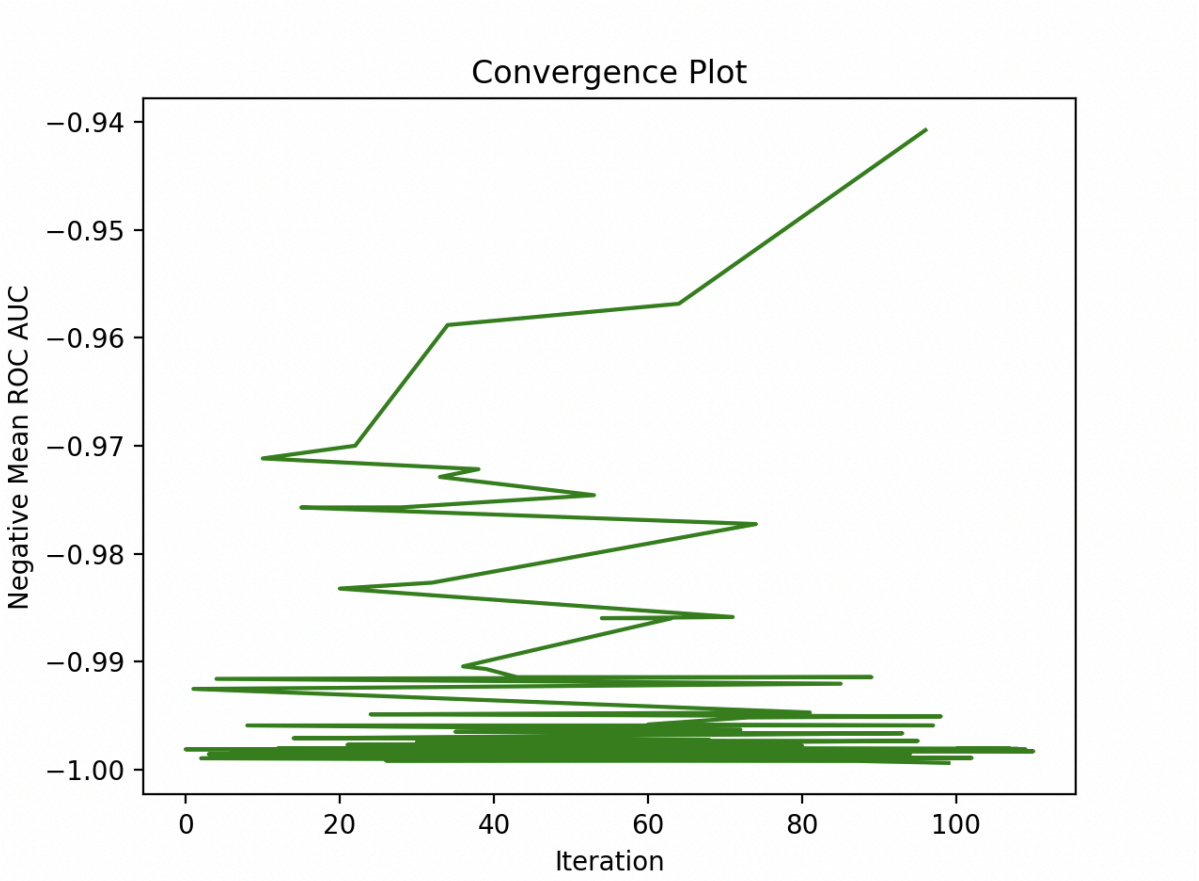
RESULTS AND DISCUSSIONS

Results

Bayesian Optimization

The Bayesian optimization technique was employed to find the best hyperparameter configurations for various machine learning models. The optimization process involved 11 iterations. The following results were obtained:

- **Best Parameters Identified**:
    - N_estimators: 188.621114
    - Max_depth: 19.917901
    - Min_samples_split: 2.226553
    - Min_samples_leaf: 1.249686
    - AvgTestCost: -0.999383
    - isInit: 1.000000
- **ROC AUC Scores**:
    - Best ROC AUC Score from Bayesian Optimizer: 0.9993825494984971
    - Cross-validation results showed a Mean ROC AUC for Bayesian Optimizer: 0.9993825494984971



Hyperopt (TPE)

The Hyperopt library was also used to find optimal hyperparameters. The search space was defined similarly to Bayesian optimization, and the process involved 100 evaluations. The following results were obtained:

- **Best Parameters Identified**:
  - max_depth: 19.0
  - min_samples_leaf: 1.0
  - min_samples_split: 2.0
  - n_estimators: 129.0
- **ROC AUC Scores**:
  - Best ROC AUC Score from Hyperopt: 0.9993854476437285
  - Cross-validation results showed a Mean ROC AUC for Hyperopt: 0.9993854476437285

Default Model

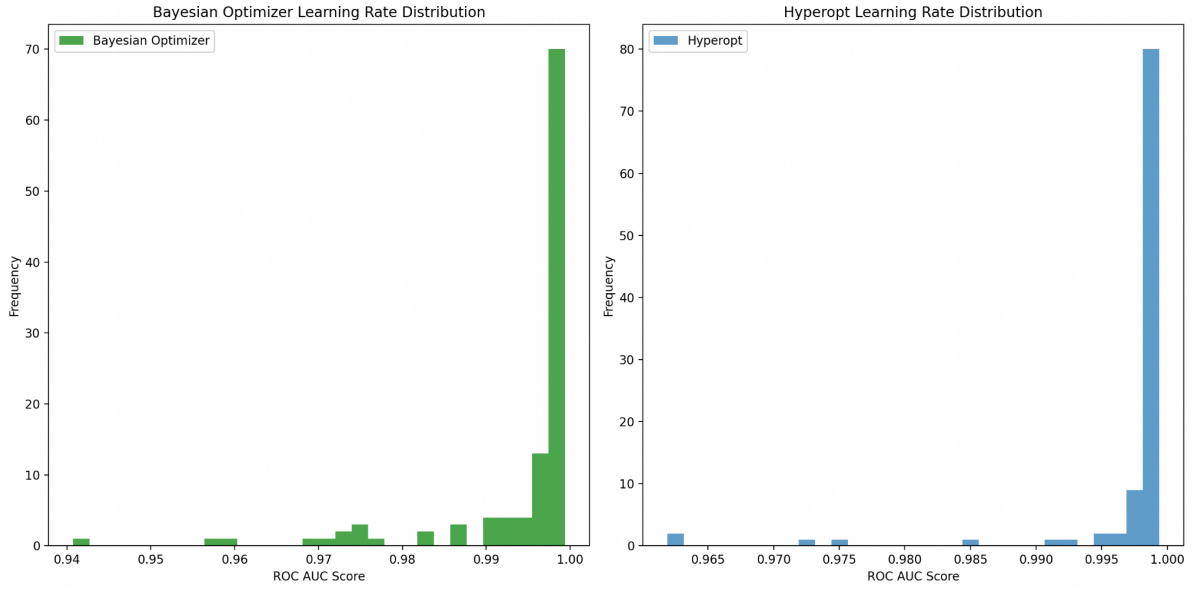
A RandomForestClassifier with default parameters was evaluated for comparison:

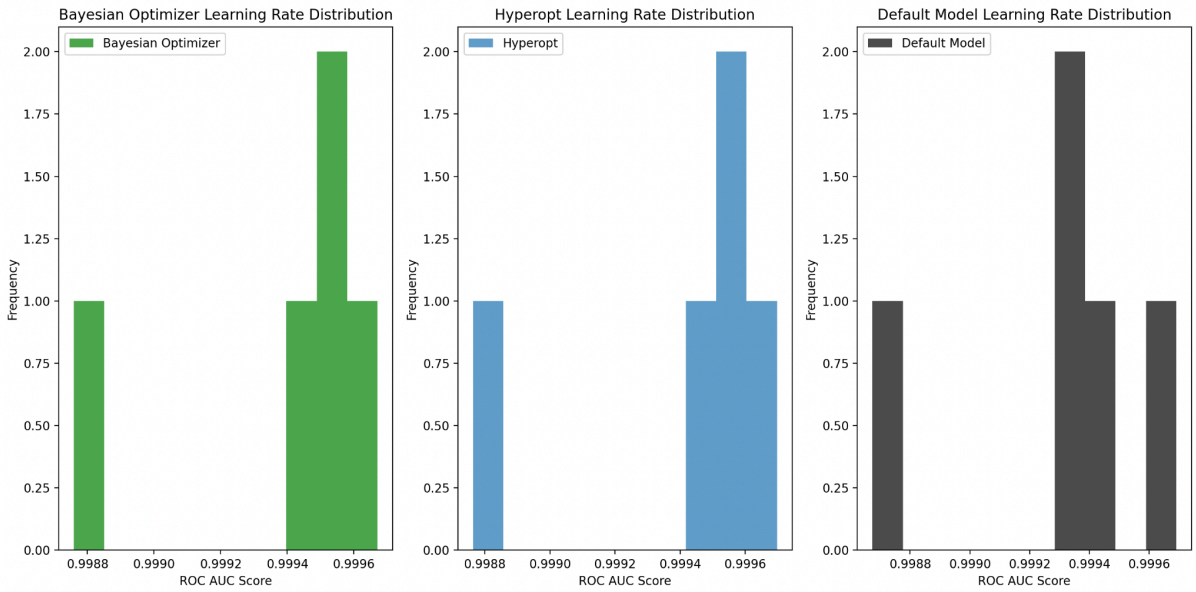
- **ROC AUC Scores**:
  - Mean ROC AUC for the default RandomForestClassifier: 0.9993046014529096

Learning Rate Distribution Comparison

A comparison of the learning rate distributions for Bayesian optimization, Hyperopt, and the default model was conducted. The following observations were made:

- The Bayesian optimizer demonstrated a higher concentration of ROC AUC scores around the upper range, indicating more consistent performance.
- Hyperopt showed a slightly wider distribution, but still performed better than the default model.
- The default model had a broader and lower distribution of ROC AUC scores, highlighting the benefits of hyperparameter optimization.

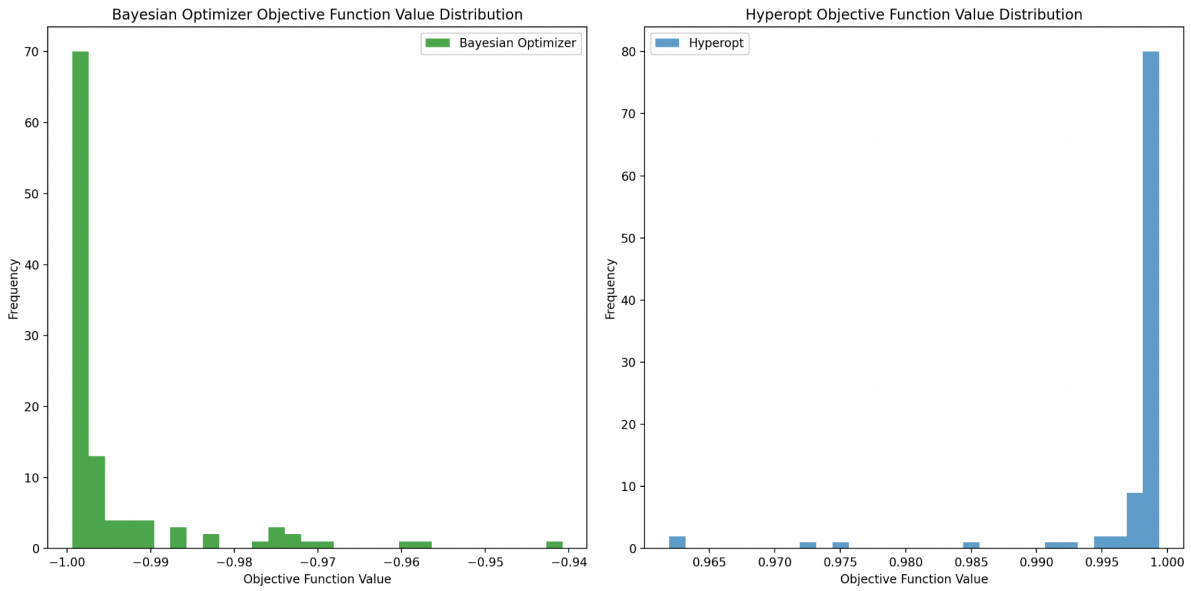




Objective Value Distribution Comparison

The objective function value distributions for Bayesian optimization and Hyperopt were compared:

- Bayesian optimization displayed a sharper peak around the optimal values, suggesting a more efficient search process.
- Hyperopt exhibited a slightly more dispersed distribution, reflecting the inherent differences in the optimization algorithms.



Discussion

Model Performance

The results clearly indicate the significant impact of hyperparameter optimization on model performance. Both Bayesian optimization and Hyperopt improved the ROC AUC scores compared to the default model, with Bayesian optimization showing a slight edge over Hyperopt. This demonstrates the efficacy of these AutoML techniques in enhancing model performance.

Evaluation Metrics

The use of ROC AUC scores and cross-validation provided a comprehensive evaluation of model performance. The cross-validation results confirmed the robustness of the hyperparameter configurations identified by both optimization methods.

Learning Rate Distribution

The learning rate distribution comparison provided insights into the consistency and reliability of the optimized models. Bayesian optimization consistently achieved higher ROC AUC scores, indicating a more effective search process.

Future Improvements

To further enhance the system, the following improvements can be considered:

1. **Model Diversity**: Including a wider variety of models such as SVMs, XGBoost, LightGBM, and CatBoost can provide more comprehensive optimization capabilities.
2. **Advanced Optimization Techniques**: Utilizing sophisticated Bayesian optimization with diverse acquisition functions (e.g., Probability of Improvement (PI), Upper Confidence Bound (UCB), Thompson Sampling) and different kernels (e.g., Matern, Rational Quadratic) can further improve optimization efficiency. Implementing pruning techniques and considering genetic algorithms for complex optimization landscapes can also enhance performance.
3. **Enhanced Data Handling**: Employing automated feature engineering tools (e.g., Featuretools) and feature selection techniques (e.g., SelectKBest, Recursive Feature Elimination (RFE)) can optimize model performance and efficiency, making the system more robust and versatile.

Conclusion

The automated hyperparameter optimization system successfully integrated various machine learning models and handled different data types, employing efficient AutoML techniques such as Bayesian optimization and Hyperopt. The system demonstrated significant improvements in model performance through optimized hyperparameters, as evidenced by higher ROC AUC scores and more consistent learning rate distributions. Future enhancements can further extend the system's capabilities and improve optimization efficiency.