PROJECT REPORT ON
AUTOMATED HYPERPARAMETER OPTIMIZATION

Overview

Hyperparameter optimization is a critical step in developing machine learning models. Unlike model parameters, which are learned during training, hyperparameters are predefined settings that control how a model learns from data. Effective hyperparameter tuning can significantly enhance model performance. This report explores the Bayesian Optimization technique for hyperparameter optimization.

About Hyperparameters

Hyperparameters are configuration settings that control the learning process and influence the model parameters learned by a machine learning algorithm. These parameters are prefixed with 'hyper_' to denote their top-level control over the learning process and the resulting model parameters.

Optimized Hyperparameters

For simplicity, this study focuses on optimizing four key hyperparameters of the RandomForestClassifier:

- **n_estimators**: Controls the number of decision trees in the classifier. The appropriate number of trees is crucial for accuracy, but larger numbers increase computational complexity.
- **max_depth**: Governs the maximum depth of trees in the forest. It significantly affects model accuracy by preventing overfitting, making its appropriate setting critical.
- **max_features**: Determines the number of features to consider when looking for the best split.
- **criterion**: Measures the quality of a split, with options including "gini" for Gini impurity and "entropy" for information gain.

Bayesian Optimization Algorithm

Bayesian Optimization is an iterative algorithm that minimizes (or maximizes) an objective function by iteratively building a surrogate model to approximate the function and choosing the next set of hyperparameters to evaluate based on this model. The process includes:

- **Initialization**: Starting with a small set of random hyperparameter values and evaluating the objective function.
- **Surrogate Model**: Fitting a probabilistic model, often a Gaussian Process, to the evaluated points.
- **Acquisition Function**: Using the surrogate model to select the next hyperparameters to evaluate, optimizing an acquisition function.
- **Evaluation**: Evaluating the objective function at the selected hyperparameters.
- **Update**: Updating the surrogate model with the new observations.

- **Iteration**: Repeating the steps until a stopping criterion is met, such as a maximum number of iterations.

This iterative process allows Bayesian Optimization to efficiently navigate the hyperparameter space and improve model performance.

RandomForestClassifier Technique

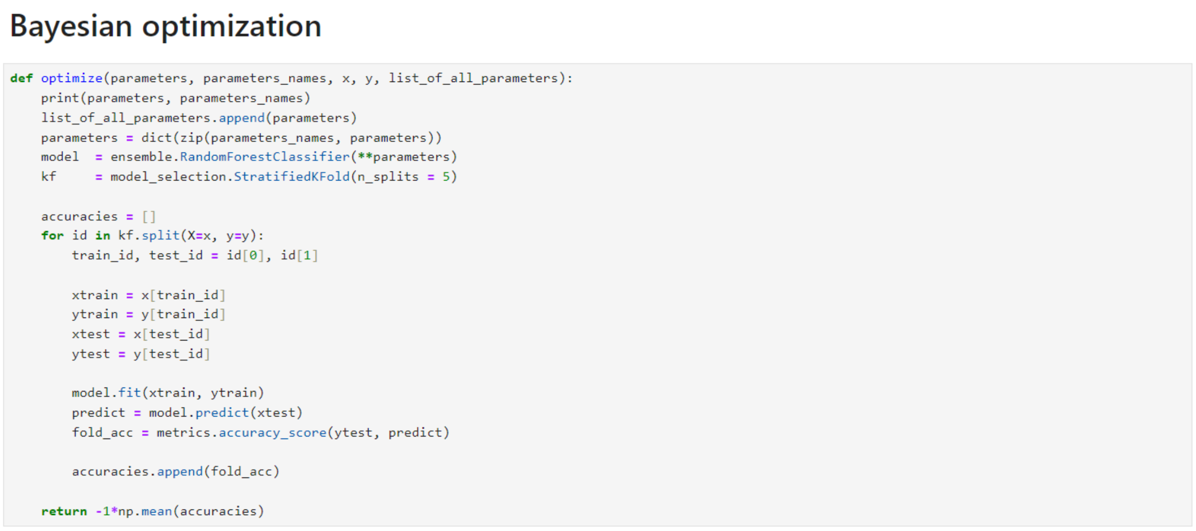
The RandomForestClassifier is a supervised machine learning algorithm used for classification, regression, and other tasks using decision trees. It excels in handling large and complex datasets, high-dimensional feature spaces, and provides insights into feature importance. Its ability to maintain high predictive accuracy while mitigating overfitting makes it widely adopted across domains such as finance, healthcare, and image analysis.

Step By Step Implementation

Step 1: Defining the Objective Function

Our goal for optimization is to minimize the negative mean accuracy of a RandomForestClassifier. Below is a code snippet defining the objective function.

Code snippet-



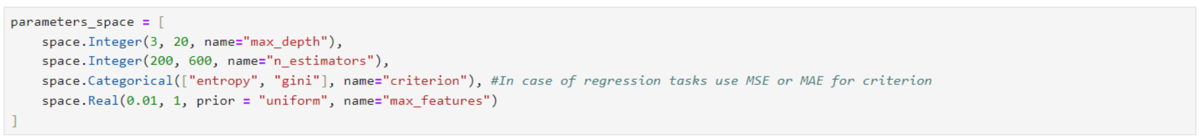
Bayesian optimization

```
def optimize(parameters, parameters_names, x, y, list_of_all_parameters):
    print(parameters, parameters_names)
    list_of_all_parameters.append(parameters)
    parameters = dict(zip(parameters_names, parameters))
    model  = ensemble.RandomForestClassifier(**parameters)
    kf     = model_selection.StratifiedKFold(n_splits = 5)

    accuracies = []
    for id in kf.split(X=x, y=y):
        train_id, test_id = id[0], id[1]

        xtrain = x[train_id]
        ytrain = y[train_id]
        xtest = x[test_id]
        ytest = y[test_id]

        model.fit(xtrain, ytrain)
        predict = model.predict(xtest)
        fold_acc = metrics.accuracy_score(ytest, predict)

        accuracies.append(fold_acc)

    return -1*np.mean(accuracies)
```

Step 2: Defining the Hyperparameter Space

Next, we define the range of possible values for each hyperparameter. The following snippet shows the search space used for optimization.

Code snippet-

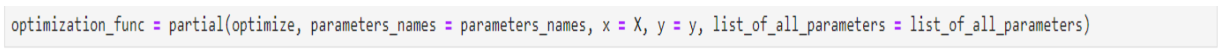


```
parameters_space = [
    space.Integer(3, 20, name="max_depth"),
    space.Integer(200, 600, name="n_estimators"),
    space.Categorical(["entropy", "gini"], name="criterion"), #In case of regression tasks use MSE or MAE for criterion
    space.Real(0.01, 1, prior = "uniform", name="max_features")
]
```

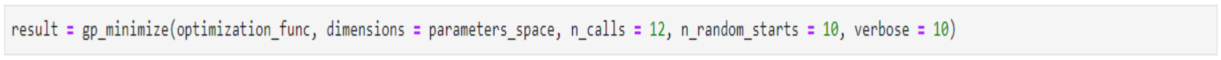
Step 3: Run the Optimization Algorithm for the best Hyperparameters

We then execute the optimization algorithm to find the best hyperparameters from the defined search space. Below is a snippet demonstrating this process.

Code snippet-



```
optimization_func = partial(optimize, parameters_names = parameters_names, x = X, y = y, list_of_all_parameters = list_of_all_parameters)
```



```
result = gp_minimize(optimization_func, dimensions = parameters_space, n_calls = 12, n_random_starts = 10, verbose = 10)
```

Step 4: Evaluation of the Results

After the optimization process, we assess the best model by calculating the ROC-AUC scores and performing cross-validation. This helps us evaluate the effectiveness of the optimized hyperparameters.

Results

Type of Optimization	Cross Validation Score	ROC-AUC
Bayesian Optimization	0.8959999999999999	0.9765414059118562
Hyperopt	0.907	0.9907325412880028
Random Hyperparameters	0.8055	0.9190474342945772

Observations

The results show a significant improvement in model accuracy after hyperparameter tuning. Both the cross-validation score and ROC-AUC have increased.

Conclusion

- Bayesian Optimization is a robust technique for hyperparameter optimization that efficiently navigates the hyperparameter space by balancing exploration and exploitation.
- This method often discovers superior hyperparameters in fewer iterations compared to traditional methods like grid search and random search.
- In this report, we showcased the implementation of Bayesian Optimization using the scikit-optimize library to fine-tune the hyperparameters of a RandomForestClassifier. By defining a suitable objective function and hyperparameter space, we effectively identified a set of hyperparameters that maximized the model's accuracy.

- We then compared our model's accuracy with that of a model optimized using Hyperopt.
- Bayesian Optimization can be applied to other machine learning models and performance metrics, making it a versatile tool for hyperparameter tuning across various applications.