**Hyperparameter Optimization for XGBoost Model**

**1. Introduction**

This document describes the process of optimizing hyperparameters for an **XGBoost Classifier** using different tuning techniques: **Grid Search, Random Search, and Bayesian Optimization**. The goal is to improve model performance by selecting the best combination of hyperparameters.

**2. Dataset Description**

The dataset used for this experiment is the **Breast Cancer Dataset** from sklearn.datasets. It contains:

* **Features**: 30 numerical attributes.
* **Target Variable**: Binary classification (Malignant = 0, Benign = 1).

The dataset is split into **80% training** and **20% testing**.

**3. Model Selection**

The model chosen for this optimization is **XGBoost (XGBClassifier)**, a powerful gradient boosting algorithm known for its high performance on structured data.

**4. Hyperparameter Tuning Methods**

**4.1 Grid Search**

**Grid Search** exhaustively searches through a predefined set of hyperparameters to find the best combination.

**Hyperparameters tested:**

* n\_estimators: [50, 100, 150]
* max\_depth: [3, 5, 7]
* learning\_rate: [0.01, 0.1, 0.2]

**Best Parameters Found:** grid\_search.best\_params\_

**4.2 Random Search**

**Random Search** randomly selects a combination of hyperparameters and evaluates them, allowing for faster tuning than Grid Search.

**Hyperparameters tested:**

* n\_estimators: Random values from [50, 100, 150, 200]
* max\_depth: Random values from [3, 5, 7, 9]
* learning\_rate: Random values from [0.01, 0.075, 0.15, 0.225, 0.3]

**Best Parameters Found:** random\_search.best\_params\_

**4.3 Bayesian Optimization**

**Bayesian Optimization** utilizes probabilistic models to predict the best set of hyperparameters instead of testing all possible combinations.

**Hyperparameters tested:**

* n\_estimators: Range (50 - 200)
* max\_depth: Range (3 - 10)
* learning\_rate: Range (0.01 - 0.3)

**Best Parameters Found:** bayesian\_search.best\_params\_

**5. Model Evaluation**

Each optimized model was evaluated using **Accuracy, Precision, Recall, and F1-Score**. Below are the performance results:

| **Tuning Method** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- | --- |
| Grid Search | 0.953216 | 0.954545 | 0.972222 | 0.963303 |
| Random Search | 0.970760 | 0.981308 | 0.972222 | 0.976744 |
| Bayesian Optimization | 0.959064 | 0.963303 | 0.972222 | 0.967742 |

**6. Conclusion**

Based on the evaluation metrics, **Random Search** achieved the highest performance. Bayesian Optimization proved to be the most efficient in terms of finding the best hyperparameters while reducing computation time compared to Grid Search.

**6.1 Future Work**

* Try tuning additional hyperparameters like subsample and colsample\_bytree.
* Experiment with other boosting algorithms like **LightGBM**.
* Implement **Optuna** for more efficient optimization.