Tree Pruning & Hyper Parameter Optimization

Class Content

- Decision Tree Pruning
- Pre-Pruning in Decision Tree
- Post-Pruning in Decision Tree
- Ensemble Machine Learning
- Bagging Algorithms
- Boosting Algorithms

- Random Forest In-depth
- Hyper Parameter Tuning using:
  - Randomized Search CV
  - Grid Search CV
- Python Implementations

Pruning Basics

Neural Network Pruning:

- Weight Pruning: Removing connections with small weights or setting them to zero.
- Neuron Pruning: Removing entire neurons from the network.
- Layer Pruning: Removing entire layers of the neural network.

Decision Tree Pruning:

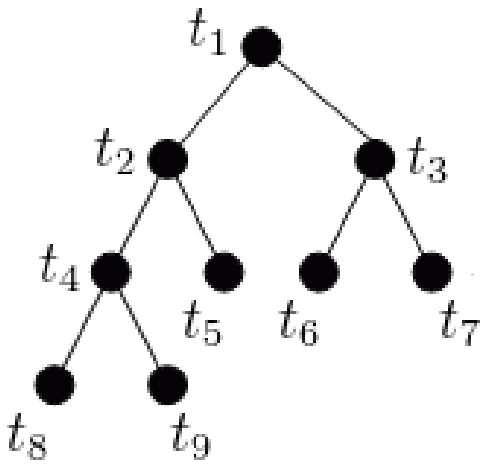
- Pre-Pruning: It involves setting a threshold on certain parameters (e.g., the maximum depth of the tree, minimum number of samples per leaf) while growing the tree. The tree construction stops when the threshold is reached, preventing the tree from growing too large.

- Post-Pruning: This technique involves growing the tree to its full depth and then removing nodes or branches that do not provide significant predictive power. Pruning decisions are made based on metrics like cross-validation accuracy or information gain.

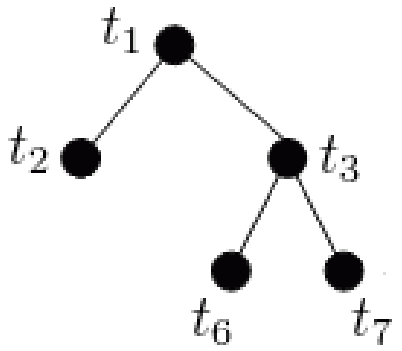
For a big dataset, pre-pruning is generally recommended over post-pruning. Pre-pruning refers to stopping the tree construction process early before it reaches its maximum depth, while post-pruning involves building the complete tree and then removing or collapsing nodes to reduce overfitting.

Decision Tree Pruning

Main Tree



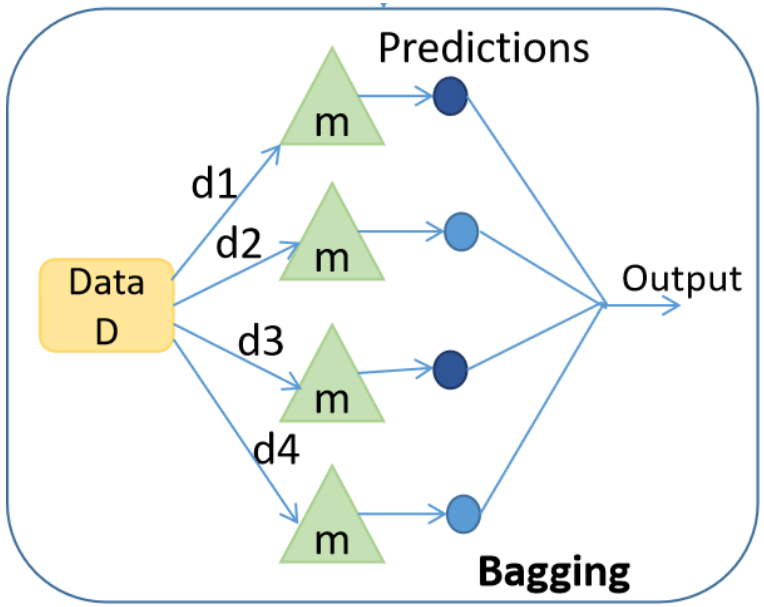
After Pruning

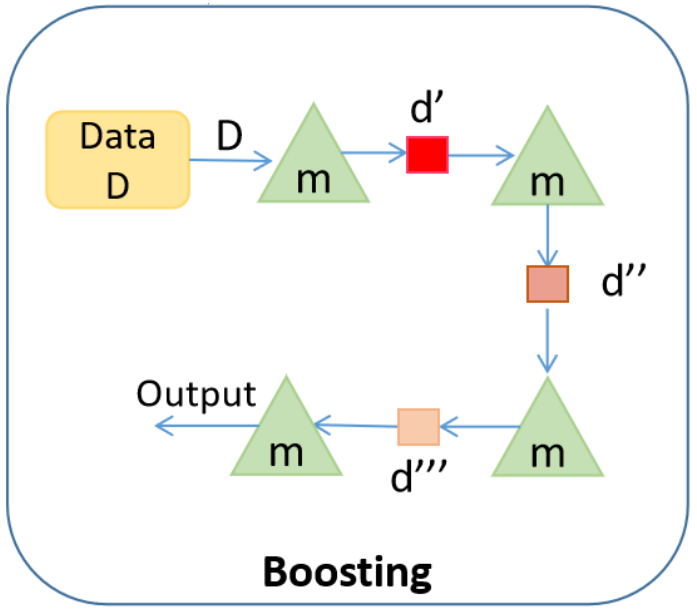


Ensemble Learning

- Ensemble learning is a machine learning technique that involves combining the predictions of multiple individual models (learners) to create a more robust and accurate model.

- The idea behind ensemble learning is that by combining several weak or moderately performing models, the ensemble can achieve better overall performance and generalization compared to any single model in the ensemble.





Ensemble Algorithms

Bagging or Bootstrap Aggregation

- Bagged Decision Trees
- Random Forest Classifier or Regressor
- Extra Trees

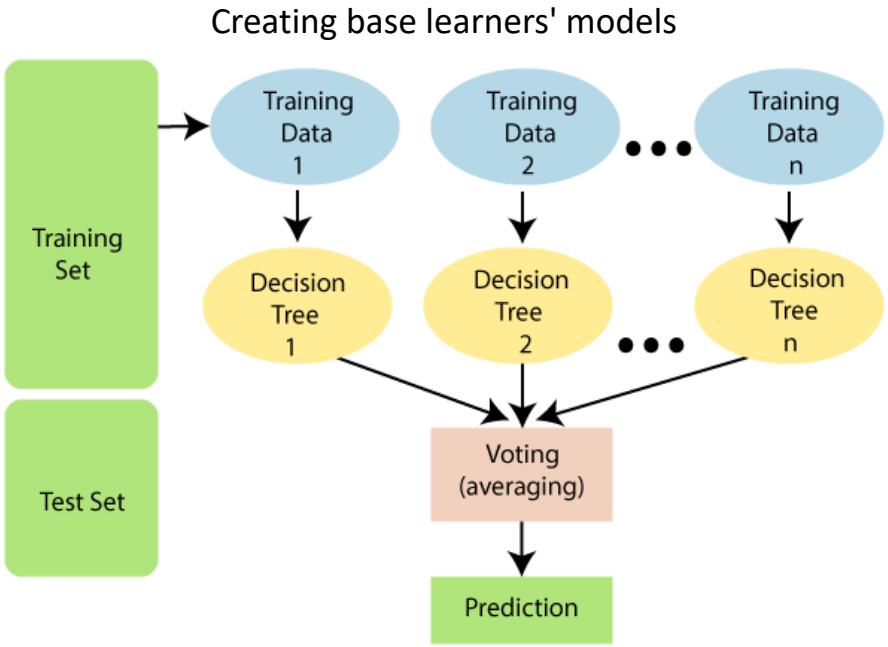
Boosting

- AdaBoost
- Gradient Boosting
- XGBoost
- CatBoost

Random Forest

Random forests or random decision forests is an **ensemble** learning method for classification, regression and other tasks that operates by constructing a **multitude of decision trees** at training time. For classification tasks, the output of the random forest is the class selected by **most trees.**

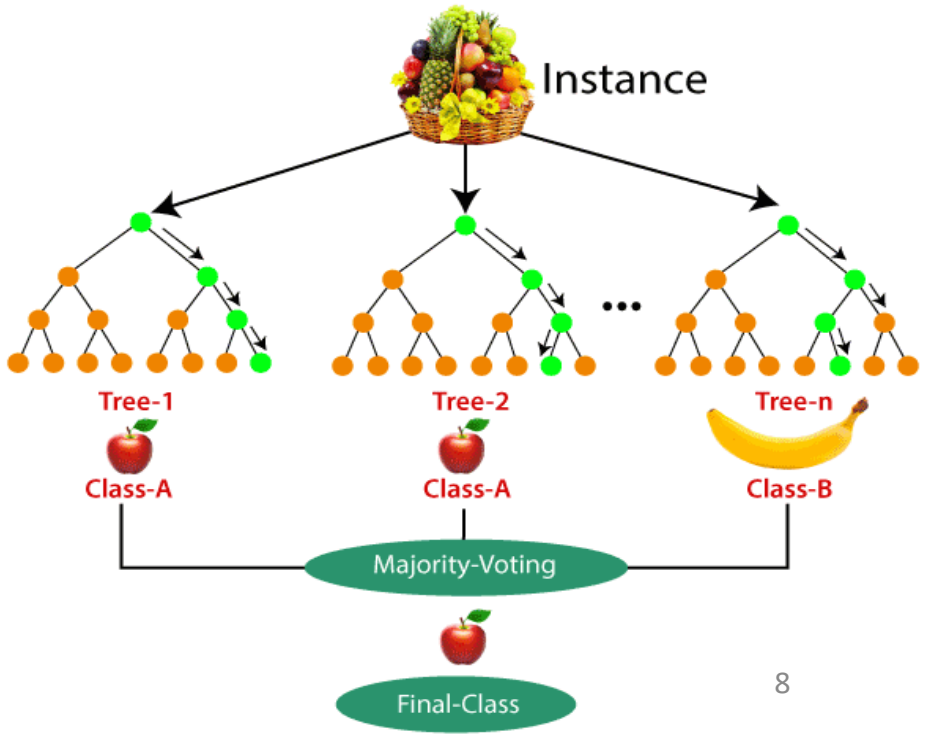
Random Forest is a classifier that contains several **decision trees** on various subsets of the given dataset and takes the **average** to improve the predictive accuracy of that dataset. Instead of relying on **one decision tree**, the random forest takes the prediction from each tree and based on the **majority votes of predictions**, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and **prevents the problem of overfitting**.



Steps involved in random forest algorithm:

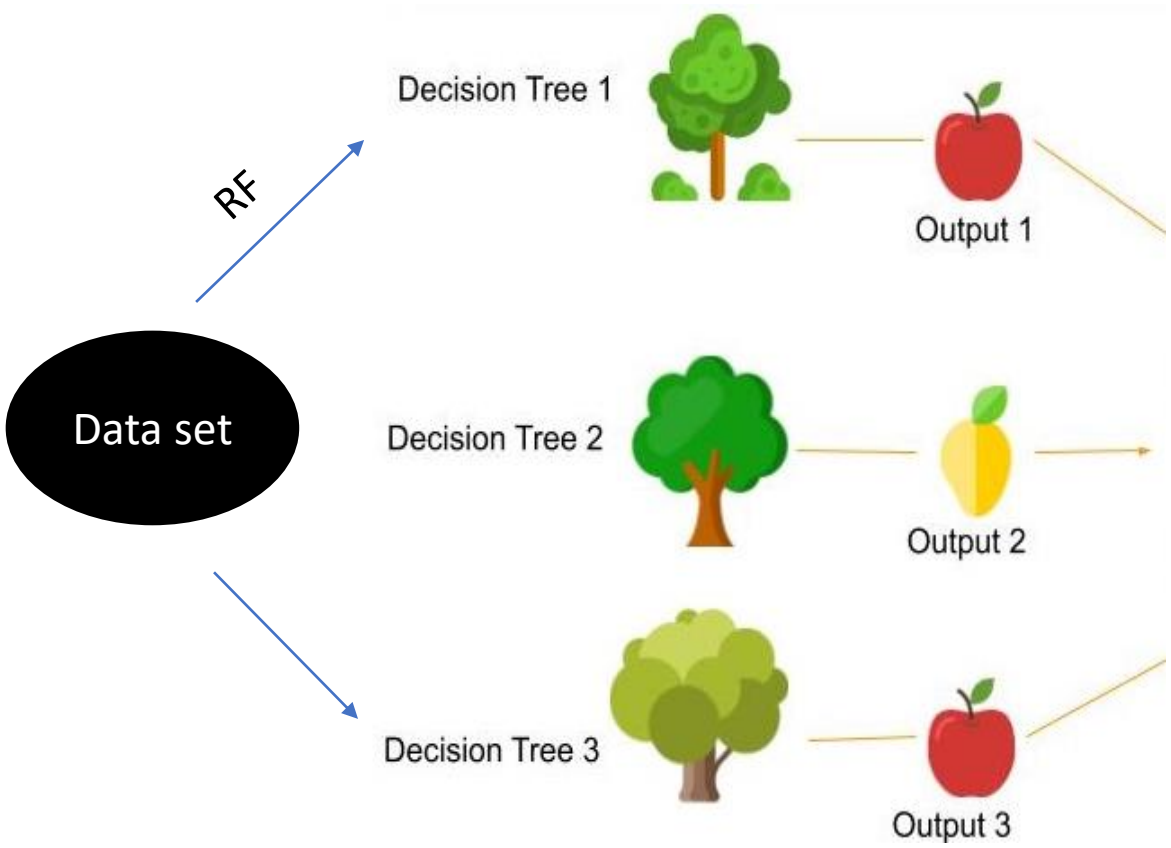
Step 1: In Random forest n number of random records are taken from the data set having **k number** of records.
Step 2: Individual **decision trees** are constructed for each sample.
Step 3: Each decision tree will generate **an output**.
Step 4: Final output is considered based on **Majority Voting** or Averaging for Classification and regression respectively.

For example: consider the fruit basket as the data as shown in the figure below. Now n number of samples are taken from the fruit basket and an **individual decision tree** is constructed for each sample. Each decision tree will generate an output as shown in the figure. The final output is considered based on **majority voting**. In the below figure you can see that the majority decision tree gives output as an apple when compared to a banana, so the final output is taken as an apple.



Random Forest

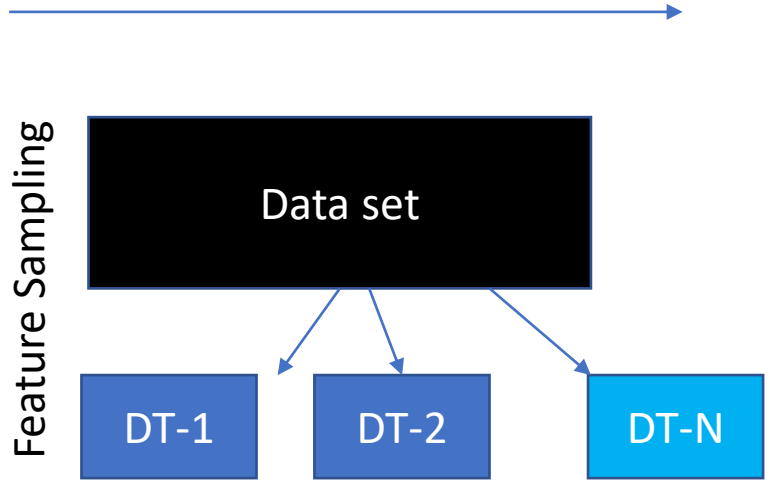
How Random Forest Works with a Simple Visualization



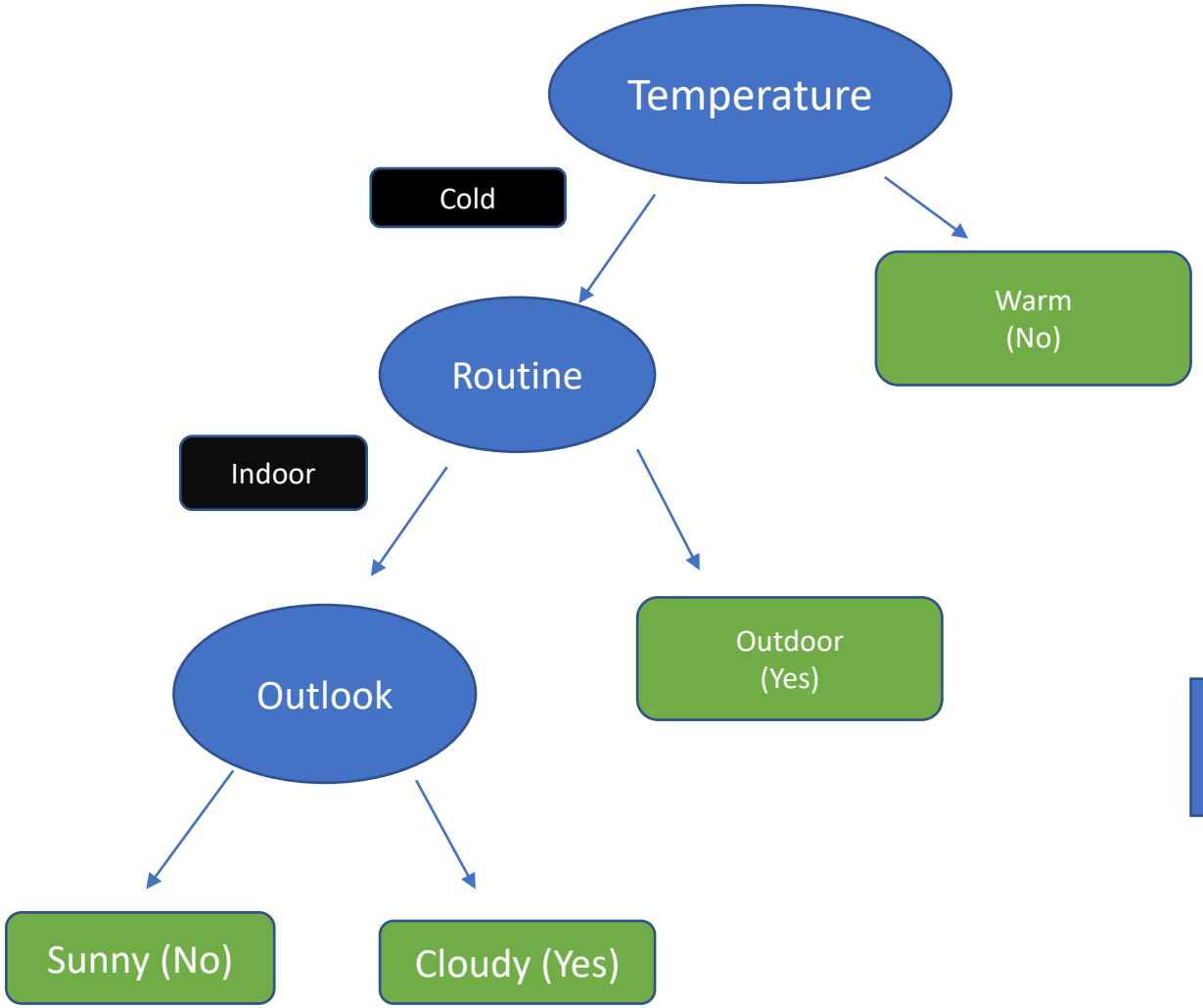
Majority Voting

Final Output

- Base Learners Model
  - Row Sampling
  - Feature Sampling



Decision Tree Vs. Random Forest



Sunny, Cold , Indoor= ??

The answer is ‘No’

Advantages and Disadvantages of Random Forest

- It reduces overfitting in decision trees and helps to improve the accuracy
- It is flexible to both classification and regression problems.
- It works well with both categorical and continuous values
- It automates missing values present in the data
- Normalizing of data is not required as it uses a rule-based approach.

However, despite these advantages, a random forest algorithm also has some **disadvantages**:

- It requires much computational power as well as resources as it builds numerous trees to combine their outputs.
- It also requires much time for training as it combines a lot of decision trees to determine the class.
- Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable.

Applications of Random Forest

- Detect and Predict the Drug Sensitivity of a Medicine
- Identify a Patient's Disease by Analyzing their Medical Records
- Predict Estimated Loss or Profit while Purchasing a Particular Stock
- Banking Industry, Credit Card Fraud Detection
- Customer Segmentation, Predicting Loan
- Healthcare and Medicine Cardiovascular Disease Prediction
- Diabetes Prediction, Breast Cancer Prediction
- Stock Market Prediction, Stock Market Sentiment Analysis, Bitcoin Price Detection, E-Commerce Product
- Recommendation Price, Optimization Search Ranking

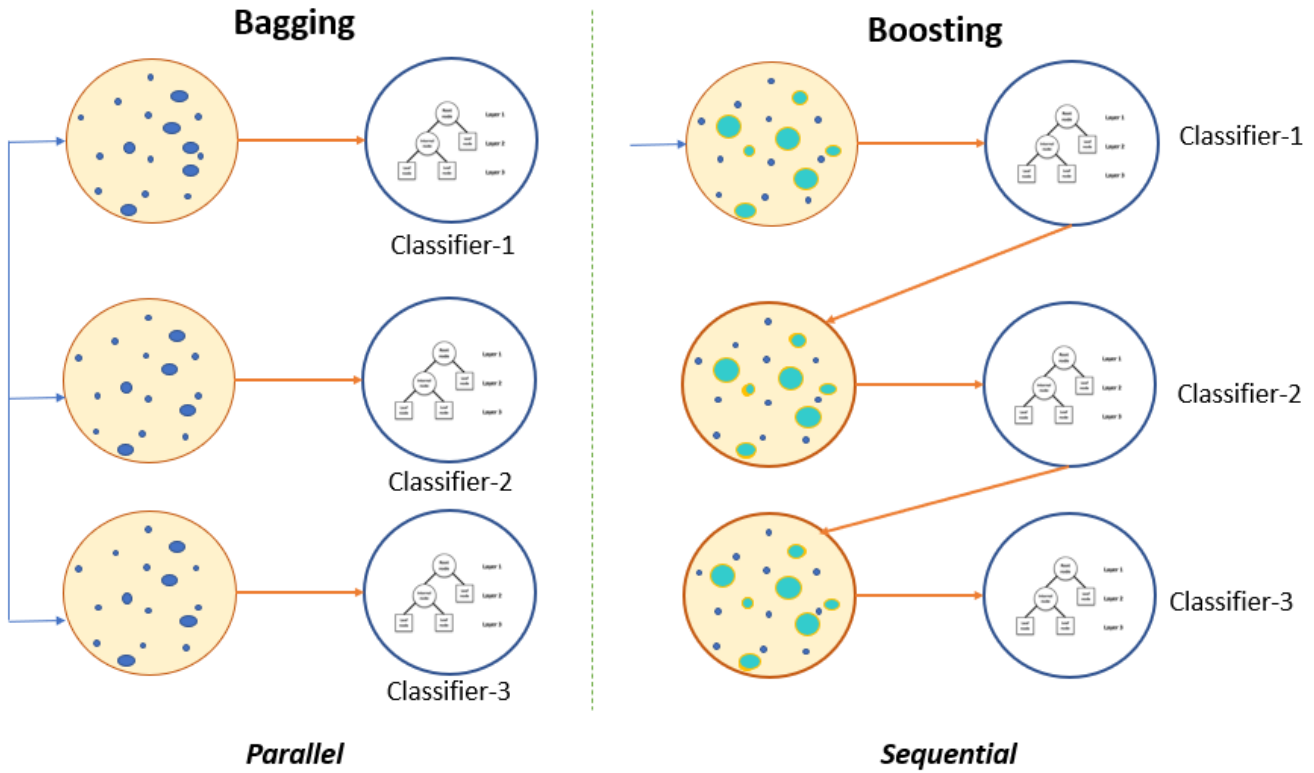
Bagging Vs. Boosting

Bagging (Bootstrap Aggregating):

- Data Sampling Technique
- Model Independence
- Parallel Training
- Aggregation Method

Boosting:

- Data Weighting
- Sequential Training
- Model Dependence
- Error-Based Weights



Hyper Param Optimization

Hyperparameter optimization, also known as hyperparameter tuning, is the process of finding the best set of hyperparameters for a machine learning model to maximize its performance on a given task. Hyperparameters are parameters that are set before the learning process begins and control aspects of the learning algorithm's behavior, rather than being learned from the data itself.

Grid Search: Grid search involves defining a grid of possible hyperparameter values and evaluating the model's performance for all combinations of these values. It can be computationally expensive, but it exhaustively searches the hyperparameter space.

Random Search: Random search randomly samples hyperparameter values from predefined ranges. It is computationally less expensive than grid search and often provides similar or even better results.

Bayesian Optimization: Bayesian optimization uses probabilistic models to predict the performance of the model for different hyperparameter values and focuses on exploring regions that are likely to yield better results.

Let’s do this with Python….