Hyperparameter Tuning

Parameter Vs Hyperparameter Tuning

Parameter

Parameters are the internal variable of a model that are learned from the data during the training process. They define the model's representation of the underlying patterns in the data.

For example:

- In a linear regression model, the parameter are the coefficient of the predictors.
- In a neural network, the parameters are the weights and biases of the nodes.
- In a decision tree, the parameter are the split point and split criteria at each node.

The goal of the training process is to find the optimal values of these parameters, which minimize the discrepancy between the model's prediction and the actual outcome.

Hyperparameter

In machine learning, hyperparameter are parameter whose values are set before the learning process begins. These parameters are not learned from the data and must be predefined. They help in controlling the learning process and can significantly influence the performance of the model.

- In a neural network, hyperparameters might include the learning rate, the number of layers in the network, or the number of nodes in each layer.
- In a support vector machine, the regularization parameter C or the kernel type can be considered as hyperparameter.
- In a decision tree, the maximum depth of the tree is a hyperparameter.

The best value for hyperparameter often cannot be determined in advance, and must be found through trial and error.

Why the word 'hyper'?

The choice of the word is primarily a naming convention to differentiate between the two types of values (internal parameters and guiding parameters) that influence the behaviour of a machine learning model. It's also a nod to the fact that the role they play is a meta one, in the sense that they control the structural aspects of the learning process itself rather than being part of the direct pattern-finding mission of the model.

GridSearch CV

GridSearch CV refers to an algorithm that performs an exhaustive search over a specified grid of hyperparameters, using cross-validation to determine which hyperparameter combination gives the best model performance.

1000 dataset

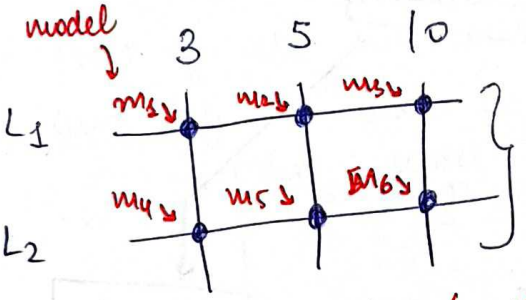
cgpa	iq	placement
10	100	1
9	90	0
8	80	1

→ model → KNN (Y/N)

↳ n_neighbour = 3, 5, 10

metric = L1, L2

algo = kd, tree---



all points represent the all the posible combination.

Knn (n-n = 3, L1)
Knn (n-n = 5, L1)
Knn (n-n = 10, L1)
Knn (n-n = 3, L2)
Knn (n-n = 5, L2)
Knn (n-n = 10, L2)

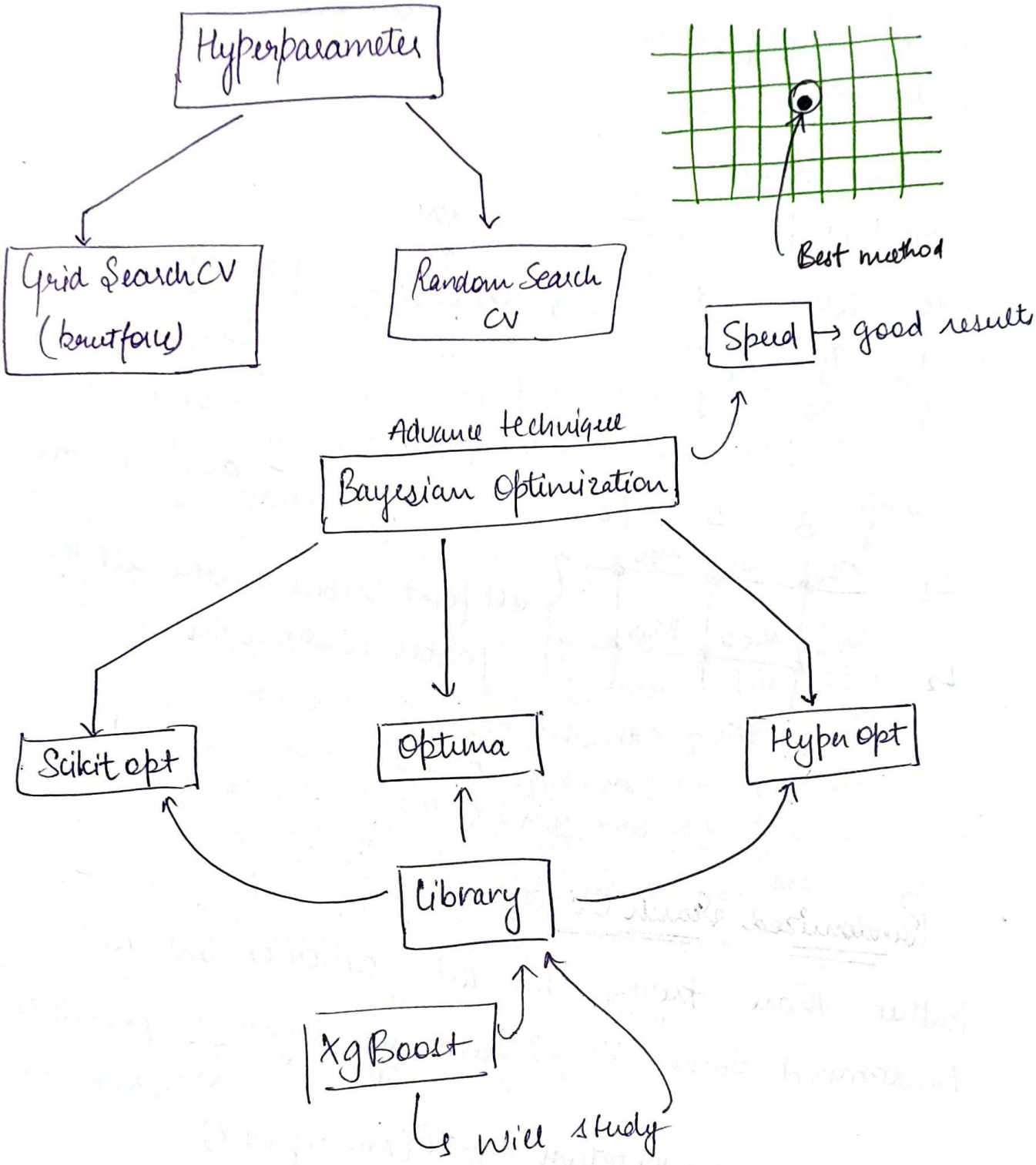
Randomized Search CV

Rather than trying the all posibilities but in Randomized Search CV I randomly tryout X posibilites
↳ 10, 20, or 40

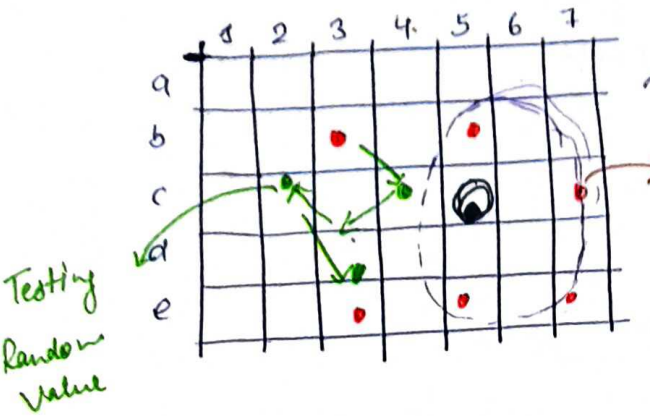
Advantage → Computation degrees (Not expensive)
→ Result fast

Disadvantage: Very high chance we'll not find best case because of limited time.

Can this be Improved?



Advance technique Intitution



→ Bayesian optimization

→ Random test

For eg:- Recipie

* So, we have to find that type of combination of salt and chilli which increase or gave good taste
  - Salt
    - 1 table spoon, 2 table spoon, 3 table spoon.
  - Chilli
    - 0.5 table spoon, 1 TS, 2 TS

* First of all we select some random values and try it and check the taste.

Salt	Chilli
1 TS	2 TS
–	–
–	–
–	–
–	–
–	–

} 10 random test and check the taste.

* With the help of 10 Random values, we can analyse the 10 Random values after increase or decrease salt and chillie. and analyse whether taste decrease or increase after changig the value of salt and chilli individualy.

* And then we use explore and exploit method.
  - exploit → means we know something about system then we do exploit.
  - explore → and explore unknown value.

* basically if know about one value like we know less salt is enough for good taste. So, know we explore different value of chilli.