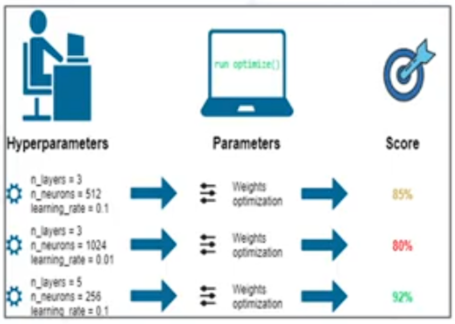
Hyperparameter Optimization :-

Machine Learning models are composed of two different types of parameters:
- **Hyperparameters** = are all the parameters which can be arbitrarily set by the user before starting training (eg. number of estimators in Random Forest).
- **Model parameters** = are instead learned during the model training (eg. weights in Neural Networks, Linear Regression).

The model parameters define how to use input data to get the desired output and are learned at training time. Instead, Hyperparameters determine how our model is structured in the first place.

Machine Learning models tuning is a type of optimization problem. We have a set of hyperparameters and we aim to find the right combination of their values which can help us to find either the minimum (eg. loss) or the maximum (eg. accuracy) of a function
This can be particularly important when comparing how different Machine Learning models performs on a dataset. In fact, it would be unfair for example to compare an SVM model with the best Hyperparameters against a Random Forest model which has not been optimized.



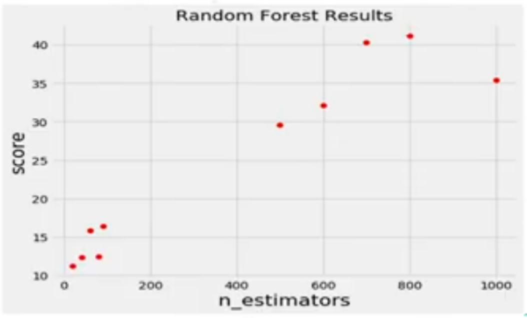
The aim of hyperparameter optimization in machine learning is to find the hyperparameters of a given machine learning algorithm that return the best performance as measured on a validation set.

f(x) - an objective score to minimize— such as RMSE or error rate— evaluated on the validation set;
x* - is the set of hyperparameters that yields the lowest value of the score
x - can take on any value in the domain X.
In simple terms, we want to **find the model hyperparameters that yield the best score on the validation set metric.**

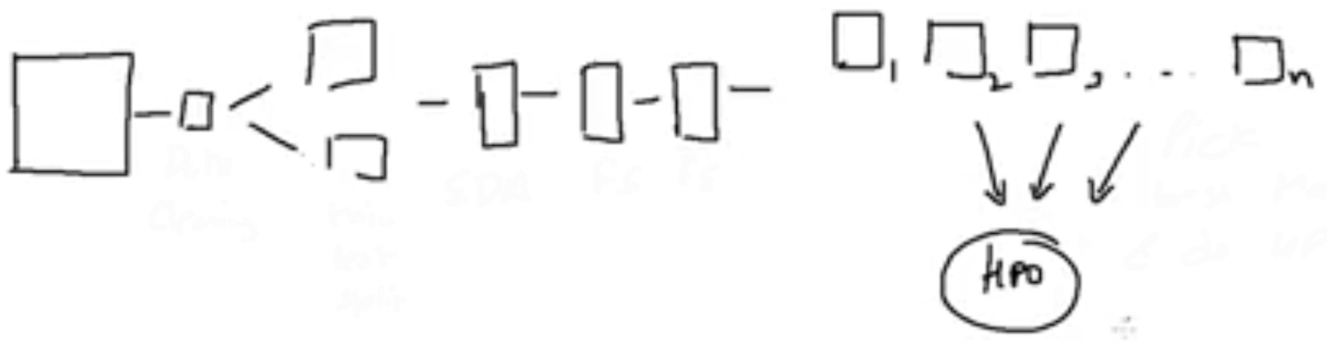
$$x^{\star} = \arg\min_{x \in \mathcal{X}} f(x)$$

Following are four common methods of hyperparameter optimization for machine learning in order of increasing efficiency:

- Grid search
- Manual
- Random search
- Bayesian model-based optimization



Flow:-



① Manual HPO:-

Manual choose set of values based on experience or random

We don't use this.

② Grid Search CV:-

← DT HP

criterion = gini, entropy — 2
max_depth = 2, 3, 4, 5, 10 — 5
min_samples_split :- 2, 5, 10, 20 — 4
min_samples_leaf :- 1, 10, 20, 50 — 4

total 160 Combinations

Grid Search will run those 160 Combinations & find best parameters.

dis:- Computationally Heavy

③ Random Search CV :-

criterion = gini, entropy → 2
max_depth = 2, 3, 4, 5, 10, 20, 100, 200 → 8
min_samples_split :- 2, 5, 10, 200, 100, 500 → 6
min_samples_leaf :- 1, 10, 20, 50, 100 → 5

$$\begin{tikzcd} \text{n\_iter : 50} \arrow[r] & \text{400 Combinations.} \end{tikzcd}$$

randomly pick 50 Combinations

↓

Give best Parameters.

2, 100, 200

108 value near 100

We get depth = 100

↓

depth = [10 20 30 40 70 80 100 120]

Again apply Random Search

↓

GS → [75 76 77 80 82] ← depth

Random Search, Grid Search is the most expensive Process.