XGBoost

XGBoost short form for **eXtreme Gradient Boosting** is an advanced machine learning algorithm designed for efficiency, speed and high performance.

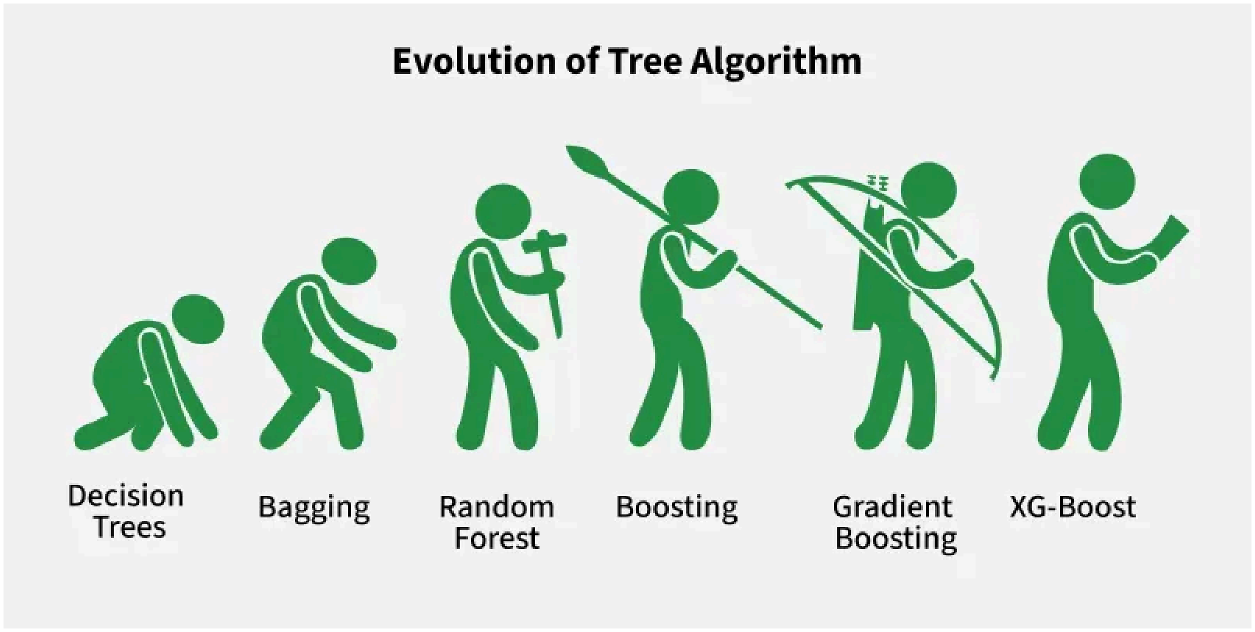
- XGBoost uses **decision trees** as its base learners and combines them sequentially to improve the model's performance. Each new tree is trained to correct the errors made by the previous tree and this process is called boosting.
- It has built-in parallel processing to train models on large datasets quickly. XGBoost also supports customizations allowing users to adjust model parameters to optimize performance based on the specific problem.

How XGBoost Works?

It builds decision trees sequentially with each tree attempting to correct the mistakes made by the previous one. The process can be broken down as follows:

1. **Start with a base learner**: The first model decision tree is trained on the data. In regression tasks this base model simply predicts the average of the target variable.
2. **Calculate the errors**: After training the first tree the errors between the predicted and actual values are calculated.
3. **Train the next tree**: The next tree is trained on the errors of the previous tree. This step attempts to correct the errors made by the first tree.
4. **Repeat the process**: This process continues with each new tree trying to correct the errors of the previous trees until a stopping criterion is met.

5. **Combine the predictions**: The final prediction is the sum of the predictions from all the trees.



1 What problem did XGBoost solve?

Classic Gradient Boosting problems:
- Slow
- Overfitting
- Hard to tune
- No regularization

XGBoost fixed all of these.

2 XGBoost explained using spam example

Step 1: Initial prediction

> "Most emails are spam with 60% probability"

Step 2: Compute residuals

Same as Gradient Boosting:

> "How wrong are we for each email?"

Step 3: Build trees to correct errors

Regularization (VERY important)

XGBoost penalizes:

- Too many tree nodes
- Very deep trees

Real-life meaning:

> "Don't create overly complicated rules unless necessary."

This avoids overfitting.

Feature importance awareness

XGBoost learns:

- Which feature gives **maximum error reduction**
- Chooses splits that reduce loss the most