Boosting Algorithms- Regression

Ada Boost
XG Boosting
LG Boosting

What is Boosting Algorithm ?

- Boosting is an ensemble learning technique that combines multiple weak learners to form a strong learner

  Key Idea is

  - Models are built sequentially
  - Each new model focuses on the errors made by the previous models
  - Final prediction is weighted sum of all the models

Steps followed in Boosting Algorithm

How boosting works

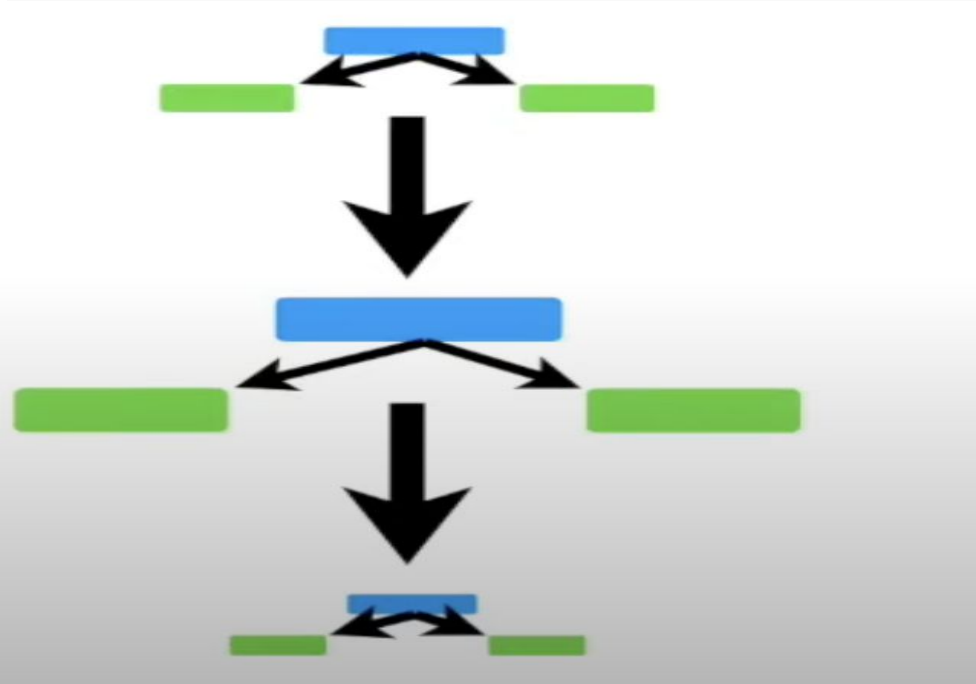
1. **Initialize** — Start with equal weights on all data points.
2. **Fit a weak model** — Train the first weak learner on the data.
3. **Re-weight the data** — Increase the weight of data points that were predicted poorly so the next model focuses more on them.
4. **Repeat** — Add more weak learners, each correcting the previous ones.
5. **Combine** — Final prediction is the weighted combination of all weak models.

Ada Boost

- Ada Boost stands for Adaptive Boosting
- Here adaboost always has one level decision tree which is called **Stumps(**it takes only one variable for a decision**)**
- Order is important in adaboost as errors in first stump makes influences the second stump and second stump error influences third stump and so on
- It builds a strong predictor by combining many weak learners
- Adjusts weights on data points → gives more importance to hard examples.
- Combines models using weighted sum (regression).

AdaBoost Example for Heart Disease Prediction

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No



Steps of prediction in AdaBoost

1️⃣ Start with equal weights for all training points.

2️⃣ At each iteration:

- Fit a **weak regressor** (e.g., decision stump) to the data.
- Compute errors: The error for a sample is proportional to how far off the predicted value is from the true value.
- Update sample weights: Give more weight to samples where the error was large → the next regressor focuses on these hard points.

3️⃣ Combine all regressors into a weighted sum

Adaboost Weighted Sum Formula

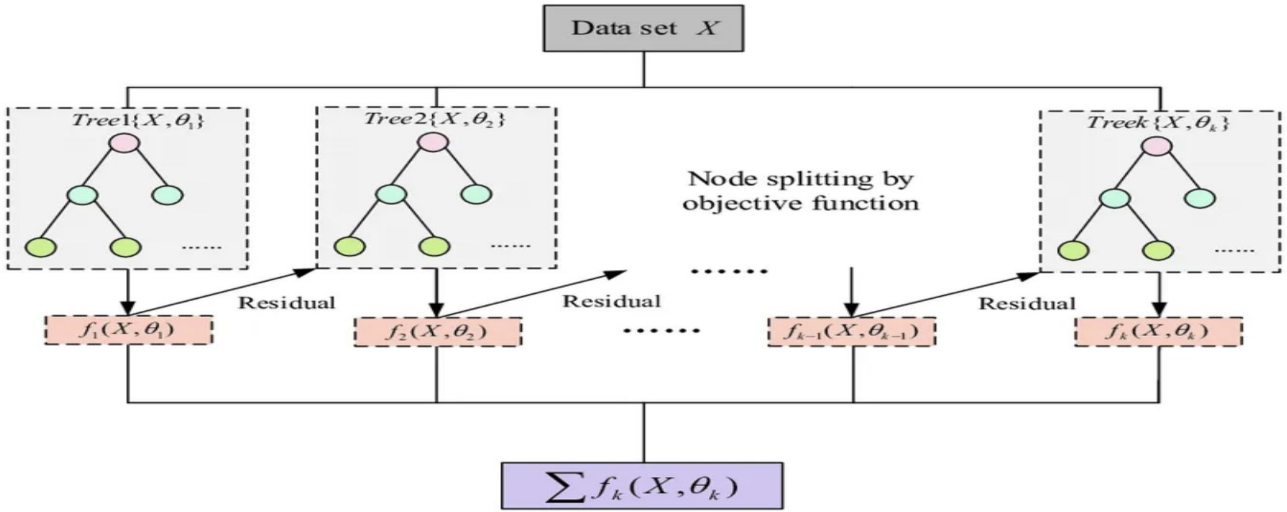
$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$$

where:

- $h_m(x)$ is the m-th weak regressor,
- α_m is its weight (depends on error).

XG Boosting

- XGBoost stands for **Extreme Gradient Boosting**. It's a powerful machine learning algorithm that builds a series of decision trees and combines them (boosting) to make better predictions.
- It uses **gradient descent** to minimize a loss function (e.g., mean squared error).
- After each round, it adjusts to focus more on the data points that are hard to predict.
- XGBoost builds trees in sequence to reduce error step-by-step.



Steps of prediction in XGBoost

1. Start with an initial prediction
Predict the average (mean) of y.Example: All houses initially predicted at $200,000.

2. Compute residuals (errors)
Residual = Actual value − Predicted value
Example:
Real = $250,000 → Residual = +$50,000
Real = $180,000 → Residual = -$20,000

3.Build the first tree
The first tree predicts these residuals.
Example:
If rooms > 5 → predict +$30,000
If area is low-cost → predict -$15,000

4.Update predictions
New prediction = Initial + (learning rate × tree prediction)

5.Build more trees
Each tree focuses on the remaining error.
Add their output to improve predictions.

6.Final prediction
Final prediction = Initial + sum of all tree corrections.

XGBoost Formula for similarity weight

$$\text{similarity score (leaf weight)} = -\frac{G}{H+\lambda}$$

Where:

- G = sum of gradients in that leaf
- H = sum of Hessians (2nd derivatives, measures confidence)
- λ = regularization parameter (controls complexity)

XGBoost Formula for Information Gain

The **gain from a split** is the difference in similarity before and after the split:

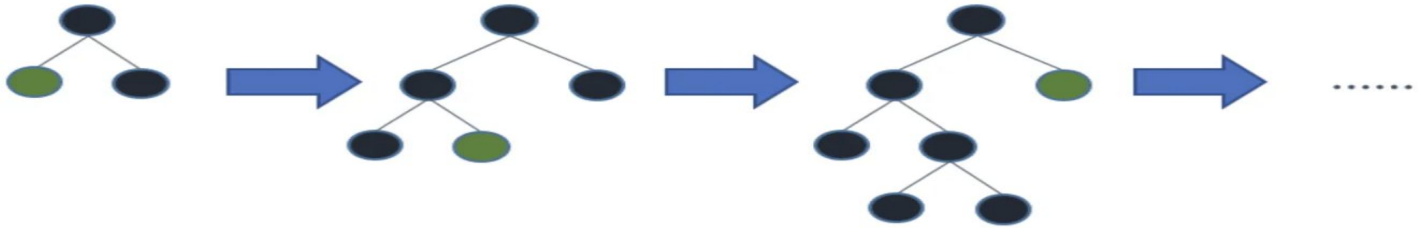
$$\boxed{\text{Gain} = \frac{1}{2}\left[\text{Similarity}_{\text{left}} + \text{Similarity}_{\text{right}} - \text{Similarity}_{\text{parent}}\right] - \gamma}$$

Where:

$$\text{Similarity}_{\text{node}} = \frac{G_{\text{node}}^2}{H_{\text{node}} + \lambda}$$

LG Boosting

- **LG** also called as LightGBM stands for **Light Gradient Boosting Machine**.
- It's a gradient boosting framework developed by Microsoft.
- It builds decision trees **one at a time** to minimize error
- It finds the leaf with the largest loss reduction (greatest gradient)
- Grows that leaf further.
- This can lead to deeper, more specialized trees → faster convergence.
- It is effective on large datasets(efficient memory usage and speed)
- XGBoost grows trees **level-wise** (all nodes at depth d before going deeper).➡ LightGBM grows trees **leaf-wise**:
- Instead of balancing the tree, it continuously split the leaf node with highest loss value resulting in deeper and asymmetrical decision tree.
- By persistently splitting the leaf nodes, the resulting decision tree can minimise prediction error loss more effectively than balanced tree splitting method
- It uses Gradient Based One Side Sampling technique to effectively predict



Steps of prediction in LG Boosting

1. **Start with initial prediction** (e.g., mean value)
2. **Compute gradients** (errors between actual & predicted)
3. **Grow tree leaf-wise to predict these gradients**
4. **Update the prediction** = previous prediction + learning rate × new tree output
5. **Repeat** for N trees
6. Final prediction = sum of all trees

LightGBM formula for leaf weight (like similarity weight)

Each leaf's optimal weight =

$$w_j = -\frac{G_j}{H_j + \lambda}$$

where:

- G_j = sum of gradients in leaf j
- H_j = sum of Hessians (2nd derivative info)
- λ = L2 regularization