(8) XG Boost (eXtreme Gradient Boosting)

→ XG boost regularizes data better than normal gradient boosted trees.

→ XG boost's objective function is the sum of loss function evaluated over all the predictions & a regularization func for all predictors (j trees).

$$obj(\theta) = \sum_{i}^{n} l(y_i - \hat{y}_i) + \sum_{j=1}^{j} \Omega(f_j)$$

→ loss function depends on the task being performed and a regularization term is described,

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

→ weights over scores.

(γT) ↓ for controlling overall no. of created leaves.

→ unlike other tree-building algorithms, XG Boost doesn't use entropy or Gini indices. Instead, it utilizes gradient (error term) and a "hessian" for creating the trees.

→ Hessian for a Regr. problem is no. of residuals and for classification problem, Hessian is a second order derivative of the loss at the current estimate.

$$h_m(x) = \frac{\partial^2 L(y, f(x))}{\partial f(x)^2} \quad f(x) = f^{(m-1)}(x)$$

Tree Building in XG Boost

(1) Initialize the tree with one leaf.

(2) Compute the "Similarity" using,

$$\text{Similarity} = \frac{\text{Gradient}^2}{\text{Hessian} + \lambda}$$

where, λ - Regularization term.

(3) Now, for splitting data into a treeform, calculate,

$$\text{Gain} = (\text{left Similarity}) + (\text{Right Similarity}) - (\text{Similarity for Root})$$

(4) For tree pruning, the parameter γ is used. The algorithm starts from the lowest level of the tree & then starts pruning based on the value of γ.

If $(\text{Gain} - \gamma) < 0$, then remove that branch

else, keep the branch.

(5) Learning is done using new eq.

$$\text{new value} = \text{old value} + \eta \times \text{Prediction}$$