**BOOSTING**

The process of building models sequentially by minimizing errors from the previous models and boosting influence of high performing variables and models.

**Gradient Boosting**

* Similar to boosting except it uses gradient descent to minimize errors.
* Builds new tree based on the error from the previous tree. [first node is the average of all outcomes]
* Tree leaf nodes/final decisions are the residuals or distance from actual values. [error]
* Scales the trees (using learning rate)
* Finally new predictions are made by this formula:

*Original average of all outcomes* + *(Learning rate* \* *Respective output leaf node)*

* Then these predictions are treated as observed vales and used to make another tree and compute new prediction values which have lower residuals than the previous ones.
* We repeat this process until we have a very powerful model.

It is optimized for variance so this tends to overfit the training data. [Variance too low means the model is too well fit for the training data]

**XG Boost**

This works in a similar way to gradient boosting except it utilizes 3 variables namely *lambda, gamma and eta.*

* Lambda is a regularization parameter, prevents overfitting
* Gamma decides if the tree is already pruned or needs further splitting. This is done by comparing gain.
* Eta is basically like the learning rate which is often taken as 0.3

Gain is defined as:

* Difference between similarity scores before split and after split of nodes.
* If gain value if lower than gamma, the tree is already pruned as much as needed but if it's greater than gamma, it needs to be split further.

Basically, gamma controls how big a tree do we need by comparing the current with the gain we actually need which we supply as gamma value.

Whenever the gain goes lesser than the gamma value, we stop as more splitting will only increase computation and not improve the outcomes much.

New predictions are again made by:

Previous prediction + (Eta\* Respective output leaf node)

Using these values, we create multiple models and at the end we have a model which would have minimum residuals for unseen data.

* Offers regularization terms which can improve model generalization and prevents overfitting.
* Good balance between Bias and Variance.

**LIGHT GBM**

* Most decision tree learning algorithms grow trees by level (depth)-wise as in they go level by level while building a tree. LightGBM grows trees leaf-wise (best-first), it will choose the leaf with max delta loss to grow.
* LightGBM uses histogram-based algorithms which divide continuous feature values (attributes) into discrete bins. This speeds up training and reduces memory usage.
* Holding number of leaves fixed, *leaf-wise* algorithms tend to achieve lower loss than *level-wise* algorithms.
* Leaf-wise may cause over-fitting when data is small, so LightGBM includes the max\_depth parameter to limit tree depth. However, trees still grow leaf-wise even when max\_depth is specified.