Boosting Methods

- **AdaBoosting** (Adaptive Boosting)
  - In AdaBoost, the successive learners are created with a focus on the ill fitteddata of the previous learner
  - Each successive learner focuses more and more on the harder to fit data i.e. their residuals in the previous tree
- **Gradient Boosting**
  - Each learner is fit on a modified version of original data. Original data is replaced with the x values and residuals from previous learner
  - By fitting new models to the residuals, the overall learner gradually improves in areas where residuals are initially high
- **XG Boost (Extreme Gradient Boosting)**
  - Upgraded implementation of Gradient Boosting. Developed for high computationalspeed, scalability, and better performance.
  - Parallel Implementation, Cross-Validation, Cache Optimization, Distributed Computation

What is the difference between bagging and boosting?

Bagging	Boosting
Parallel Training - All the weak learners are built in parallel i.e. independent of each other.	Sequential Training - Successive weak learners to improve the accuracy from the prior learners
Equal weightage - Each weak learner has equal weight in the final prediction.	Weighted average - More weight to those weak learners with better performance
Independent samples - Samples are drawn from the original dataset with replacement to train each individual weak learner	Dependent samples - Subsequent samples have more of those observations which had relatively higher errors in previous weak learners
Can help reduce variance of the model	Can help reduce bias of the model
Example: Bagging Classifier, Random Forest	Example: AdaBoost, Gradient Boosting Classifier

XGBoost Overview

- XGBoost builds upon the idea of gradient boosting algorithm with some modifications. Gradient boosted trees are built in sequence because each estimator predicts residuals of the previous estimator, which makes it slow at the time of model training as compared to building estimators in parallel

- Thus, the main concentration in XGBoost is speed enhancement and model performance

- The speed and scalability of XGBoost is due to several important features that help in better computing using concepts like parallelization, cache optimization, out of core computing and distributed computing

XGBoost - Salient features

- **Parallelization** - In XGBoost, data is stored in in-memory units, called block, and the optimal split in each tree can be found in parallel using this block structure. This reduces the time of model training significantly.

- **Cache Optimization** - XGBoost optimizes the block size for efficient parallelization and make best use of hardware.

- **Out-of-Core Computing** - This helps to handle huge data sets which do not even fit into memory.

- **Distributed Computing** - It helps to train huge models using multiple similar machines.

- **Missing Values Imputation** - XGBoost is designed to handle missing values internally. It uses a default direction for the missing values. However, the default direction can be right-child or left-child, and it is learned in the tree construction process to choose the best direction that optimizes the training loss.