Why do we need Boosting?

we have separately built six Machine Learning models for predicting whether it will rain or not. Each of these models has been built on top of the 6 distinct parameters given below to analyze and predict the weather condition:

Air temperature

Atmospheric (barometric) pressure

Humidity

Precipitation

Solar radiation

Wind

The outputs from the Machine Learning models may differ for these six parameters. The model which is evaluating air temperature may predict a sunny day. Whereas, another model may predict a rainy day based on humidity. Also, even if we predict the outcome on the basis of a single model, then there is a 50% probability of false prediction. These individual models are weak learners. But, if we combine all the weak learners to work as one, then the prediction would rely on 6 different parameters. The increase in the number of parameters will boost the accuracy of the model.

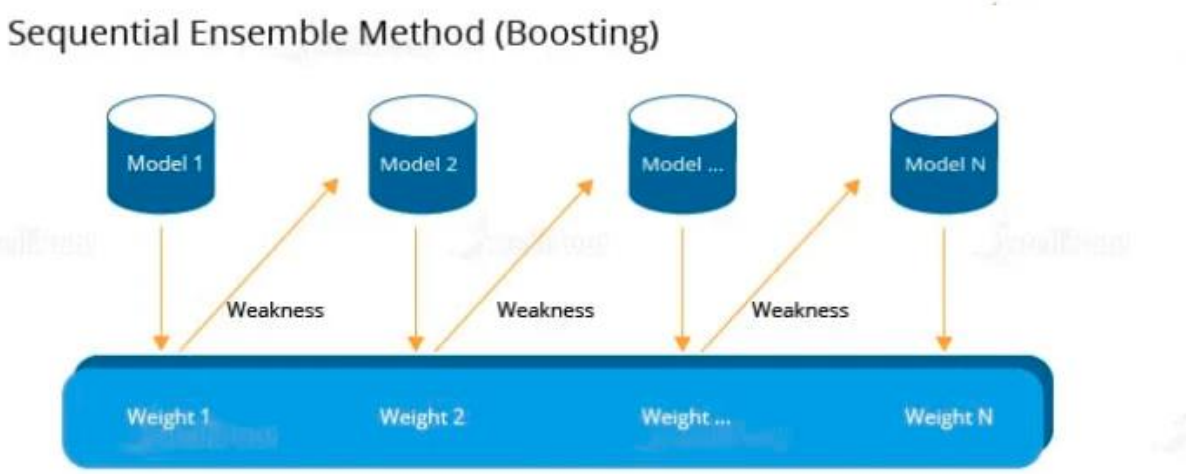
What is Boosting?

Boosting is a technique to combine weak learners and convert them into strong ones with the help of Machine Learning algorithms. It uses ensemble learning to boost the accuracy of a model. Ensemble learning is a technique to improve the accuracy of Machine Learning models. There are two types of ensemble learning:

1. Sequential Ensemble Learning

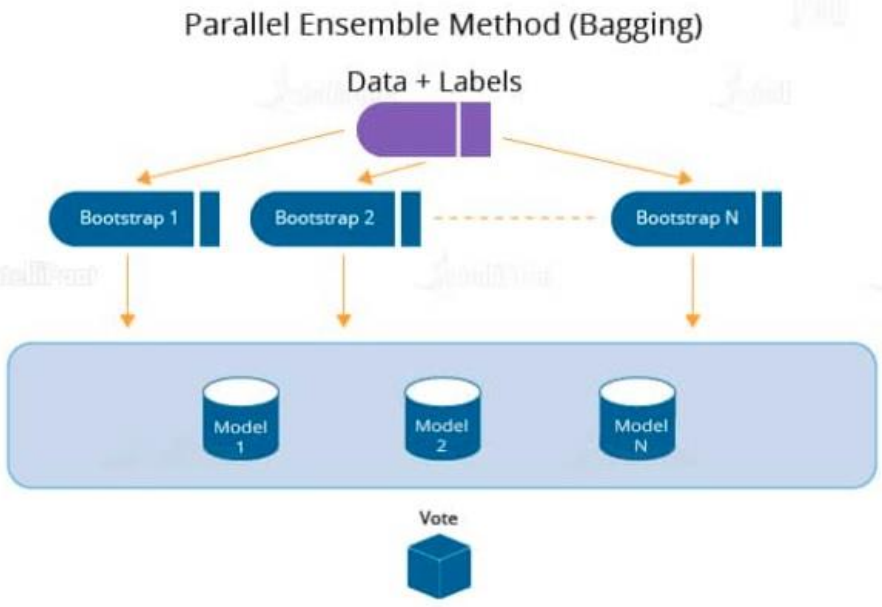
It is a boosting technique where the outputs from individual weak learners associate sequentially during the training phase. The performance of the model is boosted by assigning higher weights to the samples that are incorrectly classified. AdaBoost algorithm is an example of sequential learning that

we will learn later in this blog.



2. Parallel Ensemble Learning

It is a bagging technique where the outputs from the weak learners are generated parallelly. It reduces errors by averaging the outputs from all weak learners. The random forest algorithm is an example of parallel ensemble learning.



Mechanism of Boosting Algorithms

Boosting is creating a generic algorithm by considering the prediction of the majority of weak learners. It helps in increasing the prediction power of the Machine Learning model. This is done by training a series of weak models.

Below are the steps that show the mechanism of the boosting algorithm:

1. Reading data
2. Assigning weights to observations
3. Identification of misinterpretation (false prediction)
4. Assigning the false prediction, along with a higher weightage, to the next learner
5. Finally, iterating Step 2 until we get the correctly classified output

Types of Boosting Algorithms

Basically, there are three types of boosting algorithms

1. Adaptive Boosting (AdaBoost)

Adaptive boosting is a technique used for binary classification. For implementing AdaBoost, we use short decision trees as weak learners.

Steps for implementing AdaBoost:

1. Train the base model using the weighted training data
2. Then, add weak learners sequentially to make it a strong learner
3. Each weak learner consists of a decision tree; analyze the output of each decision tree and assign higher weights to the misclassified results. This gives more significance to the prediction with higher weights.
4. Continue the process until the model becomes capable of predicting the accurate result

2. Gradient Boosting

In Machine Learning, we use gradient boosting to solve classification and regression problems. It is a sequential ensemble learning technique where the performance of the model improves over iterations. This method creates the model in a stage-wise fashion. It infers the model by enabling the optimization of an absolute differentiable loss function. As we add each weak learner, a new model is created that gives a more precise estimation of the response variable.

The gradient boosting algorithm requires the below components to function:

1. Loss function: To reduce errors in prediction, we need to optimize the loss function. Unlike in AdaBoost, the incorrect result is not given a higher weightage in gradient boosting. It tries to reduce the loss function by averaging the outputs from weak learners.
2. Weak learner: In gradient boosting, we require weak learners to make predictions. To get real values as output, we use regression trees. To get the most suitable split point, we create trees in a greedy manner, due to this the model overfits the dataset.
3. Additive model: In gradient boosting, we try to reduce the loss by adding decision trees. Also, we can minimize the error rate by cutting down the parameters. So, in this case, we design the model in such a way that the addition of a tree does not change the existing tree.

Finally, we update the weights to minimize the error that is being calculated.

How is this useful? Gradient boosting is a highly robust technique for developing predictive models. It applies to several risk functions and optimizes the accuracy of the model's prediction. It also resolves multicollinearity problems where the correlations among the predictor variables are high.

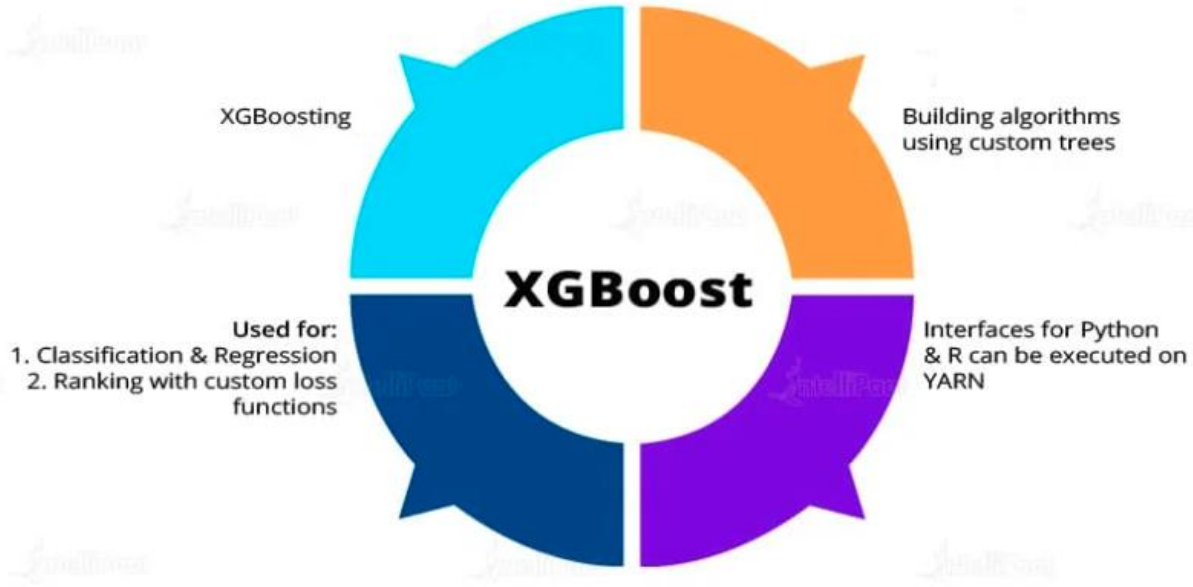
Gradient boosting machines have been successful in various applications of Machine Learning.

3. XGBoost

XGBoost algorithm is an extended version of the gradient boosting algorithm. It is basically designed to enhance the performance and speed of a Machine Learning model.

Why do we use XGBoost?

In the gradient boosting algorithm, there is a sequential computation of data. Due to this, we get the output at a slower rate. This is where we use the XGBoost algorithm. It increases the model's performance by performing parallel computations on decision trees.



What features make XGBoost unique?

XGBoost is much faster than the gradient boosting algorithm. It improves and enhances the execution process of the gradient boosting algorithm.

There are more features that make XGBoost algorithm unique and they are:

1. Fast: The execution speed of the XGBoost algorithm is high. We get a fast and efficient output due to its parallel computation.
2. Cache optimization: To manage and utilize resources, it uses cache optimization.

3. Distributed computing: If we are employing large datasets for training the Machine Learning model, then XGBoost provides us distributed computing, which helps combine multiple machines to enhance performance.