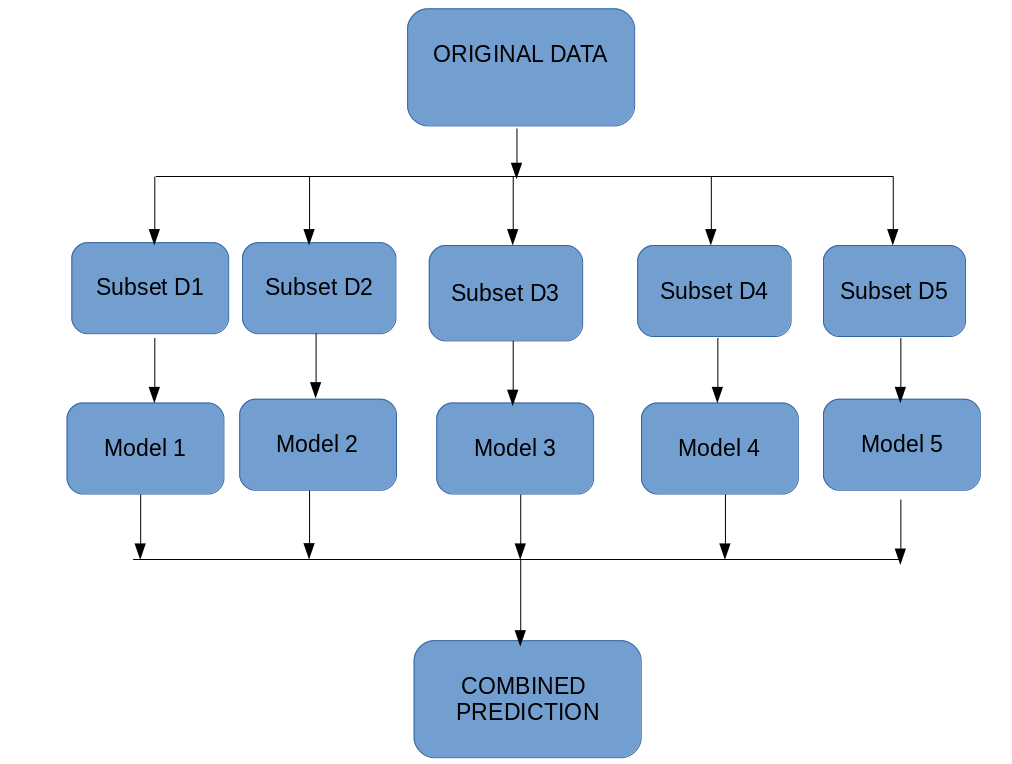
**Ensemble Techniques:**

The technique where we use multiple decision makers to make a decision is called ensemble techniques, so that we will get strong outcome.

In ensemble techniques we will resolve classification and regression problems. This algorithm arranged in such a way that the base algorithm with multiple other algorithms contribute together to make a decision. (eg: If we are using multiple decision trees to make a decision which reduces the biasness)



We have two types of ensemble techniques:

1. Bagging Technique
2. Boosting Technique

|  |  |
| --- | --- |
| **Bagging** | **Boosting** |
| Random Forest | AdaBoost |
|  | GradientBoost |
|  | xgBoost |

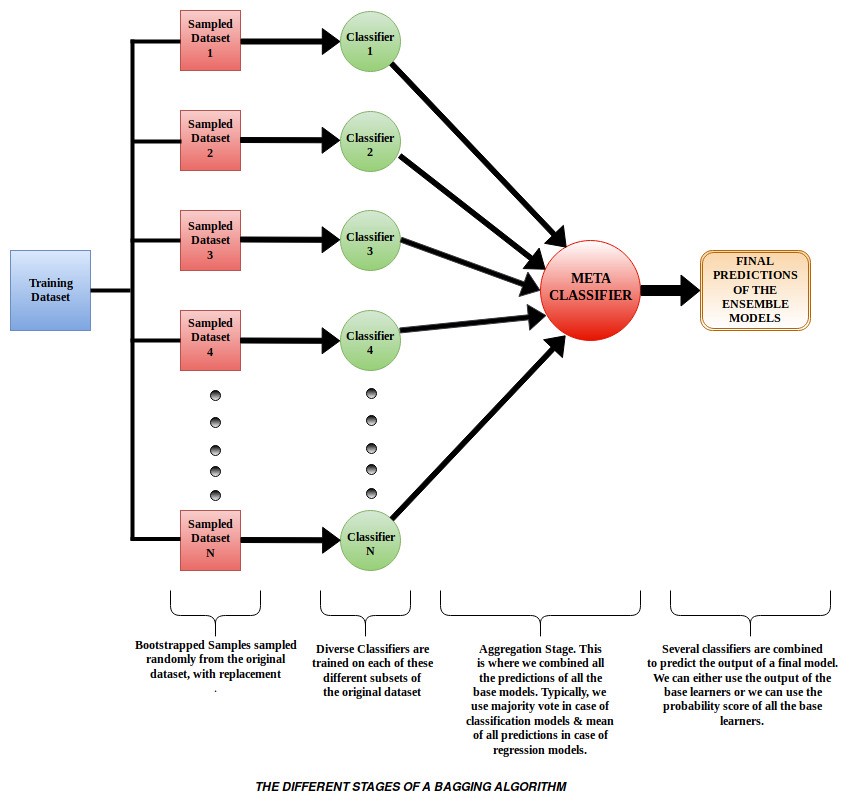
**Bagging**is a homogeneous weak learners’ model that learns from each other independently in parallel and combines them for determining the model average.

**Boosting**is also a homogeneous weak learners’ model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

**Bagging**

Bagging, also known as bootstrap aggregation, where we the dataset is divided into subsets and aggregate or combine the result.

In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once (features will be same but records may overlap). After several data samples are generated, these weak models are then trained independently, depending on the type of task the outputs are considered. In regression : average of the model output considered or classification : majority vote considered.



Here decision makers are in a single bag. It uses the base model as decision trees and all the models will get executed parallelly. The problem with decision tree is that it split all the features till it’s depth which results overfitting of model and ensemble learning technique is commonly used to reduce variance within a dataset.

In bagging if a new data(test) comes it may get executed in every models and will aggregate output gets calculated and algorithm find-out the majority voting(classification) and mean(regression).

**Random Forest:**

### Random Forest is one of the algorithm which is the part of Ensemble technique Bagging process. This is the combination of Decision Trees.

### Random forest - Wikipedia

Random Forest is one of the algorithm which is a part of ensemble technique bagging process. By default random forest can use only one algorithm that is decision tree. Decision tree divide the dataset up to its maximum depth which results overfitting (high variance) of each model. We can reduce this high variance to low variance using pre-pruning and post-pruning but with large dataset it is not possible. So in Random Forest we take aggregation of models. If we have one DT with overfitting, here we have other DT to support the claim. Here we are doing row sampling and column sampling.

In row sampling there is a chance that some data may left in the dataset bag. These data can be used as **out of bag** evaluation.

 Pros of using the Random Forest technique:

* It efficiently manages a higher-dimension data set.
* It handles missing quantities and maintains high accuracy for missing data.

 Cons of using the Random Forest technique:

* The last prediction depends on the mean predictions from the subset trees, so it will not provide an accurate value for the regression model.
* In random forest you won’t be able to get that clarity that what kind of decision tree it is using and how it is performing operation.

**How Bagging Works**

1. **Bootstrapping:** Bagging leverages a bootstrapping sampling technique to create diverse samples. This resampling method generates different subsets of the training dataset by selecting data points at random and with replacement. This means that each time you select a data point from the training dataset, you are able to select the same instance multiple times. As a result, a value/instance repeated twice (or more) in a sample.
2. **Parallel training:** These bootstrap samples are then trained independently and in parallel with each other using weak or base learners.
3. **Aggregation:**Finally, depending on the task (i.e. regression or classification), an average or a majority of the predictions are taken to compute a more accurate estimate. In the case of regression, an average is taken of all the outputs predicted by the individual classifiers; this is known as soft voting. For classification problems, the class with the highest majority of votes is accepted; this is known as hard voting or majority voting.

## **Boosting**

In boosting we try to involve multiple decision makers with weightage to boost our decision-making. Here we try to boost our decision with a decision maker with better information because every next decision maker is better than previous and here we convert weak learners to strong learners.

To convert weak learner to strong learner, we’ll combine the prediction of each weak learner using methods like:  
•   Using average/ weighted average  
•   Considering prediction has higher vote

To find weak learners, we apply base learning (ML) algorithms with a different distribution. Each time base learning algorithm is applied, it generates a new weak prediction rule. This is an iterative process. After many iterations, the boosting algorithm combines these weak rules into a single strong prediction rule.

Eg :

How to classify an email is SPAM or not? Like everyone else, our initial approach would be to identify ‘spam’ and ‘not spam’ emails using following criteria. If:

1. Email has only one image file (promotional image), It’s a SPAM.
2. Email has only link(s), It’s a SPAM.
3. Email body consist of sentence like “You won a prize money of $ xxxxxx”, It’s a SPAM.
4. Email from a respective person or any official mail NOT SPAM.

Here you can see, we’ve defined multiple rules to classify an email into ‘spam’ or ‘not spam’. But, do you think these rules individually are strong enough to successfully classify an email? No.

Individually, these rules are not strong enough to make a decision or classify an email into ‘spam’ or ‘not spam’. Therefore, these rules are called as **weak learner**.

Boosting is a sequential ensemble method that iteratively adjusts the weight of observation as per the last classification. If an observation is incorrectly classified, it increases the weight of that observation so that it can be selected as the next weak learners. It decreases the bias error and builds strong predictive models. It iterates the model till the base model’s accuracy will reach to the higher accuracy.

In simple words here we have the models in a sequence. Where there will be weak learners and they won’t be able to predict properly. We combine the weak learners sequentially and it’ll become strong learner.

## **How Boosting Algorithms works?**

To find weak rule, we apply base learning (ML) algorithms with a different distribution. Each time base learning algorithm is applied, it generates a new weak prediction rule. This is an iterative process. After many iterations, the boosting algorithm combines these weak rules into a single strong prediction rule.

Here’s another question which might haunt you, ‘How do we choose different distribution for each round?’

For choosing the right distribution, here are the following steps:

Step 1:  The base learner takes all the distributions and assign equal weight or attention to each observation.

Step 2: If there is any prediction error caused by first base learning algorithm, then we pay higher attention to observations having prediction error. Then, we apply the next base learning algorithm.

Step 3: Iterate Step 2 till the limit of base learning algorithm is reached or higher accuracy is achieved.

Finally, it combines the outputs from weak learner and creates  a strong learner which eventually improves the prediction power of the model. Boosting pays higher focus on examples which are mis-classiﬁed or have higher errors by preceding weak rules.

There are 3 types of boosting algorithms

1. AdaBoost (Adaptive Boosting)
2. Gradient Tree Boosting
3. XGBoost

**AdaBoost:**

In Adaboost initially we provide equal weight to all the records. Then we choose the root node using Information gain We plot decision tree and it will be plotted upto one level so it is called as stump(under fitting 🡪 Weak learners). We pass all the record to to predict output and findout the total error. Then we will calculate the performance of the stump. And we will update the weights, weightage will be given more to wrong prediction and less to correct prediction so it can be selected for the next model. This method works iteratively to adjust the weights by minimizing the training error. This process keep on going until we get strong output class.

We can use AdaBoost algorithms for both classification and regression problem. Any machine learning algorithms can be used as base learner in ada-boost model.

* 1. Total error

(TE = MAE/MSE/Precision/Recall/F1 score)

* 1. Performance of stump

()

* 1. New sample weight

(Update the weights :

for correctly classified points = weight \*

for incorrectly classified points = weight \* )

* 1. We will find-out the normalize weight
  2. We will create buckets
  3. Choose wrong predicted points(Weak learners) for next model.
  4. Iteratively we will convert weak learners to strong learners

Final function: f(x) =

= weights

= weak learners

If weight is high then then model is important. If weight is -ve weight is less important.

Mostly to select the wrong (error) predicted points for the next model we will create bucket(range). So that the maximum number of wrong records will get selected for the prediction for next model. We try to create stump, means creating decision tree with only one depth which results underfitted model and calculate their performance. Then we update the new weight of all the models. As the summation of updated weight is not equal to 1 we will find out normalized weight. By following the same process we finally get strong outcome.

**Gradient boosting**

Gradient boost is an ensemble technique where Decision Tree is used as base algorithm. It is used both for regression as well as classification problem. Gradient boost works sequentially by adding decision makers to an ensemble, here we try to find-out the total error and minimize it. Instead of changing weights of data points like AdaBoost, the gradient boosting trains on the residual errors of the previous decision maker. The name, gradient boosting, is used since it combines the gradient descent algorithm and boosting algorithms with boosting method.

* 1. Create a base model
  2. Residual = actual label – predicted result
  3. We find out derivative of residual
  4. New value = old value + λ.residual

We keep on doing this process to minimize residuals. By this process the error value keep on decreasing and we try to keep residual low.

In gradient boost we keep on plotting decision tree based on and residual until we minimize the residual. Every time we build decision tree residual/error value will reduce and we reduce over-fitting condition.

Final function: f(x) =

= Learning rate which can change with model

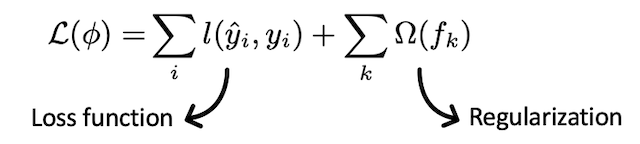
= base model

**Extreme gradient boosting or XGBoost:**

XGBoost improves the gradient boosting method even further. In XGboost we use base algorithm as decision tree. We call it as boosting because we add the output of each Decision Tree to get final output. We keep on adding outcome to make it stronger.

**XGboost Regression:**

Final function :



Where, = nT +

For XGboost loss function is being used and it will be used as per problem statement. We try to use regularization to generalize our model in a better way. To reduce overfitting we can stop heavy or drastic changes in terms of loss. The above function can control the drastic changes while the data move from one step to another.

XGboost uses objective function which is modified version of loss function with regularization parameter. As the base model is Decision Tree we need to minimize the loss and normalize the model.

XGboost doesn’t use entropy or gini-impurity,instead it uses gradient and hession to create a tree means in this model the tree will be formed based on gradient approach not gini or entropy based approach.

In gradient based approach we will plot a tree then calculate the error and findout the new values then again plot another tree based on weightage assigned to each parameter.

**XGboost classifier:**

In XGboost classifier we will create a base model which is a weak learner and give output always 0.5. Then we will calculate residual. And then similarity weight. According to the information gain we will select root node. We will do further split according to the next condition.

1. Create a base binary Decision Tree using the feature
2. Find out the residuals
3. Calculate the similarity weight

(sw = )

1. Calculate information gain
2. Calculate the loss function for logistic regression (log loss for base model and sigmoid for rest)

Final function: σ (Base learner + )

Log loss sigmoid sigmoid

For multiclass classification problem we use soft max activation function.

In XGboost we create a base model and calculate residual(R1) then find the predicted values (y^) and calculate residual(R2).

**Note:**

1. In Decision Tree scaling/normalization/standardization is not required as in both of the cases before and after scaling the data we will get the same output.

2. Random Forest/ Decision Tree is not impacted by outliers as we are segregating the datapoints based on conditions.