1. What is Ensemble Learning?

The general meaning of “ensemble” is “a group of items viewed as a whole rather than individually”. The meaning is the same in Machine Learning(ML) also. Ensemble learning is a part of ML which uses multiple learning algorithms to obtain better predictive performance than could be obtained from any of the basic or essential learning algorithms alone.

Let’s understand ensemble learning with an example. When you want to visit a restaurant, you’ll try to find out information about that restaurant from various resources like asking a friend, searching for reviews in Google, calling the restaurant for food options, etc. After considering all these opinions, you would take a decision. Ensemble models also work in the same way. They combine the decisions from multiple models to improve the performance.

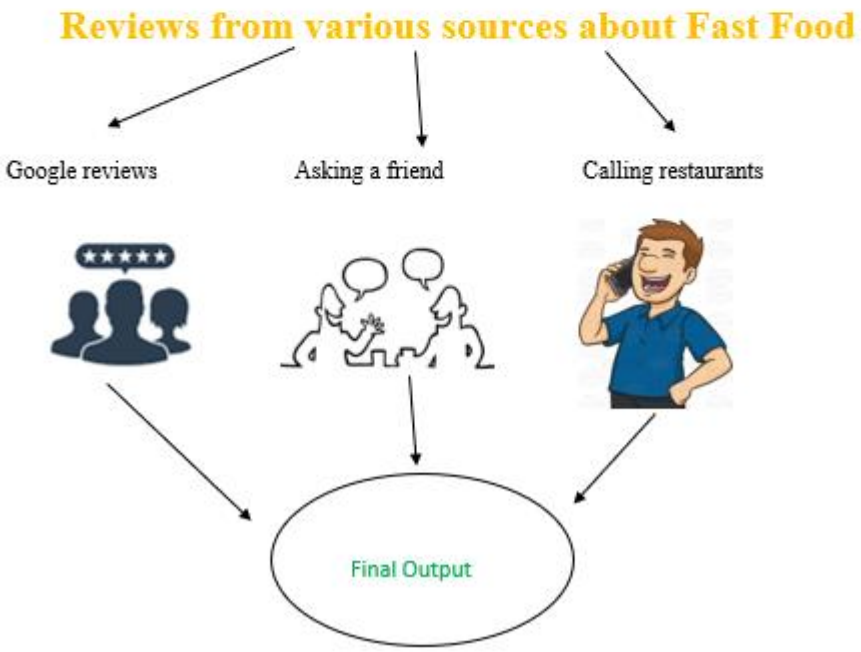

Fig. 1 Ensemble Learning example

2. Why do we need Ensemble Models?

Before understanding why we need ensemble models, let us understand what is bias and variance.

Wikipedia:

The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

The variance is an error from sensitivity to small fluctuations in the training set. A high variance may result from an algorithm modeling the random noise in the training data (overfitting).

The bias-variance trade-off is a core problem in supervised learning. Preferably, we want to select a model that accurately captures the regularities(the quality of being characterized by a fixed rate) in its training data and generalizes well to unseen data. It is typically not possible to do both simultaneously. Hence, we have to maintain an optimal balance in bias and variance.

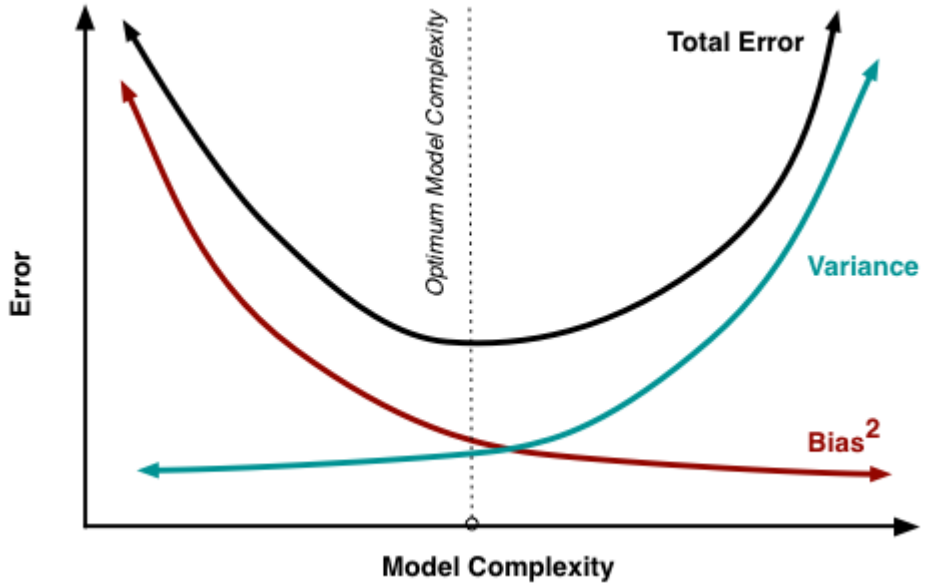

Fig. 2 Bias Variance Trade-Off

Total Error = Bias² + Variance + Irreducible Error

When you try to predict the target variable using any ML technique, the main factors for difference in actual and predicted values are noise, variance, and bias. Ensemble learning helps to reduce these factors except noise which is an irreducible error. Reducing bias and variance makes these models more robust.

3. Types of Ensemble Learning Methods

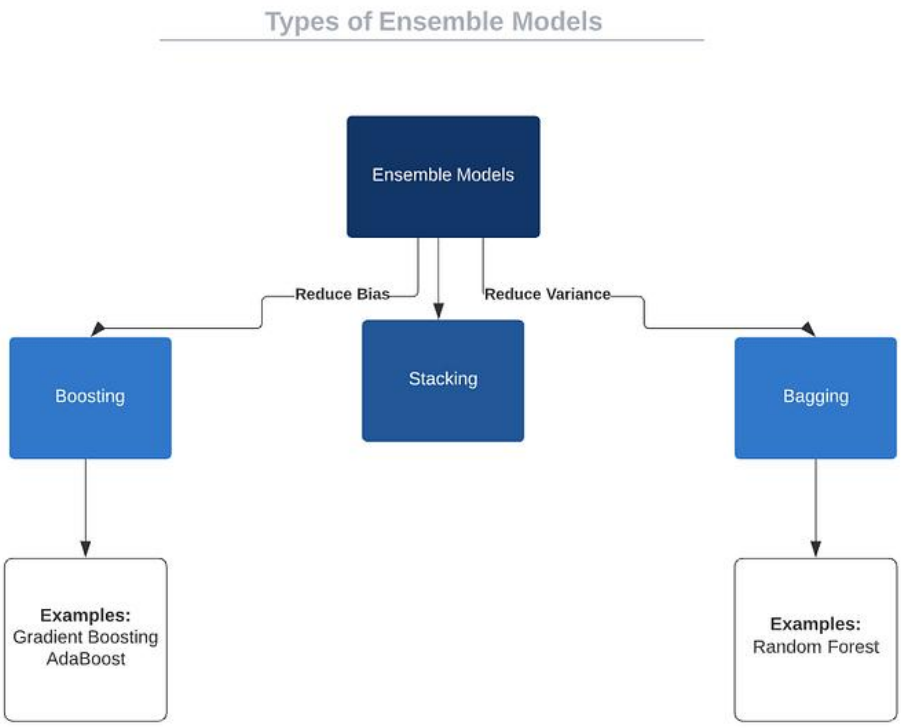

Fig. 3 Types of Ensemble Models

Bagging

Bagging is a parallel ensemble meta-algorithm and a combination of Bootstrap and Aggregation which is developed to reduce variance and improve the accuracy of ML problems that can be used in both Regression and Classification.

Bootstrapping is a technique used to generate samples by randomly choosing data points with replacement from the original dataset. These samples are called Bootstrap samples whose size is less than the original dataset.

To get most from bootstrapping, 2 hypotheses have to be satisfied:

First, the size of the dataset (N) should be large enough to capture most of the complexity of the underlying distribution, so that sampling from the dataset is a good approximation of sampling from the real distribution.

Secondly, the size of the original dataset (N) should be much greater than the bootstrap sample size (B) because the samples can be correlated.

Aggregation is a process of combining the output of each model fitted on the bootstrapped samples. For Regression, all the predicted outputs are averaged and for Classification, max voting is used.

$$\widehat{f_{bag}} = \widehat{f_1}(X) + \widehat{f_2}(X) + \cdots + \widehat{f_b}(X)$$

Fig. 4 Bagging Equation

In the above equation, X is the record for which we want to generate a prediction, LHS(fbag) is the bagged prediction, and RHS is the predictions from the individual base learner.

Let's take an example,

Given a standard training set D of size N, bagging generates M new training sets Di each of size B by sampling from D uniformly and with replacement. By the phrase sampling with replacement means some observations may be repeated in each Di.

Then Mi models are fitted using these M bootstrapped samples and combined by averaging the output (for regression) and voting (for classification)

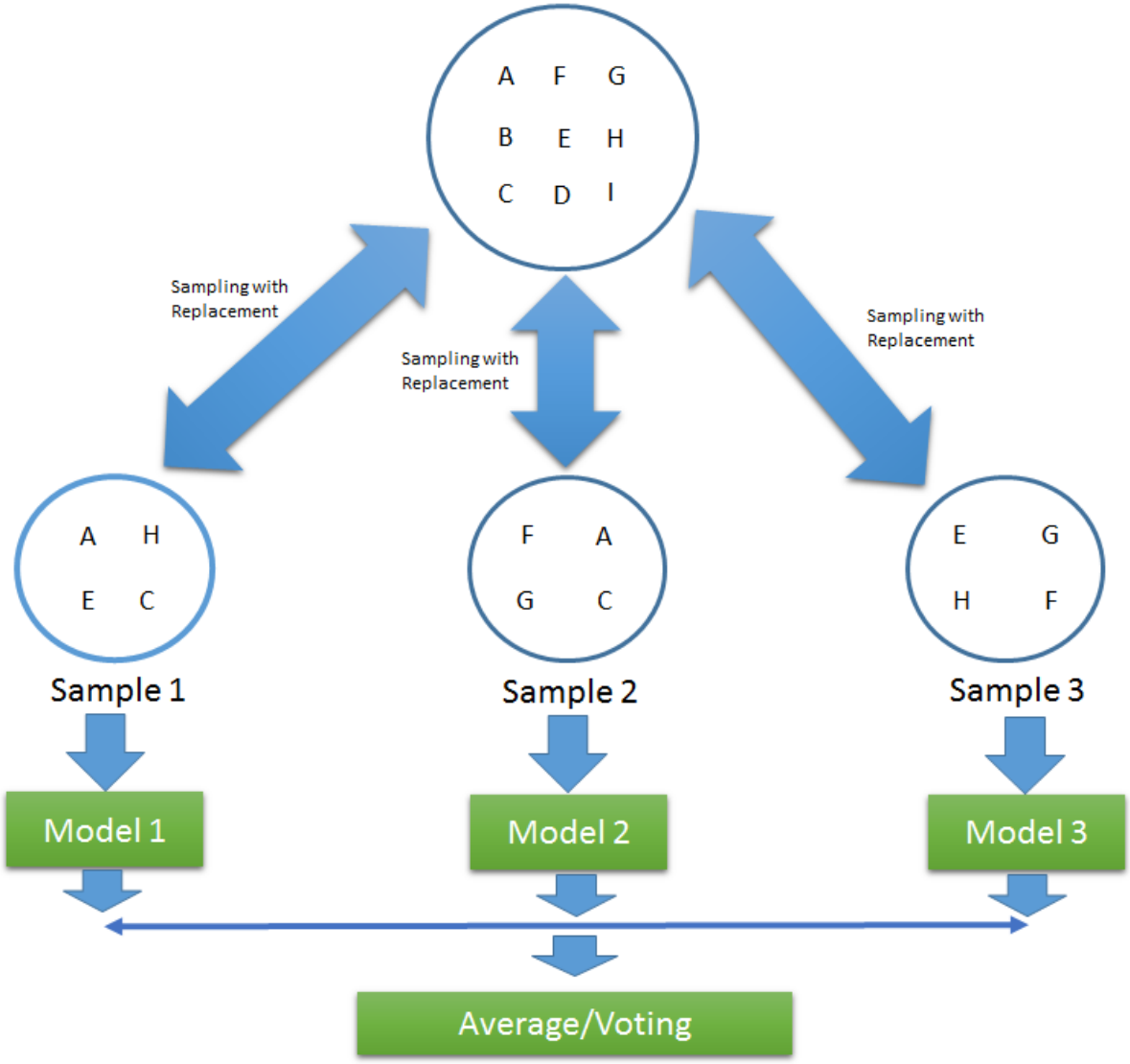

Fig. 5 Example for Bagging

Generally, for bagging the base model is usually the one with low bias and high variance. The final model's variance is reduced because when data changes, only a part of the changed data is sent to every Mi, so most of the Mi won't be affected much and hence overall variance is reduced.

Advantages

- Many weak learners aggregated typically outperform a single learner over the entire set and have less overfit.
- Reduces variance in high variance datasets
- Parallel processing for both training and evaluation.

Disadvantages

- It may not perform well in high bias datasets.
- Loss of interpretability of a model.
- Can be computationally expensive depending on the data.

Boosting

In ML, boosting is a sequential ensemble meta-algorithm primarily used for reducing bias. It is a family of ML algorithms that converts weak learners to strong learners. The basic intuition behind Boosting is to train weak learners **sequentially**, each trying to correct its predecessor which will eventually reduce the bias. Usually, a weak learner which has low variance and high bias like a shallow Decision Tree is chosen. We have to reduce bias while keeping variance low.

Mathematically,

1. Fit a decision tree to the data: F1(x) = y

2. We then fit the next decision tree to the residuals of the previous:

h1(x) = y — F1(x)

3. Add this new tree to our algorithm: F2(x) = F1(x) + h1(x)

4. Fit the next decision tree to the residuals of F2: h2(x) = y — F2(x)

5. Add this new tree to our algorithm: F3(x) = F2(x) + h2(x)

6. Continue this process until some mechanism (i.e., cross validation) tells us to stop.

The final model here is a **stagewise additive model** of b individual trees:

$$f(x) = \sum_{b=1}^{B} f^b(x)$$

Fig 6. Equation for Boosting

These models are called additive weighted models because every individual model is trained to fit on the residual error (an error that is left out) at the end of its previous stage.

As the value of b increases the final model ends up having a low residual error because, at every stage, we are fitting the model on the error. As training error reduces, bias also reduces.

Advantages

- The results of boosting will always be better than the single base learner.

Disadvantages

- One disadvantage of boosting is that it is sensitive to outliers since every classifier is obliged to fix the errors in the predecessors.

Stacking

The basic idea of Stacking is to consider **heterogeneous** weak learners, learn them in **parallel** and combine them by training a meta-model to output a prediction based on the different weak models' predictions. Practically, the more different the models are the better. All these models are constructed parallelly and independent of each other.

Step 1- Train M models on the original Dataset D.

Step 2- Make a new dataset D' which consists of the predictions from the M models in step 1.

Step 3- Now construct a second-level model or meta-model to make predictions on the new dataset D'.

Advantages

- Can improve the performance by controlling the capabilities of various well-performing models.

Disadvantages

- High training time
- High evaluation time

CODE:

For more explanation of the code

Code for Stacking algorithm for classification problem

4. Most used algorithms based on Bagging and Boosting Methods

Random Forest

Random forests, also known as random decision forests, are a widely used bagging ensemble learning method for classification and regression. The place where we find many trees is a forest. In Random Forest we use many Decision trees as the base model hence the name Forest. Random comes from the "Random" Bootstrap sampling. Random forest is a combination of **Decision Trees + Bagging + Column Sampling** (feature bagging).

Column sampling is nothing but picking random features /columns to make samples from the original dataset. The core idea of Random Forest is to make samples by picking random rows and columns and then train models on these samples and aggregate the results.

After sampling a set of points from the original dataset, the points which are not considered in that sample are called **out of bag** points and the ones which are considered are called in bag samples.

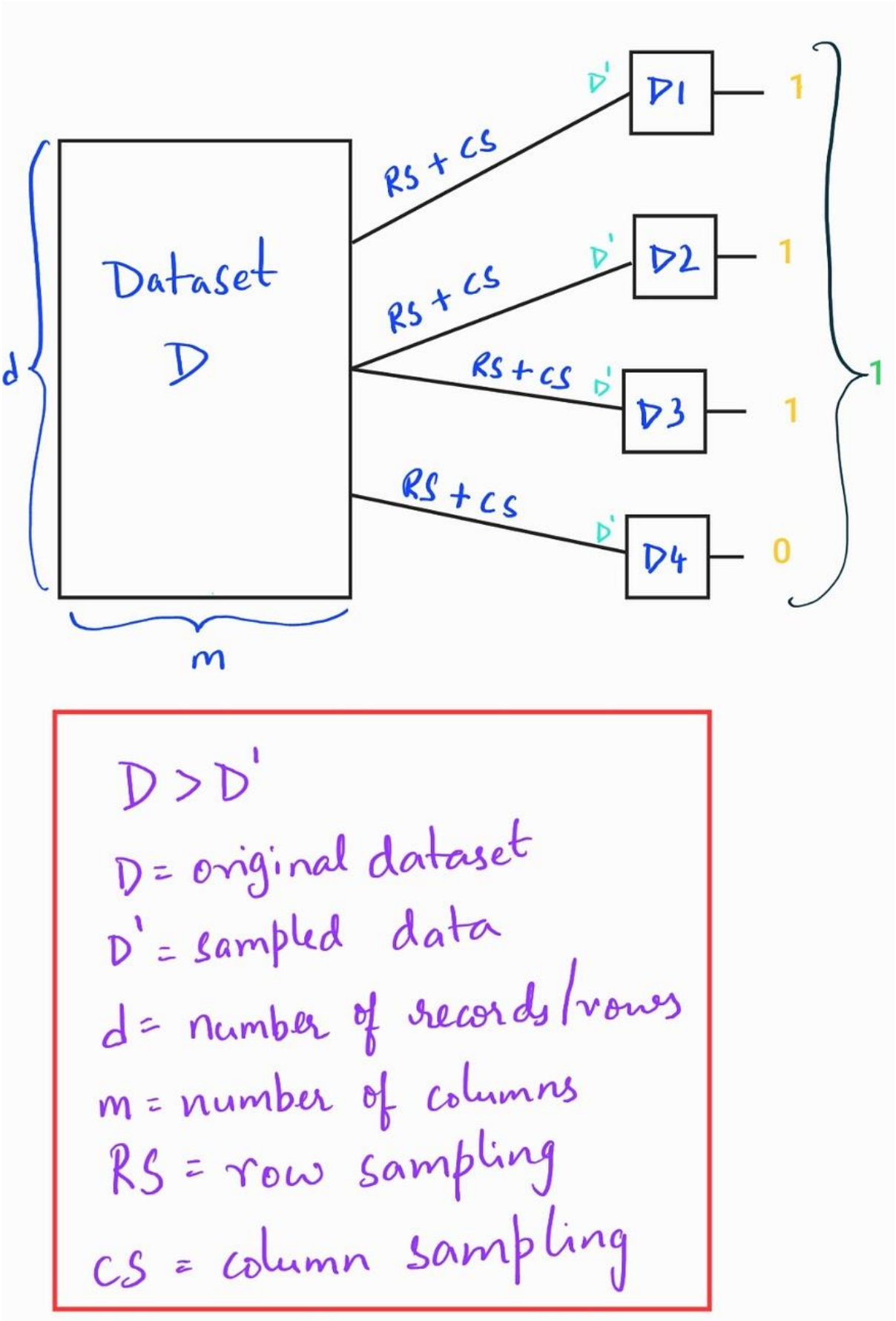

Fig. 7 Process of Random Forest Algorithm

Fruits	Colour of the fruit?	Costly or not?
Pomogranate	Red	Yes
Apple	Red	Yes
Banana	Yellow	No
Orange	Orange	No
Mango	Yellow	Yes

In bag samples

Out of bag samples

Fig. 8 Sample Data of In-Bag and Out-of-Bag points

These OOB points can be used to test the model trained on in-bag sampled data. Many libraries give OOB errors to get a sense of every model built on the sample from the original dataset.

CODE:

For more information on the code

Code for Random Forest with test scores

n_estimators, bootstrap, max_features are called hyperparameters, we have to tune them to avoid overfitting and to improve accuracy.

Gradient Boosting

Gradient boosting is an ensemble machine learning algorithm used for both regression and classification, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a **stage-wise manner** as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Gradient Boosting is built on 3 pillars: -

(I) Loss function

Choosing a perfect loss function depends on the type of problem needed to be solved. E.g.; for regression, one may use squared error and for classification logarithmic loss. The chosen loss function has to be optimized to get better results.

(II) Weak Learner

Typically, decision trees with reasonable depth are used as the weak learner. These trees are constructed greedily and we have to choose the best split points based on purity scores like Gini or Entropy.

(III) Additive Model

The output for the new tree is added to the output of the existing sequence of trees to correct or improve the final output of the model. These trees are added one at a time and existing trees in the model are not changed. A gradient descent procedure is employed to scale down the loss when adding trees.

As this algorithm is a greedy one, it can **overfit** the training dataset. There are 4 ways to regulate this outfitting problem for this algorithm: -

1. Tree Constraints:- The trees used in this algorithm needs to be controlled to avoid overfitting. Following are the most used parameters to control a tree:

• number of trees

• tree depth

• number of nodes or number of leaves

• number of observations per split

• minimum improvement to loss

2. Weighted Updates:- Whenever a new tree is added, its contribution can be weighted by controlling the learning rate. Usually, a small value is used for learning rate in the range of 0:1 to 0.3.

3. Stochastic Gradient Boosting:- The advantage of making samples in Bagging can also be and in Boosting. This reduces the correlation between the trees. At every iteration, a sample is drawn from the original dataset, this sample has used the fit the base learner.

4. Penalized Gradient Boosting:- Regularizing the final learned weights can help to avoid over-fitting. Popular regularization functions used are:

• L1 regularization

• L2 regularization

CODE:

For more information on the code

Gradient Boosting Code on sample data

5. Comparison between Bagging, Boosting, and Stacking

Criteria	Bagging	Boosting	Stacking
Goal to achieve	Reduce vairance	Reduce bias	Improve performance
Processing of base learners	Parallel	Sequential	Parallel
Function for combining models	Max voting/Averaging	Additive Models	Meta - model

Fig. 9 Comparision between various ensemble learning methods