Ensemble Methods

Ensemble is a Machine Learning concept in which the idea is to train multiple models using the same learning algorithm and use a collection of results from these models (e.g. mean of all predictions) to give a final prediction.

The main causes of error in learning are due to noise, bias and variance. Ensemble helps to minimize these factors. Ensemble methods are also designed to improve the stability and the accuracy of machine learning algorithms. Ensembling techniques are further classified into Bagging and Boosting.

Bagging is a simple ensembling technique in which we build many independent models and combine them using some model averaging techniques.

Example of bagging is Random Forest models.

Boosting is an ensemble technique in which the predictors are not made independently, but sequentially. In this technique, the subsequent predictors learn from the mistakes of the previous predictors. Therefore, it takes less time/iterations to reach close to actual predictions. We have to choose the stopping criteria carefully to avoid overfitting on the training dataset.

One example of the boosting algorithm is Gradient Boosting.

1, Random Forest model

It is an averaging algorithm based on randomized decision trees with the following key characters:

1) Each decision tree in random forests classifier or regressor is built from a bootstrap sample (sample drawn with replacement) from the training set.

2) For each split of a node, only a random subset of features are chosen and a best split is chosen.

3) The predicted result is the mode of all the results from all decision trees.

Compared to individual decision tree, the bias of the forest slightly increases but the variance decreases due to averaging. Thus the result from the random forests model is usually better compared to individual decision tree.

2, Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

It repetitively leverage the patterns in residuals and strengthen a model with weak predictions. Once we reach a stage that residuals do not have any pattern that could be modeled, we can stop modeling residuals (otherwise it might lead to overfitting). Algorithmically, we are minimizing our loss function, such that test loss reach its minima.

Steps for gradient boosting involves:

• We first model data with simple models and analyze data for errors.
• These errors signify data points that are difficult to fit by a simple model.
• Then for later models, we particularly focus on those hard to fit data to get them right.
• In the end, we combine all the predictors by giving some weights to each predictor.

Gradient boosting involves three elements:

- *A loss function to be optimized.* The loss function used depends on the type of problem being solved. For example, regression may use a squared error and classification may use logarithmic loss.

- *A weak learner to make predictions.* Decision trees are used as the weak learner in gradient boosting.

- *An additive model to add weak learners to minimize the loss function.* Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees.

3, XGBoost

XGBoost is an implementation of gradient boosted decision trees designed for faster speed and better performance. XGBoost stands for eXtreme Gradient Boosting. It dominates structured or tabular datasets on classification and regression predictive modeling problems. It is much faster than gradient boosting due to careful engineering of the implementation, including:

- **Parallelization** of tree construction using all of your CPU cores during training.
- **Distributed computing** for training very large models using a cluster of machines.
- **Out-of-core computing** for very large datasets that don't fit into memory.
- **Cache Optimization** of data structures and algorithms to make best use of hardware.

4, Comparing single model, Bagging and Boosting

To use Bagging or Boosting you must **select a base learner algorithm**. For example, if we choose a classification tree, Bagging and Boosting would consist of a pool of trees as big as we want.

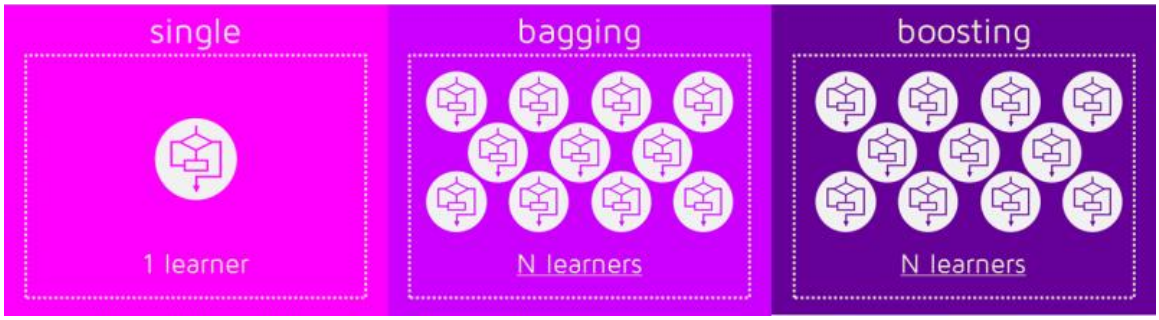

Figure 1. Comparison of single model, bagging and boosting.
Reference: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

The next step for Bagging and Boosting is to get N new training datasets by **random sampling with replacement** from the original set. In the case of Bagging, all elements have the same probability to appear in a new training dataset. In Boosting, the elements are weighted and therefore some of them will have a higher probability to appear in the new training dataset.

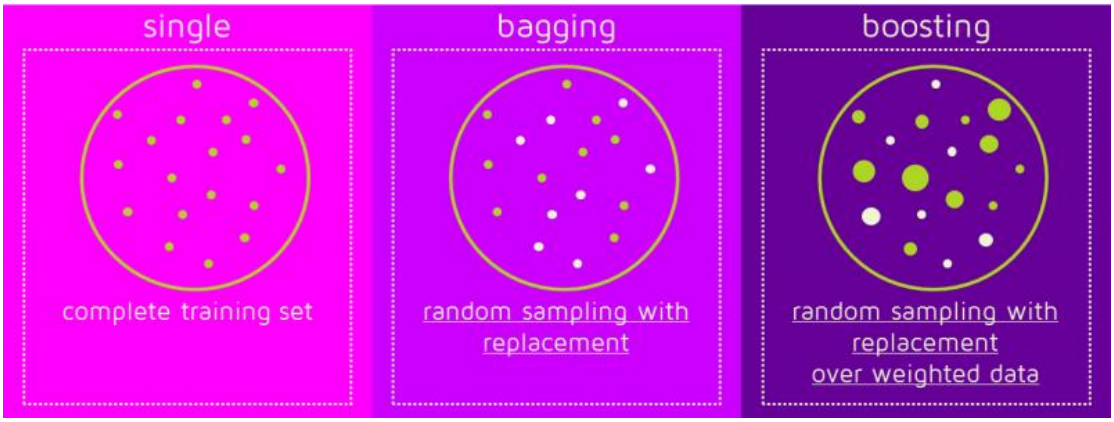

Fig 2. Training set of single model**,** Bagging and Boosting.
Reference: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

The major difference for Bagging and Boosting are the training stage is parallel for Bagging (i.e., each model is built independently) while Boosting builds the new learner in a sequential way. In Boosting algorithms each classifier is trained on data, taking into account the previous classifiers' success. After each training step, the weights are redistributed.

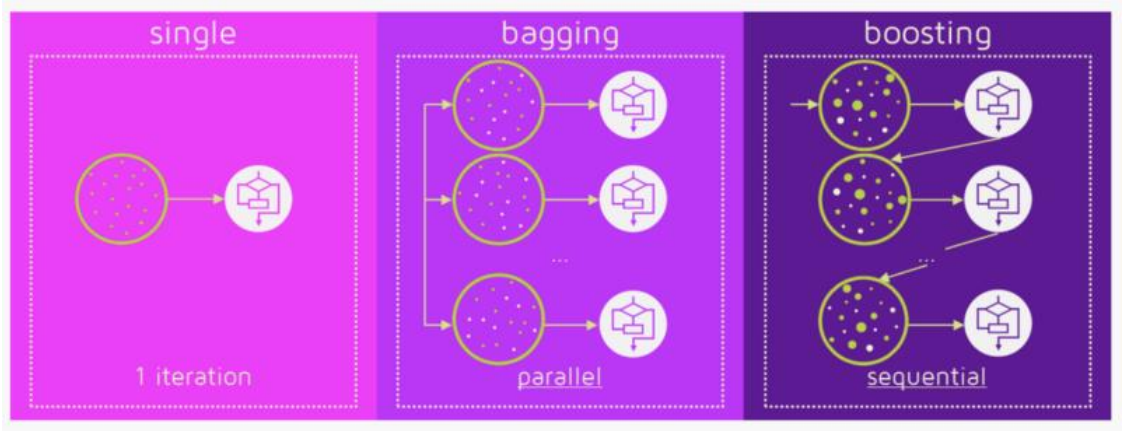

Fig 3. Single model, Bagging (independent models) and Boosting (sequential models). Reference: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

To predict the class of new data we only need to apply the N learners to the new observations. In Bagging the result is obtained by averaging the responses of the N learners (or majority vote). In Boosting, a weighted average of results from the N learners are taken.

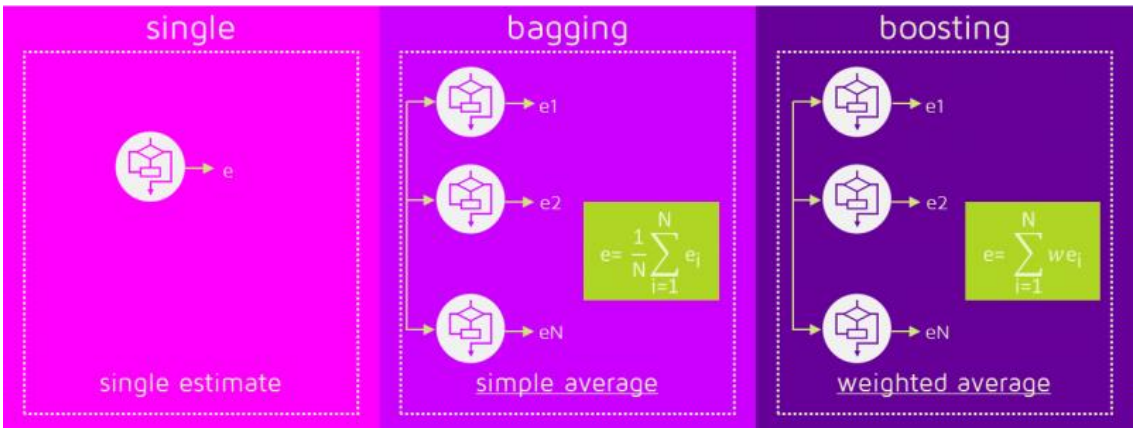

Fig 4. Getting predicted results from single model, Bagging (independent models) and Boosting (sequential models). Reference: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

References:

http://scikit-learn.org/stable/modules/ensemble.html

https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d

https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/