MACHINE LEARNING
Ensemble Learning

Trainer : Sujata Mohite
Email: sujata.mohite@sunbeaminfo.com

Overview

- Ensemble is the art of combining diverse set of learners (individual models like decision tree, logistic regression, knn) together to improvise on the stability and predictive power of the model.
- The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions.(dependent on multiple models)
- Primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one
- The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions.
- In learning models, noise, variance, and bias are the major sources of error. The ensemble methods in machine learning help minimize these error-causing factors, thereby ensuring the accuracy and stability of machine learning (ML) algorithms.
- Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting

Basic Ensemble Methods

Mode

- "mode" is the number or value that most often appears in a dataset of numbers or values.
- In this ensemble technique, machine learning professionals use a number of models for making predictions about each data point.
- The predictions made by different models are taken as separate votes.
- Subsequently, the prediction made by most models is treated as the ultimate prediction.

Mean/Average

- ensemble technique, data analysts take the average predictions made by all models into account when making the ultimate prediction.

The Weighted Average

- data scientists assign different weights to all the models in order to make a prediction, where the assigned weight defines the relevance of each model.

Advance Ensemble Learning Methods

- Ensemble
  - Bagging
    - Random Forest
  - Boosting
    - AdaBoost
    - eXtreme Gradient Based (XGBOOST)
  - Stacking

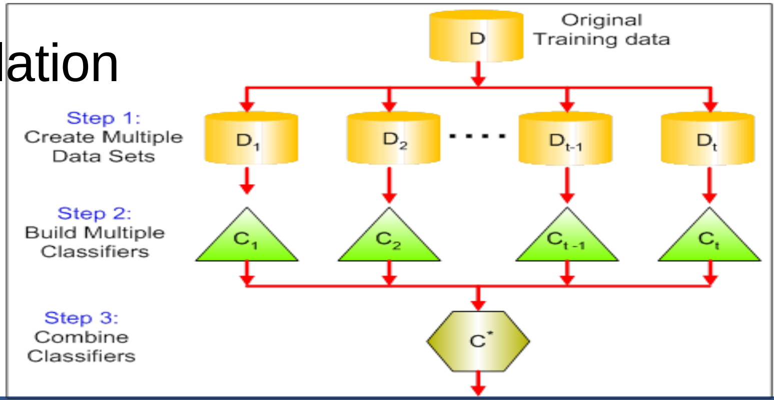
Three main classes of Ensemble Learning

- Bagging((Bootstrap Aggregating) involves fitting many decision trees on different samples of the same dataset and averaging the predictions.
- Stacking involves fitting many different models types on the same data and using another model to learn how to best combine the predictions. An iterative ensemble technique, "boosting," adjusts an observation's weight based on its last classification.
- Boosting involves adding ensemble members sequentially that correct the predictions made by prior models and outputs a weighted average of the predictions.

Bagging

Bagging

- Bagging tries to implement similar learners/ same algorithm on small sample populations and then takes a mean of all the predictions
- The primary goal of "bagging" or "bootstrap aggregating" ensemble method is to minimize variance errors in decision trees.
- The objective here is to randomly create samples of training datasets with replacement (subsets of the training data).
- The subsets are then used for training decision trees or models.
- Consequently, there is a combination of multiple models, which reduces variance, as the average prediction generated from different models is much more reliable and robust than a single model or a decision tree.
- In generalized bagging, you can use different learners on different population
- This helps us to reduce the variance error
- Algorithm
  - Random Forest (decision tree)



Boosting

Boosting

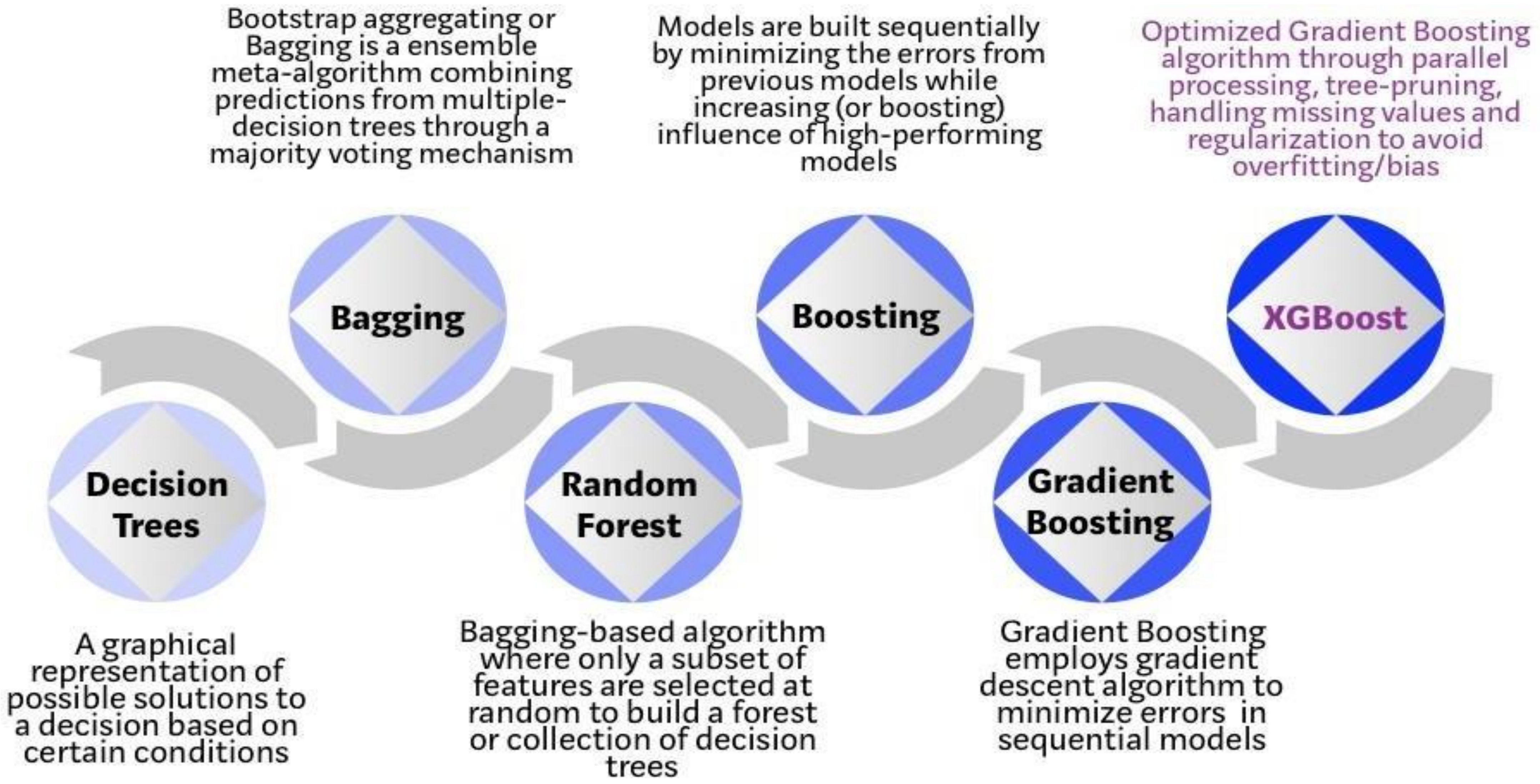
- Boosting refers to a family of algorithms that are able to convert weak learnershigh error –low accuracy) to strong learners(low error – high accuracy)
- Boosting is an iterative technique which adjust the weight of an observation based on the last classification
- An iterative ensemble technique, "boosting," adjusts an observation's weight based on its last classification.
- In case observation is incorrectly classified, "boosting" increases the observation's weight, and vice versa.
- Boosting algorithms reduce bias errors and produce strong predictive models.
- If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa
- Algorithms
  - AdaBoost
  - Gradient Boosting
  - eXtreme Gradient Boosting

XGBoost

Overview

- XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework
- XGBoost algorithm was developed as a research project at the University of Washington
- Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications
- As a result, there is a strong community of data scientists contributing to the XGBoost open source projects with ~350 contributors and ~3,600 commits on GitHub

Evolution


Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

How does it work?

- XGBoost belongs to a family of boosting algorithms that convert weak learners into strong learners
- A weak learner is one which is slightly better than random guessing

Why does it perform so well?

- XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture
- However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

System Optimization

- **Parallelization**
  - XGBoost approaches the process of sequential tree building using parallelized implementation
  - This is possible due to the interchangeable nature of loops used for building base learners; the outer loop that enumerates the leaf nodes of a tree, and the second inner loop that calculates the features
  - This nesting of loops limits parallelization because without completing the inner loop (more computationally demanding of the two), the outer loop cannot be started
  - Therefore, to improve run time, the order of loops is interchanged using initialization through a global scan of all instances and sorting using parallel threads
  - This switch improves algorithmic performance by offsetting any parallelization overheads in computation
- **Tree Pruning**
  - The stopping criterion for tree splitting within GBM framework is greedy in nature and depends on the negative loss criterion at the point of split
  - XGBoost uses 'max_depth' parameter as specified instead of criterion first, and starts pruning trees backward
  - This 'depth-first' approach improves computational performance significantly.

System Optimization

- **Hardware Optimization**
  - This algorithm has been designed to make efficient use of hardware resources
  - This is accomplished by cache awareness by allocating internal buffers in each thread to store gradient statistics
  - Further enhancements such as ‘out-of-core’ computing optimize available disk space while handling big data-frames that do not fit into memory.

Benefits

- **Parallel Computing:** It is enabled with parallel processing (using OpenMP); i.e., when you run xgboost, by default, it would use all the cores of your laptop/machine.
- **Regularization:** I believe this is the biggest advantage of xgboost. GBM has no provision for regularization. Regularization is a technique used to avoid overfitting in linear and tree-based models.
- **Enabled Cross Validation:** In R, we usually use external packages such as caret and mlr to obtain CV results. But, xgboost is enabled with internal CV function (we'll see below).
- **Missing Values:** XGBoost is designed to handle missing values internally. The missing values are treated in such a manner that if there exists any trend in missing values, it is captured by the model.
- **Flexibility:** In addition to regression, classification, and ranking problems, it supports user-defined objective functions also. An objective function is used to measure the performance of the model given a certain set of parameters. Furthermore, it supports user defined evaluation metrics as well.

Benefits

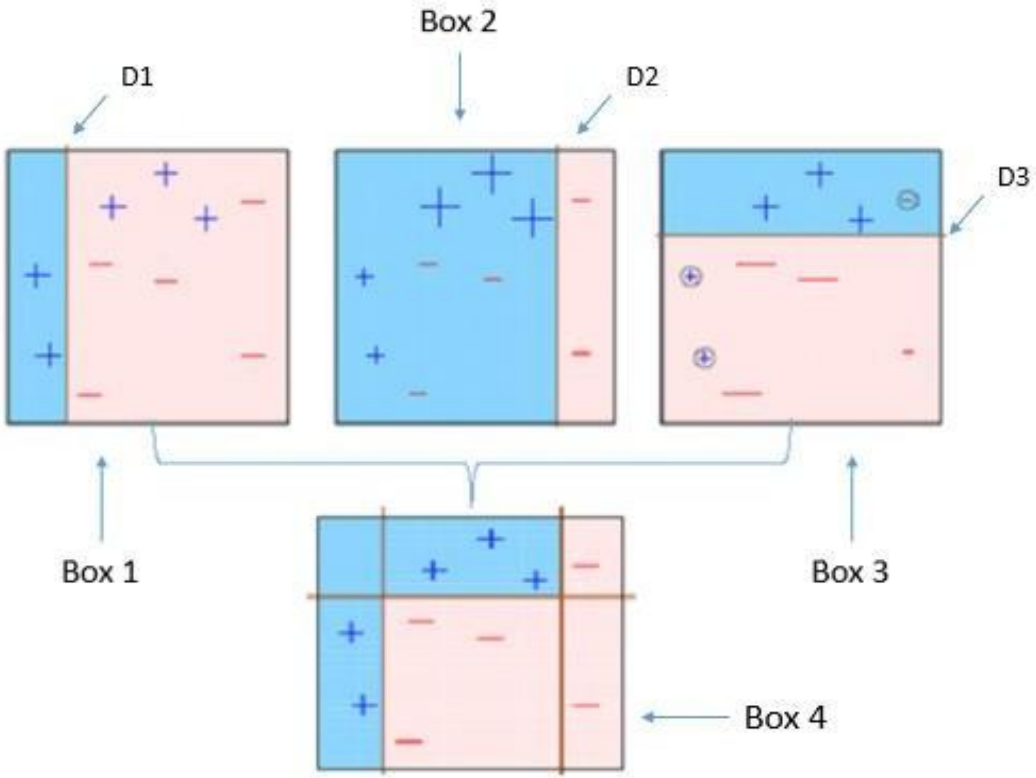
- **Availability:** Currently, it is available for programming languages such as R, Python, Java, Julia, and Scala.
- **Save and Reload:** XGBoost gives us a feature to save our data matrix and model and reload it later. Suppose, we have a large data set, we can simply save the model and use it in future instead of wasting time redoing the computation.
- **Tree Pruning:** Unlike GBM, where tree pruning stops once a negative loss is encountered, XGBoost grows the tree upto max_depth and then prune backward until the improvement in loss function is below a threshold.

How does it work?

- It combines a set of weak learners and delivers improved prediction accuracy
- At any instant t, the model outcomes are weighed based on the outcomes of previous instant t-1
- The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher
- Note that a weak learner is one which is slightly better than random guessing

How does it work?

- **1. Box 1**: The first classifier (usually a decision stump) creates a vertical line (split) at D1. It says anything to the left of D1 is **+** and anything to the right of D1 is **-**. However, this classifier misclassifies three **+** points.
- **Note** a Decision Stump is a Decision Tree model that only splits off at one level, therefore the final prediction is based on only one feature.
- **2. Box 2**: The second classifier gives more weight to the three **+** misclassified points (see the bigger size of **+**) and creates a vertical line at D2. Again it says, anything to the right of D2 is **-** and left is **+**. Still, it makes mistakes by incorrectly classifying three **-** points.
- **3. Box 3**: Again, the third classifier gives more weight to the three **-** misclassified points and creates a horizontal line at D3. Still, this classifier fails to classify the points (in the circles) correctly.
- **4. Box 4**: This is a weighted combination of the weak classifiers (Box 1,2 and 3). As you can see, it does a good job at classifying all the points correctly.



Stacking

Stacking

- Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor
- The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features
- The base level often consists of different learning algorithms and therefore stacking ensembles are often heterogeneous

Thank You!!