Decision Tree Ensembles- Bagging and Boosting

Random Forest and Gradient Boosting

We all use Decision Tree technique on daily basis to plan our life, we just don’t give a fancy name to those decision-making process.

Businesses use these supervised machine learning techniques like Decision trees to make better decisions and make more profit. Decision trees have been around for a long time and also known to suffer from bias and variance. You will have a large bias with simple trees and a large variance with complex trees.

*Ensemble methods*, which combines several decision trees to produce better predictive performance than utilizing a single decision tree. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner.

Let’s talk about few techniques to perform ensemble decision trees:

1. Bagging

2. Boosting

Bagging *(Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Here idea is to create several subsets of data from training sample chosen randomly*with replacement*. Now, each collection of subset data is used to train their decision trees. As a result, we end up with an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree.*

*Random Forest* is an extension over bagging. It takes one extra step where in addition to taking the random subset of data, it also takes the random selection of features rather than using all features to grow trees. When you have many random trees.

It’s called Random Forest .

Let’s look at the steps taken to implement Random forest:

1. Suppose there are N observations and M features in training data set. First, a sample from training data set is taken randomly with replacement.

2. A subset of M features are selected randomly and whichever feature gives the best split is used to split the node iteratively.

3. The tree is grown to the largest.

4. Above steps are repeated and prediction is given based on the aggregation of predictions from n number of trees.

Advantages of using Random Forest technique:

* Handles higher dimensionality data very well.
* Handles missing values and maintains accuracy for missing data.

Disadvantages of using Random Forest technique:

* Since final prediction is based on the mean predictions from subset trees, it won’t give precise values for the regression model.

*Boosting is another ensemble technique to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analyzing data for errors. In other words, we fit consecutive trees (random sample) and at every step, the goal is to solve for net error from the prior tree.*

When an input is misclassified by a hypothesis, its weight is increased so that next hypothesis is more likely to classify it correctly. By combining the whole set at the end converts weak learners into better performing model.

Gradient Boosting is an extension over boosting method.

*Gradient Boosting= Gradient Descent + Boosting.*

It uses gradient descent algorithm which can optimize any differentiable loss function. An ensemble of trees are built one by one and individual trees are summed sequentially. Next tree tries to recover the loss (difference between actual and predicted values).

Advantages of using Gradient Boosting technique:

* Supports different loss function.
* Works well with interactions.

Disadvantages of using Gradient Boosting technique:

* Prone to over-fitting.
* Requires careful tuning of different hyper-parameters

Random Forest

Decision trees are easy to build and easy to use and easy to interprete as well.

but in practice not that awesome

Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy

in other words works great on training data or data used to create them, but are not flexible when it comes to classifying new samples.

Then Random forest combines the simplicity of decision trees with flexibility resulting in vast improvement in accuracy.

Step 1: create a "bootstrapped" dataset

To create a bootstrapped dataset that is the same size as the original, we just randomly select samples from the original dataset

The important detail is that we're allowed to pick the same sample more than once

Step 2: we create a decision tree using the bootstrapped dataset, but only use a random subset of variables (or columns)at each step and similarly like the root

We randomly select other variables as candidates and we build the tree as usual, but only considering a random subset of variables at each step

So one tree is made,

Now go back to step1 and repeat: making a new bootstrapped dataset and build a tree considering subset of variables at each step

Ideally we'd do this many a times

Using a bootstrapped sample and constructing only a subset of the variables at each step results in a wide variety of trees

This variety is what makes Random forest more effective than individual decision trees

So now if you have a new unseen data, we take the data and run it down the first tree that we made and then run it through second and so on...and we keep track of that. After running the data down all of the trees in the random forest we see which options receive more votes.

Bootstrapping the data plus using the aggregate to make a decision is called Bagging

When we created the bootstrapped dataset we allowed duplicate entries in the bootstrapped dataset as a result some entries may not be included in the bootstrapped dataset those will be called out-of-bag-Dataset. Since out of bag data is not used to create tree so we can run it through the tree and sees if it correctly classifies as its desired output or not? then we run those out of bag samples through all other trees that were built without it and labels with the most Vote wins, it is the label that we assign to those out of bag samples.

Ultimately we can measure how accurate our random forest is by the proportion of out of bag samples that were correctly classified by the random forest

The proportion of out of bag samples that were incorrectly classified is the "Out-of-Bag Error".

**Why Random forest algorithm?**

To address why random forest algorithm I am giving you the below advantages.

* The same **random forest algorithm** or the random forest classifier can use for both classification and the regression task.
* Random forest classifier will **handle the missing** values.
* When we have more trees in the forest, we use random forest classifier to avoid the **over fitting**.

**Boosting Algorithm**

**AdaBoostAlgorithm:**  
  
In this example, we focus on boosting algorithm, especially

Ad boost algorithm.

1. Start by creating a tree on training data, where each observation is assigned an equal weight.
2. Then compute the predicted classification and weights are redetermined and assign greaterweight to those observations that are difficult to classify and lower weights to those that are easy to classify. Weights for all observations must sum to 1.
3. Second tree is grown on weighted data (weights are based on residuals or misclassification error). Idea is to improve prediction of first tree.
4. Weights are redetermined and assign higher weights if it is classified incorrectly.
5. New Model is now Tree 1 + Tree 2
6. Compute residuals or classification error from this new 2-tree model (Tree1 + Tree2) and grow 3rd tree to predict revised residuals.
7. Then subsequent trees help in classifying observations that are not well classified by preceding trees.
8. The final prediction is a weighted sum of the predictions made by previous tree models.

In this process, we have created many simple decision trees, where each tree is built for the prediction errors (classification error) of the previous tree.

**HOW Boosting works?**

We weight misclassified observations in such a way that they get properly classified in future iterations.

**Main Idea**

* Combine several simple decision trees (can be hundred or thousands)
* Each tree complements the previous ones (Avoid redundancy)
* Keep track of the errors of the previous trees

This technique is faced with a problem of over fitting.