Difference between Random Forest and Bagging

Random Forest is a bagging algorithm. Random forest reduces variance of a large number of "complex" models with low bias. While boosting reduces bias of a large number of "small" models ("weak" models) with low variance.

Bagging and Boosting are similar in that they are both **ensemble techniques**, where a set of weak learners are combined to create a strong learner that obtains better performance than a single one.

ENSEMBLE LEARNING

Ensemble methods combine several decision trees classifiers to produce better predictive performance than a single decision tree classifier. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner, thus increasing the accuracy of the model. When we try to predict the target variable using any machine learning technique, the main causes of difference in actual and predicted values are **noise, variance, and bias**. Ensemble helps to reduce these factors (except noise, which is irreducible error).

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.

BAGGING

Bootstrap Aggregation (or Bagging for short), is a simple and very powerful ensemble method. Bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees.

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.

BOOSTING

Boosting refers to a group of algorithms that utilize weighted averages to make weak learners into stronger learners. Unlike bagging that had each model run independently and then aggregate the outputs at the end without preference to any model. Boosting is all about “teamwork”. Each model that runs, dictates what features the next model will focus on.

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 analysing 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.

RANDOM FOREST

Random Forests are an improvement over bagged decision tree.

A problem with decision trees like CART is that they are greedy. They choose which variable to split on using a greedy algorithm that minimizes error. As such, even with Bagging, the decision trees can have a lot of structural similarities and in turn have high correlation in their predictions.

Combining predictions from multiple models in ensembles works better if the predictions from the sub-models are uncorrelated or at best weakly correlated. Random forest changes the algorithm for the way that the sub-trees are learned so that the resulting predictions from all of the subtrees have less correlation.
It is a simple tweak. In CART, when selecting a split point, the learning algorithm is allowed to look through all variables and all variable values in order to select the most optimal split-point.

The Random forest algorithm changes this procedure so that the learning algorithm is limited to a random sample of features of which to search.

The number of features that can be searched at each split point (m) must be specified as a parameter to the algorithm. You can try different values and tune it using cross validation.

- For classification a good default is: m = sqrt(p)
- For regression a good default is: m = p/3

Where m is the number of randomly selected features that can be searched at a split point and p is the number of input variables.

You can get an idea of the mechanism from the name itself-"random forests". A collection of trees is a forest, and the trees are being trained on subsets which are being selected at random, hence random forests.

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

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.

Random forests or **random decision forests** are an ensemble learning method
for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean

prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set

Difference between Random Forest and Decision Tree

A random forest is a collection or ensemble of decision trees.

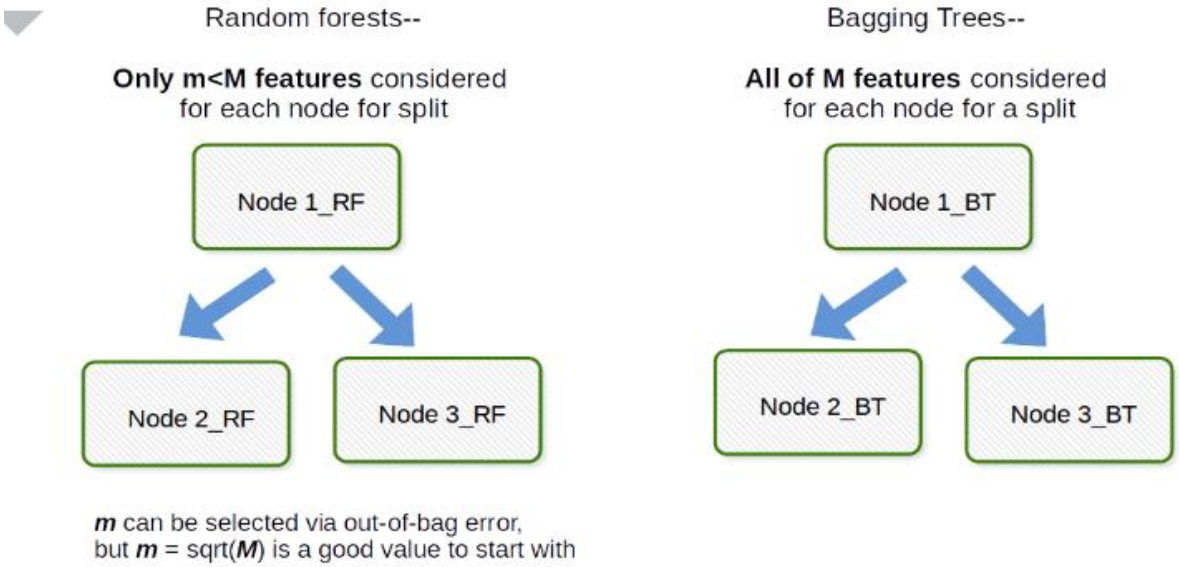
A decision tree is built using the whole dataset considering all features, but in random forests a fraction of the number of rows is selected at random and a particular number of features are selected at random to train on and a decision tree is built on this subset.

Similarly, a number of decision trees are grown, each will probably have a different subset since it is being randomly selected and hence each decision tree will be different, and each tree will vote for a particular class and the class which gets maximum number of votes is the predicted class.

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

Difference between Random Forest and Bagging

The fundamental difference between bagging and random forest is that in Random forests, only a subset of features is selected at random out of the total and the best split feature from the subset is used to split each node in a tree, unlike in bagging where all features are considered for splitting a node.



Bagging has a single parameter, which is the number of trees. All trees are fully grown binary tree (unpruned) and at each node in the tree one searches over all features to find the feature that best splits the data at that node.

Random forests has 2 parameters:

1. The first parameter is the same as bagging (the number of trees)
2. The second parameter (unique to randomforests) is **mtry** which is how many features to search over to find the best feature. this parameter is usually 1/3*D for regression and sqrt(D) for classification. thus, during tree creation randomly **mtry** number of features are chosen from all available features and the best feature that splits the data is chosen.

.

The introduction of random forests proper was first made in a paper by Leo Breiman. This paper describes a method of building a forest of uncorrelated trees using a CART like procedure, combined with randomized node optimization and *bagging*. In addition, this paper combines several ingredients, some previously known and some novel, which form the basis of the modern practice of random forests, in particular:

1. Using out-of-bag error as an estimate of the generalization error.
2. Measuring variable importance through permutation.

From bagging to random forests

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "**feature bagging**". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the *B* trees, causing them to become correlated.

Typically, for a classification problem with *p* features**, √p** (rounded down) features are used in each split. For regression problems the inventors recommend **p/3** (rounded down) with a minimum node size of 5 as the default.

Bagging is a common ensemble method that uses bootstrap sampling. Random forest is an enhancement of bagging that can improve variable selection.

Regression or classification fits generated from different bootstrap samples are correlated because of the observations that have been selected in both samples. The higher the correlation, the more similar the fit from each bootstrap and the smaller the mitigating effect of the consensus in reducing variance. For variable selection problems, strongly predictive variables that are selected in most bootstrap samples induce a strong correlation among the fits, reducing the utility of bagging.

To limit the impact of such variables, a simple but clever modification of CART bagging is used: a random forest. In this approach, at each node of the tree, a subset **m** of the **p** variables in the data is selected at random, and only these **m** variables are considered for the partition at the node. This random selection of variables reduces the similarity of trees grown from different bootstrap samples—even two trees grown from the same bootstrap sample will likely differ. Once a sufficiently large forest of trees has been grown, the results are bagged in the usual way.

There will be a value of **m** that optimizes the variance reduction relative to the computational cost. This can be estimated using the OOB error as a function of ***m***. Random forests are quite robust with respect to *m*, and rules of thumb such as using **m = p/3** for regression and **m = √p** for classification are sometimes used.

Bagging (or Bootstrap Aggregating), uses a different random subset of the original dataset for each model in the ensemble. Specifically, BigML uses by default a sampling rate of 100% with replacement for each model, this means that some of the original instances will be repeated and others left out.

Random Decision Forests extend this technique by only considering a random subset of the input fields at each split. Generally, Random Decision Forests are the most powerful type of ensemble. For datasets with many noisy fields you may need to adjust a Random Decision Forest's "random candidates" parameter for good results. Bagging, however, does not have this parameter and may occasionally give better out-of-the-box performance.

The Boosting Ensemble technique is significantly different. With Boosted Trees, tree outputs are additive rather than averaged (or decided by majority vote). Individual trees in a Boosted Tree differ from trees in bagged or random forest ensembles since they do not try to predict the objective field directly. Instead, they try to fit a "gradient" to correct mistakes made in previous iterations. This unique technique, where each tree improves on the imperfect predictions of the previously grown tree, lets you predict both categorical and numeric fields.

Out-of-bag (**OOB**) **error**, also called **out-of-bag estimate**, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing bootstrap aggregating (bagging) to sub-sample data samples used for training. OOB is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.

Subsampling allows one to define an out-of-bag estimate of the prediction performance improvement by evaluating predictions on those observations which were not used in the building of the next base learner. Out-of-bag estimates help avoid the need for an independent validation dataset, but often underestimates actual performance improvement and the optimal number of iterations