Ensemble Learning

Ensemble learning is the process of combining predictions from multiple machine learning models. These models are known as weak learners. By combining several weak learners to create a strong learner that can outperform any individual machine learning model. Ensemble learning combines multiple models to obtain better predictive performance. The two main types of ensemble learning are,

1. Bagging (or Bootstrap aggregating)
2. Boosting

1. Bagging

Bagging is an ensemble algorithm composed of two parts: Bootstrapping and aggregation.

Bootstrapping creates several subsets of original training data chosen randomly with replacement. Each subset has equal size observations and can be used to train models in parallel. By sampling with replacement, some observations may be repeated in each new training dataset. Models are trained on each of these subsets independently and the results are aggregated for the final prediction. The final prediction is decided from these models with the most votes (mode) in a classification setting. In Regression, the final prediction is an average of all the predictions. An example of bagging algorithms is the Random Forest algorithm.

2. Boosting

Boosting is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. In this technique, models are built sequentially. An example for boosting algorithms is the AdaBoost (Adaptive Boosting) Algorithm.

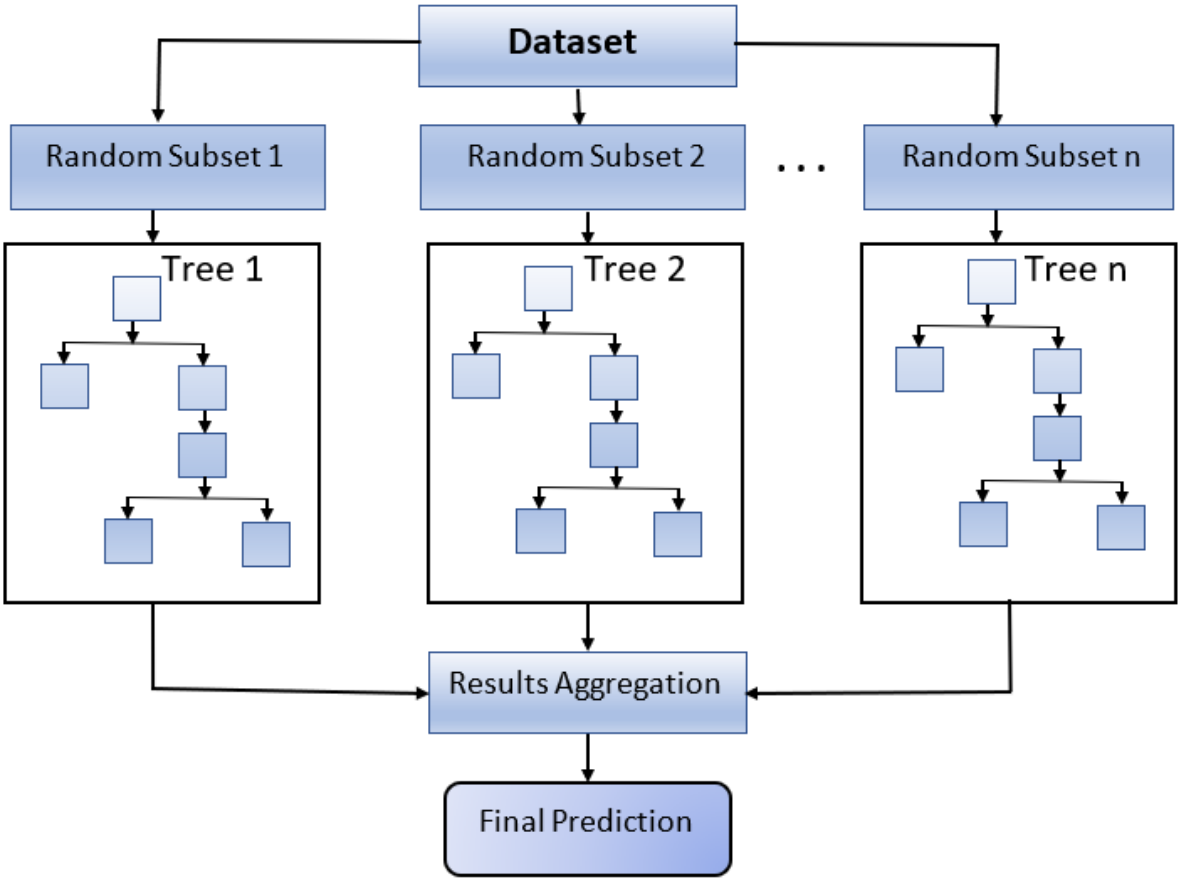
Random Forest

Random Forest is an ensemble learning algorithm that is used for classification and regression problems.

Random forest builds multiple decision trees and combines them to get a more accurate and stable prediction. Decision trees usually work top-down, by choosing a variable at each step that **best splits** the set of items. The end nodes can have a category (classification) or a continuous number (regression). But the drawback of decision trees is that the learning mechanism in decision trees is very sensitive to even small changes in data. Also, larger decision trees generally tend to overfit the data.

Random forest prevents this by allowing a whole bunch of decision trees to work together to get a better and more robust prediction. In classification, the prediction from each decision tree is a vote. The final prediction comes from the prediction with the most votes (mode). In Regression, the final prediction is an average of all predictions. If the data is modified a little, and use all those decision trees to make a prediction then prediction may be more robust. Because the prediction is not dependent on one model.

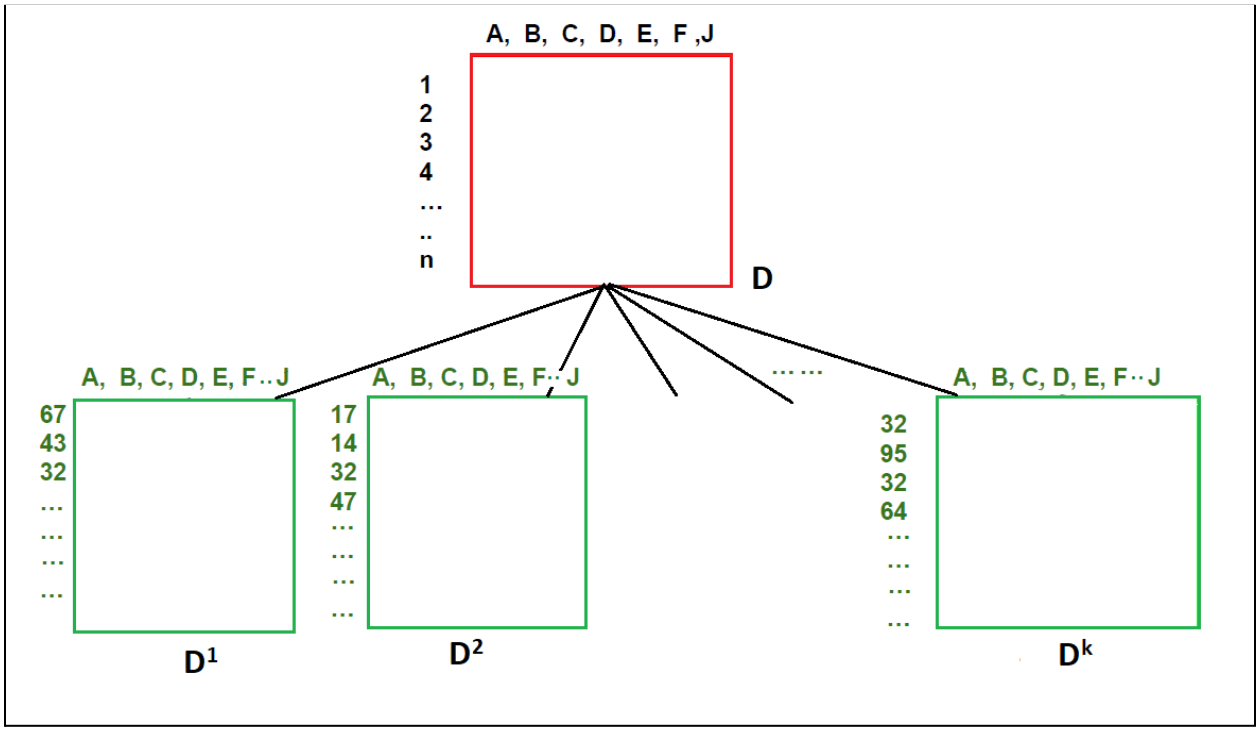
Thus each model is built on a subset of original training data, sampled with replacement. The selection of the data subset is completely randomized. This random sampling helps reduce variance in the data. This sampling approach is called bootstrap sampling. Then the results from different trees are aggregated (mean for regression and mode for classification) for the final prediction. The example representation is shown below.



Row Sampling

Consider a Dataset 'D'. This is the original training dataset that has 'n' number of observations and 10 features say A to J. Bagging randomly selects some observations from training dataset 'D' and makes training Dataset 'D^1'. Similarly again, it takes some random observations from the training dataset and makes a new dataset 'D^2'. In the same way, it makes a new Dataset D^3, D^4 to D^k.

Thus each new training dataset is made with a random selection of observations with replacement. With replacement means that some observations may be present in more than one dataset. It may be in D^1 and D^3.



In the above representation, the observation '32' is present in the many training datasets. The original dataset has 'n' observations. And each training dataset can have **n'** observations, where n'<=n. Now, we can build a decision tree for each of those datasets. Each decision tree differs slightly from one another because the data that we used to build is slightly different.

Column Sampling

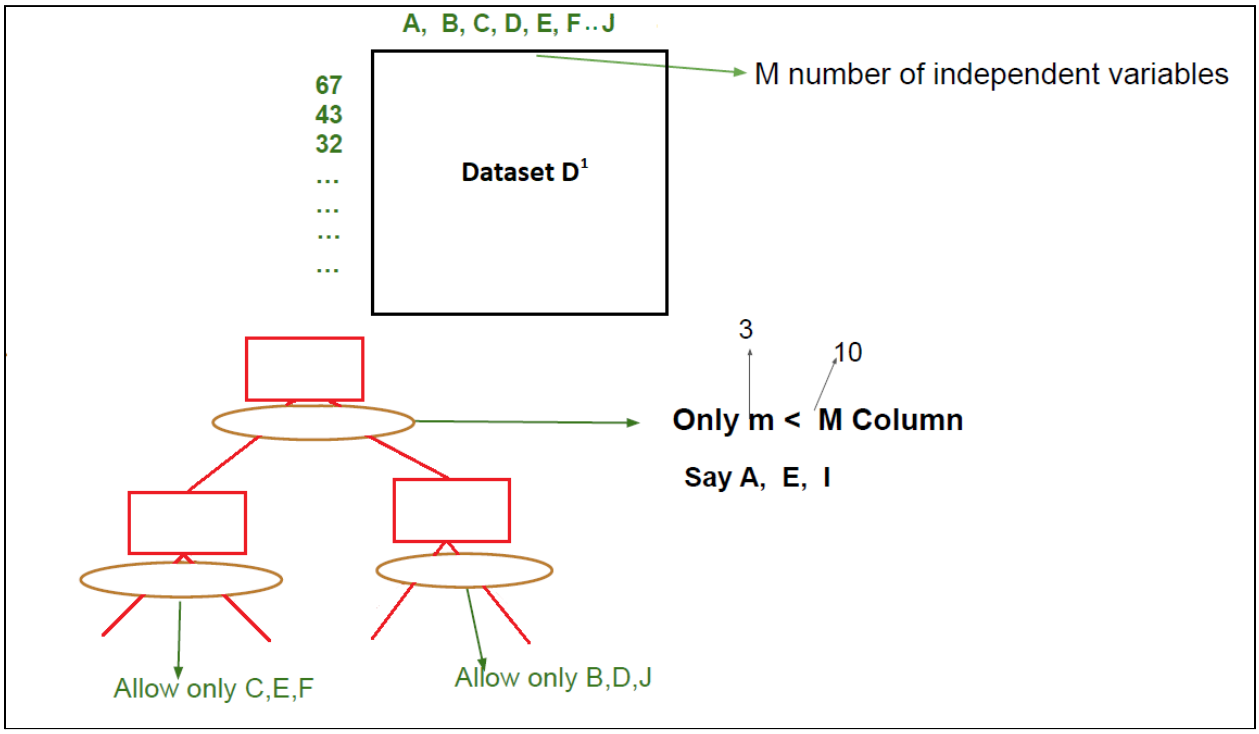
The bagging technique considers only, **'Row(observations) sampling with replacement'**. That is, all training datasets are largely going to break off at the same features throughout each model. But the Random forest algorithm also considers **'Column(Features) sampling with replacement'** with row sampling. Instead of splitting at the same features at each node, each tree can be split based on different features. The features considered for partitioning at each node are a random subset of the original set of features.

From the above example, consider the training dataset **'D^1'** with **'M'** number of independent variables, say **A** to **J**. Now we can build a decision tree for that dataset. Any decision tree algorithm works top-down, by choosing a feature at each step that best splits the set of observations. It considers all the independent features for splitting. And for each independent variable, we ask for the best split and we choose the split as the best split amongst all the variables (Ex: CART algorithm).

But in a random forest algorithm, we want each of the trees in the forest to be a little diverse from one another. It tries a subset of all the features for splitting each node in each decision tree. In our example, it considers only the **'m'** number of features which is smaller than **'M'**.

Let **'m'=3**. To build the root node, three features are selected randomly say **A, E, I,** and whichever feature gives the best split is used to split the node. The sample representation is shown below.

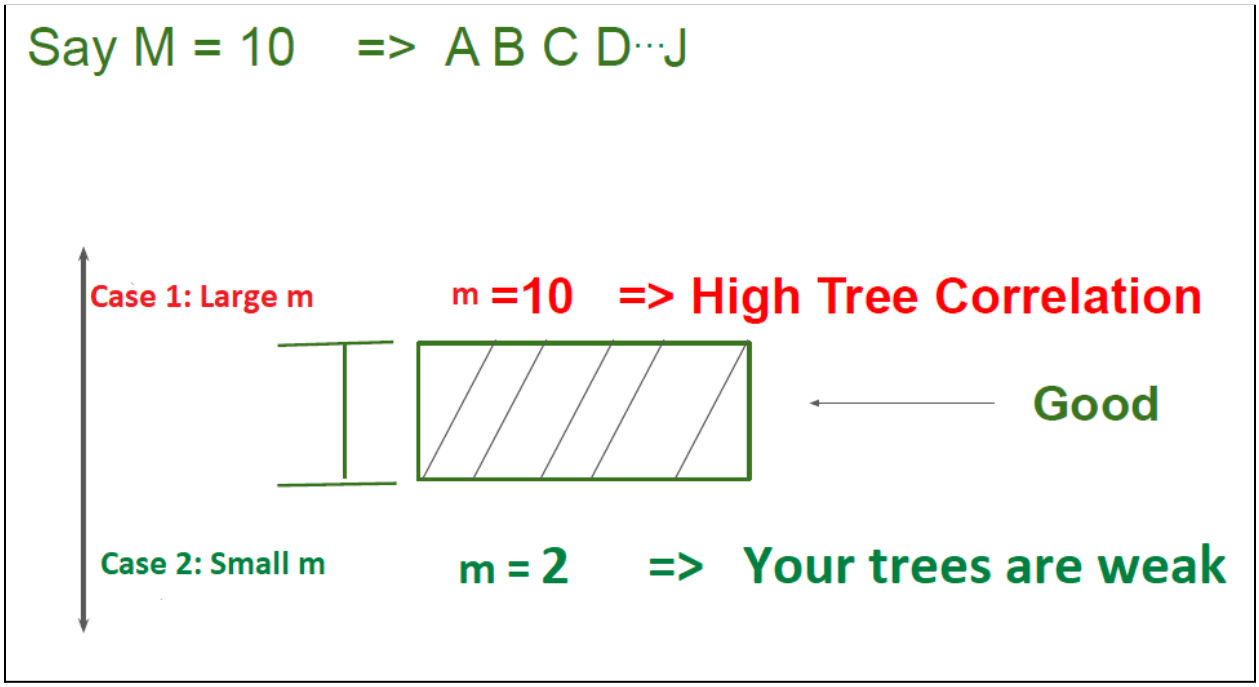
Similarly **'m'** features are randomized for each split. That is after the root node is chosen, to split a left and right subtree, again three features are randomly selected, say **C, E, F** and **B, D, J** respectively. The same process is repeated iteratively, and we allow the tree to grow fully. This ensures that each of these datasets is slightly different from one another and also the trees that we built out of those datasets are even more different from one another. Thus it gives a very diverse forest. Though we allow all the decision trees to grow larger, the final results do not overfit. While individual trees tend to overfit training data, averaging corrects this. So no need to prune the fully grown trees in random forests.



How to choose the value of 'm'?

The number of features **'m'** for splitting each node, from the total number of features **'M'** is to be chosen carefully to avoid any correlated or weak trees. If 'm' is very large, say 'm' is equal to 'M', though the datasets are slightly different, the trees built from each dataset become very correlated. If 'm' is very small, say 'm' is equal to '2', then the chance of actually catching one of the important variables in the splitting mechanism becomes lower. Then those trees have a very weak ability to predict.

Both of these extreme values (very large and very small) of ‘m’ are considered as bad choices. So depending on the dataset, the ‘m’ value somewhere between very large and very small can be considered. The sample representation is shown below.



If we have one tree, it is extremely sensitive to the data. The large set of trees are diverse enough to provide robustness. But the trees should not be too diverse from one another so that their strength becomes lower. So a good choice of ‘m’ gives us a much better prediction.

Random Forest algorithm summary

- Random Sampling with replacement
- For each subset, build a decision tree. However, only use ‘m’ randomly pick independent variables for each node’s branching possibilities
    - Do not prune
- While predicting:
    - Use each tree to make individual predictions
    - Combine predictions using voting:
        - Means for regression
        - Modes for classification

Advantages of Random forest algorithm

- It is flexible to both classification and regression problems
- It works well with both categorical and continuous values
- Does not overfit and is highly stable as the final prediction is based on majority voting or averaging
- Handles higher dimensionality data very well. Feature space is reduced because each tree does not consider all the attributes
- It automates missing values present in the data
- Normalizing of data is not required as it uses a rule-based approach

Disadvantages of Random forest algorithm

- It requires much computational power as well as resources as it builds numerous trees to combine their outputs.
- It also requires much time for training as it combines a lot of decision trees to determine the class.
- Due to the ensemble of decision trees, it also suffers model interpretability