Machine Learning to build Intelligent Systems

Manas Dasgupta

Random Forest Classifier

Structure of this Module

Random Forest

TOPICS
Recap on Decision Trees
Ensemble Techniques and Bagging Process
Introduction to Random Forest
Gini Index and Entropy
Steps in the building of a Random Forest Classifier
Random Forest in Practice
Measuring Performance
Out of Bag Error
Credit Default Prediction

Random Forest - Introduction

Random Forest *is a Supervised Learning Algorithm that is essentially an* ***Ensemble*** *of randomly created Uncorrelated Decision Trees using the* ***Bagging Process****.*

Random Forest can be used in Classification as well as in Regression. For this discussion, we will limit ourselves to Classification only.

Ensemble Methods

	Classification Result
Decision Tree 1	1
Decision Tree 2	1
Decision Tree 3	1
Decision Tree 4	0
Decision Tree 5	1
Decision Tree 6	0
Decision Tree 7	1
Decision Tree 8	0
Decision Tree 9	1
Decision Tree 10	1

Final Classification – ‘1’

In Random Forest, an **Ensemble of Decision Trees** are created that produce results (classification labels) on each observation and the classification results of each individual observation are finalized through a **voting system (wisdom-of-the-crowd)** as mentioned above.

Let’s take the following example of results on classification on an observation from a Random Forest of 10 trees.

Ensemble of Trees

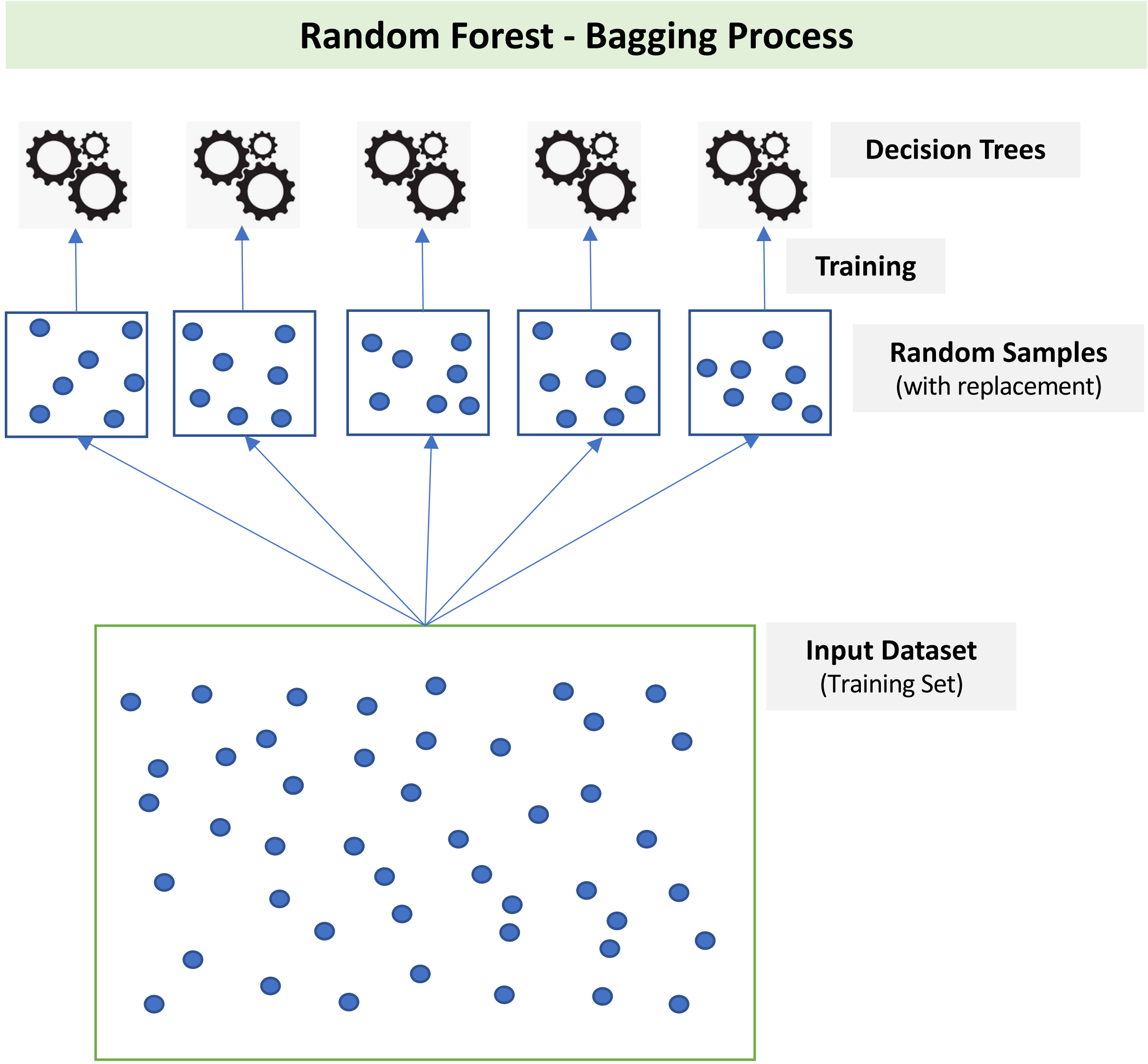
- **Uncorrelated Decision Trees**: One of the criteria for the Random Forest Ensemble of Decision Trees to be effective is that the individual ***trees have to be uncorrelated***. Any correlation between multiple trees will increase the error rate of the Random Forest Classifier.
- Random Forest Learner takes in a ***random subset of features*** for the training. The smaller the subset is compared to the total number of features, the lesser the correlation and lesser the error rate.
- The other aspect of trees being diverse is that every tree is trained with a **randomly different subset of data** using the **Bagging Process**.
- Given that a Ensemble/Forest of Decision Trees are created, the Random Forest Classifier uses the **voting method** to arrive at the final classification decisions.
- The optimisation of each Decision Tree will use the same Decision Trees methods, e.g., the Gini **Index or Entropy Scores**.

Bagging Process

One of the ways to get a diverse set of classifiers is to use the ***<u>Bagging Method</u>***. The Bagging approach uses the Training Algorithm (Decision Tree in the case of Random Forest) on ***different random subsets*** from the training set.

When sampling is performed *with* replacement, this method is called ***bagging*** (short for "***bootstrap aggregating***").

When sampling is performed *without* replacement, it is called "***pasting".***



Advantages of Random Forest

- Efficient on Large Datasets
- Can handle Large Feature sets
- Determines Feature importance
- Can handle missing data, can work with features with missing data (Decision Tree attributes)
- Can work well with unbalanced data sets
- Can work well with outlier data
- Can work well with Categorical Data
- No Scaling of Data required

Key Differences with a Decision Tree

- Decision Tree is a single tree, Random Forest is an Ensemble of Trees that uses the Bagging and Voting methods
- In Random Forest, the trees ingest a subset of the data from the Input dataset
- Each Tree in a Random Forest takes in a subset of the Features so that every tree in the forest is different. This is to introduce diversity.
- However, each tree in a Random Forest (albeit with a subset of data and features) will be created much the same manner as a normal Decision Tree.
- Gini Index or Entropy are the measures that will be considered in node branching decisions.
- Also, the same Hyperparameters that are used to drive accuracy and generalisability of the model are used in Decision Trees and Random Forest.

Decision Tree Hyperparameters

- **Criterion**
- **max_features**
- **max_depth**
- **min_samples_split**
- **min_samples_leaf**
- **min_weight_fraction_leaf**
- **max_leaf_nodes**
- **min_impurity_split**

Bagging Hyperparameters

Apart from the Decision Tree hyperparameters, the Random Forest Classifier will have a set of Bagging Hyperparameters that will enable it to control the Bagging aspects of Random Forest.

Bagging Hyperparameters	
bootstrap	True/False: True = Bagging / False = Pasting
n_jobs	The number of CPU cores to be used. -1 is for all available cores.
n_estimators	Number of Decision Trees to be built in the Forest.
oob_score	Whether to use OOB Score for evaluating the ensemble
max_samples	The %age of Input data that is to be used in the Subsamples. (or the number of Samples)

Out of Bag (OOB) Score: Since a predictor never sees the oob instances during training, it can be evaluated on these instances, without the need for a separate validation set. You can evaluate the ensemble itself by ***averaging out the OOB evaluations of each predictor***.

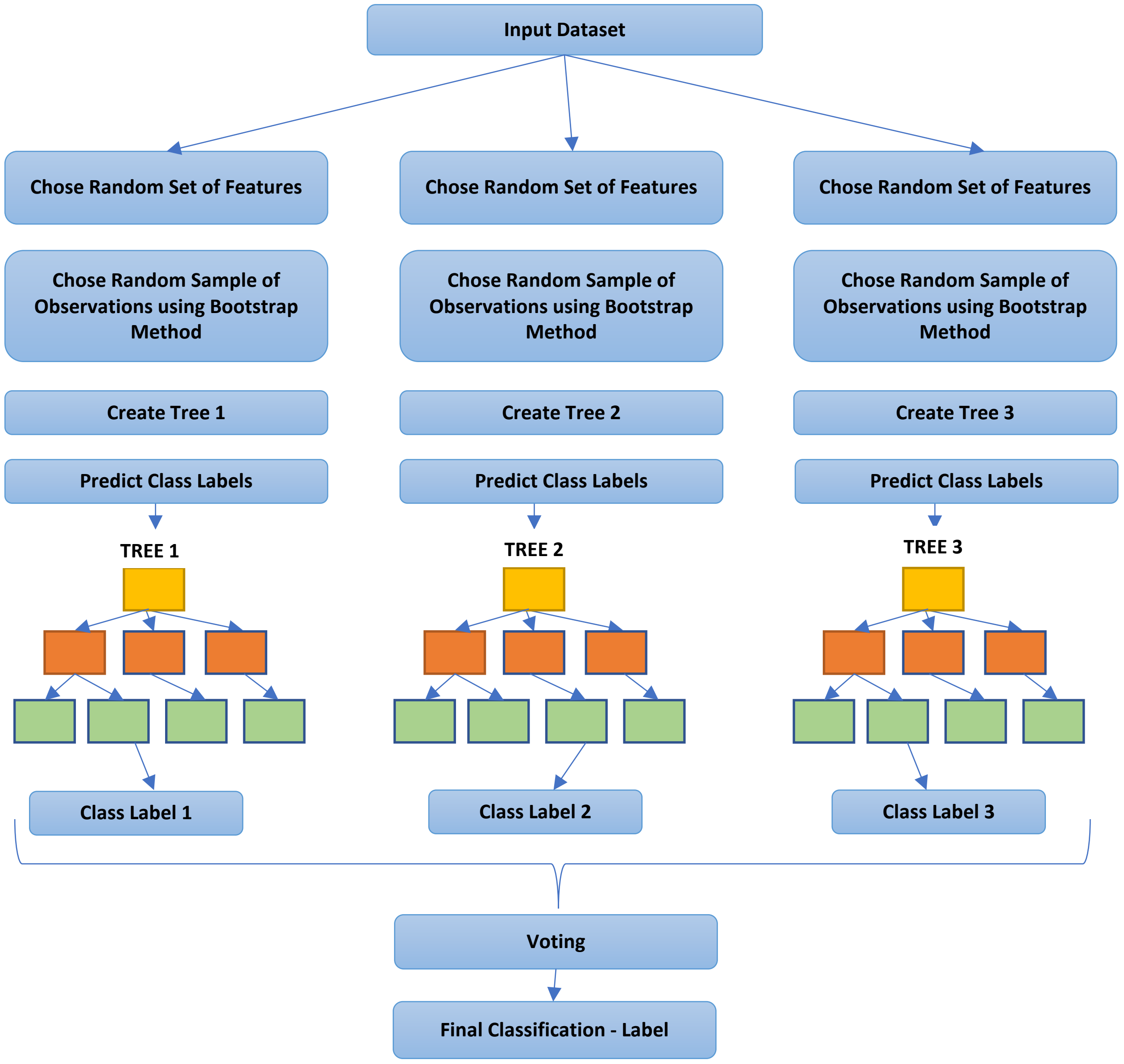
Steps in building of a Random Forest

- Randomly select ‘k’ attributes from total ‘m’ attributes where k < m, the default value of k is generally √m.
- Among the k attributes, calculate the node ‘d’ using the **best split point.**
- Split the node into a number of nodes using the **best split method**.
- Repeat the previous steps build an individual decision tree
- Build a forest by repeating all steps for n number times to create n number of trees

After the random forest trees and classifiers are created, predictions can be made using the following steps:

- Run the test data through the rules of each decision tree to predict the outcome and then
- Store that predicted target outcome
- Calculate the votes for each of the predicted targets
- Output the most highly voted predicted target as the final prediction

Random Forest Steps



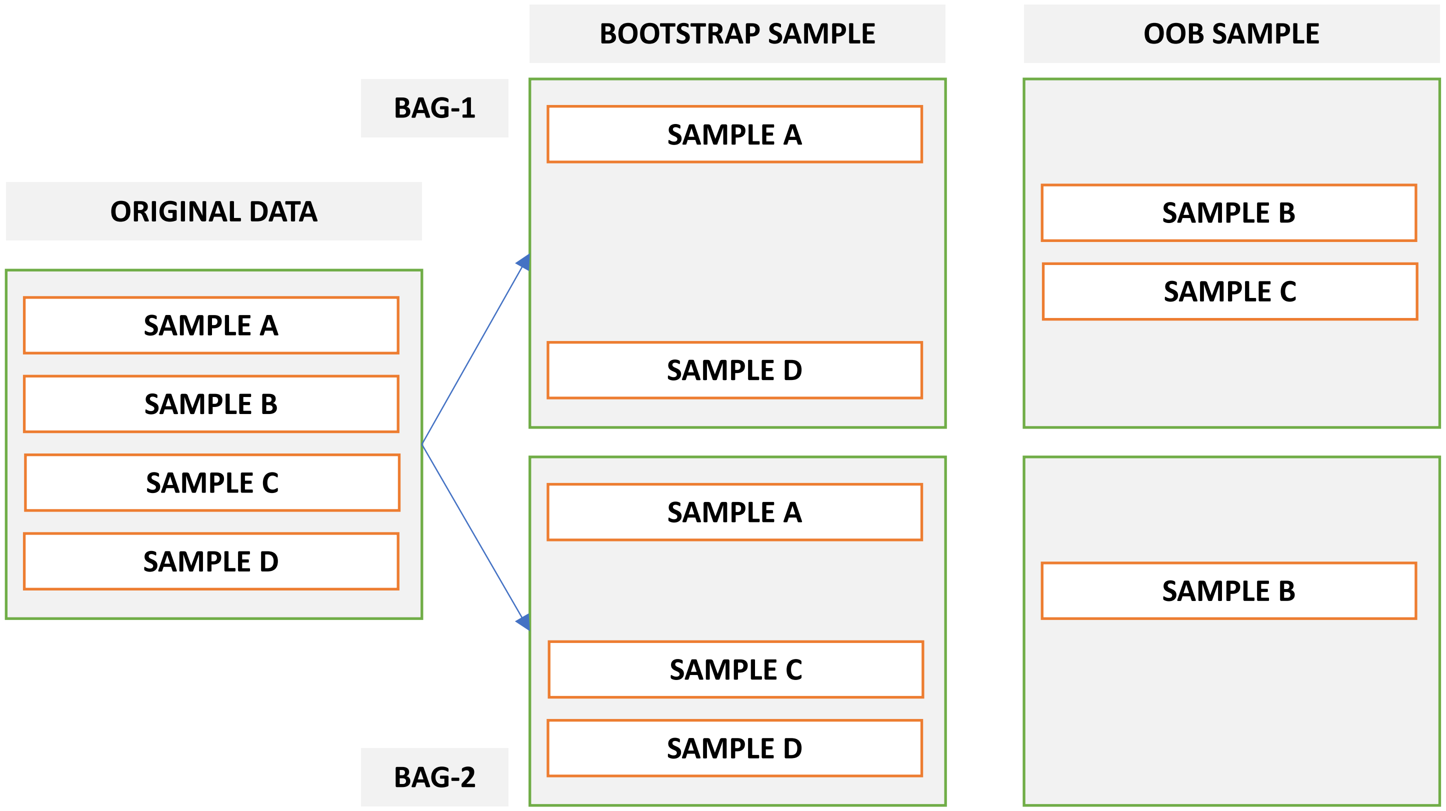
Bagging Process & Out of Bag Error

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).**

OOB error 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.

Bootstrap aggregating allows one to define an ***out-of-bag estimate of the prediction performance*** improvement by evaluating predictions on those observations that were not used in the building of the next base learner.

When **bootstrap aggregating** is performed, two independent sets are created. One set, the bootstrap sample, is the data chosen to be "in-the-bag" by sampling with replacement. The out-of-bag set is all data not chosen in the sampling process.



Feature Importance

- Feature importance refers to the mechanism of ability of the model to assign a score to the predictor features that indicates how important they are to the model in predicting the target.

- Feature importance is important to the business to know where in the business to focus on to make improvements to achieve goals. E.g. It is important for a bank to know that age and education levels are large influencers of Credit Default. They would then factor these features in Modelling interest rates for certain risky age/education ranges.

- The '***feature_importances_***' attribute of the ***RandomForestClassifier model*** holds the data on Feature Importance and we will see a visual representation in the practical section.

Project

Python Demo

We will now see a Classification problem being solved using a Random Forest model. We have an input dataset containing ***Credit history of individuals with a class attribute name 'Defaulted'*** which contains one of two possible values – 1 (Defaulted) and 0 (not Defaulted). The input dataset contains two types of data – **Demographic** (Age, Sex, Marriage, etc.) and **Behavioral** data related to the Credit (past loans, payment, number of times a credit payment has been delayed by the customer etc.).

We will use a **Random Forest Algorithm** to train the model and then use it to predict on test data set aside from the Input dataset that we have. We will also run a Cross Validation with Hyperparameter Tuning to see what optimum values of the Hyperparameters produce the best results and how to arrive at them.

Hope you have liked this Video.
Please help us by providing your Ratings and Comments for this Course!

Thank You!!
Manas Dasgupta

Happy Learning!!