**Sheza Nauman**

**Kavskills**

**Assignment**

**Random forest in ML**

**And**

**Parametric and non-parametric algorithm**

**Random forest in ML:**

Leo Breiman and Adele Cutler created the popular machine learning algorithm Random Forest, which combines the output of various decision trees to produce a single outcome. Its widespread use is motivated by its adaptability and usability because it can solve classification and regression issues.

**Working of random forest algorithm:**

Before understanding the working of the random forest algorithm in machine learning, we must look into the ensemble learning technique. **Ensemble**simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.

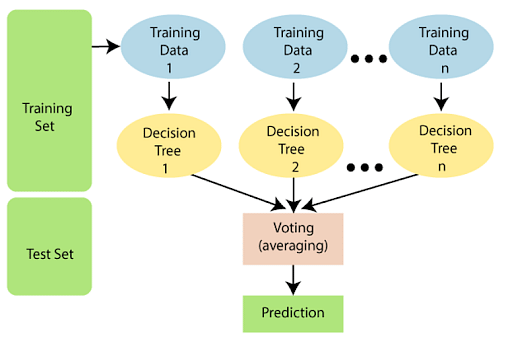
Ensemble uses two types of methods:

**Bagging:**

It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

**Boosting**

It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.



**The following steps explain the working Random Forest Algorithm:**

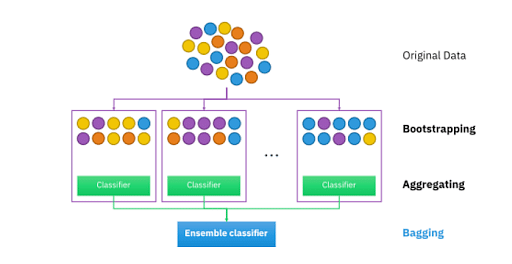
Step 1: Select random samples from a given data or training set.

Step 2: This algorithm will construct a decision tree for every training data.

Step 3: Voting will take place by averaging the decision tree.

Step 4: Finally, select the most voted prediction result as the final prediction result.

Bagging: From the principle mentioned above, we can understand Random forest uses the Bagging code. Now, let us understand this concept in detail. Bagging is also known as Bootstrap Aggregation used by random forest. The process begins with any original random data. After arranging, it is organized into samples known as Bootstrap Sample. This process is known as Bootstrapping. Further, the models are trained individually, yielding different results known as Aggregation. In the last step, all the results are combined, and the generated output is based on majority voting. This step is known as Bagging and is done using an Ensemble Classifier.





## **Essential Features of Random Forest**

* Miscellany: Each tree has a unique attribute, variety and features concerning other trees. Not all trees are the same.
* Immune to the curse of dimensionality: Since a tree is a conceptual idea, it requires no features to be considered. Hence, the feature space is reduced.
* Parallelization: We can fully use the CPU to build random forests since each tree is created autonomously from different data and features.
* Train-Test split: In a Random Forest, we don’t have to differentiate the data for train and test because the decision tree never sees 30% of the data.
* Stability: The final result is based on Bagging, meaning the result is based on majority voting or average.

**Difference between Decision Tree and Random Forest**

|  |  |
| --- | --- |
| **Decision Trees** | **Random Forest** |
| * They usually suffer from the problem of overfitting if it’s allowed to grow without any control. | * Since they are created from subsets of data and the final output is based on average or majority ranking, the problem of overfitting doesn’t happen here. |
| * A single decision tree is comparatively faster in computation. | * It is slower. |
| * They use a particular set of rules when a data set with features are taken as input. | * Random Forest randomly selects observations, builds a decision tree and then the result is obtained based on majority voting. No formulas are required here. |

## **Why Use a Random Forest Algorithm?**

There are a lot of benefits to using Random Forest Algorithm, but one of the main advantages is that it reduces the risk of overfitting and the required training time. Additionally, it offers a high level of accuracy. Random Forest algorithm runs efficiently in large databases and produces highly accurate predictions by estimating missing data.

## **Hyper-parameters in Random forest:**

Hyper-parameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

The following hyper-parameters are used to enhance the predictive power:

* n\_estimators: Number of trees built by the algorithm before averaging the products.
* max\_features: Maximum number of features random forest uses before considering splitting a node.
* mini\_sample\_leaf: Determines the minimum number of leaves required to split an internal node.

The following hyper-parameters are used to increase the speed of the model:

* n\_jobs: Conveys to the engine how many processors are allowed to use. If the value is 1, it can use only one processor, but if the value is -1,, there is no limit.
* random\_state: Controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyper-parameters and the same training data.
* oob\_score: OOB (Out Of the Bag) is a random forest cross-validation method. In this, one-third of the sample is not used to train the data but to evaluate its performance.

**Parametric and non-parametric algorithm**

Parametric and non-parametric algorithms are two different approaches in the field of machine learning and statistics. Let's dive into a concise yet thorough explanation of both:

**Parametric Algorithms**:

Parametric algorithms make specific assumptions about the functional form of the data distribution. These assumptions simplify the problem by reducing it to a finite number of parameters. Here are some key points:

**Assumptions:**

Parametric algorithms assume that the data follows a particular distribution, typically normal (Gaussian) distribution. For example, linear regression assumes a linear relationship between variables.

**Fixed Model**:

A parametric model has a fixed number of parameters that doesn't grow with the size of the dataset. These parameters are usually learned from the training data.

**Efficiency:**

Because of their assumptions and fixed model structure, parametric algorithms are often computationally efficient, making them suitable for large datasets.

**Examples:**

Linear Regression, Logistic Regression, Naive Bayes are common parametric algorithms.

**Non-Parametric Algorithms**:

Non-parametric algorithms, in contrast, do not make strong assumptions about the data distribution. They are more flexible and can adapt to complex, arbitrary data patterns. Here's a brief overview:

**No Assumptions**:

Non-parametric algorithms don't assume a fixed form of the data distribution. They let the data speak for itself, which makes them versatile.

**Variable Model Size**:

These models often have a variable number of parameters that can grow with the size of the dataset. They are data-driven and can capture intricate patterns.

**Flexibility:**

Non-parametric algorithms are particularly useful when dealing with complex, non-linear relationships, and when you can't make strong assumptions about the data.

**Examples:**

K-Nearest Neighbors (KNN), Decision Trees, Support Vector Machines (SVM), and Random Forests can be considered non-parametric.

**In summary**, parametric algorithms are like well-defined templates with specific assumptions, which can be computationally efficient but may not always capture the true underlying data distribution. Non-parametric algorithms are more flexible, accommodating a broader range of data patterns, but they can be computationally intensive and might require more data to generalize effectively.