GLA UNIVERSITY MATHURA

ML ASSIGNMENT

NAME : SRISHTI GUPTA

SEC- D

1. TECH CS 3 YEAR

CLASS ROLL NO. 68

UNIV. ROLL NO. 2215001777

TOPIC- DECISION TREE AND RANDOM FOREST

### What is Random Forest?

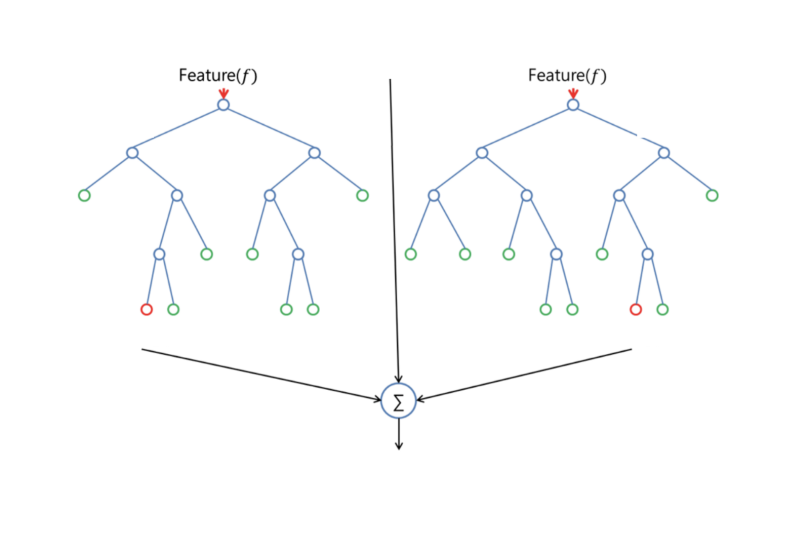
At its core, **Random Forest** is a machine learning algorithm that helps make predictions by combining multiple decision trees. Think of it like a group of experts (the trees) each giving their own opinion on a problem. The more experts you have, the more reliable your final decision is likely to be. By combining the wisdom of many decision trees, Random Forest makes the prediction process more accurate and stable than relying on just one tree.

### How Does Random Forest Work?

The beauty of **Random Forest** is in its simplicity and versatility. It can be used for both **classification** (where the goal is to sort things into categories) and **regression** (where you predict continuous values, like house prices or stock prices).

In **classification**, for instance, each decision tree in the forest makes a prediction (e.g., whether an email is spam or not), and then the final prediction is decided by taking the majority vote from all the trees. For **regression**, the trees "vote" by averaging their results.

Each tree is trained on a **random subset** of the data and uses a **random subset** of features to split the data at each step. This added randomness helps ensure that the trees are diverse, which ultimately leads to a more robust model.



### **Random Forest in Classification and Regression**

Random forest has nearly the same hyper-parameters as a decision tree or a bagging classifier. Fortunately, there’s no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also [deal with regression tasks](https://builtin.com/data-science/random-forest-python" \t "https://builtin.com/data-science/_blank) by using the algorithm’s regression.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting [a node](https://builtin.com/software-engineering-perspectives/tree-traversal" \t "https://builtin.com/data-science/_blank), it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in a random forest classifier, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

### Random Forest vs. Decision Trees

Now, imagine using just one decision tree to make a prediction. The tree looks at the data, splits it based on the features (like age, salary, etc.), and continues making splits until it reaches a decision. This can work well, but if the tree grows too deep and complex, it may start to memorize the training data instead of learning general patterns—a problem called over-fitting.

In contrast, **Random Forest** is like building a team of decision trees, each one trained on different data and with a slightly different perspective. Instead of relying on one tree that could over-fit, Random Forest takes the average prediction from all the trees, which helps prevent over-fitting and usually leads to better performance. It's like getting multiple opinions on a decision and choosing the most common answer.

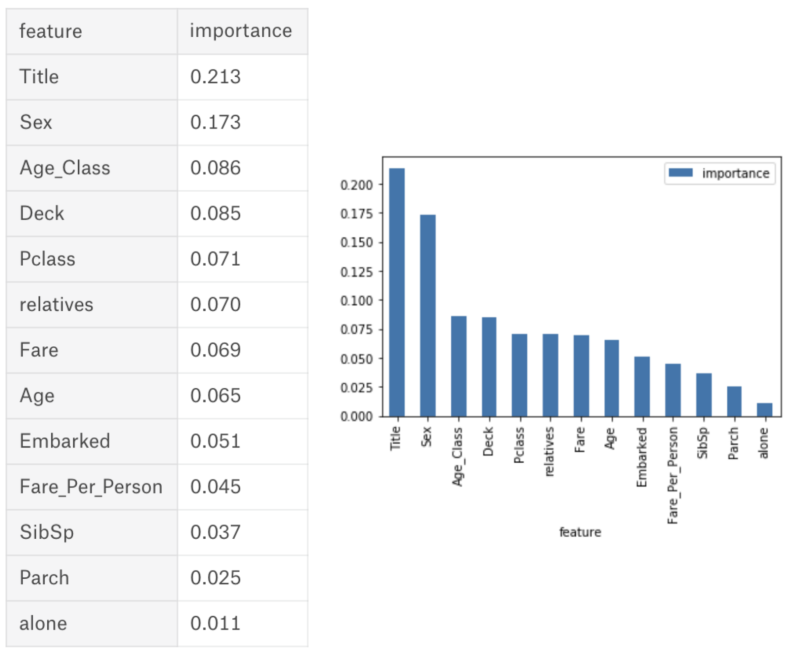
## **Random Forest Feature Importance**

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. [Sklearn](https://builtin.com/machine-learning/scikit-learn-guide" \t "https://builtin.com/data-science/_blank) provides a great tool for this that measures a [feature’s importance](https://builtin.com/data-science/feature-importance" \t "https://builtin.com/data-science/_blank) by looking at how much the tree nodes that use that feature reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results so the sum of all importance is equal to one.

If you don’t know how a decision tree works or what a leaf or node is, here is a good description from Wikipedia: “In a decision tree, each internal node represents a ‘test’ on an attribute (e.g., whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). A node that has no children is a leaf.”

By looking at the feature importance you can decide which features to possibly drop because they don’t contribute enough (or sometimes nothing at all) to the prediction process. This is important because a general rule in machine learning is that the more features you have the more likely your model will suffer [from overfitting](https://builtin.com/data-science/model-fit" \t "https://builtin.com/data-science/_blank) and vice versa.

Below is a table and visualization showing the importance of 13 features, which I used during a supervised classification project with the famous Titanic-data set on Kaggle.



### What is a Decision Tree Classification Algorithm?

A **Decision Tree** is a powerful **supervised learning** technique used for both **classification** and **regression** tasks, although it is especially popular for **classification** problems. It works by organizing data in a tree-like structure, where each decision is based on a feature of the data set, and the branches represent decision rules that guide the data towards a final outcome.

In simpler terms, a decision tree is like a flowchart or a set of "if-then" rules. It starts at a "root node" with the entire data set and makes decisions by asking questions about the data. These questions split the data into smaller subsets, which continue splitting until the tree reaches "leaf nodes," which represent the final prediction or decision.



### Why Use Decision Trees?

**Decision Trees** are favored for several reasons:

* **Intuitive and Easy to Understand**: A decision tree mimics human decision-making, making it easy for non-technical users to interpret. The tree structure clearly shows how decisions are made, step by step.
* **No Need for Data Scaling**: Decision trees don’t require feature scaling or normalization of data, which simplifies data prepossessing.
* **Handles Both Categorical and Numerical Data**: Decision trees can work with both types of data, making them versatile for different kinds of datasets.

## Decision Tree Terminologies

· ****Root Node:**** Root node is from where the decision tree starts. It represents the entire data set, which further gets divided into two or more homogeneous sets.

· ****Leaf Node:**** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

· ****Splitting:**** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

· ****Branch/Sub Tree:**** A tree formed by splitting the tree.

· ****Pruning:**** Pruning is the process of removing the unwanted branches from the tree.

· ****Parent/Child node:**** The root node of the tree is called the parent node, and other nodes are called the child nodes.

****Example:****

Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



## Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

* ****Information Gain****
* ****Gini Index****