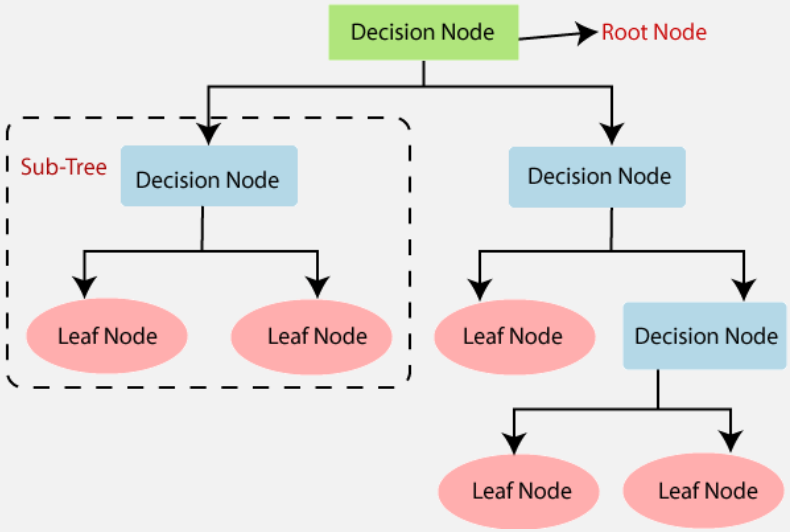
Python for Beginners

Archer Infotech , PUNE

Python – Decision Tree

Decision Tree

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**



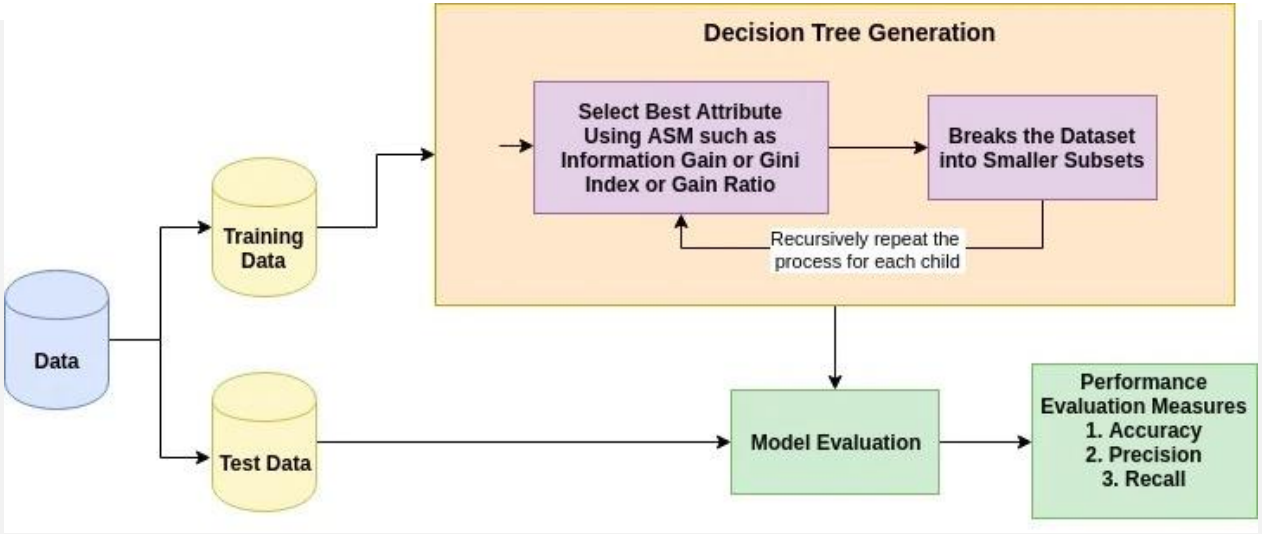
Types of Decision Tree

- **Regression Tree** : A regression tree is used when the dependent variable is **continuous**. The value obtained by leaf nodes in the training data is the **mean** response of observation falling in that region. Thus, if an unseen data observation falls in that region, its prediction is made with the mean value. This means that even if the dependent variable in training data was continuous, it will only take discrete values in the test set. A regression tree follows a **top-down greedy approach.**

- **Classification Tree** A classification tree is used when the dependent variable is **categorical**. The value obtained by leaf nodes in the training data is the **mode** response of observation falling in that region It follows a **top-down greedy approach.**

- Together they are called as **CART(classification and regression tree)**

How does Decision Tree Works ?



- Select the best attribute using Attribute Selection Measures(ASM) to split the records.
- Make that attribute a decision node and breaks the dataset into smaller subsets.
- Starts tree building by repeating this process recursively for each child until one of the condition will match:
  - All the tuples belong to the same attribute value.
  - There are no more remaining attributes.
  - There are no more instances.

Splitting Measures

Information Gain

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute

Information Gain = Entropy before splitting - Entropy after splitting

Entropy:

Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

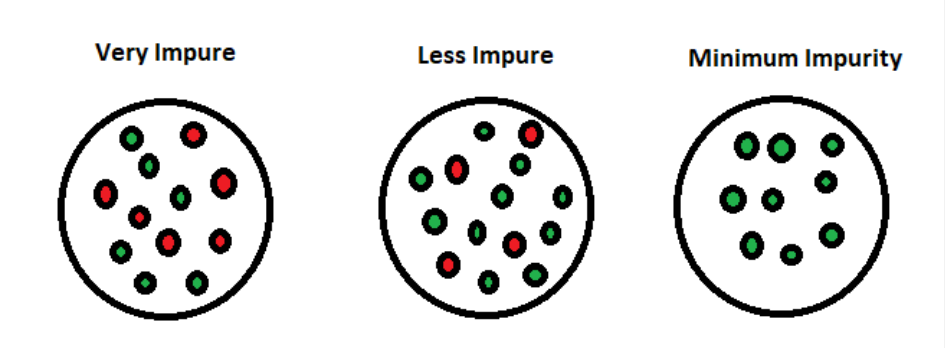
Entropy(s) = -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples
P(yes)= probability of yes
P(no)= probability of no

Splitting Measures - Entropy

Entropy is an information theory metric that measures the impurity or uncertainty in a group of observations.



The entropy may be calculated using the formula below:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

pi is the probability of randomly selecting an example in class i

Splitting Measures - Entropy

e.g. dataset 1 red 3 purple 4 yellow

$E = -(pr \log_2 pr + pp \log_2 pp + py \log_2 py)$

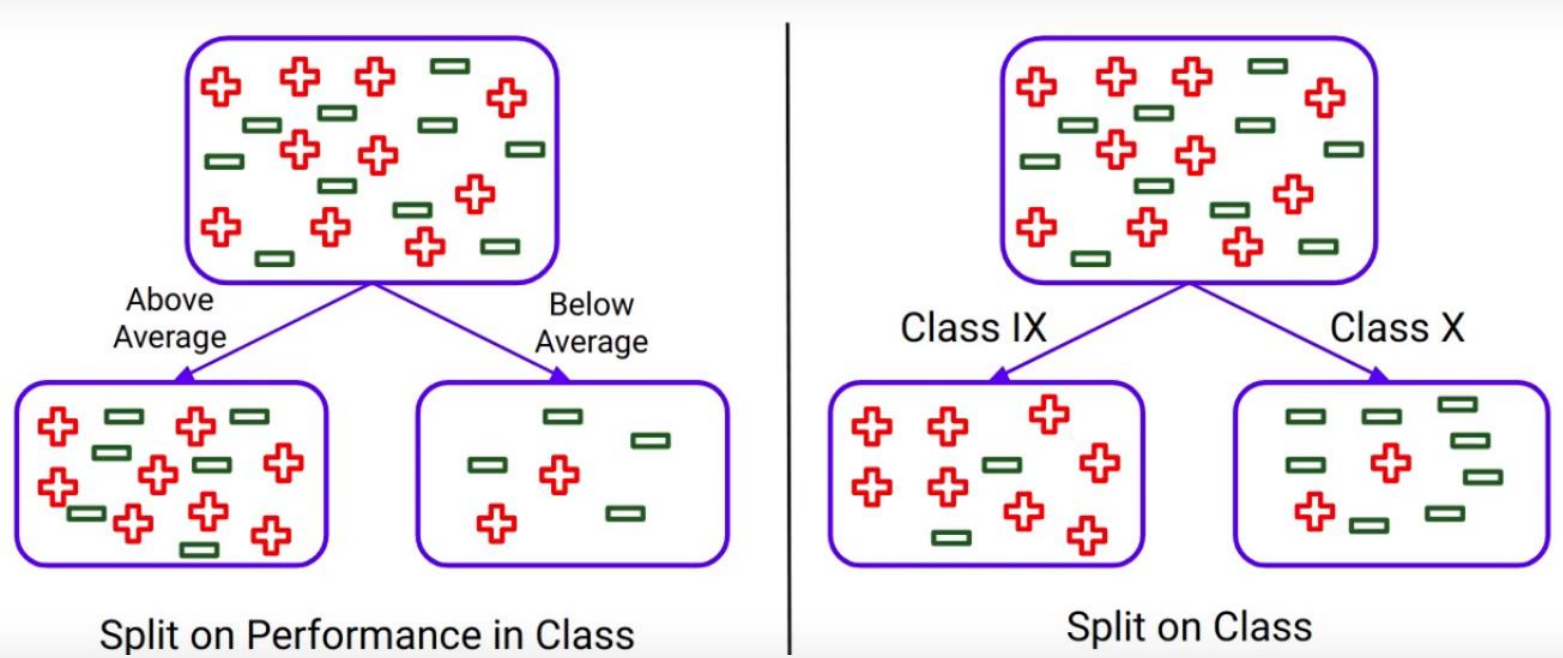
$E = -(1/8 \log_2 (1/8) + 3/8 \log_2(3/8) + 4/8 \log_2 (4/8))$

$E = 1.41$

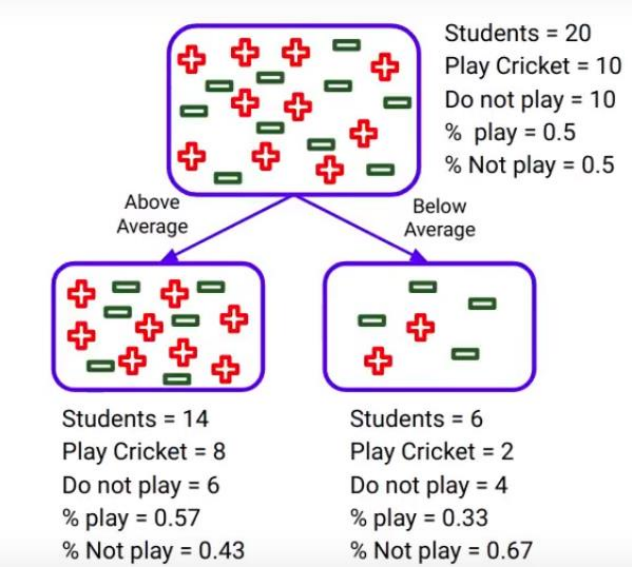
Entropy is measured between 0 and 1

Information Gain = 1 - Entropy

Splitting Measures - Entropy



Splitting Measures - Entropy



Entropy for Parent node:

$-(0.5)*\log_2(0.5) -(0.5)*\log_2(0.5) = 1$

Entropy for sub-node Above Average:

$-(0.57)*\log_2(0.57) -(0.43)*\log_2(0.43) = 0.98$

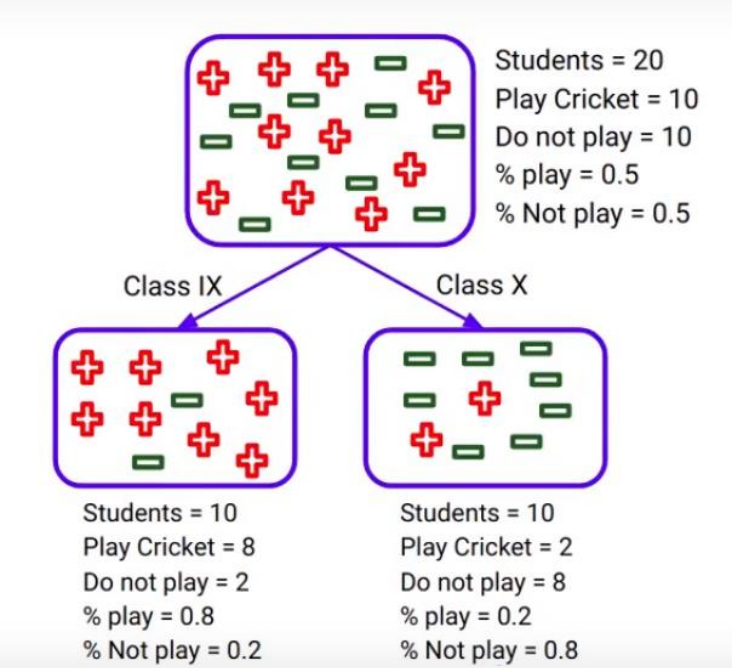
Entropy for sub-node Below Average:

$-(0.33)*\log_2(0.33) -(0.67)*\log_2(0.67) = 0.91$

Weighted Entropy: Performance in Class:

$(14/20)*0.98 + (6/20)*0.91 = 0.959$

Splitting Measures - Entropy



Split on Class

Entropy for Parent node:

$-(0.5)*\log_2(0,5) -(0.5)*\log_2(0.5) = 1$

Entropy for sub-node Class IX:

$-(0.8)*\log_2(0.8) -(0.2)*\log_2(0.2) = 0.722$

Entropy for sub-node Class X:

$-(0.2)*\log_2(0.2) -(0.8)*\log_2(0.8) = 0.722$

Weighted Entropy: Class:

$(10/20)*0.722 + (10/20)*0.722 = 0.722$

Splitting Measures - Entropy

Information Gain = 1 - Entropy

Split	Entropy	Information Gain
Performance in Class	0.959	0.041
Class	0.722	0.278

For “the Performance in class” variable information gain is **0.041** and for “the Class” variable it’s **0.278**. Lesser entropy or higher Information Gain leads to more homogeneity or the purity of the node. And hence split on the Class variable will produce more pure nodes.

Splitting Measures

Gini index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen

If all the elements belong to a single class, then it can be called pure.

The degree of Gini index varies between 0 and 1,

where, 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes.

A Gini Index of 0.5 denotes equally distributed elements into some classes.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

where p_i is the probability of an object being classified to a particular class

Example of Gini Index

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Example of Gini Index

Calculating the Gini Index for Past Trend

P(Past Trend=Positive): 6/10
P(Past Trend=Negative): 4/10

If (Past Trend = Positive & Return = Up), probability = 4/6
If (Past Trend = Positive & Return = Down), probability = 2/6

Gini index = 1 - ((4/6)^2 + (2/6)^2) = 0.45

If (Past Trend = Negative & Return = Up), probability = 0
If (Past Trend = Negative & Return = Down), probability = 4/4

Gini index = 1 - ((0)^2 + (4/4)^2) = 0

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index for Past Trend = (6/10)*0.45 + (4/10)*0 = 0.27

Example of Gini Index

Attributes/Features	Gini Index
Past Trend	0.27
Open Interest	0.47
Trading Volume	0.34

from the above table, we observe that ‘Past Trend’ has the lowest Gini Index and hence it will be chosen as the root node for how decision tree works.

Python – Random Forest

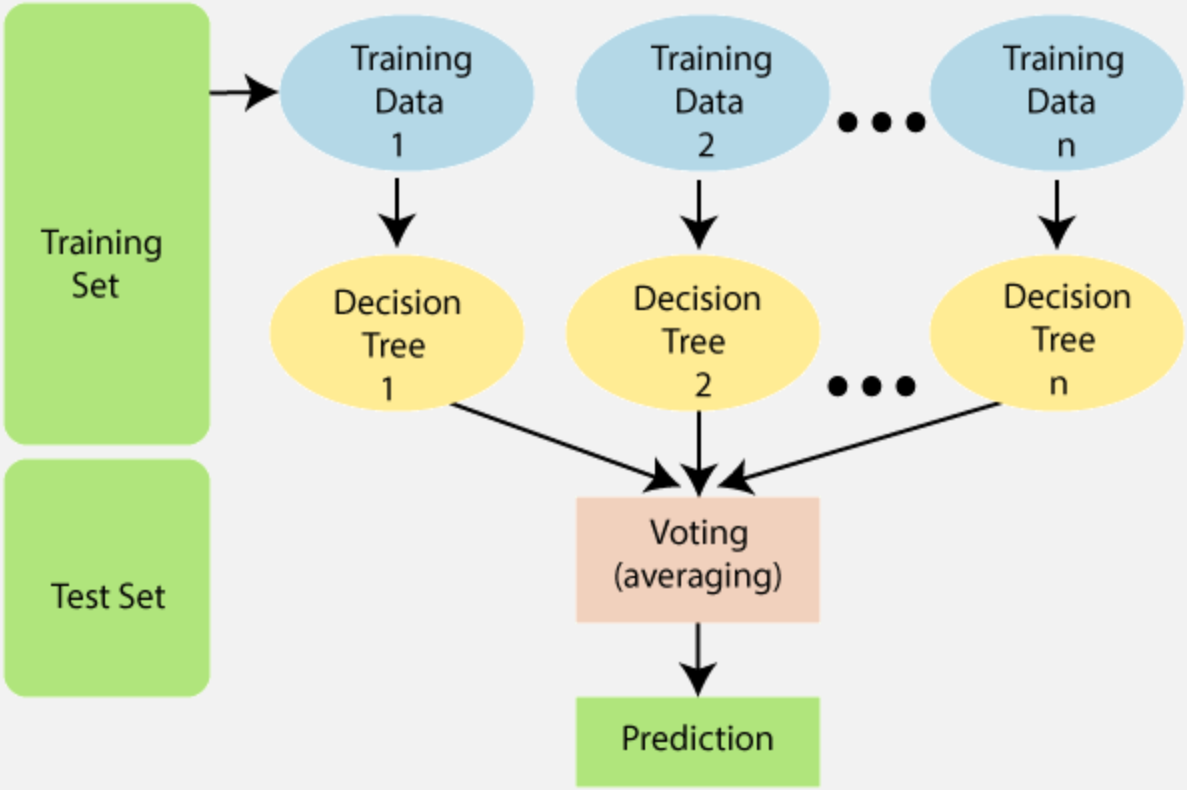
Random Forest

- **Random Forest** is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML.

- It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance*

- ***Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset*** *of the model.*

Random Forest



How Random Forest Works ?

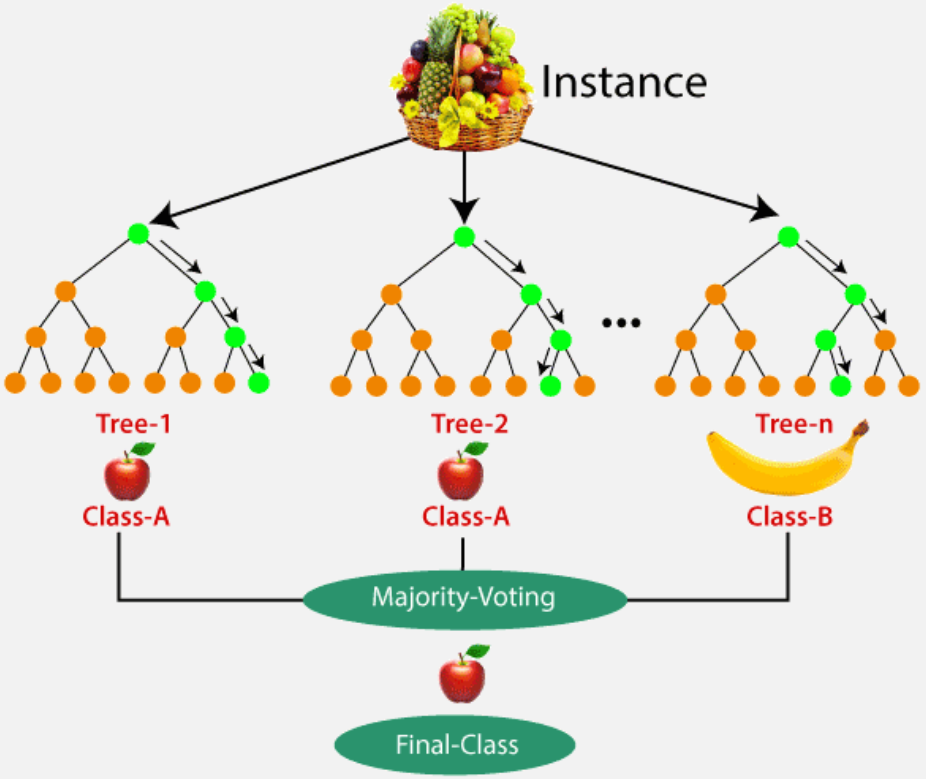
Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.



Important Parameters

Following hyperparameters increases the predictive power:

1. **n_estimators**– number of trees the algorithm builds before averaging the predictions.

2. ***max_features***– maximum number of features random forest considers splitting a node.

3. ***mini_sample_leaf***– determines the minimum number of leaves required to split an internal node.

Difference Between Decision Tree & Random Forest

Decision trees	Random Forest
1. Decision trees normally suffer from the problem of overfitting if it's allowed to grow without any control.	1. Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2. A single decision tree is faster in computation.	2. It is comparatively slower.
3. When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction.	3. Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn't use any set of formulas.

THANK YOU !!!

Amol Patil - 9822291613