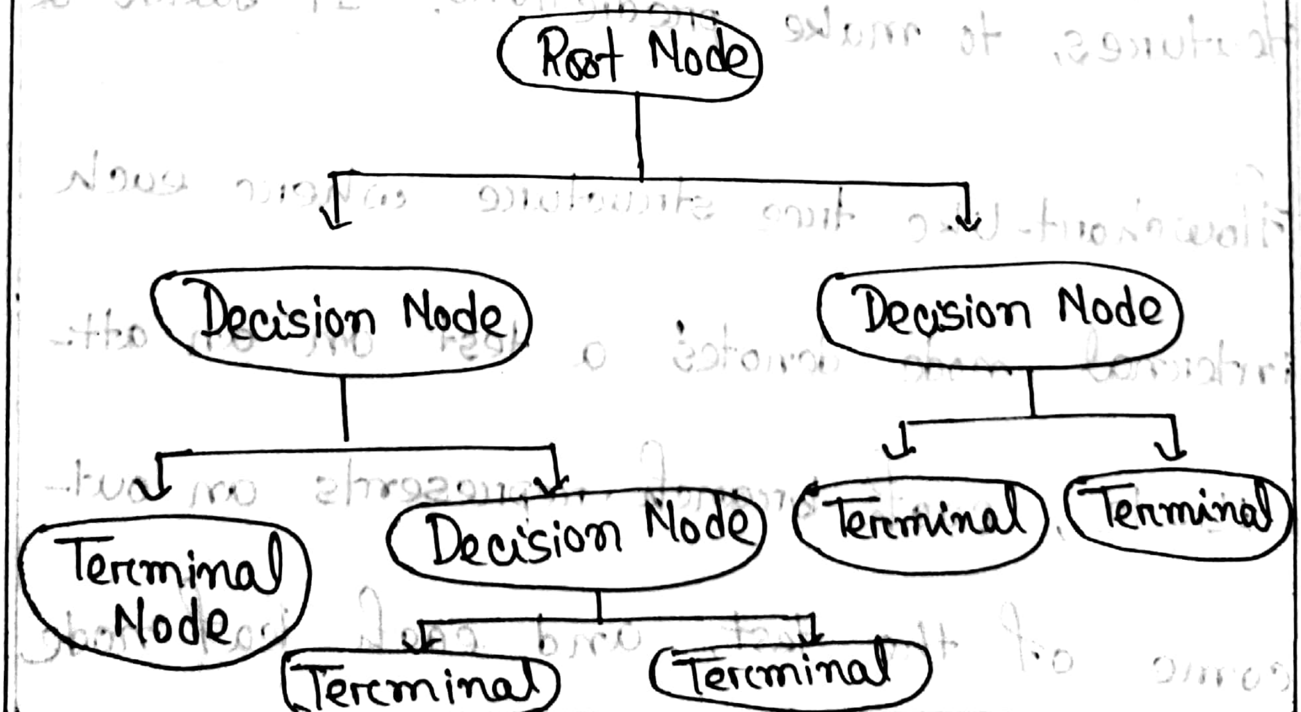
Assignment on Decision Tree Algorithm

Md. Masud Mazumder
19701070

Introduction: Decision tree is a versatile and powerful machine learning algorithm. The algorithm is used for both classification and regression problems. Decision trees mimic the human decision-making process by recursively partitioning the feature space into regions, based on the values of input features, to make predictions. It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class level.

During training, the Decision Tree algorithm selects the best attribute to split the data based on a matric such as entropy or Gini impurity, which measures the level of impurity or randomness in the subsets.



Entropy and Information Gain: Entropy is a measure of impurity in a dataset. In the context of Decision Tree, entropy is used to quantify the uncertainty or randomness in the distribution of class lebels within a node. Mathematically, the entropy of a dataset D with respect to class C is defined as:

$$H(D) = - \sum_{i=1}^{c} p_i \log_2 (p_i)$$

where p_i is the portion of samples in class i in dataset D, and c is the number of classes.

Information gain is a measure of the effectiveness of a particulare attribute in classifying the data. It quantifyes the reduction in entrcopy achiveved by splitting the dataset on a specific attribute.

The attribute with the highest information gain is choosen as the spilitting criterion. Mathematically, the information gain of an attribute A on a dataset D is given by:

$$IG(D, A) = H(D) - \sum_{v \in Values(A)} \frac{D_v}{D} \{H(D_v)$$

where Values(A) is the set of possible values of attribute A.

CART (Classification and Regression Tree):

CART is a variation of the decision tree algorithm. It can handle both classification and regression tasks. Unlike ID3, which only handles categorical attributes, CART can handle both categorical and continuous attributes. CART recursively partitions the feature space by selecting the attribute and split values that minimizes a splitting criterion, such as Gini impurity for classification or mean square error for regression.

Step by step CART:

Step 1: Begin the tree with the root node, say S, which contains the complete dataset.

Step 2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step 3: Divide the S into subsets that contains possible values for the best attributes.

Step 4: Generate the decision tree node, which contains the best attributes.

Step 5: Recursively make new decision trees using the subsets of the dataset created in step-3. Continue this process until a stage is reached where you cannot find further classify the nodes and called the final node as a leaf node.

Iterative Dichotomiser 3 (ID3):

The ID3 algorithm is a popular decision tree learning algorithm that constructs Decision Trees using a top-down, greedy approach. It recursively partitions the feature space by selecting the attribute that maximizes information gain at each node. The ~~goa~~ goal is to make the final subsets as homogeneous as possible.

ID3 algorithm:

Step 1: Check if all the samples belong to the same class.

↳ If all the samples belong to the same class, create a leaf node with that class level and return.

Step 2: If the dataset is empty, return the most common class lebel in the parent node.

Step 3: Compute entropy $H(D)$ and information gain $IG(D, A)$.

Step 4: Choose the attribute with the highest information gain as the splitting attribute for the current node.

Step 5: Split the dataset D into subsets based on the values of the selected attributes.

Step 6: Recursively apply steps 1-5 to each subset.

Step 7: Return the constructed decision tree.

Prcuning:

While ID3 constrcucts decision trcees by by maximizing information gain, it may rcesult in overcfitting, specially forc noisy data orc datasets with many att-rcibutes. Prcuning techniques, such as rceducing the trcee depth orc setting a minimum numberc of samples perc leaf node, can be applied to prcevent overc-fitting and improve the generalization of the model.

Mathematical Example:

Suppose we have a dataset with binary features X_1 and X_2, and a binary target variable Y indicating wheather a person buys a computer (1) or not (0).

Example	X_1	X_2	Y
1	0	0	0
2	1	0	0
3	1	1	1
4	0	1	1
5	0	1	1

Step 1: Compute entropy $H(Y)$:

$$H(Y) = -P(0)\log_2(P(0)) - P(1)\,H(P(1))$$

Here,

$$P(0) = \frac{2}{5} \quad \text{and} \quad P(1) = \frac{3}{5}$$

$$\therefore H(Y) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)$$

$$\approx 0.971$$

Step 2: Compute information gain for each attribute:

$$IG(Y, X) = H(Y) - \sum_{v \in Values(X)} \frac{|D_v|}{|D|} H(Y_v)$$

For X_1:

↳ $Values(X_1) = \{0, 1\}$

↳ $D_0 = \{1, 5\}$, $D_1 = \{2, 3, 4\}$

$H(Y_0) = 0$ (Pure class)

$$H(Y_1) = -\frac{1}{3}\log_2(1/3) - \frac{2}{3}\log_2\left(\frac{2}{3}\right)$$

$$\therefore IG(Y, X_1) = 0.971 - \left(\frac{2}{5}\times 0 + \frac{3}{5}\times H(Y_1)\right)$$

$$\approx 0.971 - \left(0 + \frac{3}{5}\times 0.918\right)$$

$$\approx 0.42$$

For X_2:

↳ Values $(X_2) = \{0, 1\}$

↳ $D_0 = \{1, 2\}$, $D_1 = \{3, 4, 5\}$

$$H(Y_0) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right)$$

$H(Y_1) = 0$ (Pure class)

$$\therefore IG(Y, X_2) = 0.971 - \left(\frac{2}{5}\times H(Y_0) + \frac{3}{5}\times 0\right)$$

$$= 0.971 - \left(\frac{2}{5}\times 1\right)$$

$$\approx 0.571$$

Step 3: Select attribute with highest information gain:

Since $IG(Y, X_2) > IG(Y, X_1)$, we choose X_2 as the splitting attribute.

Step 4: Split the dataset based on selected attribute:

We split the dataset into two subsets based on the values of attribute X_2:

↳ Subset D_0 (where $X_2 = 0$): instances 1,2

↳ Subset D_1 (where $X_2 = 1$): instances 3,4,5

Step 5: Recursively apply step 1-4:

We recursively apply the ID3 algorithm to each subset.

↳ For Subset D_0 (where $x_2=0$):

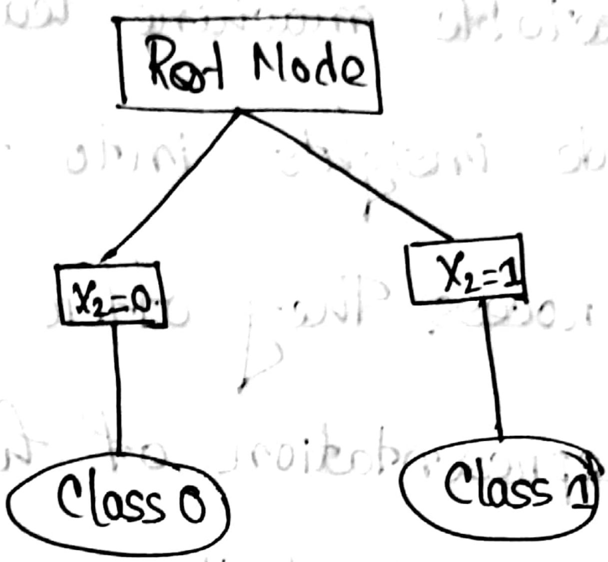
Since all the instances belong to the same class (0), we create a leaf node with label 0.

↳ For subset D_1 (where $x_2=1$):

Since all the instances belong to same class (1), we create a leaf node with label 1.

Step 6: Return the decision tree.

The constructed decision tree for the example is as follows:



Conclusion:

In conclusion, Decision Trees are powerful and interpretable machine learning models that provide insights into the decision making process. They offer a clear and intuitive representation of how input features contribute to the prediction of the target variables.

Overall, decision trees serve as valuable tools in the machine learning toolkit, offering a balance between accuracy and interpretability.