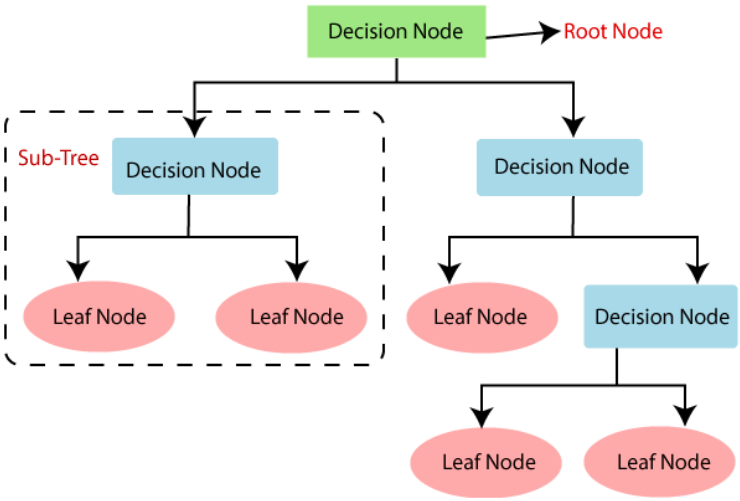
WEEK: 09 DECISION TREE [C4.5 AND CART]

- 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.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

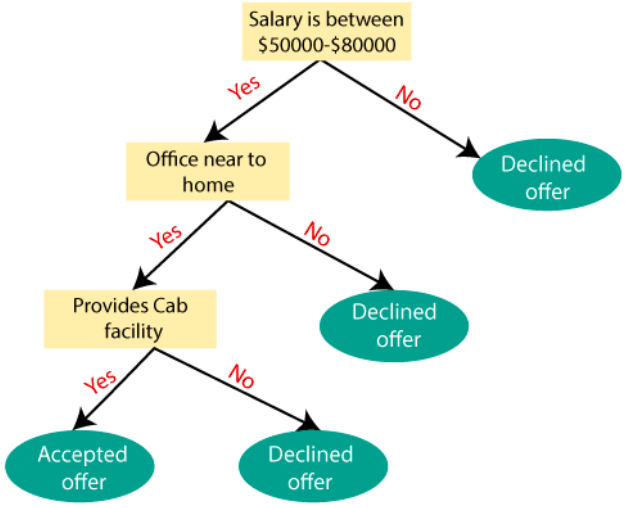
- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** 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.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says 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 attribute.
- **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 further classify the nodes and called the final node as a leaf node.

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**

***1. Information Gain*:**

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

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

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= 1- $\sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

ID3, C4.5, AND CART DECISION TREES

1. ID3 stands for Iterative Dichotomizer3 and is named such because the algorithm iteratively(repeatedly) dichotomizes(divides) features into two or more groups at each step.ID3 is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset and is the most popular algorithms used to constructing trees.

ID3 is the core algorithm for building a decision tree .It employs a top-down greedy search through the space of all possible branches with no backtracking. This algorithm uses information gain and entropy to construct a classification decision tree.

Characteristics of ID3 Algorithm

Major Characteristics of the ID3 Algorithm are listed below:

- ID3 can overfit the training data (to avoid overfitting, smaller decision trees should be preferred over larger ones).
- This algorithm usually produces small trees, but it does not always produce the smallest possible tree.
- ID3 is harder to use on continuous data (if the values of any given attribute is continuous, then there are many more places to split the data on this attribute, and searching for the best value to split by can be time-consuming).

Steps to making Decision Tree (ID3)

a) Take the Entire dataset as an input.
b) Calculate the Entropy of the target variable, As well as the predictor attributes
c) Calculate the information gain of all attributes.
d) Choose the attribute with the highest information gain as the Root Node
e) Repeat the same procedure on every branch until the decision node of each branch is finalized.

Entropy-It is used for checking the impurity or uncertainty present in the data. Entropy is used to evaluate the quality of a split. When entropy is zero the sample is completely homogeneous, meaning that each instance belongs to the same class and entropy is one when the sample is equally divided between different classes.

Formula of Entropy -

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

Information Gain- Information gain indicates how much information a particular feature/ variable give us about the final outcome.

Formula of information gain-

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

2. C4.5

C4.5 Decision Tree is a complicated Algorithm to understand. It does require a lot of background knowledge.

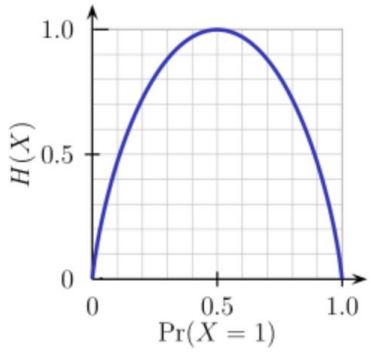
Information Entropy

Information Entropy is the measure of impurity in a given example. Let's elaborate

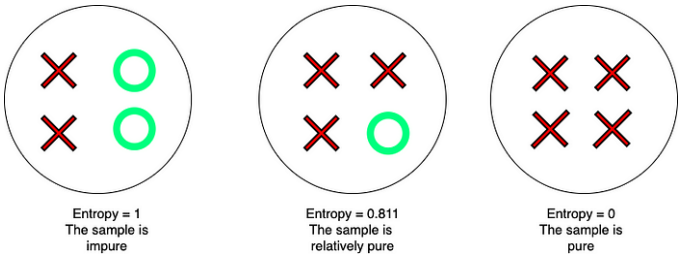
Shanon entropy or self information:

$$-\Sigma_{i=1}^{n} P(x_i) \log_b(P(x_i))$$

Where b is the base.
For a binary, the base would be 2 and the plot of the function would be as follows for percentages of 1 in the dataset:



So if the Probability of an event approaches 1 or 0, the Information Entropy tends to 0 as the output is more or less predictable.
In statistics, entropy measures the level of impurity(heterogeneity) in a dataset. A fully homogenous dataset has an entropy of 0 whereas a skewed dataset has an entropy closer to 1 as shown in the figure below:



Measure the impurity of a sample with Entropy

Information Gain

It is a parameter that is used to compute the change in entropy of a dataset before and after a transformation. Information Gain helps in feature selection. How it works is as follows.
A feature in a dataset is chosen and based on the value of the feature, the dataset is split into multiple smaller datasets. The change in the overall entropy from that of the parent to the average entropy of all the new child datasets is computed to be the Information Gain.

$$Information\ Gain = Entropy_{parent} - \sum Entropy_{child}$$

Formula for Information Gain

The higher the Information Gain the more accurately the feature divides the dataset as the resultant dataset are more homogenous with lower entropies. This indicates the feature has split(classified) the dataset accurately. On the other hand, if the Information Gain is low, it implies that the resultant datasets are more or less as heterogeneous as the parent dataset and so the feature does not provide much value.
Steps in algorithm:
· *Check for the above base cases.*
· *For each attribute a, find the normalized information gain ratio from splitting on a.*
· *Let a_best be the attribute with the highest normalized information gain.*
· *Create a decision node that splits on a_best.*
· *Recurse on the sublists obtained by splitting on a_best, and add those nodes as children of node.*

3. CART

CART stands for Classification And Regression Tree. It is a type of decision tree which can be used for both classification and regression tasks based on **non-parametric supervised learning** method. The following represents the algorithm steps. First and foremost, the data is split into training and test set.

- Take a feature K and split the training data set into two **subsets** based on some threshold of the feature, Tk. For example, if we are working through the IRIS data set, take a feature such as petal length, set the threshold as 2.25 cm. The question that would arise is how does the algorithm select K and Tk. The pair of K and Tk is selected in a way that purest **subsets** (weighted by their size) are created. This can also be called as

the cost or loss function. The following cost function is optimized (minimized).

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

$$\text{where } \begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$$

- Once the training set is split into two subsets, the algorithm splits the subsets into another subsets using the same logic.
- The above split continues in a recursive manner until the maximum depth (hyperparameter max_depth) is reached or the algorithm is unable to find the split that further reduces the impurity.
- The following are few other hyperparameters which can be set before the algorithm is run for training. These are used to regularize the model.
- min_samples_split: Minimum number of samples a node must have before it is split
- min_samples_leaf: Minimum number of samples a leaf node must have
- max_leaf_nodes: Maximum number of leaf nodes

CART decision tree is a **greedy algorithm** as it searches for optimum split right at the top most node without considering the possibility of lowest possible impurity several levels down. Note that greedy algorithms can result into a reasonably great solution but may not be optimal.

The following represents the CART cost function for regression. The cost function attempts to minimize the MSE (mean square error) for regression task. Recall that CART cost function for classification attempts to minimize impurity.

$$J(k, t_k) = \frac{m_{\text{left}}}{m} \text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m} \text{MSE}_{\text{right}} \quad \text{where } \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} \left(\hat{y}_{\text{node}} - y^{(i)}\right)^2 \\ \hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}$$

Decision Tree Pros

- Decision trees are easy to interpret and visualize.
- It can easily capture Non-linear patterns.
- It requires fewer data preprocessing from the user, for example, there is no need to normalize columns.
- It can be used for feature engineering such as predicting missing values, suitable for variable selection.
- The decision tree has no assumptions about distribution because of the non-parametric nature of the algorithm.

Decision Tree Cons

- Sensitive to noisy data. It can overfit noisy data.
- The small variation(or variance) in data can result in the different decision tree. This can be reduced by bagging and boosting algorithms.
- Decision trees are biased with imbalance dataset, so it is recommended that balance out the dataset before creating the decision tree.

Questions

Write a program to demonstrate the working of the decision tree based C4.5, and CART algorithms without using scikit-learn library. Use an appropriate data set (following) for building the decision tree and apply this knowledge to classify a new sample.

The dataset has three attributes: Outlook (Sunny, Overcast, Rainy), Temperature, Humidity and Wind (Weak, Strong). The target attribute is Play Tennis (Yes/No).

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Additional Questions

Write a program to demonstrate the working of the decision tree based C4.5, and CART algorithms with using scikit-learn library. Use an appropriate data set (above one) for building the decision tree and apply this knowledge to classify a new sample. Compare with and without using scikit-learn library, also the results.

The dataset has three attributes: Outlook (Sunny, Overcast, Rainy), Temperature, Humidity and Wind (Weak, Strong). The target attribute is Play Tennis (Yes/No).