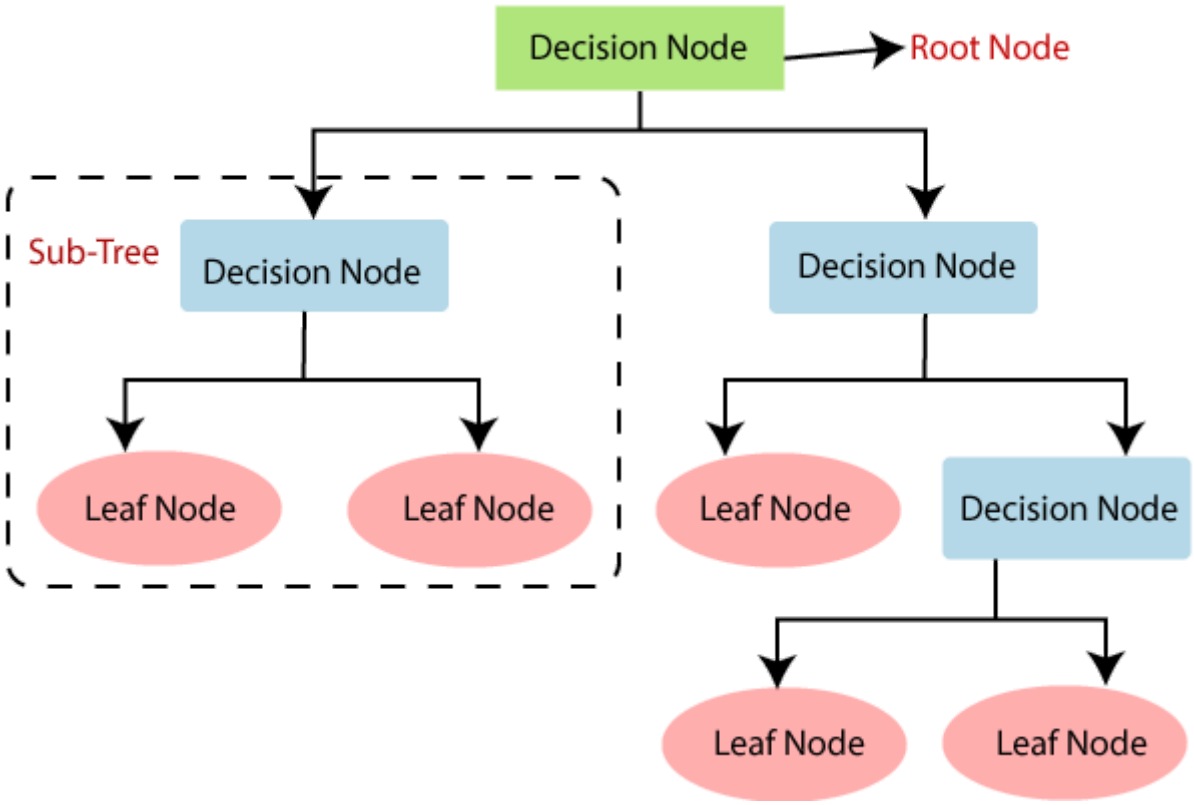
DS M 3 part-1

techworldthink • February 19, 2022

1.Decision Tree Classification Algorithm

- 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.
- A decision tree can contain categorical data (YES/NO) as well as numeric data.



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.

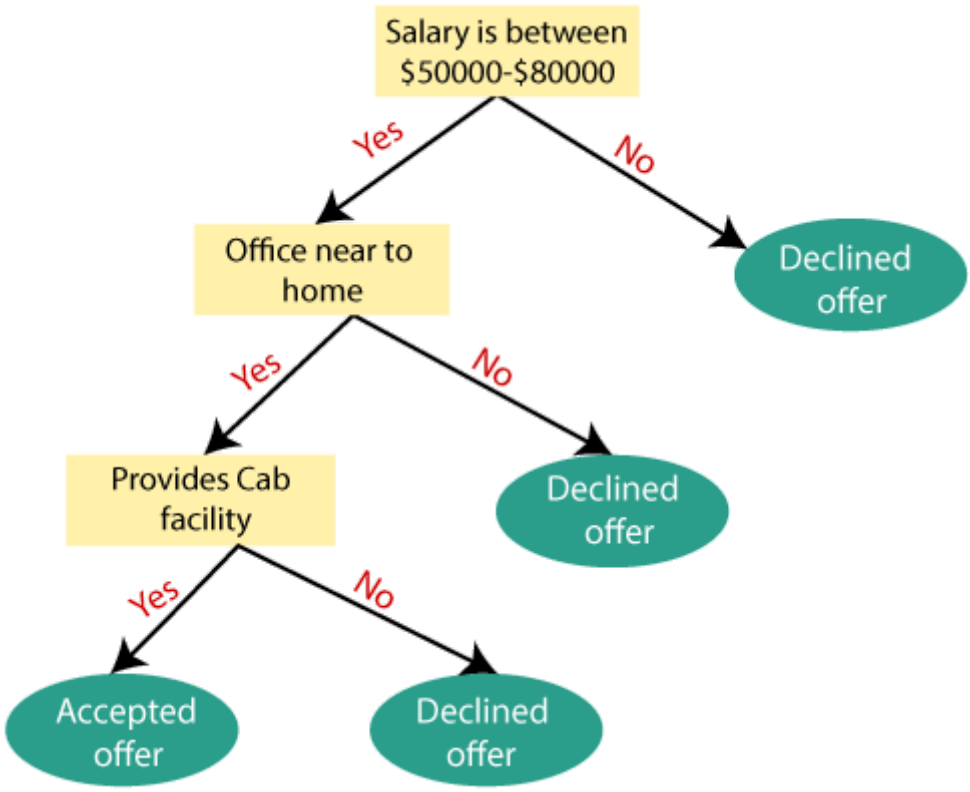
How does the Decision Tree algorithm Work?

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.

eg:



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:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

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.

The process of pruning a decision tree involves reducing its size such that it generalizes better to unseen data

One solution to this problem is to stop the tree from growing once it reaches a certain number of decisions or if the decision nodes contain only a small number of examples. This is called early stopping or pre-pruning the decision tree. As the tree avoids doing needless work, this is an appealing strategy. However, one downside is that there is no way to know whether the tree will miss subtle, but important patterns that it would have learned had it grown to a larger size.

An alternative, called post-pruning involves growing a tree that is too large, then using pruning criteria based on the error rates at the nodes to reduce the size of the tree to a more appropriate level. This is often a more effective approach than pre-pruning because it is quite difficult to determine the optimal depth of a decision tree without growing it first. Pruning the tree later on allows the algorithm to be certain that all important data structures were discovered.

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

(Overfitting: A statistical model is said to be overfitted when we train it with a lot of data (just like fitting ourselves in oversized pants!). When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too many details and noise.)

Divide and conquer

Decision trees are built using a heuristic called recursive partitioning. This approach is generally known as divide and conquer because it uses the feature values to split the data into smaller and smaller subsets of similar classes. Beginning at the root node, which represents the entire dataset, the algorithm chooses a feature that is the most predictive of the target class. The examples are then partitioned into groups of distinct values of this feature; this decision forms the first set of tree branches.

The algorithm continues to divide-and-conquer the nodes, choosing the best candidate feature each time until a stopping criterion is reached. This might occur at a node if:

- All (or nearly all) of the examples at the node have the same class
- There are no remaining features to distinguish among examples
- The tree has grown to a predefined size limit

Gini indices

The Gini split index of a data set is another feature selection measure in the construction of classifi cation trees. This measure is used in the CART algorithm.

8.7.1 Gini index

Consider a data set S having r class labels $c_1, \ldots, c_r$. Let p_i be the proportion of examples having the class label c_i. The Gini index of the data set S, denoted by Gini(S), is defined by

$$\text{Gini}(S) = 1 - \sum_{i=1}^{r} p_i^2.$$

Example

Let S be the data in Table 8.1. There are four class labels "amphi", "bird", "fish", "mammal" and "reptile". The numbers of examples having these class labels are as follows:

Number of examples with class label "amphi"	= 3
Number of examples with class label "bird"	= 2
Number of examples with class label "fish"	= 2
Number of examples with class label "mammal"	= 2
Number of examples with class label "reptile"	= 1
Total number of examples	= 10

The Gini index of S is given by

$$\begin{aligned}\text{Gini}(S) &= 1 - \sum_{i=1}^{r} p_i^2 \\ &= 1 - (3/10)^2 - (2/10)^2 - (2/10)^2 - (2/10)^2 - (1/10)^2 \\ &= 0.78\end{aligned}$$

8.6 Information gain

8.6.1 Definition

Let S be a set of examples, A be a feature (or, an attribute), S_v be the subset of S with $A = v$, and Values (A) be the set of all possible values of A. Then the *information gain of an attribute A relative to the set S*, denoted by Gain (S, A), is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

where $|S|$ denotes the number of elements in S.

8.6 Information gain

8.6.1 Definition

Let S be a set of examples, A be a feature (or, an attribute), S_v be the subset of S with $A = v$, and Values (A) be the set of all possible values of A. Then the *information gain of an attribute A relative to the set S*, denoted by Gain (S, A), is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

where $|S|$ denotes the number of elements in S.

8.6.2 Example 1

Consider the data S given in Table 8.1. We have have already seen that

$$|S| = 10$$
$$\text{Entropy}(S) = 2.2464.$$

We denote the information gain corresponding to the feature "xxx" by Gain (S, xxx).

1. **Computation of Gain $(S$, gives birth$)$**

$$A_1 = \text{gives birth}$$
$$\text{Values of } A_1 = \{\text{"yes"}, \text{"no"}\}$$
$$S_{A_1=\text{yes}} = \text{Data in Table 8.2}$$
$$|S_{A_1=\text{yes}}| = 4$$
$$\text{Entropy}(S_{A_1=\text{yes}}) = 1.5 \quad \text{(See Eq.(8.1))}$$
$$S_{A_1=\text{no}} = \text{Data in Table 8.3}$$
$$|S_{A_1=\text{no}}| = 6$$
$$\text{Entropy}(S_{A_1=\text{no}}) = 1.7925 \quad \text{(See Eq.(8.2))}$$

Now we have

$$\text{Gain}(S, A_1) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A_1)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

$$= \text{Entropy}(S) - \frac{|S_{A_1=\text{yes}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{yes}}) - \frac{|S_{A_1=\text{no}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{no}})$$
$$= 2.2464 - (4/10) \times 1.5 - (6/10) \times 1.7925$$
$$= 0.5709$$

8.5 Entropy

The degree to which a subset of examples contains only a single class is known as *purity*, and any subset composed of only a single class is called a *pure* class. Informally, entropy[3] is a measure of "impurity" in a dataset. Sets with high entropy are very diverse and provide little information about other items that may also belong in the set, as there is no apparent commonality.

Entropy is measured in bits. If there are only two possible classes, entropy values can range from 0 to 1. For n classes, entropy ranges from 0 to $\log_2(n)$. In each case, the minimum value indicates that the sample is completely homogeneous, while the maximum value indicates that the data are as diverse as possible, and no group has even a small plurality.

8.5.1 Definition

Consider a segment S of a dataset having c number of class labels. Let p_i be the proportion of examples in S having the i th class label. The entropy of S is defined as

$$\text{Entropy}\,(S) = \sum_{i=1}^{c} -p_i \log_2(p_i).$$

8.5.2 Examples

Let "xxx" be some class label. We denote by p_{xxx} the proportion of examples with class label "xxx".

1. **Entropy of data in Table 8.1**

   Let S be the data in Table 8.1. The class labels are "amphi", "bird", "fish", "mammal" and "reptile". In S we have the following numbers.

Number of examples with class label "amphi"	= 3
Number of examples with class label "bird"	= 2
Number of examples with class label "fish"	= 2
Number of examples with class label "mammal"	= 2
Number of examples with class label "reptile"	= 1
Total number of examples	= 10

   Therefore, we have:

   $$\text{Entropy}\,(S) = \sum_{\text{for all classes "xxx"}} -p_{\text{xxx}} \log_2(p_{\text{xxx}})$$

[4]Plot created using R language.

$$\begin{aligned}
&= -p_{\text{amphi}} \log_2(p_{\text{amphi}}) - p_{\text{bird}} \log_2(p_{\text{bird}}) \\
&\quad - p_{\text{fish}} \log_2(p_{\text{fish}}) - p_{\text{mammal}} \log_2(p_{\text{mammal}}) \\
&\quad - p_{\text{reptile}} \log_2(p_{\text{reptile}}) \\
&= -(3/10)\log_2(3/10) - (2/10)\log_2(2/10) \\
&\quad - (2/10)\log_2(2/10) - (2/10)\log_2(2/10) \\
&\quad - (1/10)\log_2(1/10) \\
&= 2.2464
\end{aligned}$$

The C5.0 decision tree algorithm

There are numerous implementations of decision trees, but one of the most well known is the C5.0 algorithm.

This algorithm was developed by computer scientist J. Ross Quinlan as an improved version of his prior algorithm, C4.5, which itself is an improvement over his ID3 (Iterative Dichotomiser 3) algorithm.

Although Quinlan markets C5.0 to commercial clients ,

The source code for a single-threaded version of the algorithm was made publically available, and has therefore been incorporated into programs such as R.

The C5.0 algorithm has become the industry standard for producing decision trees, because it does well for most types of problems directly out of the box.

Compared to other advanced machine learning models (such as, Black Box Methods – Neural Networks and Support Vector Machines) the decision trees built by C5.0 generally perform nearly as well but are much easier to understand and deploy.

Additionally the algorithm's weaknesses are relatively minor and can be largely avoided.

- Speed - C5.0 is significantly faster than C4.5.
- Memory usage - C5.0 is more memory efficient than C4.5.
- C5.0 gets similar results to C4.5 with considerably smaller decision trees

Strengths	Weaknesses
• An all-purpose classifier that does well on most problems • Highly-automatic learning process can handle numeric or nominal features, missing data • Uses only the most important features • Can be used on data with relatively few training examples or a very large number • Results in a model that can be interpreted without a mathematical background (for relatively small trees) • More efficient than other complex models	• Decision tree models are often biased toward splits on features having a large number of levels • It is easy to overfit or underfit the model • Can have trouble modeling some relationships due to reliance on axis-parallel splits • Small changes in training data can result in large changes to decision logic • Large trees can be difficult to interpret and the decisions they make may seem counterintuitive

Choosing the best split

The first challenge that a decision tree will face is to identify which feature to split upon. . If the segments of data contain only a single class, they are considered pure. There are many different measurements of purity for identifying splitting criteria.

C5.0 uses entropy for measuring purity. The entropy of a sample of data indicates how mixed the class values are; the minimum value of 0 indicates that the sample is completely homogenous, while 1 indicates the maximum amount of disorder. The definition of entropy is specified by

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