Decision Tree

A decision tree is one of the most powerful tools of supervised learning algorithms used for both classification and regression tasks. 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 label. It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

During training, the Decision Tree algorithm selects the best attribute to split the data based on a metric such as entropy or Gini impurity, which measures the level of impurity or randomness in the subsets. The goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split.

What is a Decision Tree?

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems. It is one of the very powerful algorithms. And it is also used in Random Forest to train on different subsets of training data, which makes random forest one of the most powerful algorithms in machine learning.

Decision Tree Terminologies

Some of the common Terminologies used in Decision Trees are as follows:

- Root Node: It is the topmost node in the tree, which represents the complete dataset. It is the starting point of the decision-making process.
- Decision/Internal Node: A node that symbolizes a choice regarding an input feature. Branching off of internal nodes connects them to leaf nodes or other internal nodes.
- Leaf/Terminal Node: A node without any child nodes that indicates a class label or a numerical value.

•Splitting: The process of splitting a node into two or more sub-nodes using a split criterion and a selected feature.
•Branch/Sub-Tree: A subsection of the decision tree starts at an internal node and ends at the leaf nodes.
•Parent Node: The node that divides into one or more child nodes.
•Child Node: The nodes that emerge when a parent node is split.
•Impurity: A measurement of the target variable’s homogeneity in a subset of data. It refers to the degree of randomness or uncertainty in a set of examples. The Gini index and entropy are two commonly used impurity measurements in decision trees for classifications task
•Variance: Variance measures how much the predicted and the target variables vary in different samples of a dataset. It is used for regression problems in decision trees. Mean squared error, Mean Absolute Error, friedman_mse, or Half Poisson deviance are used to measure the variance for the regression tasks in the decision tree.
•Information Gain: Information gain is a measure of the reduction in impurity achieved by splitting a dataset on a particular feature in a decision tree. The splitting criterion is determined by the feature that offers the greatest information gain, It is used to determine the most informative feature to split on at each node of the tree, with the goal of creating pure subsets
•Pruning: The process of removing branches from the tree that do not provide any additional information or lead to overfitting.

Attribute Selection Measures:

Construction of Decision Tree: A tree can be “*learned*” by splitting the source set into subsets based on Attribute Selection Measures. Attribute selection measure (ASM) is a criterion used in decision tree algorithms to evaluate the usefulness of different attributes for splitting a dataset. The goal of ASM is to identify the attribute that will create the most homogeneous subsets of data after the split, thereby maximizing the information gain. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data.

Entropy:

Entropy is the measure of the degree of randomness or uncertainty in the dataset. In the case of classifications, It measures the randomness based on the distribution of class labels in the dataset.

The entropy for a subset of the original dataset having K number of classes for the ith node can be defined as:

$$H_i = -\sum_{k \epsilon K}^{n} p(i,k) \log_2 p(i,k)$$

Where,

- •S is the dataset sample.
- •k is the particular class from K classes
- •p(k) is the proportion of the data points that belong to class k to the total number of data points in dataset sample S. $p(k) = \frac{1}{n} \sum I(y = k)$
- •Here p(i,k) should not be equal to zero.

Important points related to Entropy:

1.The entropy is 0 when the dataset is completely homogeneous, meaning that each instance belongs to the same class. It is the lowest entropy indicating no uncertainty in the dataset sample.

2.when the dataset is equally divided between multiple classes, the entropy is at its maximum value. Therefore, entropy is highest when the distribution of class labels is even, indicating maximum uncertainty in the dataset sample.

3.Entropy is used to evaluate the quality of a split. The goal of entropy is to select the attribute that minimizes the entropy of the resulting subsets, by splitting the dataset into more homogeneous subsets with respect to the class labels.

4.The highest information gain attribute is chosen as the splitting criterion (i.e., the reduction in entropy after splitting on that attribute), and the process is repeated recursively to build the decision tree.

Gini Impurity or index:

Gini Impurity is a score that evaluates how accurate a split is among the classified groups. The Gini Impurity evaluates a score in the range between 0 and 1, where 0 is when all observations belong to one class, and 1 is a random distribution of the elements within classes. In this case, we want to have a Gini index score as low as possible. Gini Index is the evaluation metric we shall use to evaluate our Decision Tree Model.

$$\text{Gini Impurity} = 1 - \sum p_i^2$$

•p_i is the proportion of elements in the set that belongs to the i_{th} category.

Information Gain:

Information gain measures the reduction in entropy or variance that results from splitting a dataset based on a specific property. It is used in decision tree algorithms to determine the usefulness of a feature by partitioning the dataset into more homogeneous subsets with respect to the class labels or target variable. The higher the information gain, the more valuable the feature is in predicting the target variable.

The information gain of an attribute A, with respect to a dataset S, is calculated as follows:

$$\text{Information Gain(H, A)} = H - \sum \frac{|H_V|}{|H|} H_v$$

where

•A is the specific attribute or class label

•|H| is the entropy of dataset sample S

•$|H_V|$ is the number of instances in the subset S that have the value v for attribute A

Information gain measures the reduction in entropy or variance achieved by partitioning the dataset on attribute A. The attribute that maximizes information gain is chosen as the splitting criterion for building the decision tree.

Information gain is used in both classification and regression decision trees. In classification, entropy is used as a measure of impurity, while in regression, variance is used as a measure of impurity. The information gain calculation remains the same in both cases, except that entropy or variance is used instead of entropy in the formula.

How does the Decision Tree algorithm Work?
The decision tree operates by analyzing the data set to predict its classification. It commences from the tree’s root node, where the algorithm views the value of the root attribute compared to the attribute of the record in the actual data set. Based on the comparison, it proceeds to follow the branch and move to the next node.
The algorithm repeats this action for every subsequent node by comparing its attribute values with those of the sub-nodes and continuing the process further. It repeats until it reaches the leaf node of the tree. The complete mechanism can be better explained through the algorithm given below.

- •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 nodeClassification and Regression Tree algorithm.

Advantages of the Decision Tree:

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

Disadvantages of the Decision Tree:

1. The decision tree contains lots of layers, which makes it complex.

2. It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
3. For more class labels, the computational complexity of the decision tree may increase.

What are appropriate problems for Decision tree learning?
Although a variety of decision tree learning methods have been developed with somewhat differing capabilities and requirements, decision tree learning is generally best suited to problems with the following characteristics:

1. Instances are represented by attribute-value pairs:
In the world of decision tree learning, we commonly use attribute-value pairs to represent instances. An instance is defined by a predetermined group of attributes, such as temperature, and its corresponding value, such as hot. Ideally, we want each attribute to have a finite set of distinct values, like hot, mild, or cold. This makes it easy to construct decision trees. However, more advanced versions of the algorithm can accommodate attributes with continuous numerical values, such as representing temperature with a numerical scale.

2. The target function has discrete output values:
The marked objective has distinct outcomes. The decision tree method is ordinarily employed for categorizing Boolean examples, such as yes or no. Decision tree approaches can be readily expanded for acquiring functions with beyond dual conceivable outcome values. A more substantial expansion lets us gain knowledge about aimed objectives with numeric outputs, although the practice of decision trees in this framework is comparatively rare.

3. Disjunctive descriptions may be required:
Decision trees naturally represent disjunctive expressions.

4.The training data may contain errors:
“Techniques of decision tree learning demonstrate high resilience towards discrepancies, including inconsistencies in categorization of sample cases and discrepancies in the feature details that characterize these cases.”

5. The training data may contain missing attribute values:
In certain cases, the input information designed for training might have absent characteristics. Employing decision tree approaches can still be possible despite experiencing unknown features in some training samples. For instance, when considering the level of humidity throughout the day,

this information may only be accessible for a specific set of training specimens.

Practical issues in learning decision trees include:

- Determining how deeply to grow the decision tree,
- Handling continuous attributes,
- Choosing an appropriate attribute selection measure,
- Handling training data with missing attribute values,
- Handling attributes with differing costs, and
- Improving computational efficiency.
-

To build the Decision Tree, CART (Classification and Regression Tree) algorithm is used. It works by selecting the best split at each node based on metrics like Gini impurity or information Gain. In order to create a decision tree. Here are the basic steps of the CART algorithm:

1. The root node of the tree is supposed to be the complete training dataset.
2. Determine the impurity of the data based on each feature present in the dataset. Impurity can be measured using metrics like the Gini index or entropy for classification and Mean squared error, Mean Absolute Error, friedman_mse, or Half Poisson deviance for regression.
3. Then selects the feature that results in the highest information gain or impurity reduction when splitting the data.
4. For each possible value of the selected feature, split the dataset into two subsets (left and right), one where the feature takes on that value, and another where it does not. The split should be designed to create subsets that are as pure as possible with respect to the target variable.
5. Based on the target variable, determine the impurity of each resulting subset.
6. For each subset, repeat steps 2–5 iteratively until a stopping condition is met. For example, the stopping condition could be a maximum tree depth, a minimum number of samples required to make a split or a minimum impurity threshold.

7.Assign the majority class label for classification tasks or the mean value for regression tasks for each terminal node (leaf node) in the tree.

Classification and Regression Tree algorithm for Classification

Let the data available at node m be Q_m and it has n_m samples. and t_m as the threshold for node m. then, The classification and regression tree algorithm for classification can be written as :

$$G(Q_m, t_m) = \frac{n_m^{Left}}{n_m} H(Q_m^{Left}(t_m)) + \frac{n_m^{Right}}{n_m} H(Q_m^{Right}(t_m))$$

Here,

- H is the measure of impurities of the left and right subsets at node m. it can be entropy or Gini impurity.
- n_m is the number of instances in the left and right subsets at node m.

To select the parameter, we can write as:

$$t_m =_{t_m} H(Q_m, t_m)$$

Classification and Regression Tree algorithm for Regression

Let the data available at node m be Q_m and it has n_m samples. and t_m as the threshold for node m. then, The classification and regression tree algorithm for regression can be written as :

$$G(Q_m, t_m) = \frac{n_m^{Left}}{n_m} MSE(Q_m^{Left}(t_m)) + \frac{n_m^{Right}}{n_m} MSE(Q_m^{Right}(t_m))$$

Here,

$$MSE_{Q_m} = \sum_{y \epsilon Q_m} (\bar{y}_m - y)^2$$

$$\text{where, } \bar{y}_m = \frac{1}{n_m} \sum_{y \epsilon Q_m} y$$

- MSE is the mean squared error.
- n_m is the number of instances in the left and right subsets at node m.

To select the parameter, we can write as:

$$t_m =_{t_m} MSE(Q_m, t_m)$$

Strengths and Weaknesses of the Decision Tree Approach

The strengths of decision tree methods are:

- Decision trees are able to generate understandable rules.

•Decision trees perform classification without requiring much computation.

•Decision trees are able to handle both continuous and categorical variables.

•Decision trees provide a clear indication of which fields are most important for prediction or classification.

•Ease of use: Decision trees are simple to use and don’t require a lot of technical expertise, making them accessible to a wide range of users.

•Scalability: Decision trees can handle large datasets and can be easily parallelized to improve processing time.

•Missing value tolerance: Decision trees are able to handle missing values in the data, making them a suitable choice for datasets with missing or incomplete data.

•Handling non-linear relationships: Decision trees can handle non-linear relationships between variables, making them a suitable choice for complex datasets.

•Ability to handle imbalanced data: Decision trees can handle imbalanced datasets, where one class is heavily represented compared to the others, by weighting the importance of individual nodes based on the class distribution.

The weaknesses of decision tree methods :

•Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.

•Decision trees are prone to errors in classification problems with many classes and a relatively small number of training examples.

•Decision trees can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

•Decision trees are prone to overfitting the training data, particularly when the tree is very deep or complex. This can result in poor performance on new, unseen data.

•Small variations in the training data can result in different decision trees being generated, which can be a problem when trying to compare or reproduce results.

•Many decision tree algorithms do not handle missing data well, and require imputation or deletion of records with missing values.

•The initial splitting criteria used in decision tree algorithms can lead to biased trees, particularly when dealing with unbalanced datasets or rare classes.

•Decision trees are limited in their ability to represent complex relationships between variables, particularly when dealing with nonlinear or interactive effects.

•Decision trees can be sensitive to the scaling of input features, particularly when using distance-based metrics or decision rules that rely on comparisons between values.