**Decision Tree**

**Introduction to decision tree:**

Decision trees are a supervised machine learning method used as a data classification and regression model. A decision tree divides the data into sub-trees that are composed of other sub-trees and/or leaf nodes. A decision tree is made up of three different kinds of nodes.

* Decision Nodes: These nodes contain two or more branches.
* Leaf Nodes: The nodes at the bottom that represent decision.
* Root Node: A decision node that is at the highest level.

**Types of decision tree:**

There are 2 types of Decision trees:

1. **Classification trees:** are used when the dataset needs to be divided into classes that correspond to the target variable. It is used in classification type problem where the target variable is discrete values such as Male or Female, True or False, Spam or Not Spam.
2. **Regression trees:** are used in prediction-type problems when the target variable is continuous. Example, price, salary, age, etc.

**Advantages/disadvantages of decision tree:**

**Advantages:**

* Decision trees are easier to understand and consume due to their simple mathematical intuition and visual representations.
* Decision trees require less effort for data preparation during pre-processing than other algorithms.
* Normalization and scaling of data are not required for a decision tree.
* Missing values in the data have no significant impact on the process of building a decision tree.
* Decision trees can be used for classification and regression tasks, making them more versatile than other algorithms.

**Disadvantages:**

* Complex decision trees are prone to overfitting and do not scale well to new data.
* High variance estimators are used because a minor change in the data can cause a significant change in the structure of the decision tree, resulting in instability.
* The decision tree often requires more time to train the model.
* Not fully supported in scikit-learn machine learning library.

**Assumptions while creating Decision Tree:**

Some of the assumptions we make when using Decision Tree are as follows:

* Initially, the entire training set is regarded as the root.
* Categorical feature values are preferred. If the values are continuous, they are discretized before the model is built.
* Records are distributed recursively based on attribute values.
* A statistical approach is used to place attributes as the tree's root or internal node.

**How to choose the best attribute at each node:**

If the dataset has N attributes, deciding which attributes to place at the root or at different levels of the tree as internal nodes is a difficult step. The problem cannot be solved by randomly selecting any node to be the root. If we take a random approach, we may get bad results with low accuracy. Researchers proposed some criteria for resolving this attribute selection problem, like entropy and information gain.

* **Entropy**: Entropy quantifies the randomness of the information being processed. The higher the entropy, the more difficult it is to draw any conclusions from the data. The entropy function relative to a boolean classification, as the proportion, , of positive examples varies between 0 and 1 shown in Figure 1.

Chart

Description automatically generated with medium confidence

Figure 1. Entropy function relative to a boolean classification.

* S is a sample of training examples
* is the proportion of positive examples in S
* is the proportion of negative examples in S
* Entropy measures the impurity of S
* **Information Gain:** Information gain is a statistical property that quantifies how well a specific attribute separates training examples based on their target classification. The goal of building a decision tree is to find an attribute with the highest information gain and the lowest entropy.

Where Values(A) is the set of all possible values for attribute A, and Sv, is the subset of S for which attribute A has value v.

**How to create Decision Tree: Designing decision tree on Play Tennis data**

## **What is decision tree analysis used for?**

Figure 2. depicts an example of a learned decision tree. This decision tree categorizes Saturday mornings based on whether they are suitable for playing tennis.

Diagram

Description automatically generated

Figure 2. A decision tree for the Play Tennis concept. An example is classified by traversing the tree to the appropriate leaf node and returning the classification associated with that leaf (in this case, Yes or No).

Table 1. Play tennis sample dataset

Table

Description automatically generated

Consider the Table 1. and use the ID3 algorithm to create the decision tree and determine which attribute is the best classifier. The following steps to be performed to do this:

Step 1: Apply entropy formula and compute entropy of whole dataset.

There are 14 examples of some boolean concept, with 9 positive and 5 negative examples . The entropy of S in relation to this boolean classification is:

Note:

If all members of S belong to the same class, the entropy is **zero(0).**

If dataset contains equal number of positive and negative class, the entropy is **1.**

If the dataset has unequal number of positive and negative class, the entropy is between 0 and 1**.**

Step 2: Calculate the information gain of each attribute in the dataset.

For example, attribute ***wind*** which has values *weak* and *strong*. S is the collection containing 14 examples, [9+, 5-]. Of these 14 example 6 of the positive and 2 of the negative examples [6+,2-] when ***wind =*** *weak* and for remainder 6 examples, 3 of the positive and 3 of the negative example [3+,3-] when ***wind =*** *strong.* The information gain due to sorting the original 14 examples by the attribute ***Wind*** may then be calculated as:

Similarly, calculate the information gain of remaining attributes. The information gain of all 4 attributes are:

Over the training examples, the ***Outlook*** attribute provides the best prediction of the target attribute, PlayTennis, according to the information gain measure. Therefore, ***Outlook*** will be consider as root node of the decision tree and branches are created below the root for each of its possible values (i.e., Sunny, Overcast, and Rain). The resultant partial decision tree as shown in below Figure 3.

Diagram

Description automatically generated

Figure 3. The partially learned decision tree produced by the first 2 steps of ID3.

Step 3: Apply Step 2 to decide which attribute should be left leaf node of the decision tree.

Calculate the information gain of remaining attributes when attribute ***Outlook=****sunny.* The resultant information gain of attributes when ***Outlook=****sunny* are:

The results shows that attribute ***Humidity*** has highest information gain. Therefore, ***Humidity*** will be consider as the left leaf node of the decision tree. The ***Outlook***=*Overcast* descendant has only positive examples and therefore becomes a leaf node with classification Yes.

Step 4: Apply the same procedure for other two nodes by selecting the attribute with highest information gain relative to the new subsets of examples.

**Decision tree overfitting:**

The ID3 algorithm described grows each branch of the tree just deeply enough to classify the training examples perfectly. Although this is sometimes a reasonable strategy, it can lead to problems when there is noise in the data or when the number of training examples is insufficient to produce a representative sample of the true target function. This simple algorithm can produce trees that ***overfit*** the training examples in either of these cases.

Consider error of hypothesis h over

* training data: *error*train(*h*)
* entire distribution D of data: *error*D(*h*)

Hypothesis h H ***overfits*** training data if there is an alternative hypothesis  H such that *errortrain(h)* < *errortrain()* and *errorD(h)* > *errorD()*

Figure 4 depicts the impact of overfitting in a typical decision tree learning application. As the tree grows, its accuracy over the training examples increases monotonically. However, the accuracy measured across the independent test examples increases at first, then decreases.

Chart, line chart

Description automatically generated

Figure 4. Overfitting in decision tree learning

**Avoiding Overfitting:**

How can we avoid overfitting?

* stop growing when data is split not statistically significant.
* grow the entire tree, then post-prune it.

How to Choose the Best Tree:

* Analyse performance using training data.
* Evaluate performance using a separate validation dataset.

There are two different methods for handling overfitting problems in decision trees.

**Reduced-Error Pruning:** Pruning a decision node involves removing the subtree rooted at that node, converting it to a leaf node, and assigning the most common classification of the training examples associated with that node. Only nodes are removed if the resulting pruned tree outperforms the original on the validation set. Figure 5 depicts the effect of reduced-error pruning on the decision tree's accuracy.

Chart, line chart

Description automatically generated

Figure 5. Effect of reduced-error pruning in decision tree learning

**Rule Post-Pruning:** A technique known as rule post-pruning is a highly effective method for discovering high-accuracy hypotheses. Rule post-pruning involves the following steps:

* Convert the tree into an equivalent set of rules.
* Each rule should be pruned independently of the others.
* Sort the final rules into the desired order for use.

**Evaluation of decision tree model:** After training, the decision tree classifier must be evaluated on previously unseen testing data. A confusion matrix is used to summarize the performance of a classification algorithm in order to evaluate it. There are two ways to evaluate the classification models.

* **Confusion Matrix:** A confusion matrix will show us the classification model's performance and the types of errors it produces. It provides a summary of correct and incorrect predictions for each category.

When evaluating the performance of a classification model, four outcomes are possible.

|  |  |  |
| --- | --- | --- |
|  | **Predicted**  **1** | **Predicted**  **0** |
| **Actual**  **1** | True Positive  **(TP)** | False Negative  **(FN)** |
| **Actual**  **0** | False Positive  **(FP)** | True Negative  **(TN)** |

**TP:** It occurs when the actual and predicted values are both positive.

**FN:** It occurs when the actual value is positive but the predicted value is negative.

**TN:** It occurs when the actual observation is negative and the model predicted value is negative.

**FP:** It occurs when the actual value is negative and the model predicted value is positive.

* **Classification Report :** A classification report is another way to assess the performance of the classification model. It displays the model's precision, recall, f1, and support scores. We can print the following classification reports using the below code:

**Precision:**  It is determined as the sum of true positives across all classes divided by the sum of true positives and false positives across all classes.

**Recall:** It is determined as the sum of true positives across all classes divided by the sum of true positives and false negatives across all classes.

**F1-Score:** It is the harmonic mean of two additional measures, namely precision and recall.

The performance metrics R-square, MSE, MAE, and MAPE will be used to evaluate the decision tree regression model. Session 3 covered the theoretical and implementation aspects of these metrics. The below code is used to evaluate the performance of decision tree regression.