ASSIGNMENT

ON

DATA SCIENCE
&
MACHINE LEARNING

Submitted By,

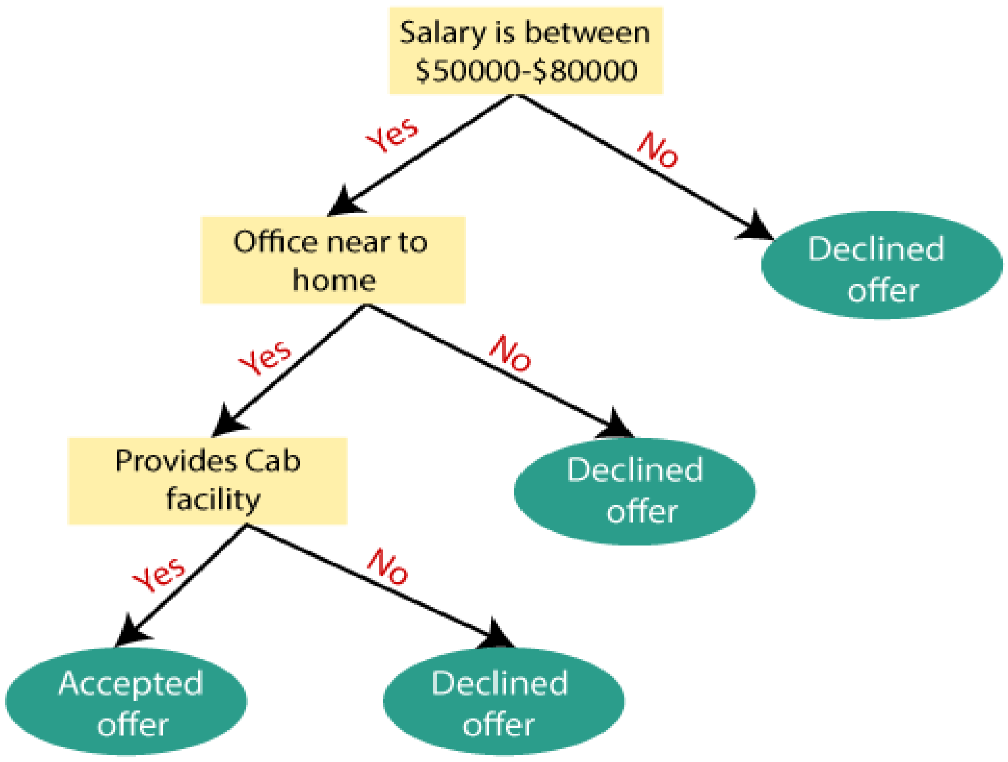
Febin Babu

MCA – A

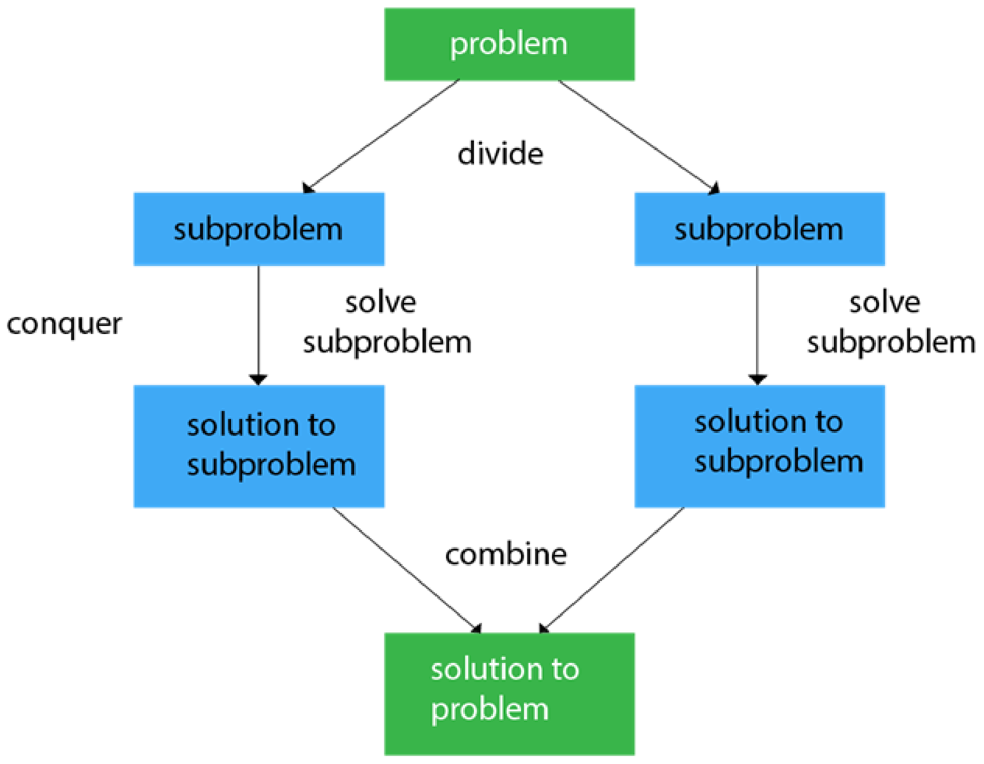
Roll No: 51

DECISION TREES

- Decision trees are an approach used in supervised machine learning, a technique which uses labelled input and output datasets to train models. The approach is used mainly to solve classification problems, which is the use of a model to categorise or classify an object.
- Decision tree learners are powerful classifiers, which utilize a tree structure to model the relationships among the features and the potential outcomes.
- A decision tree classifier uses a structure of branching decisions, which channel examples into a final predicted class value.
- A great benefit of decision tree algorithms is that the flowchart like tree structure is not necessarily exclusively for the learner's internal use.
- After the model is created, many decision tree algorithms output the resulting structure in a human-readable format.
- Decision trees particularly appropriate for applications in which the classification mechanism needs to be transparent for legal reasons, or in case the results need to be shared with others in order to inform future business practices.



Divide and conquer Approach



A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

Decision trees are built using a heuristic called recursive partitioning. This approach is also commonly known as divide and conquer because it splits the data into subsets, which are then split repeatedly into even smaller subsets, and so on and so forth until the process stops when the algorithm determines the data within the subsets are sufficiently homogenous, or another stopping criterion has been met.

At first, the root node represents the entire dataset, since no splitting has transpired. Next, the decision tree algorithm must choose a feature to split upon; ideally, it chooses the feature most predictive of the target class. The examples are then partitioned into groups according to the distinct values of this feature, and the first set of tree branches are formed. Working down each branch, the algorithm continues to divide and conquer the data, choosing the best candidate feature each time to create another decision node, until a stopping criterion is reached.

Divide and conquer might stop at a node in a case that:
– All (or nearly all) of the examples at the node have the same class
– There are no remaining features to distinguish among the examples
– The tree has grown to a predefined size limit.

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is 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.

Construction of Decision Tree: A tree can be *"learned"* by splitting the source set into subsets based on an attribute value test. 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. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

The C5.0 decision tree 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 Iterative Dichotomiser 3 (ID3) algorithm.

The C5.0 algorithm has become the industry standard to produce decision trees, because it does well for most types of problems directly out of the box. Compared to other advanced machine learning models, 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.

Choosing the best split

The first challenge that a decision tree will face is to identify which feature to split upon. 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 pure.

There are various measurements of purity that can be used to identify the best decision tree splitting candidate. – C5.0 uses entropy, a concept borrowed from information theory that quantifies the randomness, or disorder, within a set of class values. – 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. The decision tree hopes to find splits that reduce entropy, ultimately increasing homogeneity within the groups.

Typically, 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 log2 (n). In each case, the minimum value indicates that the sample is completely homogenous, while the maximum value indicates that the data are as diverse as possible, and no group has even a small plurality.

In the mathematical notion, entropy is specified as the given formula. Here, for a given segment of data (S), the term c refers to the number of class levels and pi refers to the proportion of values falling into class level i.

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

To use entropy to determine the optimal feature to split upon, the algorithm calculates the change in homogeneity that would result from a split on each possible feature, which is a measure known as information gain. The information gain for a feature F is calculated as the difference between the entropy in the segment before the split (S1) and the partitions resulting from the split (S2):

InfoGain(F) = Entropy(s1) - Entropy(s2)

One complication is that after a split, the data is divided into more than one partition. Therefore, the function to calculate Entropy(S2) needs to consider the total entropy across all of the partitions. It does this by weighing each partition's entropy by the proportion of records falling into the partition. In simple terms, the total entropy resulting from a split is the sum of the entropy of each of the n partitions weighted by the proportion of examples falling in the partition (wi).

Pruning the decision tree

A decision tree can continue to grow indefinitely, choosing splitting features and dividing the data into smaller and smaller partitions until each example is perfectly classified or the algorithm runs out of features to split on. However, if the tree grows overly large, many of the decisions it makes will be overly specific and the model will be overfitted to the training data. 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 when 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 to this approach 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 intentionally too large and pruning leaf 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.