UNIT-III

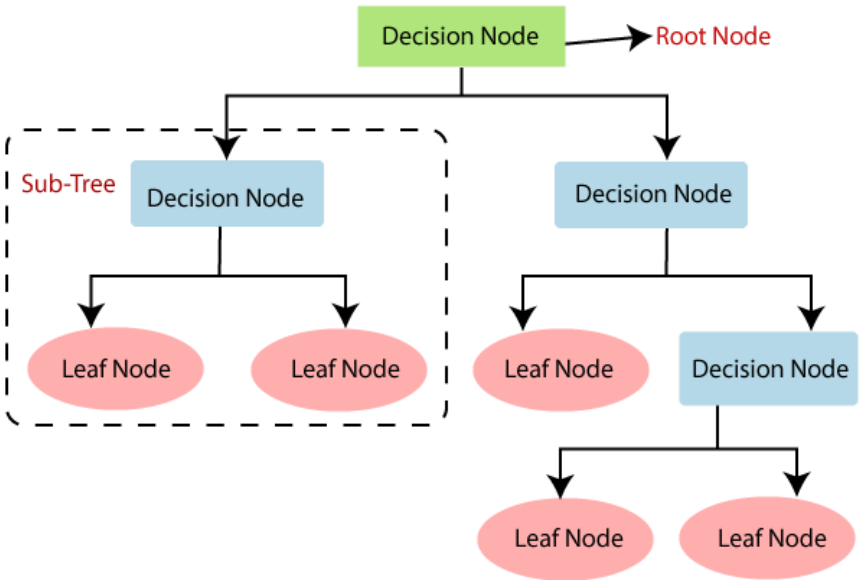
Learning with Trees: Decision Trees, Constructing Decision Trees, Classification and Regression Trees.

Ensemble Learning: Boosting, Bagging, Different ways to combine classifiers, Basic Statistics, Gaussian Mixture Models, Nearest Neighbour Methods.

Unsupervised Learning: K Means Algorithm.

Decision Trees:

- 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 tests 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.



Example:

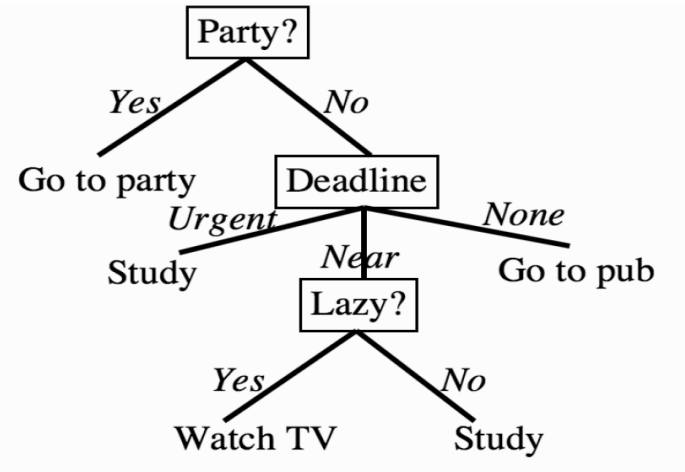


FIGURE 12.1 A simple decision tree to decide how you will spend the evening.

- One of the reasons that decision trees are popular is that we can turn them into a set of logical disjunctions (if ... then rules) that then go into program code very simply.
  Ex: if there is a party then go to it
  if there is not a party and you have an urgent deadline then study

Constructing Decision Trees:

Types of Decision Tree Algorithms:

- ID3: This algorithm measures how mixed up the data is at a node using something called entropy. It then chooses the feature that helps to clarify the data the most.
- C4.5: This is an improved version of ID3 that can handle missing data and continuous attributes.
- CART: This algorithm uses a different measure called Gini impurity to decide how to split the data. It can be used for both classification (sorting data into categories) and regression (predicting continuous values) tasks.

ID3 Algorithm:

Entropy in Information Theory:

- Entropy measures the amount of impurity in a set of features.
- The entropy H of a set of probabilities p_i is:

$$\text{Entropy}(p) = -\sum_i p_i \log_2 p_i,$$

- where the logarithm is base 2 because we are imagining that we encode everything using binary digits (bits), and we define 0 log 0 = 0.

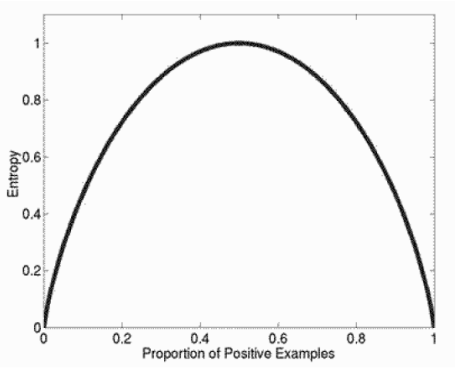


FIGURE 12.2 A graph of entropy, detailing how much information is available from finding out another piece of information given what you already know.

- If all of the examples are positive, then we don't get any extra information from knowing the value of the feature for any particular example, since whatever the value of the feature, the example will be positive. Thus, the entropy of that feature is 0.
- However, if the feature separates the examples into 50% positive and 50% negative, then the amount of entropy is at a maximum, and knowing about that feature is very useful to us.
- For our decision tree, the best feature to pick as the one to classify on now is the one that gives you the most information, i.e., the one with the highest entropy.

Information Gain:

- It is defined as the entropy of the whole set minus the entropy when a particular feature is chosen.

$$\text{Gain}(S, F) = \text{Entropy}(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} \text{Entropy}(S_f). \qquad (12.2)$$

- The ID3 algorithm computes this information gain for each feature and chooses the one that produces the highest value.

The ID3 Algorithm

- If all examples have the same label:
  - return a leaf with that label
- Else if there are no features left to test:
  - return a leaf with the most common label
- Else:
  - choose the feature $\hat{F}$ that maximises the information gain of S to be the next node using Equation (12.2)
  - add a branch from the node for each possible value f in $\hat{F}$
  - for each branch:
    - calculate S_f by removing $\hat{F}$ from the set of features
    - recursively call the algorithm with S_f, to compute the gain relative to the current set of examples

C4.5 Algorithm:

- It is an improved version of ID3.
- Pruning is another method that can help us avoid overfitting.
- It helps in improving the performance of the Decision tree by cutting the nodes or sub-nodes which are not significant.
- Additionally, it removes the branches which have very low importance.
- There are mainly 2 ways for pruning:
- **Pre-pruning** – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance while growing the tree.
- **Post-pruning** – once our tree is built to its depth, we can start pruning the nodes based on their significance.
- C4.5 uses a different method called rule post-pruning.
- This consists of taking the tree generated by ID3, converting it to a set of if-then rules, and then pruning each rule by removing preconditions if the accuracy of the rule increases without it.
- The rules are then sorted according to their accuracy on the training set and applied in order.
- The advantages of dealing with rules are that they are easier to read and their order in the tree does not matter, just their accuracy in the classification.
- For Continuous Variables, the simplest solution is to discretise the continuous variable.
- Computation complexity of Decision Tree is O(dnlogn) where n is number of data points, d is number of dimensions.

Classification Example: construct the decision tree to decide what to do in the evening

Deadline?	Is there a party?	Lazy?	Activity
Urgent	Yes	Yes	Party
Urgent	No	Yes	Study
Near	Yes	Yes	Party
None	Yes	No	Party
None	No	Yes	Pub
None	Yes	No	Party
Near	No	No	Study
Near	No	Yes	TV
Near	Yes	Yes	Party
Urgent	No	No	Study

We start with which feature has to selected as a root node?

Compute Entropy of S:

$$\begin{aligned}\text{Entropy}(S) &= -p_{\text{party}}\log_2 p_{\text{party}} - p_{\text{study}}\log_2 p_{\text{study}} \\ &- p_{\text{pub}}\log_2 p_{\text{pub}} - p_{\text{TV}}\log_2 p_{\text{TV}} \\ &= -\frac{5}{10}\log_2\frac{5}{10} - \frac{3}{10}\log_2\frac{3}{10} - \frac{1}{10}\log_2\frac{1}{10} - \frac{1}{10}\log_2\frac{1}{10} \\ &= 0.5 + 0.5211 + 0.3322 + 0.3322 = 1.6855 \qquad (12.11)\end{aligned}$$

find which feature has the maximal information gain:

$$\begin{aligned}\text{Gain}(S,\text{Deadline}) &= 1.6855 - \frac{|S_{\text{urgent}}|}{10}\text{Entropy}(S_{\text{urgent}}) \\ &- \frac{|S_{\text{near}}|}{10}\text{Entropy}(S_{\text{near}}) - \frac{|S_{\text{none}}|}{10}\text{Entropy}(S_{\text{none}}) \\ &= 1.6855 - \frac{3}{10}\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) \\ &- \frac{4}{10}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4}\right) \\ &- \frac{3}{10}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) \\ &= 1.6855 - 0.2755 - 0.6 - 0.2755 \\ &= 0.5345 \qquad (12.12)\end{aligned}$$

$$\begin{aligned}\text{Gain}(S,\text{Party}) &= 1.6855 - \frac{5}{10}\left(-\frac{5}{5}\log_2\frac{5}{5}\right)\\ &- \frac{5}{10}\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{1}{5}\log_2\frac{1}{5} - \frac{1}{5}\log_2\frac{1}{5}\right)\\ &= 1.6855 - 0 - 0.6855\\ &= 1.0 \qquad (12.13)\end{aligned}$$

$$\begin{aligned}\text{Gain}(S,\text{Lazy}) &= 1.6855 - \frac{6}{10}\left(-\frac{3}{6}\log_2\frac{3}{6} - \frac{1}{6}\log_2\frac{1}{6} - \frac{1}{6}\log_2\frac{1}{6} - \frac{1}{6}\log_2\frac{1}{6}\right)\\ &- \frac{4}{10}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right)\\ &= 1.6855 - 1.0755 - 0.4\\ &= 0.21 \qquad (12.14)\end{aligned}$$

- Therefore, the root node will be the party feature, which has two feature values (‘yes’ and ‘no’), so it will have two branches coming out of it.
-

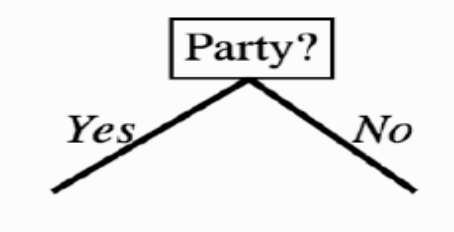


FIGURE 12.6 The decision tree after one step of the algorithm.

When we look at the ‘yes’ branch, we see that in all five cases where there was a party we went to it, so we just put a leaf node there, saying ‘party’.

- For the ‘no’ branch, out of the five cases there are three different outcomes, so now we need to choose another feature.
- The five cases we are looking at are:

Deadline?	Is there a party?	Lazy?	Activity
Urgent	No	Yes	Study
None	No	Yes	Pub
Near	No	No	Study
Near	No	Yes	TV
Urgent	No	Yes	Study

- We’ve used the party feature, so we just need to calculate the information gain of the other two over these five examples:

$$\begin{aligned}
\text{Gain}(S, \text{Deadline}) &= 1.371 - \frac{2}{5}\left(-\frac{2}{2}\log_2\frac{2}{2}\right) \\
&\quad - \frac{2}{5}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) - \frac{1}{5}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) \\
&= 1.371 - 0 - 0.4 - 0 \\
&= 0.971 \qquad (12.15)
\end{aligned}$$

$$\begin{aligned}
\text{Gain}(S, \text{Lazy}) &= 1.371 - \frac{4}{5}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4}\right) \\
&\quad - \frac{1}{5}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) \\
&= 1.371 - 1.2 - 0 \\
&= 0.1710 \qquad (12.16)
\end{aligned}$$

- Here, Deadline feature has maximum information gain. Hence, we selected Deadline feature for splitting data.

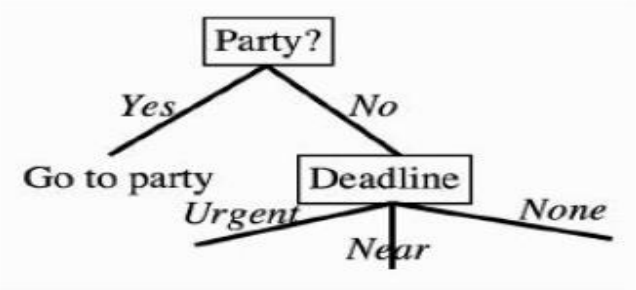
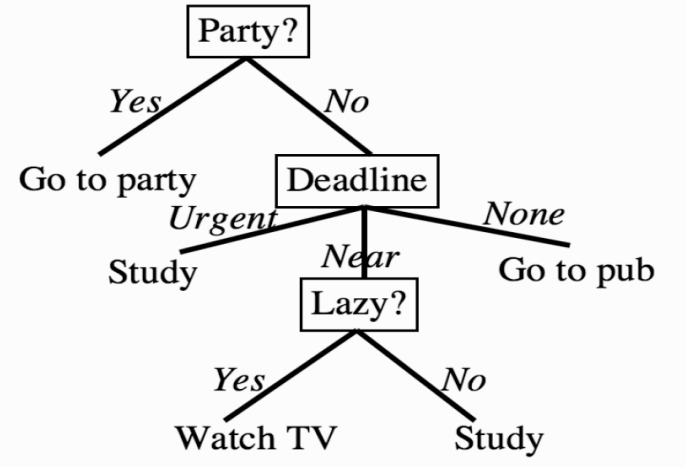


FIGURE 12.7 The tree after another step.

- Finally, we will get the following decision tree.



Classification and Regression Trees(CART):

- It is another well-known tree-based algorithm, CART, whose name indicates that it can be used for both classification and regression.

Gini Impurity:

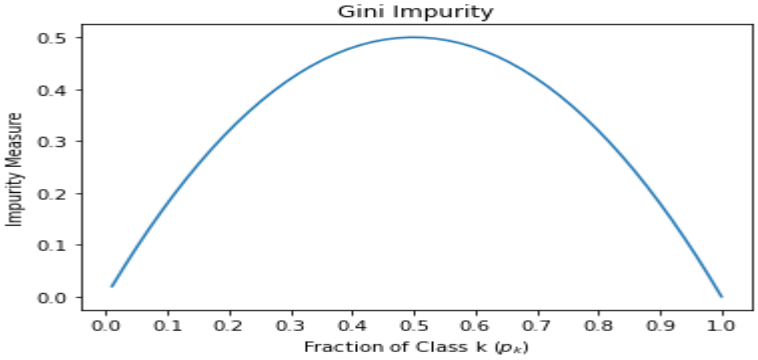
- It is the probability of misclassifying a randomly chosen element in a set.
- The 'impurity' in the name suggests that the aim of the decision tree is to have each leaf node represent a set of data points that are in the same class, so that there are no mismatches. This is known as purity.
- If a leaf is pure then all of the training data within it have just one class.
- Consider a dataset D that contains samples from k classes.
- The probability of samples belonging to class i at a given node can be denoted as p_i. Then the Gini Impurity of is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

- The node with uniform class distribution has the highest impurity.
- The minimum impurity is obtained when all records belong to the same class.

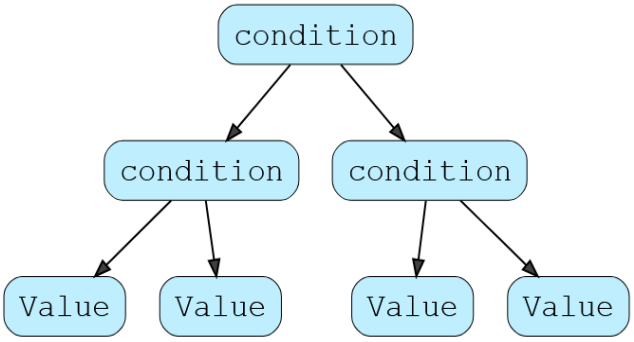
	Count		Probability		Gini Impurity
	n_1	n_2	p_1	p_2	$1 - p_1^2 - p_2^2$
Node A	0	10	0	1	$1 - 0^2 - 1^2 = 0$
Node B	3	7	0.3	0.7	$1 - 0.3^2 - 0.7^2 = 0.42$
Node C	5	5	0.5	0.5	$1 - 0.5^2 - 0.5^2 = 0.5$

- An attribute with the smallest Gini Impurity is selected for splitting the node.



Regression in Trees:

- A Regression tree is an algorithm where the target variable is continuous and the tree is used to predict its value**.**



- Regression Tree works by splitting the training data recursively into smaller subsets based on specific criteria.
- The objective is to split the data in a way that minimizes the residual reduction (Sum of Squared Error) in each subset.
- **Residual Reduction-** Residual reduction is a measure of how much the average squared difference between the predicted values and the actual values for the target variable is reduced by splitting the subset. The lower the residual reduction, the better the model fits the data.
- **Splitting Criteria-** CART evaluates every possible split at each node and selects the one that results in the greatest reduction of residual error in the resulting subsets. This process is repeated until a stopping criterion is met, such as reaching the maximum tree depth or having too few instances in a leaf node.