6.1 Introduction

It has been observed in the proceeding chapter that complex decisions cannot be made at a single point of time using a single piece of information. A complex decision requires several inputs that represent the states of nature. They provide many possible actions, with every available action having its own consequences. When the consequence of one action determines the new state of nature, decision making becomes sequential. Imagine a decision problem that has single initial state of nature and two actions are available. If all actions are deterministic, there will be two possible states of nature as the consequences of the two available actions. Suppose further that there are two possible actions corresponding two every state of nature, and the consequence of every action is a new state of nature. This decision problem will have one initial state of nature, two possible states of nature as the consequence of one decision, and hence one action. The second decision and the corresponding action will generate a total of 4 new states of nature. The third decision with its corresponding actions will generate a total of 8 consequences, each of them representing a new state of nature. In this way, a sequence of \( k \) decisions will give rise to a total of \( 2^k \) states of nature.

In order to illustrate the point, let’s consider an example of a family that goes out for lunch. The family has to decide whether to wait or not at the restaurant they reach. Obviously, the first question they ask is if there is enough space. They would wait only if there is sufficient space for the family. They approach another restaurant, without waiting, otherwise. If they find out that they will have to wait, the next question will be as to how long is the estimated waiting time before they get to be seated. If the estimated waiting time is less than 30 minutes, they decide to wait. Otherwise, they find if some family member is very if nobody is desperately hungry. If some family member is very hungry, and they decide to wait if nobody is desperately hungry. If some family member is very hungry they do not wait and decide to go to another restaurant. If they decide not to wait, they ask if there is another restaurant nearby, and decide to go to a nearby restaurant only if it is less than 15 minutes away by walk. Otherwise, they do not find it worthwhile to walk to another restaurant and prefer to wait. If there is a nearby restaurant
within 15 minutes of walk, they also check if it is raining. They can go to another restaurant only if it is not raining. Otherwise they need to wait to decide to wait. On the other hand, if the estimated waiting time exceeds 30 minutes, they may check if it is Friday or Saturday before deciding to wait because every other restaurant is also likely to be crowded on these two days. Even then, some family member may want to know if the restaurant has a bar and decide to wait if there is one. Waiting for some more time way then felt to be worth its while.

It can be seen in this simple example that the final decision to wait or not to wait is arrived at only after so much deliberation, after asking so many questions and after getting so much information. What is more important for an analyst is to identify all the possible situations that lead to the decision of waiting as well as all the possible situations that lead to the decision of not waiting. Theoretically, the simplest questions can have only two possible answers that can be represented by yes and no, or true and false. Such questions, or attributes, are said to be binary. When several such attributes are to be considered one after another, it is necessary to use Boolean algebra. A Boolean function may be expressed in the tabular form. Such a table possesses $2^k$ rows when there are $k$ attributes in the functions. Since the value of the Boolean function can be true or false, the number of possible Boolean functions with $k$ binary attributes as their arguments is $2^{2^k}$. In particular, when $k = 6$, the total number of possible distinct Boolean functions is 18,446,744,073,709,551,616. It is then easy to imagine how this number grows with increasing values of $k$. An alternative to the tabular representation of a Boolean function is the tree representation. A tree is a graph theoretical concept and hence it is necessary to review some basic and relevant concepts in graph theory in order to understand the use of trees to represent sequential decision making processes.

6.2 Graph Theory

It has been found by researchers that graph diagrams are better for understanding sequential decisions. The concept of a graph, therefore, plays a significant task in the study of processes of sequential-decision making. The beginning of the relevant part of graph theory is the concept of a directed graph.

Directed Graph: It is nothing but a pair $G = (V, E)$, where $V$ is a finite set of points that are known as nodes or vertices of the graph and $E$ is a set of pairs of elements in $V$, where elements of $E$ are called edges of links of the graph.
A directed graph can be shown in the pictorial form as a figure. The figure of a directed graph, on its own, has its vertices represented by points in a plane. Directed line segments joining pairs of vertices represents the edges or links between connected nodes or vertices. Consider a graph that five vertices, in order that
\[ V = \{u, v, w, x, y\} \]
and the following edges:
\[ E = \{(u, v), (v, u), (v, w), (v, x), (w, y), (y, x)\} \]. The edge \((u, v)\) is a link from the vertex \(u\) to the vertex \(v\) because the edge \((u, v)\) is directed. All the edges in a directed graph are directed or unidirectional. As a result, the diagram shows the edge \((u, v)\) by a line segment joining the points representing vertices \(u\) and \(v\) with an arrowhead at the end connected to the vertex \(v\). If a graph contains the edges \((u, v)\) and \((v, u)\), then the line segment connecting the two points in the diagram has arrowheads at both the ends and is then said to be a bidirectional link. It is a common practice to identify a directed graph with its diagram.

The diagram of the graph defined above can be shown as follows:

**Figure 6.1 Graph 1.**

```
  u    v    x
 /    /    \\
 w    y
```

Let another graph, called graph 2 have the vertices given by:
\[ V = \{1, 2, 3, 4, 5, 6\} \]
and edges given by:
\[ E = \{(1, 2), (6, 2), (6, 3), (3, 6), (3, 4), (4, 5)\} \].

**Figure 6.2 Graph 2**

```
   1
    \   /   /
     2  3  4
    /  /  /
   5  6
```

In a directed graph, if \((u,v)\) is an edge of the graph, \(u\) is a predecessor of \(v\) or \(v\) is said to be a successor of \(u\). For instance, in the figure of Graph 1, \(v\) is a predecessor of \(x\) and \(w\), whereas in the figure of Graph 2, the vertices 2 and 3 are both successors of the vertex 6.
A graph is said to contain a path from a vertex $u$ to another vertex $v$ if there are vertices $w_1, w_2, \ldots, w_k$ in the graph such that the pairs $(u, w_1), (w_1, w_2), (w_2, w_3), \ldots, (w_{k-1}, w_k), (w_k, v)$ are all edges of the directed graph. As a matter of fact, in such a case, the collection of edges listed above is also called a path of the graph from the vertex $u$ to the vertex $v$. A path from a vertex $u$ to another vertex $v$ describes how the vertex $v$ can be reached starting from the vertex $u$ if one follows the edges of the directed graph. If there is a path from the vertex $u$ to the vertex $v$, then it is also practice to describe this fact by saying that the vertex $u$ can be joined with the vertex $v$. The number of edges in a path is also called the length of the path. The path described above will sometimes denoted by $u \rightarrow w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow \ldots \rightarrow w_{k-1} \rightarrow w_k \rightarrow v$. In Graph 2, for instance, there is a path from vertex 6 to vertex 5 (the path $6 \rightarrow 3 \rightarrow 4 \rightarrow 5$), but there is no path from the vertex 3 to the vertex 1. In the directed graph in Graph 1, there are two paths from the vertex $v$ to the vertex $x$ (the path $v \rightarrow x$ and the path $v \rightarrow w \rightarrow y \rightarrow x$). Since the vertex $v$ is a successor or of the vertex $u$, there are two paths from the vertex $u$ to the vertex $x$ (the path $u \rightarrow v \rightarrow x$ and the path $u \rightarrow v \rightarrow w \rightarrow y \rightarrow x$).

When $(u,v)$ or $u \rightarrow v$ is an edge of a directed graph, the vertex $u$ is known as the origin of the edge. The edge is said to start from the vertex $u$. In this case, the vertex $v$ is called the end point or destination of the edge $(u,v)$ and the edge is said to end into the vertex $v$. As for the vertices are concerned, the edge $(u,v)$ is an departing edge for the vertex $u$ while the same edge is an arriving edge for the vertex $v$.

A vertex of a directed graph is called a terminal vertex if there is no edge starting from it. For example, the vertex $x$ in the directed graph of Graph 1 is terminal vertex, while the vertices 2 and 5 are the two terminal vertices in the directed graph of Graph 2.

Corresponding to each and every directed graph $G = (V, E)$, there is one more directed graph that is derived very naturally from the given graph. This new graph is known as the backward graph of $G$. It is the directed graph with same vertices that the given graph $G$ has. The edges of the backward graph are also same as the edges of the given graph $G$, but are in the opposite direction. In other words, the backward graph $(V, E')$ of the directed graph $G = (V, E)$ has the same vertices $V$ and the edges in the following set:

$E' = \{(u, v) \mid (v, u) \in E\}$. It is observe that the paths of the backward graph are the same paths of the original directed graph, but oriented in the reverse direction. For example, Graph $1'$ is the backward graph of Graph 1 and Graph $2'$ is the backward graph of Graph 2.
If $e = (u, v)$ is an edge of the directed graph $G$, $e$ is said to be incident outside $u$ and incident into $v$. In this case, the vertex $v$ is said to be an out-neighbour of $u$, whereas $u$ is called on in-neighbour of $v$.

Further, $N^+(u)$ indicates the set all out-neighbors of $u$ in $G$. Similarly, $N^-(u)$ indicates the set of all in-neighbors of $u$ in $G$. When there is no ambiguity and hence an explicit reference to $G$ is not necessary, these two sets are denoted by $N^+(u)$ and $N^-(u)$ respectively.

An edge $e$ is incident with a vertex $u$ if it is either incident into or incident out of $u$. The number of edges incident out of vertex $u$ is said to be the out degree of the vertex $u$ and is denoted by $d^+_G(u)$ or simply by $d^+_u$. Similarly, the number of edges that are incident into a vertex $v$ is called its in-degree and is denoted by $d^-_G(v)$ or simply by $d^-_v$.

For the directed graph Graph 1, it can be verified that $d^+_u = 1$, $d^-_u = 3$, $d^+_v = 1$, $d^-_v = 2$, and $d^-_x = 1$. Similarly, in Graph 2, $d^+_u = 1$, $d^-_u = 3$, $d^+_v = 1$, $d^-_v = 1$, $d^-_x = 0$, and $d^-_y = 0$.

The degree of a vertex $v$ in $G$ is said to be the sum of its in-degree and its out-degree. In other words, it is proved that $d(v) = d^+_v + d^-_v$ for all $v \in V$. It may be noted that every edge adds a count of one to the in-degree of one vertex while adding a count of one to the out-degree of another vertex. As a result, the total of in-degrees of all vertices must equal the total of out-degrees of all vertices.

Mathematically, this can be written as
\[
\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = m(G),
\]

where \( m(G) \) is the total number of edges in \( G \), that is, the cardinality of \( E \).”

A vertex of \( G \) is pendant if its degree is 1. It should then be obvious that, for a pendant vertex \( v \), either \( d^+(v) = 1 \) and \( d^-(v) = 0 \) or \( d^+(v) = 0 \) and \( d^-(v) = 1 \).

It is necessary to define some more terms related to a directed graph.

A walk in a graph \( G \) is an alternating of sequence of vertices and edges, beginning and ending with vertices.

Let \( W : v_0e_1v_1e_2v_2...e_pv_p \) be a walk in \( G \), where \( v_0, v_1, ..., v_p \) are vertices and \( e_1, e_2, ..., e_p \) are edges. It is clear from the expression for that the edge \( e_i \) has \( v_{i-1} \) and \( v_i \) at its two ends. The vertex \( v_0 \) is the origin and the vertex \( v_p \) is the terminus of \( W \). It is common to say that the walk \( W \) joins \( v_0 \) and \( v_p \). It is also alternatively referred to as a \( v_0-v_p \) walk. When the graph \( G \) is simple, a walk is determined by the sequence of the vertices in the walk. The walk \( W \) is said to be closed if \( v_0 = v_p \) and is said to be open otherwise. A trail is a special case of a walk where all edges appearing in the walk are distinct. A path is also a special case of a walk where all the vertices are distinct. It should be easy to note that a path is automatically a trial in \( G \). It is a common practice to omit edges when specifying a path. A cycle is a closed trial in which all the vertices are distinct. The length of a walk is the number of edges in it. A walk of length zero consists of only a single vertex.

Having developed an adequate understanding of graph theory, it may now be appropriate to define a tree as a graph theoretic structure. The literature on graph theory contains several definitions of a tree. Since a tree is a connected graph, it is better to first define a connected graph before defining a tree.

A graph \( G \) is known to be connected if there is a path among every pair of vertices.

A graph which is no connected is said to be disconnected. A graph having only one vertex is disconnected, even if there is no edge in such a graph. A graph that has no edges but has two or more vertices is necessarily disconnected.

A directed graph \( T \) is known to be a tree if there exists a unique vertex \( R \), known as the root of the tree, that has no edges going into it and for every vertex \( v \) dissimilar to \( R \), there exists a only one path from the root \( R \) to the vertex \( v \).
If is easy to see that Graph 1 and Graph 2 are not trees. Graph 3 given below is an example of a tree.

**Figure6.5 Graph 3**

![Diagram of Graph 3]

Note that the notation T for a tree is a reminder that it is not a general directed graph G. Similarly, there is some specific terminology that applies to trees and is convenient while also being easy to adopt.

If e = (u, v) is an edge of a tree, then u is called the parent of the vertex v and v is referred to as a child of the vertex u.

If there is a path from a vertex u to another vertex v, then the vertex u is called an ancestor of the vertex v and the vertex v is said to be a descendant of the vertex u.

In terms of this new terminology, the root R of a tree is the ancestor of every other vertex in the tree and, conversely, every other vertex of the tree is a descendant of the root R.

It may be noted that every path, P(u,v) is the tree with the root u, and terminal vertex v.

A branch of a tree T is a directed graph with vertices opening from a vertex u and comprising all of its descendants attached through their primary edges. The branch starting at a vertex u is denoted by T_u. It is very easy to verify that T_u, on its own, is a tree having the vertex u as its root vertex. The tree T_u is called a sub-tree because the set of vertices (and also edges) of T_u is a subset of vertices (and also edges) of the original tree T.

A formal definition of a sub-tree will make later discussion convenient.

**Sub-tree:** A tree S is said to be a sub-tree of a tree T if and only if

a) The vertices of S form a subset of the vertices of T,
b) The edges between vertices in $S$ are precisely the same edges. Joining these vertices when considered as vertices in the tree $T$,

c) The terminal vertices of $S$ form a subset of the terminal vertices of $T$, and
d) The root of the trees is same as the root of the tree $T$.

It is also interesting to note some graph theoretic properties of a tree. A tree $T$ satisfies any of the following equivalent conditions:

- $T$ is connected and it has no cycles.
- $T$ is acyclic and a simple cycle gets formed if an edge is added to $T$.
- $T$ is connected, but does not continue to be connected if any edge is removed from $T$.
- Any two vertices in $T$ may be connected through a unique path.

If $T$ has a finite number of vertices and if this number is indicated by $n$, the above conditions are also equivalent to the following conditions:

- $T$ is connected and has $n-1$ edges.
- $T$ has no simple cycles and has $n-1$ edges.

For a tree, an internal vertex, also known as an inner vertex or a branch vertex, is a vertex of degree at least two. Similarly, an external vertex, also known as an outer vertex, a terminal vertex, or a leaf, is a vertex of degree 1.

A irreducible tree is a tree that has no vertex of degree 2.

The term “tree” was coined by the British mathematician Arthur Cayley in 1857.

It is interesting to note how rapidly the number of trees grows as the number of vertices in the tree grows. If $T_n$ denotes the number of distinct trees having $n$ vertices, then it is easy to show that $T_1 = 1$, $T_2 = 1$, $T_3 = 2$, $T_4 = 4$, $T_5 = 9$, $T_6 = 20$, $T_7 = 48$, $T_8 = 115$, $T_9 = 286$, $T_{10} = 719$, $T_{11} = 1842$, $T_{12} = 4766$ and so on. The number of trees $T_n$ having $n$ vertices satisfies the following recurrence relations:

$$T_{n+1} = \frac{1}{n} \sum_{j=1}^{n} \left( \sum_{d|j} d \cdot T_d \right) T_{n-j+1},$$

Where $T_0 = 0$ and $T_1 = 1$. The summation inside the brackets is over all $d$ that divide $j$. Otter (1948) showed that

$$\alpha = \lim_{n \to \infty} \frac{T_n}{T_{n-1}} = 2.955765 ....$$
Where $\alpha$ is the only a positive root of the equation

$$T\left(\frac{1}{x}\right) = 1,$$

Here $T(x) = \sum_{n=0}^{\infty} T_n x^n$ is the generating function.

Graph theory contains a result that specifies the relationship between the number of edges in a tree and the number of vertices. The concerned theorem and its proof are as follows:

**Theorem:** The number of edges in a tree having $n$ vertices is $n-1$. Conversely, a connected graph having $n$ vertices and $n-1$ edges is a tree.

**Proof:** Let $T$ be a tree having $n$ vertices. The first part of the theorem will be proved by using mathematical induction. It is simple to make sure the theorem for $n=1$ and $n=2$, and arrive at the conclusion that $m = n - 1$. Here, by convention, $n$ indicates the number of vertices and $m$ indicates the number of edges.

Suppose, the result is true for all trees having $n-1$ or fewer vertices, where $n \geq 3$, let $T$ be a tree having $n$ vertices. Let $e = (u, v)$ be an edge of $T$, after that $e = (u,v)$ is a unique path in $T$ connecting $u$ and $v$. Therefore, removal of $e$ from $T$ will result in a disconnected graph that will have two components, say $T_1$ and $T_2$. Both $T_1$ and $T_2$ are connected graphs. Being connected sub-graphs of the $T$, both $T_1$ and $T_2$ are themselves trees. Further, since $n(T_1)$ and $n(T_2)$ are both less than $n(T)$, the induction hypothesis state that $m(T_1) = n(T_1) - 1$ and $m(T_2) = n(T_2) - 1$. Finally, consider $m(T) = m(T_1) + m(T_2) + 1$, the last term accounting for the removed edge that belongs to $T$ but to neither $T_1$ nor $T_2$. Therefore,

$$m(T) = m(T_1) + m(T_2) + 1$$

$$= (n(T_1) - 1) + (n(T_2) - 1) + 1$$

$$= n(T_1) + n(T_2) - 1$$

$$= n(T) - 1$$

As no vertex was removed from $T$ to obtain $T_1$ and $T_2$, this proves that the result for $T$. The principle of mathematical induction implies that the result holds for all $n \geq 1$.

Conversely, let $G$ is a connected graph with $n$ vertices and $n-1$ edges. Let $T$ be the minimal spanning tree of $G$, then $T$ will have $n$ vertices and $n-1$ edges, being a tree itself. As a consequence, $T$ must be same as $G$, since no edge has to be removed form $G$ for obtaining $T$. Thus, $G$ is a tree, proving the second (or converse) part of the theorem.
What is interesting is to note how scientists would use heuristics to make the same claim without resorting to such a lengthy, though rigorous, mathematical proof. A tree having \( n \) vertices has a unique root vertex and \( n-1 \) non-root vertices. Every non-root or descendant vertex requires an edge incident into it. Since there are \( n-1 \) descendant vertices, a tree having \( n \) vertices must have at least \( n-1 \) edges. This proves the necessity part. On the other hand, \( n-1 \) edges makes a tree a connected graph, implying that no more edges are necessary to obtain a connected graph. This establishes sufficiency of \( n-1 \) edges for a tree having \( n \) vertices.

Having understood the structure of trees, it is, therefore, required to appreciate the nature of decision rules with respect to connect decision making with tree construction, so that decision trees and their properties can be understood and investigated.

6.3 Nature of Decision Rules

6.3.1 Introduction

It is essential to know the nature of decisions, and rules regarding the decision in order to understand the relationship between decisions and trees. To begin with, it may be useful to understand what a decision is.

The dictionary meaning of the verb ‘decide’ is ‘to determine, to end, to resolve, to settle, and to make up one’s mind’, and so on. A person in the power is known to be a ‘decision-maker’ and a person who makes up his or her mind is known to be ‘decisive’. The meaning of its Latin root is ‘to cut away’. This means that making a decision is really a process of ‘cutting away the surrounding irrelevant details to the enable one to see a path to an objective and to follow that path with all its implications’. Making a decision does not mean allowing the events to have their course on their own. Outcomes will occur even when no decision is taken, but then the consequences need not be desirable.

Decisions can be classified into the following three types:

- **Strategic Decisions**: These are known as long-term decisions about the overall direction of an organization.
- **Tactical Decisions**: These are known as short-term decisions for doing things effectively and efficiently remaining well within the existing strategy.
- **Operational Decisions**: These are small decision involving immediate actions to be taken.
The decisions being discussed in this chapter are the last type, that is, operational decisions, and therefore have immediate consequences. Further, there are three different situations that require one to make a decision. These are as follows:

- Crisis: A crisis implies a sudden or unexpected event which requires a quick and urgent action which in turn requires a rapid decision or decisions.
- Problem: A situation that emerges slowly and gradually over time, but that is unclear until developing in a critical situation where a decision is imperative.
- Opportunity: An occasion that offers a reward for doing something and therefore requires a decision as to whether or not to make use of the opportunity.

Literature illustrates adaptive decisions and innovative decisions. Adaptive decisions are the decisions which contain human judgment and hence cannot be automated or computerized. Innovative decisions have no precedent and therefore cannot expect rules or guidelines developed with experience. It is also necessary to understand that all actions need not be due to decisions. Some actions may be taken out of a habit. Habitual behavior can be distinguished from rule-based decision-making by noting that rule require a significant amount of information-processing, while habits are automatic reactions that need not require any amount of information-processing. It must also be noted that habitual behavior can lack rationality since it may not depend on the available information to a decision maker. A good decision, nevertheless, is expected to be rational in the least. It may therefore be necessary as well as appropriate to describe rational decision making at this stage.

Rational decision making involves a various steps leading to the optimal decision under prevailing conditions as shown below:

1. Defining the situation that requires a decision and set criteria that the subsequent decision must satisfy.
2. Identifying alternative choices or options and collecting information on them.
3. Comparing all options with the existing criteria.
4. Selecting the appropriate option that satisfies a maximum number of criteria, treating this as the best option.
5. Implement the selected option.

6.3.2 Hard and Soft Decisions
When different available decisions are to be compared, what are the criteria for such a comparison? Do the criteria depend on how simple or complicated the decision is to implement? Do the criteria take into consideration the possible consequences of the available decisions? Rational decisions are those decisions that not only take the situation and the context in consideration, but also take the possible consequences, and most importantly, the most likely or the expected consequence, in considerations. A decision is usually considered to be hard if it involve serious or grave consequences and, in some cases, involve the commitment of a high level of resources. Consider, for example, the following two situations.

Situation 1: Your friend wants to borrow Rs. 50 from you.
Situation 2: Your friend wants to borrow Rs. 50,000 from you.

The decision of whether or not to lend the requested money to your friend may basically depend on this creditworthiness, but possible consequences are not comparable due to the difference in the magnitudes of the two amounts involved. So, some decisions are harder than others. An obvious questions, then, is what makes a decision hard. A decision is considered to be hard if

- The situation is uncertain.
- Moreover, a decision is also hard if
  - The situation is inherently complex with many other issues.
  - There are several objectives, but some of them are blocked.
  - Different perspectives can lead to different conclusions.

A robust and consistent approach to decision making, supported by the appropriate analysis has the following benefits:

- It deals with complexities by giving a formation for organizing issues. It has been found that the humans have a real problem when dealing with five or more variables.
- It identifies the underlying uncertainty and presents that in a structured manner.
- It deals with regard to a multiplicity of objectives and trade offs.
- It illustrates various perspectives and strengthens their logical presentation from a view of obtaining consensus, especially when many opinions are present.
- It encourages flexibility to change with changing circumstances because new circumstances may invalidate the earlier decisions.
- It provides a record of how the decision is reached, whatever has been taken into
considerations, whoever has been involved in making the decision, and so on. It is, at the same time, necessary to identify what can go wrong in decision making by identifying some common errors in decision making.

- **Haste:** Haste should not be confused with the notion of speed. A decision can be made in haste if it is made before all the facts are available or without taking all the facts into consideration. It is said in this context that “Decide in haste—regret at leisure”.

- **Narrow perspective:** It implies the activity of not having a sufficiently broad perspective may result in addressing an incorrect issue because the real issue may be judged previously due to narrowing of the perspective. It is also possible that a narrow perspective leads to an inappropriate framework of analysis that confines the consideration and keeps it away from the real issue.

- **Over-confidence:** The decision-maker may have over-confidence while making the decision itself or while understanding the issues and relevant information.

- **Thumb Rules:** The decision-maker may rely on approximate frameworks or shortcuts, instead of carrying out appropriate analysis.

- **Filtering:** Humans have a natural tendency of filtering and screening out unpleasant facts or information which do not support their notions of the decisions they want to make.

- **Juggling:** The decision-maker may lack the necessary analytical framework and, as a result, may be attempting to manage several variables or other bits of information manually.

A proper design for decision-making has the following seven steps:

- a) Defining the actual decision to be made accurately
- b) Understanding the context, in which the decision is to be made
- c) Identifying the existing choices
- d) Solving the consequences of each and every choice
- e) Prioritizing the choice in vision of their consequences and choose one choice
- f) Reviewing the selected option
- g) Taking action to implement the selected option

The decision is said to be made only when step 7 has been completed. Without going further in a detailed discussion of this framework, it may suffice to note that several real-life problems are not so simple as to be handled with this framework. It is therefore necessary as well
as convenient to break up the main problem into simple sub-problems, so that one sup-problem is solved after another. It is possible in such a situation that different decisions for the same sub-problem lead to different consequences and therefore to different consequent sub-problems. The consideration of the entire decision then cannot be presented in a single step. It requires several steps, with some steps following some other steps, while some steps may not be related to one another. What is important is to solve a simple problem that requires a one-step solution, and the action taken at a particular stage has a consequence that changes the context and hence the consequent problem is different. This consideration leads to a decision process, rather than an occasion for making a decision. As the name suggests, a decision process has several steps, and the entire process can be represented in a tabular form or a diagrammatic form. The tabular representation would produce a decision table, while a diagrammatic representation would produce a decision tree. Since graphical presentations are easier to comprehend than tabular presentations with numerical entries, decision trees have become a popular, standard, and common way of presenting a decision process.

It is the right time to get the main topic of this chapter, namely, decision trees. Some basic concepts of decision trees may be required before discussing their properties and advantages as well as limitations in optimal decision making.

6.4 Components of a Decision Trees Construction

The major mechanism of a decision tree form are nodes and branches, where as the most significant steps in constructing a decision tree form are as follows:

i) splitting, ii) stopping, and iii) pruning.

1. Nodes: Nodes can be of three types, and these are described here as follows:

a) A root node: Every decision tree has a unique root node which is also known as a decision node since it represents a decision resulting in a division of all the elements of data in two or more than two mutually exclusive subsets.

b) Internal nodes: The internal nodes are the outcome of one of the conditions tested at their parent nodes, and represent one of the available choices in the structure of a tree. Internal nodes are also called chance nodes. The edge that is incident in an internal node is connected to their parent node where as the edge which is incident out of an internal node is joined to its child nodes.

c) Leaf nodes: Every leaf node represents the ultimate outcome of a combination of events
or decisions. Leaf nodes are also called end nodes.

2. Branches: Branches representing the outcome spring from the root node and internal nodes. The decision tree is prepared out of a developed hierarchy i.e. pecking order of branches. Each and every path rising since the root node to a leaf node passes through the internal nodes, represents a decision rule. Each of these paths can be represented as a combination of several if … then rules. Every path begins at the root node and ends in a unique leaf node. The branches of a tree connect a parent node to two or more child nodes, but there cannot be a single child node of two parent nodes.

3. Splitting: Parent nodes are split into child nodes using input variables that are associated with the target variable. One parent node is split using one input variable, but this does not mean that the no. of splits will be equal to the no. of input variables. One input variable may be used many times at various nodes depending on the selection criterion for the splitting variable. Input variables can be either discrete or can be continuous. The range of a continuous input variable is partitioned to generate two or more categories. During the process of building the model, it is first required to identify the most important input variable. The records at the current node are then split into two or more categories according to the status of this variable. The split generates child nodes. The same process is carried out at every node until it is no more useful to split a node. Such a node is declared to be a leaf node and is assigned a label according to the status of the outcome variable at the node. The input variable is selected from among the potential input variables by using features which are associated with the degree of purity of the resulting child nodes. These features involve entropy, Gini index, classification error, information gain and gain ratio. The procedure of splitting is continued till a pre-decided criterion of homogeneity or stopping rule is satisfied. It is a common experience that every potential input variable may not be used in building the decision tree model. On the other hand, as has been noted earlier, some input variables can be applied lot of times at different levels and nodes of a decision tree.

4. Stopping: While building a statistical model, it is necessary to simultaneously consider the two competing characteristics of complexity and robustness. As the complexity of a model increases, its reliability in predicting future results decreases. An outermost position is to develop such a complex model that every leaf node of the resulting tree is 100 percent pure, that is, all records have achieved their target values. A decision tree built in this way would be an over fit because it fits 100 percent to the available data, but would not be good for generalization. This is so
because every leaf node contains very few records and so it cannot reliably predict future cases, thus lacking robustness. A tree can possess a low training error by have a large number of leaves, but its test, that is, generalization error is very often large. Developing deep and inaccurate trees should be avoided by imposing a stopping criterion. A series of stopping criteria found in literature is given below:

a) The node is pure, i.e. all observations in the node belong to only one class.
b) The size of the node is not as large as the minimum allowed size of a parent node.
c) A split of the current node that can result in a positive impurity gain would produce a child node smaller in size than the minimal allowed size of a child node.
d) The tree reaches the maximal depth.
e) The number of leaf nodes reaches the maximal permissible number.
f) The largest possible impurity gain is smaller than the specified threshold value.

An alternative to imposing a stopping rule is to develop a complete tree and removing branches with poor classification precision. The process of reducing a fully grown tree is known as pruning. It is believed by several researchers as well as practitioners that trees developed by pruning are more accurate than the trees developed by applying a traditional stopping rule.

5. Pruning: Pruning implies an activity of reducing a grown tree. Stopping rules, sometimes, failed to perform well. Another approach to develop a good quality decision tree is to develop a big tree first, and then cut down it to a minimum size by removing unnecessary nodes that do not add sufficient amount of information. One of the most common methods of choosing from the best possible sub-trees is to regard as the proportion of records with fault in estimation. Another method to select the best alternative is to use a legalization dataset. In such case, the sample is separated in two subsamples known as training set and test set. A decision tree develops on the training dataset and then tests on the test dataset. A third method, suggested by some researchers, is to split data into 10 parts, often called ‘folds’, and then developing the model on 9 of them before testing it on the 10th part. This procedure is repeated for all 10 possible combinations and then error in prediction is averaged over the ten repetitions.

Pruning is of two types, namely, pre-pruning (also known as forward pruning) and post-pruning (also known as backward pruning). Generation of non-significant branches is prevented through the use of chi-square testes and multiple-comparison adjustment methods by pre-pruning. Post-pruning, on the other hand, is used after creating a fully-developed decision tree.
for removing branches in order to attain the precision of the common classification when it is applied to the validation dataset.

Pruning is sometimes viewed as a form of regularization. It is a common practice to regularize a numerical model by modifying its purpose of including a term penalizing for complexity. Breiman et al. (1984) have applied this approach to pruning trees. This approach involves a new term, i.e. risk, and then the objective of learning a tree is defined as a risk minimization. The term ‘risk’ covers classification error as well as various impurity measures.

Let $r(t)$ be the risk associated with node $t$. The risk for the tree $T$ is the sum of risk values over all its leaf nodes. Therefore

$$r(t) = \sum_{t \in L(T)} r(t)$$

in which $L(T)$ is a set of leaf nodes in the tree $T$.

Any risk is estimated on the training dataset and it is low biased and very often, it approaches zero when the depth of a tree grows. Somewhat contrary to this phenomenon, the risk, estimated on the test data, reaches its minimum at some optimal depth of the tree and then grows as the tree develops deeper. Since complexity of the tree is understood by measuring its leaf nodes, the risk function is defined as

$$\tilde{r}(T) = r(T) + \alpha |L(T)|$$

where $|L(T)|$ is the number of leaf nodes in tree $T$ and $\alpha \geq 0$ is the penalty coefficient. The penalized risk is minimized for a fixed $\alpha$ to obtain the optimal tree $T^*(\alpha)$.

If $\alpha$ is set to zero, no penalty is imposed on the tree complexity. A full tree is itself the optimal tree $T^*(0)$ or the optimal tree is obtained by removing those splits from the full tree that do not reduce the risk. As the value of $\alpha$ is increased, the size of the optimal tree $T^*(\alpha)$ is reduced. When $\alpha$ has a large value, the optimal tree gets reduced to the root node only. A sequence of trees achieved by the parameter $\alpha$ is called the optimal pruning sequence. The optimal tree $T^*(\alpha_1)$ contains the (optimal) sub-tree $T^*(\alpha_2)$ when $\alpha_1 \leq \alpha_2$.

Suppose the optimal tree $T^*(\alpha_0)$ is obtained at the zero pruning level $\alpha_0$. It is necessary to obtain the smallest value of $\alpha$ for the initial pruning level, say $\alpha_1$. When a branch $T$ is reduced, it is substituted by its root node $t$. A value $\alpha^* \alpha$ is said to be critical when the two risks, one without
pruning and one with pruning are equal. The critical value \( \alpha^* \) is therefore a solution of the equation

\[
    r(t) + \alpha^* = r(T) + \alpha^*|L(T)|
\]

This gives, after solving for \( \alpha^* \),

\[
    \alpha^* = \frac{r(t) - r(T)}{|L(T)| - 1}
\]

Proceeding sequentially, the optimal pruning sequence can be found. The next question is to known the level to which the tree should be pruned. Minimization of the classification of error measured on a unique test dataset is one of the possible ways of choosing the optimal pruning level. This course of action is the most preferred if the training and test dataset are sufficiently large. It may be necessary to use cross-validation if only a limited amount of data is available. The sensitive problem is such a case is that the optimal pruning sequence may not be the same for different folds.

To be more specific, if 10-fold cross-validation is used, 10 trees grow, say \( \{T_k\}_{k=1}^{10} \) on nine tenths of the data each and the optimal pruning sequence is developed for every tree. It is not certain that every tree \( T_k \) is likely to have the same number of pruning levels as well as the same optimal pruning sequence \( \{\alpha^*_m\}_{m=0}^M \) as \( T \), the tree developed on the complete data. Every \( T_k \) can be assumed to be sufficiently close to \( T \) if the number of folds is sufficiently large. Suppose, for the sake of simplicity, that the maximal pruning level for \( M \) as well as for every \( T_k \) is \( M \). Then the optimal sequence \( \{\alpha^*_m\}_{m=0}^M \) is computed for \( T \). Suppose further that the optimal tree at the \( m \)th pruning level \( T^*_m(\alpha_m) \). It should be noted that \( \alpha_m \) is chosen as the smallest value of \( \alpha \) that allows elimination of some leaf nodes at the pruning level \( m \). It is then not necessary that \( \alpha_m \) will reduce trees in all folds to the \( m \)th pruning level. In other words, for some \( k \), \( T^*_k(\alpha_m) \) may be pruned to the level \( m-1 \), and not the level \( m \). Moreover, this level may even be different for different folds \( k \) or for different \( T^*_k(\alpha_m) \).

One possible way to handle this situation is to use the following proposed method. This method uses the fact that for any \( \alpha \) in the interval \( \alpha_m \leq \alpha < \alpha_{m+1} \), the optimal tree \( T^*_m(\alpha) \) is pruned at the \( m \)th pruning level. The proposal is prune every fold to the geometric mean \( \alpha^*_m = \sqrt{\alpha_m \cdot \alpha_{m+1}} \), instead of \( \alpha_m \). In this way, even though \( T^*_k(\alpha_m) \) may not be necessarily pruned to the \( m \)th level,
it may be pruned to the expected level than $T_k^\star (\alpha_m)$. It is then suggested to apply the pruned tree $T_k^\star (\alpha_m)$ in every fold to the test dataset and to count the number of misclassifications over all the folds to obtain the average rate of misclassification over the entire data.

There is no unique and simple solution to this problem. There is still a challenge of obtaining the optimal pruning strategy for optimizing the tree that will also have a similar performance in predicting new data points. This is an open research problem and is left open for researchers to work on.

6.5 Decision Tree Algorithms

It is worth noting that decision trees form one of the oldest machine learning algorithms. As a result, it should not be surprising to find that researchers and professional have taken a great amount of interest in developing new algorithms as well as software for decision tree induction. Many developers treat decision trees to be equivalent to classification trees and develop their methods accordingly. A quick search of the literature returned more than algorithms. It is not claimed that this search is exhaustive or even comprehensive. The following timeline shows the growth and the rate of growth of the interest that developers have shown in the topic.

1966: Hunt and others developed concept learning system framework
1979: Quinlan developed ID3
1984: Breimann and others published CART
1989: CS-ID3, GINI altered priors
1991: EG2, IDX
1995: ICET (Turney, 1995)
1998: C4.5CS (Ting, 1998)

\[
\begin{align*}
U \text{ Boost} \\
\text{Cost-U Boost} \\
\end{align*}
\] (Ting and Zheng, 1998)

1999: Meta Cost (Domingos, 1999)
\quad Ada Cost (Fan et al., 1999)
2000: Meta Cost – A (Ting, 2000 a)
\quad Meta Cost – CSB (Ting, 2000 a)
\quad CSB1 (Ting, 2000 b)
\quad CSB2 (Ting, 2000 b)
Cost plus Prior Probability
Cost only (Lin and McClean, 2000)

2002: AUC Split (Fern et al., 2002)
Multi tree (Estruch et al., 2002)

2003: SST Boost (Merler et al., 2003)
Costing (Zedrozny et al., 2003a, 2003b)
Powell’s Method
Eval Count
Max Cost (Margineantu and Dietterich, 2003)
Avg Cost

2004: GBSE
GBSE – T (Abe et al., 2004)
DT with Minimal Cost (Ling et al., 2004)
DT with Minimal Cost Under Resource Constraints (Qin et al., 2004)

2005: DTNB (Sheng and Ling, 2005)
ECCO (Omielan, 2005)
CSNL (Vadera, 2005a)
LDT (Vadera, 2005b)
CGP (Li et al., 2005)
Performance (Ni et al., 2005)

2006: B-PET
B-LOT (Moret et al., 2006)
CS Tree (Ling et al., 2006a)
Lazy Tree (Ling et al., 2006b)
CS Gain
CS Gain Ratio

2007: CS-C4.5 (Freitas et al., 2007)
JOUS Boost (Mease et al., 2007)
ACT (Esmeir and Markovitch, 2007)
PM (Liu, 2007)
CTS-DT (Zhang et al., 2007)
6.6 Comparisons between Some Decision Tree Algorithms

It is easy to understand why it is practically impossible to review all the algorithms. Even then, some of the popular algorithms are compared in some aspects in order to gain information on their relative strengths and weaknesses.

6.6.1 Comparison between ID3, C4.5, C5.0 and CART

1. Type of data
   - ID3 : Categorical
   - C4.5 : Continuous and Categorical
   - C5.0 : Continuous and Categorical, dates, times, timestamps
   - CART : Continuous and nominal attributes data

2. Speed
   - ID3 : Low
   - C4.5 : Faster than ID’3
   - C5.0 : Highest
   - CART : Average

3. Pruning
   - ID3 : No Pruning
   - C4.5 : Pre-Pruning
   - C5.0 : Pre-Pruning
   - CART : Post-Pruning

4. Missing Values
   - ID3 : Cannot deal with
   - C4.5 : Cannot deal with
   - C5.0 : Can deal with
   - CART : Can deal with

5. Formula
ID3 : Uses information entropy and information gain
C4.5 : Uses split information and gain ratio
C5.0 : Same as C4.5
CART : Uses Gini diversity index.

6.6.2 Comparative Analysis of Decision Tree Methods

Algorithm : ID3
Measure : Entropy information gain
Attribute : Categorical
Procedure : Top-down decision tree construction
Pruning : Pre-pruning using a single pass algorithm

Algorithm : C4.5
Measure : Gain ratio
Attribute : Continuous, Categorical, & Missing values
Procedure : Top-down decision tree construction
Pruning : Post-pruning using a single pass algorithm

Algorithm : CART
Measure : Gini diversity index
Attribute : Numeric & Categorical
Procedure : Constructs binary decision tree
Pruning : Post pruning based on cost-complexity

Algorithm : SLIQ
Measure : Gini index
Attribute : Numeric & Categorical
Procedure : Decision tree construction in a breadth first manner
Pruning : Post-pruning based on MDL principle

Algorithm : SPRINT
Measure : Gini index
Attribute : Continuous & Categorical
Procedure : Decision tree construction in a breadth first manner
Pruning : Post-pruning based on MDL principle