# Chapter 8

# Decision trees

"Decision tree learning is a method for approximating discrete valued target functions, in which the learned function is represented by a decision tree. Decision tree learning is one of the most widely used and practical methods for inductive inference." ([4] p.52)

## 8.1 Decision tree: Example

Consider the following situation. Somebody is hunting for a job. At the very beginning, he decides that he will consider only those jobs for which the monthly salary is at least Rs.50,000. Our job hunter does not like spending much time traveling to place of work. He is comfortable only if the commuting time is less than one hour. Also, he expects the company to arrange for a free coffee every morning! The decisions to be made before deciding to accept or reject a job offer can be schematically represented as in Figure 8.6. This figure represents a *decision tree*[1].

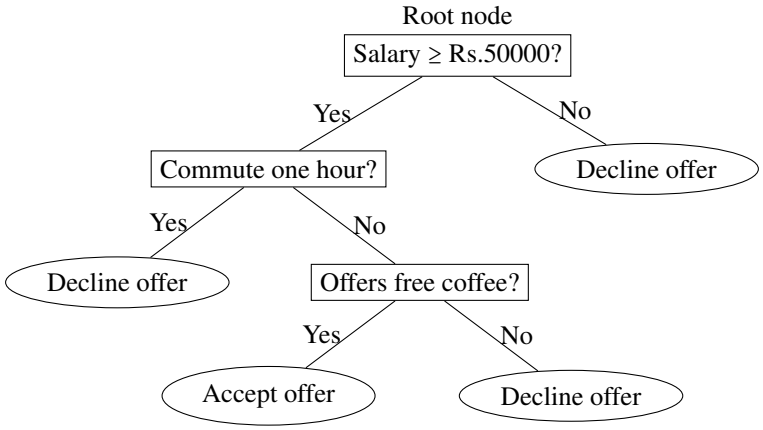


Figure 8.1: Example for a decision tree

Here, the term "tree" refers to the concept of a tree in graph theory in mathematics[2]. *In graph theory, a tree is defined as an undirected graph in which any two vertices are connected by exactly one path.* Using the conventions of graph theory, the decision tree shown in Figure 8.6 can be represented as a graph-theoretical tree as in Figure 8.2. Since a decision tree is a graph-theoretical tree, all terminology related to graph-theoretical trees can be applied to describe decision trees also. For example, in Figure 8.6, the nodes or vertices shown as ellipses are called the *leaf nodes*. All other nodes, except the root node, are called the *internal nodes*.

[1] In such diagrams, the "tree" is shown upside down with the root node at the top and all the leaves at the bottom.

[2] The term "tree" was coined in 1857 by the British mathematician Arthur Cayley (see Wikipedia).

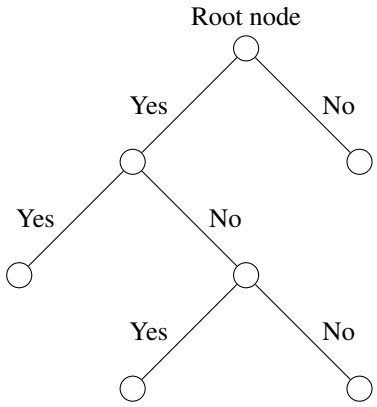


Figure 8.2: The graph-theoretical representation of the decision tree in Figure 8.6

## 8.2 Two types of decision trees

There are two types of decision trees.

1. **Classification trees**

   Tree models where the target variable can take a discrete set of values are called *classification trees*. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

2. **Regression trees**

   Decision trees where the target variable can take continuous values (real numbers) like the price of a house, or a patient's length of stay in a hospital, are called *regression trees*.

## 8.3 Classification trees

We illustrate the concept with an example.

### 8.3.1 Example

**Data**

| Nam | Features | | | | Class label |
|---|---|---|---|---|---|
| | gives birth | aquatic animal | aerial animal | has legs | |
| human | yes | no | no | yes | mammal |
| python | no | no | no | no | reptile |
| salmon | no | yes | no | no | fish |
| frog | no | semi | no | yes | amphibian |
| bat | yes | no | yes | yes | bird |
| pigeon | no | no | yes | yes | bird |
| cat | yes | no | no | yes | mammal |
| shark | yes | yes | no | no | fish |
| turtle | no | semi | no | yes | amphibian |
| salamander | no | semi | no | yes | amphibian |

Table 8.1: The vertebrate data set

Consider the data given in Table 8.1 which specify the features of certain vertebrates and the class to which they belong. For each species, four features have been identified: “gives birth”, ”aquatic animal”, “aerial animal” and “has legs”. There are five class labels, namely, “amphibian”, “bird”, “fish”, “mammal” and “reptile”. The problem is how to use this data to identify the class of a newly discovered vertebrate.

**Construction of the tree**

**Step 1**

We split the set of examples given in Table 8.1 into disjoint subsets according to the values of the feature “gives birth”. Since there are only two possible values for this feature, we have only two subsets: One subset consisting of those examples for which the value of “gives birth” is “yes” and one subset for which the value is “no”. The former is given in Table 8.2 and the latter in Table 8.3. This stage of the classification can be represented as in Figure 8.3.

| Name | Gives birth | Aquatic animal | Aerial animal | Has legs | Class label |
|---|---|---|---|---|---|
| human | yes | no | no | yes | mammal |
| bat | yes | no | yes | yes | bird |
| cat | yes | no | no | yes | mammal |
| shark | yes | yes | no | no | fish |

Table 8.2: The subset of Table 8.1 with “gives birth” = ”yes"

| Name | gives birth | aquatic animal | aerial animal | has legs | Class label |
|---|---|---|---|---|---|
| python | no | no | no | no | reptile |
| salmon | no | yes | no | no | fish |
| frog | no | semi | no | yes | amphibian |
| pigeon | no | no | yes | yes | bird |
| turtle | no | semi | no | yes | amphibian |
| salamander | no | semi | no | yes | amphibian |

Table 8.3: The subset of Table 8.1 with “gives birth” = ”no"

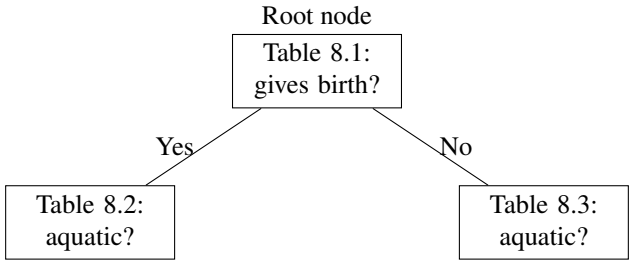


Figure 8.3: Classification tree

**Step 2**

We now consider the examples in Table 8.2. We split these examples based on the values of the feature “aquatic animal”. There are three possible values for this feature. However, only two of

| Name | gives birth | aquatic animal | aerial animal | has legs | Class label |
|---|---|---|---|---|---|
| human | yes | no | no | yes | mammal |
| bat | yes | no | yes | yes | bird |
| cat | yes | no | no | yes | mammal |

Table 8.5: The vertebrate data set

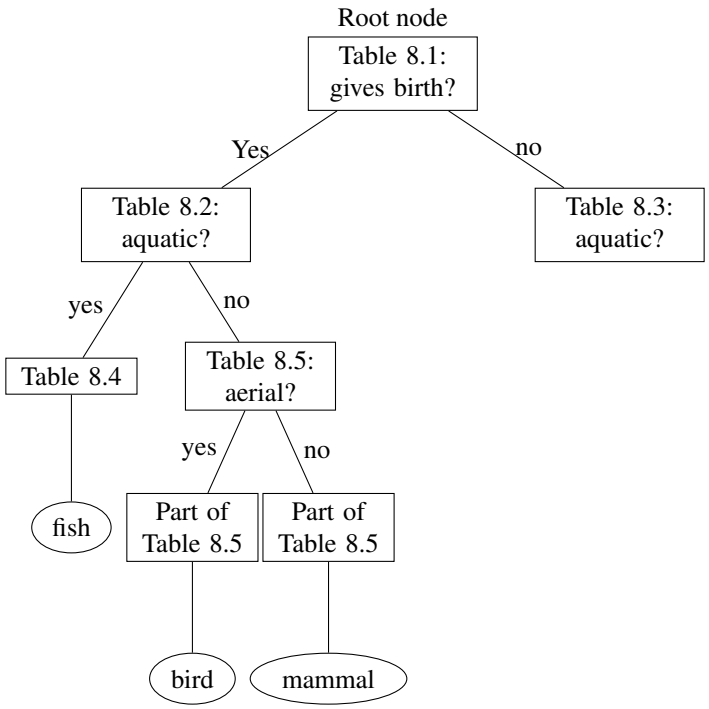


Figure 8.4: Classification tree

these appear in Table 8.2. Accordingly, we need consider only two subsets. These are shown in Tables 8.4 and 8.5.

| Name | gives birth | aquatic animal | aerial animal | has legs | Class label |
|---|---|---|---|---|---|
| shark | yes | yes | no | no | fish |

Table 8.4: The vertebrate data set

- Table 8.4 contains only one example and hence no further splitting is required. It leads to the assignment of the class label “fish”.
- The examples in Table 8.5 need to be split into subsets based on the values of “aerial animal”. It can be seen that these subsets immediately lead to unambiguous assignment of class labels: The value of “no” leads to “mammal” and the value “yes” leads to ”bird”.

At this stage, the classification tree is as shown in Figure 8.4

**Step 3**

Next we consider the examples in Table 8.3 and split them into disjoint subsets based on the values of "aquatic animal". We get the examples in Table 8.6 for "yes", the examples in Table **??** for "no" and the examples in Table **??** for "semi". We now split the resulting subsets based on the values of

| Name | gives birth | aquatic animal | aerial animal | has legs | Class label |
|---|---|---|---|---|---|
| salmon | no | yes | no | no | fish |

Table 8.6: The vertebrate data set

| Name | gives birth | aquatic animal | aerial animal | has legs | Class label |
|---|---|---|---|---|---|
| frog | no | semi | no | yes | amphibian |
| turtle | no | semi | no | yes | amphibian |
| salamander | no | semi | no | yes | amphibian |

Table 8.7: The vertebrate data set

| Name | gives birth | aquatic animal | aerial animal | has legs | Class label |
|---|---|---|---|---|---|
| python | no | no | no | no | reptile |
| pigeon | no | no | yes | yes | bird |

Table 8.8: The vertebrate data set

"has legs", etc. Putting all these together, we get the the diagram in Figure 8.5 as the classification tree for the data in Table 8.1.

### 8.3.2 Classification tree in rule format

The classification tree shown in Figure 8.5 can be presented as a set of rules in the form of an algorithm.

**Algorithm for classification of vertebrates**

```
1.  if give birth = "yes" then
2.      if aquatic = "yes" then
3.          return class = "fish"
4.      else
5.          if aerial = "yes" then
6.              return class = "bird"
7.          else
8.              return class = "mammal"
9.          end if
10.     end if
11. else
12.     if aquatic = "yes" then
13.         return class = "fish"
```

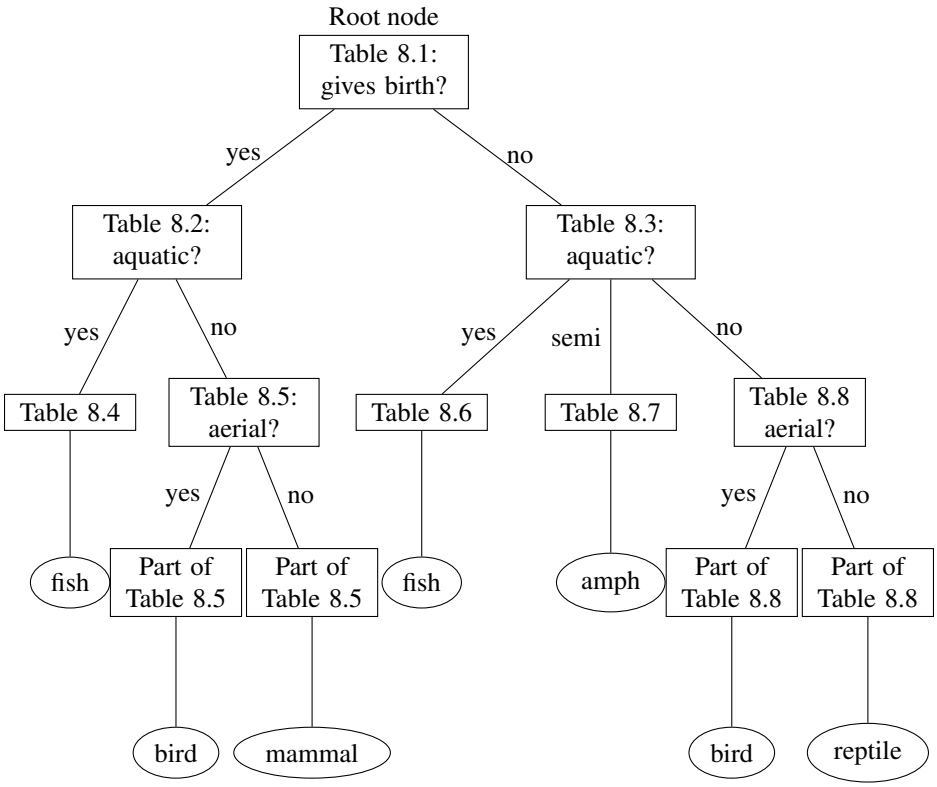


Figure 8.5: Classification tree

```
14.      end if
15.      if aquatic = "semi" then
16.           return  class = "amphibian"
17.      else
18.           if aerial = "yes" then
19.                return  class = "amphibian"
20.           else
21.                return  class = "reptile"
22.           end if
23.      end if
24. end if
```

### 8.3.3 Some remarks

**1. On the elements of a classification tree**

The various elements in a classification tree are identified as follows.

- Nodes in the classification tree are identified by the feature names of the given data.
- Branches in the tree are identified by the values of features.
- The leaf nodes identified by are the class labels.

**2. On the order in which the features are selected**

In the example discussed above, initially we chose the feature "gives birth" to split the data set into disjoint subsets and then the feature "aquatic animal", and so on. There was no theoretical justification for this choice. We could as well have chosen the feature "aquatic animal", or any other feature, as the initial feature for splitting the data. The classification tree depends on the order in which the features are selected for partitioning the data.

**3. Stopping criteria**

A real-world data will contain much more example record than the example we considered earlier. In general, there will be a large number of features each feature having several possible values. Thus, the corresponding classification trees will naturally be more complex. In such cases, it may not be advisable to construct all branches and leaf nodes of the tree. The following are some of commonly used criteria for stopping the construction of further nodes and branches.

- 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.

## 8.4 Feature selection measures

If a dataset consists of $n$ attributes then deciding which attribute is to be to placed at the root or at different levels of the tree as internal nodes is a complicated problem. It is not enough that we just randomly select any node to be the root. If we do this, it may give us bad results with low accuracy.

The most important problem in implementing the decision tree algorithm is deciding which features are to be considered as the root node and at each level. Several methods have been developed to assign numerical values to the various features such that the values reflect the relative importance of the various features. These are called the *feature selection measures*. Two of the popular feature selection measures are *information gain* and *Gini index*. These are explained in the next section.

## 8.5 Entropy

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 a *pure* class. Informally, entropy[3] is a measure of "impurity" in a dataset. 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.

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 $\log_2(n)$. In each case, the minimum value indicates that the sample is completely homogeneous, while the maximum value indicates that the data are as diverse as possible, and no group has even a small plurality.

### 8.5.1 Definition

Consider a segment $S$ of a dataset having $c$ number of class labels. Let $p_i$ be the proportion of examples in $S$ having the $i$ th class label. The entropy of $S$ is defined as

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

[3]From German Entropie "measure of the disorder of a system," coined in 1865 (on analogy of Energie) by German physicist Rudolph Clausius (1822-1888), in his work on the laws of thermodynamics, from Greek entropia "a turning toward," from en "in" + trope "a turning, a transformation,"

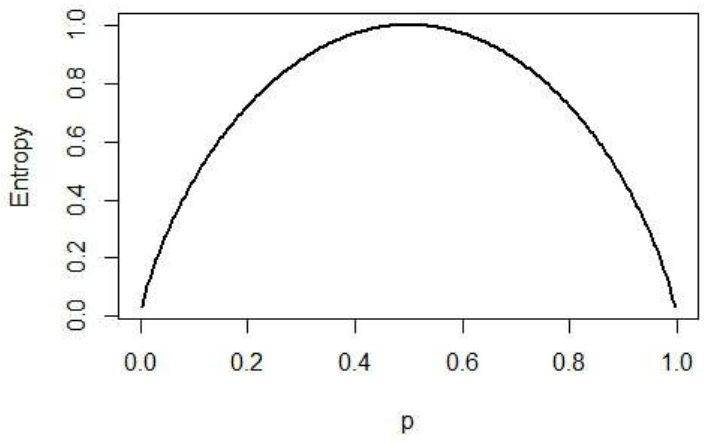


Figure 8.6: Plot of $p$ vs. Entropy

**Remark**

In the expression for entropy, the value of $0 \times \log_2(0)$ is taken as zero.

**Special case**

Let the data segment $S$ has only two class labels, say, "yes" and "no". If $p$ is the proportion of examples having the label "yes" then the proportion of examples having label "no" will be $1 - p$. In this case, the entropy of $S$ is given by

$$\text{Entropy}\,(S) = -p\log_2(p) - (1-p)\log_2(1-p).$$

If we plot the values of graph of Entropy $(S)$ for all possible values of $p$, we get the diagram shown in Figure 8.6[4].

### 8.5.2 Examples

Let "xxx" be some class label. We denote by $p_{\text{xxx}}$ the proportion of examples with class label "xxx".

1. **Entropy of data in Table 8.1**

   Let $S$ be the data in Table 8.1. The class labels are "amphi", "bird", "fish", "mammal" and "reptile". In $S$ we have the following numbers.

   | | |
   |---|---|
   | Number of examples with class label "amphi" | = 3 |
   | Number of examples with class label "bird" | = 2 |
   | Number of examples with class label "fish" | = 2 |
   | Number of examples with class label "mammal" | = 2 |
   | Number of examples with class label "reptile" | = 1 |
   | Total number of examples | = 10 |

   Therefore, we have:

$$\text{Entropy}\,(S) = \sum_{\text{for all classes "xxx"}} -p_{\text{xxx}}\log_2(p_{\text{xxx}})$$

[4]Plot created using R language.

$$
\begin{aligned}
&= -\,p_{\text{amphi}}\log_2(p_{\text{amphi}}) - p_{\text{bird}}\log_2(p_{\text{bird}}) \\
&\quad - p_{\text{fish}}\log_2(p_{\text{fish}}) - p_{\text{mammal}}\log_2(p_{\text{mammal}}) \\
&\quad - p_{\text{reptile}}\log_2(p_{\text{reptile}}) \\
&= -\,(3/10)\log_2(3/10) - (2/10)\log_2(2/10) \\
&\quad - (2/10)\log_2(2/10) - (2/10)\log_2(2/10) \\
&\quad - (1/10)\log_2(1/10) \\
&= \ 2.2464
\end{aligned}
$$

2. **Entropy of data in Table 8.2**

   Consider the segment $S$ of the data in Table 8.1 given in Table 8.2. For quick reference, the table has been reproduced below:

| Name | Gives birth | Aquatic animal | Aerial animal | Has legs | Class label |
|---|---|---|---|---|---|
| human | yes | no | no | yes | mammal |
| bat | yes | no | yes | yes | bird |
| cat | yes | no | no | yes | mammal |
| shark | yes | yes | no | no | fish |

   Three class labels appear in this segment, namely, “bird”, “fish” and “mammal”. We have:

| | |
|---|---|
| Number of examples with class label “bird” | 1 |
| Number of examples with class label “fish” | 1 |
| Number of examples with class label “mammal” | 2 |
| Total number of examples | 4 |

   Therefore we have

$$
\begin{aligned}
\text{Entropy}\,(S) &= \sum_{\text{for all classes “xxx”}} -p_{\text{xxx}}\log_2(p_{\text{xxx}}) \\
&= -\,p_{\text{bird}}\log_2(p_{\text{bird}}) - p_{\text{fish}}\log_2(p_{\text{fish}}) \\
&\quad - p_{\text{mammal}}\log_2(p_{\text{mammal}}) \\
&= -\,(1/4)\log_2(1/4) - (1/4)\log_2(1/4) - (2/4)\log_2(2/4) \\
&= -\,(1/4)\times(-2) - (1/4)\times(-2) - (2/4)\times(-1) \\
&= \ 1.5
\end{aligned}
\tag{8.1}
$$

3. **Entropy of data in Table 8.3**

   Consider the segment $S$ of the data in Table 8.1 given in Table 8.3. For quick reference, the table has been reproduced below:

| Name | gives birth | aquatic animal | aerial animal | has legs | Class label |
|---|---|---|---|---|---|
| python | no | no | no | no | reptile |
| salmon | no | yes | no | no | fish |
| frog | no | semi | no | yes | amphibian |
| pigeon | no | no | yes | yes | bird |
| turtle | no | semi | no | yes | amphibian |
| salamander | no | semi | no | yes | amphibian |

   Four class labels appear in this segment, namely, “amphi”, “bird”, “fish” and “reptile”. We have:

| | |
|---|---|
| Number of examples with class label "amphi" | 3 |
| Number of examples with class label "bird" | 1 |
| Number of examples with class label "fish" | 1 |
| Number of examples with class label "reptile" | 1 |
| Total number of examples | 6 |

Therefore, we have:

$$\begin{aligned}\text{Entropy}\,(S) &= \sum_{\text{for all classes "xxx"}} -p_{\text{xxx}} \log_2(p_{\text{xxx}}) \\ &= -p_{\text{amphi}} \log_2(p_{\text{amphi}}) - p_{\text{bird}} \log_2(p_{\text{bird}}) - p_{\text{fish}} \log_2(p_{\text{fish}}) \\ &\quad - p_{\text{reptile}} \log_2(p_{\text{reptile}}) \\ &= -(3/6)\log_2(3/6) - (1/6)\log_2(1/6) - (1/6)\log_2(1/6) \\ &\quad - (1/6)\log_2(1/6) \\ &= 1.7925 \end{aligned} \tag{8.2}$$

## 8.6 Information gain

### 8.6.1 Definition

Let $S$ be a set of examples, $A$ be a feature (or, an attribute), $S_v$ be the subset of $S$ with $A = v$, and Values $(A)$ be the set of all possible values of $A$. Then the *information gain of an attribute $A$ relative to the set $S$*, denoted by Gain $(S, A)$, is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}\,(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

where $|S|$ denotes the number of elements in $S$.

### 8.6.2 Example 1

Consider the data $S$ given in Table 8.1. We have have already seen that

$$\begin{aligned} |S| &= 10 \\ \text{Entropy}\,(S) &= 2.2464. \end{aligned}$$

We denote the information gain corresponding to the feature "xxx" by Gain $(S, \text{xxx})$.

1. **Computation of Gain $(S,$ gives birth$)$**

$$\begin{aligned} A_1 &= \text{gives birth} \\ \text{Values of } A_1 &= \{\text{"yes"}, \text{"no"}\} \\ S_{A_1=\text{yes}} &= \text{Data in Table 8.2} \\ |S_{A_1=\text{yes}}| &= 4 \\ \text{Entropy}\,(S_{A_1=\text{yes}}) &= 1.5 \quad \text{(See Eq.(8.1))} \\ S_{A_1=\text{no}} &= \text{Data in Table 8.3} \\ |S_{A_1=\text{no}}| &= 6 \\ \text{Entropy}\,(S_{A_1=\text{no}}) &= 1.7925 \quad \text{(See Eq.(8.2))} \end{aligned}$$

Now we have

$$\text{Gain}(S, A_1) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A_1)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

$$
\begin{aligned}
&= \text{Entropy}(S) - \frac{|S_{A_1=\text{yes}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{yes}}) \\
&\qquad - \frac{|S_{A_1=\text{no}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{no}}) \\
&= 2.2464 - (4/10) \times 1.5 - (6/10) \times 1.7925 \\
&= 0.5709
\end{aligned}
$$

2. **Computation of Gain** $(S,$ **aquatic**$)$

$$
\begin{aligned}
A_2 &= \text{aquatic} \\
\text{Values of } A_2 &= \{\text{“yes”}, \text{“no”}, \text{“semi”}\} \\
S_{A_2=\text{yes}} &= \text{See Table 8.1} \\
|S_{A_2=\text{yes}}| &= 2 \\
\text{Entropy } (S_{A_2=\text{yes}}) &= -p_{\text{fish}} \log_2(p_{\text{fish}}) \\
&= -(2/2) \log_2(2/2) \\
&= 0 \\
S_{A_2=\text{no}} &= \text{See Table 8.1} \\
|S_{A_2=\text{no}}| &= 5 \\
\text{Entropy } (S_{A_2=\text{no}}) &= -p_{\text{mammal}} \log_2(p_{\text{mammal}}) - p_{\text{reptile}} \log_2(p_{\text{reptile}}) \\
&\quad - p_{\text{bird}} \log_2(p_{\text{bird}}) \\
&= -(2/5) \times \log_2(2/5) - (1/5) \times \log_2(1/5) \\
&\quad - (2/5) \times \log_2(2/5) \\
&= 1.5219 \\
S_{A_2=\text{semi}} &= \text{See Table 8.1} \\
|S_{A_2=\text{semi}}| &= 3 \\
\text{Entropy } (S_{A_2=\text{semi}}) &= -p_{\text{amphi}} \log_2(p_{\text{amphi}}) \\
&= -(3/3) \times \log_2(3/3) \\
&= 0 \\
\text{Gain}(S, A_2) &= \text{Entropy}(S) - \sum_{v \in \text{Values}(A_2)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v) \\
&= \text{Entropy}(S) - \frac{|S_{A_1=\text{yes}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{yes}}) \\
&\qquad - \frac{|S_{A_1=\text{no}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{no}}) \\
&\qquad - \frac{|S_{A_1=\text{semi}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{semi}}) \\
&= 2.2464 - (2/10) \times 0 - (5/10) \times 1.5219 - (3/3) \times 0 \\
&= 1.48545
\end{aligned}
$$

3. **Computations of Gain** $(S,$ **aerial animal**$)$ **and Gain** $(S,$ **has legs**$)$
   These are left as exercises.

## 8.7 Gini indices

The Gini split index of a data set is another feature selection measure in the construction of classification trees. This measure is used in the CART algorithm.

### 8.7.1 Gini index

Consider a data set $S$ having $r$ class labels $c_1, \ldots, c_r$. Let $p_i$ be the proportion of examples having the class label $c_i$. The Gini index of the data set $S$, denoted by $\text{Gini}(S)$, is defined by

$$\text{Gini}(S) = 1 - \sum_{i=1}^{r} p_i^2.$$

**Example**

Let $S$ be the data in Table 8.1. There are four class labels "amphi", "bird", "fish", "mammal" and "reptile". The numbers of examples having these class labels are as follows:

| | |
|---|---|
| Number of examples with class label "amphi" | = 3 |
| Number of examples with class label "bird" | = 2 |
| Number of examples with class label "fish" | = 2 |
| Number of examples with class label "mammal" | = 2 |
| Number of examples with class label "reptile" | = 1 |
| Total number of examples | = 10 |

The Gini index of $S$ is given by

$$\begin{aligned}\text{Gini}(S) &= 1 - \sum_{i=1}^{r} p_i^2 \\ &= 1 - (3/10)^2 - (2/10)^2 - (2/10)^2 - (2/10)^2 - (1/10)^2 \\ &= 0.78\end{aligned}$$

### 8.7.2 Gini split index

Let $S$ be a set of examples, $A$ be a feature (or, an attribute), $S_v$ be the subset of $S$ with $A = v$, and $\text{Values}(A)$ be the set of all possible values of $A$. Then the *Gini split index of $A$ relative to $S$*, denoted by $\text{Gini}_{\text{split}}(S, A)$, is defined as

$$\text{Gini}_{\text{split}}(S, A) = \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Gini}(S_v).$$

where $|S|$ denotes the number of elements in $S$.

## 8.8 Gain ratio

The *gain ratio* is a third feature selection measure in the construction of classification trees.

Let $S$ be a set of examples, $A$ a feature having $c$ different values and let the set of values of $A$ be denoted by $\text{Values}(A)$. We defined the information gain of $A$ relative to $S$, denoted by $\text{Gain}(S, A)$, by

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values(A)}} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

We now define the*split information* of $A$ relative to $S$, dented by $\text{SplitInformation}(S, A)$, by

$$\text{SplitInformation}(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where $S_1, \ldots S_c$ are the $c$ subsets of examples resulting from partitioning $S$ into the $c$ values of the attribute $A$. The *gain ratio* of $A$ relative to $S$, denoted by $\text{GainRatio}(S, A)$, by

$$\text{GainRatio}(S, A) = \frac{\text{Gain(S,A)}}{\text{SplitInformation}(S, A)}.$$

### 8.8.1 Example

Consider the data $S$ given in Table 8.1. Let $A$ denote the attribute "gives birth".We have have already seen that

$$\begin{aligned} |S| &= 10 \\ \text{Entropy } (S) &= 2.2464 \\ \text{Gain}(S, A) &= 0.5709 \end{aligned}$$

Now we have

$$\begin{aligned} \text{SplitInformation}(S, A) &= -\frac{|S_{\text{yes}}|}{|S|}\log_2\frac{|S_{\text{yes}}|}{|S|} - \frac{|S_{\text{no}}|}{|S|}\log_2\frac{|S_{\text{no}}|}{|S|} \\ &= -\frac{4}{10}\times\log_2\frac{4}{10} - \frac{6}{10}\times\log_2\frac{6}{10} \\ &= 0.9710 \\ \text{GainRatio} &= \frac{0.5709}{0.9710} \\ &= 0.5880 \end{aligned}$$

In a similar way we can compute the gain ratios Gain($S$, "aquatic"), Gain($S$, "aerial") and Gain($S$, "has legs").

## 8.9 Decision tree algorithms

### 8.9.1 Outline

---

**Decision tree algorithm: Outline**

1. Place the "best" feature (or, attribute) of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for a feature.
3. Repeat Step 1 and Step 2 on each subset until we find leaf nodes in all the branches of the tree.

---

### 8.9.2 Some well-known decision tree algorithms

1. ID3 (Iterative Dichotomiser 3) developed by Ross Quinlan
2. C4.5 developed by Ross Quinlan
3. C5.0 developed by Ross Quinlan
4. CART (Classification And Regression Trees)
5. 1R (One Rule) developed by Robert Holte in 1993.
6. RIPPER (Repeated Incremental Pruning to Produce Error Reduction) Introduced in 1995 by William W. Cohen.

As an example of decision tree algorithms, we discuss the details of the ID3 algorithm and illustrate it with an example.

## 8.10 The ID3 algorithm

Ross Quinlan, while working at University of Sydney, developed the ID3 (Iterative Dichotomiser 3)[5] algorithm and published it in 1975.

**Assumptions**

- The algorithm uses information gain to select the most useful attribute for classification.
- We assume that there are only two class labels, namely, "+" and "−". The examples with class labels "+" are called positive examples and others negative examples.

### 8.10.1 The algorithm

**Notations**

The following notations are used in the algorithm:

| | |
|---|---|
| $S$ | The set of examples |
| $C$ | The set of class labels |
| $F$ | The set of features |
| $A$ | An arbitrary feature (attribute) |
| Values($A$) | The set of values of the feature $A$ |
| $v$ | An arbitrary value of $A$ |
| $S_v$ | The set of examples with $A$ = $v$ |
| Root | The root node of a tree |

---

**Algorithm ID3($S$, $F$, $C$)**

1. Create a root node for the tree.
2. **if** (all examples in $S$ are positive) **then**
3. **return** single node tree Root with label "+"
4. **end if**
5. **if** (all examples are negative) **then**
6. **return** single node tree Root with label "–"
7. **end if**
8. **if** (number of features is 0) **then**
9. **return** single node tree Root with label equal to the most common class label.
10. **else**
11. Let $A$ be the feature in $F$ with the highest information gain.
12. Assign $A$ to the Root node in decision tree.
13. **for all** (values $v$ of $A$) **do**
14. Add a new tree branch below Root corresponding to $v$.
15. **if** ($S_v$ is empty) **then**
16. Below this branch add a leaf node with label equal to the most common class label in the set $S$.
17. **else**
18. Below this branch add the subtree formed by applying the same algorithm ID3 with the values ID3($S_v, C, F - \{A\}$).
19. **end if**
20. **end for**
21. **end if**

---

[5] *dichotomy*: A division into two parts or classifications especially when they are sharply distinguished or opposed

### 8.10.2 Example

**Problem**

Use ID3 algorithm to construct a decision tree for the data in Table 8.9.

| Day | outlook | temperature | humidity | wind | playtennis |
|---|---|---|---|---|---|
| D1 | sunny | hot | high | weak | no |
| D2 | sunny | hot | high | strong | no |
| D3 | overcast | hot | high | weak | yes |
| D4 | rain | mild | high | weak | yes |
| D5 | rain | cool | normal | weak | yes |
| D6 | rain | cool | normal | strong | no |
| D7 | overcast | cool | normal | strong | yes |
| D8 | sunny | mild | high | weak | no |
| D9 | sunny | cool | normal | weak | yes |
| D10 | rain | mild | normal | weak | yes |
| D11 | sunny | mild | normal | strong | yes |
| D12 | overcast | mild | high | strong | yes |
| D13 | overcast | hot | normal | weak | yes |
| D14 | rain | mild | high | strong | no |

Table 8.9: Training examples for the target concept "PlayTennis"

**Solution**

Note that, in the given data, there are four features but only two class labels (or, target variables), namely, "yes" and "no".

**Step 1**

We first create a root node for the tree (see Figure 8.7).

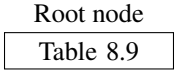


Figure 8.7: Root node of the decision tree for data in Table 8.9

**Step 2**

Note that not all examples are positive (class label "yes") and not all examples are negative (class label "no"). Also the number of features is not zero.

**Step 3**

We have to decide which feature is to be placed at the root node. For this, we have to calculate the information gains corresponding to each of the four features. The computations are shown below.

(i) Calculation of Entropy $(S)$

$$\begin{aligned}\text{Entropy}\,(S) &= -p_{\text{yes}}\log_2(p_{\text{yes}}) - p_{\text{no}}\log_2(p_{\text{no}}) \\ &= -(9/14)\times\log_2\big(9/14\big) - (5/14)\times\log_2\big(5/14\big) \\ &= 0.9405\end{aligned}$$

(ii) Calculation of Gain $(S, \text{outlook})$

The values of the attribute "outlook" are "sunny", " overcast" and "rain". We have to calculate Entropy $(S_v)$ for $v$ = sunny, $v$ = overcast and $v$ = rain.

$$\begin{aligned}
\text{Entropy}\,(S_{\text{sunny}}) &= -(3/5) \times \log_2\left(3/5\right) - (2/5) \times \log_2\left(2/5\right) \\
&= 0.9710 \\
\text{Entropy}\,(S_{\text{overcast}}) &= -(4/4) \times \log_2\left(4/4\right) \\
&= 0 \\
\text{Entropy}\,(S_{\text{rain}}) &= -(3/5) \times \log_2\left(3/5\right) - (2/5) \times \log_2\left(2/5\right) \\
&= 0.9710 \\
\text{Gain}\,(S, \text{ outlook}) &= \text{Entropy}\,(S) - \frac{|S_{\text{sunny}}|}{|S|} \times \text{Entropy}\,(S_{\text{sunny}}) \\
&\quad - \frac{|S_{\text{overcast}}|}{|S|} \times \text{Entropy}\,(S_{\text{overcast}}) \\
&\quad - \frac{|S_{\text{rain}}|}{|S|} \times \text{Entropy}\,(S_{\text{rain}}) \\
&= 0.9405 - (5/14) \times 0.9710 - (4/14) \times 0 \\
&\quad - (5/14) \times 0.9710 \\
&= 0.2469
\end{aligned}$$

(iii) Calculation of Gain $(S, \text{temperature})$

The values of the attribute "temperature" are "hot", "mild" and "cool". We have to calculate Entropy $(S_v)$ for $v$ = hot, $v$ = mild and $v$ = cool.

$$\begin{aligned}
\text{Entropy}\,(S_{\text{hot}}) &= -(2/4) \times \log_2\left(2/4\right) - (2/4) \times \log_2\left(2/4\right) \\
&= 1.0000 \\
\text{Entropy}\,(S_{\text{mild}}) &= -(4/6) \times \log_2\left(4/6\right) - (2/6) \times \log_2\left(2/6\right) \\
&= 0.9186 \\
\text{Entropy}\,(S_{\text{cool}}) &= -(3/4) \times \log_2\left(3/4\right) - (1/4) \times \log_2\left(1/4\right) \\
&= 0.8113 \\
\text{Gain}\,(S, \text{ temperature}) &= \text{Entropy}\,(S) - \frac{|S_{\text{hot}}|}{|S|} \times \text{Entropy}\,(S_{\text{hot}}) \\
&\quad - \frac{|S_{\text{mild}}|}{|S|} \times \text{Entropy}\,(S_{\text{mild}}) \\
&\quad - \frac{|S_{\text{cool}}|}{|S|} \times \text{Entropy}\,(S_{\text{cool}}) \\
&= 0.9405 - (4/14) \times 1.0000 - (6/14) \times 0.9186 \\
&\quad - (4/14) \times 0.8113 \\
&= 0.0293
\end{aligned}$$

(iv) Calculation of Gain $(S, \text{humidity})$ and Gain $(S, \text{wind})$

The following information gains can be calculated in a similar way:

$$\begin{aligned}
\text{Gain}\,(S, \text{ humidity}) &= 0.151 \\
\text{Gain}\,(S, \text{ wind}) &= 0.048
\end{aligned}$$

**Step 4**

We find the highest information gain whic is the maximum among Gain$(S, \text{outlook})$, Gain$(S, \text{temperature})$, Gain$(S, \text{humidity})$ and Gain$(S, \text{wind})$. Therefore, we have:

$$\begin{aligned}\text{highest information gain} &= \max\{0.2469, 0.0293, 0.151, 0.048\} \\ &= 0.2469\end{aligned}$$

This corresponds to the feature "outlook". Therefore, we place "outlook" at the root node. We now split the root node in Figure 8.7 into three branches according to the values of the feature "outlook" as in Figure 8.8.

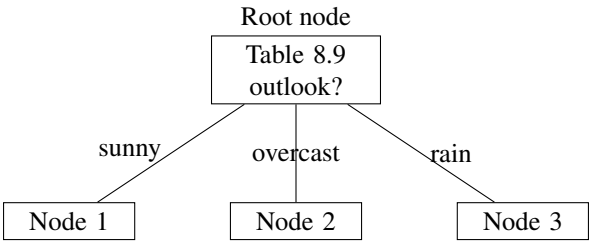


Figure 8.8: Decision tree for data in Table 8.9, after selecting the branching feature at root node

**Step 5**

Let $S^{(1)} = S_{\text{outlook=sunny}}$. We have $\left|S^{(1)}\right| = 5$. The examples in $S^{(1)}$are shown in Table 8.10.

| Day | outlook | temperature | humidity | wind | playtennis |
|---|---|---|---|---|---|
| D1 | sunny | hot | high | weak | no |
| D2 | sunny | hot | high | strong | no |
| D8 | sunny | mild | high | weak | no |
| D9 | sunny | cool | normal | weak | yes |
| D11 | sunny | mild | normal | strong | yes |

Table 8.10: Training examples with outlook = "sunny"

$$\begin{aligned}\text{Gain}(S^{(1)}, \text{temp}) = \text{Entropy}(S^{(1)}) &- \frac{\left|S^{(1)}_{\text{temp = hot}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{temp = hot}}) \\ &- \frac{\left|S^{(1)}_{\text{temp = mild}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{temp = mild}}) \\ &- \frac{\left|S^{(1)}_{\text{temp = cool}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{temp = cool}}) \\ = &\left[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\right] \\ &- (2/5) \times \left[-(2/2)\log(2/2))\right] \\ &- (2/5) \times \left[-(1/2)\log(1/2) - (1/2)\log_2(1/2)\right] \\ &- (1/5) \times \left[-(1/1)\log(1/1)\right] \\ = &\ 0.5709\end{aligned}$$

$$\begin{aligned}
\text{Gain}(S^{(1)}, \text{hum}) &= \text{Entropy}(S^{(1)}) - \frac{\left|S^{(1)}_{\text{hum = high}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{hum = high}}) \\
&\qquad - \frac{\left|S^{(1)}_{\text{hum = normal}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{hum = normal}}) \\
&= \left[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\right] \\
&\quad - (3/5) \times \left[-(3/3)\log(3/3))\right] \\
&\quad - (2/5) \times \left[-(2/2)\log(2/2)\right] \\
&= 0.9709 \\
\text{Gain}(S^{(1)}, \text{wind}) &= \text{Entropy}(S^{(1)}) - \frac{\left|S^{(1)}_{\text{wind = weak}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{wind = weak}}) \\
&\qquad - \frac{\left|S^{(1)}_{\text{wind = strong}}\right|}{\left|S^{(1)}\right|} \times \text{Entropy}(S^{(1)}_{\text{wind = strong}}) \\
&= \left[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\right] \\
&\quad - (3/5) \times \left[-(2/3)\log(2/3) - (1/3)\log_2(1/3))\right] \\
&\quad - (2/5) \times \left[-(1/2)\log(1/2) - (1/2)\log(1/2)\right] \\
&= 0.0110
\end{aligned}$$

The maximum of Gain$(S^{(1)}, \text{temp})$, Gain$(S^{(1)}, \text{hum})$ and Gain$(S^{(1)}, \text{wind})$ is Gain$(S^{(1)}, \text{hum})$. Hence we place "humidity" at Node 1 and split this node into two branches according to the values of the feature "humidity" to get the tree in Figure 8.9.

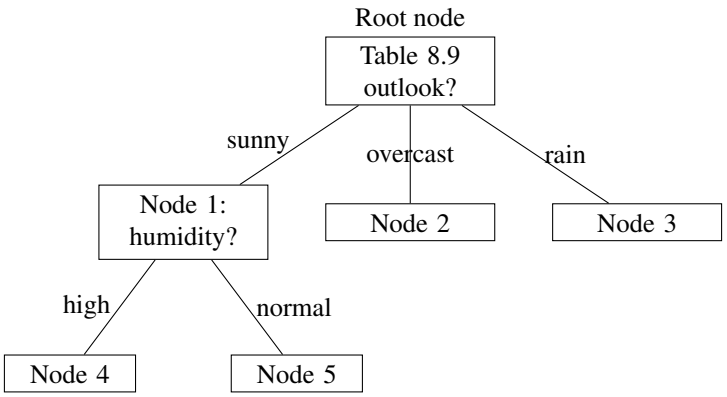


Figure 8.9: Decision tree for data in Table 8.9, after selecting the branching feature at Node 1

**Step 6**

It can be seen that all the examples in the the data set corresponding to Node 4 in Figure 8.9 have the same class label "no" and all the examples corresponding to Node 5 have the same class label "yes". So we represent Node 4 as a leaf node with value "no" and Node 5 as a leaf node with value "yes". Similarly, all the examples corresponding to Node 2 have the same class label "yes". So we convert Node 2 as a leaf node with value " yes. Finally, let $S^{(2)} = S_{\text{outlook = rain}}$. The highest information gain for this data set is Gain$(S^{(2)}, \text{humidity})$. The branches resulting from splitting this node corresponding to the values "high" and "normal" of "humidity" lead to leaf nodes with class labels "no" and "yes". With these changes, we get the tree in Figure 8.10.

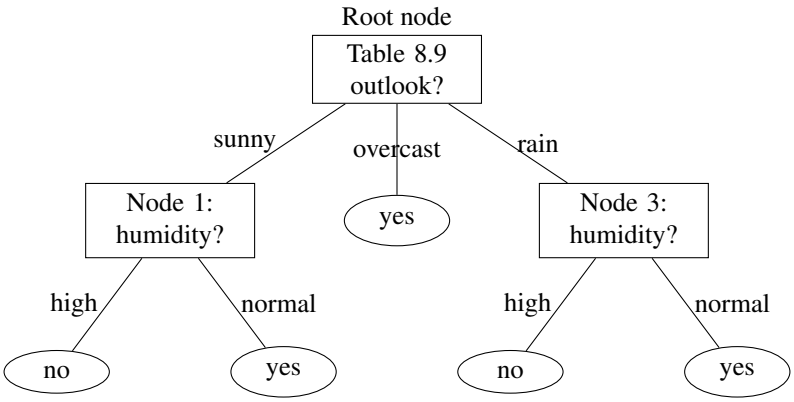


Figure 8.10: Decision tree for data in Table 8.9

## 8.11 Regression trees

A *regression problem* is the problem of determining a relation between one or more independent variables and an output variable which is a real continuous variable and then using the relation to predict the values of the dependent variables. Regression problems are in general related to prediction of numerical values of variables. Trees can also be used to make such predictions. A tree used for making predictions of numerical variables is called a *prediction tree* or a *regression tree*.

### 8.11.1 Example

Using the data in Table 8.11, construct a tree to predict the values of $y$.

| $x_1$ | 1 | 3 | 4 | 6 | 10 | 15 | 2 | 7 | 16 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $x_2$ | 12 | 23 | 21 | 10 | 27 | 23 | 35 | 12 | 27 | 17 |
| $y$ | 10.1 | 15.3 | 11.5 | 13.9 | 17.8 | 23.1 | 12.7 | 43.0 | 17.6 | 14.9 |

Table 8.11: Data for regression tree

**Solution**

We shall construct a *raw decision tree* (a tree constructed without using any standard algorithm) to predict the value of $y$ corresponding to any untabulated values of $x_1$ and $x_2$.

Step 1. We arbitrarily split the values of $x_1$ into two sets: One set defined by $x_1 < 6$ and the other set defined by $x_1 \geq 6$. This splits the data into two parts. This yields the tree in Figure **??**.

| $x_1$ | 1 | 3 | 4 | 2 | 0 |
|---|---|---|---|---|---|
| $x_2$ | 12 | 23 | 21 | 35 | 17 |
| $y$ | 10.1 | 15.3 | 11.5 | 12.7 | 14.9 |

Table 8.12: Data for regression tree

Step 2. In Figure 8.12, consider the node specified by Table 8.12. We arbitrarily split the values of $x_2$ into two sets: one specified by $x_2 < 21$ and one specified by $x_2 \geq 21$. Similarly, the node specified by Table 8.13, we split the values of $x_2$ into sets: one specified by $x_2 < 23$

| $x_1$ | 6 | 10 | 15 | 7 | 16 |
|---|---|---|---|---|---|
| $x_2$ | 10 | 27 | 23 | 12 | 27 |
| $y$ | 13.9 | 17.8 | 23.1 | 43.0 | 17.6 |

Table 8.13: Data for regression tree

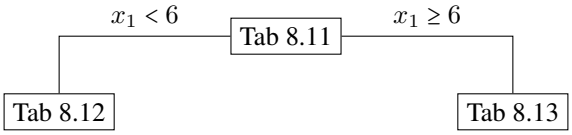


Figure 8.11: Part of a regression tree for Table 8.11

and one specified by $x_2 \geq 23$. The split data are given in Table 8.14(a) - (d). This gives us the tree in Figure 8.12.

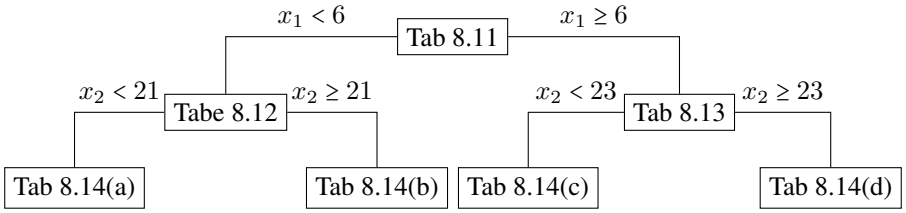


Figure 8.12: Part of regression tree for Table 8.11

Step 3. We next make the nodes specified by Table 8.14(a), ..., Tab 8.14(d) into leaf nodes. In each of these leaf nodes, we write the average of the values in the corresponding table (this is a standard procedure). For, example, at Table 8.14(a), we write $\frac{1}{2}(10.1 + 14.9) = 12.5$. Then we get Figure 8.13.

| $x_1$ | 1 | 0 |
|---|---|---|
| $x_2$ | 12 | 17 |
| $y$ | 10.1 | 14.9 |

(a)

| $x_1$ | 3 | 4 | 2 |
|---|---|---|---|
| $x_2$ | 23 | 21 | 35 |
| $y$ | 15.3 | 11.5 | 12.7 |

(b)

| $x_1$ | 6 | 7 |
|---|---|---|
| $x_2$ | 10 | 12 |
| $y$ | 13.9 | 43.0 |

(c)

| $x_1$ | 10 | 15 | 16 |
|---|---|---|---|
| $x_2$ | 27 | 23 | 27 |
| $y$ | 17.8 | 23.1 | 17.6 |

(d)

Table 8.14: Data for regression tree

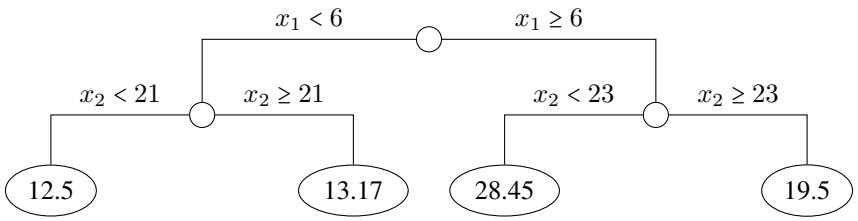


Figure 8.13: A regression tree for Table 8.11

Step 4. Figure 8.13 is the final raw regression tree for predicting the values of $y$ based on the data in Table 8.11.

### 8.11.2 An algorithm for constructing regression trees

Starting with a learning sample, three elements are necessary to determine a regression tree:

1. A way to select a split at every intermediate node
2. A rule for determining when a node is terminal
3. A rule for assigning a value for the output variable to every terminal node

**Notations**

| | | |
|---|---|---|
| $x_1, x_2, \ldots, x_n$ | : | The input variables |
| $N$ | : | Number of samples in the data set |
| $y_1, y_2, \ldots, y_N$ | : | The values of the output variables |
| $T$ | : | A tree |
| $c$ | : | A leaf of $T$ |
| $n_c$ | : | Number of data elements in the leaf $c$ |
| $C$ | : | The set of indices of data elements which are in the leaf $c$ |
| $m_c$ | : | The mean of the values of $y$ which are in the leaf $c$ |
| $S_T$ | : | Sum of squares of errors in $T$ |

We have

$$m_c = \frac{1}{n_c} \sum_{i \in C} y_i$$

$$S_T = \sum_{c \in \text{leaves}(T)} \sum_{i \in C} (y_i - m_c)^2$$

**Algorithm**

**Step 1.** Start with a single node containing all data points. Calculate $m_c$ and $S_T$.

**Step 1.** If all the points in the node have the same value for all the independent variables, stop.

**Step 1.** Otherwise, search over all binary splits of all variables for the one which will reduce $S_T$ as much as possible.

(a) If the largest decrease in $S_T$ would be less than some threshold $\delta$, or one of the resulting nodes would contain less than $q$ points, stop and if $c$ is a node where we have stopped, then assign the value $m_c$ to the node.

(b) Otherwise, take that split, creating two new nodes.

**Step 1.** In each new node, go back to Step 1.

**Remarks**

1. We have seen entropy and information defined for discrete variables. We can define them for continuous variables also. But in the case of regression trees, it is more common to use the sum of squares. The above algorithm is based on sum of squares of errors.

2. The CART algorithm mentioned below searches every distinct value of every predictor variable to find the predictor variable and split value which will reduce $S_T$ as much as possible.

3. In the above algorithm, we have given the simplest criteria for stopping growing of trees. More sophisticated criteria which produce much less error have been developed.

### 8.11.3 Example

Consider the data given in Table 8.11.

1. Computation of $S_T$ for the entire data set. Initially, there is only one node. So, we have:

$$\begin{aligned} m_c &= \frac{1}{n_c}\sum_{c\in C} y_i \\ &= \frac{1}{10}(10.1 + 15.3 + \cdots + 14.9) \\ &= 17.99 \\ S_T &= \sum_{c\in \text{leaves}(T)} \sum_{i\in C} (y_i - m_c)^2 \\ &= (10.1 - 17.99)^2 + (15.3 - 17.99)^2 + \cdots + (14.9 - 17.99)^2 \\ &= 817.669 \end{aligned}$$

2. As suggested in the remarks above, we have to search every distinct value of $x_1$ and $x_2$ to find the predictor variable and split value which will reduce $S_T$ as much as possible.

3. Let us consider the value 6 of $x_1$. This splits the data set into two parts $c_1$ and $c_2$. Let $c_1$ be the part defined by $x_1 < 6$ and $c_2$ the part defined by $x_1 \geq 6$. $S_1$ is given in Table 8.12 and $S_2$ by Table 8.13.Now

$$\text{leaves}(T) = \{c_1, c_2\}.$$

Let $T_1$ be the tree corresponding to this partition. Then

$$\begin{aligned} S_{T_1} &= \sum_{c\in \text{leaves}(T_1)} \sum_{i\in C} (y_i - m_c)^2 \\ &= \sum_{i\in C_1} (y_i - m_{c_1})^2 + \sum_{i\in C_2} (y_i - m_{c_2})^2 \\ m_{c_1} &= \frac{1}{n_{c_1}} \sum_{i\in C_1} y_i \\ &= \frac{1}{5}(10.1 + 15.3 + 11.5 + 12.7 + 14.9) \\ &= 12.9 \end{aligned}$$

$$
\begin{aligned}
m_{c_2} &= \frac{1}{n_{c_2}} \sum_{i \in C_2} y_i \\
&= \frac{1}{5}(13.9 + 17.8 + 23.1 + 43.0 + 17.6) \\
&= 23.08 \\
S_{T_1} &= [(10.1 - 12.9)^2 + \cdots + (14.9 - 12.9)^2] + \\
&\quad [(13.9 - 23.08)^2 + \cdots + (17.6 - 23.08)^2] \\
&= 558.588
\end{aligned}
$$

The reduction in sum of squares of errors is

$$S_T - S_{T_1} = 817.669 - 558.588 = 259.081.$$

4. In this way, we have compute the reduction in the sum of squares of errors corresponding to all other values of $x_1$ and each of the values of $x_2$ and choose the one for which the reduction is maximum.

5. The process has be continued. (Software package may be required to complete the problem.)

## 8.12 CART algorithm

We have seen how decision trees can be used to create a model that predicts the value of a target (or dependent variable) based on the values of several input or independent variables.

The CART, or *Classification And Regression Trees* methodology, was introduced in 1984 by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone as an umbrella term to refer to the following types of decision trees:

- *Classification trees* where the target variable is categorical and the tree is used to identify the "class" within which a target variable would likely fall into.
- *Regression trees* where the target variable is continuous and tree is used to predict it's value.

The main elements of CART are:

- Rules for splitting data at a node based on the value of one variable
- Stopping rules for deciding when a branch is terminal and can be split no more
- A prediction for the target variable in each terminal node

## 8.13 Other decision tree algorithms

### 8.13.1 The C4.5 algorithm

The C4.5 algorithm is an algorithm developed by Ross Quinlan as an improvement of the ID3 algorithm. The following are some of the improvements incorporated in C4.5.

- Handling both continuous and discrete attributes
- Handling training data with missing attribute values
- Handling attributes with differing costs
- Pruning trees after creation

### 8.13.2 The C5.0 algorithm

The C5.0 algorithm represents a further improvement on the C4.5 algorithm. This was also developed by Ross Quinlan.

- Speed - C5.0 is significantly faster than C4.5.
- Memory usage - C5.0 is more memory efficient than C4.5.
- C5.0 gets similar results to C4.5 with considerably smaller decision trees.

The C5.0 algorithm is one of the most well-known implementations of the the decision tree algorithm. The source code for a single-threaded version of the algorithm is publicly available, and it has been incorporated into programs such as R. The C5.0 algorithm has become the industry standard to produce decision trees.

## 8.14 Issues in decision tree learning

In thie next feww sections, we discuss some of the practical issues in learning decision trees.

## 8.15 Avoiding overfitting of data

When we construct a decision tree, the various branches are grown (that is, sub-branches are constructed) just deeply enough to perfectly classify the training examples. This leads to difficulties when there is noise in the data or when the number of training examples are too small. In these cases the algorithm can produce trees that overfit the training examples.

#### Definition

We say that a hypothesis *overfits* the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances, including instances beyond the training set.

#### Impact of overfitting

Figure 8.14 illustrates the impact of overfitting in a typical decision tree learning. From the figure, we can see that the accuracy of the tree over training examples increases monotonically whereas the accuracy measured over independent test samples first increases then decreases.

### 8.15.1 Approaches to avoiding overfitting

The main approach to avoid overfitting is *pruning*. Pruning is a technique that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

- We may apply pruning earlier, that is, before it reaches the point where it perfectly classifies the training data.
- We may allow the tree to overfit the data, and then post-prune the true.

Now there is the problem of what criterion is to be used to determine the correct final tree size. One commonly used criterion is to use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.

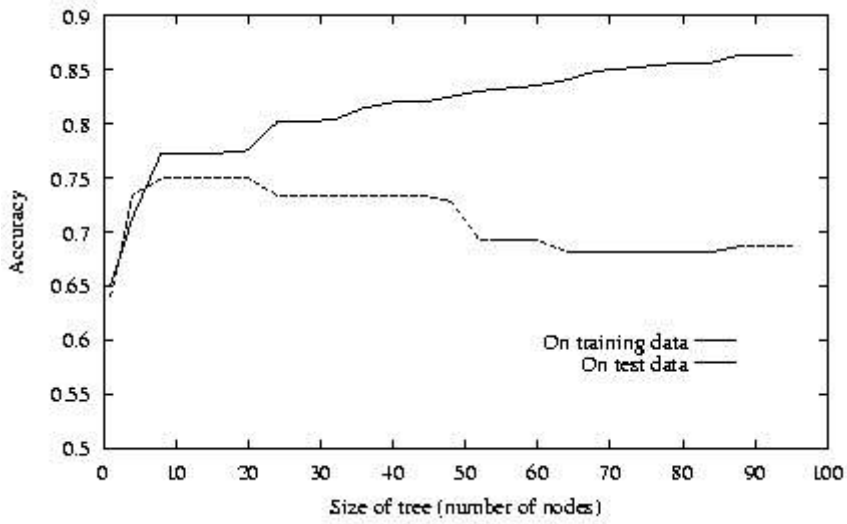


Figure 8.14: Impact of overfitting in decision tree learning

| Case | Temperature | Headache | Nausea | Decision (Flue) |
|---|---|---|---|---|
| 1 | high | ? | no | yes |
| 2 | very high | yes | no | yes |
| 3 | ? | no | no | no |
| 4 | high | yes | yes | yes |
| 5 | high | ? | yes | no |
| 6 | normal | yes | no | no |
| 7 | normal | no | yes | no |
| 8 | ? | yes | ? | yes |

Table 8.15: A dataset with missing attribute values

### 8.15.2 Reduced error pruning

In *reduced-error pruning*, we consider each of the decision tress to be a candidate for pruning. Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated to that node. Nodes are removed only if the resulting pruned tree performs no worse than the original over validation set. Nodes are pruned iteratively, always choosing the node whose removal most increases the accuracy over the validation set. Pruning of nodes is continued until further pruning decreases the accuracy over the validation set.

## 8.16 Problem of missing attributes

Table 8.15 shows a dataset with missing attribute values. the missing values are indicated by "?"s.

The following are some of the methods used to handle the problem of missing attributes.

- Deleting cases with missing attribute values
- Replacing a missing attribute value by the most common value of that attribute

- Assigning all possible values to the missing attribute value
- Replacing a missing attribute value by the mean for numerical attributes
- Assigning to a missing attribute value the corresponding value taken from the closest $t$ cases, or replacing a missing attribute value by a new value

---

## 8.17 Sample questions

**(a) Short answer questions**

1. Explain the concept of a decision tree with an example.
2. What are the different types of decision trees?
3. Define the entropy of a dataset.
4. Write a formula to compute the entropy of a two-class dataset.
5. Define information gain and Gini index.
6. Give the names of five different decision-tree algorithms.
7. Can decision tree be used for regression? If yes, explain how. If no, explain why.
8. What is the difference between classification and regression trees?

**(b) Long answer questions**

1. Explain classification tree using an example.
2. Consider the following set of training examples:

| Instance | Classification | $a_1$ | $a_2$ |
|---|---|---|---|
| 1 | + | T | T |
| 2 | + | T | T |
| 3 | − | T | F |
| 4 | + | F | F |
| 5 | − | F | T |
| 6 | − | F | T |

   (a) What is the entropy of this collection of training examples with respect to the target function "classification"?

   (b) What is the information gain of $a_2$ relative to these training examples?

3. Explain the ID3 algorithm for learning decision trees.
4. Explain CART algorithm.
5. What are issues in decision tree learning? How are they overcome?
6. Describe an algorithm to construct regression trees.
7. What do you mean by information gain and entropy? How is it used to build the decision trees? Illustrate using an example.
8. Use ID3 algorithm to construct a decision tree for the data in the following table.

| Instance no. | Class label | $x_1$ | $x_2$ |
|---|---|---|---|
| 1 | 1 | T | T |
| 2 | 1 | T | T |
| 3 | 0 | T | F |
| 4 | 1 | F | F |
| 5 | 0 | F | T |
| 6 | 0 | F | T |

9. Use ID3 algorithm to construct a decision tree for the data in the following table.

| Gender | Car ownership | Travel cost | Income level | Class (mode of transportation) |
|---|---|---|---|---|
| Male | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Female | 1 | Cheap | Medium | Train |
| Female | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Male | 0 | Standard | Medium | Train |
| Female | 1 | Standard | Medium | Train |
| Female | 1 | Expensive | High | Car |
| Male | 2 | Expensive | Medium | Car |
| Female | 2 | Expensive | High | Car |

10. Use ID3 algorithm to construct a decision tree for the data in the following table.

| Age | Competition | Type | Class (profit) |
|---|---|---|---|
| Old | Yes | Software | Down |
| Old | No | Software | Down |
| Old | No | Hardware | Down |
| Mid | Yes | Software | Down |
| Mid | Yes | Hardware | Down |
| Mid | No | Hardware | Up |
| Mid | No | Software | Up |
| New | Yes | Software | Up |
| New | No | Hardware | Up |
| New | No | Software | Up |

11. Construct a decision tree for the following data.

| Class label (risk) | Collateral | Income | Debt | Credit history |
|---|---|---|---|---|
| high | none | low | high | bad |
| high | none | middle | high | unknown |
| moderate | none | middle | low | unknown |
| high | none | low | low | unknown |
| low | none | upper | low | unknown |
| low | adequate | upper | low | unknown |
| high | none | low | low | bad |
| moderate | adequate | upper | low | bad |
| low | none | upper | low | good |
| low | adequate | upper | high | good |
| high | none | low | high | good |
| moderate | none | middle | high | good |
| low | none | upper | high | good |
| high | none | middle | high | bad |