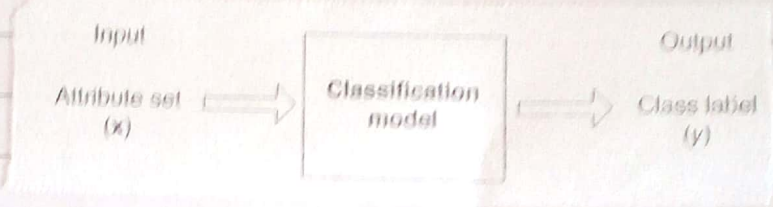
III CLASSIFICATION

Basic Concepts, Decision Trees, and Model Evaluation

3.1 Preliminaries

- Classification, which is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications.
  Eg: • Detecting spam emails based on message header & content
  • Categorizing cells as malignant (or) benign based on the results of MRI scans.



- The input data for a classification task is a collection of records.
- Each record, also known as an instance (or) example, is characterized by a tuple (x, y), where x is the attribute set and y is a special attribute, designated as the class label, also known as category / target attribute).
- Table 3.1 in the next page shows a sample data set used for classifying vertebrates into one of the following categories: mammal, bird, fish, reptile and amphibian.
- The attribute set includes properties of a vertebrate such as its body temperature, skin cover, gives birth, aquatic creature, has legs, aerial creature, hibernates.
- Most of the attributes are discrete but some are continuous.
- The class label is always discrete attribute for classification.

- Regression which is a predective modelling task has class label attribute as continuous.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

Classification : It is the task of learning a target function f that maps each attribute set x to one of the predefined class labels y.

Classification model: The target function is also known informally as a classification model. A classification model is useful for both Descriptive and Predictive Modelling.

3.1.1 Descriptive Modeling

- A classification model can serve as an explanatory tool to distinguish between objects of different classes.

Eg: For biologists and others the descriptive model can summarize the above label 3.1 data to explain what features define a vertebrate as a mammal, reptile, bird, fish or amph.

3.1.2. Predictive Modeling

- A classification model can also be used to predict the class label of unknown records. As shown in figure 3.3, a classification model can be treated as a black box that automatically assigns a class label when presented with the attribute set of an unknown record.

Eg: Suppose we are given the following characteristics of a creature known as a gila monster:

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class Label
gila monster	cold-blooded	scales	no	no	no	yes	yes	?

- We can use a classification model built from the data set shown in table 3.1 to determine the class to which the creature belongs.

- Classification techniques are mostly for predicting (or) describing data sets with binary (or) nominal categories.
- They are less effective for ordinal attributes.
- Sub class - super class relationships among categories are ignored.

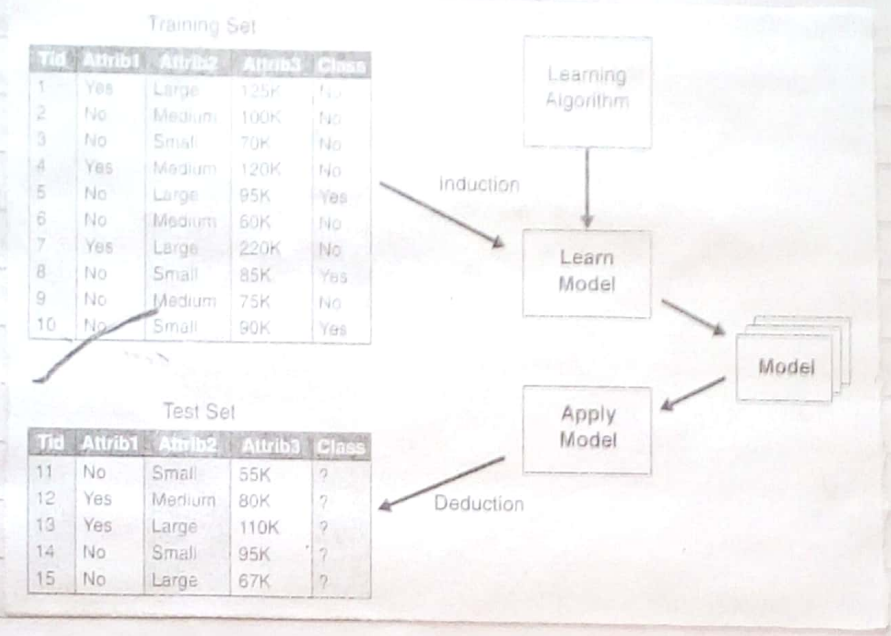
3.2 General Approach to Solving a Classification Problem

- A classification technique / classifier is a systematic approach to build classification models from an input data set.

Eg: Decision tree classifiers, rule-based classifiers, neural networks, naive Bayes classifiers and Support vector machines.

- Every technique uses a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data.
- The model generated should fit the input data well and correctly predict the class labels of records it has never seen before.

- The following figure 3.2 shows a general approach for solving classification problems.



- First, a training set consisting of records whose class labels are known must be provided.
- The training set is used to build a classification model, which is subsequently applied to the test set, which consists of records with unknown class labels.
- Model performance evaluation is based on the counts of test records predicted correctly and incorrectly. These counts are tabulated in the table 3.2 known as confusion matrix.

Table 4.2. Confusion matrix for a 2-class problem.

		Predicted Class	
		$Class = 1$	$Class = 0$
Actual Class	$Class = 1$	f_{11}	f_{10}
	$Class = 0$	f_{01}	f_{00}

- The above table 3.2 depicts the confusion matrix for a binary classification problem
- Each entry f_{ij} in the table denotes the number of records

from class i predicted to be of class j.

- For instance, f_{01} is the number of records from class 0 incorrectly predicted as class 1.
- Correct predictions : $f_{11} + f_{00}$, Incorrect predictions : $f_{10} + f_{01}$
- Confusion matrix gives the information of the model performance, but if the performance is a single number then it is more convenient to compare with different models.
- This can be done using performance metric such as accuracy,

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total no. of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

- Model performance can also been expressed in terms on error rate,

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total no. of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

- Many classification algorithms seek models that attain the highest accuracy (or) equivalently, the lowest error rate when applied to the test set.

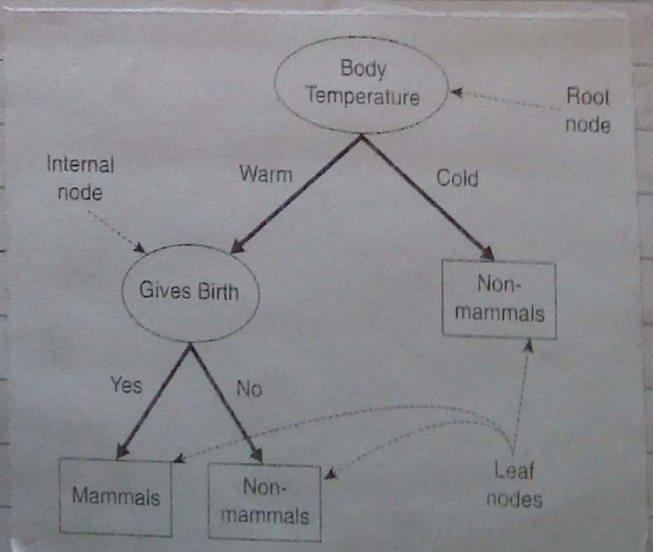
3.3 Decision Tree Induction

- Decision tree classifier is widely used classification technique.

3.3.1 How a Decision Tree Works?

- To illustrate the work of decision tree for classification consider the data of the table 3.1, instead of classifying the vertebrates into five distinct groups, we assign them to mammals and non-mammals.

- Suppose a new species is discovered by scientists. How can we tell whether it is mammal (or) non-mammal?
  - One approach is to put a series of questions about the characteristics of the species. The first question is whether the species is cold- (or) warm - blooded. If it is cold-blooded then it is definitely not a mammal. Otherwise, it is either a bird (or) a mammal.
  - In the latter case, we need to ask a follow-up question: Do the females of the species give birth? Those that do give birth are definitely mammals, while those that do not are non-mammals.
- The above example illustrates how we can solve a classification problem by asking a series of questions about the attributes of test record and each time a answer is received, follow-up question is also asked until we reach to a conclusion about the class label of the test record.
- These questions and their possible answers are organized in the form of decision tree, which is of nodes and directed edges.
- Figure 3.3 shows the decision tree for the mammal classification problem.



- The tree has three types of nodes:

a) root node : has no incoming edges and zero (or) more outgoing edges

b) internal node : which has exactly one incoming edge and two / more outgoing edges

c) Leaf (or) terminal node : which has exactly one incoming edge and no outgoing edges.

- In a decision tree, each leaf node is assigned a class label.
- The non-terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records, that have different characteristics.
- For example, root node shown in figure 3.3 uses the attribute Body Temperature to separate warm- and cold-blooded vertebrates. As all the cold-blooded are non-mammals, the right child of the root node is created with leaf node labelled non-mammals. If vertebrate is warm-blooded, a subsequent attribute, Gives birth is used to differentiate mammals from other warm-blooded creature mostly are birds.

Name	Body temperature	Gives Birth	...	Class
Flamingo	Warm	No	...	?

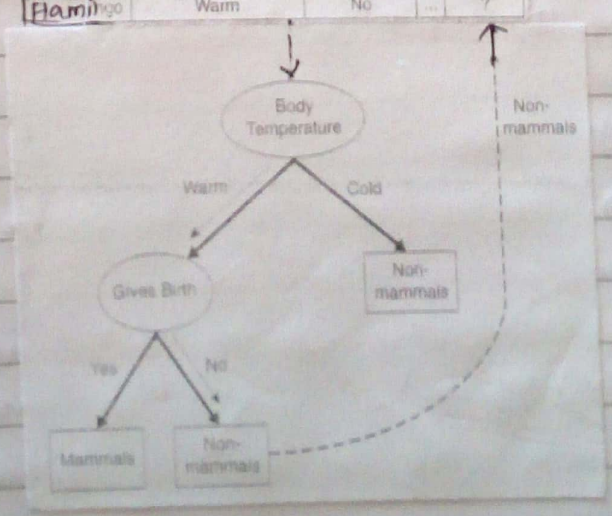


Figure 3.4

- Classifying a test record is straightforward once a decision tree has been constructed.
- Starting from the root node, we follow the appropriate branch based on the outcome of the test which leads to either to another internal node (or) to a leaf node.
- The class label associated with the leaf node is then assigned to the record.
- Figure 3.4 traces the path in the decision tree that is used to predict the class label of a flamingo. The path terminates at a leaf node labeled non-mammals

3.3.2 How to Build a Decision Tree

- There are exponentially many decision trees that can be constructed from a given set of attributes.
- Finding optimal tree is computationally infeasible because of the exponential size of the search space.
- Nevertheless, efficient algorithm can be developed (used) to induce an accurate decision tree in a reasonable time employing a greedy strategy by making a series of locally optimal decisions to where to partitioning of the data.
- Hunt's algorithm is one such algorithm which is basis of many existing decision tree induction algorithms, including ID3, C4.5 and CART.

Hunt's Algorithm

- A decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets.
- Let D_t be set of training records that are associated with node t and $y = \{y_1, y_2, \ldots y_c\}$ be the class labels. The following is a recursive definition of Hunt's algorithm.

Step 1: If all the records in D_t belong to the same class y_t, then t is a leaf node labeled as y_t.

Step 2: If D_t contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node

Working Procedure : Illustration

- Consider the problem of predicting whether a loan applicant will repay her loan obligations (or) become delinquent, subsequently defaulting on her loan.
- A training set for this problem can be constructed by examining the records of previous borrowers. In the example shown in the following figure 3.5, each record contains the personal information of a borrower along with a class label indicating whether the borrower has defaulted on loan payments.

Figure 3.5

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

binary, categorical, continuous, class

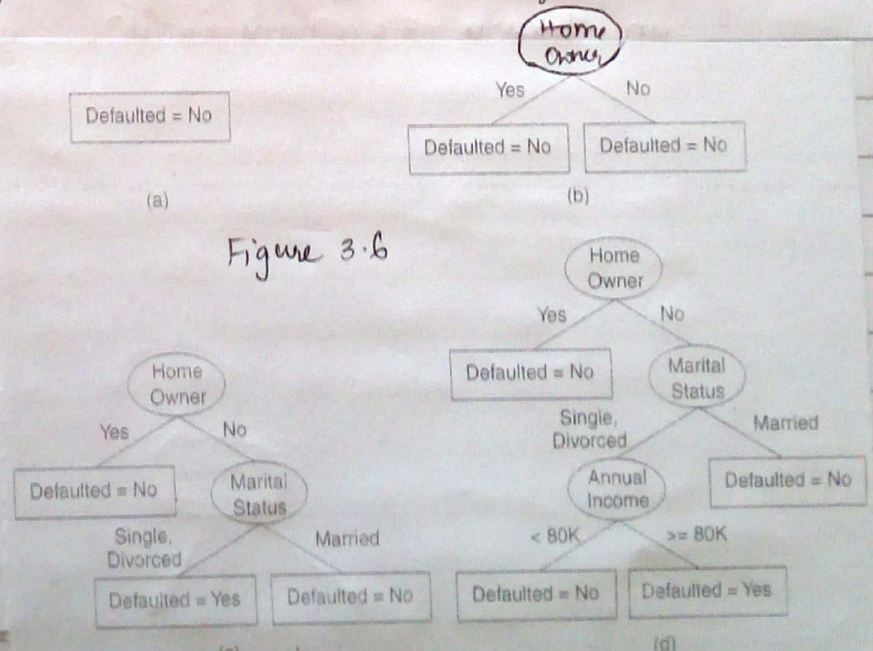


Figure 3.6

- The initial tree for the classification problem consists a single node with class label Defaulted = No (shown in figure 3.6(a)) which means that most of the borrowers successfully repaid their loans.
- However, the tree needs to be refined since the root node contains records from both classes. The records are subsequently divided into smaller subsets based on the outcomes of the Home Owner test condition, as shown in figure 3.6(b).
- The justification for choosing this attribute test condition is discussed later, we assume that this is best criterion for splitting the data at this point.
- Hunt's algorithm is then applied recursively to each child of the root node. From the training set given in fig 3.5, notice that all borrowers who are home owners successfully repaid their loans.
- The left child of the root is therefore a leaf node labelled Defaulted = No (figure 3.6(b)).
- For the right child, we need to continue applying the recursive step of Hunt's algorithm until all the records belong to the same class.
- The trees resulting from each recursive step are shown in figures 3.6(c) and 3.6(d).

- Hunt's algorithm will work if every combination of attribute values is present in the training data and each combination has a unique class label. These assumptions are too stringent for use in most practical situations. Additional conditions are needed to handle the following cases:

1. It is possible for some of the child nodes created in step 2 to be empty ie, there are no records associated with these nodes. In this case the node is declared a leaf node

with the same class label as the majority class of training records associated with its parent node.

2. In step 2, if all the records associated with D_t have identical attribute values (except for the class label), then it is not possible to split these records any further. In this case, the node is declared a leaf node with the same class label as the majority class of training records associated with this node.

Design Issues of Decision Tree Induction

- A learning algorithm for inducting decision trees must address the following two issues:

1. How should the training records be split?
   - Each recursive step of the tree-growing process must select an attribute test condition to divide the records into smaller subsets. For this, the algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.

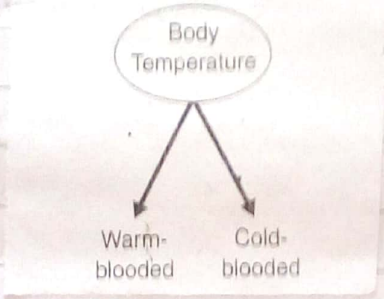
2. How should the splitting procedure stop?
   - A stopping condition is needed to terminate the tree-growing process. One possible approach is to continue expanding a node until either all the records belong to the same class (or) all the records have identical attribute values. Here both conditions stop the decision tree induction, other criteria can be imposed to allow the tree growing procedure to terminate earlier.

3.3.3 Methods for Expressing Attribute Test Conditions

- Decision tree induction algorithms must provide a method for expressing an attribute test condition and its corresponding outcomes for different attribute types.

1. Binary Attributes

- Figure 3.7 shows the test condition for a binary attribute generates two potential outcomes



2. Nominal Attributes

- A nominal attribute can have many values, its test condition can be expressed in two ways, shown in figure 3.8 (Multiway split)
- For a multiway split shown in the figure 3.8(a), the number of outcomes depends on the number of distinct values for the corresponding attribute Eg: marital status has single, married, or divorced values.
- CART algorithm produce only binary splits by considering all $2^{k-1}-1$ ways of creating a binary partition of k attribute values.
- Figure 3.8(b) shows three diff ways of grouping into 2 subsets

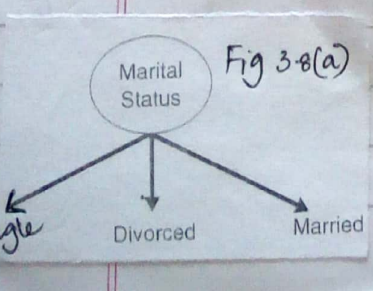


Fig 3.8(a)

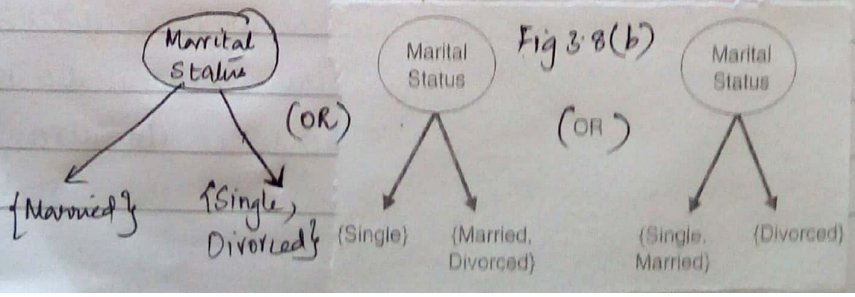
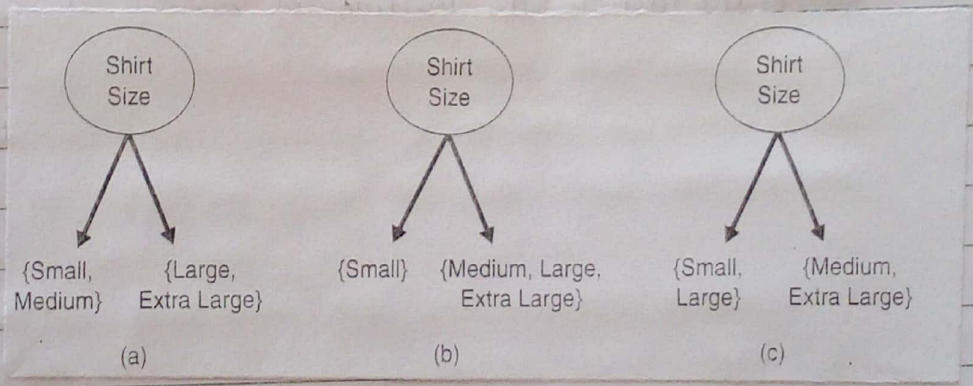


Fig 3.8(b)

3. Ordinal Attributes

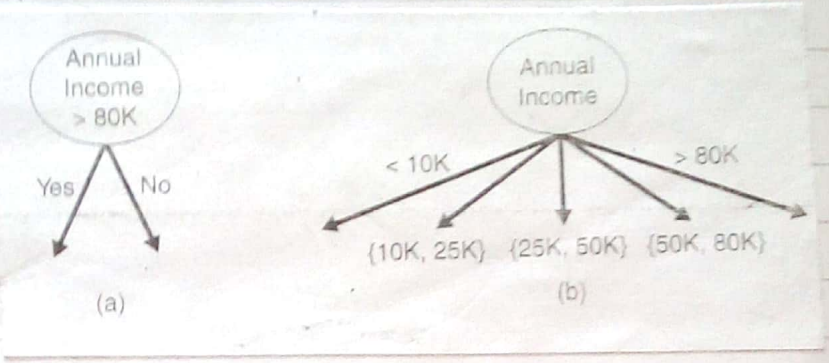
- These attributes can also produce binary (or) multiway splits.
- Ordinal attribute values can be grouped as long as the grouping does not violate the order property of the attribute values.
- Figure 3.9 illustrates various ways of splitting training records based on the Shirt size attribute. The groupings shown in Figures 3.9(a) and 3.9(b) preserve the order among the attribute values, whereas the grouping shown in figure 3.9(c) violates this property because it combines the attribute values Small and Large into the same partition while Medium and Extra Large are combined into another partition.



4. Continuous Attributes

- For these type of attributes, the test condition can be expressed as a comparision test $(A < v)$ (or) $(A \geq v)$ with binary outcomes, or a range query with outcomes of the form $v_i \leq A \leq v_{i+1}$, $i = 1 \cdots k$, the difference between these two is shown in the figure 3.10 (next page)
- For binary case, the algorithm must consider all possible split positions v and it selects the one that produces the best partition.
- For multiway, the algorithm consider all possible ranges of continuous values.

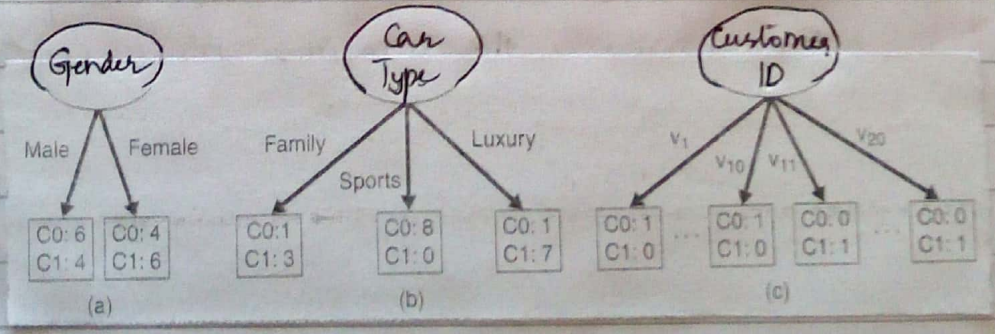
- Applying discretization is one approach, and after applying a new ordinal value will be assigned to each discretized interval.



3.3.4 Measures for Selecting the Best Split

- Measures are defined in terms of the class distribution of the records before and after splitting.
- Let $p(i|t)$ denote the fraction of records belonging to the class i at a given node t. Sometimes referred as p_i.
- In a two-class problem, the class distribution at any node can be written as (p_0, p_1), where $p_1 = 1 - p_0$.

- To illustrate, consider the test conditions in the figure 3.11, The class distribution before splitting is (0.5, 0.5) because records are equal in each class.
- If we split the data using the Gender attribute, then the class distributions of the child nodes are (0.6, 0.4) and (0.4, 0.6) respe.
- Splitting on the second attribute, Car type will result in purer partitions.



- The measures for best split are based on the degree of impurity of the child nodes.
- The smaller the degree of impurity, the more skewed the class distribution.
- For example, a node with class distribution (0,1) has zero impurity, whereas (0.5, 0.5) has highest impurity. Some of the impurity measures are

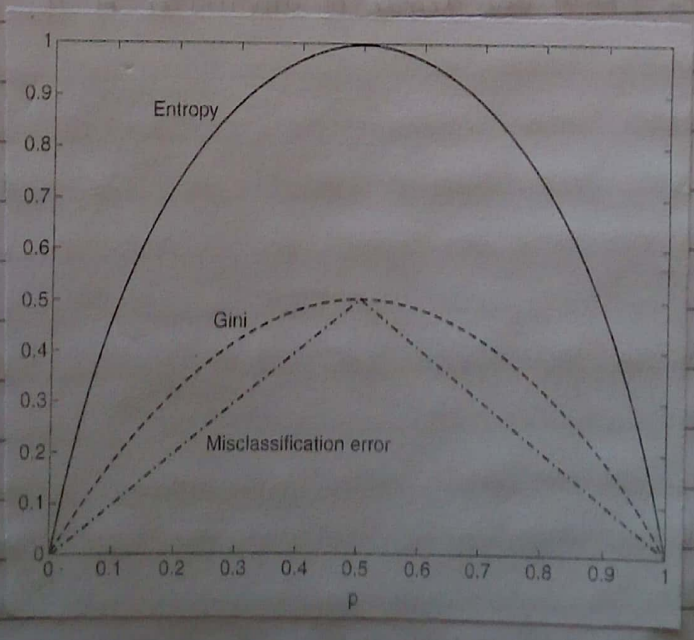
$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$

$$\text{Classification error} = 1 - \max_i [p(i|t)]$$

where c = no. of classes and $0 \log_2 0 = 0$ in entropy cal's.

- Following figure 3.12 compares the values of the impurity measures for binary classification problems.



- p refers to the fraction of records that belong to one of the two classes.

- When $p = 0.5$ all three measures attain their maximum value.
- The minimum values for the measures are attained when $p = 0$ (or) 1.
- The following examples and the figure 3.12 illustrate the consistency among different impurity measures.

Node N_1	Count
Class=0	0
Class=1	6

Gini $= 1 - (0/6)^2 - (6/6)^2 = 0$
Entropy $= -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$
Error $= 1 - \max[0/6, 6/6] = 0$

Node N_2	Count
Class=0	1
Class=1	5

Gini $= 1 - (1/6)^2 - (5/6)^2 = 0.278$
Entropy $= -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$
Error $= 1 - \max[1/6, 5/6] = 0.167$

Node N_3	Count
Class=0	3
Class=1	3

Gini $= 1 - (3/6)^2 - (3/6)^2 = 0.5$
Entropy $= -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$
Error $= 1 - \max[3/6, 3/6] = 0.5$

- Based on these calculations, node N_1 has the lowest impurity value, followed by N_2 and N_3.
- To determine how well a test condition performs, we need to compare the degree of impurity of the parent node (before splitting) with the degree of impurity of the child nodes (after splitting).
- The larger their difference, the better the test condition. The gain, Δ, is a criterion that can be used to determine the goodness of a split:

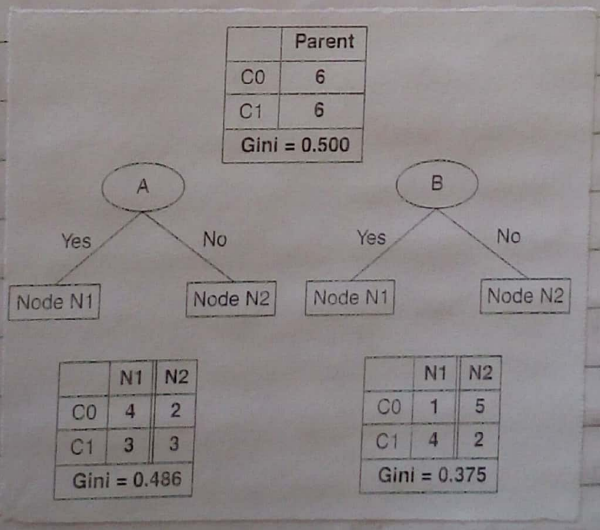
$$\text{Gain}(\Delta) = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j),$$

where $I(\cdot)$ is the impurity measure of a given node, N is the total number of records at the parent node, k is the number of attribute values, and $N(v_j)$ is the number of records associated with the child node, v_j.
- Decision tree induction algorithms often choose a test condition that maximizes the gain Δ.

- When entropy is used as the impurity measure in the gain (Δ) equation, the difference in entropy is known as the information gain (Δ_{info}).

Splitting of Binary Attributes

- Consider the diagram shown in the figure 3.13. Suppose there are two ways to split the data into smaller subsets.
- Before splitting, the Gini index is 0.5 since there are an equal number of records from both classes.
- If attribute A is chosen to split the data, the Gini index for Node N_1 is 0.4898, $N_2 = 0.480$.
- The weighted average of the Gini index for the descendent nodes is $(7/12) \times 0.4898 + (5/12) \times 0.480 = 0.486$, similarly for attribute B, it is 0.375.
- Since the subsets for attribute B have smaller Gini index, it is preferred over Attribute A



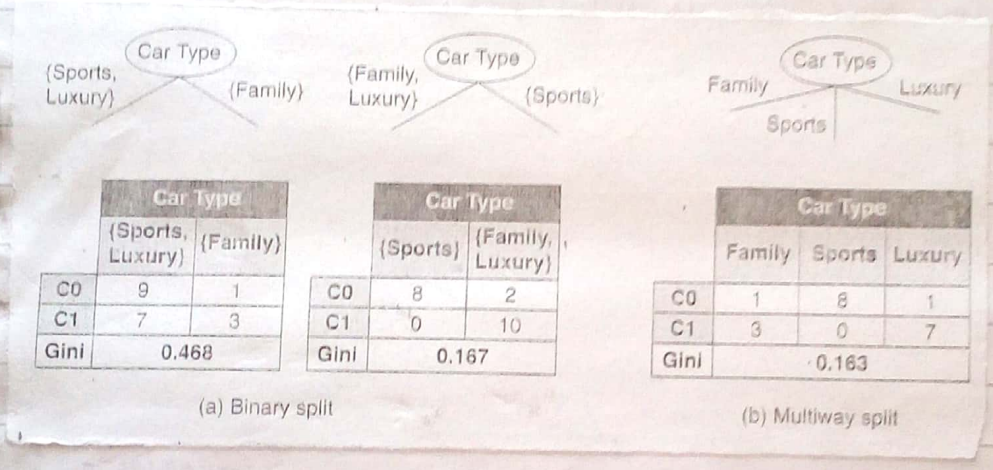
Splitting of Nominal Attributes

- A nominal attribute can produce either binary (or) multiway splits, as shown in the figure 3.14 (next page).
- The computation of the Gini index for a binary split is similar to that shown for determining binary attributes.

- For the first binary grouping of the {Car Type} attribute, the Gini index of {Sports, Luxury} is 0.4922 and the Gini index of {Family} is 0.3750. The weighted average Gini index for the grouping is equal to

$$16/20 \times 0.4922 + 4/20 \times 0.3750 = 0.468.$$

- For the second binary grouping of {Sports} and {Family, Luxury}, the Gini index (weighted average) is 0.167.
- Since the second group has a lower gini index it subsets are much purer.



Car Type	{Sports, Luxury}	{Family}
C0	9	1
C1	7	3
Gini	0.468	

Car Type	{Sports}	{Family, Luxury}
C0	8	2
C1	0	10
Gini	0.167	

(a) Binary split

Car Type	Family	Sports	Luxury
C0	1	8	1
C1	3	0	7
Gini	0.163		

(b) Multiway split

- For the multiway split, the Gini index is computed for every attribute value. Since Gini({Family}) = 0.375, Gini({Sports}) = 0 and Gini({Luxury}) = 0.219, the overall Gini index for the multiway split is equal to

$$4/20 \times 0.375 + 8/20 \times 0 + 8/20 \times 0.219 = 0.163$$

- The multiway splits has a smaller Gini index compared to both two-way splits.
- The two-way split actually merges some of the outcomes of a multiway split, and thus, results in less pure subsets.

Splitting of Continuous Attributes

- Consider the example shown in Fig 3.15, in which the test

condition Annual Income $\leq v$ is used to split the training records for the loan default classification problem.

- A brute-force method for finding v is to consider every value of the attribute in the N records as a candidate split position.
- For each candidate v, the data set is scanned once to count the number of records with annual income less than (or) greater than v.
- We then compute the Gini index for each candidate and choose the one that gives the lowest value, which is computationally complex.
- To reduce the complexity, the training records are sorted based on their annual income, where candidate split positions are identified by taking the mid points between the two adjacent sorted values : 55, 65, 72 and so on.
- However, unlike the brute-force approach, we do not have to examine all N records when evaluating Gini Index of a candidate split position.
- For the first candidate, v = 55, none of the records has annual income less than $55K. As a result, the Gini index for the descendent node with Annual Income < $55K is zero. On the other hand, the number of records with annual income greater than (or) equal to $55K is 3 (Yes) and 7 (No), respectively. Thus the Gini index for this node is 0.420. The overall Gini index for the candidate split position is equal to $0 \times 0 + 1 * 0.420 = 0.420$.
- This procedure is repeated until the Gini index values for all candidates are computed as shown in figure 3.15.
- The best split position is 97 as it's Gini index is small.
- This approach allows us to reduce the number of candidate split positions from 11 to 2.
- Figure 3.15 is shown in the next page.

Gain Ratio

- Impurity measures such as entropy and Gini index tend to favor attributes that have a large number of distinct values.
- Fig 3.11 shows three alternative test conditions for partitioning the data set shown in the following table 3.3.

- Comparing the first test condition, Gender, with the second, Car Type, it is easy to see the Car Type seems to provide a better way of splitting the data science it produces purer descendant nodes.
- If we compare both conditions with Customer ID, the latter produce purer partitions.
- Customer ID is not a predictive attribute because its value is unique for each record.
- Even in a less extreme situations, a test condition that results in a large number of outcomes may not be desirable because the number of records associated with each partition is too small to enable us to make any reliable predictions.
- The two strategies for overcoming the above problem are as following
  a) Restrict the test conditions to binary splits only and is implemented CART algorithm.
  b) Modify the splitting criterion to take into account the number of outcomes produced by the attribute test condition.

  Eg: C4.5 uses the "gain ratio" as splitting criterion to determine the goodness of a split.

$$\text{Gain ratio} = \frac{\Delta_{info}}{\text{Split info}}$$

$$\text{where Split info} = -\sum_{i=1}^{k} P(v_i) \log_2 P(v_i)$$

k = total no. of splits.

3.3.5 Algorithm for Decision Tree Induction

- The following is the skeleton of the decision tree induction algorithm called Tree Growth.

- The input to this algorithm consists of the training records E and the attribute set F.
- The algorithm works by recursively selecting the best attribute to split the data (step 7) and expanding the leaf nodes of the tree (steps 11 and 12) until the stopping criterion is met (step 1).

Algorithm Explanation

1. The createNode() function is to create a new node. A node has either a test condition denoted as node.test_cond (or) class label as node.label.