UNIT – 3
Classification

What is Classification?

Classification, which is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications.
Examples include detecting spam email messages based upon the message header and content, categorizing cells as malignant or benign based upon the results of MRI scans, and classifying galaxies based upon their shape

Classification as the task of mapping an input attribute set $\mathbf{x}$ into its class label y.

The input data for a classification task is a collection of records(*training set*). 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 or target attribute).

Is classification a supervised learning problem?--Yes

Goal: previously unseen records should be assigned a class as accurately as possible.
A test set is used to determine the accuracy of the model.
Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.
Definition: Classification is the task of learning a **target function** *f* that maps each attribute set **x** to one of the predefined class labels *y*.

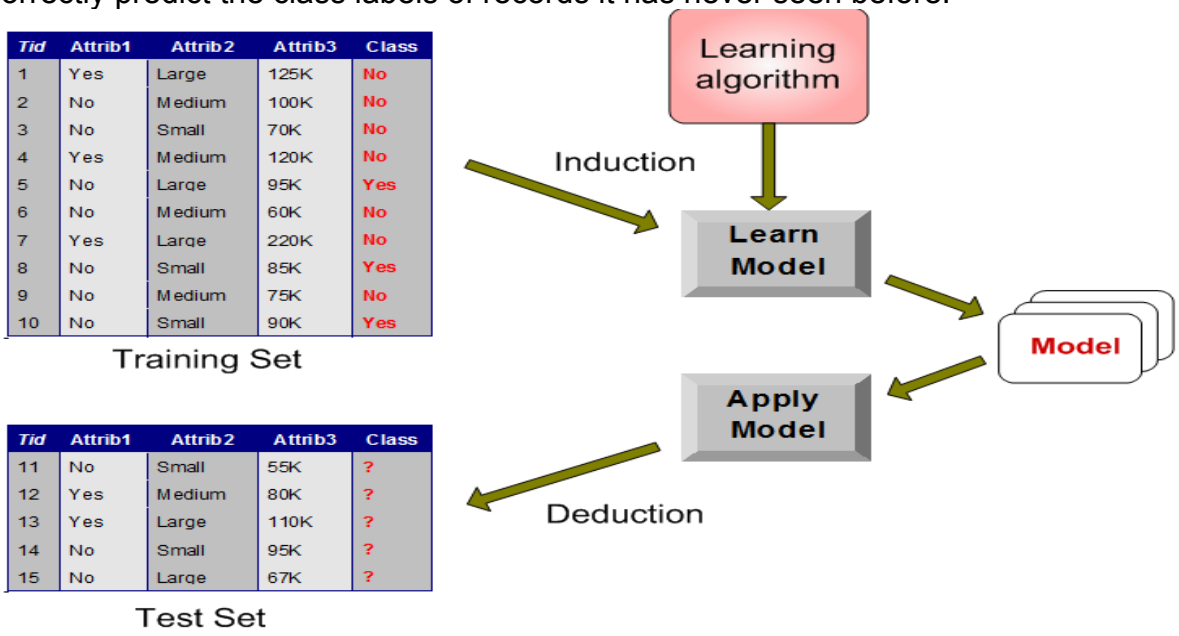
The target function is also known informally as a classification model. A classification model is useful for the following purposes

- Descriptive Modeling: A classification model can serve as an explanatory tool to distinguish between objects of different classes.
    - Table explains what features define a borrower as a defaulter or not.
- Predictive Modeling: A classification model can also be used to predict the class label of unknown records.

General Approach to Solving a Classification Problem

A classification technique (or classifier) is a systematic approach to building classification models from an input data set. Examples include decision tree classifiers, rule-based classifiers, neural networks, support vector machines, and naive Bayes classifiers. Each technique employs 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 by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before.



Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

General Approach to Solving a Classification Problem

Performance metrics:

Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model.

These counts are tabulated in a table known as a confusion matrix.

f01 is the number of records from class 0 incorrectly predicted as class 1

Table 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}

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Decision Tree Induction:

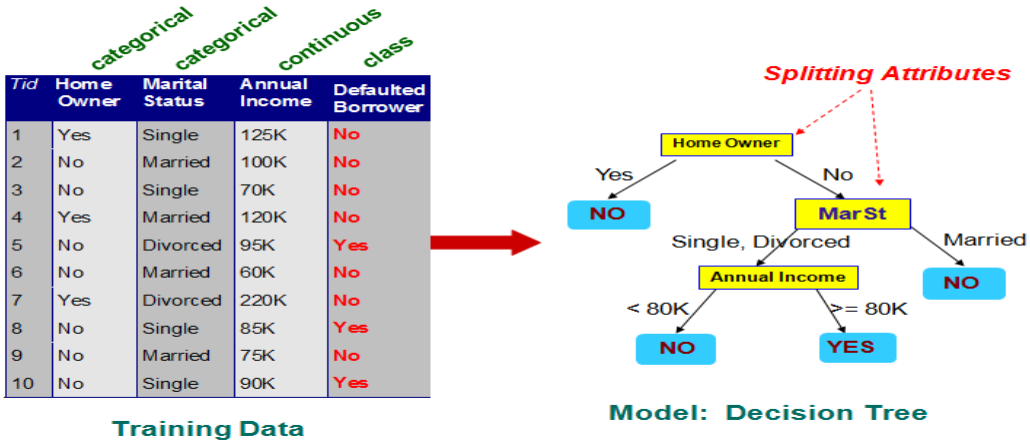
How a Decision Tree Works?

- The tree has three types of nodes:
  - Root node that has no incoming edges and zero or more outgoing edges.
  - Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges.
  - Leaf or terminal nodes, each of 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.

How to Build a Decision Tree?

- In principle, there are exponentially many decision trees that can be constructed from a given set of attributes.
- While some of the trees are more accurate than others, finding the optimal tree is computationally infeasible because of the exponential size of the search space. Hunt's algorithm, which is the basis of many existing decision tree induction algorithms, including ID3, C4.5, and CART.



Hunt's Algorithm:

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successive purer subsets.

Let D_t be the 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.

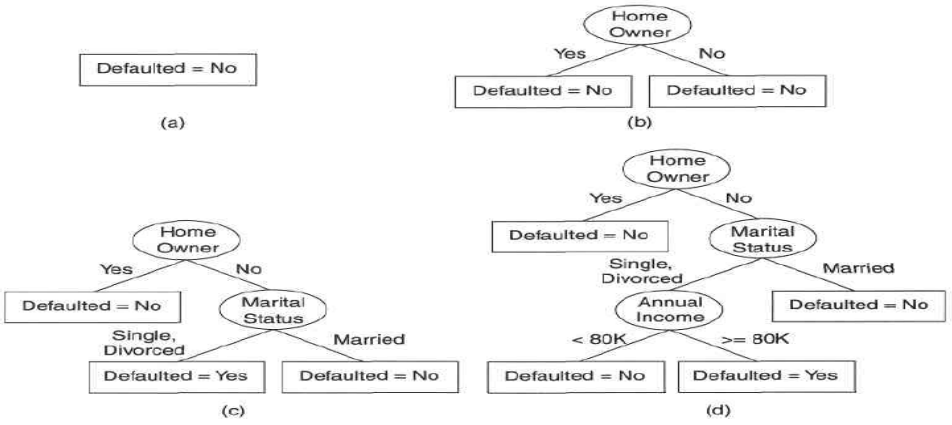


Figure · Hunt's algorithm for inducing decision trees.

Design Issues of Decision Tree Induction

A learning algorithm for inducing decision trees must address the following two issues.

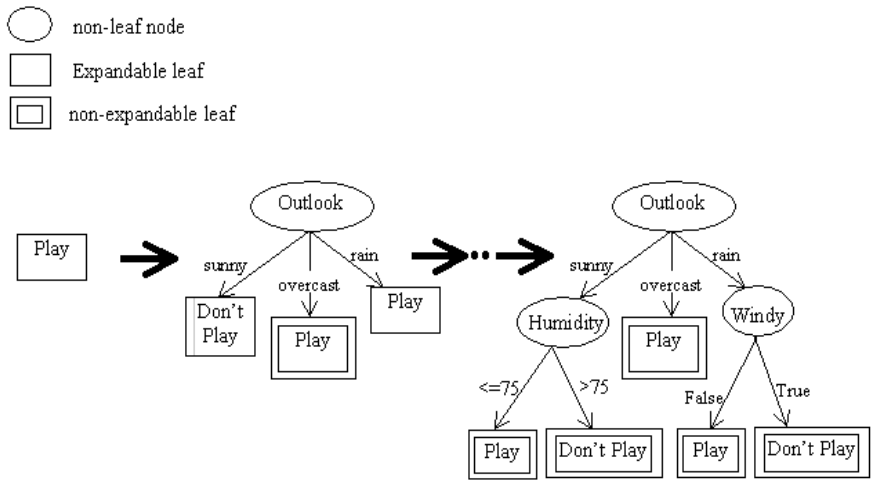
1. How should the training records be split?
   Each recursive step of the tree-growing process must select an attribute test condition to divide the records into smaller subsets
2. How should the splitting procedure stop?
   A stopping condition is needed to terminate the tree-growing process. A possible strategy is to continue expanding a node until either all the records belong to the same class or all the records have identical attribute values

Classification Problem-2

Outlook	Temp(F)	Humidity(%)	Windy?	Class
sunny	75	70	true	Play
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play
rain	65	70	true	Don't Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Table 2.1: A small training data set [2]

Solution:



(a)Initial Classification Tree (b)Intermediate Classification Tree (c) Final Classification Tree

Figure 2.1: Demonstration of Hunt's Method

Classification Problem-3

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	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

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

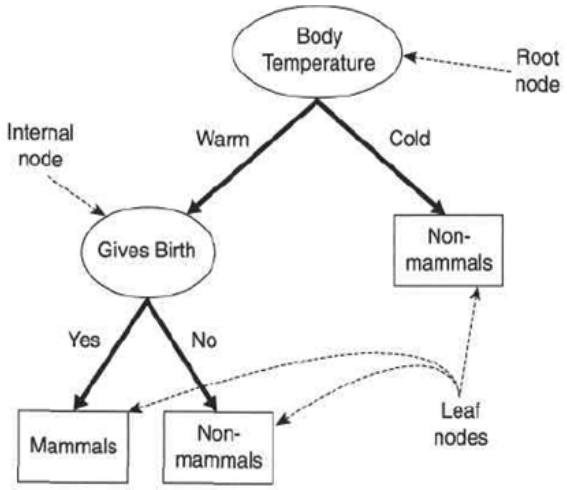
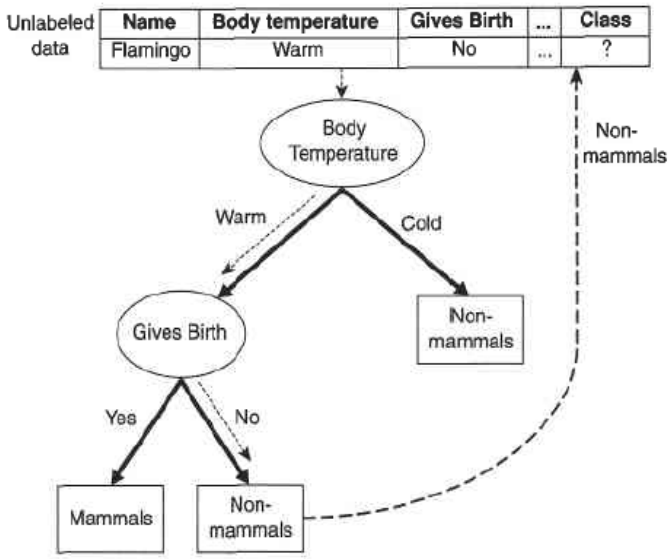


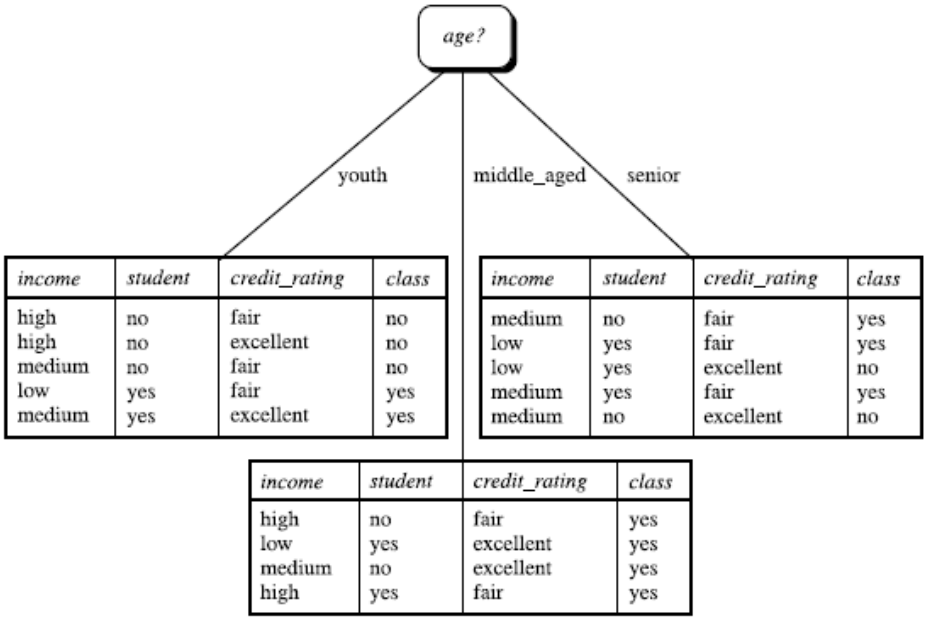
Figure A decision tree for the mammal classification problem.



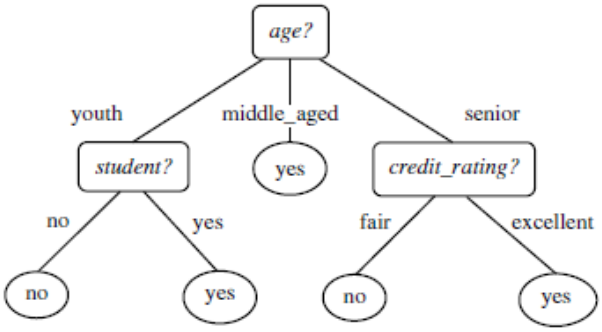
Classification Problem-4

Class-labeled training tuples from the *AllElectronics* customer database.

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no



The attribute *age* has the highest information gain and therefore becomes the splitting attribute at the root node of the decision tree. Branches are grown for each outcome of *age*. The tuples are shown partitioned accordingly.



A decision tree for the concept *buys_computer*, indicating whether a customer at *AllElectronics* is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

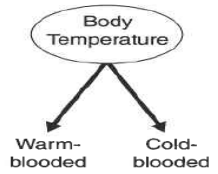
Methods for Expressing Attribute Test Conditions

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

Depends on attribute types
- Binary
- Nominal
- Ordinal
- Continuous

Depends on number of ways to split
- 2-way split
- Multi-way split

Splitting Based on Binary Attributes: The test condition for a binary attribute generates two potential outcomes `



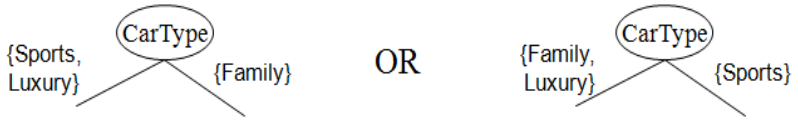
Test condition for binary attributes.

Splitting Based on Nominal Attributes:

Since a nominal attribute can have many values, its test condition can be expressed in two ways.

1.**Multi-way split**: The number of outcomes depends on the number of distinct values for the corresponding attribute.

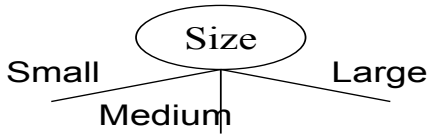
2.Binary split: Divides values into two subsets. Need to find optimal partitioning.



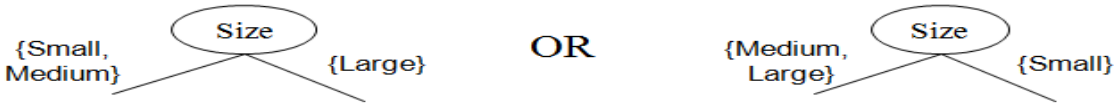
Splitting Based on Ordinal Attributes:

Ordinal attribute values can be grouped as long as the grouping does not violate the order property of the attribute values

Multi-way split: Use as many partitions as distinct values.



Binary split: Divides values into two subsets. Need to find optimal partitioning.



Splitting Based on Continuous Attributes:

Different ways of handling

- Discretization to form an ordinal categorical attribute
  - Static – discretize once at the beginning
  - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
- Binary Decision: (A < v) or (A ≥ v)
  - consider all possible splits and finds the best cut
  - can be more compute intensive

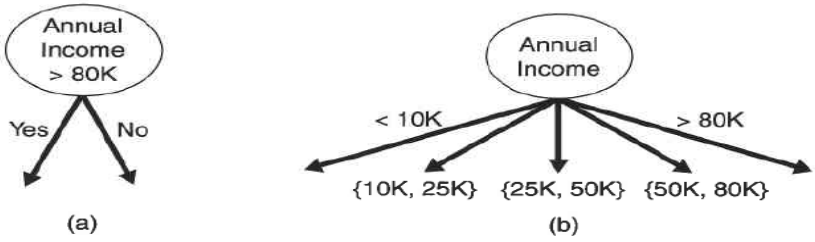


Figure Test condition for continuous attributes.

Measures for Selecting the Best Split:

There are many measures that can be used to determine the best way to split the records. These measures are defined in terms of the class distribution of the records before and after splitting.

Measures of Node Impurity:

- Gini Index
- Entropy
- Misclassification error

Let $p(i|t)$ denote the fraction of records belonging to class *i* at a given node *t*.

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

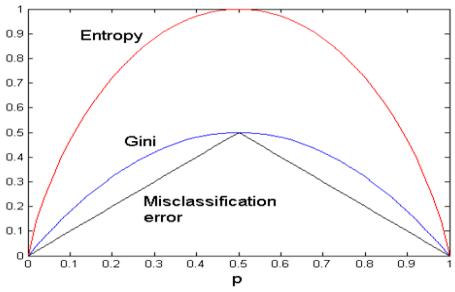
$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2,$$

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

where c is the number of classes and $0 \log_2 0 = 0$ in entropy calculations

Comparison among the impurity measures for binary classification problem

For a 2-class problem:



- 'P' refers to the fraction of records that belong to one of the two classes
- All three measures attain their maximum value when the class distribution is uniform (i.e., when P = 0.5).
- The minimum values for the measures are attained when all the records belong to the same class (i.e., when P equals 0 or 1).

Examples of computing the different impurity measures:

Node N_1	Count
Class=0	0
Class=1	6

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

Node N_2	Count
Class=0	1
Class=1	5

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

Node N_3	Count
Class=0	3
Class=1	3

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

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:

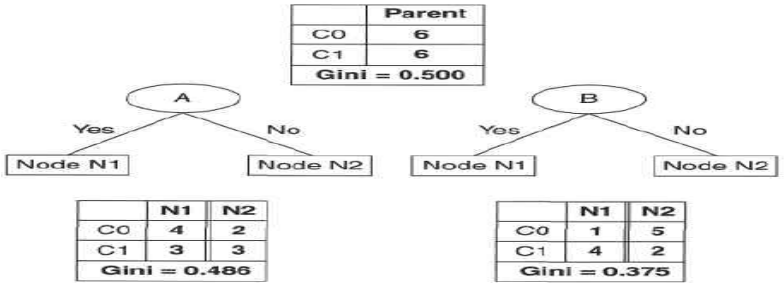
$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j),$$

where *I(.)* 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 . Since *I(parent)* is the same for all test conditions, maximizing the gain is equivalent to minimizing the weighted average impurity measures of the child nodes.

Splitting of Binary Attributes:

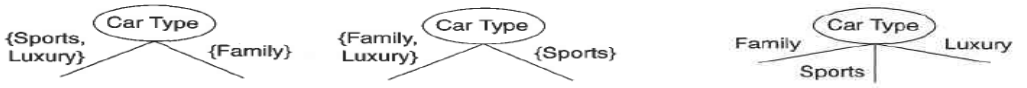
Suppose there are two ways to split the data into smaller subsets. Before splitting, the Gini index is 0.5 since
there are an equal number of records from both classes. If attribute A is chosen to split the data, the Gini index for node N1 is 0.4898, and for node N2, it is 0.480. The weighted average of the Gini index for the descendent nodes is (7/12) x 0.4898 + (5/12) x 0.480 = 0.486. Similarly, Gini index for attribute B is 0.375.
Since the subsets for attribute B have a smaller Gini index, it is preferred over attribute A.



Splitting binary attributes.

Splitting of Nominal Attributes:

A nominal attribute can produce either binary or multiway splits.



	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	

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

(a) Binary split

(b) Multiway split

Splitting nominal attributes.

Splitting of Continuous Attributes:

- ✓ Brute-force method for finding *v* is to consider every value of the attribute in the *N* records as a candidate split position.
- ✓ For efficient computation: Sort the attribute on values
- ✓ 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.

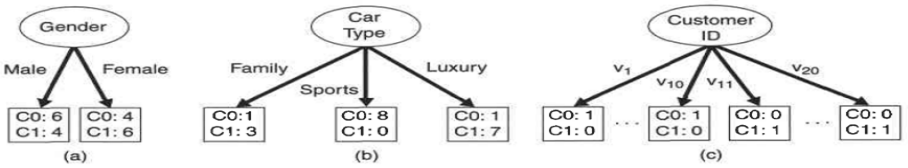
Class	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	*0.300*	0.343	0.375	0.400	0.420

Figure Splitting continuous attributes.

- ✓ This problem can be further optimized by considering only candidate split positions located between two adjacent records with different class labels.
- ✓ Therefore, the candidate split positions at v = $55K, $65K, $72K, $87K, $92K, $110K, $I22K, $772K, and $230K are ignored because they are located between two adjacent records with the same class labels.
- ✓ This approach allows us to reduce the number of candidate split positions from 11 to 2.

Gain Ratio

Impurity measures such as entropy and Gini index tend to favor attributes that have a large number of distinct values



Multiway versus binary splits.

If we compare Gender and Car Type with Customer ID, it produce purer partitions
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.

There are two strategies for overcoming this problem.

- ✓ The first strategy is to restrict the test conditions to binary splits only.
  - ✓ This strategy is employed by decision tree algorithms such as **CART.**
- ✓ Another strategy is to modify the splitting criterion to take into account the number of outcomes produced by the attribute test condition.
  - ✓ For example, in the **C4.5** decision tree algorithm, a splitting criterion known as gain ratio is used to determine the goodness of a split.

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

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

k is the total number of splits
This example suggests that if an attribute produces a large number of splits, its split information will also be large, which in turn reduces its gain ratio.

Algorithm for Decision Tree Induction:

```
TreeGrowth (E, F)
 1: if stopping_cond(E,F) = true then
 2:    leaf = createNode().
 3:    leaf.label = Classify(E).
 4:    return leaf.
 5: else
 6:    root = createNode().
 7:    root.test_cond = find_best_split(E, F).
 8:    let V = {v|v is a possible outcome of root.test_cond }.
 9:    for each v ∈ V do
10:       E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:       child = TreeGrowth(E_v, F).
12:       add child as descendent of root and label the edge (root → child) as v.
13:    end for
14: end if
15: return root.
```

- The createNode() function extends the decision tree by creating a new node.
- A node in the decision tree has either a test condition, denoted as node.test-cond, or a class label, denoted as node.label.
- find_best_split() function determines which attribute should be selected as the test condition for splitting the training records
- Classify() function determines the class label to be assigned to a leaf node

- Stopping_cond() function is used to terminate the tree-growing process by testing whether all the records have either the same class label or the same attribute values

Model Overfitting:

- The errors committed by a classification model are generally divided into two types:
  - Training errors (resubstitution error or apparent error)
  - Generalization errors.
- Training error, is the number of misclassification errors committed on training records
- Generalization error is the expected error of the model on previously unseen records
- A Model must have low training error as well as low generalization error.

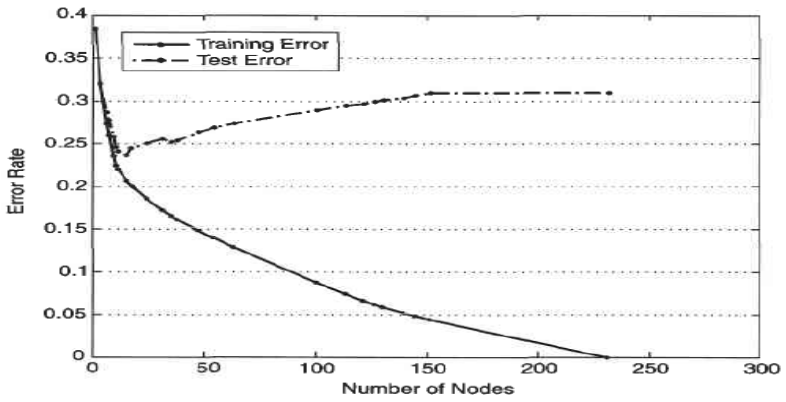
Model underfitting

- The training and test error rates of the model are large when the size of the tree is very small. This situation is known as model underfitting.

Model overfitting

Once the tree becomes too large, its test error rate begins to increase even though its training error rate continues to decrease. This phenomenon is known as model overfitting.

- Overfitting Due to Presence of Noise
- Overfitting Due to Lack of Representative Samples



Training and test error rates.

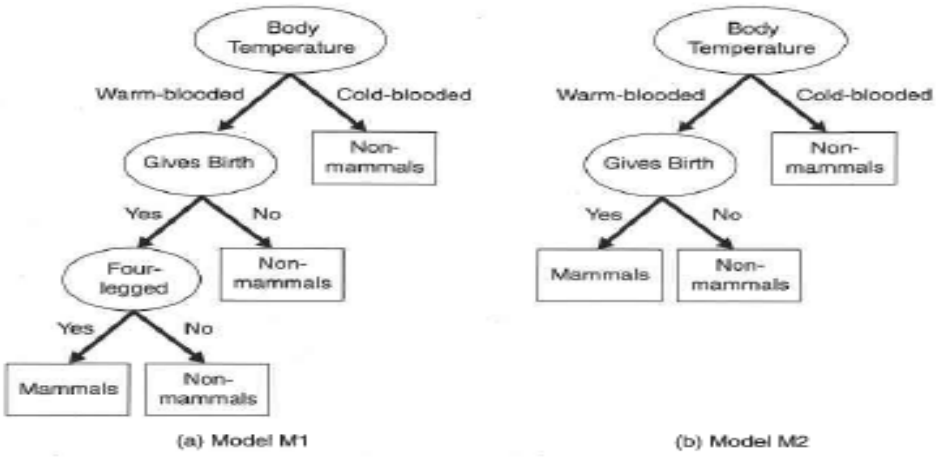
1. **Overfitting Due to Presence of Noise**
   Two of the ten training records are mislabeled: bats and whales are classified as non-mammals instead of mammals.

Table An example training set for classifying mammals. Class labels with asterisk symbols represent mislabeled records.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

Table 4.4. An example test set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no



The training error for the tree is zero, its error rate on the test set is 30%. Both humans and dolphins were misclassified as nonmammals because their attribute values for Body Temperature, Gives Birth, and Four-legged are identical to the mislabeled records in the training set.

2. Overfitting Due to Lack of Representative Samples

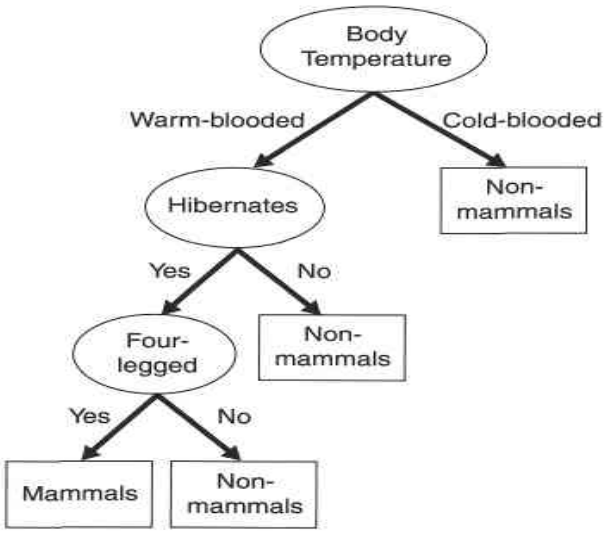
Models that make their classification decisions based on a small number of training records are also liable to overfitting

Humans, elephants, and dolphins are misclassified because the decision tree classifies all warm-blooded vertebrates that do not hibernate as non-mammals(ie.eagle)

This example clearly demonstrates the danger of making wrong predictions when there are not enough representative examples (Here only one ie. eagle) at the leaf nodes of a decision tree.

Table An example training set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
salamander	cold-blooded	no	yes	yes	no
guppy	cold-blooded	yes	no	no	no
eagle	warm-blooded	no	no	no	no
poorwill	warm-blooded	no	no	yes	no
platypus	warm-blooded	no	yes	yes	yes



Determine when to stop splitting:

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
  Eg. Above decision tree

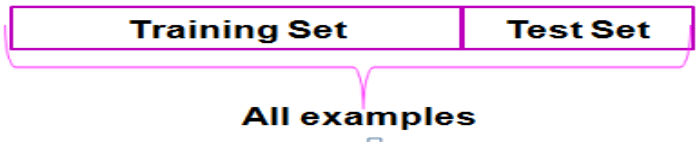
Evaluating the Performance of a Classifier:

Some of the methods commonly used to evaluate the performance of a classifier

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Bootstrap
  - Sampling with replacement

1. **The holdout method**
   The original data with labeled examples is partitioned into two disjoint sets

- Training set: used to train the classifier
- Test set (or 'hold out' set) : used to estimate the error rate of the trained classifier
- The proportion of data reserved for training and for testing is typically at the discretion of the analysts
- E.g. 50-50 or Two-thirds for training and one-third for testing



The holdout method has several well-known limitations:

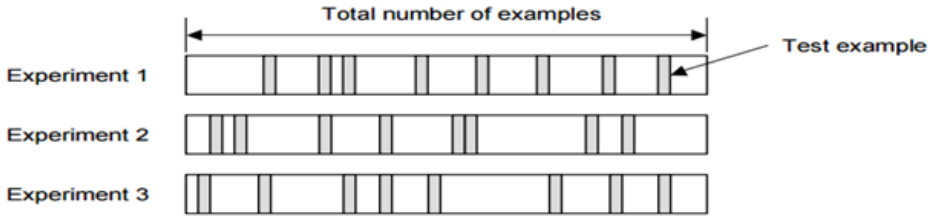
– First, fewer labeled examples are available for training because some of the records are withheld for testing.

– Second, the model may be highly dependent on the composition of the training and test sets. The smaller the training set size, the larger the variance of the model.

2. **Random Subsampling**
   - The holdout method can be repeated several times to improve the estimation of a classifier's performance. This approach is known as random subsampling.
   - Each split randomly selects a (fixed) no. examples without replacement
   - This estimate is significantly better than the holdout estimate

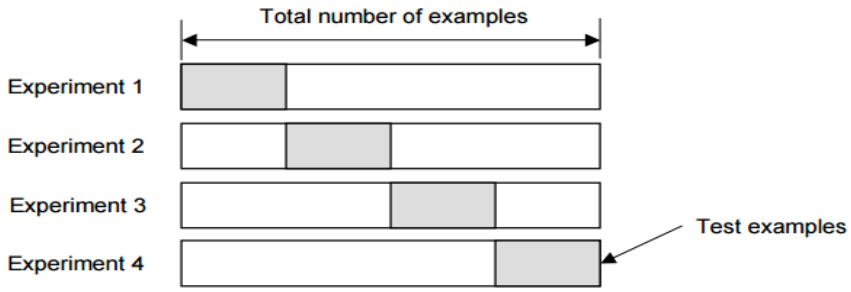
Let acc_i be the model accuracy during the i^{th} iteration.

The overall accuracy is given by $acc_{\text{sub}} = \sum_{i=1}^{k} acc_i/k.$



3. **Cross-Validation**
   - An alternative to random subsampling is cross-validation
   - In this approach, Partition the data into two equal-sized subsets.
   - First, we choose one of the subsets for training and the other for testing.
   - We then swap the roles of the subsets so that the previous training set becomes the test set and vice versa.
   - This approach is called a **twofold** cross-validation
   - The total error is obtained by summing up the errors for both runs.
   - In this example, each record is used exactly once for training and once for testing.

- **The k-fold cross-validation** method generalizes above approach by segmenting the data into k equal-sized partitions (Eg-4)
- During each run, one of the partitions is chosen for testing, while the rest of them are used for training.
- This procedure is repeated k times so that each partition is used for testing exactly once



- A **special case** of the k-fold cross-validation method sets k = N, the size of the data set. In this so-called **leave-one-out approach**, each test set contains only one record.
- This approach has the advantage of utilizing as much data as possible for training.
- In addition, the test sets are mutually exclusive and they effectively cover the entire data set.
- The drawback of this approach is that it is computationally expensive to repeat the procedure N times.

4. **Bootstrap Method**
   - The methods presented so far assume that the training records are sampled without replacement.
   - In the bootstrap approach, the training records are sampled with replacement.
   - i.e., a record already chosen for training is put back into the original pool of records so that it is equally likely to be redrawn.
   - The bootstrap method is also called the ***0.632 bootstrap***
   - This means the training data will contain approximately 63.2% of the instances and the test data will contain approximately 36.8% of the instances.

   Estimating Error with the Bootstrap Method:
   - The error estimate on the test data will be very pessimistic because the classifier is trained on just ~63% of the instances.
   - Therefore, combine it with the training error:

     $$err = 0.632 \cdot e_{\text{test instances}} + 0.368 \cdot e_{\text{training instances}}$$

   - The training error gets less weight than the error on the test data.
   - Repeat process several times with different replacement samples; average the results