UNIT-4

Classification

- What is Classification
- General Approach to classification
- Decision Tree Induction
- What is cluster Analysis
- Overview of basic clustering methods
- K Means Centroid based technique
- Data Mining Applications

Unit – 4 Classification

4.8 What is Classification?

Classification is a classic data mining technique and/or problem of identifying set of categories (subpopulations) based on machine learning. Basically, classification is used to classify each item in a set of data into one of a predefined set of classes or groups. Classification method makes use of mathematical techniques such as decision trees, linear programming, neural network and statistics. Classification is the, a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

Note that

- Classification constructs the classification model by using **training data set.**
- Classification predicts the value of classifying **attribute or class** label.

For example: Classification of bank customers on the basis on accounts (Current/ saving/ Loan/ Overdraft etc. Same University gives the class to the students based on marks like,

If percentage >= 70, then distinction class
Else If percentage >=60 and percentage <70, then First class
Else If percentage >=50 and percentage <60, then Second class
Else If percentage >=35 and percentage <50, then Pass class
Else Fail class

4.9 General Approach to classification:

Data classification is a two-step process, consisting of a learning step (where a classification model is constructed) and a classification step (where the model is used to predict class labels for given data). The process is shown for the loan application data of following figure (The data are simplified for illustrative purposes. In reality, we may expect many more attributes to be considered.

In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels.

A tuple, X, is represented by an n-dimensional attribute vector, X = (x1, x2,..., xn), depicting n measurements made on the tuple from n database attributes, respectively, A1, A2,..., An. 1 Each tuple, X, is assumed to belong to a predefined class as determined by another database attribute called the class label attribute. The class label attribute is discrete-valued and unordered. It is categorical (or nominal) in that each value serves as a category or class. The individual tuples making up the training set are referred to as training tuples and are randomly sampled from the database under analysis. In the context

of classification, data tuples can be referred to as samples, examples, instances, data points, or objects. 2

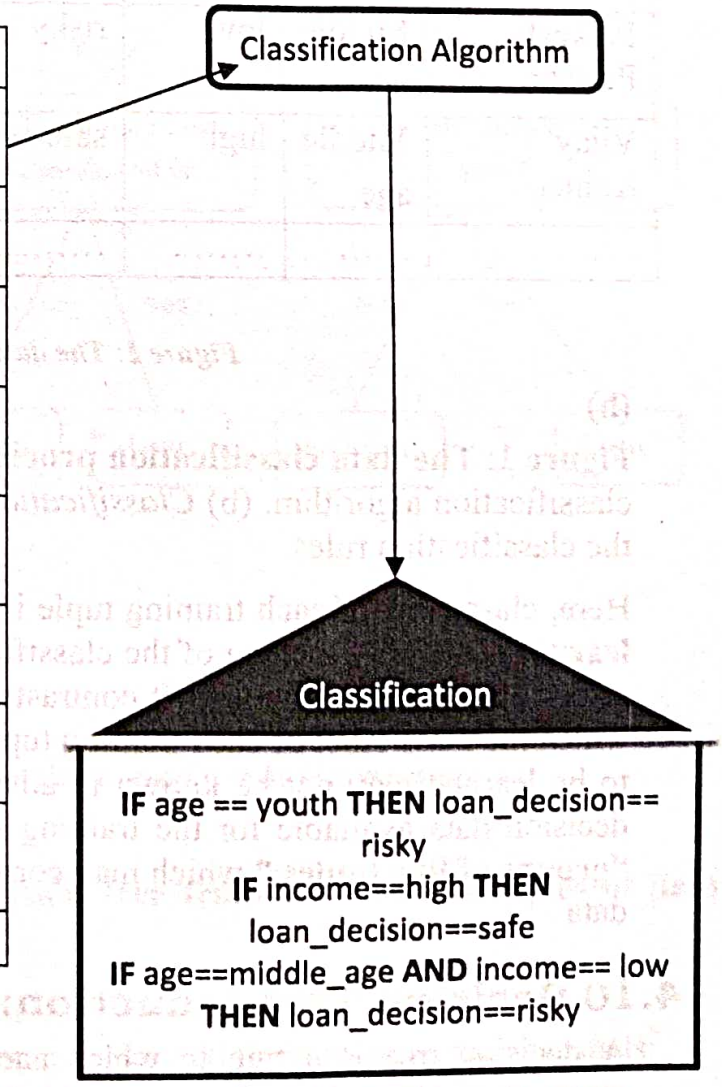
Example :

I. Building the Classifier or Model (***training data)***
II. Using Classifier for Classification ***(Test data)***

❖ **Building the Classifier or Model:**

- This step is the learning step or the learning phase.
- In this step the classification algorithms build the classifier.
- The classifier is built from the training set made up of database tuples and their associated class labels.
- Each tuple that constitutes the training set is referred to as a category or class. These tuples can also be referred to as sample, object or data points.

Training Data			
Name	**age**	**Income**	**Loan decision**
Mahesh Pandya	youth	low	risky
Vishal Soni	youth	low	risky
Chirag Sharma	Middle age	high	safe
Amit Patel	Middle age	low	risky
Abdul Pathan	senior	low	safe
Sweta Shah	senior	medium	safe
Paresh Shah	Middle age	high	safe
.....	.....	.....	.....



(a) Using Classifier for Classification:

In this step, the classifier is used for classification. Here the test data is used to estimate the accuracy of classification rules. The classification rules can be applied to the new data tuples if the accuracy is considered acceptable.

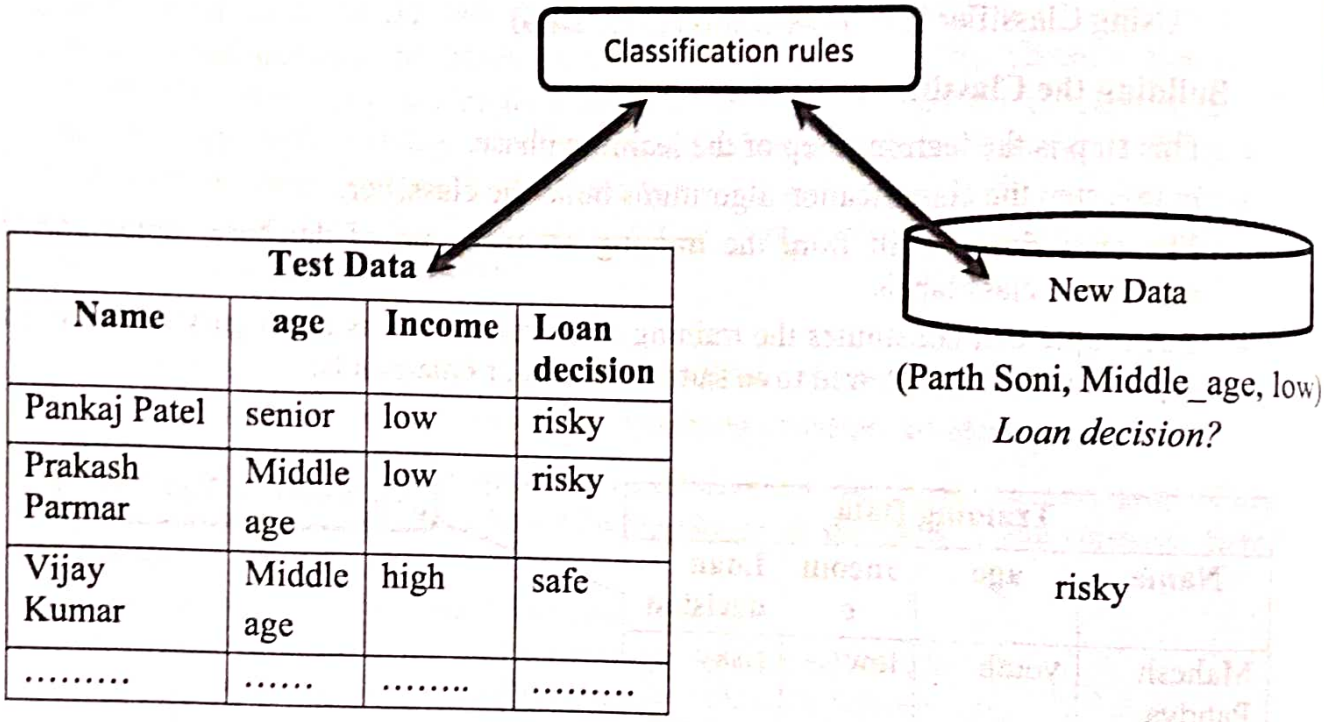


Figure 1: The data classification process

(b)

Figure 1: The data classification process: (a) *Learning:* Training data are analyzed by a classification algorithm. (b) *Classification*: Test data are used to estimate the accuracy of the classification rules.

Here, class label of each training tuple is provided, this step is also known as **supervised learning** (i.e., the learning of the classifier is "supervised" in that it is told to which class each training tuple belongs). It contrasts with **unsupervised learning** (or clustering), in which the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance. For example, if we did not have the loan decision data available for the training set, we could use clustering to try to determine **"groups of like tuples,"** which may correspond to risk groups within the loan application data.

4.10 Decision Tree Induction:

A decision tree is a tree in which each branch node represents a choice between numbers of alternatives, here, and tree is a structure that includes a root node, branch

and leaf nodes (each leaf node represents a classification or *decision*). Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node. For example, we might have a decision tree to help a company decide whether an employee should be offered a loan or not by range of income of employees.

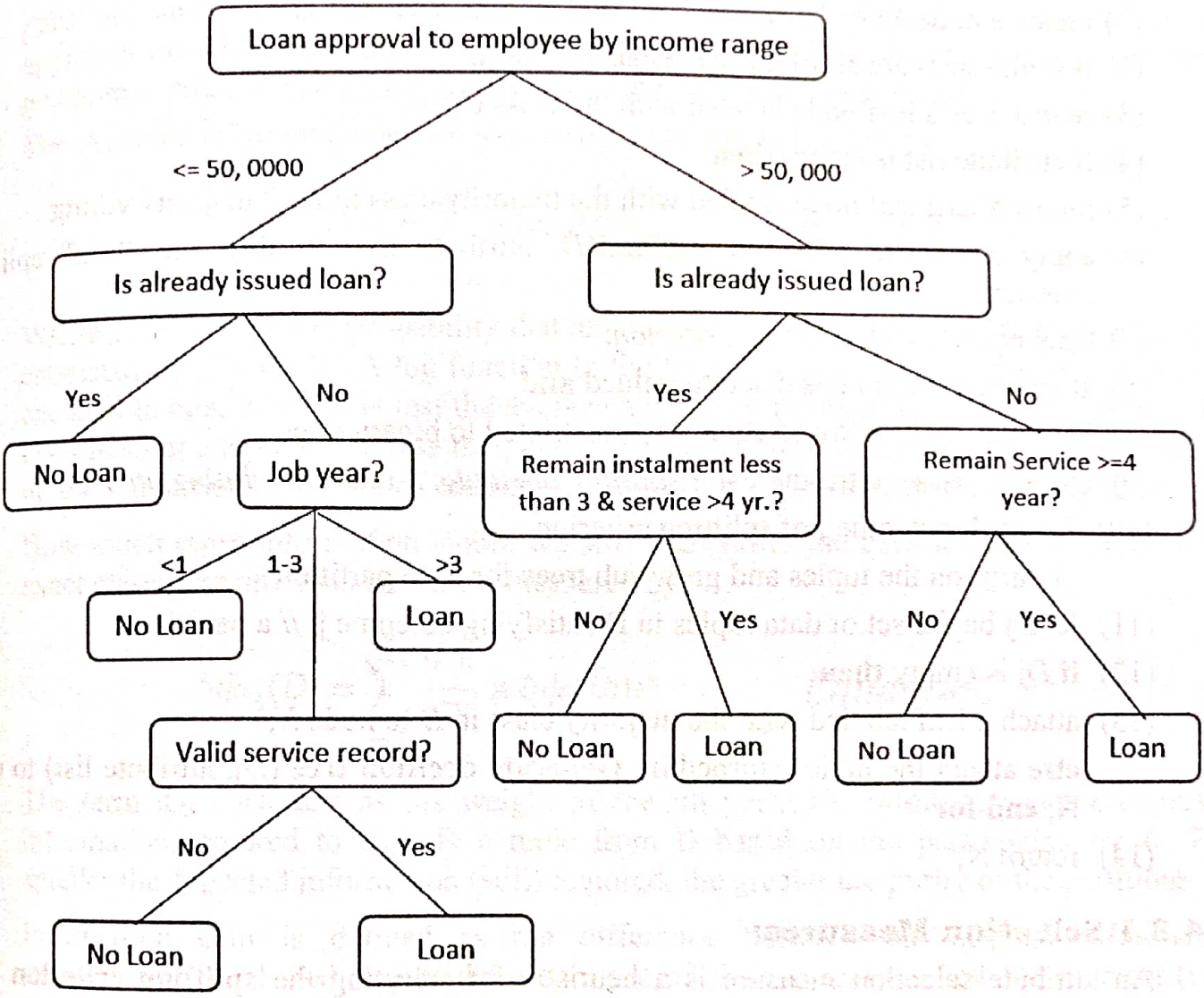


Figure 2: Decision tree for loan sanction to employee by income range

❖ **Decision tree Algorithm:**

Generate decision tree. Generate a decision tree from the training tuples of data partition, D.

Input:

- ⇨ Data partition, D, which is a set of training tuples and their associated class labels;
- ⇨ Attribute list, the set of candidate attributes;

⇨ Attribute selection method, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a splitting attribute and, possibly, either a split-point *or splitting subset.*

Output: A decision tree.

- **Method:**

(1) create a node *N*;
(2) **if** tuples in *D* are all of the same class, *C*, **then**
(3) return *N* as a leaf node labeled with the class *C*;
(4) **if** attribute list is empty **then**
(5) return *N* as a leaf node labeled with the majority class in *D*; // majority voting
(6) apply **Attribute selection method**(*D*, attribute list) to **find** the "best" splitting criterion;
(7) label node N with *splitting criterion*;
(8) **if** splitting attribute is discrete-valued **and**
Multiday splits allowed **then** // not restricted to binary trees
(9) attribute list ← attribute list – *splitting attribute*; // remove *splitting attribute*
(10) **for each** outcome j of splitting criterion
// partition the tuples and grow sub trees for each partition
(11) let *D*j be the set of data tuples in D satisfying outcome j; // a partition
(12) **if** *D*j is empty **then**
(13) attach a leaf labeled with the majority class in *D* to node *N*;
else attach the node returned by **Generate decision tree** (*D*j, attribute list) to node N; **end for**
(14) return N;

4.3.1 Selection Measures:

An attribute selection measure is a heuristic for selecting the **splitting criterion that "best" separates a given data partition**, *D*, of class-labeled training tuples into individual classes. If we were to split *D* into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be pure (i.e., all the tuples that fall into a given partition would belong to the same class). Conceptually, the "best" splitting criterion is the one that most closely results in such a scenario. Attribute selection measures are also known as splitting rules because they determine how the tuples at a given node are to be split. The attribute selection measure provides a ranking for each attribute describing the given training tuples. The attribute having the best score for the measure4 is chosen as the splitting attribute for the given tuples.

4.10.1.1 Information Gain:

ID3 uses information gain as its attribute selection measure. This measure is based on pioneering work by Claude Shannon on information theory, which studied the value or "information content" of messages. Let node N represent or hold the tuples of partition D. The attribute with the highest information gain is chosen as the splitting attribute for node N. This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or "impurity" in these partitions. Such an approach minimizes the expected number of tests needed to classify a given tuple and guarantees that a simple (but not necessarily the simplest) tree is found.

The expected information needed to classify a tuple in D is given by

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i), \qquad \textit{Equation 1}$$

Where ***pi*** is the nonzero probability that an arbitrary tuple in *D* belongs to class **Ci** and is estimated **by |Ci, D|/|D|.** A log function to the base 2 is used, because the information is encoded in bits. Info *(D)* is just the average amount of information needed to identify the class label of a tuple in *D*. Note that, at this point, the information we have is based solely on the proportions of tuples of each class. Info (*D*) is also known as the entropy of *D*.

How much more information would we still need (after the partitioning) to arrive at an exact classification? This amount is measured by

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j). \qquad \textit{Equation 2}$$

The term |Dj | |D| acts as the weight of the jth partition. InfoA (D) is the expected information required to classify a tuple from D based on the partitioning by A. The smaller the expected information (still) required, the greater the purity of the partitions.

Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is,

$$\text{Gain (A)} = \text{Info (D)} - \text{InfoA (D)} \qquad \textit{Equation 3}$$

In other words, Gain (A) tells us how much would be gained by branching on A. It is the expected reduction in the information requirement caused by knowing the value of A. The attribute A with the highest information gain, Gain (A), is chosen as the splitting attribute at node N. This is equivalent to saying that we want to partition on the attribute A that would do the "best classification," so that the amount of information still required to finish classifying the tuples is minimal (i.e., minimum InfoA (D)).

Table: 1 Class-Labeled Training Tuples from the ABC Company employee Database

RID	Age	Income	Employee	Credit-rating	Class: Buys Computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	Middle-age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	Middle-age	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-age	medium	no	excellent	yes
13	Middle-age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Induction of a decision tree using information gain. Table 1 presents a training set, *D*, of class-labeled tuples randomly selected from the ABC company employee database. The class label attribute, buys computer, has two distinct values (namely, {**yes, no**}); therefore, there are two distinct classes (i.e., **m = 2**). Let class **C1** correspond to yes and class **C2** correspond to no. There are ***nine tuples of class yes*** and ***five tuples of class no***. A (root) node *N* is created for the tuples in *D*. To find the splitting criterion for these tuples, we must compute the information gain of each attribute. We first use to compute the expected information needed to classify a tuple in *D* **using equation 1.**

Example 1

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

Next, we need to compute the expected information requirement for each attribute. Let's start with the attribute age. We need to look at the distribution of yes and no tuples for each category of age. For the age category "youth," there are two yes tuples and three no tuples. For the category "middle aged," there are four yes tuples and zero no tuples. For the category "senior," there are three yes tuples and two no tuples. **Using Equation 2, the**

expected information needed to classify a tuple in D if the tuples are partitioned according to age is -

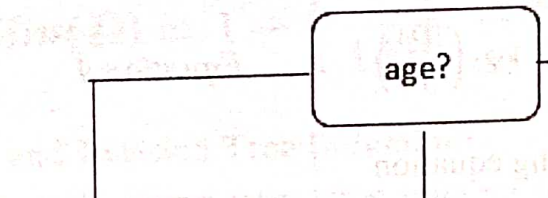
$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) = 0.694 \text{ bits.}$$

Hence, the gain in information from such a partitioning would be -

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Similarly, we can compute Gain (income) = 0.029 bits, Gain (student) = 0.151 bits, and Gain (credit rating) = 0.048 bits. Because age has the highest information gain among the attributes, it is selected as the splitting attribute. Node N is labeled with age, and branches are grown for each of the attribute's values.

The tuples are then partitioned accordingly, as shown in *figure* 3. Notice that the tuples falling into the partition for age = middle aged all belong to the same class. Because they all belong to class "yes," a leaf should therefore be created at the end of this branch and labeled "yes".



income	employee	Credit-rate	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	employee	Credit-rate	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	employee	Credit rate	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Figure: 3 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.

4.10.1.2 Gain Ratio:

The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values. For example, consider an attribute that acts as a unique identifier such as product ID. A split on product ID would result in a large number of partitions (as many as there are values), each one containing just one tuple. Because each partition is pure, the information required to classify data set D based on this partitioning would be Info product ID (D) = 0. Therefore, the information gained by partitioning on this attribute is maximal. Clearly, such a partitioning is useless for classification. The gain ratio, which attempts to overcome this bias. It applies a kind of normalization to information gain using a "split information" value defined analogously with Info (D) as

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right). \qquad \textit{Equation 4}$$

Gain ratio defined by the following equation

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)} \qquad \textit{Equation 5}$$

Computation of gain ratio for the attribute income:

A test on income splits the data of **Table 1** into three partitions, namely low, medium, and high, containing four, six, and four tuples, respectively. To compute the gain ratio of income, we first use **Equation 4** to obtain

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right)$$

$$= 1.557.$$

From **Example 1**, we have Gain (income) = 0.029. Therefore, Gain Ratio (income) = 0.029/1.557 = **0.019.**

4.10.1.3 Gini Index:

Gini index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. If all the elements belong to a single class, then it can be called pure. The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes. A Gini Index of 0.5 denotes equally distributed elements into some classes.

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

Where *pi is* the probability of an object being classified to a particular class.

Induction of a decision tree using the Gini index. Let D be the training data shown earlier in **Table 1**, where there are **nine (9) tuples** belonging to the class buys computer = ***yes*** and the remaining **five (5) tuples** belong to the class buys computer = ***no***. A (root) node *N* is created for the tuples in *D*. We use above equation for the Gini index to compute the impurity of *D*.

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

4.3.2 Scalability and Decision Tree Induction:

In data mining applications, very large training sets of millions of tuples are common. Most often, the training data will not fit in memory! Therefore, decision tree construction becomes inefficient due to swapping of the training tuples in and out of main and cache memories. More scalable approaches, capable of handling training data that are too large to fit in memory, are required. Earlier strategies to "save space" included discretizing continuous-valued attributes and sampling data at each node. These techniques, however, still assume that the training set can fit in memory.

Several scalable decision tree induction methods have been introduced in recent studies. Rainforest, for example, adapts to the amount of main memory available and applies to any decision tree induction algorithm. The method maintains an AVC-set (where "AVC" stands for "Attribute-Value, Class label") for each attribute, at each tree node, describing the training tuples at the node. The AVC-set of an attribute A at node N gives the class label counts for each value of A for the tuples at N. following figure shows AVC sets for the tuple data of **Table 1**

age	Buy Computer	
	yes	no
youth	2	3
Middle-age	4	0
Senior	3	2

income	Buy Computer	
	yes	no
low	3	1
medium	4	2
high	2	2

company	Buy Computer	
	yes	no
yes	6	1
no	3	4

credit_rating	Buy Computer	
	yes	no
fair	6	2
excellent	3	3

Figure 4 Data structures to hold aggregate information regarding the training data to improving the scalability of decision tree induction.

4.3.3 Tree Pruning:

After finishing a decision tree built, many branches will reflect anomalies in the training data due to noise. Tree pruning methods use statistical measures to remove the least-reliable branches. An ***unpruned tree*** (figure 2) and a pruned version of it are shown in following figure 5. Pruned trees tend to be smaller and less complex and, thus, easier to comprehend. They are usually faster and better at correctly classifying independent test data (i.e., of previously unseen tuples) than unpruned trees.

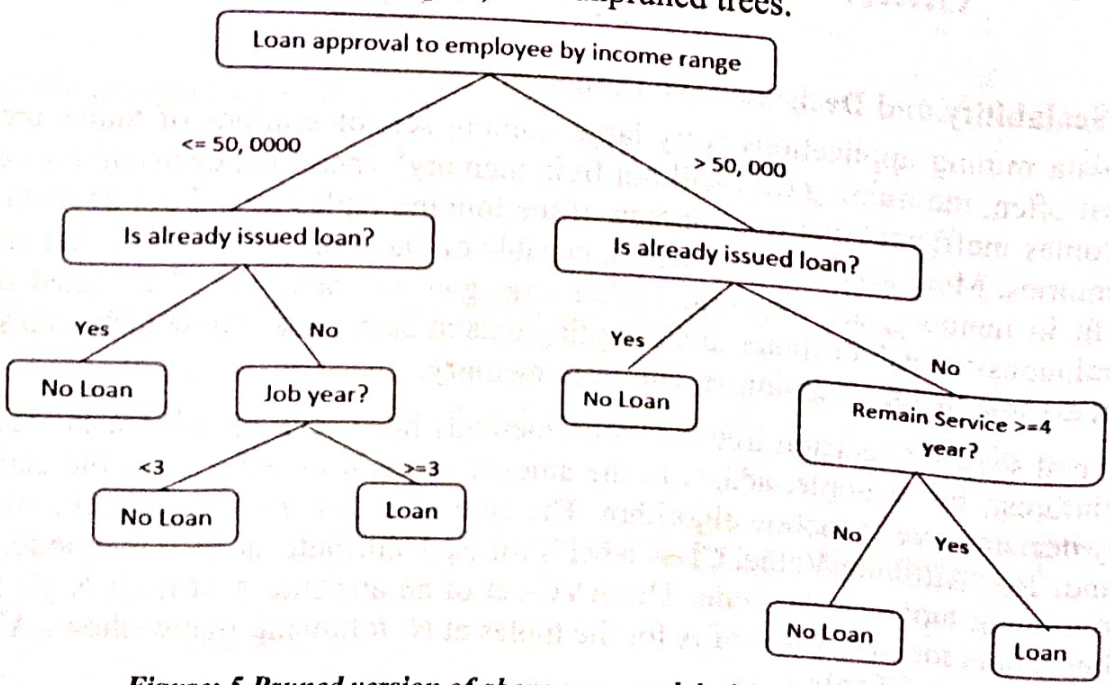


Figure: 5 Pruned version of above unpruned decision tree of figure 2

There are two common approaches to tree pruning: pre-pruning and post-pruning.

Pre-pruning and post-pruning. In the **pre-pruning** approach, a tree is "pruned" by halting its construction early (*e.g., by deciding not to further split or partition the subset of training tuples at a given node*). Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.

The second and more common approach is **post-pruning**, which removes sub-trees from a "fully grown" tree. A sub-tree at a given node is pruned by removing its branches and replacing it with a leaf. The leaf is labeled with the most frequent class among the sub-tree being replaced. For example, notice the sub-tree at node "*Valid Service Record?*" and "*Remain Installment less than 3 year*" in the unpruned tree of Figure 8.6. Suppose that the most common class within this sub-tree is "job year?" In the pruned version of the tree, the sub-tree in question is pruned by replacing it with the leaf "*job year*?" and "*Remain Installment less than 3 year*"

Decision trees can suffer from **repetition** and **replication** in following figure, making them overwhelming to interpret. *Repetition* occurs when an attribute is repeatedly tested along a given branch of the tree (e.g., *"is already issued loan?"* followed by *"issued loan?"*). **In *replication*, duplicate sub-trees exist within the tree**. These situations can obstruct the accuracy and comprehensibility of a decision tree. The use of multivariate splits (splits based on a combination of attributes) can prevent these problems.

4.11 What is cluster Analysis?

Cluster analysis or simply **clustering** is the process of partitioning a set of abstract data objects (or observations) into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other **clusters**. The set of clusters resulting from a cluster analysis can be referred to as a **clustering**. In this context, different clustering methods may generate different clustering on the same data set by the clustering algorithm. Therefore, clustering is useful in that it can lead to the discovery of previously unknown groups within the data

Cluster analysis used in many applications such as business intelligence, image pattern recognition, Web search, biology, and security. In business intelligence, clustering can be used to organize a large number of customers into groups, where customers within a group share strong similar characteristics. This facilitates the development of business strategies for enhanced customer relationship management. Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their similarity. Data clustering research include data mining, statistics, machine learning, spatial database technology, information retrieval, Web search, biology, and marketing.

Cluster analysis tools based on k-means, k-medoids, and several other methods also have been built into many statistical analysis software packages or systems, such as S-Plus, SPSS, and SAS. Here **classification is known as supervised learning** in machine learning because the class label information is given, that is, the learning algorithm is supervised in that it is told the class membership of each training tuple. **Clustering is known as unsupervised learning** because the class label information is not present. For this reason, clustering is a form of *learning by observation*, rather than *learning by examples*.

Note that,

- A cluster of data objects can be treated as one group.
- While doing cluster analysis, we first partition the set of data into groups based on data similarity and then assign the labels to the groups.
- Advantage of clustering above classification is that, it is adjustable to changes and helps single out useful features that distinguish different groups.

4.12 Overview of basic clustering methods:

Following are the basic methods of the clustering:

4.12.1 Partitioning methods:

In Partitioning methods given a set of ***n*** objects, a partitioning method constructs ***k*** partitions of the data, where each partition represents a cluster and $k \leq n$. That is, it divides the data into *k* groups such that each group must contain at least one object. In other words, partitioning methods conduct one-level partitioning on data sets. The basic partitioning methods typically adopt exclusive cluster separation. That is, each object must belong to exactly one group.

Most partitioning methods are distance-based. Given ***k,*** the number of partitions to construct, a partitioning method creates an initial partitioning. It then uses an **iterative relocation technique** that attempts to improve the partitioning by moving objects from one group to another. The general criterion of a good partitioning is that objects in the same cluster are **"close"** or related to each other, whereas objects in different clusters are **"far apart"** or very different. There are various kinds of other criteria for judging the quality of partitions. Traditional partitioning methods can be extended for subspace clustering, rather than searching the full data space. This is useful when there are many attributes and the data are sparse

Most partitioning applications adopt popular heuristic methods, such as ***greedy approaches*** like the ***k-means*** and the ***k-medoids*** **algorithms**, which progressively improve the clustering quality and approach a local optimum. These heuristic clustering methods work well for finding spherical-shaped clusters in small- to medium-size databases.

4.12.2 Hierarchical methods:

A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either ***agglomerative or divisive***, based on how the hierarchical decomposition is formed.

The ***agglomerative approach***, also called the bottom-up approach, starts with each object forming a separate group. It successively merges the objects or groups close to one another, until all the groups are merged into one (the topmost level of the hierarchy), or a termination condition holds.

The ***divisive approach***, also called the top-down approach, starts with all the objects in the same cluster. In each successive iteration, a cluster is split into smaller clusters, until eventually each object is in one cluster, or a termination condition holds.

Hierarchical clustering methods can be ***distance-based or density- and continuity based.*** Various extensions of hierarchical methods consider clustering in subspaces as well.

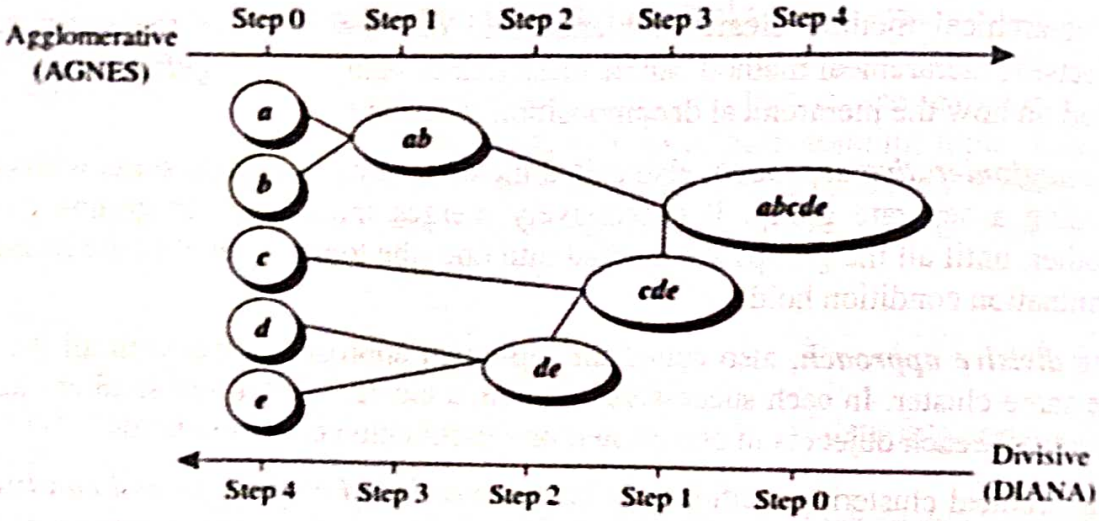
4.5.2.1 Agglomerative versus Divisive Hierarchical Clustering:

A hierarchical clustering method can be either agglomerative or divisive, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or top-down (splitting) fashion. Let's have a closer look at these strategies.

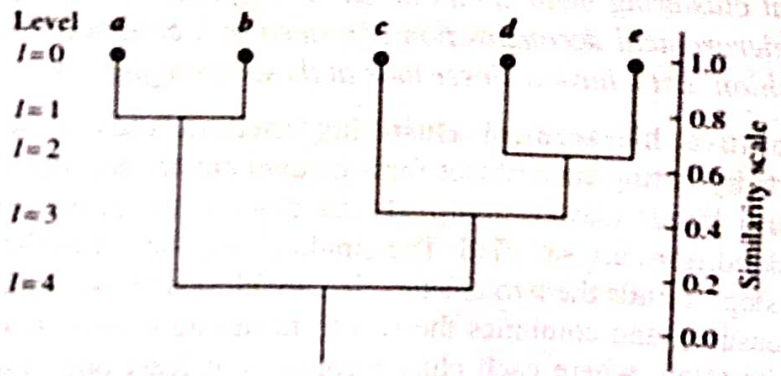
An **agglomerative hierarchical clustering method** uses a ***bottom-up strategy.*** It typically starts by letting each object form its own cluster and iteratively merges clusters into larger and larger clusters, until all the objects are in a single cluster or certain termination conditions are satisfied. The single cluster becomes the hierarchy's root. For the merging step, it finds the two clusters that are closest to each other (according to some similarity measure), and combines the two to form one cluster. Because two clusters are merged per iteration, where each cluster contains at least one object, an agglomerative method requires at most n iterations.

A **divisive hierarchical clustering method** employs a ***top-down strategy***. It starts by placing all objects in one cluster, which is the hierarchy's root. It then divides the root cluster into several smaller sub clusters, and recursively partitions those clusters into smaller ones. The partitioning process continues until each cluster at the lowest level is coherent enough—either containing only one object, or the objects within a cluster are sufficiently similar to each other.

In either agglomerative or divisive hierarchical clustering, a user can specify the desired number of clusters as a termination condition.



Agglomerative and divisive hierarchical clustering on data objects {a,b,c,d,e}.



Dendrogram representation for hierarchical clustering of data objects {a,b,c,d,e}.

Figure 6 Agglomerative and divisive hierarchical clustering of data objects

The figures 5 denote agglomerative versus divisive hierarchical clustering. *above figures* shows the application of AGNES (AGglomerative NESting), an agglomerative hierarchical clustering method, and DIANA (DIvisive ANAlysis), a divisive hierarchical clustering method, on a data set of five objects, {a, b, c, d, e}.

4.5.3 Density-based methods:

Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty in discovering clusters of arbitrary shapes. Other clustering methods have been developed based on the notion of *density.*

Their general idea is to continue growing a given cluster as long as the density (number of objects or data points) in the "**neighborhood**" exceeds some threshold. For example, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise or outliers and discover clusters of arbitrary shape.

Density-based methods can divide a set of objects into multiple exclusive clusters, or a hierarchy of clusters. Typically, density-based methods consider exclusive clusters only, and do not consider fuzzy clusters. Moreover, density-based methods can be extended from full space to subspace clustering.

DBSCAN: (Density-based spatial clustering of applications with noise) Density-Based Clustering Based on Connected Regions with High Density.

"How can we find dense regions in density-based clustering?" The density of an object o can be measured by the number of objects close to **o**. **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) finds core objects, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters.

"How does DBSCAN quantify the neighborhood of an object?" A user-specified parameter $\epsilon > 0$ is used to specify the radius of a neighborhood we consider for every object. The ϵ-neighborhood of an object o is the space within a radius ϵ centered at **o**. Due to the fixed neighborhood size parameterized by ϵ, the density of a neighborhood can be measured simply by the number of objects in the neighborhood. To determine whether a neighborhood is dense or not, DBSCAN uses another user-specified parameter, *MinPts*, which specifies the density threshold of dense regions. An object is a core object if the ϵ-neighborhood of the object contains at least *MinPts* objects. Core objects are the pillars of dense regions.

4.5.3.1 DBSCAN algorithm requires two parameters:

eps: It defines the neighborhood around a data point for example, if the distance between two points is lower or equal to 'eps' then they are considered as neighbors. If the *eps* value is chosen too small then large part of the data will be considered as outliers.

MinPts: Minimum number of neighbors (data points) within *eps* radius. Larger the dataset, the larger value of *MinPts* must be chosen. As a general rule, the minimum *MinPts* can be derived from the number of dimensions *D* in the dataset as, *MinPts* >= *D*+1. The minimum value of *MinPts* must be chosen at least 3.

Density based spatial clustering of applications with noise

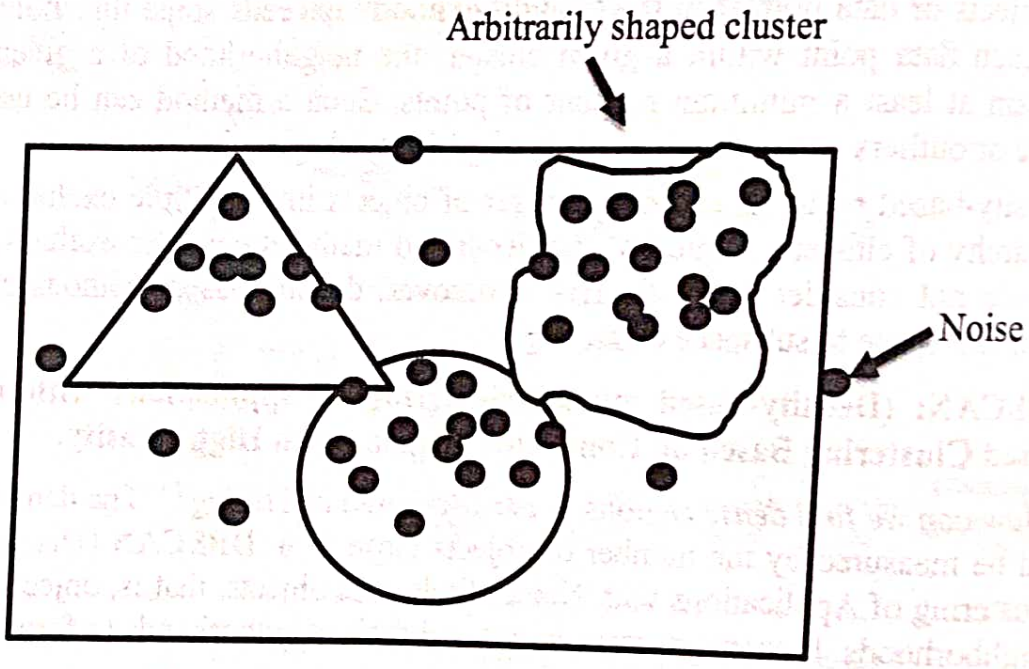


Figure: 7 Density based spatial clustering of applications with noise

In this algorithm, we have 3 types of data points

Core Point: A point is a core point if it has more than MinPts points within eps.

Border Point: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.

Noise or outlier: A point which is not a core point or border point.

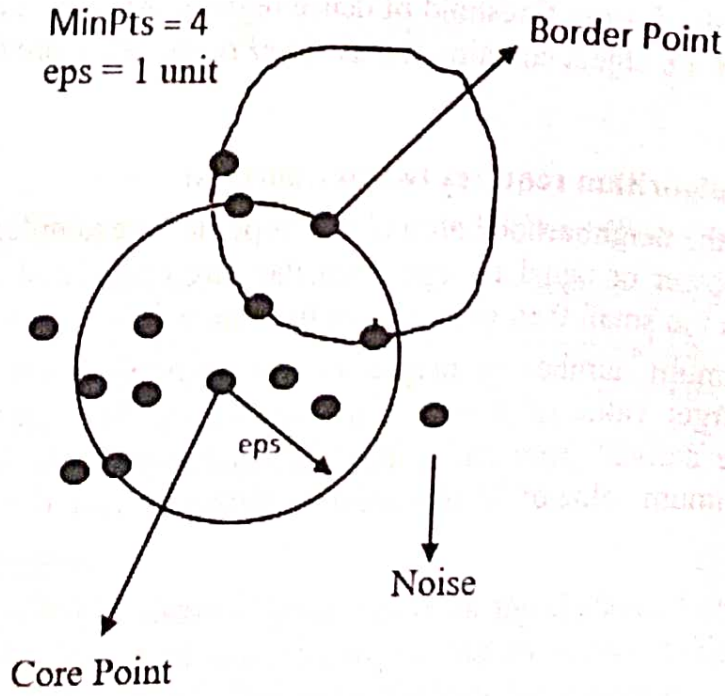


Figure: 8 Three types of data points

4.5.3.2 DBSCAN clustering algorithm:

```
DBSCAN (dataset, eps, MinPts)
{
    # Cluster index
    C = 1
    For each unvisited point p in dataset
     {
       Mark p as visited    # find neighbours
        Neighbours N = find the neighbouring points of p

            If |N| >= MinPts:
            N = N U N'
              If p' is not a member of any cluster
              add p' to cluster C
        }
}
```

4.5.3.3 Grid-based methods:

Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All the clustering operations are performed on the **grid structure** (i.e., on the **quantized space**).

The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.

Using grids is often an efficient approach to many spatial data mining problems, including clustering. Therefore, grid-based methods can be integrated with other clustering methods such as density-based methods and hierarchical methods.

Method	General Characteristics
Partitioning methods	➢ Find mutually exclusive clusters of spherical shape ➢ Distance-based – May use mean or medoid (etc.) to represent cluster center ➢ Effective for small- to medium-size data sets
Hierarchical methods	➢ Clustering is a hierarchical decomposition (i.e., multiple levels)

	➢ Cannot correct erroneous merges or splits ➢ May incorporate other techniques like micro clustering or consider object "linkages"
Density-based methods	➢ Can find arbitrarily shaped clusters ➢ Clusters are dense regions of objects in space that are separated by low-density regions ➢ Cluster density: Each point must have a minimum number of points within its "neighborhood" ➢ May filter out outliers
Grid-based methods	➢ Use a multi-resolution grid data structure ➢ Fast processing time (typically independent of the number of data objects, yet dependent on grid size)

4.13 Partitioning Method

4.6.1 K Means: A Centroid based technique

Suppose a data set D, contains n objects in Euclidean space. Partitioning methods distribute the objects in D into k clusters, C1,...,Ck , that is, Ci ⊂ D and Ci ∩ Cj = ∅ for ($1 \le i, j \le k$). An objective function is used to assess the partitioning quality so that objects within a cluster are similar to one another but dissimilar to objects in other clusters.

A centroid-based partitioning technique uses the centroid of a cluster, Ci, to represent that cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean or medoid of the objects (or points) assigned to the cluster. The difference between an object $p \in Ci$ and ci , the representative of the cluster, is measured by dist (p,ci), where dist (x, y) is the Euclidean distance between two points x and y. The quality of cluster Ci can be measured by the within cluster variation, which is the sum of squared error between all objects in Ci and the centroid ci, defined as

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2,$$

Where E is the sum of the squared error for all objects in the data set; p is the point in space representing a given object; and ci is the centroid of cluster Ci (both p and ci are multidimensional). In other words, for each object in each cluster, the distance from the object to its cluster center is squared, and the distances are summed. This objective function tries to make the resulting k clusters as compact and as separate as possible.

"How does the k-means algorithm work?" The k-means algorithm defines the centroid of a cluster as the mean value of the points within the cluster. It proceeds as follows. First, it randomly selects k of the objects in D, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the Euclidean distance between the object and the cluster mean. The k-means algorithm then iteratively improves the within-cluster variation. For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration. All the objects are then reassigned using the updated means as the new cluster centers. The iterations continue until the assignment is stable,

4.6.1.1 K-means Algorithm:

The k-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

K: the number of clusters,

D: a data set containing n objects.

Output:

A set of *k* clusters.

Method:

1. arbitrarily choose k objects from D as the initial cluster centers;
2. **repeat**
3. (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
4. Update the cluster means, that is, calculate the mean value of the objects for each cluster;
5. **until** no change;

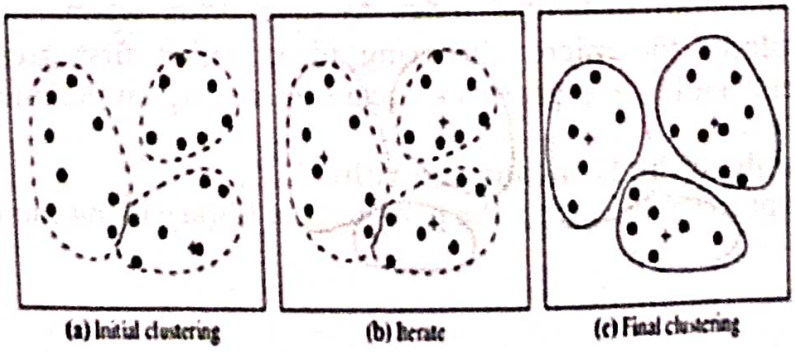


figure 9 Clustering of a set of objects using the k-means method; for (b) update cluster centers and reassign objects accordingly (the mean of each cluster is marked by a +).

Clustering by k-means partitioning. Consider a set of objects located in 2-D space, as depicted in above figure (a). Let k = 3, that is, the user would like the objects to be partitioned into three clusters.

According to the algorithm in above figure 9, we arbitrarily choose three objects as the three initial cluster centers, where cluster centers are marked by a +. Each object is assigned to a cluster based on the cluster center to which it is the nearest. Such a distribution forms silhouettes encircled by dotted curves, as shown in above figure (a).

Next, the cluster centers are updated. That is, the mean value of each cluster is recalculated based on the current objects in the cluster. Using the new cluster centers, the objects are redistributed to the clusters based on which cluster center is the nearest. Such a redistribution forms new silhouettes encircled by dashed curves, as shown in above figure **(b).**

This process iterates, leading to above figure **(c).** The process of iteratively reassigning objects to clusters to improve the partitioning is referred to as iterative relocation. Eventually, no reassignment of the objects in any cluster occurs and so the process terminates. The resulting clusters are returned by the clustering process.

The k-means method can be applied only when the mean of a set of objects is defined. This may not be the case in some applications such as when data with nominal attributes are involved. The k-modes method is a variant of k-means, which extends the k-means paradigm to cluster nominal data by replacing the means of clusters with modes. It uses new dissimilarity measures to deal with nominal objects and a frequency-based method to update modes of clusters. The k-means and the k-modes methods can be integrated to cluster data with mixed numeric and nominal values.

"How can we make the k-means algorithm more scalable?"

One approach to making the k-means method more efficient on large data sets is to use a good-sized set of samples in clustering. Another is to employ a filtering approach that uses a spatial hierarchical data index to save costs when computing means. A third approach explores the micro clustering idea, which first groups nearby objects into "micro clusters" and then performs k-means clustering on the micro clusters.

4.6.1.2. Example of the K-means algorithm:

Suppose we want to set the group the persons to a library using their age (one-dimensional space)

as follows, Here,

n = 19

15, 15, 16, 19, 19, 20, 20, 21, 22, 28, 35, 40, 41, 42, 43, 44, 60, 61, 65

Initial clusters (*Random centroid or average*)

k = 2

c1 = 16

c2 = 22

Distance 1 = xi-c1

Distance 2 = xi-c2

Iteration 1:

c1 = 15.33

c2 = 36.25

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	16	22	1	7	1	(Mean) 15.33
15	16	22	1	7	1	
16	16	22	0	6	1	
19	16	22	9	3	2	
19	16	22	9	3	2	
20	16	22	16	2	2	
20	16	22	16	2	2	
21	16	22	25	1	2	
22	16	22	36	0	2	(Mean) 36.25
28	16	22	12	6	2	
35	16	22	19	13	2	
40	16	22	24	18	2	
41	16	22	25	19	2	
42	16	22	26	20	2	
43	16	22	27	21	2	
44	16	22	28	22	2	
60	16	22	44	38	2	
61	16	22	45	39	2	
65	16	22	49	43	2	

Iteration 2:

c1= 18.56

c2= 45.90

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	15.33	36.25	0.33	21.25	1	
15	15.33	36.25	0.33	21.25	1	
16	15.33	36.25	0.67	20.25	1	
19	15.33	36.25	3.67	17.25	1	(Mean) 18.56
19	15.33	36.25	3.67	17.25	1	
20	15.33	36.25	4.67	16.25	1	
20	15.33	36.25	4.67	16.25	1	
21	15.33	36.25	5.67	15.25	1	
22	15.33	36.25	6.67	14.25	1	
28	15.33	36.25	12.67	8.25	2	
35	15.33	36.25	19.67	1.25	2	
40	15.33	36.25	24.67	3.75	2	
41	15.33	36.25	25.67	4.75	2	
42	15.33	36.25	26.67	5.75	2	(Mean) 45.9
43	15.33	36.25	27.67	6.75	2	
44	15.33	36.25	28.67	7.75	2	
60	15.33	36.25	44.67	23.75	2	
61	15.33	36.25	45.67	24.75	2	
65	15.33	36.25	49.67	28.75	2	

Iteration 3:

c1 = 19.50

c2 = 47.89

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	18.56	45.9	3.56	30.9	1	
15	18.56	45.9	3.56	30.9	1	
16	18.56	45.9	2.56	29.9	1	
19	18.56	45.9	0.44	26.9	1	
19	18.56	45.9	0.44	26.9	1	(Mean)
20	18.56	45.9	1.44	25.9	1	19.50
20	18.56	45.9	1.44	25.9	1	
21	18.56	45.9	2.44	24.9	1	
22	18.56	45.9	3.44	23.9	1	
28	18.56	45.9	9.44	17.9	1	
35	18.56	45.9	16.44	10.9	2	
40	18.56	45.9	21.44	5.9	2	
41	18.56	45.9	22.44	4.9	2	
42	18.56	45.9	23.44	3.9	2	(Mean)
43	18.56	45.9	24.44	2.9	2	47.89
44	18.56	45.9	25.44	1.9	2	
60	18.56	45.9	41.44	14.1	2	
61	18.56	45.9	42.44	15.1	2	
65	18.56	45.9	46.44	19.1	2	

Iteration 4:

c1 = 19.50

c2 = 47.89

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	19.5	47.89	4.50	32.89	1	
15	19.5	47.89	4.50	32.89	1	
16	19.5	47.89	3.50	31.89	1	(Mean)
19	19.5	47.89	0.50	28.89	1	19.50
19	19.5	47.89	0.50	28.89	1	

20	19.5	47.89	0.50	27.89	1	
20	19.5	47.89	0.50	27.89	1	
21	19.5	47.89	1.50	26.89	1	
22	19.5	47.89	2.50	25.89	1	
28	19.5	47.89	8.50	19.89	1	
35	19.5	47.89	15.50	12.89	2	(Mean) 47.89
40	19.5	47.89	20.50	7.89	2	
41	19.5	47.89	21.50	6.89	2	
42	19.5	47.89	22.50	5.89	2	
43	19.5	47.89	23.50	4.89	2	
44	19.5	47.89	24.50	3.89	2	
60	19.5	47.89	40.50	12.11	2	
61	19.5	47.89	41.50	13.11	2	
65	19.5	47.89	45.50	17.11	2	

Here, No change between Centroid of iterations 3 and iterations 4 By using clustering so, stop the Iteration further, ***2 groups Identified as*** **15-28 and 35-65.**

4.14 Data Mining Applications:

4.7.1 Data Mining for Financial Data Analysis:

Most banks and financial institutions offer a wide variety of banking, investment, and credit services, some also offer insurance and stock investment service. Financial data collected in the banking and financial industry are often relatively complete, reliable, and of high quality, which facilitates systematic data analysis and data mining. Here we present a few typical cases.

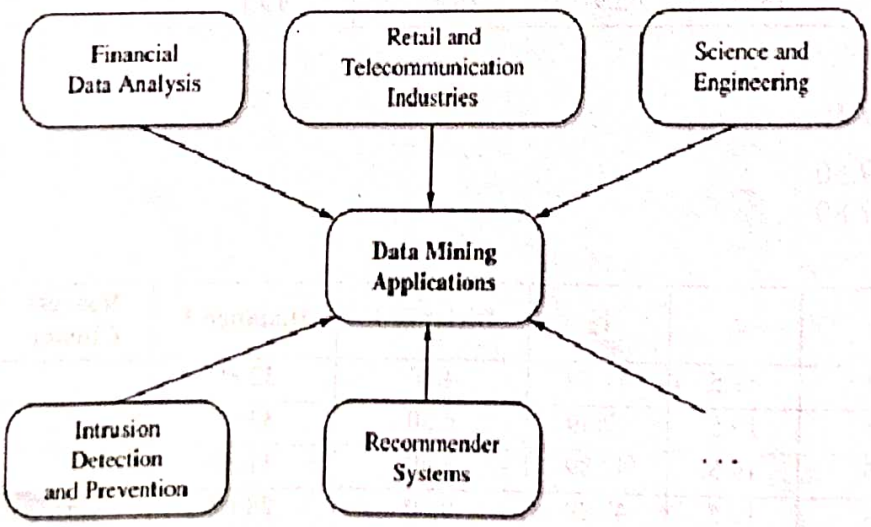


Figure 10: Common data mining application domains

- **Design and construction of data warehouses for multidimensional data analysis and data mining:** Multidimensional data analysis methods should be used to analyze the general properties of such data. For example, a company's financial officer may want to view the debt and revenue changes by month, region, and sector, and other factors, along with maximum, minimum, total, average, trend, deviation, and other statistical information. Data warehouses, data cubes, characterization and class comparisons, clustering, and outlier analysis will all play important roles in financial data analysis and mining.

- **Classification and clustering of customers for targeted marketing:** Classification and clustering methods can be used for customer group identification and targeted marketing. Customers with similar behaviors regarding loan payments may be identified by multidimensional clustering techniques. These can help identify customer groups, associate a new customer with an appropriate customer group.

- **Detection of money laundering and other financial crimes:** To detect money laundering and other financial crimes, it is important to integrate information from multiple, heterogeneous databases e.g., bank transaction databases and federal or state crime history databases, as long as they are potentially related to the study. Multiple data analysis tools can then be used to detect unusual patterns, such as large amounts of cash flow at certain periods, by certain groups of customers. Useful tools include data visualization, linkage and information network analysis tools to identify links among different customers and activities, classification tools (to filter unrelated attributes and rank the highly related ones), clustering tools (to group different cases), outlier analysis tools (to detect unusual amounts of fund transfers or other activities), and sequential pattern analysis tools (to characterize unusual access sequences). These tools may identify important relationships and patterns of activities and help investigators focus on suspicious cases for further detailed examination.

4.7.2 Data Mining for Retail and Telecommunication Industries:

Today, most major chain that customers can make purchases online. Some businesses, such as Amazon.com exist solely online, without any physical store locations. Retail data provide a rich source for data mining. Retail data mining can help identify customer buying behaviors, discover customer shopping patterns and trends, improve the quality of customer service, achieve better customer retention and satisfaction, enhance goods consumption ratios, design more effective goods transportation and distribution policies, and reduce the cost of business.

A few examples of data mining in the retail industry are outlined as follows:

- **Design and construction of data warehouses:** Because retail data cover a wide range including sales, customers, employees, goods transportation, consumption, and services, there can be many ways to design a data warehouse for this industry. The levels of detail

to include can vary considerably. The outcome of preliminary data mining exercises can be used to help guide the design and development of data warehouse structures.

- **Multidimensional analysis of sales, customers, products, time, and region:** The retail industry requires timely information regarding customer needs, product sales, trends, and fashions, as well as the quality, cost, profit, and service of commodities. It is therefore important to provide powerful multidimensional analysis and visualization tools, including the construction of sophisticated data cubes according to the needs of data analysis.
- **Customer retention—analysis of customer loyalty:** Here, Customer loyalty and purchase trends can be analyzed systematically. Goods purchased at different periods by the same customers can be grouped into sequences. Sequential pattern mining can then be used to investigate changes in customer consumption or loyalty and suggest adjustments on the pricing and variety of goods to help retain customers and attract new ones.
- **Fraudulent analysis and the identification of unusual patterns:** It is important to identify potentially fraudulent users and their atypical usage patterns; second, detect attempts to gain fraudulent entry or unauthorized access to individual and organizational accounts; and third discover unusual patterns that may need special attention. Many of these patterns can be discovered by multidimensional analysis, cluster analysis, and outlier analysis.

4.7.3 Data Mining in Science and Engineering:

- **Data warehouses and data preprocessing:** Data preprocessing and data warehouses are serious for information exchange and data mining. Creating a warehouse often requires finding means for resolving unpredictable or incompatible data collected in multiple environments and at different time periods. Methods are needed for integrating data from heterogeneous sources and for identifying events.
- **Mining in complex data types:** Scientific data sets are heterogeneous in nature, such as multimedia data, as well as data with sophisticated, deeply hidden semantics e.g., genomic and proteomic data. Robust and dedicated analysis methods are needed for handling spatiotemporal data, biological data, related concept hierarchies, and complex semantic relationships. For example, in bioinformatics, a research problem is to identify regulatory influences on genes. Gene regulation refers to how genes in a cell are switched on or off to determine the cell's functions. Thus, to understand a biological process we need to identify the participating genes and their regulators. This requires the development of sophisticated data mining methods to analyze large biological data sets for clues about regulatory influences on specific genes, by finding DNA segments ("regulatory sequences") mediating such influence.
- **Graph-based and network-based mining:** graphs and networks may be used to capture many of the spatial, topological, geometric, biological, and other relational characteristics present in scientific data sets. In graph or network modeling, each object to be mined is

represented by a vertex in a graph, and edges between vertices represent relationships between objects.Graphs can be used to model chemical structures, biological pathways, and data generated by numeric simulations such as fluid-flow simulations. The success of graph or network modeling, however, depends on improvements in the scalability and efficiency of many graph-based data mining tasks such as classification, frequent pattern mining, and clustering.

- **Domain-specific knowledge:** High-level graphical user interfaces and visualization tools are required for scientific data mining systems. These should be integrated with existing domain-specific data and information systems to guide researchers and general users in searching for patterns, interpreting and visualizing discovered patterns, and using discovered knowledge in their decision making.
- **Data mining in computer science**, technically, data mining is the computational process of analyzing data from different perspective, dimensions, angles and categorizing/summarizing it into meaningful information. Data Mining can be applied to any type of data e.g. Data Warehouses, Transactional Databases, Relational Databases, Multimedia Databases, Spatial Databases, Time-series Databases, World Wide Web.
- **Data mining in engineering**, many engineering processes need real-time responses, and so mining data streams in real time often becomes a critical component. Such communication exists in many forms, including news, blogs, articles, web pages, online discussions, product reviews, twitters, messages, advertisements, and communications, both on the Web and in various kinds of social networks.

4.7.4 Data Mining for Intrusion Detection and Prevention:

The majority of intrusion detection and prevention systems use either signature based detection or anomaly-based detection.

4.7.4.1 Signature-based detection: This method of detection utilizes signatures, which are attack patterns that are preconfigured and predetermined by domain experts. A signature-based intrusion prevention system monitors the network traffic for matches to these signatures. Once a match is found, the intrusion detection system will report the anomaly and an intrusion prevention system will take additional appropriate actions. Note that since the systems are usually quite dynamic, the signatures need to be updated laboriously whenever new software versions arrive or changes in network configuration or other situations occur. Another drawback is that such a detection mechanism can only identify cases that match the signatures. That is, it is unable to detect new or previously unknown intrusion tricks.

4.7.4.2 Anomaly-based detection: This method builds models of normal network behavior (called profiles) that are then used to detect new patterns that significantly deviate from the profiles. Such deviations may represent actual intrusions or simply be new behaviors that need to be added to the profiles. The main advantage of anomaly detection is that it

may detect novel intrusions that have not yet been observed. Typically, a human analyst must sort through the deviations to ascertain which represent real intrusions. A limiting factor of anomaly detection is the high percentage of false positives. New patterns of intrusion can be added to the set of signatures to enhance signature-based detection.

Data mining methods can help an intrusion detection and prevention system to enhance its performance in various ways as follows.

New data mining algorithms for intrusion detection: Data mining algorithms can be used for both signature-based and anomaly-based detection. In signature-based detection, training data are labeled as either "normal" or "intrusion." A classifier can then be derived to detect known intrusions.

Association, correlation, and discriminative pattern analyses help select and build discriminative classifiers: Association, correlation, and discriminative pattern mining can be applied to find relationships between system attributes describing the network data. Such information can provide insight regarding the selection of useful attributes for intrusion detection. New attributes derived from aggregated data may also be helpful such as summary counts of traffic matching a particular pattern.

Analysis of stream data: Due to the transient and dynamic nature of intrusions and malicious attacks, it is crucial to perform intrusion detection in the data stream environment. Moreover, an event may be normal on its own, but considered malicious if viewed as part of a sequence of events.

Distributed data mining: Intrusions can be launched from several different locations and targeted to many different destinations. Distributed data mining methods may be used to analyze network data from several network locations to detect these distributed attacks.

Visualization and querying tools: Visualization tools should be available for viewing any anomalous patterns detected. Such tools may include features for viewing associations, discriminative patterns, clusters, and outliers. Intrusion detection systems should also have a graphical user interface that allows security analysts to pose queries regarding the network data or intrusion detection results.

Exercises

❖ Explain in details of the below questions:

1. Define decision tree induction and its pruning with proper example.
2. Define Selection Measures (*information gain* and *Gain ratio*) with proper example (training data).
3. Explain K Means: A Centroid based technique.
4. What is cluster?, Explain any three clustering methods.
5. What is classification? Explain in your words.
6. Explain any three data mining applications.

❖ Solve the below MCQs :

1. Which one from this can be considered as the final output of the hierarchal type of clustering?
   (a) A tree which displays how the close thing are to each other
   (b) Assignment of each point to clusters
   (c) Finalize estimation of cluster centroids
   (d) All of this
2. Data Mining System Classification consists of?
   (a) Database Technology (b) Machine Learning
   (c) Information Science (d) All of this
3. Which one is not clustering method?
   (a) Partitioning (b) Relational
   (c) Hierarchical (d) Density based
4. Tree pruning is the processing cycle of
   (a) Abstraction (b) normalization
   (c) reduction (d) all of this
5. What are the functions of Data Mining?

(a) Association (b) Prediction

(c) Clustering (d) all of these

6. Classification is based on -
(a) Subjective Data (b) Objective data

(c) Selection of data (d) all of this

7. Decision tree are normalised by
(a) Pruning (b) indexing

(c) classification (d) test data

8. Leaf node of the decision tree always make a *decision* (True / False)

9. clustering is also called unsupervised learning (True / False)

10. classifier is used for classification and new data (True / False)

Answers

1. a
2. d
3. b
4. d
5. d
6. d
7. a
8. True
9. True
10. True

GUJARAT UNIVERSITY

April – 2021

B.C.A SEM – VI

CC- 308 Data Mining and Data Warehouse (New Course)

Time: 2:00 Hrs. **Model Paper** **Total Marks: 50**

Instruction: All Questions of **section I** carry equal marks.

Attempt any **two** Questions in **section I**

Question 5 in section II **is COMPULSORY,** Attempt any **Five.**

SECTION - I

Q. 1 A: What is Data Mining? List the KDD steps? 10

B: Which types of technologies are used in mining of data? 10

Q. 2 A: 1. Explain ETL (*Extraction, Transformation* and *Loading*) of data warehouse. 10

2. Explain data warehouse definition by W. H. Inmon.

B: Explain OLAP operations in brief. 10

Q. 3 A: Explain Apriori Algorithm with proper example. 10

B: Explain any two Data Reduction Strategies. 10

Q. 4 A: Define decision tree induction and its pruning with proper example. 10

B: Define Selection Measures (*information gain* and *Gain ratio*) with proper example (training data). 10

SECTION- II

Q.5 **Answers the following Questions (Any 5 x 2 marks each)** 10

1. Data warehouse is the subset of the data mining (True / False)

2. Classification and regression is based on
   (a) Training data (b) Test data
   (c) New data (d) Mata-data
3. Clustering is also called classification of data set (True / False)
4. OLAP stand for _____
5. Data are arranged in subject or department wise, is called
   (a) Data Cube (b) Mata-data
   (c) data Mart (d) data repository
6. Data cube is the part of the multi-dimensional data (True / False)
7. Data cleaning is process of
   (a) Remove noisy data (b) delete data
   (c) Update data (d) Sharp data
8. Mined data result may be based on
   (a) Tabular (b) graphical
   (c) Diagrammatical (d) all of these
9. Decision tree are normalised by
   (a) Pruning (b) indexing
   (c) classification (d) training data
10. clustering is called supervised learning (True / False)

☺ ☺ ☺ ☺ ☺