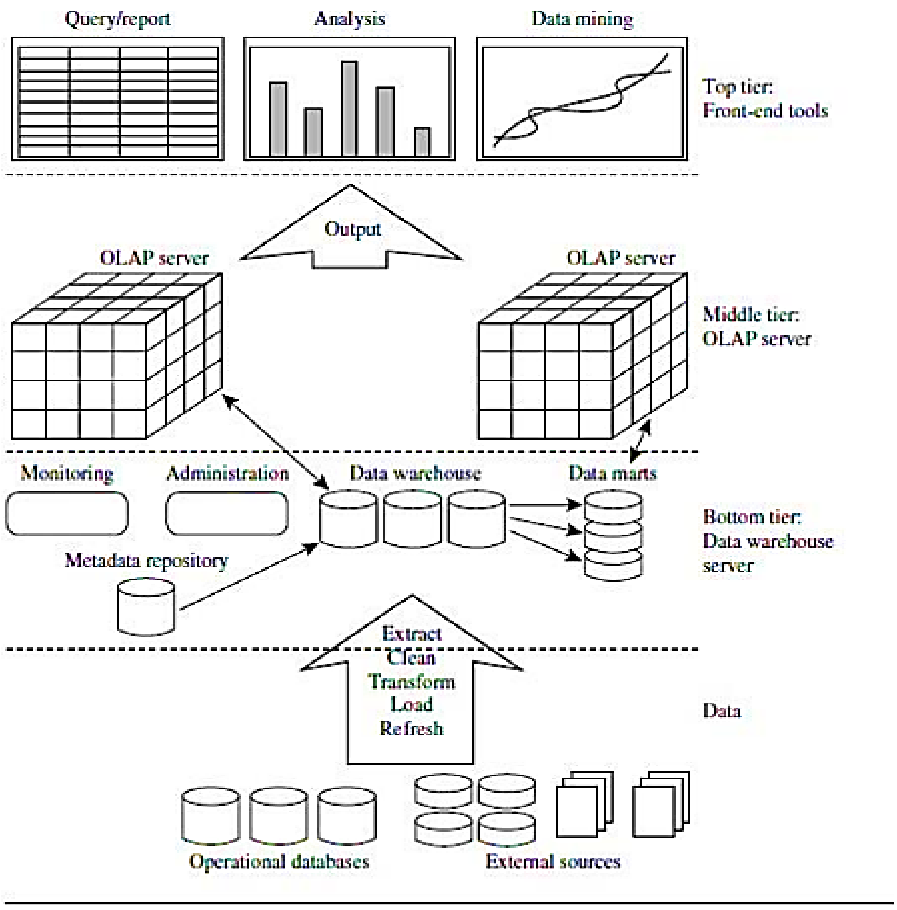
VTU Examinations – June 2022
Data Mining and Data Warehousing
Scheme of Evaluation

Module – 1

1 (a). With neat diagram explain three tier data warehouse.



- The bottom tier is a warehouse database server that is almost always a relational database system.
- A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server.
- The middle tier is an OLAP server that is typically implemented using either (1) a relational OLAP (ROLAP) model or (2) a multidimensional OLAP (MOLAP) model.
- The top tier is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools.

1 (b). List and explain Data Warehouse Models.

- There are three data warehouse models.
  Enterprise warehouse:
  • An enterprise warehouse collects all of the information about subjects spanning the entire organization.

• It provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cross- functional in scope.

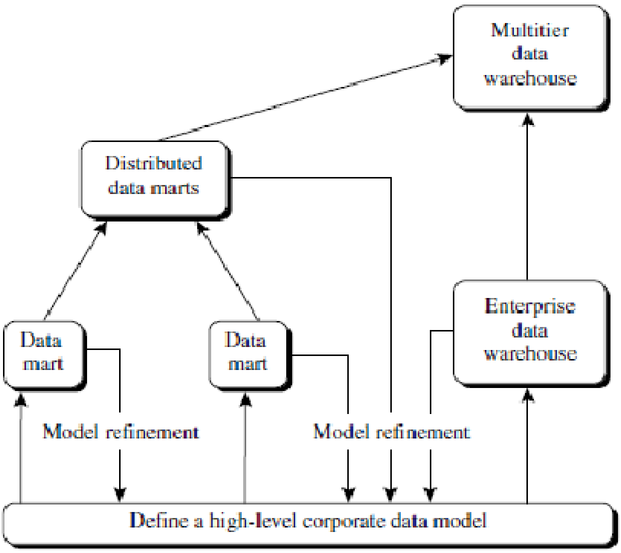
- **Data mart:**

  • A data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to specific selected subjects. For example, a marketing data mart may confine its subjects to customer, item, and sales. The data contained in data marts tend to be summarized.

- **Virtual warehouse:**

  • A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized.

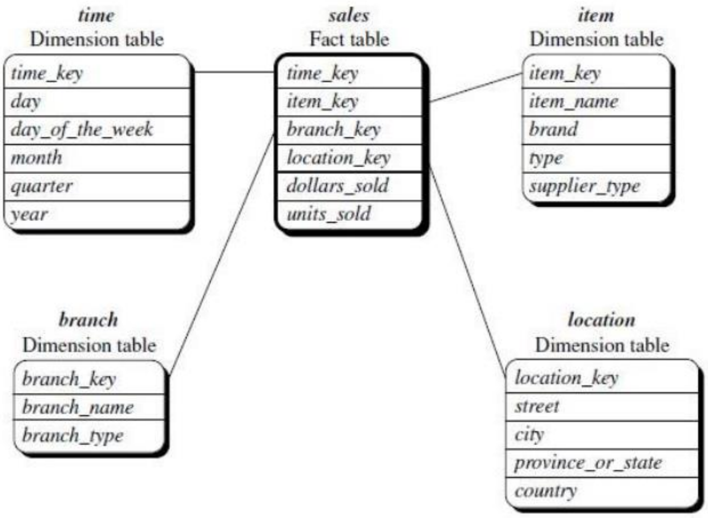
  • A virtual warehouse is easy to build but requires excess capacity on operational database servers.



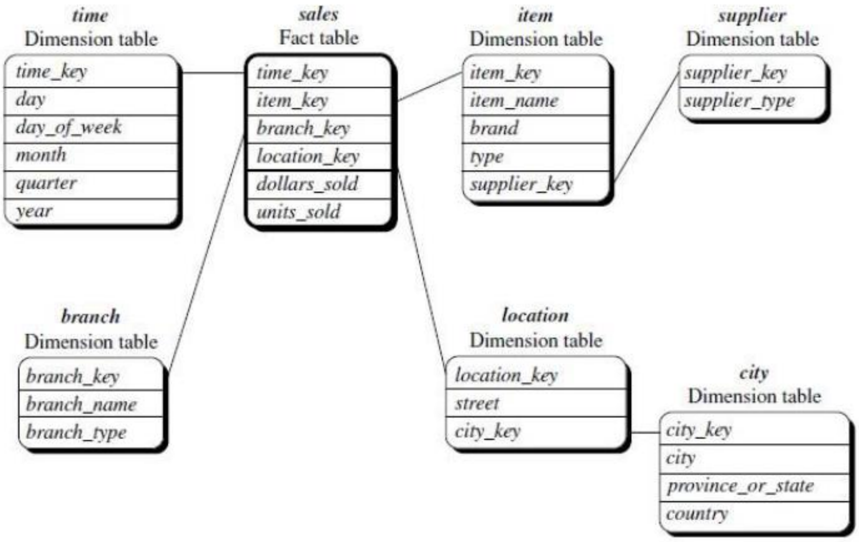
2 (a). With suitable example, explain star schema, Snow Flake Schema, Fact Constellation Schema for Multidimensional database.

Schemas for multidimensional data models

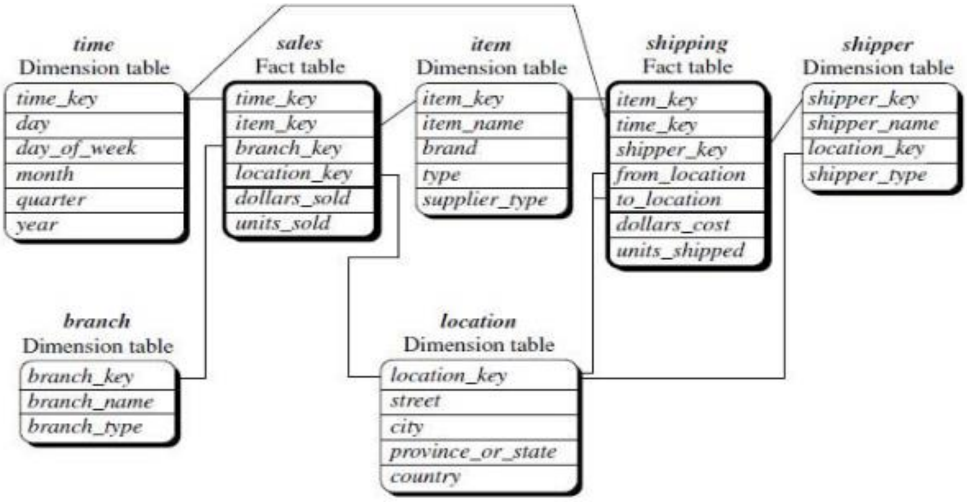
 Star schema: A fact table in the middle connected to a set of dimension tables



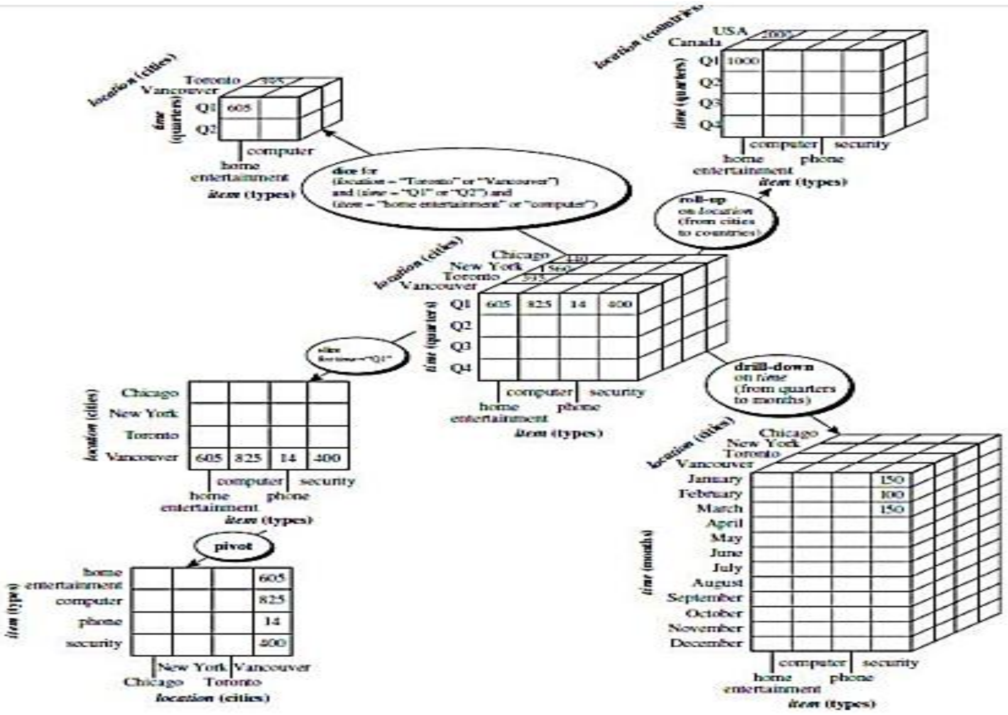
 Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake



 Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation.



2 (b). Explain OLAP operations with example.



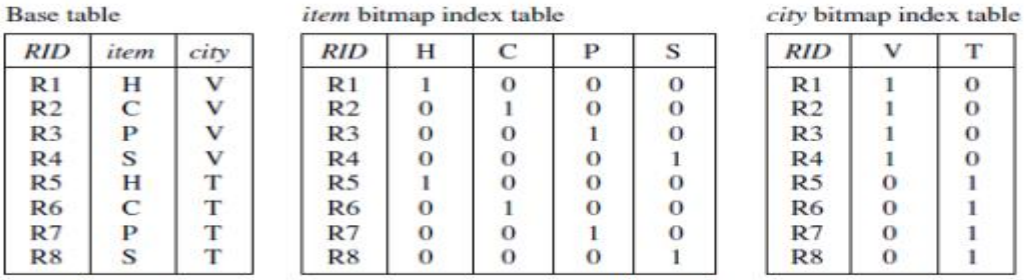
- The **roll-up** operation also called as the drill-up operation performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction.
- **Drill-down** can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions.
- The **slice** operation performs a selection on one dimension of the given cube, resulting in a subcube and the **dice** operation defines a subcube by performing a selection on two or more dimensions.

Pivot (also called rotate) is a visualization operation that rotates the data axes in view to provide an alternative data presentation.

Module – 2

3(a). Explain OLAP Data indexing for Bitmap index and Join index.

- In the bitmap index for a given attribute, there is a distinct bit vector, Bv, for each value v in the attribute's domain.
- If the attribute has the value v for a given row in the data table, then the bit representing that value is set to 1 in the corresponding row of the bitmap index. All other bits for that row are set to 0.



Base table

RID	*item*	*city*
R1	H	V
R2	C	V
R3	P	V
R4	S	V
R5	H	T
R6	C	T
R7	P	T
R8	S	T

item bitmap index table

RID	H	C	P	S
R1	1	0	0	0
R2	0	1	0	0
R3	0	0	1	0
R4	0	0	0	1
R5	1	0	0	0
R6	0	1	0	0
R7	0	0	1	0
R8	0	0	0	1

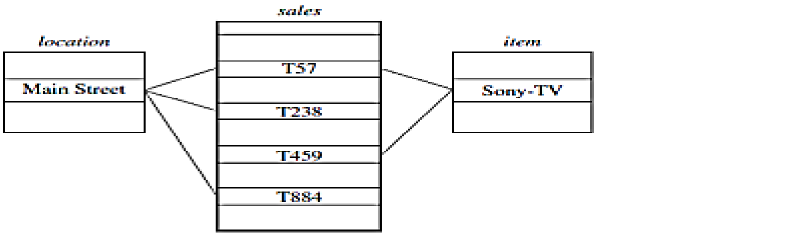
city bitmap index table

RID	V	T
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

Note: H for "home entertainment," C for "computer," P for "phone," S for "security," V for "Vancouver," T for "Toronto."

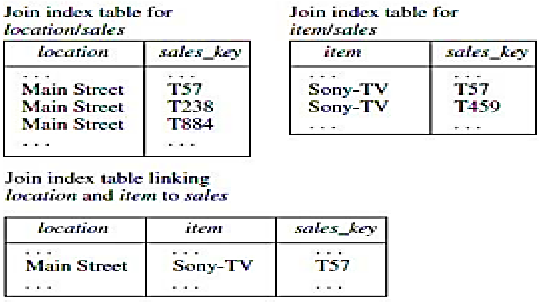
Indexing OLAP data using bitmap indices.

- The jc y processing. Traditional indexing maps the value in a given column to a list of rows having that value.
- In contrast, join indexing registers the joinable rows of two relations from a relational database. For example, if two relations R(RID, A) and S(B, SID) join on the attributes A and B, then the join index record contains the pair (RID, SID), where RID and SID are record identifiers from the R and S relations, respectively.



5 Linkages between a *sales* fact table and *location* and *item* dimension tables.

Linkages between a *sales* fact table and *location* and *item* dimension tables.



Join index table for *location/sales*

location	*sales_key*
...	...
Main Street	T57
Main Street	T238
Main Street	T884
...	...

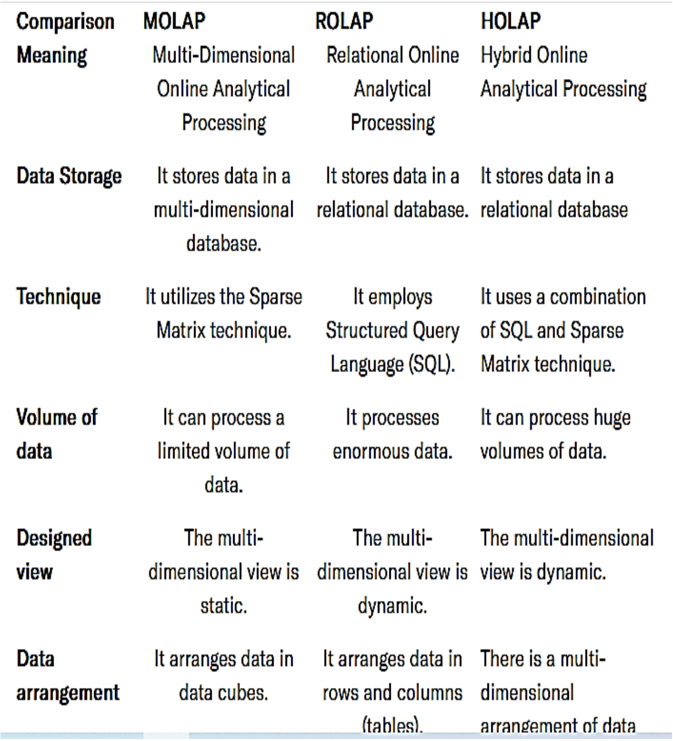
Join index table for *item/sales*

item	*sales_key*
...	...
Sony-TV	T57
Sony-TV	T459
...	...

Join index table linking *location* and *item* to *sales*

location	*item*	*sales_key*
...	...	...
Main Street	Sony-TV	T57
...	...	...

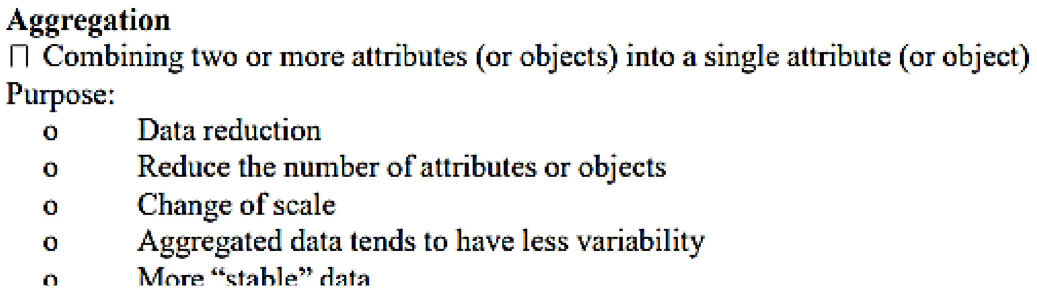
3(b). Differentiate ROLAP, MOLAP and HOLAP servers.



Comparison	MOLAP	ROLAP	HOLAP
Meaning	Multi-Dimensional Online Analytical Processing	Relational Online Analytical Processing	Hybrid Online Analytical Processing
Data Storage	It stores data in a multi-dimensional database.	It stores data in a relational database.	It stores data in a relational database
Technique	It utilizes the Sparse Matrix technique.	It employs Structured Query Language (SQL).	It uses a combination of SQL and Sparse Matrix technique.
Volume of data	It can process a limited volume of data.	It processes enormous data.	It can process huge volumes of data.
Designed view	The multi-dimensional view is static.	The multi-dimensional view is dynamic.	The multi-dimensional view is dynamic.
Data arrangement	It arranges data in data cubes.	It arranges data in rows and columns (tables).	There is a multi-dimensional arrangement of data

4(a). Explain Data – Preprocessing steps and the challenges faced in Data Mining.

Aggregation • Sampling • Dimensionality Reduction • Feature subset selection • Feature creation • Discretization and Binarization • Attribute Transformation

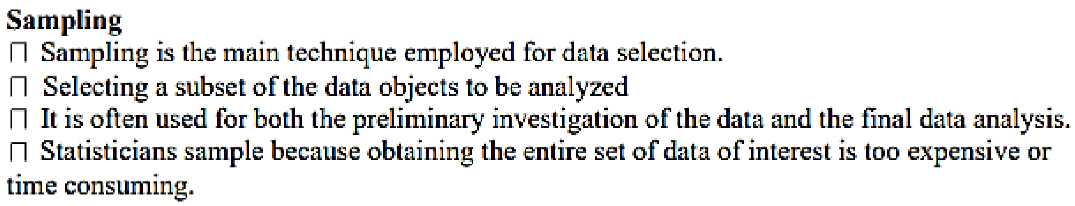


Aggregation

 Combining two or more attributes (or objects) into a single attribute (or object)

Purpose:

- Data reduction
- Reduce the number of attributes or objects
- Change of scale
- Aggregated data tends to have less variability
- More "stable" data



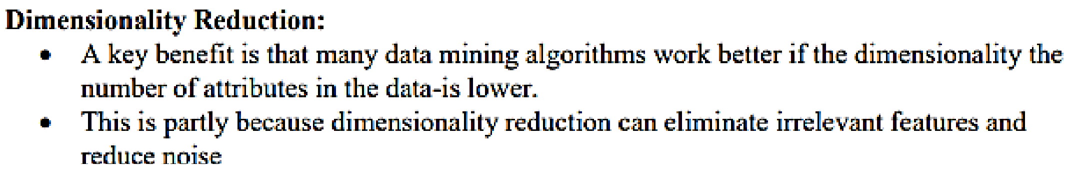
Sampling

 Sampling is the main technique employed for data selection.

 Selecting a subset of the data objects to be analyzed

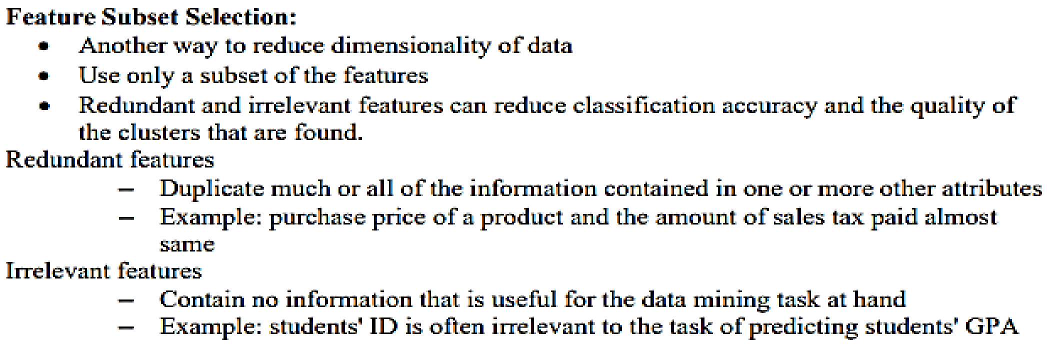
 It is often used for both the preliminary investigation of the data and the final data analysis.

 Statisticians sample because obtaining the entire set of data of interest is too expensive or time consuming.



Dimensionality Reduction:

- A key benefit is that many data mining algorithms work better if the dimensionality the number of attributes in the data-is lower.
- This is partly because dimensionality reduction can eliminate irrelevant features and reduce noise



Feature Subset Selection:

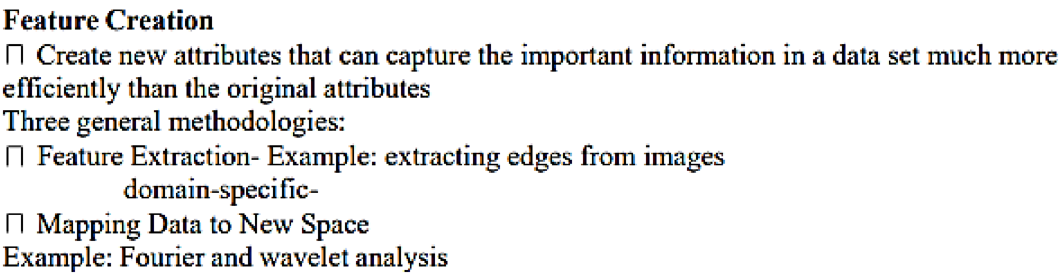
- Another way to reduce dimensionality of data
- Use only a subset of the features
- Redundant and irrelevant features can reduce classification accuracy and the quality of the clusters that are found.

Redundant features

- Duplicate much or all of the information contained in one or more other attributes
- Example: purchase price of a product and the amount of sales tax paid almost same

Irrelevant features

- Contain no information that is useful for the data mining task at hand
- Example: students' ID is often irrelevant to the task of predicting students' GPA



Feature Creation

- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes

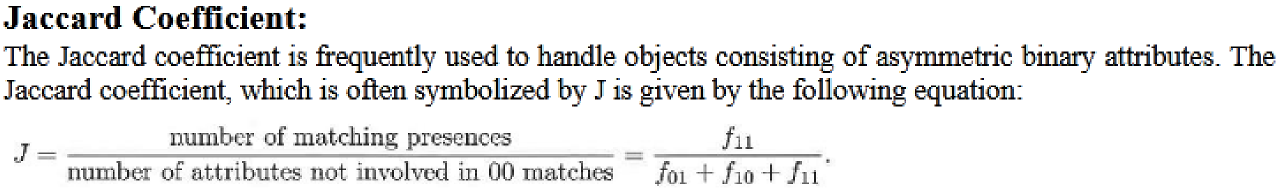
Three general methodologies:

- Feature Extraction- Example: extracting edges from images
  domain-specific-
- Mapping Data to New Space

Example: Fourier and wavelet analysis

4(b). Briefly explain Similarity and Dissimilarity between the objects. Find the SMC and Jacquard coefficient of two binary vectors. X=(1,0,0,0,0,0,0,0,0,0) and Y=(0,0,0,0,0,0,0,0,0,1)

X=(1,0,0,0,0,0,0,0,0,0)
Y=(0,0,0,0,0,0,0,0,0,1)

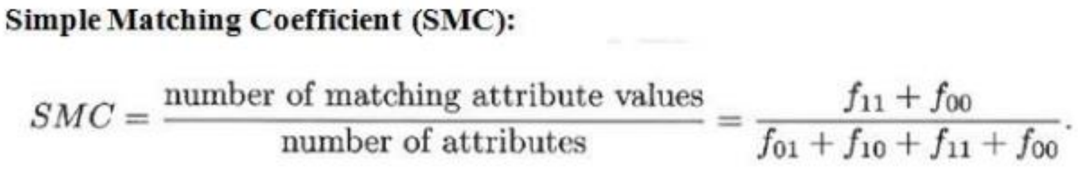


Jaccard Coefficient:

The Jaccard coefficient is frequently used to handle objects consisting of asymmetric binary attributes. The Jaccard coefficient, which is often symbolized by J is given by the following equation:

$$J = \frac{\text{number of matching presences}}{\text{number of attributes not involved in 00 matches}} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}.$$

Jaccard(x, y) = 0 / (1 + 1 + 0) = 0/2 = 0



Simple Matching Coefficient (SMC):

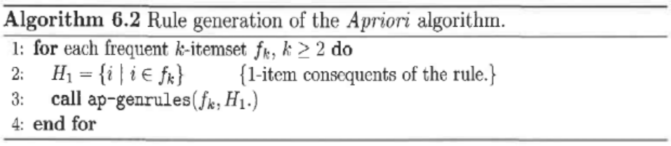
$$SMC = \frac{\text{number of matching attribute values}}{\text{number of attributes}} = \frac{f_{11} + f_{00}}{f_{01} + f_{10} + f_{11} + f_{00}}.$$

SMC = (0+8) / (1+1+0+8) = 8/10 = 0.8

Module – 3

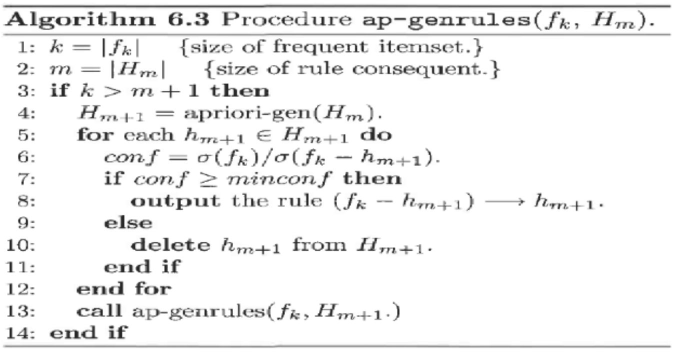
5(a). Explain the rule generation in Apriori algorithm with example.

A pseudocode for the rule generation step is shown in Algorithms 6.2 and 6.3. Note the similarity between the ap-genrules procedure given in Algorithm 6.3 and the frequent itemset generation procedure given in Algorithm 6.1. The only difference is that, in rule generation, we do not have to make additional passes over the data set to compute the confidence of the candidate rules. Instead, we determine the confidence of each rule by using the support counts computed during frequent itemset generation.



Algorithm 6.2 Rule generation of the *Apriori* algorithm.

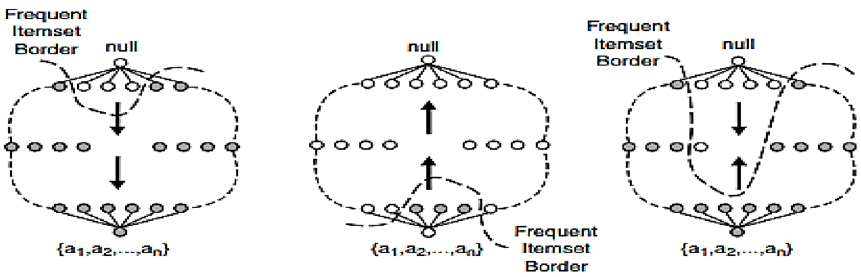
```
for each frequent $k$-itemset $f_k$, $k \geq 2$ do
    $H_1 = \{i \mid i \in f_k\}$        {1-item consequents of the rule.}
    call ap-genrules($f_k$, $H_1$.)
end for
```



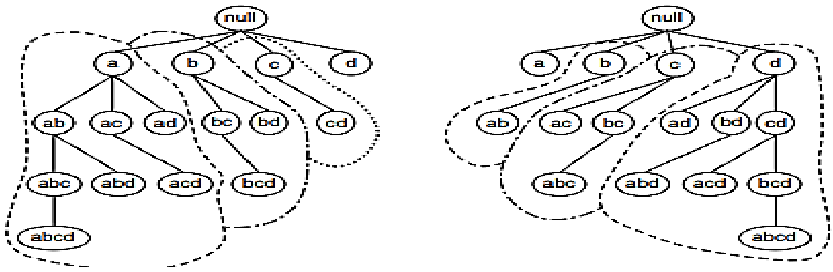
Algorithm 6.3 Procedure ap-genrules(f_k, H_m).

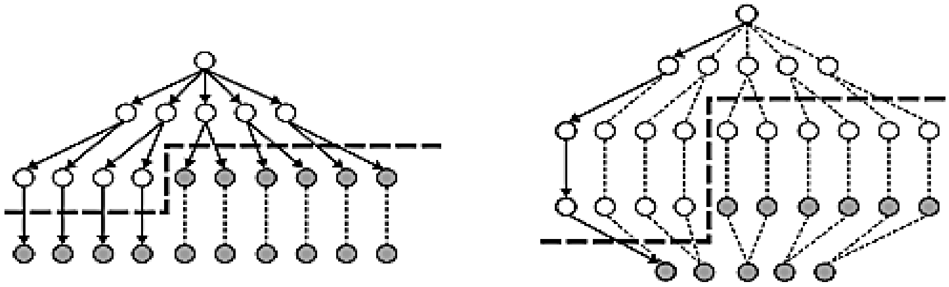
```
$k = |f_k|$      {size of frequent itemset.}
$m = |H_m|$      {size of rule consequent.}
if $k > m + 1$ then
    $H_{m+1}$ = apriori-gen($H_m$).
    for each $h_{m+1} \in H_{m+1}$ do
        $conf = \sigma(f_k)/\sigma(f_k - h_{m+1})$.
        if $conf \geq minconf$ then
            output the rule $(f_k - h_{m+1}) \longrightarrow h_{m+1}$.
        else
            delete $h_{m+1}$ from $H_{m+1}$.
        end if
    end for
    call ap-genrules($f_k$, $H_{m+1}$.)
end if
```

5(b). Explain the Alternative method for generating frequent itemset.



(a) General-to-specific (b) Specific-to-general (c) Bidirectional





(a) Breadth first (b) Depth first

6(a). Briefly explain FP growth algorithm

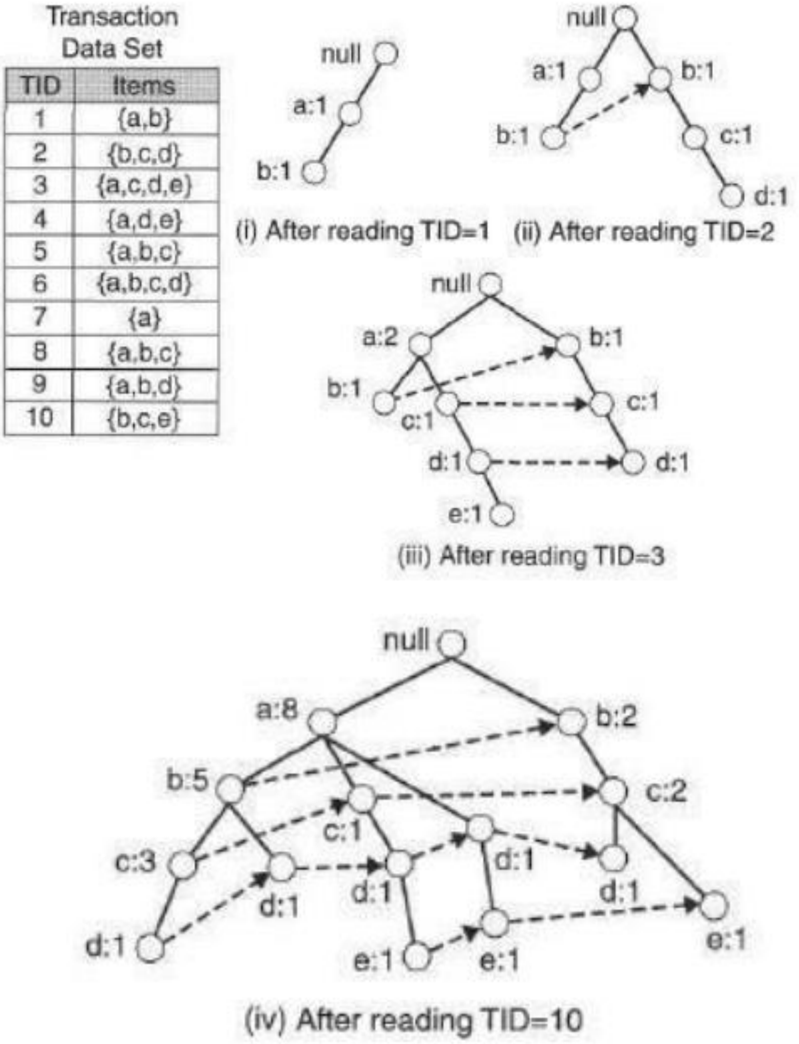
FP-Growth Algorithm

- □ Apriori: uses a generate-and-test approach – generates candidate itemsets and tests if they are frequent
  - – Generation of candidate itemsets is expensive(in both space and time)
  - – Support counting is expensive
    Subset checking (computationally expensive) Multiple Database scans
- □ FP-Growth: allows frequent itemset discovery without candidate itemset generation. Two step approach:
  - – Step 1: Build a compact data structure called the FP-tree
    - • Built using 2 passes over the data-set.
  - – Step 2: Extracts frequent itemsets directly from the FP-tree

Step 1: FP-Tree Construction

□ FP-Tree is constructed using 2 passes over the data-set: Pass 1: Scan data and find support for each item.

- – Discard infrequent items.
- – Sort frequent items in decreasing order based on their support.
  - – Use this order when building the FP-Tree,so common prefixescan be shared.



Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

(i) After reading TID=1 (ii) After reading TID=2

(iii) After reading TID=3

(iv) After reading TID=10

Figure 6.24. Construction of an FP-tree.

6(b). Explain the objective measure of Interestingness for evaluating association patterns.

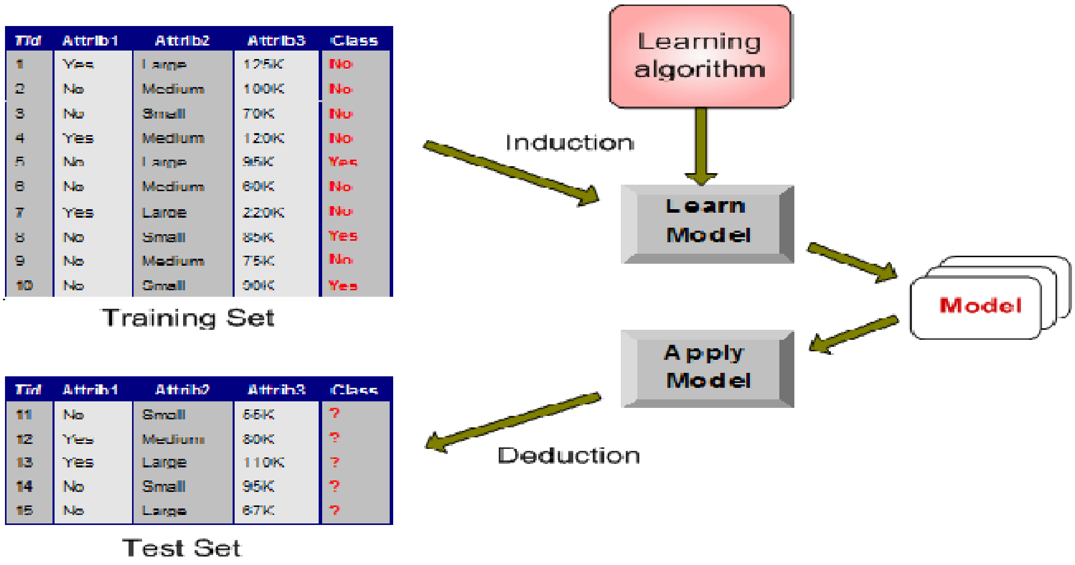
Objective Measures of Interestingness

An objective measure is a data-driven approach for evaluating the quality of association patterns. It is domain-independent and requires minimal input from the users, other than to specify a threshold for filtering low-quality patterns. An objective measure is usually computed based on the frequency counts tabulated in a contingency table. Table 6.7 shows an example of a contingency table for a pair of binary variables, A and B. We use the notation A (B) to indicate that A (B) is absent from a transaction. Each entry fij in this 2 × 2 table denotes a frequency count. For example, f11 is the number of times A and B appear together in the same transaction, while f01 is the number of transactions that contain B but not A. The row sum f1+ represents the support count for A, while the column sum f+1 represents the support count for B. Finally, even though our discussion focuses mainly on asymmetric binary variables, note that contingency tables are also applicable to other attribute types such as symmetric binary, nominal, and ordinal variables.

Module – 4

7(a). With a neat block diagram, explain general approach to solve classification problems with application.

- 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.
- Therefore, a key objective of the learning algorithm is to build models with good generalization capability; i.e., models that accurately predict the class labels of previously unknown records



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

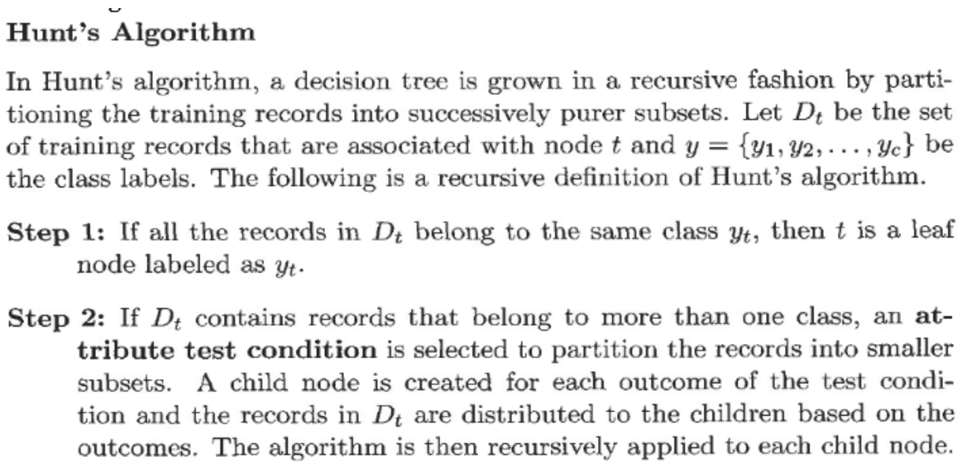
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

7(b). Explain with example, how to build decision tree using Hunt's algorithm.

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.
- Nevertheless, efficient algorithms have been developed to induce a reasonably accurate, albeit suboptimal, decision tree in a reasonable amount of time.
- These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data.
- One such algorithm is Hunt's algorithm, which is the basis of many existing decision tree induction algorithms, including ID3, C4.5, and CART. This section presents a high-level discussion of Hunt's algorithm and illustrates some of its design issues.



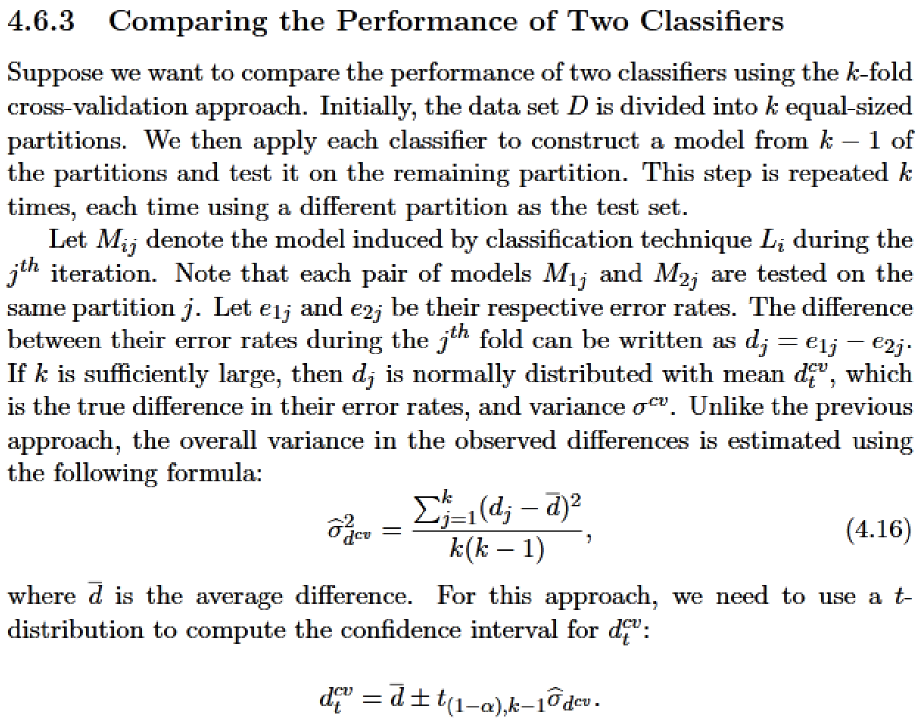
Hunt's Algorithm

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

8(a). Explain different methods for comparing classifier.



4.6.3 Comparing the Performance of Two Classifiers

Suppose we want to compare the performance of two classifiers using the k-fold cross-validation approach. Initially, the data set D is divided into k equal-sized partitions. We then apply each classifier to construct a model from $k-1$ of the partitions and test it on the remaining partition. This step is repeated k times, each time using a different partition as the test set.

Let M_{ij} denote the model induced by classification technique L_i during the j^{th} iteration. Note that each pair of models M_{1j} and M_{2j} are tested on the same partition j. Let e_{1j} and e_{2j} be their respective error rates. The difference between their error rates during the j^{th} fold can be written as $d_j = e_{1j} - e_{2j}$. If k is sufficiently large, then d_j is normally distributed with mean d_t^{cv}, which is the true difference in their error rates, and variance σ^{cv}. Unlike the previous approach, the overall variance in the observed differences is estimated using the following formula:

$$\widehat{\sigma}^2_{d^{cv}} = \frac{\sum_{j=1}^{k}(d_j - \overline{d})^2}{k(k-1)}, \tag{4.16}$$

where $\overline{d}$ is the average difference. For this approach, we need to use a t-distribution to compute the confidence interval for d_t^{cv}:

$$d_t^{cv} = \overline{d} \pm t_{(1-\alpha),k-1}\widehat{\sigma}_{d^{cv}}.$$

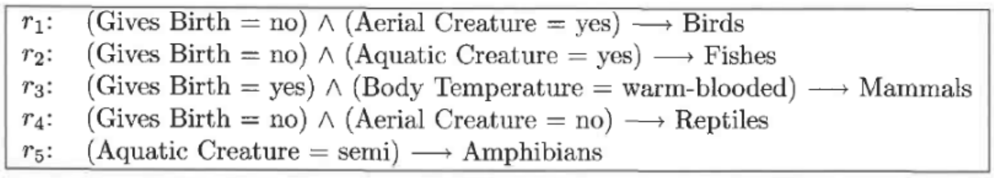
8(b). Explain the rule based classifier with example.

Rule Based Classifiers

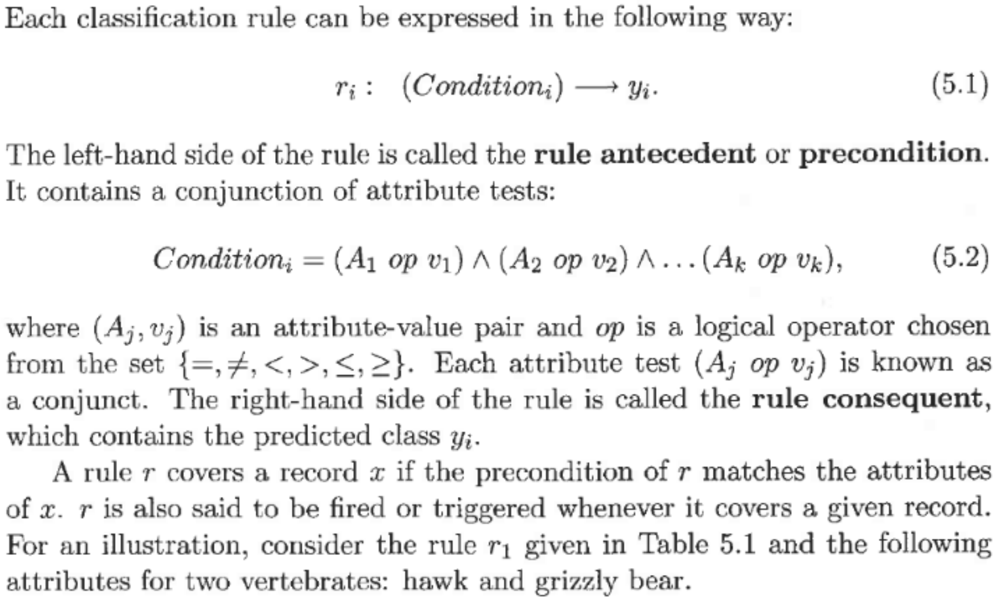
A rule-based classifier is a technique for classifying records using a collection of "if . . .then. . ." rules.

Table 5.1 shows an example of a model generated by a rule-based classifier for the vertebrate classification problem. The rules for the model are represented in a disjunctive normal form, R = (r_1 V r_2V...r_n), where R is known as the rule set and r_i's are the classification rules or disjuncts.

Table 5.1. Example of a rule set for the vertebrate classification problem.



r_1:	(Gives Birth = no) ∧ (Aerial Creature = yes) ⟶ Birds
r_2:	(Gives Birth = no) ∧ (Aquatic Creature = yes) ⟶ Fishes
r_3:	(Gives Birth = yes) ∧ (Body Temperature = warm-blooded) ⟶ Mammals
r_4:	(Gives Birth = no) ∧ (Aerial Creature = no) ⟶ Reptiles
r_5:	(Aquatic Creature = semi) ⟶ Amphibians



Each classification rule can be expressed in the following way:

$$r_i: \quad (Condition_i) \longrightarrow y_i. \tag{5.1}$$

The left-hand side of the rule is called the **rule antecedent** or **precondition**. It contains a conjunction of attribute tests:

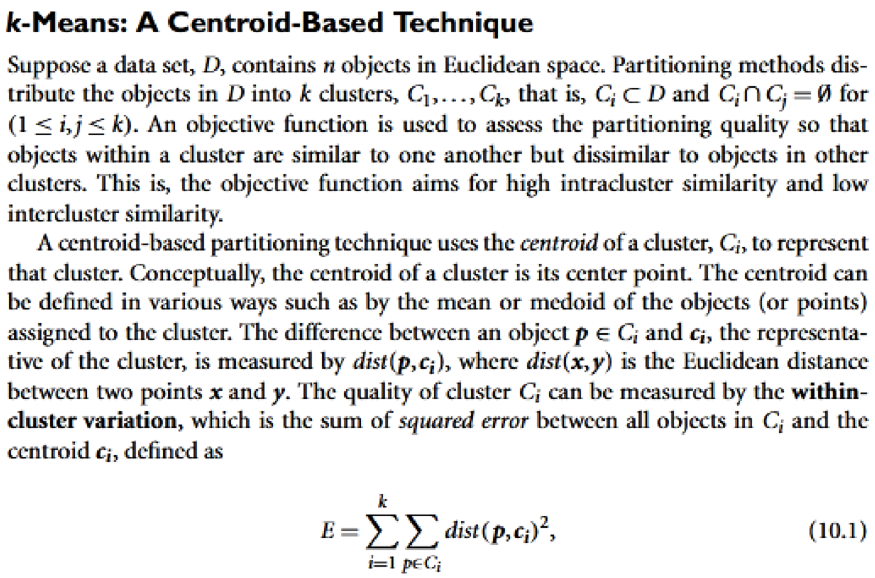
$$Condition_i = (A_1 \; op \; v_1) \wedge (A_2 \; op \; v_2) \wedge \ldots (A_k \; op \; v_k), \tag{5.2}$$

where (A_j, v_j) is an attribute-value pair and *op* is a logical operator chosen from the set $\{=, \neq, <, >, \leq, \geq\}$. Each attribute test $(A_j \; op \; v_j)$ is known as a conjunct. The right-hand side of the rule is called the **rule consequent**, which contains the predicted class y_i.

A rule r covers a record x if the precondition of r matches the attributes of x. r is also said to be fired or triggered whenever it covers a given record. For an illustration, consider the rule r_1 given in Table 5.1 and the following attributes for two vertebrates: hawk and grizzly bear.

Module – 5

9(a). Describe K – means clustering algorithm. What are its limitations?

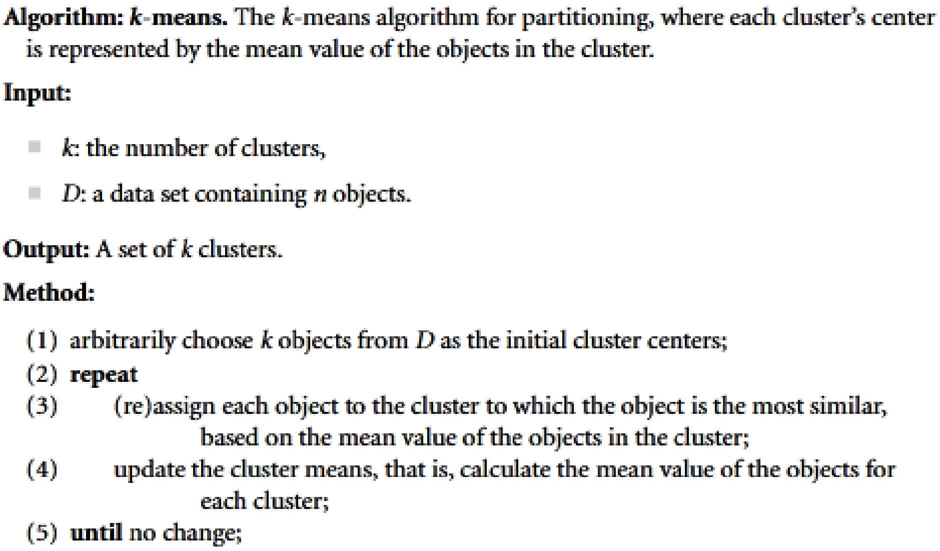


k-Means: A Centroid-Based Technique

Suppose a data set, D, contains n objects in Euclidean space. Partitioning methods distribute the objects in D into k clusters, $C_1, \ldots, C_k$, that is, $C_i \subset D$ and $C_i \cap C_j = \emptyset$ for $(1 \leq i, j \leq k)$. 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. This is, the objective function aims for high intracluster similarity and low intercluster similarity.

A centroid-based partitioning technique uses the *centroid* of a cluster, C_i, 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 $\boldsymbol{p} \in C_i$ and $\boldsymbol{c_i}$, the representative of the cluster, is measured by $dist(\boldsymbol{p}, \boldsymbol{c_i})$, where $dist(\boldsymbol{x}, \boldsymbol{y})$ is the Euclidean distance between two points $\boldsymbol{x}$ and $\boldsymbol{y}$. The quality of cluster C_i can be measured by the **within-cluster variation**, which is the sum of *squared error* between all objects in C_i and the centroid $\boldsymbol{c_i}$, defined as

$$E = \sum_{i=1}^{k} \sum_{\boldsymbol{p} \in C_i} dist(\boldsymbol{p}, \boldsymbol{c_i})^2, \tag{10.1}$$



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

Input:

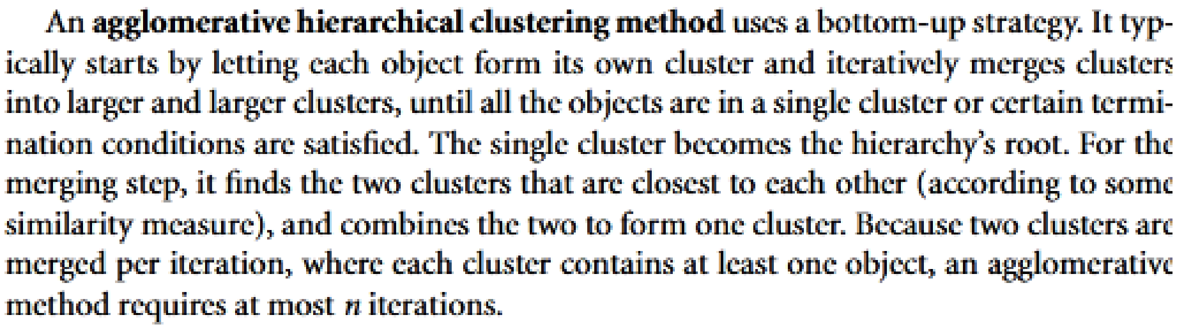
- k: the number of clusters,
- D: a data set containing n objects.

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;
```

9(b). With example, explain Agglomerative Hierarchical clustering with example.

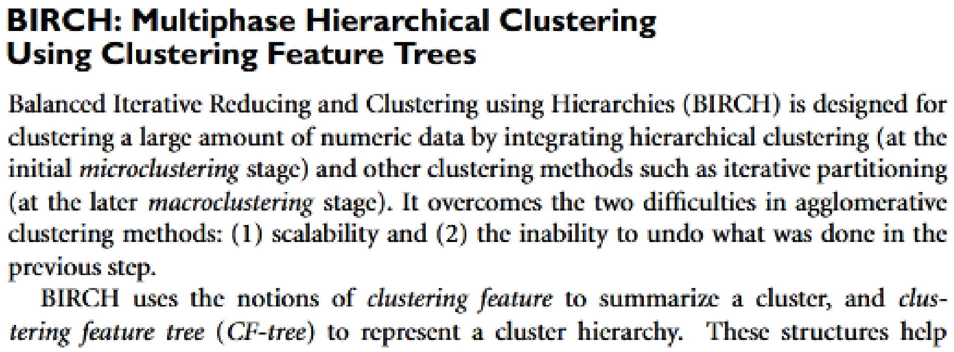


An **agglomerative hierarchical clustering method** uses a bottom-up strategy. It typically starts by letting each object form its own cluster and iteratively merges clusters into larger and larger clusters, until all the objects are in a single cluster or certain termination conditions are satisfied. 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.

10(a). With Time and Space complexity, explain DBSCAN clustering algorithm.

DBSCAN: 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 > 0 is used to specify the radius of a neighborhood we consider for every object. The -neighborhood of an object o is the space within a radius centered at o.
Due to the fixed neighborhood size parameterized by , 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

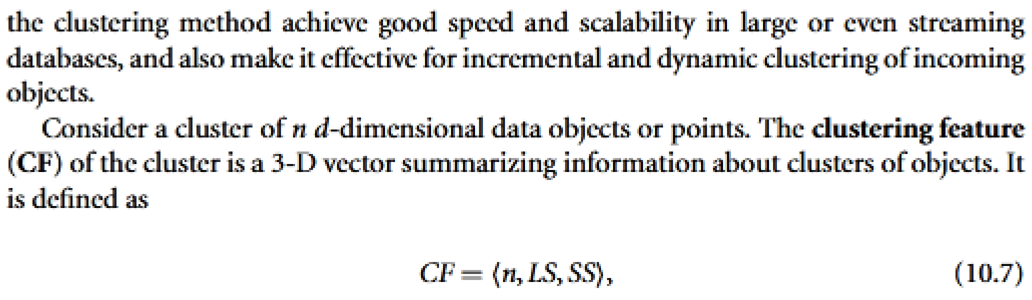
10(b). Explain the BIRCH scalable algorithm.



BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is designed for clustering a large amount of numeric data by integrating hierarchical clustering (at the initial *microclustering* stage) and other clustering methods such as iterative partitioning (at the later *macroclustering* stage). It overcomes the two difficulties in agglomerative clustering methods: (1) scalability and (2) the inability to undo what was done in the previous step.

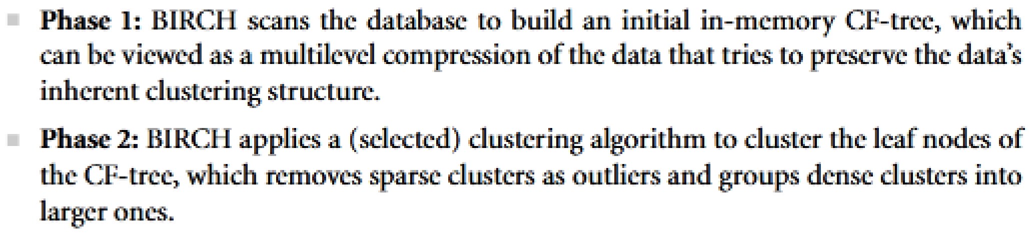
BIRCH uses the notions of *clustering feature* to summarize a cluster, and *clustering feature tree* (*CF-tree*) to represent a cluster hierarchy. These structures help



the clustering method achieve good speed and scalability in large or even streaming databases, and also make it effective for incremental and dynamic clustering of incoming objects.

Consider a cluster of *n* *d*-dimensional data objects or points. The **clustering feature** (**CF**) of the cluster is a 3-D vector summarizing information about clusters of objects. It is defined as

$$CF = \langle n, LS, SS \rangle, \qquad (10.7)$$



- **Phase 1:** BIRCH scans the database to build an initial in-memory CF-tree, which can be viewed as a multilevel compression of the data that tries to preserve the data's inherent clustering structure.
- **Phase 2:** BIRCH applies a (selected) clustering algorithm to cluster the leaf nodes of the CF-tree, which removes sparse clusters as outliers and groups dense clusters into larger ones.