MODULE- II

Data Preprocessing: Data Preprocessing Concepts, Data Cleaning, Data integration and transformation, Data Reduction, Discretization and concept hierarchy.

Data preprocessing

Today's real-world databases are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size (often several gigabytes or more) and their likely origin from multiple, heterogeneous sources. **Low-quality data will lead to low-quality mining results**. "How can the data be preprocessed in order to help improve the quality of the data and, consequently, of the mining results? How can the data be preprocessed so as to improve the efficiency and ease of the mining process?"

There are several data preprocessing techniques.

Data cleaning can be applied to remove noise and correct inconsistencies in data**.**

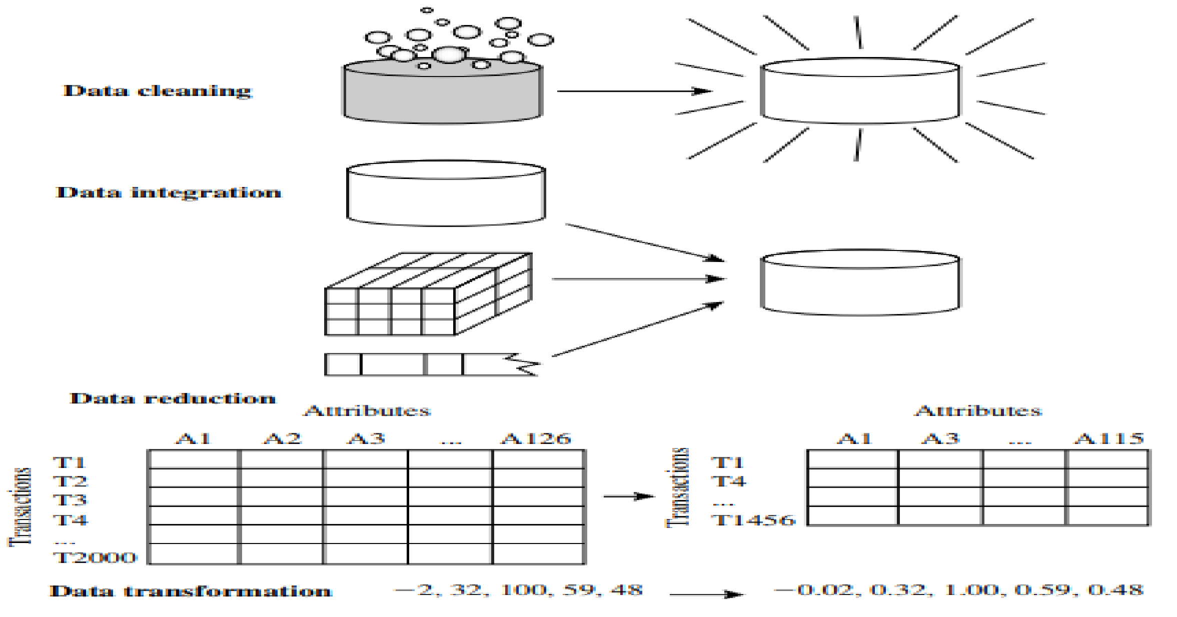
Data integration merges data from multiple sources into a coherent data store such as a data warehouse.

Data reduction obtains a reduced representation of the data set that is smaller in volume yet produces the same or (almost same) analytical results.

Data transformations techniques transforms or consolidated the data into forms appropriate for mining

These techniques are not mutually exclusive; they may work together. For example, data cleaning can involve transformations to correct wrong data, such as by transforming all entries for a date field to a common format.

Data processing techniques, when applied before mining, can substantially **improve the overall quality of the patterns mined and/or the time required for the actual mining**.



Forms of data preprocessing.

Data Cleaning

Real-world data tend to be incomplete, noisy, and inconsistent. **Data cleaning (or data cleansing) routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.** In this section, you will study basic methods for data cleaning.

❖ **Missing Values**

Imagine that you need to analyze AllElectronics sales and customer data. You note that many tuples have no recorded value for several attributes such as customer income. How can you go about filling in the missing values for this attribute? Let's look at the following methods.

1. **Ignore the tuple:** This is usually done when the class label is missing (assuming the mining task involves classification). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.

2. **Fill in the missing value manually:** In general, this approach is time consuming and may not be feasible given a large data set with many missing values.

3. **Use a global constant to fill in the missing value:** Replace all missing attribute values by the same constant such as a label like "Unknown" or $-\infty$. If missing values are replaced by, say, "Unknown," then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common—that of "Unknown." Hence, although this method is simple, it is not recommended.

4. **Use the attribute mean to fill in the missing value**: For example, suppose that the data distribution regarding the income of AllElectronics customers is symmetric and that the mean income is $56,000. Use this value to replace the missing value for income.

5. **Use the attribute mean or median for all samples belonging to the same class as the given tuple:** For example, if classifying customers according to credit risk, we may replace the missing value with the mean income value for customers in the same credit risk category as that of the given tuple.

6. **Use the most probable value to fill in the missing value:** This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for income.

Methods 3 through 6 bias the data—the filled-in value may not be correct. Method 6, however, is a popular strategy. In comparison to the other methods, it uses the most information from the present data to predict missing values. By considering the other attributes' values in its estimation of the missing value for income, there is a greater chance that the relationships between income and the other attributes are preserved.

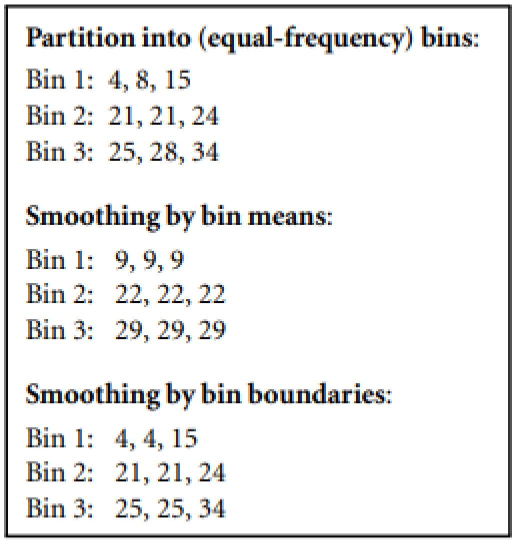
❖ **Noisy Data**

What is noise?" Noise is a random error or variance in a measured variable. . Given a numeric attribute such as, say, price, how can we "smooth" out the data to remove the noise? Following are the data smoothing techniques.

1. **Binning:** Binning methods smooth a sorted data value by consulting its "neighborhood," that is, the values around it. The sorted values are distributed into a number of "buckets," or bins. Because binning methods consult the neighborhood of values, they perform local smoothing. The following illustrates some binning techniques. In this example, the data for price are first sorted and then partitioned into equal-frequency bins of size 3 (i.e., each bin contains three values). In **smoothing by bin means**, each value in a bin is replaced by the mean value of the bin. For example, the mean of the values 4, 8, and 15 in Bin 1 is 9. Therefore, each original value in this bin is replaced by the value 9.

   Similarly, **smoothing by bin medians** can be employed, in which each bin value is replaced by the bin median.

Sorted data for *price* (in dollars): 4, 8, 15, 21, 21, 24, 25, 28, 34



Partition into (equal-frequency) bins:
Bin 1: 4, 8, 15
Bin 2: 21, 21, 24
Bin 3: 25, 28, 34

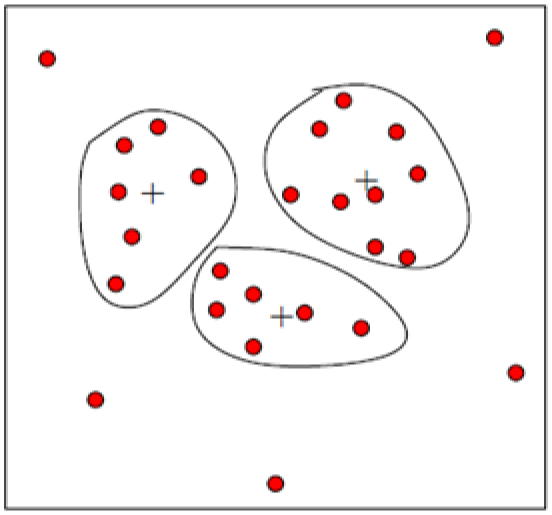
Smoothing by bin means:
Bin 1: 9, 9, 9
Bin 2: 22, 22, 22
Bin 3: 29, 29, 29

Smoothing by bin boundaries:
Bin 1: 4, 4, 15
Bin 2: 21, 21, 24
Bin 3: 25, 25, 34

Binning methods for data smoothing.

In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value. In general, larger the width, greater the effect of the smoothing. Alternatively, bins may be equal width, where the interval range of values in each bin is constant.

2. **Clustering:** Outliers may be detected by clustering, for example, where similar values are organized into groups, or "clusters." Intuitively, values that fall outside of the set of clusters may be considered outliers. [Refer Figure]



: Outliers may be detected by clustering analysis.

3.Regression: Data smoothing can also be done by regression, a technique that conforms data values to a function. **Linear regression** involves finding the "best" line to fit two attributes (or variables) so that one attribute can be used to predict the other**. Multiple linear regression** is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.

Many data smoothing methods are also used for data discretization (a form of data transformation) and data reduction. For example, the binning techniques described before reduce the number of distinct values per attribute. This acts as a form of data reduction for logic-based data mining methods, such as decision tree induction, which repeatedly makes value comparisons on sorted data. Concept hierarchies are a form of data discretization that can also be used for data smoothing. A concept hierarchy for *price*, for example, may map *real* price values into *inexpensive, moderately priced*, and *expensive*, thereby reducing the number of data values to be handled by the mining process.

Data integration

Data mining often requires **data integration**—the merging of data from multiple data stores. Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve **the accuracy and speed** of the subsequent data mining process.

Issues in data integration

1. **Entity identification problem**

Schema integration and object matching can be tricky. How can equivalent real-world entities from multiple data sources be matched up? This is referred to **as the entity identification problem.** For example, how can the data analyst or the computer be sure that *customer_id* in one database and *cust_number* in another refer to the same attribute? Examples of metadata for each attribute include the name, meaning, data type, and range of values permitted for the attribute, and null rules for handling blank, zero, or null values. Such **metadata** can be used to help **avoid errors in schema integration**. **The metadata may also be used to help transform the data**.

2. Redundancy

Redundancy is another important issue in data integration. An attribute (such as annual revenue, for instance) may be redundant if it can be "derived" from another attribute or set of attributes. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set. Some redundancies can be detected by **correlation analysis**. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data.

For numeric attributes, we can evaluate the correlation between two attributes, A and B, by computing the **correlation coefficient** (also known as Pearson's product moment coefficient, named after its inventor, Karl Pearson).

$$r_{A,B} = \frac{cov(A,B)}{\sigma_A \sigma_B}$$

$$cov(A,B) = \sum_{i=1}^{N} \frac{(A_i - \bar{A})(B_i - \bar{B})}{N-1}$$

where N is the number of tuples, A_i and B_i are the respective values of A and B in tuple i, $\bar{A}$ and $\bar{B}$ are the respective mean values of A and B, σ_A and σ_B are the respective standard deviations of A and B. Note that $-1 \leq r_{A,B} \leq +1$. If $r_{A,B}$ is greater than 0, then A and B are positively correlated, meaning that the values of A increase as the values of B increase. The higher the value, the stronger the correlation (i.e., the more each attribute implies the other). Hence, a higher value may indicate that A (or B) may be removed as a redundancy. If the resulting value is equal to 0, then A and B are independent and there is no correlation between them. If the resulting value is less than 0, then A and B are negatively correlated, where the values of one attribute increase as the values of the other attribute decrease. This means that each attribute discourages the other.

For categorical (discrete) data, a correlation relationship between two attributes, A and B, can be discovered by a $\chi 2$ (chi-square) test. Suppose A has c distinct values, namely a1 , a2 , . . . ac . B has r distinct values, namely b1, b2 , . . . br . The data tuples described by A and B can be shown as a contingency table, with the c values of A making up the columns and the r values of B making up the rows. Let (Ai , B j) denote the event that attribute A takes on value ai and attribute B takes on value b j , that is, where ($A = a_i$,$B = b_j$). Each **and every possible (A_i, B_j) joint event has its own cell (or slot) in the table. The $\chi 2$ value** (also known as the Pearson $\chi 2$ statistic) is computed as:

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

where oij is the observed frequency (i.e., actual count) of the joint event and eij is the expected frequency which can be computed as,

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{n},$$

where n is the number of data tuples.

Please note: Problems discussed in the class

The χ2 statistic tests the hypothesis that A and B are independent, that is, there is no correlation between them. The test is based on a significance level, with .(r − 1) × (c − 1) degrees of freedom. We illustrate the use of this statistic in Example 3.1. If the hypothesis can be rejected, then we say that A and B are statistically correlated.

3. Tuple duplication

In addition to detecting redundancies between attributes, duplication should also be detected at the tuple level (e.g., where there are two or more identical tuples for a given unique data entry case). The use of denormalized tables (often done to improve performance by avoiding joins) is another source of data redundancy. Inconsistencies often arise between various duplicates, due to inaccurate data entry or updating some but not all of the occurrences of the data. For example, if a purchase order database contains attributes for the purchaser's name and address instead of a key to this information in a purchaser database, discrepancies can occur, such as the same purchaser's name appearing with different addresses within the purchase order database.

4. Data value conflict detection and resolution

A third important issue in data integration is the *detection and resolution of data value conflicts*. For example, for the same real-world entity, attribute values from different sources may differ. This may be due to differences in representation, scaling, or encoding. For instance, a *weight* attribute may be stored in metric units in one system and British imperial units in another. For a hotel chain, the *price* of rooms in different cities may involve not only different currencies but also different services (such as free breakfast) and taxes. An attribute in one system may be recorded at a lower level of abstraction than the "same" attribute in another. For example, the *total sales* in one database may refer to one branch of *All Electronics*, while an attribute of the same name in another database may refer to the total sales for *All Electronics* stores in a given region.

Data transformation

In *data transformation*, the data are transformed or consolidated into forms appropriate for mining. Data transformation can involve the following:

- Smoothing, which works to remove noise from the data. Such techniques include binning, regression, and clustering.
- Aggregation, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total

amounts. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.

- Generalization of the data, where low-level or "primitive" (raw) data are replaced by higher-level concepts through the use of concept hierarchies. For example, categorical attributes, like *street*, can be generalized to higher-level concepts, like *city* or *country*. Similarly, values for numerical attributes, like *age*, may be mapped to higher-level concepts, like *youth, middle-aged*, and *senior*.
- Normalization, where the attribute data are scaled so as to fall within a small specified range, such as -1:0 to 1:0, or 0:0 to 1:0.
- Attribute construction (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process.

Data normalization

To help avoid dependence on the choice of measurement units, the data should be normalized or standardized. This involves transforming the data to fall within a smaller or common range such as [−1, 1] or [0.0, 1.0].

Normalizing the data attempts to give all attributes an equal weight. Normalization is particularly useful for classification algorithms involving neural networks or distance measurements such as nearest-neighbor classification and clustering.

Mainly three types of normalization

Min-max normalization

Min-max normalization performs a linear transformation on the original data. Suppose that *min*$_A$ and *max*$_A$ are the minimum and maximum values of an attribute, *A*. Min-max normalization maps a value, *v*, of *A* to *v*0 in the range [*new_min*$_A$;*new_max*$_A$] by computing

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A.$$

Min-max normalization preserves the relationships among the original data values. It will encounter an "out-of-bounds" error if a future input case for normalization falls outside of the original data range for *A*.

Example

Min-max normalization. Suppose that the minimum and maximum values for the attribute *income* are \$12,000 and \$98,000, respectively. We would like to map *income* to the range [0.0, 1.0]. By min-max normalization, a value of \$73,600 for *income* is transformed to $\frac{73,600-12,000}{98,000-12,000}(1.0-0)+0=0.716$. ■

Z-score normalization (zero-mean normalization)

The values for an attribute, *A*, are normalized based on the mean and standard deviation of *A*. A value, *v*, of *A* is normalized to *v'* by computing

$$v' = \frac{v - \bar{A}}{\sigma_A},$$

where $\bar{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A. This method of normalization is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliers that dominate the min-max normalization.

z-score normalization Suppose that the mean and standard deviation of the values for the attribute *income* are \$54,000 and \$16,000, respectively. With z-score normalization, a value of \$73,600 for *income* is transformed to $\frac{73{,}600-54{,}000}{16{,}000} = 1.225$. ■

Normalization by decimal scaling

Normalization by **decimal scaling** normalizes by moving the decimal point of values of attribute A. The number of decimal points moved depends on the maximum absolute value of A. A value, v , of A is normalized to v' by computing

$$v'_i = \frac{v_i}{10^j},$$

where j is the smallest integer such that $max(|v'_i|) < 1$.

Suppose that the recorded values of A range from −986 to 917. The maximum absolute value of A is 986. To normalize by decimal scaling, we therefore divide each value by 1000 (i.e., j = 3) so that −986 normalizes to −0.986 and 917 normalizes to 0.917.

Attribute construction

In **attribute construction**, new attributes are constructed from the given attributes and added in order to help to improve accuracy and understanding of structure in high dimensional data. For example, we may wish to add the attribute area based on the attributes height and width. Attribute construction can help alleviate the fragmentation problem when decision tree algorithms are used for classification, where an attribute is repeatedly tested along a path in derived decision tree. By combining attributes, attribute construction can discover missing information about the relationships between data attributes that can be useful for knowledge discovery.

Data reduction

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results.

Data reduction strategies include the following

1.Data Cube Aggregation, where **aggregation operations are applied to the construction of a data cube.**

2. **Dimensionality reduction-** Reducing the number of random variables or attributes under consideration.

Various methods for dimensionality reduction are

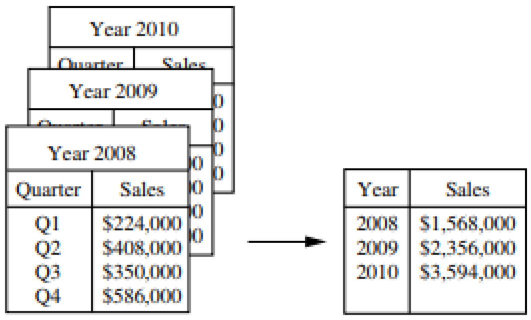
1. Wavelet transforms
2. Principal Component analysis
3. Attribute subset selection

3. **Numerosity reduction:** Replace the original data volume by alternative smaller forms of data representations.
Two types- Parametric and non-parametric methods.

4. Generalization- Discretization and concept hierarchy generation
Where raw data values for attributes are replaced by ranges or higher conceptual levels. Data discretization is a form of numerosity reduction that is very useful for the automatic generation of concept hierarchies. Discretization and concept hierarchy generation are powerful tools for data mining, in that they allow the mining of data at multiple levels of abstraction.

5. Data compression
Obtain a reduced or compressed representation of the original data. Compression techniques can be lossy or lossless.

Data cube aggregation
Imagine that you have collected the data for your analysis. These data consist of the AllElectronics sales per quarter, for the years 2008 to 2010. You are, however, interested in the annual sales (total per year), rather than the total per quarter. Thus, the data can be aggregated so that the resulting data summarize the total sales per year instead of per quarter. This aggregation is illustrated in the following figure. The resulting data set is smaller in volume, without loss of information necessary for the analysis task.



Year 2010

Year 2009

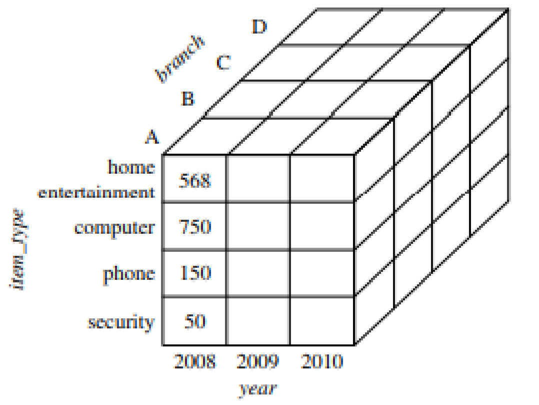
Year 2008

Quarter	Sales
Q1	$224,000
Q2	$408,000
Q3	$350,000
Q4	$586,000

Year	Sales
2008	$1,568,000
2009	$2,356,000
2010	$3,594,000

Sales data for a given branch of *AllElectronics* for the years 2008 through 2010. On the *left*, the sales are shown per quarter. On the *right*, the data are aggregated to provide the annual sales.

Data cubes store multidimensional aggregated information. For example, Figure shows a data cube for multidimensional analysis of sales data with respect to annual sales per item type for each AllElectronics branch.



A data cube for sales at *AllElectronics.*

Attribute subset selection

Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant. For example, if the task is to classify customers based on whether or not they are likely to purchase a popular new CD at AllElectronics when notified of a sale, attributes such as the customer's telephone number are likely to be irrelevant, unlike attributes such as age or music taste. Although it may be possible for a domain expert to pick out some of the useful attributes, this can be a difficult and time consuming task, especially when the data's behavior is not well known. Leaving out relevant attributes or keeping irrelevant attributes may be detrimental, causing confusion for the mining algorithm employed. This can result in discovered patterns of poor quality. In addition, the added volume of irrelevant or redundant attributes can slow down the mining process.

Attribute subset selection reduces the data set size by removing such attributes or dimensions from it. The methods of attribute subset selection are applied. The goal of attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. Mining on a reduced set of attributes has an additional benefit: It reduces the number of attributes appearing in the discovered patterns, helping to make the patterns easier to understand.

"How can we find a 'good' subset of the original attributes?" For n attributes, there are 2^n possible subsets. An exhaustive search for the optimal subset of attributes can be prohibitively expensive, especially as n and the number of data classes increase. Therefore, heuristic methods that explore a reduced search space are commonly used for attribute subset selection. These methods are typically greedy in that, while searching through attribute space, they always make what looks to be the best choice at the time. Their strategy is to make a locally optimal choice in the hope that this will lead to a globally optimal solution. Such greedy methods are effective in practice and may come close to estimating an optimal solution.

The “best” (and “worst”) attributes are typically determined using tests of statistical significance, which assume that the attributes are independent of one another. Many other attribute evaluation measures can be used such as the information gain measure used in building decision trees for classification.

Basic heuristic methods of attribute subset selection include the techniques that follow, some of which are illustrated in following figure.

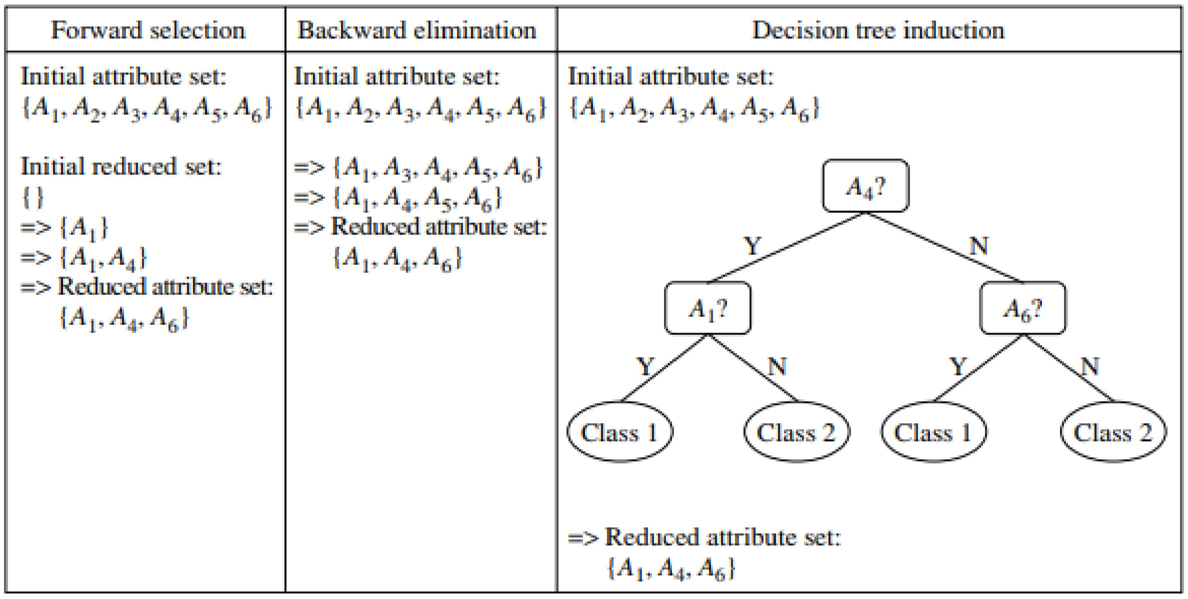
1. **Stepwise forward selection**: The procedure starts with an empty set of attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.

2. **Stepwise backward elimination:** The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.

3. **Combination of forward selection and backward elimination:** The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.

4. **Decision tree induction: Decision tree algorithms** (e.g., ID3, C4.5, and CART) were originally intended for classification. Decision tree induction constructs a flowchart like structure where each internal (non leaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the “best” attribute to partition the data into individual classes. When decision tree induction is used for attribute subset selection, a tree is constructed from the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree form the reduced subset of attributes.

The stopping criteria for the methods may vary. The procedure may employ a threshold on the measure used to determine when to stop the attribute selection process.



Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Initial reduced set: $\{\}$ => $\{A_1\}$ => $\{A_1, A_4\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$

Greedy (heuristic) methods for attribute subset selection.

Numerosity reduction

Numerosity reduction techniques replace the original data volume by alternative, smaller forms of data representation. These techniques may be parametric or nonparametric.

Parametric methods, a model is used to estimate the data, so that typically only the data parameters need to be stored, instead of the actual data. (Outliers may also be stored.) **Regression and log-linear** models are examples.

Regression and log-linear models can be used to approximate the given data. In linear regression, the data are modeled to fit a straight line. For example, a random variable, Y (called a response variable), can be modeled as a linear function of another random variable, x (called a predictor variable), with the equation

$y = wx + b$,

Where the variance of y is assumed to be constant. In the context of data mining, x and y are numeric database attributes. The coefficients, w and b (called regression coefficients), specify the slope of the line and the y-intercept, respectively. These coefficients can be solved for by the method of least squares, which minimizes the error between the actual line separating the data and the estimate of the line. Multiple linear regression is an extension of (simple) linear regression, which allows a response variable, y, to be modeled as a linear function of two or more predictor variables.

Nonparametric methods -for storing reduced representations of the data include **histograms, clustering, and sampling.**

1. **Histograms**

Histograms use binning to approximate data distributions and are a popular form of data reduction. A histogram for an attribute, A, partitions the data distribution of A into disjoint subsets, referred to as buckets or bins. If each bucket represents only a single attribute– value/frequency pair, the buckets are called singleton buckets. Often, buckets instead represent continuous ranges for the given attribute.

Example Histograms. The following data are a list of AllElectronics prices for commonly sold items (rounded to the nearest dollar). The numbers have been sorted: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30.

Figure 3.7 shows a histogram for the data using singleton buckets. To further reduce the data, it is common to have each bucket denote a continuous value range for the given attribute. In Figure 3.8, each bucket represents a different $10 range for price.

"How are the buckets determined and the attribute values partitioned?" There are several partitioning rules, including the following:

Equal-width: In an equal-width histogram, the width of each bucket range is uniform (e.g., the width of $10 for the buckets in Figure 3.8).

Equal-frequency (or **equal-depth**): In an equal-frequency histogram, the buckets are created so that, roughly, the frequency of each bucket is constant (i.e., each bucket contains roughly the same number of contiguous data samples).

V-optimal: If we consider all of the possible histograms for a given number of buckets, the V-optimal histogram is the one with least variance. Histogram is a weighted sum of the all original values that each bucket represents, where bucket weight is equal to the number of values in the bucket.

MaxDiff: In a MaxDiff histogram, we consider the difference between each pair of adjacent values. A bucket boundary is established between pair for pairs having k-1 largest differences, where k is user-specified

V-optimal and MaxDiff are most accurate and practical histograms. Histograms are highly effective at approximating both sparse and dense data, as well as highly skewed and uniform data.

(Refer notes for problems on histograms)

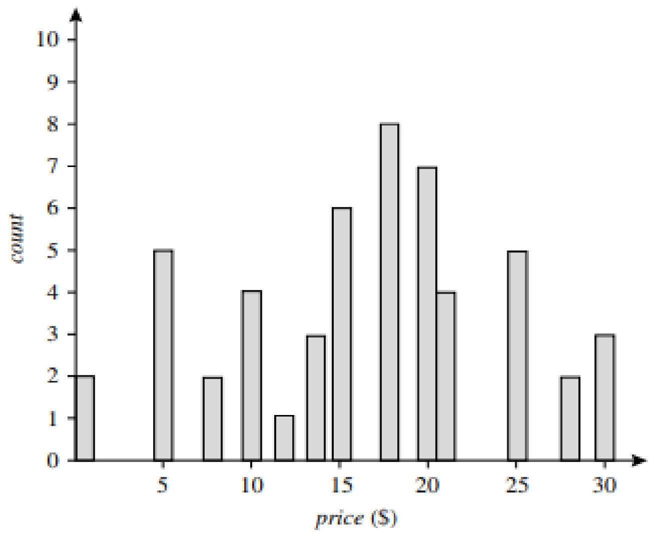


Figure 3.7 A histogram for *price* using singleton buckets—each bucket represents one price–value/frequency pair.

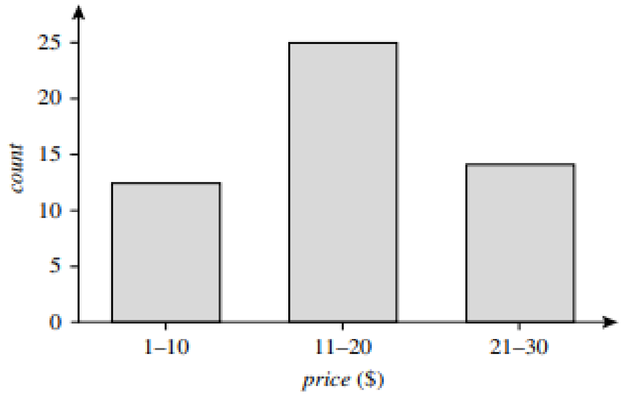
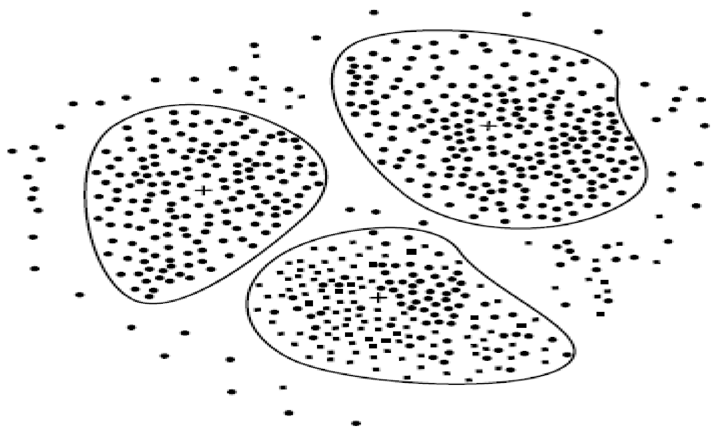


Figure 3.8 An equal-width histogram for *price*, where values are aggregated so that each bucket has a uniform width of $10.

2. **Clustering**

- Figure shows a 2-D plot of customer data with respect to customer locations in a city. Three data clusters are visible.
- Clustering techniques consider data tuples as objects. They partition the objects into groups, or clusters, so that objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters. Similarity is commonly defined in terms of how "close" the objects are in space, based on a distance function.
- The "quality" of a cluster may be represented by its diameter, the maximum distance between any two objects in the cluster. Centroid distance is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid (denoting the "average object," or average point in space for the cluster).

In data reduction, the cluster representations of the data are used to replace the actual data. The effectiveness of this technique depends on the data's nature. It is much more effective for data that can be organized into distinct clusters than for smeared data.



A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters. Each cluster centroid is marked with a "+", representing the average point in space for that cluster. Outliers may be detected as values that fall outside of the sets of clusters.

3. **Sampling**

Sampling can be used as a data reduction technique because it allows a large data set to be represented by a much smaller random data sample (or subset). Suppose that a large data set, D, contains N tuples. Let's look at the most common ways that we could sample D for data reduction, as illustrated in Figure 3.9.

1. **Simple random sample without replacement (SRSWOR) of size s:** This is created by drawing s of the N tuples from D (s < N), where the probability of drawing any tuple in D is 1/N, that is, all tuples are equally likely to be sampled.

2. **Simple random sample with replacement (SRSWR) of size s:** This is similar to SRSWOR, except that each time a tuple is drawn from D, it is recorded and then replaced. That is, after a tuple is drawn, it is placed back in D so that it may be drawn again.

3. **Cluster sample:** If the tuples in D are grouped into M mutually disjoint "clusters," then an SRS of s clusters can be obtained, where s < M. For example, tuples in a database are usually retrieved a page at a time, so that each page can be considered a cluster. A reduced data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the tuples.

4. **Stratified sample:** If D is divided into mutually disjoint parts called strata, a stratified sample of D is generated by obtaining an SRS at each stratum. This helps ensure a representative sample, especially when the data are skewed. For example, a stratified sample may be obtained from customer data, where a stratum is created for each customer age group. In this way, the age group having the smallest number of customers will be sure to be represented.

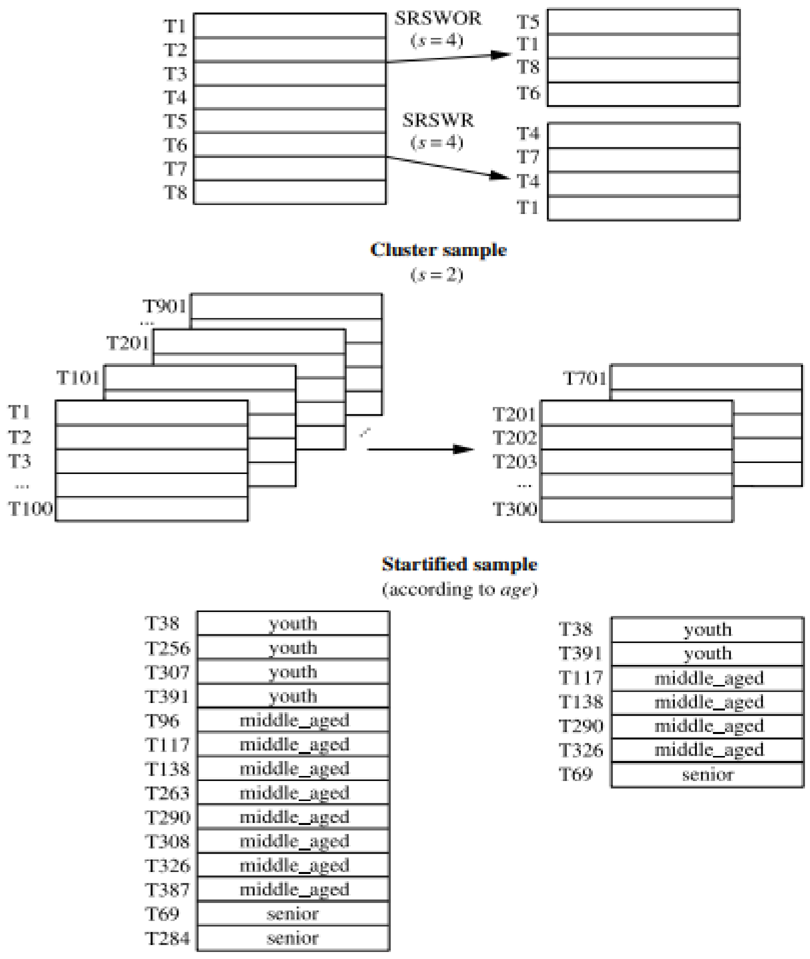


Figure 3.9 Sampling can be used for data reduction.

Discretization and concept hierarchy generation

Discretization techniques can be used to reduce the number of values for a continuous attribute, by dividing the range of attribute into intervals. Interval labels (e.g., 0–10, 11–20, etc.) can then be used to replace actual data values.

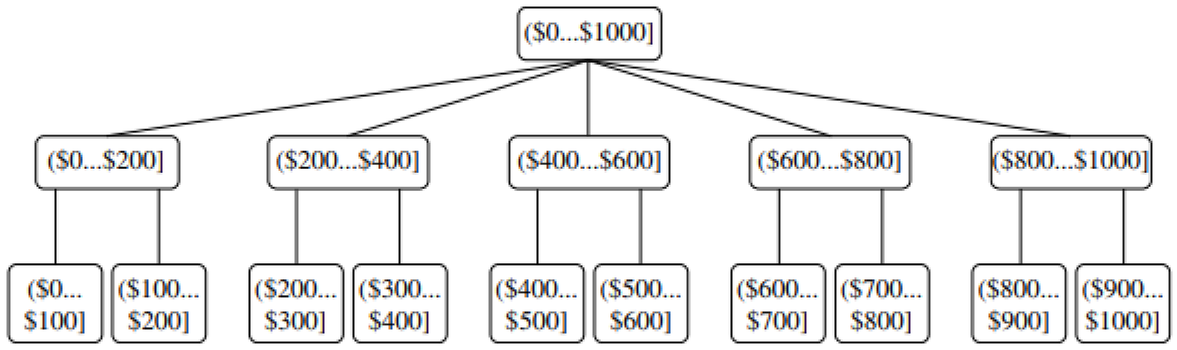
Techniques to generate decision trees for classification can be applied to discretization. Such techniques employ a top-down splitting approach. Unlike the other methods mentioned so far, decision tree approaches to discretization are supervised, that is, they make use of class label information.

For example, we may have a data set of patient symptoms (the attributes) where each patient has an associated diagnosis class label. Class distribution information is used in the calculation and determination of split-points (data values for partitioning an attribute range). Intuitively, the main idea is to select split-points so that a given resulting partition contains as many tuples of the same class as possible. Entropy is the most commonly used measure for this purpose. To discretize a numeric attribute, A, the method selects the value of A that has the minimum entropy as a split- point, and recursively partitions the resulting intervals to arrive at a hierarchical discretization. Such discretization forms a concept hierarchy for A.

A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts. A concept hierarchy for given numeric attribute defines a discretization of the attribute. Concept hierarchies can also be used to reduce the data by collecting and replacing low-level-concepts (such as numeric values for the attribute *age*) by higher level concepts (such as *young*, *middle-aged* or *senior)*.

Figure shows a concept hierarchy for the attribute price. More than one concept hierarchy can be defined for the same attribute to accommodate the needs of various users.

Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert. Fortunately, many hierarchies are implicit within the database schema and can be automatically defined at the schema definition level. The concept hierarchies can be used to transform the data into multiple levels of granularity.



A concept hierarchy for the attribute *price*, where an interval ($$X \ldots \$Y$] denotes the range from $$X$ (exclusive) to $$Y$ (inclusive).

Discretization and Concept Hierarchy Generation for Numeric Data

It is difficult to specify concept hierarchies for numeric attributes due to wide diversity of possible data ranges and the frequent updates of the data values. Such manual specification will be also quite arbitrary.

Concept hierarchies for numeric data can be constructed automatically based on data distribution analysis. We examine four methods for numeric concept hierarchy generation: binning, histogram analysis, cluster based analysis and entropy based discretization.

1. Binning

Binning is a top-down splitting technique based on a specified number of bins. These methods are also used as discretization methods for data reduction and concept hierarchy generation. For example, attribute values can be discretized by applying equal-width or equal-frequency binning, and then replacing each bin value by the bin mean or median, as in smoothing by bin means or smoothing by bin medians, respectively. These techniques can be applied recursively to the resulting partitions to generate concept hierarchies.

Binning does not use class information and is therefore an unsupervised discretization technique. It is sensitive to the user-specified number of bins, as well as the presence of outliers.

2. Histogram Analysis

Like binning, histogram analysis is an unsupervised discretization technique because it does not use class information. A histogram partitions the values of an attribute, A, into disjoint ranges called buckets or bins. Various partitioning rules can be used to define histograms .In an equal-width histogram, for example, the values are partitioned into equal-size partitions or ranges (e.g., earlier in Figure for price, where each bucket has a width of $10). With an equal-frequency histogram, the values are partitioned so that, ideally, each partition contains the same number of data tuples. The histogram analysis algorithm can be applied recursively to each partition in order to automatically generate a multilevel concept hierarchy, with the procedure terminating once a prespecified number of concept levels has been reached.

A minimum interval size can also be used per level to control the recursive procedure. This specifies the minimum width of a partition, or the minimum number of values for each partition at each level. Histograms can also be partitioned based on cluster analysis of the data distribution, or the minimum number of values for each partition at each level.

A concept hierarchy for ***price,*** generated is shown in Figure 3.13.

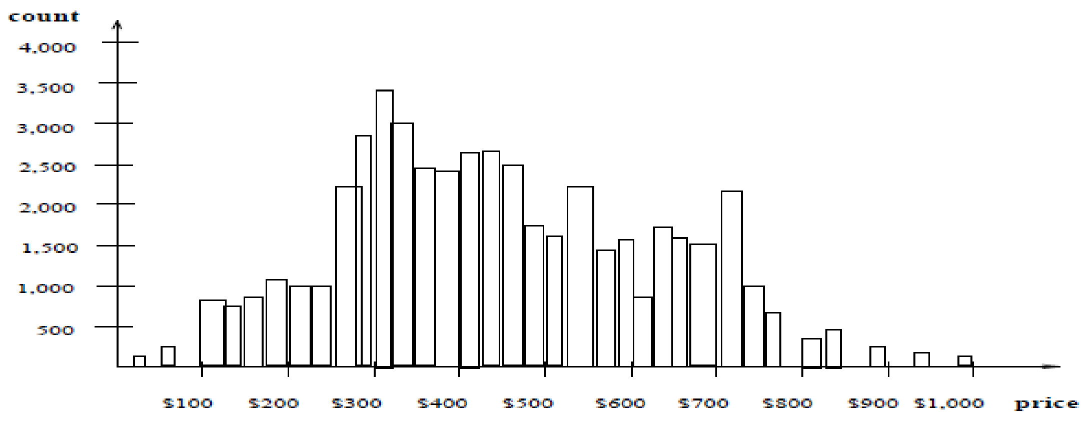


Figure 3.13: Histogram showing the distribution of values for the attribute *price*.

3. Cluster analysis: is a popular data discretization method.

A clustering algorithm can be applied to discretize a numeric attribute, A, by partitioning the values of A into clusters or groups. Clustering takes the distribution of A into consideration, as well as the closeness of data points, and therefore is able to produce high-quality discretization results. Clustering can be used to generate a concept hierarchy for A by following either a top-down splitting strategy or a bottom-up merging strategy, where each cluster forms a node of the concept hierarchy. In the former, each initial cluster or partition may be further decomposed into several sub clusters, forming a lower level of the hierarchy. In the latter, clusters are formed by repeatedly grouping neighboring clusters in order to form higher-level concepts.

4. **Entropy Based Discretization**

An information-based measure called entropy can be used to recursively partition the values of a numeric attribute A, resulting in a hierarchical discretization. Such a discretization forms a numerical concept hierarchy for the attribute. Given a set of data tuples S, the basic method for entropy based discretization is as follows:

Each value of A can be considered a potential interval boundary or threshold T. For ex. A value *v* of A can partition S into two subset satisfying the conditions A<*v* and A≥ *v* thereby creating a binary discretization.

Given S, the threshold value selected is the one that maximizes the information gain resulting from subsequent partitioning. The information gain is

$$I(S,T) = \frac{|S_1|}{|S|}Ent(S_1) + \frac{|S_2|}{|S|}Ent(S_2),$$

Where S1 and S2 are samples satisfying conditions A<*v* and A≥ *v*.

The entropy function Ent() is based on class distribution of the samples in the set.

The entropy of S1 = $-\sum_{i=1}^{m} p_i \log_2(p_i),$

Where pi is the probability of class i in S1. Determined by dividing the number of samples of class i in S1 by total number of samples in S1. The value of Ent (S2) can be calculated similarly.

The process of determining a threshold value is recursively applied to each partition obtained until some stopping criterion is met. Such as

Ent(s) – I(S, T) > **δ**

Entropy based discretization can reduce data size. Entropy based discretization uses class information. This makes it more likely that the interval bodies' are defined to occur in places that may help improve classification accuracy.

5. **Interval Merging by χ^2 Analysis**

- ChiMerge is a Chi Square-based discretization method
- Bottom-up approach by finding the best neighboring intervals and then merging these to form larger intervals, recursively.
- The method is supervised in that it uses class information.
- ChiMerge proceeds as follows.
    - Initially, each distinct value of a numerical attribute A is considered to be one interval.
    - Chi Square tests are performed for every pair of adjacent intervals.
    - Adjacent intervals with the least Chi Square values are merged together, because low Chi Square values for a pair indicate similar class distributions.
    - This merging process proceeds recursively until a predefined stopping criterion is met.

Concept hierarchy generation for categorical data

Categorical attributes have a finite (but possibly large) number of distinct values, with no ordering among the values. Examples include geographic location, job category, and item type.

Four methods for the generation of concept hierarchies for nominal data, are as follows.

1. **Specification of a partial ordering of attributes explicitly at the schema level by users or experts:** Concept hierarchies for nominal attributes o**r** dimensions typically involve a group of attributes. A user or expert can easily define a concept hierarchy by specifying a partial or total ordering of the attributes at the schema level. For example, suppose that a relational database contains the following group of attributes: street, city, province or state, and country. Similarly, a data warehouse location dimension may contain the same attributes. A hierarchy can be defined by specifying the total ordering among these attributes at the schema level such as street < city < province or state < country.

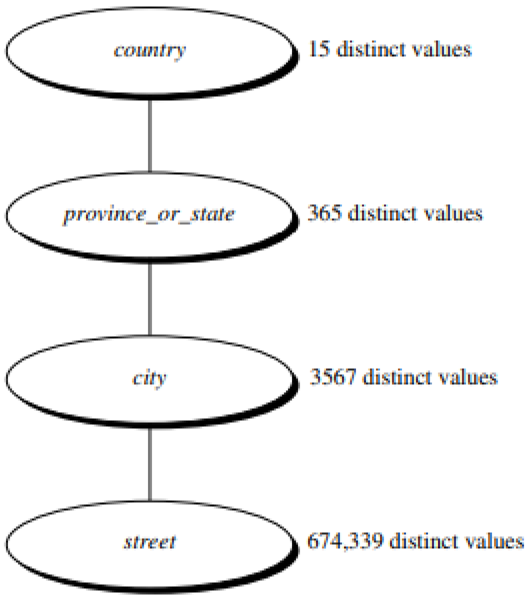
2. **Specification of a portion of a hierarchy by explicit data grouping**: This is essentially the manual definition of a portion of a concept hierarchy. In a large database, it is unrealistic to define an entire concept hierarchy by explicit value enumeration. On the contrary, we can easily specify explicit groupings for a small portion of intermediate-level data. For example, after specifying that state and country form a hierarchy at the schema level, a user could define some intermediate levels manually, such as "{Kerala, Tamilnadu} ⊂ South India and "{Punjab, Haryana"} ⊂ North India.

3. **Specification of a set of attributes, but not of their partial ordering**: A user may specify a set of attributes forming a concept hierarchy, but omit to explicitly state their partial ordering. The system can then try to automatically generate the attribute ordering so as to construct a meaningful concept hierarchy.

   An_attribute defining a high concept level (e.g., country) will usually contain a smaller number of distinct values than an attribute defining a lower concept level (e.g., street). Based on this observation, a concept hierarchy can be automatically generated based on the number of distinct values per attribute in the given attribute set. The attribute with the most distinct values is placed at the lowest hierarchy level. The lower the number of distinct values an attribute has, the higher it is in the generated concept hierarchy.

This heuristic rule works well in many cases. A concept hierarchy for location can be generated automatically, as illustrated in Figure. First, sort the attributes in ascending order based on the number of distinct values in each attribute. This results in the following (where the number of distinct values per attribute is shown in parentheses): country (15), province or state (365), city (3567), and street (674,339).

Second, generate the hierarchy from the top down according to the sorted order, with the first attribute at the top level and the last attribute at the bottom level.



Automatic generation of a schema concept hierarchy based on the number of distinct attribute values.

Specification of only a partial set of attributes: Sometimes a user can be careless when defining a hierarchy, or have only a vague idea about what should be included in a hierarchy. Consequently, the user may have included only a small subset of the relevant attributes in the hierarchy specification. For example, instead of including all of the hierarchically relevant attributes for location, the user may have specified only street and city. To handle such partially specified hierarchies, it is important to embed data semantics in the database schema so that attributes with tight semantic connections can be pinned together. In this way, the specification of one attribute may trigger a whole group of semantically tightly linked attributes to be "dragged in" to form a complete hierarchy.

Example - Suppose that a data mining expert (serving as an administrator) has pinned together the five attributes number, street, city, province or state, and country, because they are closely linked semantically regarding the notion of location. If a user were to specify only the attribute city for a hierarchy defining location, the system can automatically drag in all five semantically related attributes to form a hierarchy. The user may choose to drop any of these attributes (e.g., number and street) from the hierarchy, keeping city as the lowest conceptual level.

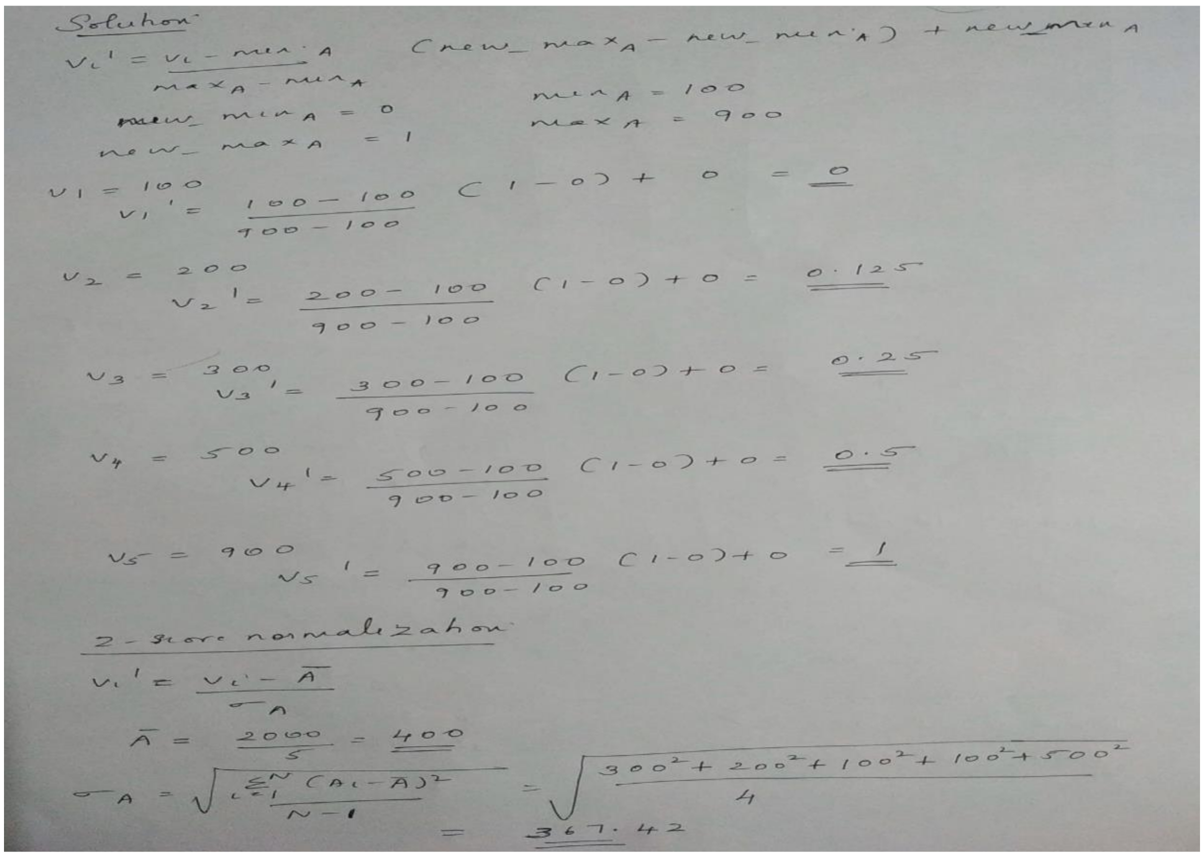
MODULE II UNIVERSITY QUESTIONS

MAY 2019

1. Why do we need data transformation? What are the different ways of data transformation? (4 marks)
2. Use the two methods below to normalize the following group of data:
   1000, 2000,3000,5000,9000 (4 marks)
   i) min-max normalization by setting min=0 and max=1
   ii) z-score normalization
   Solution: Refer class notes
3. **3.** Summarize the various pre-processing activities involved in data mining. (9 marks)

OCTOBER 2019

1. Use the two methods below to normalize the following group of data:
   100,200,300,500,900 (4 marks)
   i) min-max normalization by setting min=0 and max=1
   ii) z-score normalization



<u>Solution</u>

$$v_i' = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

$new\_min_A = 0$ $min_A = 100$

$new\_max_A = 1$ $max_A = 900$

$v1 = 100$

$$v_1' = \frac{100 - 100}{900 - 100}(1 - 0) + 0 = \underline{\underline{0}}$$

$v_2 = 200$

$$v_2' = \frac{200 - 100}{900 - 100}(1 - 0) + 0 = \underline{\underline{0.125}}$$

$v_3 = 300$

$$v_3' = \frac{300 - 100}{900 - 100}(1 - 0) + 0 = \underline{\underline{0.25}}$$

$v_4 = 500$

$$v_4' = \frac{500 - 100}{900 - 100}(1 - 0) + 0 = \underline{\underline{0.5}}$$

$v_5 = 900$

$$v_5' = \frac{900 - 100}{900 - 100}(1 - 0) + 0 = \underline{\underline{1}}$$

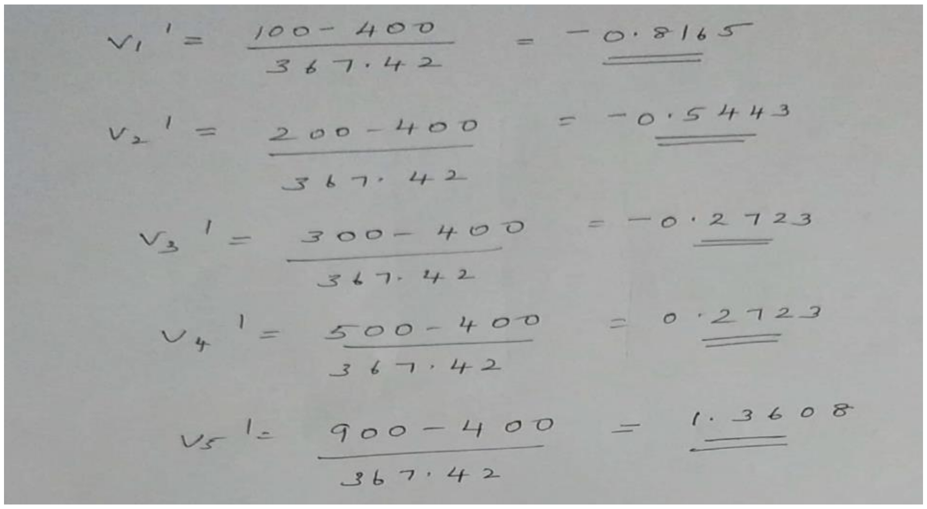
<u>z-score normalization</u>

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A}$$

$$\bar{A} = \frac{2000}{5} = \underline{\underline{400}}$$

$$\sigma_A = \sqrt{\frac{\sum_{i=1}^{N}(A_i - \bar{A})^2}{N-1}} = \sqrt{\frac{300^2 + 200^2 + 100^2 + 100^2 + 500^2}{4}}$$

$$= \underline{\underline{367.42}}$$



$$v_1' = \frac{100-400}{367.42} = -0.8165$$

$$v_2' = \frac{200-400}{367.42} = -0.5443$$

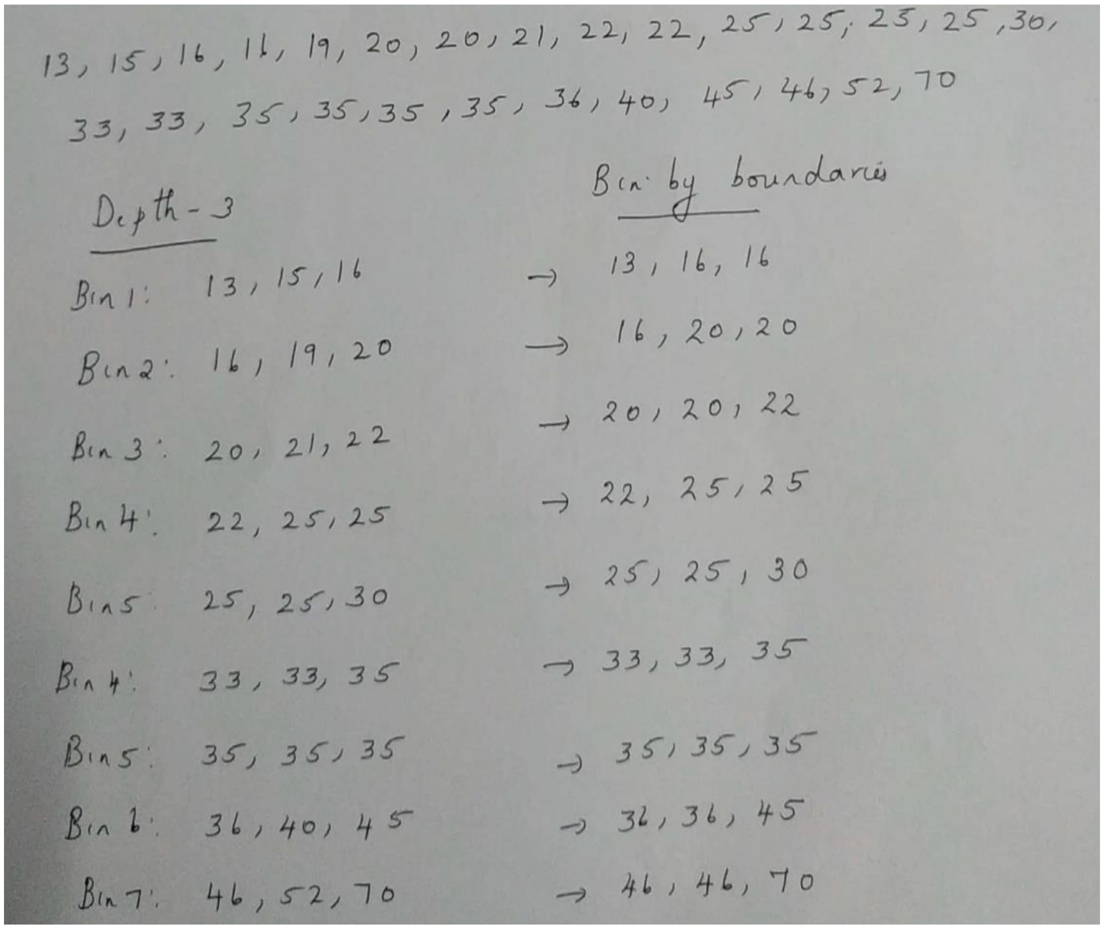
$$v_3' = \frac{300-400}{367.42} = -0.2723$$

$$v_4' = \frac{500-400}{367.42} = 0.2723$$

$$v_5' = \frac{900-400}{367.42} = 1.3608$$

2. Explain the attribute selection method in decision trees. (4 marks)

3. The following data is given in increasing order for the attribute age:
13,15,16,16,19,20,20,21,22,22,,25,25,25,25,30,33,33,35,35,35,35, 36,40,45,46,52,70.
a) Use smoothing by bin boundaries to smooth these data using bin depth of 3. (3 marks)



13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70

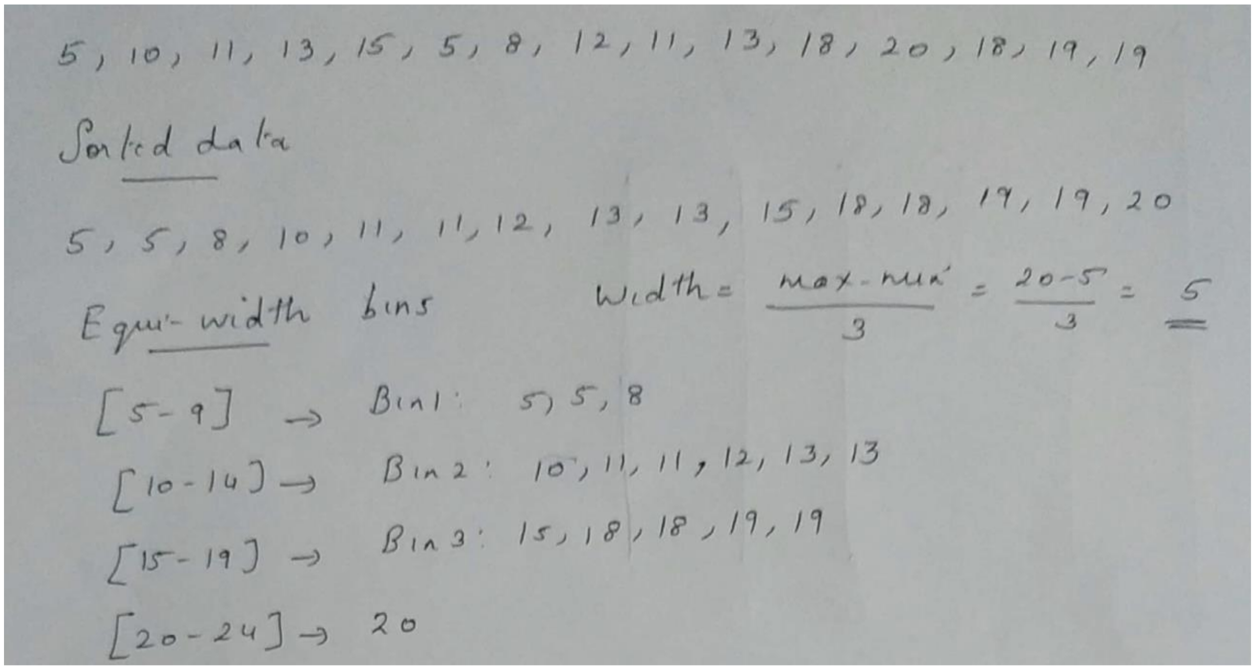
Depth - 3		Bin by boundaries
Bin 1:	13, 15, 16	→ 13, 16, 16
Bin 2:	16, 19, 20	→ 16, 20, 20
Bin 3:	20, 21, 22	→ 20, 20, 22
Bin 4:	22, 25, 25	→ 22, 25, 25
Bin 5:	25, 25, 30	→ 25, 25, 30
Bin 4:	33, 33, 35	→ 33, 33, 35
Bin 5:	35, 35, 35	→ 35, 35, 35
Bin 6:	36, 40, 45	→ 36, 36, 45
Bin 7:	46, 52, 70	→ 46, 46, 70

b) How might you determine outliers in the data (3 marks)
c) What other methods are there for data smoothing? (3marks)

4. Explain the following procedures for attribute subset selection
   a) Stepwise forward selection (3 marks)
   b) Stepwise backward selection (3 marks)
   c) A combination of forward and backward selection (3 marks)
5. Real-world data tend to be incomplete, noisy, and inconsistent. What are the various approaches adopted to clean the data? (5 marks)

SEPTEMBER 2020

1. What is the purpose of data discretization in data mining? List out any four data discretization strategies. (4marks)
2. List out any four methods to handle missing attribute values in a dataset. (2 marks)
3. Suppose a group of 15 sales price records has been given as follows:
   5, 10, 11, 13, 15, 5, 8, 12, 11, 13, 18, 20, 18, 19, 19
   Draw a three bucket Equi- width histogram (3 marks)



5, 10, 11, 13, 15, 5, 8, 12, 11, 13, 18, 20, 18, 19, 19

Sorted data

5, 5, 8, 10, 11, 11, 12, 13, 13, 15, 18, 18, 19, 19, 20

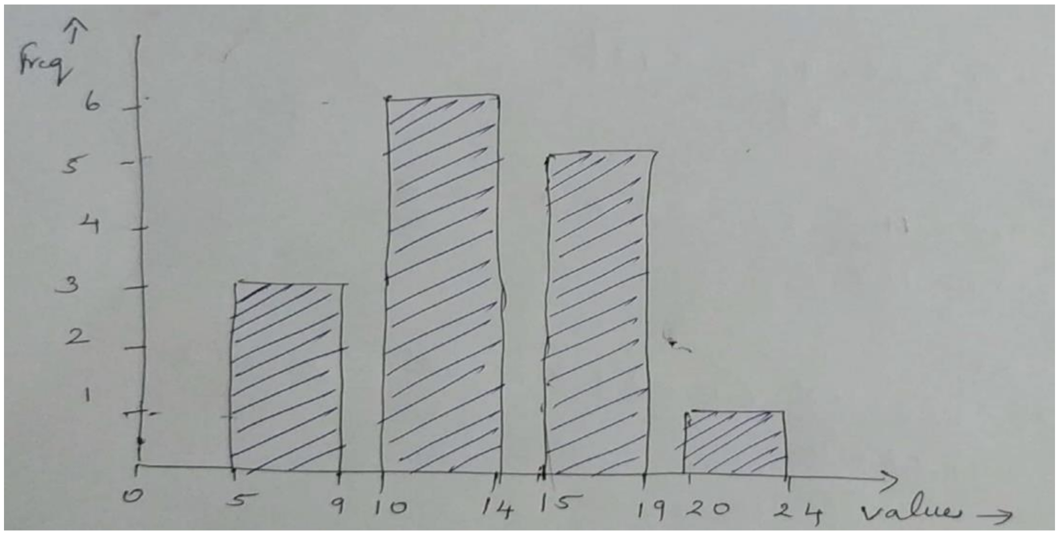
Equi-width bins Width = $\frac{max - min}{3} = \frac{20-5}{3} = 5$

[5-9] → Bin1: 5, 5, 8

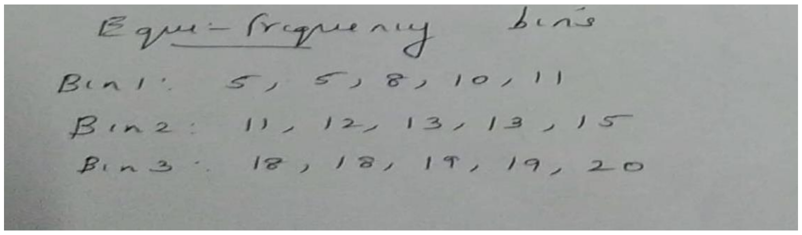
[10-14] → Bin 2: 10, 11, 11, 12, 13, 13

[15-19] → Bin 3: 15, 18, 18, 19, 19

[20-24] → 20



Draw a three bucket Equi- depth histogram. (3 marks)

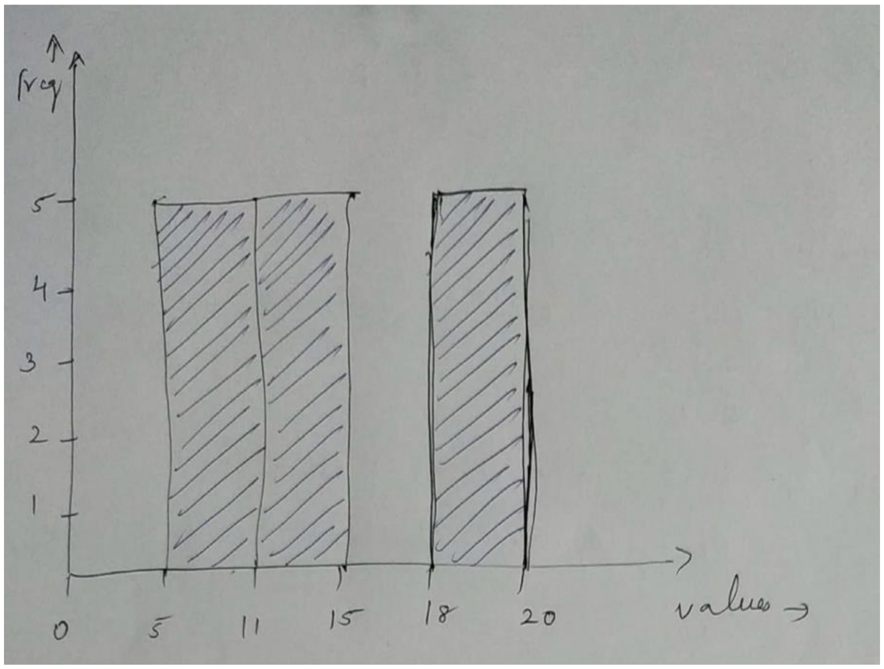


Equi-frequency bins

Bin 1: 5, 5, 8, 10, 11

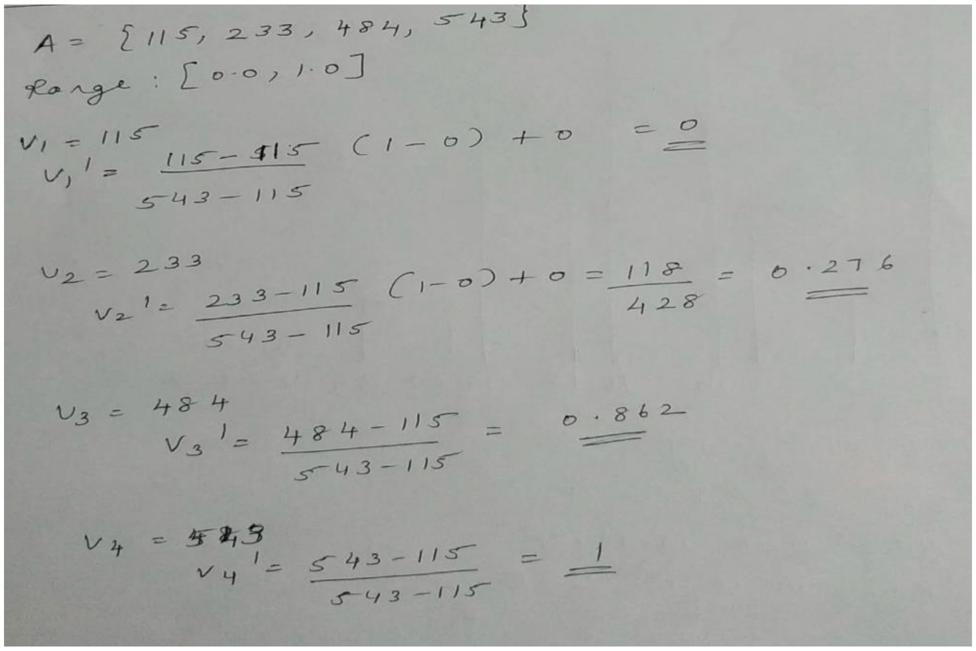
Bin 2: 11, 12, 13, 13, 15

Bin 3: 18, 18, 19, 19, 20



4. How Numerosity reduction is done by max diff histogram. (3 marks).
5. A set of data is given: A = {115, 233, 484, 543}. Normalize the data by min-max normalization. (Range: [0.0, 1.0]). (4 marks).

Solution



$A = \{115, 233, 484, 543\}$

Range : $[0.0, 1.0]$

$v_1 = 115$

$$v_1' = \frac{115 - 115}{543 - 115}(1 - 0) + 0 = \underline{\underline{0}}$$

$v_2 = 233$

$$v_2' = \frac{233 - 115}{543 - 115}(1 - 0) + 0 = \frac{118}{428} = \underline{\underline{0.276}}$$

$v_3 = 484$

$$v_3' = \frac{484 - 115}{543 - 115} = \underline{\underline{0.862}}$$

$v_4 = 543$

$$v_4' = \frac{543 - 115}{543 - 115} = \underline{\underline{1}}$$

6. Explain different OLAP operations on multi-dimensional data with suitable examples. (6 marks).

Prepared by
Dr. Hema Krishnan
Assistant Professor Senior Grade, CSE, FISAT