76938

Q.1	(a)	**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. Missing Values 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. 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. 4. Use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing value: 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. If the data distribution for a given class is skewed, the median value is a better choice. 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 **Data smoothing** 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. 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. 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.
	(b)	**True positives** .TP/: These refer to the positive tuples that were correctly labeled by the classifier. Let TP be the number of true positives. **True negatives** .TN/: These are the negative tuples that were correctly labeled by the classifier. Let TN be the number of true negatives. **False positives** .FP/: These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class buys computer D no for which the classifier predicted buys computer D yes). Let FP be the number of false positives. **False negatives** .FN/: These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys computer D yes for which the classifier predicted buys computer D no). Let FN be the number of false negatives. These terms are summarized in the **confusion matrix** of Figure The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes.

Predicted class

Actual class		yes	no	Total
	yes	TP	FN	P
	no	FP	TN	N
	Total	P'	N'	P + N

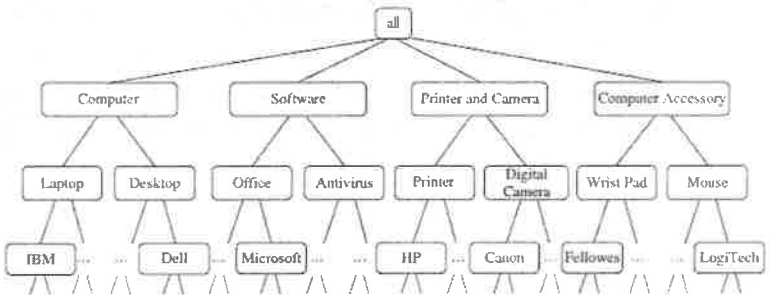
Classes	buys_computer = yes	buys_computer = no	Total	Recognition (%)
buys_computer = yes	6954	46	7000	99.34
buys_computer = no	412	2588	3000	86.27
Total	7366	2634	10,000	95.42

(c) **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;

(d) Association rules generated from mining data at multiple abstraction levels are called **multiple-level** or **multi level association rules**. Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework. In general, a top-down strategy is employed, where counts are accumulated for the calculation of frequent itemsets at each concept level, starting at concept level 1 and working downward in the hierarchy toward the more specific concept levels, until no more frequent itemsets can be found. For each level, any algorithm for discovering frequent itemsets may be used, such as Apriori or its variations. Example:

Task-Relevant Data, *D*

TID	Items Purchased
T100	Apple 17" MacBook Pro Notebook, HP Photosmart Pro b9180
T200	Microsoft Office Professional 2010, Microsoft Wireless Optical Mouse 5000
T300	Logitech VX Nano Cordless Laser Mouse, Fellowes GEL Wrist Rest
T400	Dell Studio XPS 16 Notebook, Canon PowerShot SD1400
T500	Lenovo ThinkPad X200 Tablet PC, Symantec Norton Antivirus 2010
...	...



A concept hierarchy defines a sequence of mappings from a set of low-level

		concepts to a higher-level, more general concept set. Data can be generalized by replacing low-level concepts within the data by their corresponding higher-level concepts, or ancestors, from a concept hierarchy. Figure shows concept hierarchy has five levels, respectively referred to as levels 0 through 4, starting with level 0 at the root node for all (the most general abstraction level). Here, level 1 includes computer, software, printer and camera, and computer accessory; level 2 includes laptop computer, desktop computer, office software, antivirus software, etc.; and level 3 includes Dell desktop computer, . . . ,Microsoft office software, etc. Level 4 is the most specific abstraction level of this hierarchy.
Q.2	(a)	The **na¨□ve Bayesian** classifier, works as follows: **1.** Let D be a training set of tuples and their associated class labels. Each tuple is represented by an n-dimensional attribute vector, **X** D .x1, x2, : : : , xn/, depicting n measurements made on the tuple from n attributes, respectively, A1, A2, : : : , An. **2.** Suppose that there are m classes, C1, C2, : : : , Cm. Given a tuple, **X**, the classifier will predict that **X** belongs to the class having the highest posterior probability, conditioned on **X**. That is, the na¨□ve Bayesian classifier predicts that tuple **X** belongs to the class Ci if and only if $P(C_i\|X) > P(C_j\|X)$ for $1 \le j \le m, j \ne i.$ Thus, we maximize P.(Ci |**X**). The class Ci for which P.(Ci|**X**) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem $P(C_i\|X) = \frac{P(X\|C_i)P(C_i)}{P(X)}.$ As P(.**X**) is constant for all classes, only P.(**X**|Ci) P(Ci) needs to be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P.(C1) P.(C2) _ _ _ P.(Cm), and we would therefore maximize P.(**X**|Ci). Otherwise, we maximize P.(**X**|Ci)P.(Ci). Note that the class prior probabilities may be estimated by P.(Ci)=|Ci,D|/|D| where |Ci,D| is the number of training tuples of class Ci in D. **4.** Given data sets with many attributes, it would be extremely computationally expensive To reduce computation in evaluating P.(**X**|Ci), the na¨□ve assumption of **class-conditional independence** is made. This presumes that the attributes' values are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus, $P(X\|C_i) = \prod_{k=1}^{n} P(x_k\|C_i)$ $= P(x_1\|C_i) \times P(x_2\|C_i) \times \cdots \times P(x_n\|C_i).$
	(b)	**Bagging** Given a set, D, of d tuples, **bagging** works as follows. For iteration i .i D 1, 2, : : : , k/, a training set, Di , of d tuples is sampled with replacement from the original set of tuples, D. Bagging stands for bootstrap aggregation. Each training set is a bootstrap sample, Because sampling with replacement is used, some of the original tuples of D may not be included in Di , whereas others may occur more than once. A classifier model, Mi , is learned for each training set, Di . To

		classify an unknown tuple, **X**, each classifier, Mi , returns its class prediction, which counts as one vote. The bagged classifier, M_, counts the votes and assigns the class with the most votes to **X**. Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple. The bagged classifier often has significantly greater accuracy than a single classifier derived from D, the original training data. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.
Q.3	(a)	The main strategy behind density-based clustering methods, is to find clusters as dense regions in the data space, separated by sparse regions. 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. 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 neighbourhood** 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 □ -neighborhood of the object contains at least MinPts objects. Core objects are the pillars of dense regions. Given a set, D, of objects, we can identify all core objects with respect to the given parameters, □ and MinPts. The clustering task is therein reduced to using core objects and their neighborhoods to form dense regions, where the dense regions are clusters. **p** is directly density-reachable from another object **q** if and only if **q** is a core object and in □ neighborhood of **q**. Using the directly density-reachable relation, a core bject can "bring" all objects from its □ -neighborhood into a dense region.
	(b)	An **outlier** is a data object that deviates significantly from the rest of the objects, as if it were generated by a different mechanism. Outliers are different from noisy data. In general, noise is not interesting in data analysis, including outlier detection. For example, in credit card fraud detection, a customer's purchase behavior can be modeled as a random variable. A customer may generate some "noise transactions" that may seem like "random errors" or "variance," such as by buying a bigger lunch one day, or having one more cup of coffee than usual. Such transactions should not be treated as outliers; otherwise, the credit card company would incur heavy costs from verifying that many transactions. The company may also lose customers by bothering them with multiple false alarms. As in many other data analysis and data mining tasks, noise should be removed before outlier detection. Outliers are interesting because they are suspected of not being generated by the same mechanisms as the rest of the data. Therefore, in outlier detection, it is important to justify *why* the outliers detected are generated by some other mechanisms. **Types of Outliers** In general, outliers can be classified into three categories, namely global outliers, contextual (or conditional) outliers, and collective outliers. **Global Outliers** In a given data set, a data object is a **global outlier** if it deviates significantly

		from the rest of the data set. Global outliers are sometimes called *point anomalies*, and are the simplest type of outliers **Contextual Outliers** "The temperature today is 28_C. Is it exceptional (i.e., an outlier)?" It depends, for example, on the time and location! If it is in winter in Toronto, yes, it is an outlier. If it is a summer day in Toronto, then it is normal. **Collective Outliers** Suppose for the shop *AllElectronics*, there are thousands of orders and shipments every day. If the shipment of an order is delayed, it may not be considered an outlier because, statistically, delays occur from time to time. However, you have to pay attention if 100 orders are delayed on a single day. Those 100 orders as a whole form an outlier, although each of them may not be regarded as an outlier if considered individually.
Q.4	(a)	**Market basket analysis**. This process analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets". The discovery of these associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket? This information can lead to increased sales by helping retailers do selective marketing and plan their shelf space. Items that are frequently purchased together can be placed in proximity to further encourage the combined sale of such items. If customers who purchase computers also tend to buy antivirus software at the same time, then placing the hardware display close to the software display may help increase the sales of both items. In an alternative strategy, placing hardware and software at opposite ends of the store may entice customers who purchase such items to pick up other items along the way. For instance, after deciding on an expensive computer, a customer may observe security systems for sale while heading toward the software display to purchase antivirus software, and may decide to purchase a home security system as well. Market basket analysis can also help retailers plan which items to put on sale at reduced prices. If customers tend to purchase computers and printers together, then having a sale on printers may encourage the sale of printers as well as computers. If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of that item. Each basket can then be represented by a Boolean vector of values assigned to these variables. The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently associated or purchased together. These patterns can be represented in the form of association rules.
	(b)	To improve efficiency : following techniques can be used: **Hash-based technique** A hash-based technique can be used to reduce the size of the candidate k-itemsets, Ck, for $k > 1$. For example, when scanning each transaction in the database to generate the frequent 1-itemsets, L1, we can generate all the 2-itemsets for each transaction, hash them into the different buckets of a hash table structure, and increase the corresponding bucket counts. A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent and thus should be removed from the candidate set.

		Transaction reduction (reducing the number of transactions scanned in future iterations): A transaction that does not contain any frequent k-itemsets can be marked or removed from further consideration. **Partitioning** (partitioning the data to find candidate itemsets): A partitioning technique can be used that requires just two database scans to mine the frequent itemsets. **Sampling** (mining on a subset of the given data): The basic idea of the sampling approach is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D. **Dynamic itemset counting** (adding candidate itemsets at different points during a scan): A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan. This leads to fewer database scans than with Apriori for finding all the frequent itemsets.
Q.5	(a)	Mean58 Median54 Mode70 Interquartile range16
	(b)	BI(Business Intelligence) is a set of processes, architectures, and technologies that convert raw data into meaningful information that drives profitable business actions. It is a suite of software and services to transform data into actionable intelligence and knowledge. BI has a direct impact on organization's strategic, tactical and operational business decisions. BI supports fact-based decision making using historical data BI tools perform data analysis and create reports, summaries, dashboards, maps, graphs, and charts to provide users with detailed intelligence about the nature of the business. A decision support system (DSS) is a computerized program used to support determinations, judgments, and courses of action in an organization or a business. A DSS sifts through and analyzes massive amounts of data, compiling comprehensive information that can be used to solve problems and in decision-making. Typical information used by a DSS includes target or projected revenue, sales figures or past ones from different time periods, and other inventory- or operations-related data. A decision support system gathers and analyzes data, synthesizing it to produce comprehensive information reports. In this way, as an informational application, a DSS differs from an ordinary operations application, whose function is just to collect data. The DSS can either be completely computerized or powered by humans. In some cases, it may combine both. The ideal systems analyze information and actually make decisions for the user. At the very least, they allow human users to make more informed decisions at a quicker pace.
Q.6	(a)	An attribute is a data field, representing a characteristic or feature of a data object. Attributes describing a customer object can include, for example, customer ID, name, and address. Observed values for a given attribute are known as observations. A set of attributes used to describe a given object is called an attribute vector (or feature vector). The type of an attribute Nominal Attributes Nominal means "relating to names." The values of a nominal attribute are

symbols or names of things. Each value represents some kind of category, code, or state, and so nominal attributes are also referred to as categorical. The values do not have any meaningful order. Eg. hair color and marital status describing person objects.

Binary Attributes

A binary attribute is a nominal attribute with only two categories or states: 0 or 1, where 0 typically means that the attribute is absent, and 1 means that it is present. Binary attributes are referred to as Boolean if the two states correspond to true and false.

Eg. Given the attribute smoker describing a patient object, 1 indicates that the patient smokes, while 0 indicates that the patient does not.

A binary attribute is symmetric if both of its states are equally valuable and carry the same weight; that is, there is no preference on which outcome should be coded as 0 or 1. One such example could be the attribute gender having the states male and female.

A binary attribute is asymmetric if the outcomes of the states are not equally important,such as the positive and negative outcomes of a medical test for HIV.

Ordinal Attributes

An ordinal attribute is an attribute with possible values that have a meaningful order or ranking among them, but the magnitude between successive values is not known. examples of ordinal attributes include grade (e.g., A+,A, B+,B, C+, C) and professional rank..

Numeric Attributes

A numeric attribute is quantitative; that is, it is a measurable quantity, represented in integer or real values. Numeric attributes can be interval-scaled or ratio-scaled.

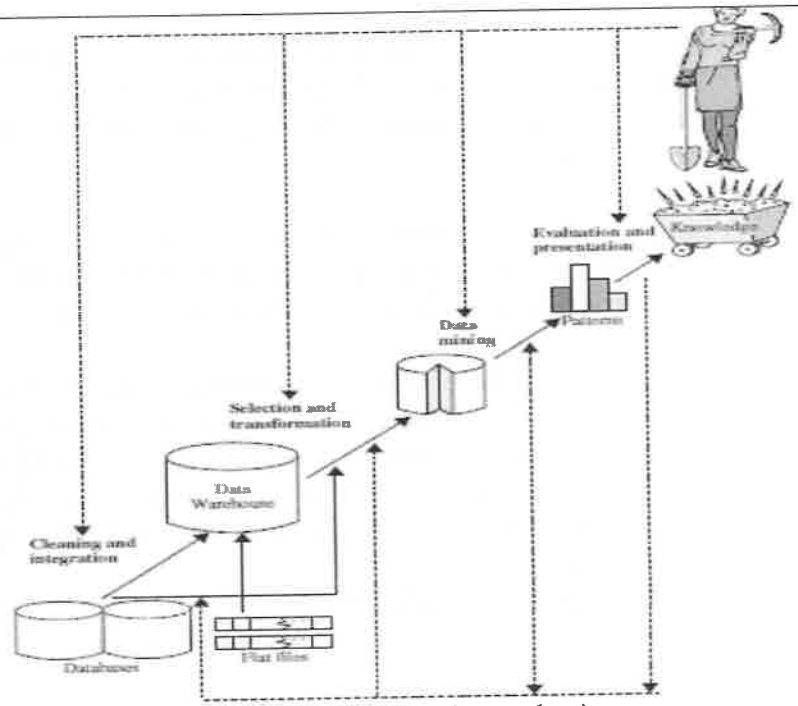
Interval-Scaled Attributes

Interval-scaled attributes are measured on a scale of equal-size units. The values of interval-scaled attributes have order and can be positive, 0, or negative. Thus, in addition to providing a ranking of values, such attributes allow us to compare and quantify the difference between values. A temperature attribute is interval-scaled.

Ratio-Scaled Attributes

A ratio-scaled attribute is a numeric attribute with an inherent zero-point. That is, if a measurement is ratio-scaled, we can speak of a value as being a multiple (or ratio) of another value. examples of ratio-scaled attributes include count attributes such as years of experience (e.g., the objects are employees) and number of words (e.g., the objects are documents).

(b)



1. Data cleaning (to remove noise and inconsistent data)
2. Data integration (where multiple data sources may be combined)
3. Data selection (where data relevant to the analysis task are retrieved from the database)
4. Data transformation (where data are transformed and consolidated into forms
appropriate for mining by performing summary or aggregation operations)
5. Data mining (an essential process where intelligent methods are applied to extract
data patterns)
6. Pattern evaluation (to identify the truly interesting patterns representing knowledge
based on interestingness measures)
7. Knowledge presentation (where visualization and knowledge representation techniques
are used to present mined knowledge to users)
Steps 1 through 4 are different forms of data preprocessing, where data are prepared
for mining. The data mining step may interact with the user or a knowledge base. The
interesting patterns are presented to the user and may be stored as new knowledge in the
knowledge base.