**FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING**

**Department of Computer Engineering**

**Class Test 1**

**(2019-20)**

**Class/Sem./Branch –TE/VI/COMP                                                 Course code: CSC603**

**Subject: Data warehousing and mining (DWM)                          Date: 28/03/2020**

**Total marks: 20**

**Name: Vedant Sahai                Roll no. 8364**

1. Bayesian classifiers is

|  |  |
| --- | --- |
| A. | A class of learning algorithm that tries to find an optimum classification of a set of examples using the probabilistic theory. |
| B. | Any mechanism employed by a learning system to constrain the search space of a hypothesis |
| C. | An approach to the design of learning algorithms that is inspired by the fact that when people encounter new situations, they often explain them by reference to familiar experiences, adapting the explanations to fit the new situation. |
| D. | None of these |

Ans: - A.]

1. Decision Trees are built using:

Select one:

* 1. heuristics

b. greedy algorithms

c. dynamic programming

d. divide and conquer strategy

Ans: - B.]

1. Give some alternative terms for data mining.

Ans: -

Data Archaeology, Information Harvesting, Information Discovery, Knowledge Extraction, Database Mining

1. What is meant by pattern?

Ans: -

**Patterns** are patterns (e.g., itemsets, subsequence, or substructures) that appear frequently in a data set. For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set is a [*frequent itemset*](https://www.sciencedirect.com/topics/computer-science/frequent-itemsets). A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (frequent) sequential pattern. A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or sub sequences. If a substructure occurs frequently, it is called a (frequent) structured pattern. Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data. Moreover, it helps in [data classification](https://www.sciencedirect.com/topics/computer-science/data-classification), clustering, and other [data mining](https://www.sciencedirect.com/topics/computer-science/data-mining) tasks. Thus, [frequent pattern mining](https://www.sciencedirect.com/topics/computer-science/mining-frequent-pattern) has become an important data mining task and a focused theme in [data mining research](https://www.sciencedirect.com/topics/computer-science/data-mining-research)

1. The problem of Curse of Dimensionality is associated with:

Select one:

a. increasing data points

b. increasing noise in data

c. increasing dimensions

d. increasing users

Ans: - C.]

1. Which of following function selects a median as centroid-

a-k-means

b-k-medoids

c-optics

Ans: - B.]

1. Give few techniques to improve the efficiency of Apriori algorithm.

Ans: -

Many variations of the Apriori algorithm have been proposed that focus on improving the efficiency of the Apriori algorithm. Several of these variations are summarized as follows:

1. 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-itemsetes, L1, from the candidate 1-itemsets in C1, we can generate all of the 2-itemsets for each transaction, hash them into a different buckets of a hash table structure and increase the corresponding bucket counts: a. H(x,y)=((order of x)X10+(order of y)) mod 7 b. A 2-itemset whose corresponding bucket count in the hash table is below the threshold cannot be frequent and thus should be removed from the candidate set.

2. Transaction reduction – a transaction that does not contain any frequent itemset cannot contain any frequent k+1 itemsets. Therefore, such a transaction can be marked or removed from further consideration because subsequent scans of the database for j-itemsets, where j>k, will not require it. 3. 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. It consists of two phases. In phase I, the algorithm subdivides the transactions of D into n non-overlapping partitions. If the minimum support threshold for transactions in D is min\_sup, then the minimum support count for a partition is min\_sup X the number of transactions in that partition. For each partition, all frequent itemsets within the partition are found. These are referred to as local frequent itemsets. A local frequent itemset may or may not be frequent with respect to the entire database, D. Any itemset that is potentially frequent with respect to D must occur as a frequent itemset in at least one of the partitions. Therefore, all local frequent itemsets are candidate itemsets with respect to D. The collection of frequent itemsets from all partitions forms the global candidate itemsets with respect to D. In phase II, a second 10 scan of D is conducted in which the actual support of each candidate is assessed in order to determine the global frequent itemsets

4. Sampling (mining on a subset of a 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. In this way, we trade off some degree of accuracy against efficiency. The sample size of S is such that the search for frequent itemsets in S can be done in main memory, and so only one scan of the transactions in S in required overall. Because we are searching for frequent itemsets in S rather than in D, it is possible that we will miss some of the global frequent itemsets. To lessen this possibility, we use a lower support threshold than minimum support to find the frequent itemsets local to S.

5. 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, which determines new candidate itemsets only immediately before each complete database scan. The resulting algorithm requires fewer database scan than Apriori.

1. What are the things suffering the performance of Apriori candidate generation technique?

Ans: -

Apriori, while historically significant, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms. Candidate generation generates large numbers of subsets (the algorithm attempts to load up the candidate set with as many as possible before each scan). Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any maximal subset S only after all ({\displaystyle 2^{|S|}-1}2^(|S|) -1) of its proper subsets. The algorithm scans the database too many times, which reduces the overall performance. Due to this, the algorithm assumes that the database is Permanent in the memory. Also, both the time and space complexity of this algorithm are very high: {\displaystyle O(2^{|D|})}O(2^|D|), thus exponential, where {\displaystyle |D|}|D| is the horizontal width (the total number of items) present in the database. Later algorithms such as [Max-Miner](https://en.wikipedia.org/w/index.php?title=Max-Miner&action=edit&redlink=1) try to identify the maximal frequent item sets without enumerating their subsets, and perform "jumps" in the search space rather than a purely bottom-up approach.

1. Describe the method of generating frequent itemsets without candidate generation.

Ans: -

In many cases the Apriori candidate generate-and-test method significantly reduces the size of candidate sets, leading to good performance gain. However it can suffer from two nontrivial costs: Transactions in D Divide D into n partitions Find the frequent itemsets local to each partition (1 scan) Combine all local frequent itemsets to form candidate itemset Find global frequent itemsets among candidates (1 scan) Frequent itemsets in D 11 It may need to generate a huge number of candidate sets.

For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 candidate 2-itemsets. It may need to repeatedly scan the database and check a large set of candidates by pattern matching. The solution is the frequent-pattern growth, or simply FP-growth, which mines the complete set of frequent itemsets without candidate generation. This method adopts a divide-and-conquer strategy as follows: first it compresses the database representing frequent items into frequent-pattern tree, or FP-tree, which retains the itemset association information. It then divides the compressed database into a set of conditional databases, each associated with one frequent item or pattern fragment, and mines each such database separately.

Let us create the FP-tree for the example from Table 3:

• First we scan the database and determine the set of frequent items (1-itemsets) and their support counts(frequencies): L={{I2:7},{I1:6},{I3:6},{I4:2},{I5:2}}

• Then we create the root of the FP-tree and label it with “null”

• We take each transaction, sort the items according to descending support count, and create a branch for it. For example, the scan of the first transaction “T100:I1, I2, I5”, which contain tree items: I2, I1 and I5 in sorted descending, leads to the construction of the first branch of the tree: (I2:1), (I1:1), (I5:1).

• The second transaction T200 contains the items I2 and I4. This would result a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common prefix, i2, with the existing path for T100. Therefore, we instead increment the count of the 12 nodes by 1 and create a new node (I4:1), which is linked as a child of (I2:2). In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1 and nodes for the items following the prefix are created and linked accordingly. 12 To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links. In this way the problem of mining frequent pattern in database is transformed to that of mining the FP-tree. The FP-tree is mined as follows: Start from each frequent length-1 pattern, as an initial suffix pattern, construct its conditional pattern base, a sub-database, which consists of the set of prefix paths in the FP-tree co-occurring with the suffix pattern, then construct its conditional FP-tree and perform mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree. l Let us consider I5, which is the last item in L. I5 occurs in two branches of the FP-tree: ¡ (I2, I1, I5:1) ¡ (I2, I1, I3, I5:1) l I5 is a suffix, so its corresponding two prefix paths are ¡ (I2, I1:1) ¡ (I2, I1, I3:1) l Its conditional FP-tree contains only a single path: (I2:2, I1:2); I3 is removed because its support count of 1 is less than the minimum support count l The single path generates all the combinations of frequent patterns: ¡ {I2,I5:2} ¡ {I1,I5:2} 13 ¡ {I2, I1, I5:2} l For I4 exist 2 prefix path, which form the conditional pattern base: ¡ {{I2, I1:1},{I2:1}} l This generates a single-node conditional FP-tree: ¡ (I2:2) l The frequent pattern: {I2, I1:2} The following table shows the frequent pattern generated for each node: Item Conditional Pattern Base Conditional FP-tree Frequent Pattern Generated I5 {{I2, I1:1}, {I2, I1, I3:1}} (I2:2, I1:2) {I2, I5:2}, {I1, I5:2}, {I2, I1, I5:2} I4 {{I2, I1:2}, {I2:1}} (I2:2) {I2, I4:2} I3 {{I2, I1:2}, {I2:2}, {I1:2}} (I2:4, I1:2), (I1:2), (I2:4) {I2, I3:4}, {I1, I3:4}, {I2, I1, I3:2}, {I2, I1:4} I1 {{I2:4}} (I2:4) {I2, I1:4} The FP-growth method transforms the problem of finding long frequent patterns to searching for shorter ones recursively and then concatenating the suffix. It uses the least frequent items as a suffix, offering good selectivity. The method substantially reduces the search costs. T

he FP-growth algorithm: mine frequent itemsets using an FP-tree by pattern fragment growth.

Input: § D, a transaction database § min\_sup, the minimum support count threshold

Output: the complete set of frequent patterns.

Method:

(1) the FP-tree is constructed

(2) The FP-tree is mined by calling FP-growth(FP\_tree, null): procedure FP\_growth(Tree, α) if Tree contains a single path P then for each combination (denoted as β) of the nodes in the path P generate pattern βUα with support\_count = minimum support count of nodes in β; else for each ai in the header of Tree{ generate pattern β=aiUα with support\_count = ai .support\_count construct β’s conditional pattern base and then β’s conditional FP\_tree Treeβ ; if Treeβ != 0 then call FP\_growth(Treeβ , β); }

1. Define Post Pruning and pre pruning.

Ans: -

**Post-pruning**

Post-pruning is also known as backward pruning. In this, first Generate the decision tree and then remove non-significant branches. Post-pruning a decision tree implies that we begin by generating the (complete) tree and then adjust it with the aim of improving the classification accuracy on unseen instances. There are two principal methods of doing this. One method that is widely used begins by converting the tree to an equivalent set of rules. Another commonly used approach aims to retain the decision tree but to replace some of its subtrees by leaf nodes, thus converting a complete tree to a smaller pruned one which predicts the classification of unseen instances at least as accurately. There are various methods for the post pruning.

**Pre-pruning**

Pre-pruning is also called forward pruning or online-pruning. Pre-pruning prevent the generation of non-significant branches. Pre-pruning a decision tree involves using a ‘termination condition’ to decide when it is desirable to terminate some of the branches prematurely as the tree is generated. When constructing the tree some significant measures can be used to assess the goodness of a split. If partitioning the tuples at a node would result the split that falls below a prespecified threshold, then further partitioning of the given subset is halted otherwise it is expanded. High threshold result in oversimplified trees, whereas low threshold result in very little simplification. There are various approaches for the pre-pruning.

1. Define CLARA and CLARANS?

Ans: -

**CLARA**

CLARA Designed by Kaufman and Rousseeuw to handle large data sets, CLARA (Clustering Large Applications) relies on sampling [19]. Instead of finding representative objects for the entire data set, CLARA draws a sample of the data set, applies PAM on the sample, and finds the medoids of the sample. The point is that, if the sample is drawn in a sufficiently random way, the medoids of the sample would approximate the medoids of the entire data set. To come up with better approximations, CLARA draws multiple samples and gives the best clustering as the output. Here, for accuracy, the quality of a clustering is measured based on the average dissimilarity of all objects in the entire data set, and not only of those objects in the samples. Experiments reported in indicate that five samples of size 40 þ 2k give satisfactory results.

**CLARANS**

**CLARANS (Clustering Large Applications based on Randomized Search)** is efficient and effective and is the best practice for spatial data mining. CLARANS applies a strategy to search in a certain graph. A node in this graph, denoting it as , is represented by a set of objects, ,. Here, **k** is the predefined value to choose the k medoids; as a result, the nodes in the graph are a set of . If two nodes, , and  are neighbours, then . Each node in the  graph represents a set of medoids and the cluster related to it. As a result, a cost is related to each node; this cost is the total distance between any objects and the medoid represents its cluster. The cost differential of two neighbours can be calculated with the cost measure function introduced in the PAM algorithm.

1. Define Chameleon method?

Ans: -

Chameleon is a hierarchical clustering algorithm that overcomes the limitations of the existing models and the methods present in the data warehousing. This method operates on the sparse graph having nodes that represent the data items and edges represent the weights of the data items. The representation of it allows large data set to be created and operated on successfully. The method finds the clusters that are used in the data set using the two-phase algorithm. The first phase consists of the graph partitioning that allows the clustering of the data items into large number of sub-clusters. Second phases use an agglomerative hierarchical clustering algorithm to search for the clusters that are genuine and can be combined together with the sub-clusters that are produced.

1. Define Wave Cluster?

Ans: -

The WAVE clustering algorithm is a grid-based clustering algorithm. It depends on the relation between spatial dataset and multidimensional signals. The idea is that the cluster in a multidimensional spatial dataset turns out to be more distinguishable after a wavelet transformation, that is, after applying wavelets to the input data or the preprocessed input dataset. The dense part segmented by the sparse area in the transformed result represents clusters.

The characteristics of the WAVE cluster algorithm are as follows:

* Efficient for a large dataset
* Efficient for finding various shapes of clusters
* Insensitive to noise or outlier
* Insensitive with respect to the input order of a dataset
* Multiresolution, which is introduced by wavelet transforms
* Applicable to any numerical dataset

1. What is the use of Regression?

Ans: -

In simple words: The purpose of regression analysis is to predict an outcome based on a historical data. This historical data is understood using regression analysis and this understanding helps us build a model which to predict an outcome based on this regression model. It helps us predict and that is why it is called predictive analysis model.  
Example: If I want to predict what type of people buy a wine. I would find data on people who buy wine. Their age, height, financial status, etc. So, analysing this data I can build a model to predict whether a person would buy wine or not. So, regression analysis is used to predict the behaviour of a dependent variable (people who buy a wine) based on the behaviour of a few/large no. of independent variables (age, height, financial status).

1. What is lattice of cuboids?

Ans: -

It computes the aggregates over all the subsets of the dimension specified in the operation. The n-D cuboid which holds the lowest level of summarisation is called the base cuboid. The topmost 0-D cuboid which holds the highest level of summarisation is called apex cuboid. The lattice of cuboids forms a data cube. Data cube is a 3D range of values which is used to explain time sequence of image data.

1. What is apex cuboid?

Ans: -

In data warehousing literature, an n D base cube is called a base cuboid. The top most 0 D cuboid, which holds the highest level of summarization, is called the apex cuboid.

1. Give some data mining tools.

Ans: -

Rapid Miner, Weka, Orange, R, Knime, Rattle, Tanagra, XL Miner,

1. Discuss the requirements of clustering in data mining.

Ans: -

The main requirements that a clustering algorithm should satisfy are:

* scalability;
* dealing with different types of attributes;
* discovering clusters with arbitrary shape;
* minimal requirements for domain knowledge to determine input parameters;
* ability to deal with noise and outliers;
* insensitivity to order of input records;
* high dimensionality;
* interpretability and usability.

1. Explain data mining applications for Biomedical and DNA data analysis.

Ans: -

In recent times, we have seen a tremendous growth in the field of biology such as genomics, proteomics, functional Genomics and biomedical research. Biological data mining is a very important part of Bioinformatics. Following are the aspects in which data mining contributes for biological data analysis −

* Semantic integration of heterogeneous, distributed genomic and proteomic databases.
* Alignment, indexing, similarity search and comparative analysis multiple nucleotide sequences.
* Discovery of structural patterns and analysis of genetic networks and protein pathways.
* Association and path analysis.
* Visualization tools in genetic data analysis.

1. Give Various forms of visualizing the discovered patterns.

Ans: -

* Data mining result visualization:

Visualization of data mining results is the presentation of the results or knowledge obtained from data mining in visual forms. Such forms may include scatter plots and boxplots, as well as decision trees, association rules, clusters, outliers, and generalized rules

* Data mining process visualization:

This type of visualization presents the various processes of data mining in visual forms so that users can see how the data are extracted and from which database or data warehouse they are extracted, as well as how the selected data are cleaned, integrated, pre-processed, and mined.

* Interactive visual data mining:

In (interactive) visual data mining, visualization tools can be used in the data mining process to help users make smart data mining decisions. For example, the data distribution in a set of attributes can be displayed using coloured sectors

1. Give the name of proximity function in k-medoid.

Ans: -

Voronoi Iteration

1. Write support and confidence formula.

Ans: -

LET

T: (A set of) All transactions that customers make and are recorded in the stores system. (Since most of the customers use credit/debit cards, there is a unique number associated to their shopping list.)

Basket: A set of all items bought by a customer.

Item-set: A set of items that we are interested in.

Now, let’s assume that we analyzed customers’ transactions and realized that “many of them”, if they had *wine* in their baskets, they also have *cheese*! Why can’t we put *cheese* at the beginning of an aisle and *wine* at the end of the next one and put all tempting items that a customer with a bottle of *wine* may need a nudge to buy, in between!!

So, there has to be a way to evaluate the importance of a discovered rule. Here comes the support and confidence.

Suppose the rule we discovered is as follows

Wine -> Cheese (Support: 9% Confidence: 65%)

Support: is the percentage of transactions in T that contain both *wine* and *Cheese* together. (9% of all baskets had these 2 items together.)

**support(A→B)=P(A∪B)**

Confidence: is the percentage of transactions in T, containing *wine*, that also contain *Cheese*. In other words, the probability of having *Cheese*, given that *wine* is already in the basket. (65% of all those who bought *Wine*, also bought *Cheese*.)

**confidence(A→B)=P(B|A)**

1. Explain over fitting and under fitting in classification

Ans: -

**Overfitting Classification**

[Overfitting](https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/) refers to a model that models the training data too well.

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the model’s ability to generalize. Overfitting is more likely with nonparametric and nonlinear models that have more flexibility when learning a target function. As such, many nonparametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns. For example, decision trees are a nonparametric machine learning algorithm that is very flexible and is subject to overfitting training data. This problem can be addressed by pruning a tree after it has learned in order to remove some of the detail it has picked up.

**Underfitting in Classification**

Underfitting refers to a model that can neither model the training data nor generalize to new data. An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data. Underfitting is often not discussed as it is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms. Nevertheless, it does provide a good contrast to the problem of overfitting.

1. Explain confusion matrix.

Ans: -

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

The confusion matrix shows the ways in which your classification model  
is confused when it makes predictions.

It gives your insight not only into the errors being made by your classifier but more importantly the types of errors that are being made. It is this breakdown that overcomes the limitation of using classification accuracy alone.

1. How to calculate accuracy from confusion matrix.

Ans: -

Definition of the Terms:

Positive (P): Observation is positive (for example: is an apple).

Negative (N): Observation is not positive (for example: is not an apple).

True Positive (TP): Observation is positive, and is predicted to be positive.

False Negative (FN): Observation is positive, but is predicted negative.

True Negative (TN): Observation is negative, and is predicted to be negative.

False Positive (FP): Observation is negative, but is predicted positive.

Classification Rate/Accuracy:  
Classification Rate or Accuracy is given by the relation:

Accuracy = (TP+TN)/(TP+TN+FP+FN)

However, there are problems with accuracy. It assumes equal costs for both kinds of errors. A

99% accuracy can be excellent, good, mediocre, poor or terrible depending upon the problem.