USN | | | | | | | | | | |

Internal Test II – April 2018

Sub: Data Warehousing & Data Mining Sub Code: 16MCA442 Branch: MCA

Date: 17/04/18 Duration: 90 min's Max Marks: 50 Sem / Sec: A

Answer any FIVE FULL Questions from each part.

		MARKS	OBE CO	OBE RBT
Part I-1	What is anti monotone property of support? Explain Apriori Algorithm with example.	[10]	CO4	L2
2 (a)	Explain FP Growth algorithm with example.	[10]	CO4	L2
Part II-3	What is Rule based classifier? Explain Sequential Covering Algorithm.	[10]	CO3	L2
4 (a)	Explain the property of Rule Based classifier.	[5]	CO3	L2
(b)	What is nearest neighbor classifier? Explain.	[5]	CO3	L2
Part III-5(a)	Explain Hunt Algorithm for decision tree induction.	[5]	CO4	L2
(b)	What are node impurity measures? Explain with example.	[5]	CO3	L2
6	Explain the alternative method of item sets generation.	[10]	CO3	L2
Part IV-7(a)	Write a detailed note on Maximal frequent itemsets and Closed frequent itemsets.	[5]	CO4	L2
(b)	Explain in detail about various evaluation criteria for classification methods.	[5]	CO6	L3
8	Discuss in detail about various techniques for improving the accuracy of classification methods.	[10]	CO6	L3
Part V-9(a)	Describe Multiclass Problem in detail.	[5]	CO6	L2
(b)	Explain Ripper Algorithm.	[5]	CO5	L2
10	State Apriori principle for generating itemsets that are frequent. Construct itemset lattice for itemset I ={I1,I2,I3,I4} and list all the itemsets subsets.	[10	CO4	L2

Part I – 1Q :What is anti monotone property of support? Explain Apriori Algorithm with example.

Ans: Apriori principle:

- If an itemset is frequent, then all of its subsets must also be frequent
  - Apriori principle holds due to the following property of the support measure: $\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$
- Support of an itemset never exceeds the support of its subsets
- This is known as the anti-monotone property of support

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3 = 41$
With support-based pruning,
6 + 6 + 1 = 13

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

Apriori Algorithm
Method:

- Let k=1
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate by scanning the DB
  - Eliminate candidates that are infrequent, leaving only those that are frequent

Q2: Explain FP Growth algorithm with example.

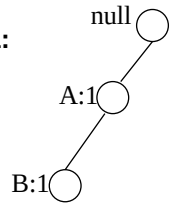
Ans: **FP-growth Algorithm**

- Use a compressed representation of the database using an FP-tree
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

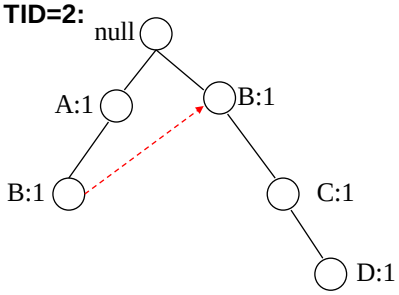
FP-tree construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



After reading TID=2:



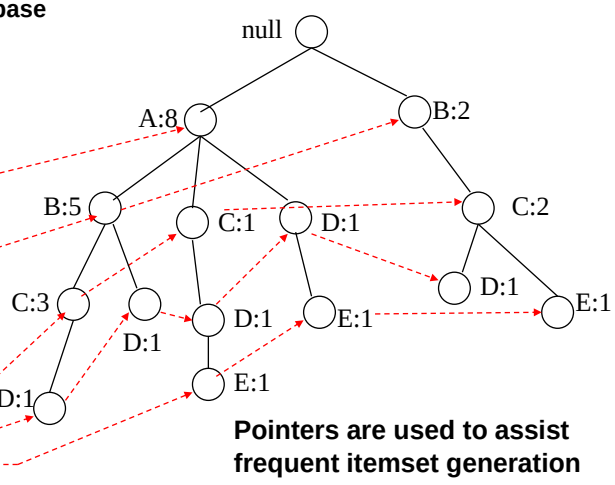
FP-Tree Construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Transaction Database

Header table

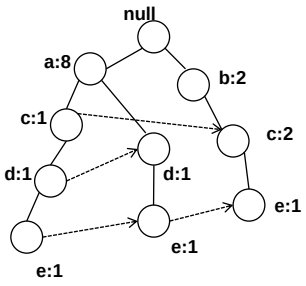
Item	Pointer
A	
B	
C	
D	
E	



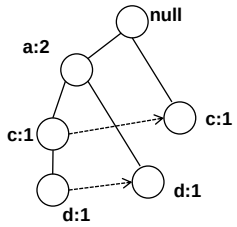
Pointers are used to assist frequent itemset generation

Frequent Itemset Generation in FP-Growth Algorithm

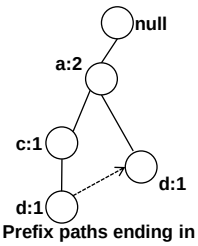
- Generates frequent itemsets from an FP-tree by exploring the tree in a bottom-up fashion.
- Algorithm looks for frequent itemsets ending in e first, followed by d,c,b and finally a.



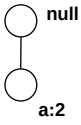
(a)Prefix Paths ending in e



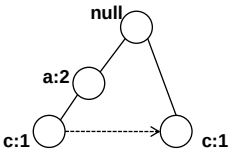
(b) Conditional FP-tree for e



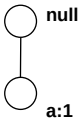
© Prefix paths ending in de



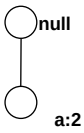
(d) Conditional FP-tree for de



(e) Prefix paths ending in ce



(f)Conditional FP-tree for ce



(g)Prefix paths ending in ae

- Used divide and conquer strategy to split the problem into smaller subproblems.
- Gather all the paths(prefix paths) containing node e.
- Count the support for e. Assume that the minimum support count is 2, {e} is declared a frequent itemset because its support count is 3.
- Solve the subproblems of finding frequent itemsets ending in de, ce, be, and ae.
- Convert the prefix paths into conditional FP-tree.
- Support counts along the prefix paths must be updated.
- The prefix paths are truncated by removing the nodes for e.
- Ignore the infrequent item from subsequent analysis.
- FP-growth uses the conditional FP-tree for e to solve the subproblems of finding frequent itemset ending in de,ce, and ae
- To find the frequent itemsets ending in de, gather the prefix from the conditional FP-tree for e.

- Obtain the support count for{d,e} by adding the frequency counts associated with node d .
- Support count is 2 so {d,e} is declared a frequent itemset.
- Construct the conditional FP-tree for de.
- Update the support count and remove the infrequent item c.
- Tree contains only one item a, whose support count is equal to minsup, extracts the frequent itemset {a,d,e}
- Generate frequent itemsets ending in ce.
- Only {c,e} is found to be frequent.
- Then generate next frequent item ending in ae which is {a,e}

Part II-Q3: What is Rule based classifier? Explain Sequential Covering Algorithm.

Ans: **Rule-Based Classifier**

- Classify records by using a collection of "if…then…" rules
    - Rule: (*Condition*) $\rightarrow y$
    - where
        - *Condition* is a conjunctions of attributes
        - *y* is the class label
    - *LHS*: rule antecedent or precondition
    - *RHS*: rule consequent
    - Examples of classification rules:
        - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
        - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ Evade=No

Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Application of Rule-Based Classifier

- A rule *r* covers an instance **x** if the attributes of the instance satisfy the condition of the rule

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird
The rule R3 covers the grizzly bear => Mammal

Direct Method: Sequential Covering

Extracts the rules one class at a time for data sets having more than two classes.
Criterion for choosing class depend on the factor such as class prevalence.

1. Start from an empty decision list, R.
2. Grow a rule using the Learn-One-Rule function
3. Add the new rule to the bottom of the decision list
4. Remove training records covered by the rule
5. Repeat Step (2) and (3) until stopping criterion is met

Sequential Covering Algorithm:

1:Let E be the training records and A be the set of attribute-value pairs, {(Aj,Vj)}.
2: Let Yo be an ordered set of classes {y1,y2,….yk}
3:Let R={ } be the initial rule list.
4: for each class y E Yo – {yk} do
5: while stopping condition is not met do
6: r ← Learn-One-Rule(E,A,y).
7: Remove training records from E that are covered by r.
8: Add r to the bottom of the rule list: R->RVr.
9: end while
10: end for
11: Insert the default rule, {}->yk, to the bottom of the rule list R

Q4a: Explain the property of Rule Based classifier.

Ans: **Characteristics of Rule-Based Classifier**

Mutually exclusive rules

- Classifier contains mutually exclusive rules if the rules are independent of each other
- Every record is covered by at most one rule

Exhaustive rules

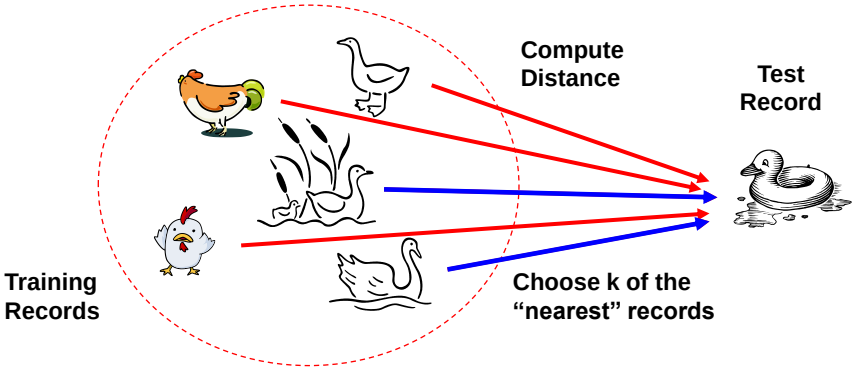
- Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
- Each record is covered by at least one rule

Q4b: What is nearest neighbor classifier? Explain.

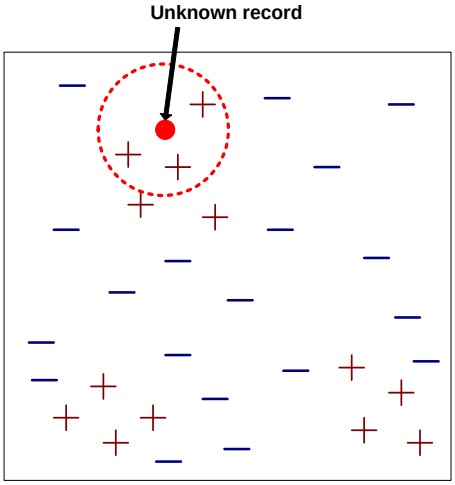
Ans: **Nearest Neighbor Classifiers:**

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

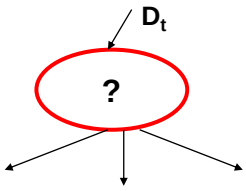
Part III- Q5 a: Explain Hunt Algorithm for decision tree induction.

Ans:

General Structure of Hunt's Algorithm

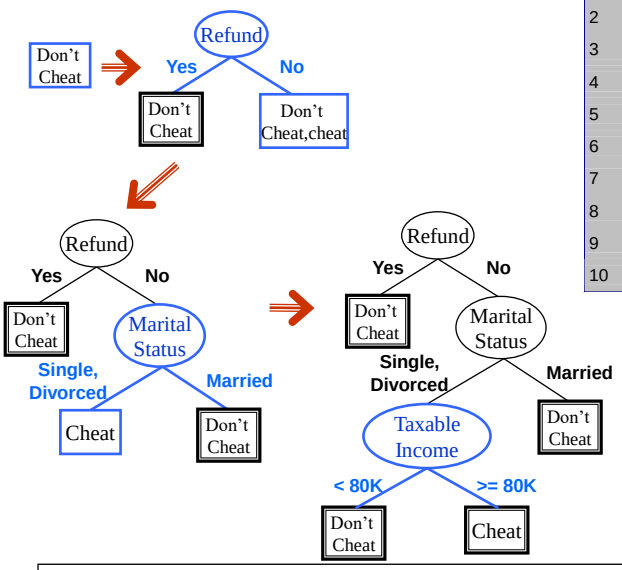
- Let D_t be the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Hunt's Algorithm

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Q 5b: What are node impurity measures? Explain with example.

Ans: **Measures of Node Impurity**

- Gini Index
- Entropy
- Misclassification error

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

C1	1
C2	5

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)² – (5/6)² = 0.278

C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)² – (4/6)² = 0.444

Splitting Based on GINI

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  - Measures homogeneity of a node.
    - Maximum (log n_c) when records are equally distributed among all classes implying least information
    - Minimum (0.0) when all records belong to one class, implying most information
  - Entropy based computations are similar to the GINI index computations

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

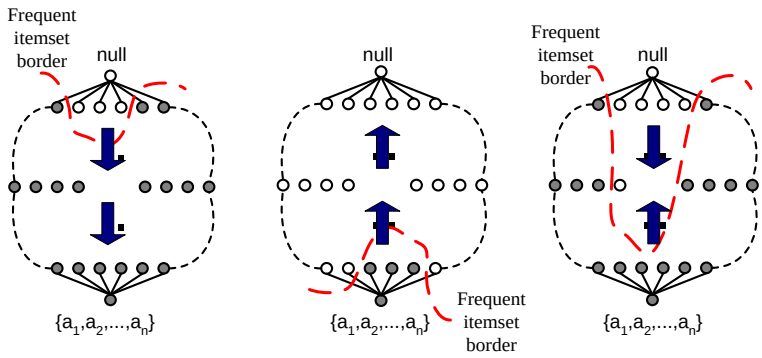
- Measures misclassification error made by a node.
  - Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Q 6: Explain the alternative method of item sets generation.

Ans:

Alternative Methods for Frequent Itemset Generation

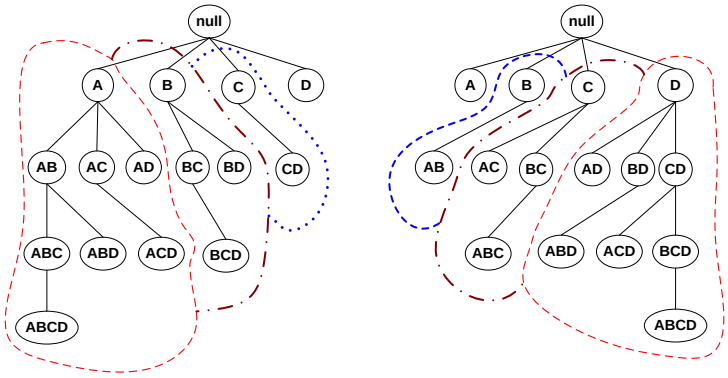
- Traversal of Itemset Lattice
  - General-to-specific vs Specific-to-general



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

Alternative Methods for Frequent Itemset Generation

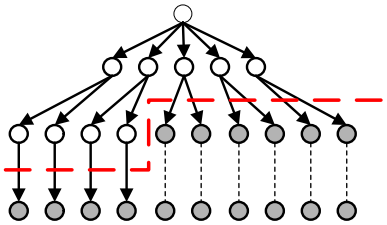
- Traversal of Itemset Lattice
  - Equivalent Classes



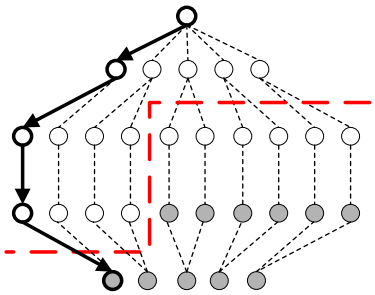
(a) Prefix tree (b) Suffix tree

Alternative Methods for Frequent Itemset Generation

- Traversal of Itemset Lattice
  - Breadth-first vs Depth-first



(a) Breadth first



(b) Depth first

Alternative Methods for Frequent Itemset Generation

- Representation of Database
  - horizontal vs vertical data layout

Horizontal Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

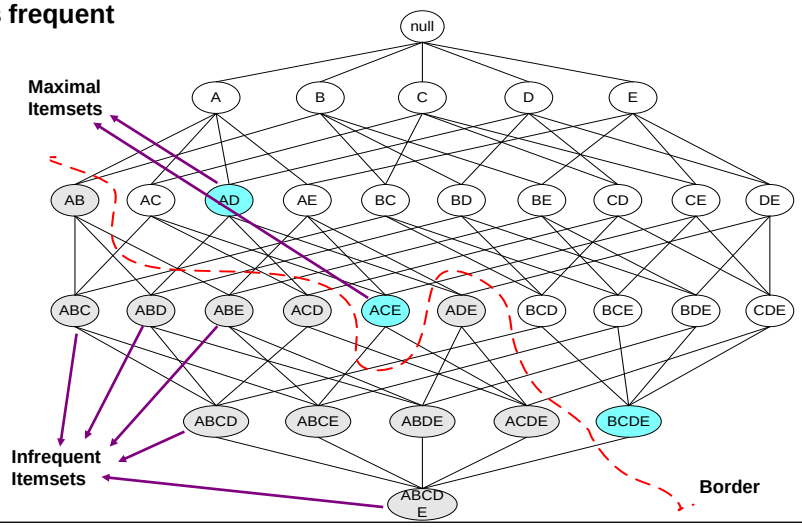
Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

Part IV- Q7 a: Write a detailed note on Maximal frequent itemsets and Closed frequent itemsets.
Ans:

Maximal Frequent Itemset

An itemset is maximal frequent if none of its immediate supersets is frequent



Closed Itemset

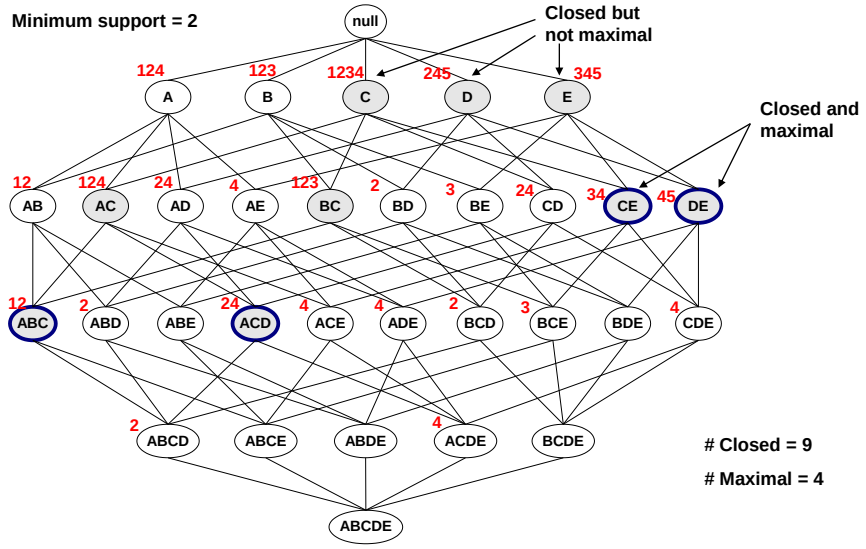
- An itemset is closed if none of its immediate supersets has the same support as the itemset

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

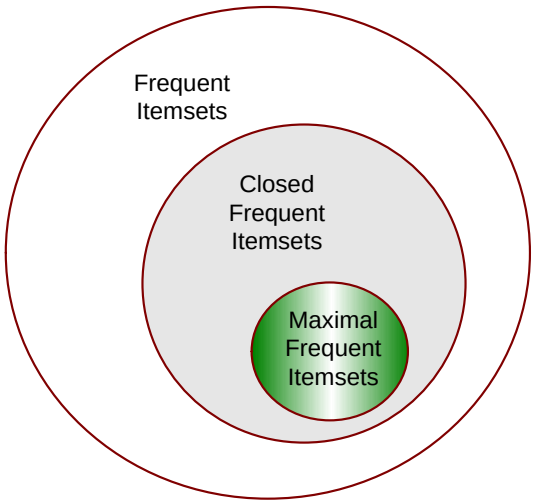
Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Maximal vs Closed Frequent Itemsets



Maximal vs Closed Itemsets



Q 7 b: Explain in detail about various evaluation criteria for classification methods.

Methods of Estimation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
  - A balance must be achieved
- Random sub sampling
  - Repeated holdout
  - take mean of the trials
- Cross validation or k- fold cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Repeat k times
  - Take mean for accuracy estimate

Ans:

- Leave-one-out:
  - One of the training sample is taken out for testing
  - Model is generated using the remaining training data
  - Result is coded as 1 or 0
  - Take average
  - Useful when the dataset is small
- Stratified sampling
  - oversampling vs undersampling
- Bootstrap
  - Sampling with replacement

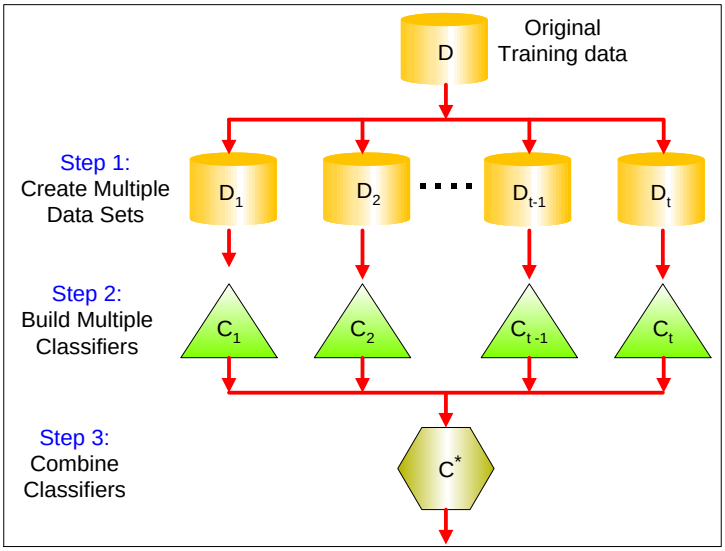
Other Evaluation Criteria for Classification Methods

- Speed
- Robustness
- Scalability
- Interpretability
- Goodness of the model
- Flexibility
- Time complexity

Q 8: Discuss in detail about various techniques for improving the accuracy of classification methods.

Ans:

General Idea



Bagging

- Sampling with replacement
- Each sample has the same size as the original data

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Some instances may appear several times while other may be omitted.
- Build classifier on each bootstrap sample

- Each sample has probability $(1 – 1/n)^n$ of being selected

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

• Example 4 is hard to classify

• Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Part V- Q 9 a: Describe Multiclass Problem in detail

Ans:

Multiclass Problem

- To divide data into more than two categories.
- Two approaches
- One- against-rest(1-r) approach
- One against one approach
- One- against-rest(1-r) approach:
- Decomposes the multiclass problem into K binary problems.
- For each class yi € Y, create a binary problem where all instances that belong to yi are considered positive examples, while the remaining instances are considered negative examples.
- Construct Binary classifier is to separate instances of class yi from the rest of the classes.

- One against one approach:
- Constructs K(K-1)/2 binary classifiers.
- Each classifier is used to distinguish between a pair of classes,(yi,yj).
- Instances not belonging to either yi or yj are ignored while constructing the binary classifier for (yi,yj).

- In both method, a test instance is classified by combining the predictions made by the binary classifiers.
- Voting scheme is used to combine the predictions.
- Class with highest number of votes is assigned to the test instance.

Q 9 b: Explain Ripper Algorithm.

Ans:

Direct Method: RIPPER

- Growing a rule:
  - Start from empty rule
  - Add conjuncts as long as they improve FOIL's information gain
  - Stop when rule no longer covers negative examples
  - Prune the rule immediately using incremental reduced error pruning
  - Measure for pruning: v = (p-n)/(p+n)
    - p: number of positive examples covered by the rule in the validation set
    - n: number of negative examples covered by the rule in the validation set
  - Pruning method: delete any final sequence of conditions that maximizes v

Direct Method: RIPPER

- Building a Rule Set:
  - Use sequential covering algorithm
    - Finds the best rule that covers the current set of positive examples
    - Eliminate both positive and negative examples covered by the rule
  - Each time a rule is added to the rule set, compute the new description length
    - stop adding new rules when the new description length is d bits longer than the smallest description length obtained so far

Q 10: State Apriori principle for generating itemsets that are frequent. Construct itemset lattice for itemset I ={I1,I2,I3,I4} and list all the itemsets subsets.

- λ Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
    - λ Apriori principle holds due to the following property of the support measure:
  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support