USN

Internal Assessment Test 3 – July 2021

Sub:	Data Mining and Data warehousing				Sub Code:	18CS641/17CS651	Branch:	ISE
Date:	30/07/2021	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VI A,B&C	OBE

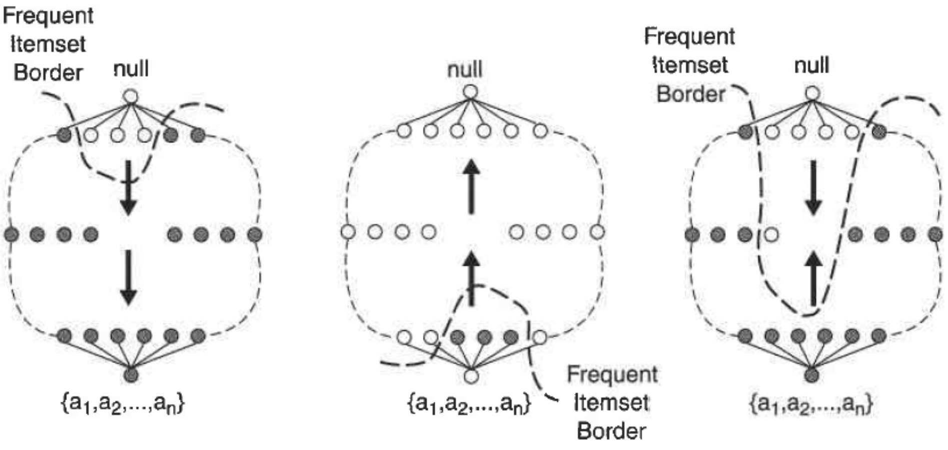
	<u>Answer any FIVE FULL Questions</u>	MARKS	CO	RBT
1	Discuss about various alternate methods for generating frequent itemsets with diagrams	[10]	CO3	L2

Traversal of Itemset Lattice: [3 marks explanation + 1 mark Diagram]

- A search for frequent itemsets can be conceptually viewed as a traversal on the itemset lattice shown in Figure 6.19.
- The search strategy employed by an algorithm dictates how the lattice structure is traversed during the frequent itemset generation process.

1. General-to-Specific versus Specific-to-General:

- The Apriori, algorithm uses a general-to-specific search strategy, where pairs of frequent (k- l)-itemsets are merged to obtain candidate k-itemsets.
- This general to-specific search strategy is effective, provided the maximum length of a frequent itemset is not too long.
- The configuration of frequent itemsets that works best with this strategy is shown in Figure 6.19(a), where the darker nodes represent infrequent itemsets.
- Alternatively, a specific to-general search strategy looks for more specific frequent itemsets first, before finding the more general frequent itemsets.
- **This strategy is useful to discover maximal frequent itemsets in dense transactions,** where the frequent itemset border is located near the bottom of the lattice, as shown in Figure 6.19(b).
- The Apriori, principle can be applied to prune all subsets of maximal frequent itemsets. Specifically, if a candidate k-itemset is maximal frequent, we do not have to examine any of its subsets of size k - l.
- **However, if the candidate k-itemset is infrequent, we need to check all of its k - 1 subsets in the next iteration.**
- Another approach is to combine both general-to-specific and specific-to-general search strategies.
- This bidirectional approach requires more space to store the candidate itemsets, but it can help to rapidly identify the frequent itemset border, given the configuration shown in Figure 6.19(c).

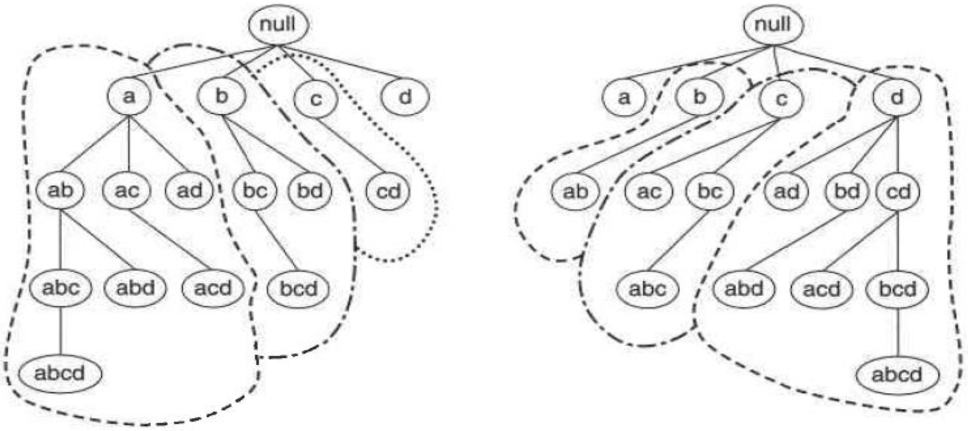


(a) General-to-specific (b) Specific-to-general (c) Bidirectional

Figure 6.19. General-to-specific, specific-to-general, and bidirectional search.

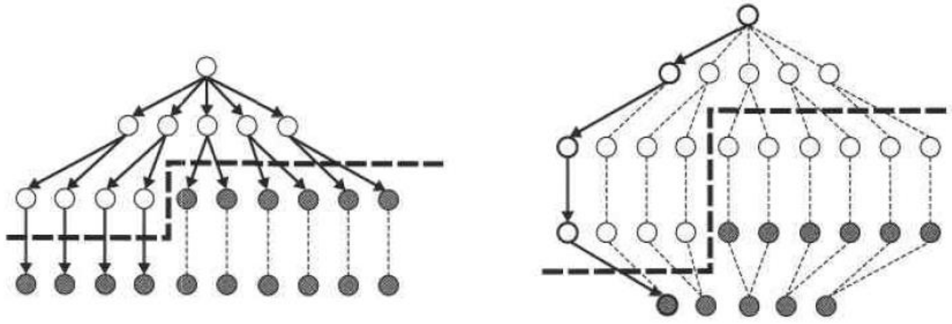
Equivalence Classes: **[2 marks explanation + 1 mark Diagram]**

- Another way to envision the traversal is to first partition the lattice into disjoint groups of nodes (or equivalence classes).
- A frequent itemset generation algorithm searches for frequent itemsets within a particular equivalence class first before moving to another equivalence class.
- As an example, the level-wise strategy used in the Apriori algorithm can be considered to be partitioning the lattice on the basis of itemset sizes; i.e., the algorithm discovers all frequent l-itemsets first before proceeding to larger-sized itemsets.
- **Equivalence classes can also be defined according to the prefix or suffix labels of an itemset.**
- In this case, two itemsets belong to the same equivalence class if they share a common prefix or suffix of length k.
- In the prefix-based approach, the algorithm can search for frequent itemsets starting with the prefix a before looking for those starting with prefixes b, c and so on.
- Both prefix-based and suffix-based equivalence classes can be demonstrated using the tree-like structure shown in Figure 6.20.



(a) Prefix tree. (b) Suffix tree.

Figure 6.20. Equivalence classes based on the prefix and suffix labels of itemsets.



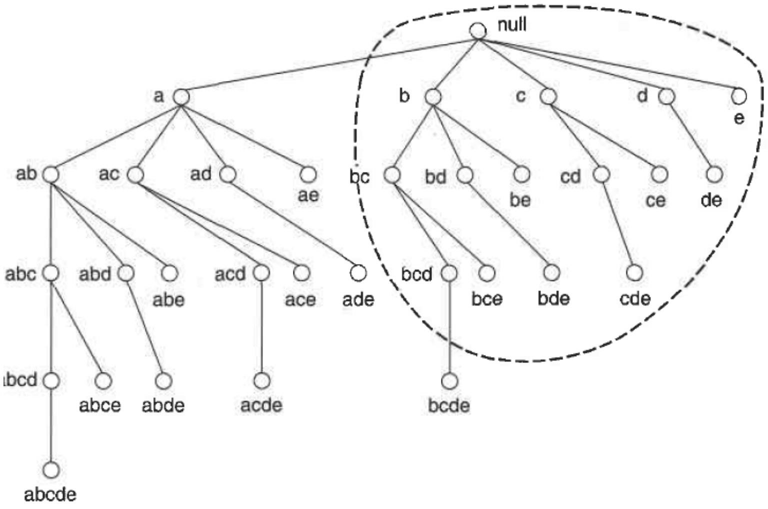
(a) Breadth first

(b) Depth first

Figure 6.21. Breadth-first and depth-first traversals.

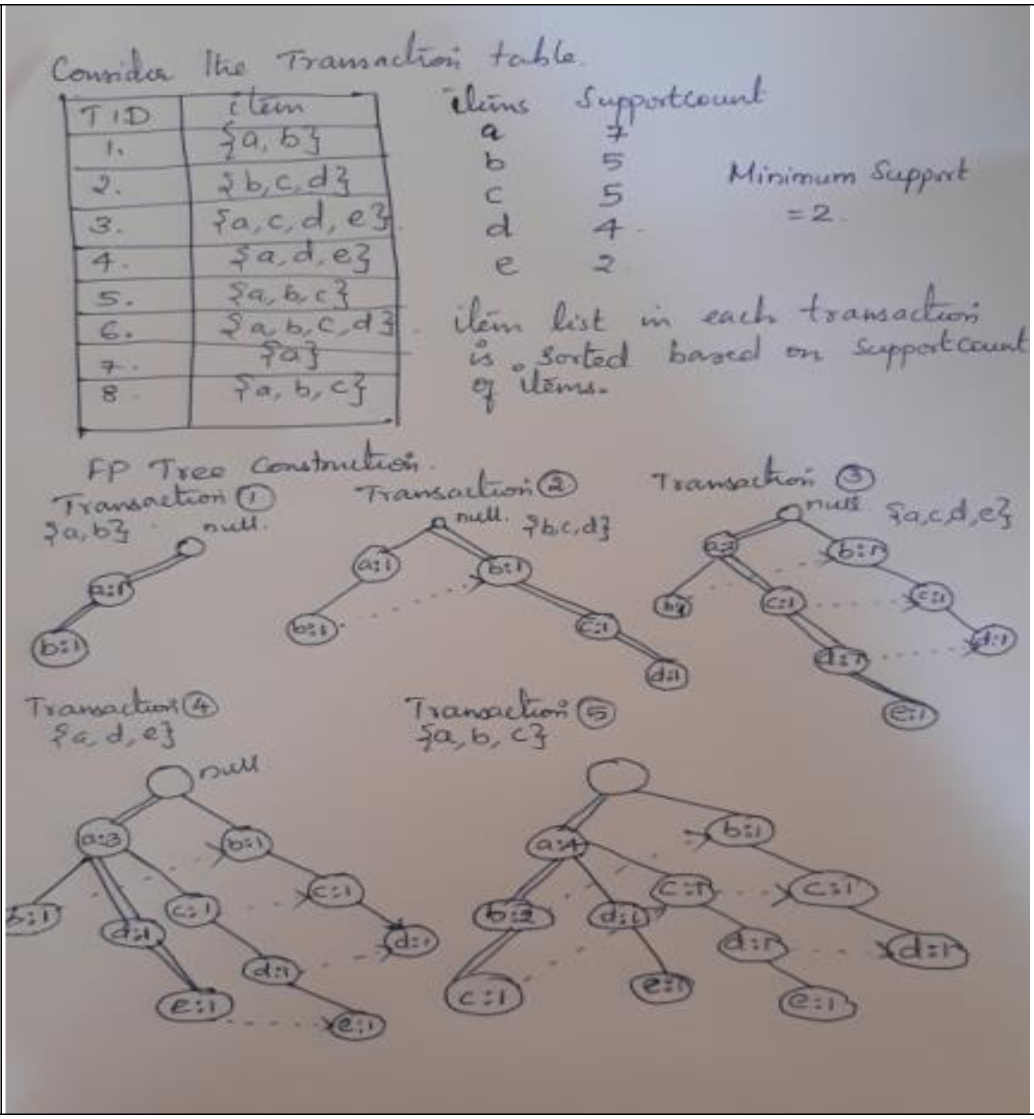
Breadth-First versus Depth-First: [2 marks explanation + 1 mark Diagram]

- The Apriori, algorithm traverses the lattice in a breadth-first manner as shown in Figure 6.21(a).
- It first discovers all the frequent 1-itemsets, followed by the frequent 2-itemsets, and so on, until no new frequent itemsets are generated.
- The itemset lattice can also be traversed in a depth-first manner, as shown in Figures 6.21(b) and 6.22.
- The algorithm can start from, say, node a, in Figure 6.22, and count its support to determine whether it is frequent.
- If so, the algorithm progressively expands the next level of nodes, i.e., ab, abc, and so on, until an infrequent node is reached, say, abcd.
- It then backtracks to another branch, say, abce, and continues the search from there.
- The deprth-first approach is often used by algorithms designed to find maximal frequent itemsets.
- This approach allows the frequent itemset border to be detected more quickly than using a breadth-first approach.
- Once a maximal frequent itemset is found, substantial pruning can be performed on its subsets.



- **Figure 6.22.** Generating candidate itemsets using the depth-first approach.
- A maximal frequent itemset is defined as a frequent itemset for which none of its immediate supersets are frequent.
- For example, if the node bcde shown in Figure 6.22 is maximal frequent, then

<table>
<tr>
<td></td>
<td>the algorithm does not have to visit the subtrees rooted at bd,, be, c, d, and e because they will not contain any maximal frequent itemsets.
• However, if abc is maximal frequent, only the nodes such as ac and bc are not maximal frequent (but the subtrees of ac and bc may still contain maximal frequent itemsets).
• The depth-first approach also allows a different kind of pruning based on the support of itemsets.
• For example, suppose the support for {a,b,c} is identical to the support for {a, b}. The subtrees rooted at abd and abe can be skipped because they are guaranteed not to have any maximal frequent itemsets.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a)</td>
<td>Consider the following transaction data set. Construct the FP trees by showing the tress separately after reading each transaction. Find the Frequent Itemset using FP growth algorithm.
<table>
<tr><th>TID</th><th>ITEM</th></tr>
<tr><td>1</td><td>{a, b}</td></tr>
<tr><td>2</td><td>{b, c, d}</td></tr>
<tr><td>3</td><td>{a, c, d, e}</td></tr>
<tr><td>4</td><td>{a, d, e}</td></tr>
<tr><td>5</td><td>{a, b, c}</td></tr>
<tr><td>6</td><td>{a, b, c, d}</td></tr>
<tr><td>7</td><td>{a}</td></tr>
<tr><td>8</td><td>{a, b, c}</td></tr>
</table>
</td>
<td>[8]</td>
<td>CO3</td>
<td>L3</td>
</tr>
</table>



Consider the Transaction table.

T ID	item
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}

items	Supportcount
a	7
b	5
c	5
d	4
e	2

Minimum Support = 2.

item list in each transaction is sorted based on supportcount of items.

FP Tree Construction.

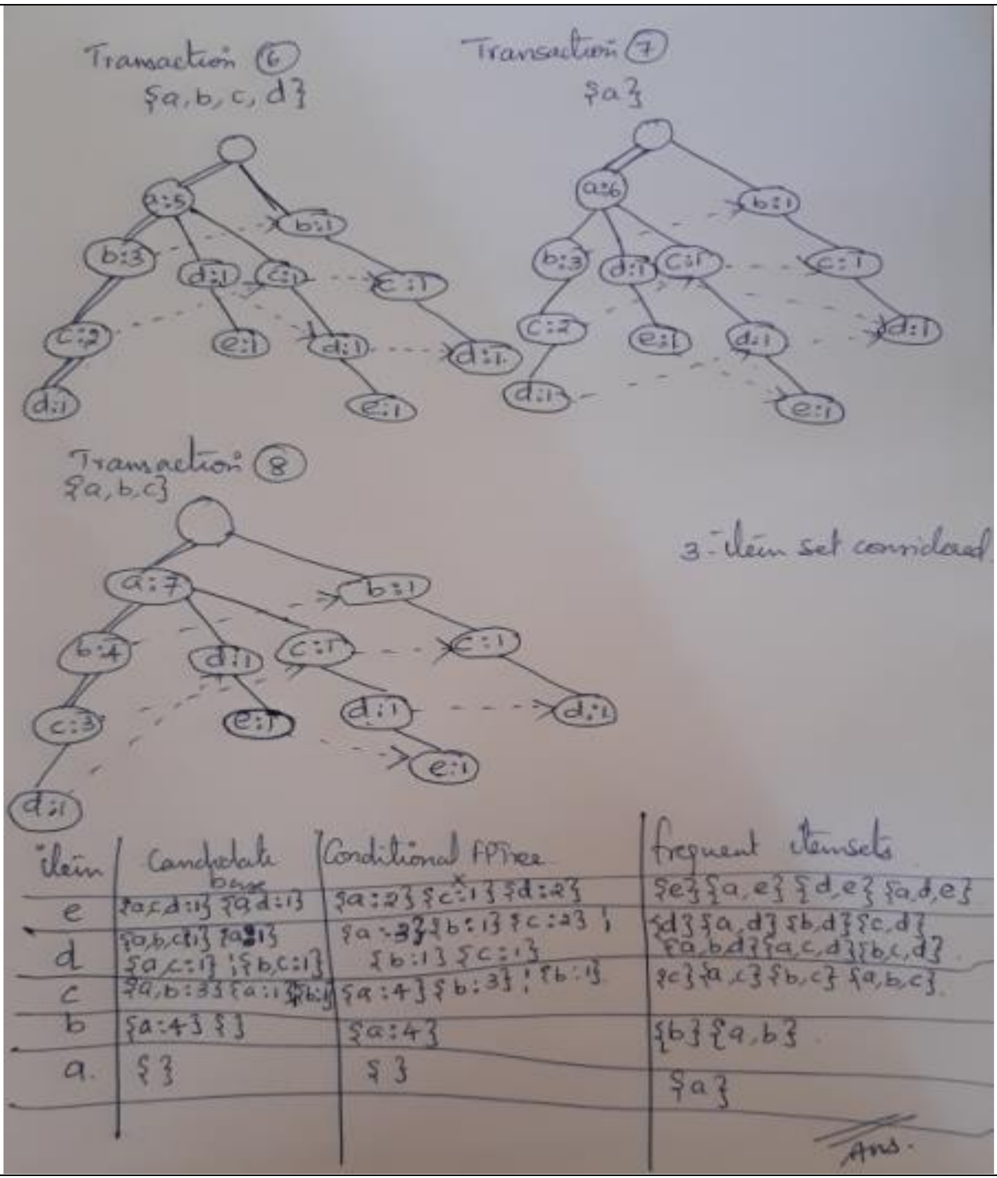
Transaction ①
{a, b}

Transaction ②
{b, c, d}

Transaction ③
{a, c, d, e}

Transaction ④
{a, d, e}

Transaction ⑤
{a, b, c}



item	Candidate base	Conditional FPTree	frequent itemsets
e	{a,c,d:1} {a,d:1}	{a:2} {c:1} {d:2}	{e} {a,e} {d,e} {a,d,e}
d	{a,b,c:1} {a:1} {a,c:1} ; {b,c:1}	{a:3} {b:1} {c:2} ; {b:1} {c:1}	{d} {a,d} {b,d} {c,d} {a,b,d} {a,c,d} {b,c,d}
c	{a,b:3} {a:1} {b:1}	{a:4} {b:3} ; {b:1}	{c} {a,c} {b,c} {a,b,c}
b	{a:4} { }	{a:4}	{b} {a,b}
a	{ }	{ }	{a}

Ans.

2b) Explain Contingency table with example. [2] CO3 L2

An objective measure is usually computed based on the frequency counts **tabulated in a contingency table.** **[1 Mark]**

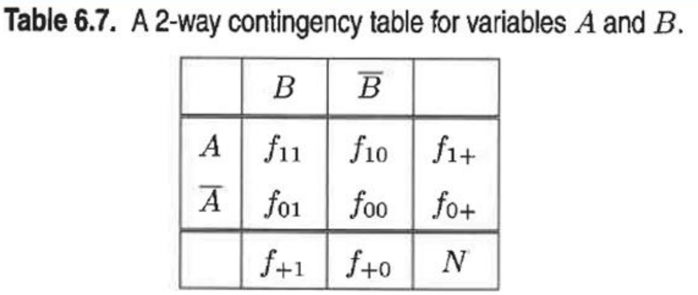


Table 6.7. A 2-way contingency table for variables A and B.

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

[1 mark] Table and Explanation

3a) Explain how to build Decision tree using Hunt's algorithm with example. [7] CO4 L2

Algorithm [3 marks]

Hunt's Algorithm

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets. Let D_t be the set of training records that are associated with node t and $y = \{y_1, y_2, \ldots, y_c\}$ be the class labels. The following is a recursive definition of Hunt's algorithm.

Step 1: If all the records in D_t belong to the same class y_t, then t is a leaf node labeled as y_t.

Step 2: If D_t contains records that belong to more than one class, an **attribute test condition** is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Figure 4.6. Training set for predicting borrowers who will default on loan payments.

Construction of the tree: [2 marks]

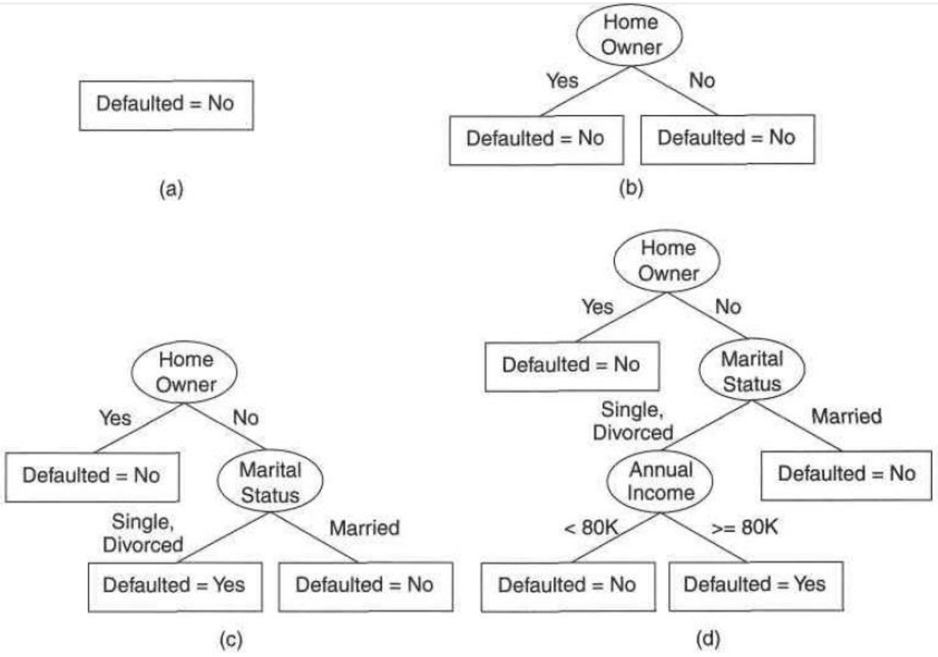


Figure 4.7. Hunt's algorithm for inducing decision trees.

Explanation: [2 marks]

- **The tree, however, needs to be refined since the root node contains records from both classes. The records are subsequently divided into smaller subsets based on the outcomes of the *Home Owner* test condition,** as shown in Figure 4.7(b).
- The justification for choosing this attribute test condition will be discussed later.
- For now, we will assume that this is the best criterion for splitting the data at

this point.

- Hunt's algorithm is then applied recursively to each child of the root node.
- From the training set given in Figure 4.6, notice that all borrowers who are home owners successfully repaid their loans.
- The left child of the root is therefore a leaf node labeled Defaulted = No (see Figure 4.7(b)).
- For the right child, we need to continue applying the recursive step of Hunt's algorithm until all the records belong to the same class. The trees resulting from each recursive step are shown in Figures 4.7(c) and (d).

3b) Consider the training examples shown in the table below for a binary classification problem. What is the entropy of this collection of training examples with respect to the positive class? [3] CO4 L3

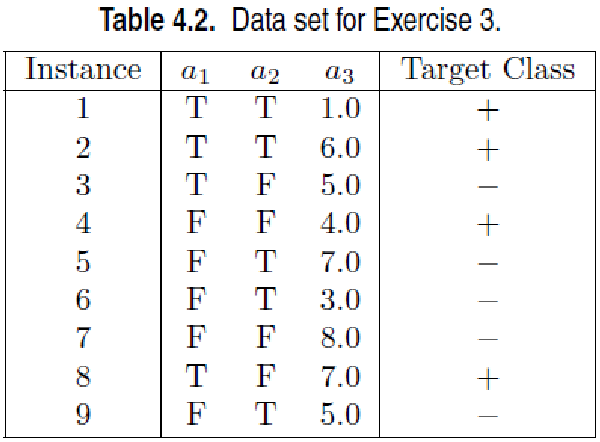


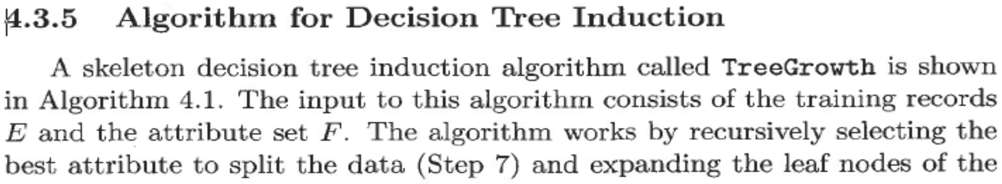
Table 4.2. Data set for Exercise 3.

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

There are four positive examples and five negative examples. Thus,
P(+) = 4/9 and P(−) = 5/9. **[1 mark]**
The entropy of the training examples is
−4/9 log2(4/9) − 5/9 log2(5/9) = 0.9911. **[2 marks]**

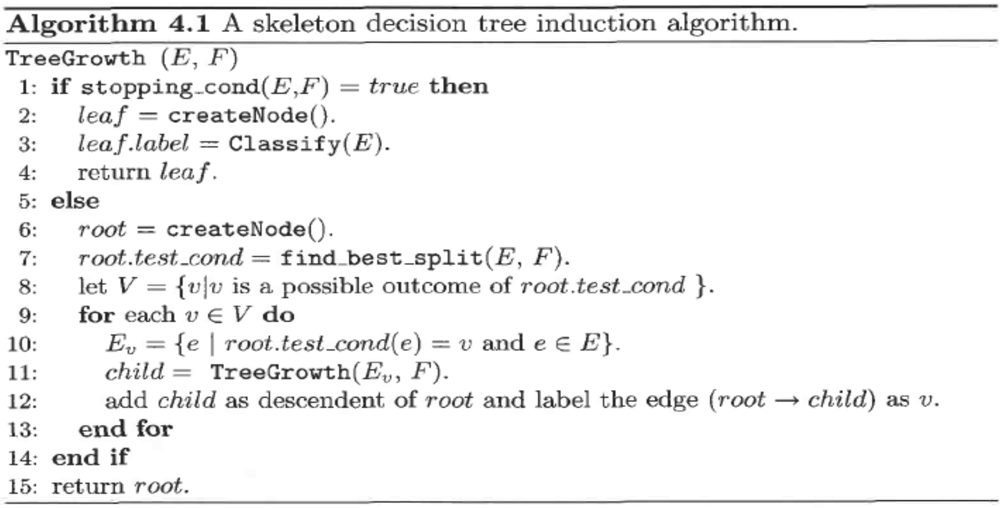
4a) Explain in detail about Decision tree induction algorithm with example dataset. [8] CO4 L2

Algorithm [4 marks]



4.3.5 Algorithm for Decision Tree Induction

A skeleton decision tree induction algorithm called TreeGrowth is shown in Algorithm 4.1. The input to this algorithm consists of the training records E and the attribute set F. The algorithm works by recursively selecting the best attribute to split the data (Step 7) and expanding the leaf nodes of the



Algorithm 4.1 A skeleton decision tree induction algorithm.

```
TreeGrowth (E, F)
 1: if stopping_cond(E,F) = true then
 2:     leaf = createNode().
 3:     leaf.label = Classify(E).
 4:     return leaf.
 5: else
 6:     root = createNode().
 7:     root.test_cond = find_best_split(E, F).
 8:     let V = {v|v is a possible outcome of root.test_cond }.
 9:     for each v ∈ V do
10:         E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:         child = TreeGrowth(E_v, F).
12:         add child as descendent of root and label the edge (root → child) as v.
13:     end for
14: end if
15: return root.
```

Example Construction of Decision Tree with dataset[4 marks]

4b) What is Baysian Belief Network? How does it differ from Naive based classifier? [2] CO4 L2

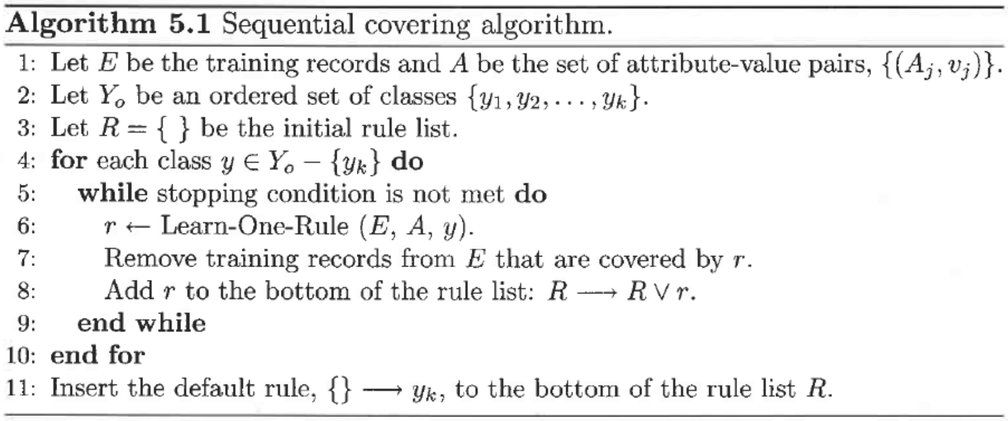
- **A** Bayesian belief network (BBN), or simply, Bayesian network, provides a graphical representation of the probabilistic relationships among a set of random variables.

There are two key elements of a Bayesian network: [1mark]

1. A directed acyclic graph (dag) encoding the dependence relationships among a set of variables.
2. A probability table associating each node to its immediate parent nodes

- **Differences: [1mark]**
- provides a graphical representation of the probabilistic relationships among a set of random variables
- Naive Bayes classifiers may seem too rigid, especially for classification problems in which the attributes are somewhat correlated.

5a) Explain Rule based classifier sequential algorithm with illustration [8] CO4 L2

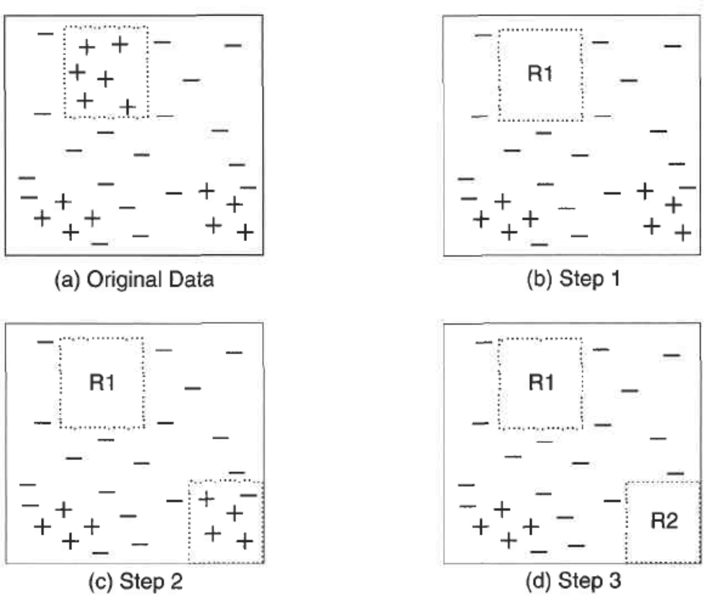


Algorithm 5.1 Sequential covering algorithm.

```
Let $E$ be the training records and $A$ be the set of attribute-value pairs, $\{(A_j, v_j)\}$.
Let $Y_o$ be an ordered set of classes $\{y_1, y_2, \ldots, y_k\}$.
Let $R = \{\ \}$ be the initial rule list.
for each class $y \in Y_o - \{y_k\}$ do
    while stopping condition is not met do
        $r \leftarrow$ Learn-One-Rule $(E, A, y)$.
        Remove training records from $E$ that are covered by $r$.
        Add $r$ to the bottom of the rule list: $R \longrightarrow R \vee r$.
    end while
end for
Insert the default rule, $\{\} \longrightarrow y_k$, to the bottom of the rule list $R$.
```

Algorithm +explanation[3+2 marks]

Explanation of Diagram 2[3 marks]



(a) Original Data (b) Step 1

(c) Step 2 (d) Step 3

Figure 5.2. An example of the sequential covering algorithm.

5b) Given the data set compute the confidence and accuracy for the rule [2] CO4 L3

Refund = yes --> No

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

Confidence= 3/10=30**% [1 Mark]**

Accuracy=3/3=100% **[1 Mark]**

6a)	Explain various methods for evaluating the performance of the classifiers Hold Out **Method [1.5 marks]** Random Sampling**[1.5 marks]** Cross validation**[1.5 marks]** Bootstrap method**[1.5 marks]**	[6]	CO4	L2

→ To do this class labels of the test records must be known.

· Methods for evaluating the performance of a classifier

Hold out Method.

→ Original data with labeled examples is partitioned into 2 disjoint sets called the training and the test sets.

→ classification Model is then induced from the training set and its performance is evaluated on the test set.

→ Proportion : 50-50 or $2/3 - 1/3$.

→ Accuracy can be estimated based on the accuracy of the induced model on the test set.

Limitations:

① → Fewer labeled examples are available. Other records are held for testing.

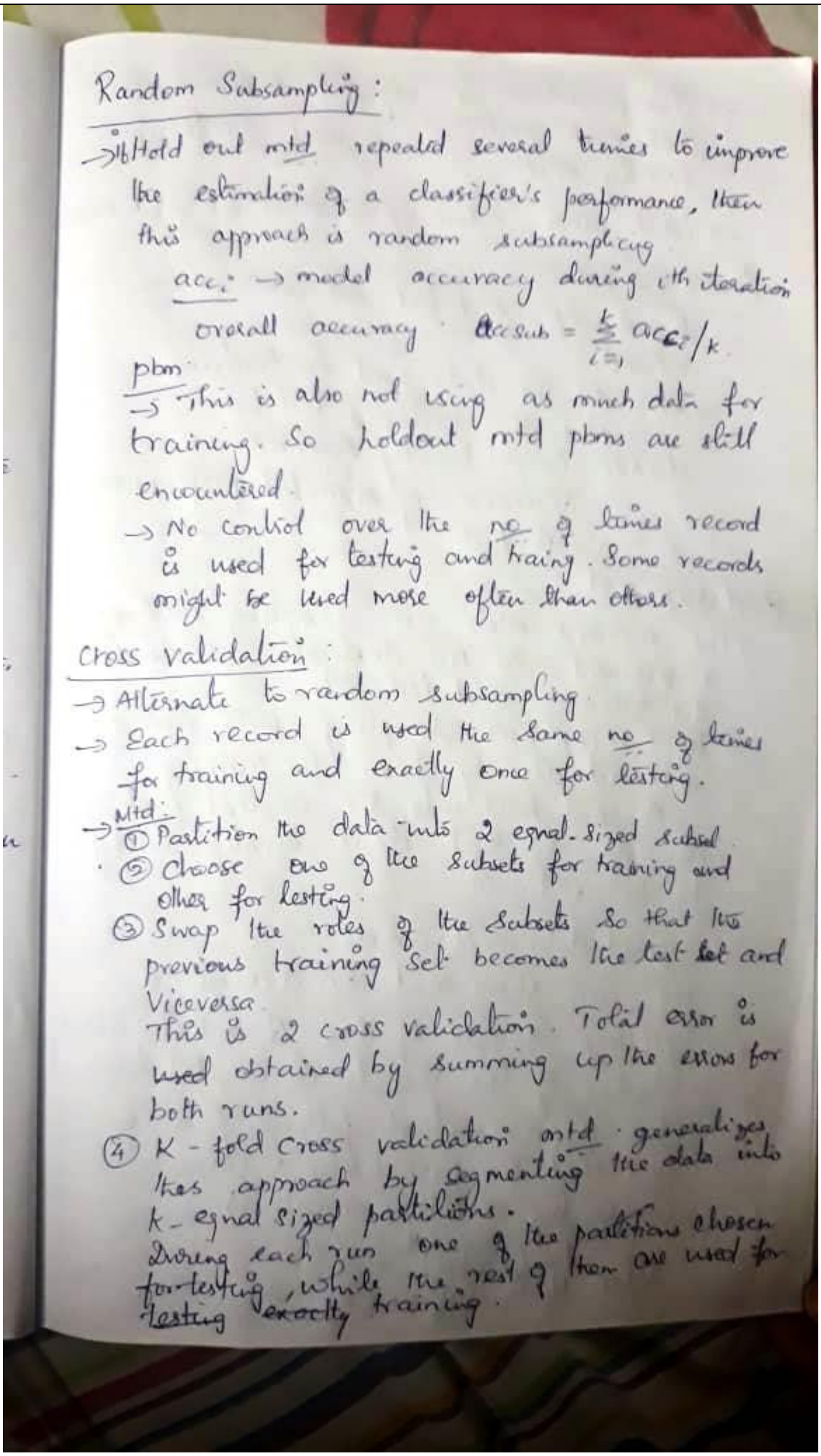
→ Induced Model may not be as good as when all the records are used for training.

② → Model highly dependent on the Composition of the training and test sets.

→ Smaller the training set size, the larger the variance of the model.

→ If training set is too large, accuracy from smaller test ~~test~~ set is less reliable.

③. Training and test sets are dependent. They are subset of original data. a class may be over represented in one subset and will be under represented in the other, and vice versa.



Random Subsampling :

→ If Hold out mtd repeated several times to improve the estimation of a classifier's performance, then this approach is random subsampling

acc_i → model accuracy during ith iteration

overall accuracy $acc_{sub} = \sum_{i=1}^{k} acc_i / k$

pbm

→ This is also not using as much data for training. So holdout mtd pbms are still encountered.

→ No control over the no of times record is used for testing and traing. Some records might be used more often than others.

Cross validation :

→ Alternate to random subsampling

→ Each record is used the same no of times for training and exactly once for testing.

→ Mtd:

① Partition the data into 2 equal-sized subset

② Choose one of the subsets for training and other for testing.

③ Swap the roles of the subsets so that the previous training set becomes the test set and viceversa.

This is 2 cross validation. Total error is used obtained by summing up the errors for both runs.

④ K-fold cross validation mtd generalizes this approach by segmenting the data into k-equal sized partitions.

During each run one of the partitions chosen for for testing, while the rest of them are used for ~~testing~~ ~~exactly~~ training.

→ This procedure is repeated k times so that each partition is used for testing exactly once.

→ Total error = Sum up the errors for all k runs

Leave one out:

→ $k = N$. i.e, each test set contains one record.

→ Adv: All the data used for training.
Mutually exclusive test test

disAdv: Computationally expensive.
Variance of estimated performance is high.

Bootstrap mtd:

→ The previous mtds use sampling without replacement. So No duplicate records in the training and test sets.

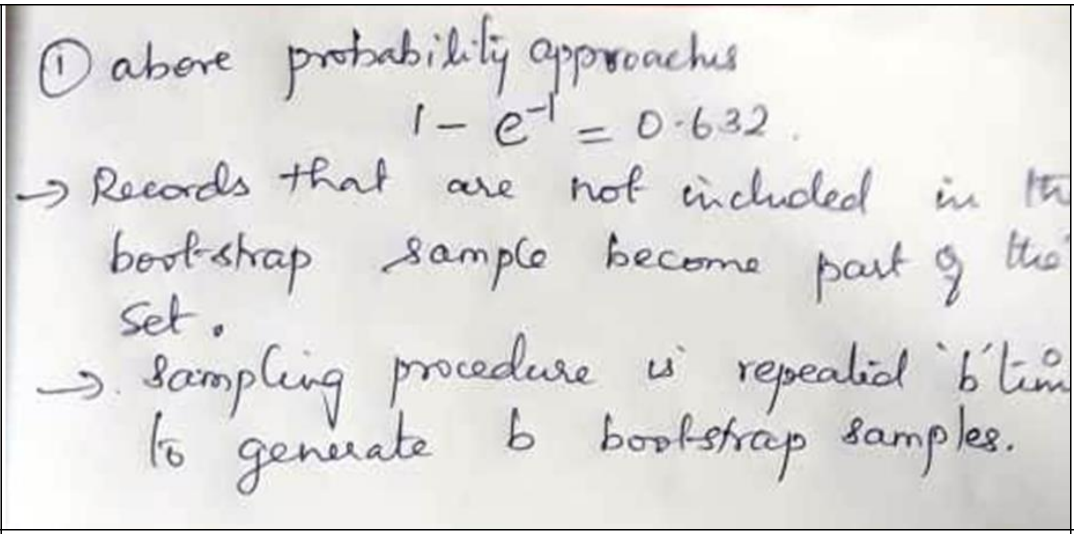
→ Boot strap uses sampling with replacement
→ record already chosen for training is put back into the original pool of records so that it is equally likely to be redrawn.

→ Original data → N records.
We can show that on average a bootstrap sample of size N contains about 63.2% of the records in original data.

→ This approximation follows from the fact that the probability a record chosen by Bootstrap sample is

$$1-(1-1/N)^N \longrightarrow ①$$

When N is sufficiently large then the



① above probability approaches
$1 - e^{-1} = 0.632$.

→ Records that are not included in th
bootstrap sample become part of the
set.

→ Sampling procedure is repeated 'b' tim
to generate b bootstrap samples.

6b) Given the training set, Classify the test record given below using Naive Bayes classifier. [4] CO4 L3

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	no	Yes	?

A: attributes
M: mammals
N: non-mammals

[1 mark for each step]

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{5}{7} \times \frac{5}{7} = 0.3748$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{6}{13} \times \frac{9}{13} = 0.0189$$

$$P(A|M)P(M) = 0.3748 \times \frac{7}{20} = 0.13118$$

$$P(A|N)P(N) = 0.0189 \times \frac{13}{20} = 0.012285$$

P(A|M)P(M) > P(A|N)P(N)
=> Mammals