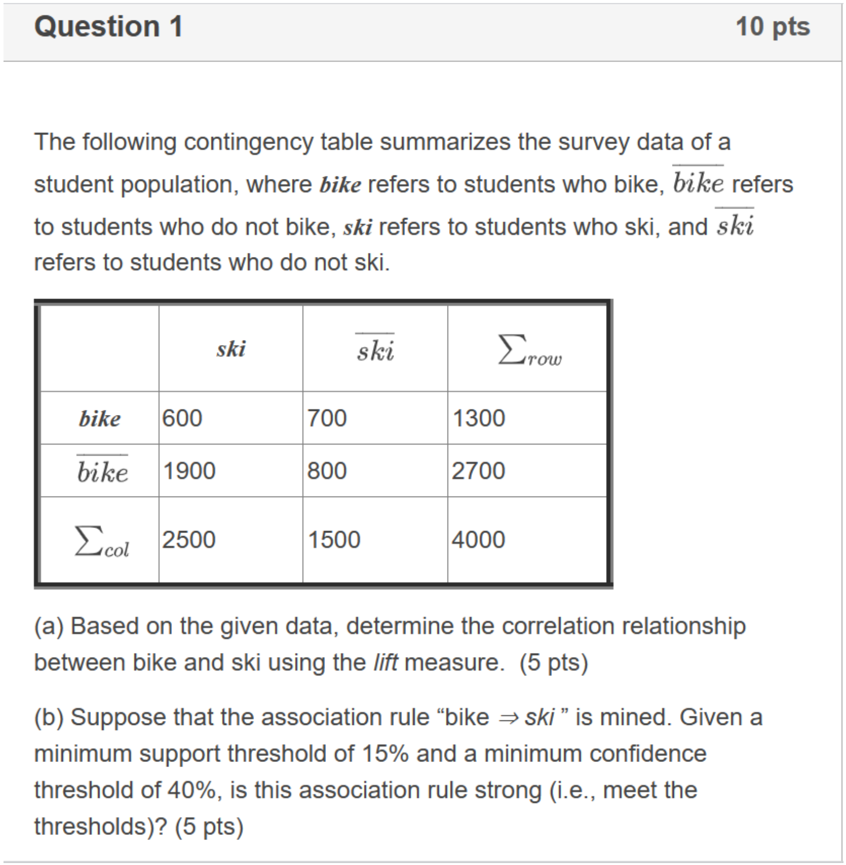


Question 1 **10 pts**

The following contingency table summarizes the survey data of a student population, where $\boldsymbol{bike}$ refers to students who bike, $\overline{bike}$ refers to students who do not bike, $\boldsymbol{ski}$ refers to students who ski, and $\overline{ski}$ refers to students who do not ski.

	$\boldsymbol{ski}$	$\overline{ski}$	$\sum_{row}$
$\boldsymbol{bike}$	600	700	1300
$\overline{bike}$	1900	800	2700
$\sum_{col}$	2500	1500	4000

(a) Based on the given data, determine the correlation relationship between bike and ski using the *lift* measure. (5 pts)

(b) Suppose that the association rule "bike ⇒ *ski* " is mined. Given a minimum support threshold of 15% and a minimum confidence threshold of 40%, is this association rule strong (i.e., meet the thresholds)? (5 pts)

Answer:

(a) lift(A, B) = P(A U B)/P(A)P(B)

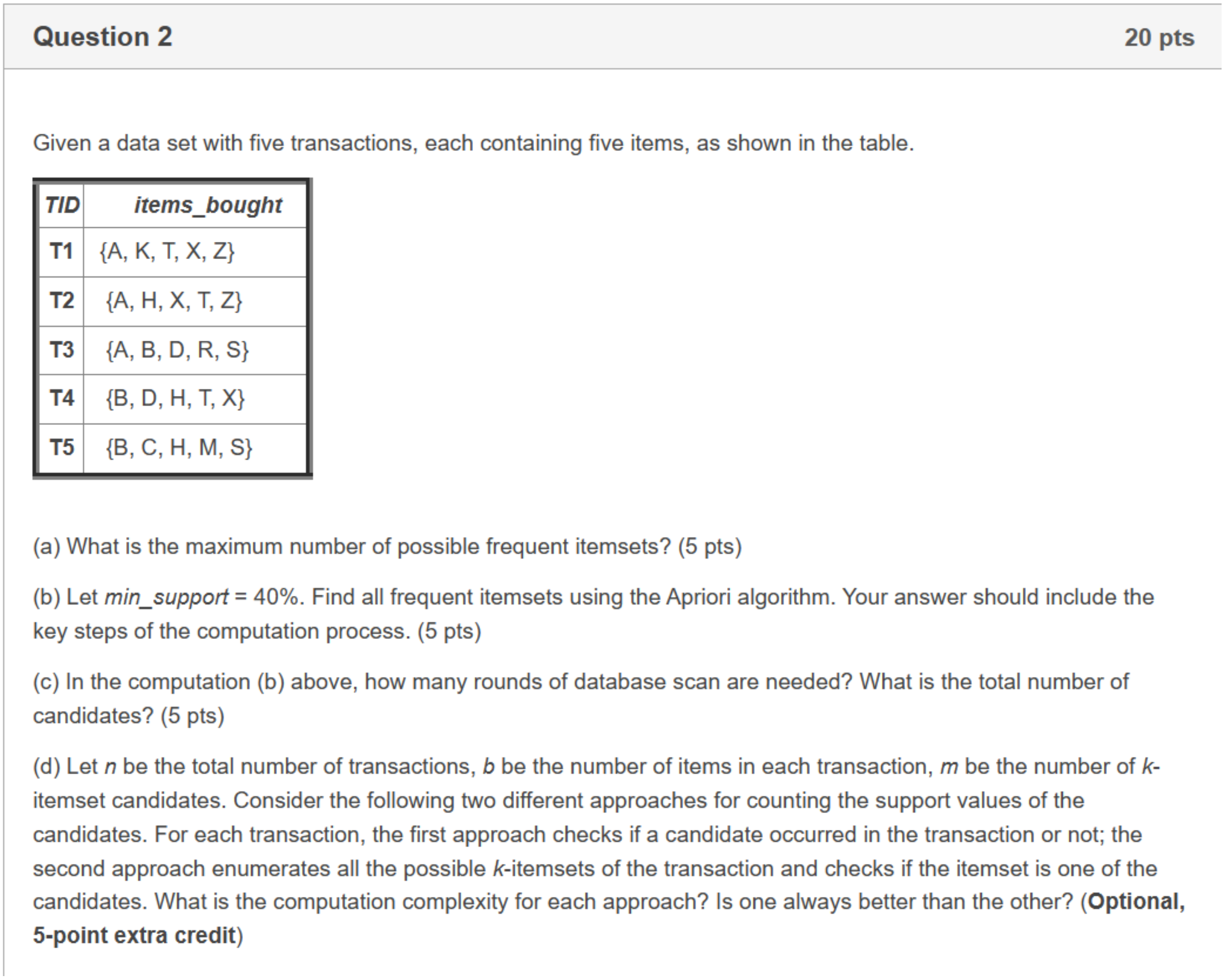
Here, A: bike, B: ski
lift = P(bike U ski)/P(bike)P(ski)

P(bike U ski) = 600/4000 = 0.15

P(bike) = 1300/4000 = 0.325
P(ski) = 2500/4000 = 0.625
lift = 0.15/(0.325 * 0.625) = 0.738
Since lift < 1, bike and ski are negatively correlated.

(b) support(bike => ski) = P(bike U ski) = 600/4000 = 0.15 = 15%
confidence(bike => ski) = P(bike | ski) = P(bike U ski)/P(bike)

= (600/4000)/(1300/4000) = 600/1300
= 0.46 = 46%
Since support(15%) >= minimum support threshold(15%) and confidence(46%) >= minimum confidence threshold(40%), the association rule "bike => ski" is strong.



Question 2 **20 pts**

Given a data set with five transactions, each containing five items, as shown in the table.

TID	*items_bought*
T1	{A, K, T, X, Z}
T2	{A, H, X, T, Z}
T3	{A, B, D, R, S}
T4	{B, D, H, T, X}
T5	{B, C, H, M, S}

(a) What is the maximum number of possible frequent itemsets? (5 pts)

(b) Let *min_support* = 40%. Find all frequent itemsets using the Apriori algorithm. Your answer should include the key steps of the computation process. (5 pts)

(c) In the computation (b) above, how many rounds of database scan are needed? What is the total number of candidates? (5 pts)

(d) Let *n* be the total number of transactions, *b* be the number of items in each transaction, *m* be the number of *k*-itemset candidates. Consider the following two different approaches for counting the support values of the candidates. For each transaction, the first approach checks if a candidate occurred in the transaction or not; the second approach enumerates all the possible *k*-itemsets of the transaction and checks if the itemset is one of the candidates. What is the computation complexity for each approach? Is one always better than the other? (**Optional, 5-point extra credit**)

Answer:

Solution a)

From the transactions, we observe that there are 12 unique items. And, the maximum number of items in any transaction is 5. So, the maximum number of possible frequent itemsets is calculated as the following sum,

12C1 + 12C2 + 12C3 + 12C4 + 12C5 = 12+66+220+495+792 = 1585

Solution b)

Step 1: Generate frequent 1-itemsets

Count the occurrences of each item in the dataset:

A: 3/5 = 60%

B: 3/5 = 60%

C: 1/5 = 20%

D: 2/5 = 40%

H: 3/5 = 60%

K: 1/5 = 20%

M: 1/5 = 20%

R: 1/5 = 20%

S: 2/5 = 40%

T: 3/5 = 60%

X: 3/5 = 60%

Z: 2/5 = 40%

Select the frequent 1-itemsets that meet the minimum support threshold:

{A}

{B}

{D}

{H}

{S}

{T}

{X}

{Z}

Step 2: Generate frequent 2-itemsets

Join frequent 1-itemsets to generate candidate 2-itemsets:

{A, B}

{A, D}

{A, H}

{A, S}

{A, T}

{A, X}

{A, Z}

{B, D}

{B, H}

{B, S}

{B, T}

{B, X}

{B, Z}

{D, H}

{D, S}

{D, T}

{D, X}

{D, Z}

{H, S}

{H, T}

{H, X}

{H, Z}

{S, T}

{S, X}

{S, Z}

{T, X}

{T, Z}

{X, Z}

Count the support of each candidate 2-itemset:

{A, B}: 1/5 = 20%

{A, D}: 1/5 = 20%

{A, H}: 1/5 = 20%

{A, S}: 1/5 = 20%

{A, T}: 2/5 = 40%

{A, X}: 2/5 = 40%

{A, Z}: 2/5 = 40%

{B, D}: 2/5 = 40%

{B, H}: 2/5 = 40%

{B, S}: 2/5 = 40%

{B, T}: 1/5 = 20%

{B, X}: 1/5 = 20%

{B, Z}: 0/5=   0%

{D, H}: 1/5 = 20%

{D, S}: 1/5 = 20%

{D, T}: 1/5 = 20%

{D, X}: 1/5 = 20%

{D, Z} : 0/5 = 0%

{H, S}: 1/5 = 20%

{H, T}: 2/5 = 40%

{H, X}: 2/5 = 40%

{H, Z}: 1/5 = 20%

{S, T}: 0/5 = 0%

{S, X}: 0/5 = 0%

{S, Z}: 0/5 =0%

{T, X}: 3/5 = 60%

{T, Z}: 2/5 = 40%

{X, Z}: 2/5 = 40%

Select the frequent 2-itemsets that meet the minimum support threshold:

{A, T}

{A, X}

{A, Z}

{B, D}

{B, H}

{B, S}

{H, T}

{H, X}

{T, X}

{T, Z}

{X, Z}

Step 3 : Join frequent 2-itemsets to generate candidate 3-itemsets:

{A,T,X}

{A,T,Z}

{A,X,Z}

{A,T, H}

{A,H,X}

{B,D,H}

{B,D,S}

{B,H,T}

{B,H,X}

{B,H,S}

{H,T,X}

{T,X,Z}

{H, X, Z}

{H,T,Z}

Count the support of each candidate 3-itemset:

{A,T,X} : 2/5 = 40%

{A,T,Z} : 2/5 = 40%

{A,X,Z} : 2/5 = 40%

{A, T, H}: 1/5 = 20%

{A,H,X}: 1/5 = 20%

{B,D,H} : 1/5 = 20%

{B,D,S} : 1/5 = 20%

{B,H,S} : 1/5 = 20%

{B,H,T}: 1/5 = 20%

{B,H,X} :1/5 = 20%

{H,T,X} : 2/5 = 40%

{T,X,Z} : 2/5 = 40%

{H,T,Z}: 1/5 = 20%

Select the frequent 3-itemsets that meet the minimum support threshold:

{A,T,X}

{A,T,Z}

{A,X,Z}

{H,T,X}

{T,X,Z}

Step 4: Join frequent 3-itemsets to generate candidate 4-itemsets:

{A, T, X, Z}

{A,H,T,X}

{H,T,X, Z}

Count the support of each candidate 4-itemset:

{A, T, X, Z} => 2/5 => 40%

{A,H,T,X} => 1/ 5 => 20%

{H,T,X, Z} => 1 / 5=> 20%

Only {A, T, X, Z} => 2/4 => 50% meet the minimum support requirement.

Now, frequent 5-itemsets, can not be created. Hence we stop here.

Solution c) To find the number of scans of the database we have to go through the Apriori Algorithm

For every iteration to find frequent item set, we join the frequent k-1 itemsets and then we scan the database to count the support for each item.

We first scan the database once to generate frequent 1-itemsets.

Then we created the frequent 2 itemset from the first and scanned the database to know the support and the same way we created frequent 3 itemset using the 2 frequent itemset

We required 4 rounds of database scan since we stopped on frequent 4-itemsets.

Total number of candidates are= Candidates in frequent 1-itemsets + frequent 2 -itemsets + frequent 3-itemsets + frequent 4-itemsets

Total number of candidates= 12+28+14+3

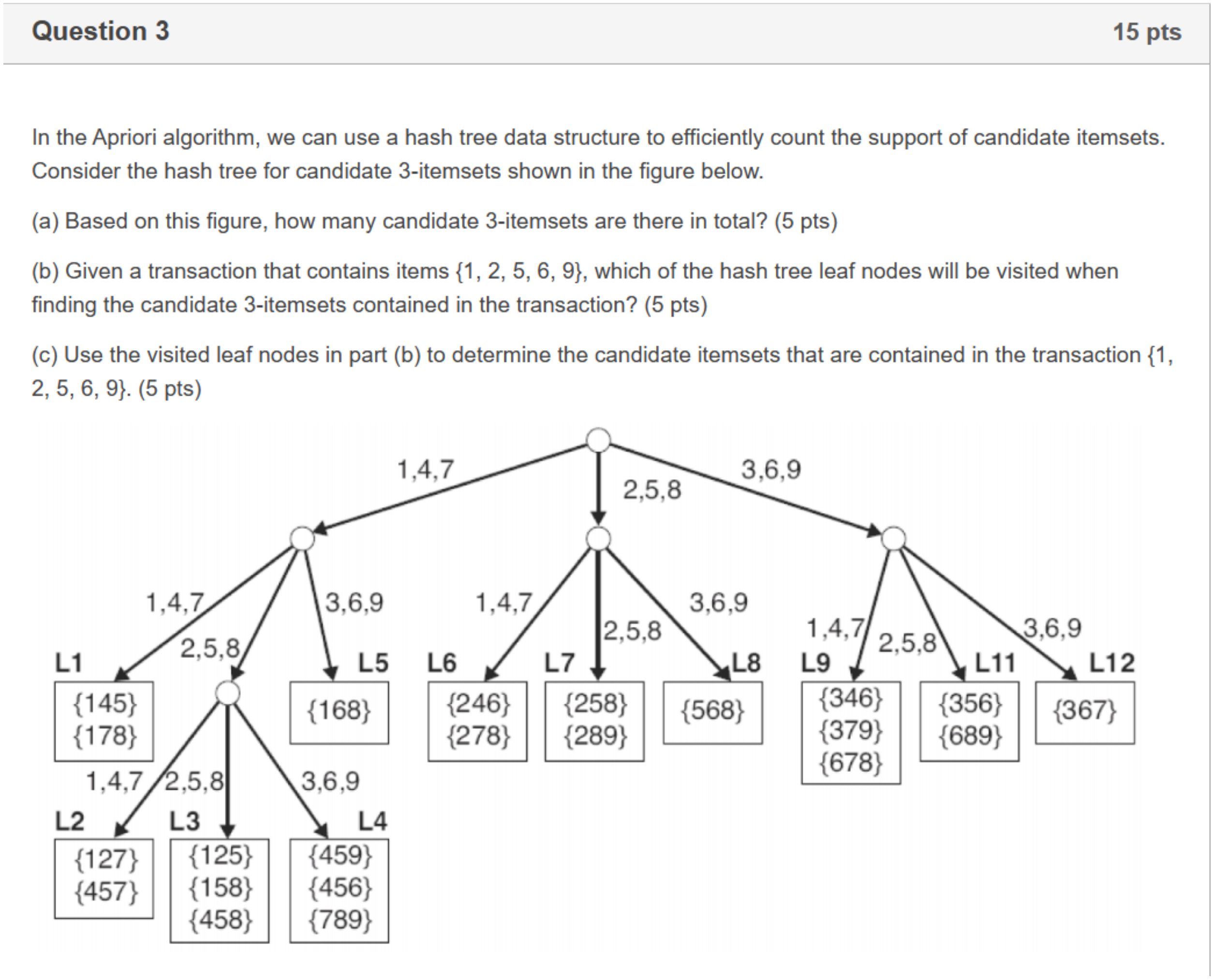
Total number of candidates = 57

Solution d)

The first approach will have a computational complexity of O(n*m). This is because, for each of the n transactions, we check m candidates.

The second approach will have a higher complexity. Specifically, this approach has a computational complexity of O(n * 2^b). This is because for each transaction, we are potentially generating all subsets of the items in the transaction, which is 2^b for b items, and then we check if each of these is a candidate.

It depends on the specific circumstances. If the number of candidates m is significantly less than 2^b, then the first approach would be more efficient. Suppose almost all subsets of items are candidates (i.e., m is close to 2^b). In that case, the second approach may be more efficient because generating all subsets would be unavoidable, and checking if a subset is a candidate can be done efficiently with a proper data structure such as a hash set.



Question 3 15 pts

In the Apriori algorithm, we can use a hash tree data structure to efficiently count the support of candidate itemsets. Consider the hash tree for candidate 3-itemsets shown in the figure below.

(a) Based on this figure, how many candidate 3-itemsets are there in total? (5 pts)

(b) Given a transaction that contains items {1, 2, 5, 6, 9}, which of the hash tree leaf nodes will be visited when finding the candidate 3-itemsets contained in the transaction? (5 pts)

(c) Use the visited leaf nodes in part (b) to determine the candidate itemsets that are contained in the transaction {1, 2, 5, 6, 9}. (5 pts)

Solution a) From the figure, the total number of candidate 3-itemsets is the total number of 3-itemsets in the leaf nodes (L1 to L12) of the hash tree i.e. 22.

So, all the number of 3-itemsets in the leaf nodes we get,

Total=L1 + L2 + L3+ L4 + L5 + L6 + L7 + L8 + L9 + L10 + L11 + L12

Total= 2+2+3+3+1+2+2+1+3+2+1 = 22

Solution b) Let us consider each of the 3-itemset we can form using the transaction {1,2,5,6,9}.

{1, 2, 5}, {1, 2, 6}, {1, 2, 9}, {1, 5, 6}, {1, 5, 9}, {1, 6, 9}, {2, 5, 6}, {2, 5, 9}, {2, 6, 9}, {5, 6, 9}.

We get a total of 10, 3-itemsets. For each 3-itemset, we traverse through the hash tree and mark the leaf nodes that are visited.

We will denote the traversal using this way-

We will start from the root.

left- traversing down leftmost of the tree path

right -traversing down rightmost of the tree path

middle, traversing down middle of the tree path

{1,2,5} ->Root to left, then middle and then middle again. We reach L3

{1,2,6} ->Root to left, then middle and then right. We reach L4

{1,2,9} -> Root to left , then middle and then right , we reach L4.

{1,5,6} -> Root to left, then middle and then right we reach L4

{1,5,9} -> Root to left , then middle and then right we reach L4

{1,6,9} -> Root to left, then right and we reach L5

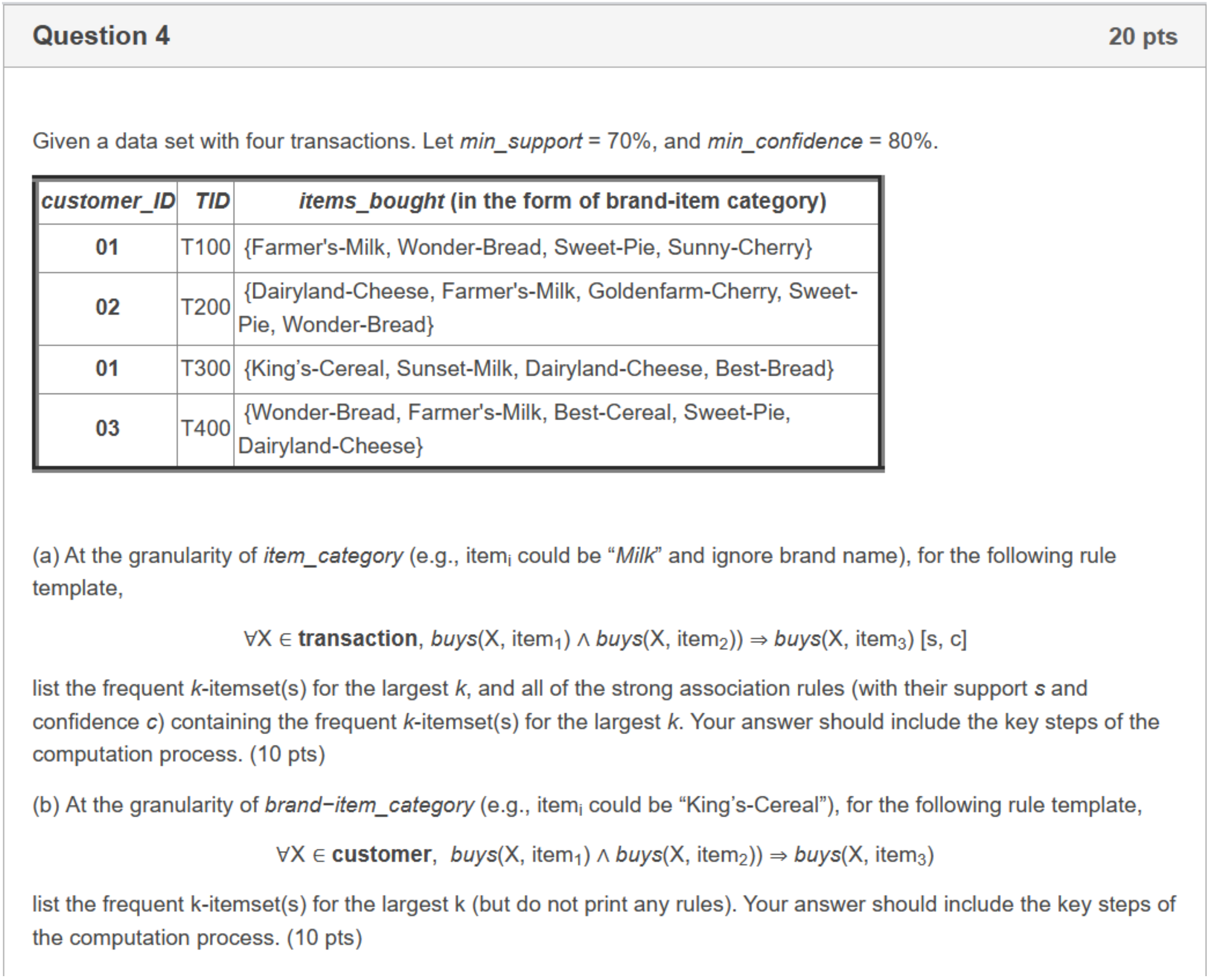
{2,5,6} -> Root to middle, and then middle and we reach L7

{2,5,9} -> Root to middle , again middle and we reach L7

{2,6,9} -> Root to middle, then right and we reach. L8

{5,6,9} ->Root to middle and then right we reach L8

Solution c) The candidate itemsets present in the transactions can be determined by whether the 3 itemsets of the transaction were present in the leaf nodes where we reached traversing through the tree. The visited nodes are L3, L4, L5, L7, and L8 and the only 3-itemsets of the transaction that is present in the visited nodes is {1,2,5}. So {1,2,5} is the answer.



Question 4 **20 pts**

Given a data set with four transactions. Let *min_support* = 70%, and *min_confidence* = 80%.

customer_ID	*TID*	*items_bought* (in the form of brand-item category)
01	T100	{Farmer's-Milk, Wonder-Bread, Sweet-Pie, Sunny-Cherry}
02	T200	{Dairyland-Cheese, Farmer's-Milk, Goldenfarm-Cherry, Sweet-Pie, Wonder-Bread}
01	T300	{King's-Cereal, Sunset-Milk, Dairyland-Cheese, Best-Bread}
03	T400	{Wonder-Bread, Farmer's-Milk, Best-Cereal, Sweet-Pie, Dairyland-Cheese}

(a) At the granularity of *item_category* (e.g., item$_i$ could be "*Milk*" and ignore brand name), for the following rule template,

$$\forall X \in \textbf{transaction},\ buys(X, \text{item}_1) \wedge buys(X, \text{item}_2)) \Rightarrow buys(X, \text{item}_3)\ [s, c]$$

list the frequent *k*-itemset(s) for the largest *k*, and all of the strong association rules (with their support *s* and confidence *c*) containing the frequent *k*-itemset(s) for the largest *k*. Your answer should include the key steps of the computation process. (10 pts)

(b) At the granularity of *brand−item_category* (e.g., item$_i$ could be "King's-Cereal"), for the following rule template,

$$\forall X \in \textbf{customer},\ buys(X, \text{item}_1) \wedge buys(X, \text{item}_2)) \Rightarrow buys(X, \text{item}_3)$$

list the frequent k-itemset(s) for the largest k (but do not print any rules). Your answer should include the key steps of the computation process. (10 pts)

Solution (a) At the granularity of item_category, let's find the frequent k-itemsets for the largest k and the strong association rules.

Step 1: Generate frequent 1-itemsets, with the count

{Milk}

{Bread}

{Pie}

{Cherry}

{Cheese}

{Cereal}

Let us count the number of occurrences of items.

{Milk}	4/4 ->100%
{Bread}	4/4 ->100%
{Pie}	3/4 -> 75%
{Cherry}	2/4->50%
{Cheese}	3/4 -> 75%
{Cereal}	2/4 -> 50%

The minimum support threshold is 70%, which means an item_category must have a support of at least 70% (≥ 2.8) to be considered frequent.

Frequent 1-itemsets:

{Milk}

{Bread}

{Pie}

{Cheese}

Step 2: Generate frequent 2-itemsets by joining the frequent 1-itemsets.

{Milk, Bread}

{Milk, Pie}

{Milk, Cheese}

{Bread, Pie}

{Bread, Cheese}

{Pie, Cheese}

Count the occurrences or support of each joined 2-itemset:

{Milk, Bread}	4/4 -> 100%
{Milk, Pie}	3/4 -> 75%
{Milk, Cheese}	3/4 -> 75%
{Bread, Pie}	3/4 -> 75%
{Bread, Cheese}	3/4 -> 75%
{Pie, Cheese}	2/4 -> 50%

The minimum support threshold is 70%, so the remaining

Frequent 2-itemsets:

{Milk, Bread}

{Milk, Pie}

{Milk, Cheese}

{Bread, Pie}

{Bread, Cheese}

Step 3: Generate frequent 3-itemsets by joining frequent 2-itemsets:

{Milk, Bread, Pie}

{Milk, Bread, Cheese}

{Milk, Pie, Cheese}

{Bread, Pie, Cheese}

Count the occurrences or support of each generated 3 -itemset:

{Milk, Bread, Pie}	3/4 -> 75%
{Milk, Bread, Cheese}	3/4->75%
{Milk, Pie, Cheese}	2/4 -> 50%
{Bread, Pie, Cheese}	2/4 -> 50%

The minimum support threshold is 70%, so the remaining

Frequent 3-itemsets:

{Milk, Bread, Pie}

{Milk, Bread, Cheese}

Step 4: Generate frequent 4-itemsets by joining frequent 3-itemsets:

{Milk, Bread, Pie, Cheese)

Count the occurrences or support of joined 4-itemset:

{Milk, Bread, Pie, Cheese}   2/4 -> 50%

There are no frequent 4-itemsets that satisfy the minimum support threshold of 70%. Hence we stop here.

Frequent k-itemsets for the largest k i.e 3:

{Milk, Bread, Pie}

{Milk, Bread, Cheese}

Now let us generate association rules.

From the frequent k-itemsets, generate all possible non-empty subsets.

Rules:

1- {Milk, Bread, Pie}:

{Milk} => {Bread, Pie}

{Bread} => {Milk, Pie}

{Pie} => {Milk, Bread}

{Milk, Bread} => {Pie}

{Milk, Pie} => {Bread}

{Bread, Pie} => {Milk}

2- {Milk, Bread, Cheese}:

{Milk} => {Bread, Cheese}

{Bread} => {Milk, Cheese}

{Cheese} => {Milk, Bread}

{Milk, Bread} => {Cheese}

{Milk, Cheese} => {Bread}

{Bread, Cheese} => {Milk}

Let us calculate the support and confidence for each rule:

For the rules obtained from {Milk, Bread, Pie}:

{Milk} => {Bread, Pie}:

Confidence = Support({Milk, Bread, Pie}) / Support({Milk}) = (3) / (4) = 0.75

{Bread} => {Milk, Pie}:

Confidence = Support({Milk, Bread, Pie}) / Support({Bread}) = (3) / (4) = 0.75

{Pie} => {Milk, Bread}

Confidence = Support({Milk, Bread, Pie}) / Support({Pie}) = (3) / (3) = 1

{Milk, Bread} => {Pie}:

Confidence = Support({Milk, Bread, Pie}) / Support({Milk, Bread}) = (3) / (4) = 0.75

{Milk, Pie} => {Bread}:

Confidence = Support({Milk, Bread, Pie}) / Support({Milk, Pie}) = (3) / (3) = 1

{Bread, Pie} => {Milk}:

Confidence = Support({Milk, Bread, Pie}) / Support({Bread, Pie}) = (3) / (3) = 1

Since confidence is 80%, the selected rules are:

{Pie} => {Milk, Bread},

{Milk, Pie} => {Bread}

and {Bread, Pie} => {Milk}.

Now, for the rules derived from {Milk, Bread, Cheese}:

{Milk} => {Bread, Cheese}:

Confidence = Support({Milk, Bread, Cheese}) / Support({Milk}) = (3) / (4) = 0.75

{Bread} => {Milk, Cheese}:

Confidence = Support({Milk, Bread, Cheese}) / Support({Bread}) = (3) / (4) = 0.75

{Cheese} => {Milk, Bread}:

Confidence = Support({Milk, Bread, Cheese}) / Support({Cheese}) = (3) / (3) = 1

{Milk, Bread} => {Cheese}:

Confidence = Support({Milk, Bread, Cheese}) / Support({Milk, Bread}) = (3) / (4) = 0.75

{Milk, Cheese} => {Bread}:

Confidence = Support({Milk, Bread, Cheese}) / Support({Milk, Cheese}) = (3) / (3) = 1

{Bread, Cheese} => {Milk}:

Confidence = Support({Milk, Bread, Cheese}) / Support({Bread, Cheese}) = (3) / (3) = 1

Since confidence is 80%, the selected rules are:

{Cheese} => {Milk, Bread},

{Milk, Cheese} => {Bread},

and {Bread, Cheese} => {Milk}.

So, the association rules with 70% support and 80% confidence are:

{Pie} => {Milk, Bread},

{Milk, Pie} => {Bread},

{Bread, Pie} => {Milk},

{Cheese} => {Milk, Bread},

{Milk, Cheese} => {Bread},

{Bread, Cheese} => {Milk}.

But according to rule given in question we will be removing the one on which on left side we have only one term.

So the final answer for the association rules with 70% support and 80% confidence are:

{Milk, Pie} => {Bread},

{Bread, Pie} => {Milk},

{Milk, Cheese} => {Bread},

{Bread, Cheese} => {Milk}.

Solution (b) At the granularity of brand-item_category, let's find the frequent k-itemsets for the largest k.

Since the customer id is same for the T100 and T300, we combine the two sets in them. Also we have to consider two products different if the brands are different and even thought the product is same

Step 1: Generate frequent 1-itemsets, with the count

{Farmer's-Milk}	3 -> 3/3 =100%
{Wonder-Bread}	3-> 3/3 =100%
{Sweet-Pie}	3-> 3/3 =100%
{Sunny-Cherry}	1 -> 1/3= 33%
{Dairyland-Cheese}	3-> 3/3 =100%

{Goldenfarm-Cherry}	1-> 1/3 =33%
{King's-Cereal}	1-> 1/3= 33%
{Sunset-Milk}	1-> 1/3= 33%
{Best-Bread}	1-> 1/3 =33%
{Best-Cereal}	1-> 1/3 =33%

The minimum support threshold is 70%, so the remaining

Frequent 1-itemsets:

Farmer's-Milk

Wonder-Bread

Sweet-Pie

Dairyland-Cheese

Step 2: Generate frequent 2-itemsets by joining frequent 1-itemsets:

{Farmer's-Milk, Wonder-Bread}

{Farmer's-Milk, Sweet-Pie}

{Farmer's-Milk, Dairyland-Cheese}

{Wonder-Bread, Sweet-Pie}

{Wonder-Bread, Dairyland-Cheese}

{Sweet-Pie, Dairyland-Cheese}

Count the occurrences or support of each generated 2-itemset:

(Farmer's-Milk, Wonder-Bread)	3 -> 3/3= 100%
(Farmer's-Milk, Sweet-Pie)	3-> 3/3 =100%
(Farmer's-Milk, Dairyland-Cheese)	3-> 3/3 =100%
(Wonder-Bread, Sweet-Pie)	3-> 3/3= 100%
(Wonder-Bread, Dairyland-Cheese)	3-> 3/3= 100%
(Sweet-Pie, Dairyland-Cheese)	3-> 3/3= 100%

The minimum support threshold is 70%, so we have to remove the itemset whose support is less than it, so the remaining

{Farmer's-Milk, Dairyland-Cheese}

{Farmer's-Milk, Wonder-Bread}

{Farmer's-Milk, Sweet-Pie}

{Wonder-Bread, Sweet-Pie}

{Wonder-Bread, Dairyland-Cheese}

{Sweet-Pie, Dairyland-Cheese}

Step 3: Let's generate frequent 3-itemsets by joining frequent 2-itemsets:

{Farmer's-Milk, Wonder-Bread, Sweet-Pie}

{Farmer's-Milk, Wonder-Bread, Dairyland-Cheese}

{Farmer's-Milk, Sweet-Pie, Dairyland-Cheese}

{Wonder-Bread, Sweet-Pie, Dairyland-Cheese}

Count the support of each joined 3-itemset:

{Farmer's-Milk, Wonder-Bread, Sweet-Pie}   3 -> 3/3= 100%

{Farmer's-Milk, Wonder-Bread, Dairyland-Cheese} 3 -> 3/3= 100%

{Farmer's-Milk, Sweet-Pie, Dairyland-Cheese}   3 -> 3/3= 100%

{Wonder-Bread, Sweet-Pie, Dairyland-Cheese}   3 -> 3/3= 100%

All the sets meets the minimum threshold of 70%

Step 4: Let's generate frequent 4-itemsets by joining frequent 3-itemsets:

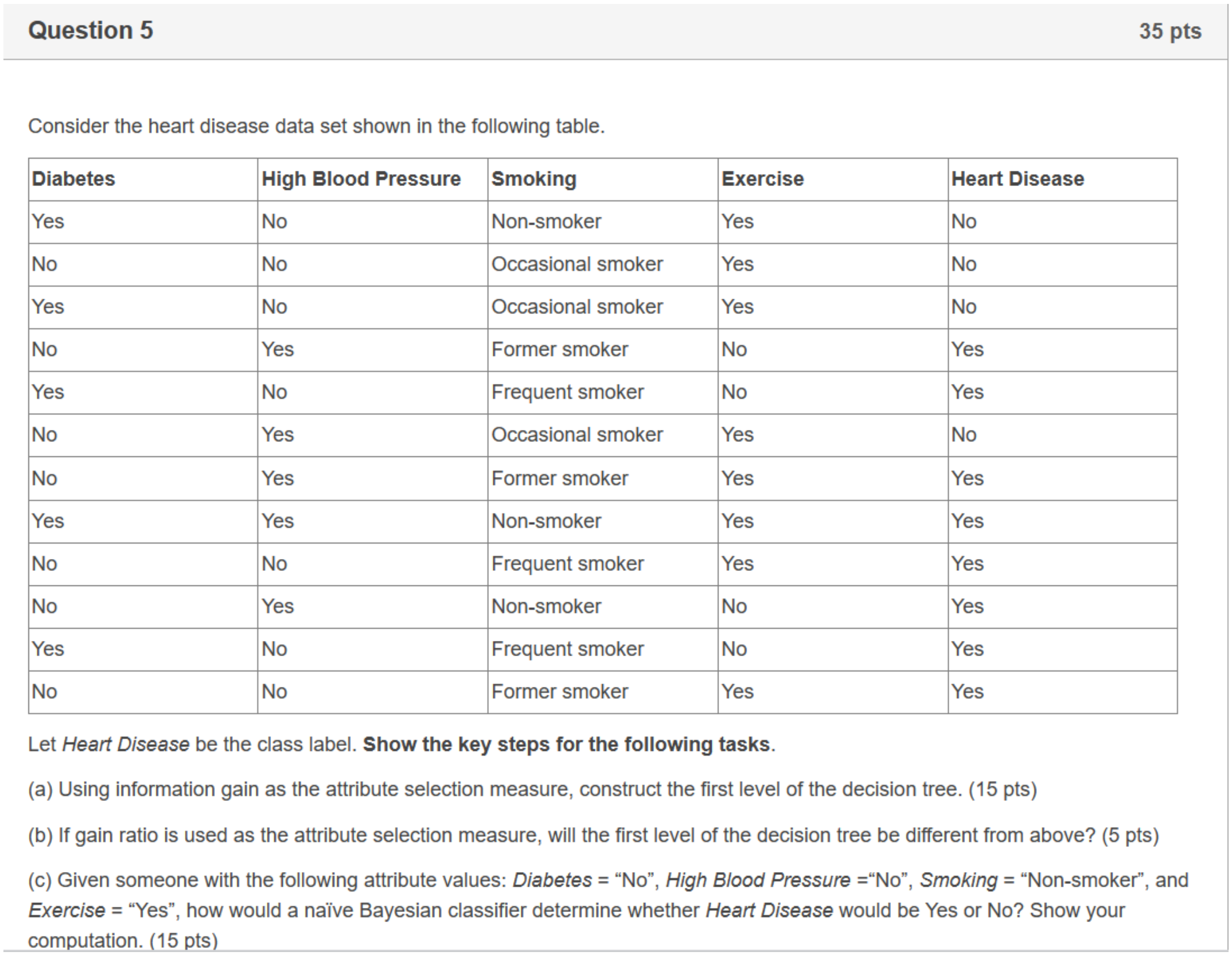
{Farmer's-Milk, Wonder-Bread, Sweet-Pie, Dairyland-Cheese}

Count the support

(Farmer's-Milk, Wonder-Bread, Sweet-Pie, Dairyland-Cheese)   3/3 -> 100%

Now let us try to generate the 5 itemset, but it’s not possible to generate.

Hence, list the frequent k-itemset(s) for the largest k i.e. 4 is
{Farmer's-Milk, Wonder-Bread, Sweet-Pie, Dairyland-Cheese}



Question 5 35 pts

Consider the heart disease data set shown in the following table.

Diabetes	High Blood Pressure	Smoking	Exercise	Heart Disease
Yes	No	Non-smoker	Yes	No
No	No	Occasional smoker	Yes	No
Yes	No	Occasional smoker	Yes	No
No	Yes	Former smoker	No	Yes
Yes	No	Frequent smoker	No	Yes
No	Yes	Occasional smoker	Yes	No
No	Yes	Former smoker	Yes	Yes
Yes	Yes	Non-smoker	Yes	Yes
No	No	Frequent smoker	Yes	Yes
No	Yes	Non-smoker	No	Yes
Yes	No	Frequent smoker	No	Yes
No	No	Former smoker	Yes	Yes

Let *Heart Disease* be the class label. **Show the key steps for the following tasks**.

(a) Using information gain as the attribute selection measure, construct the first level of the decision tree. (15 pts)

(b) If gain ratio is used as the attribute selection measure, will the first level of the decision tree be different from above? (5 pts)

(c) Given someone with the following attribute values: *Diabetes* = "No", *High Blood Pressure* ="No", *Smoking* = "Non-smoker", and *Exercise* = "Yes", how would a naïve Bayesian classifier determine whether *Heart Disease* would be Yes or No? Show your computation. (15 pts)

Solution a)

From the given dataset, first, calculate the expected information needed to classify Heart disease attribute

Info(Heart Disease) = -Sum(pi log(pi) for i=1 to m

= -(8/12)log(8/12) – (4/12)log(4/12)
= -2/3*(-.585) - (1/3)*(-1.585)
= 0.39+0.5283
= 0.9183
Now calculate the information needed to classify Heart Disease using each of the attributes

Diabetes (Values: Yes/No)
Info(Heart, Diabetes) = (5/12)*I(3,2) + (7/12)*I(5,2)
I(3,2) = (-3/5)(log2(3/5)) - (2/5)(log2(2/5)) = 0.971
I(5,2) = (-5/7)(log2(5/7)) - (2/7)(log2(2/7)) = 0.8631
Info(Heart, Diabetes) = 0.908

Info-Gain(Diabetes) = Info(Heart Disease) - Info(Heart, Diabetes) = 0.9183-0.908 = 0.0103

High Blood Pressure (Values: Yes/No)
Info(Heart, High Blood Pressure) = (5/12)*I(4,1) + (7/12)*I(4,3)
I(4,1) = (-4/5)(log2(4/5)) - (1/5)(log2(1/5)) = 0.722
I(4,3) = (-4/7)(log2(4/7)) - (3/7)(log2(3/7)) = 0.985

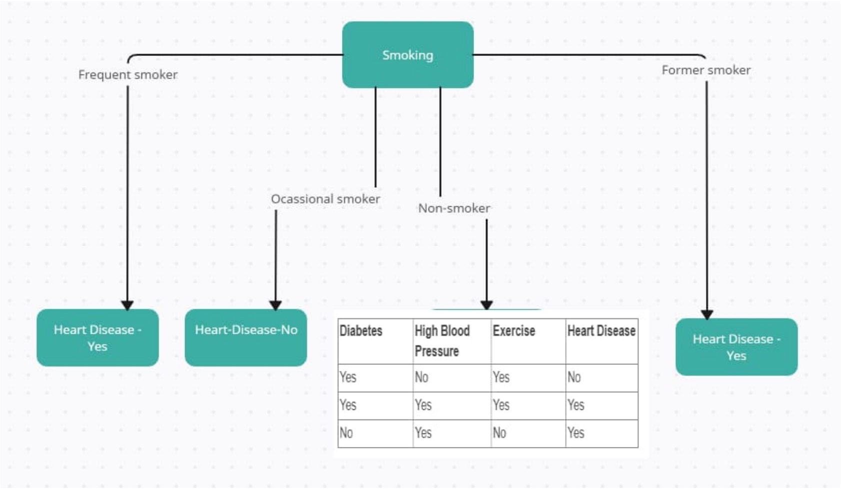
Replacing the values in original eqn, Info(Heart, High Blood Pressure) = 0.8755
Info-Gain(High Blood Pressure) = Info(Heart Disease) - Info(Heart, High Blood Pressure) = 0.9183-0.8755 = 0.0428

Smoking (Values: Frequent smoker, Former smoker, Non-smoker, Occasional smoker)
Info(Heart, Smoking) = (3/12)*I(3,0) + (3/12)*I(3,0) + (3/12)*I(2,1) + (3/12)*I(0,3)
I(3,0) = (-3/3)(log2(3/3)) - (0/3)(log2(0/3)) = 0
I(2,1) = (-2/3)(log2(2/3)) - (1/3)(log2(1/3)) = 0.9183
I(0,3) = (-0/3)(log2(0/3)) - (3/3)(log2(3/3)) = 0
Info(Heart, Smoking) = 0.2295
Info-Gain(Smoking) = Info(Heart Disease) - Info(Heart, Smoking) = 0.9183-0.2295 = 0.6888

Exercise (Values: Yes/No)
Info(Heart, Exercise) = (8/12)*I(4,4) + (4/12)*I(4,0)
I(4,4) = (-4/8)(log2(4/8)) - (4/8)(log2(4/8)) = 1
I(4,0) = (-4/4)(log2(4/4)) - (0/4)(log2(0/4)) = 0
Therefore, Info(Heart, Exercise) = 2/3 = 0.6667
Info-Gain(Exercise) = Info(Heart Disease) - Info(Heart, Exercise) = 0.9183-0.6667 = 0.2516

Based on the above info-gain values for all attributes, Smoking is selected to as the root node for the decision tree.

The level 1 decision tree for the same is shown below



Diabetes	High Blood Pressure	Exercise	Heart Disease
Yes	No	Yes	No
Yes	Yes	Yes	Yes
No	Yes	No	Yes

Solution b)

Gain-Ratio(Attr) = Info-Gain(Attr)/Split-Information(Attr)

Diabetes
Split-Information(Diabetes) = (-5/12)*log2(5/12) + (-7/12)*log2(7/12) = 0.9798
Gain-Ratio(Diabetes) = Info-Gain(Diabetes)/Split-Information(Diabetes) = 0.0103/0.9798 = 0.0105
High Blood Pressure
Split-Information(High Blood Pressure) = (-5/12)*log2(5/12) + (-7/12)*log2(7/12) = 0.9798
Gain-Ratio(High Blood Pressure) = Info-Gain(High Blood Pressure)/Split-Information(High Blood Pressure) = 0.0428/0.9798 = 0.04368
Smoking
Split-Information(Smoking) = 4*(-3/12)*log2(3/12) = 2
Gain-Ratio(Smoking) = Info-Gain(Smoking)/Split-Information(Smoking) = 0.6888/2 = 0.3444
Exercise
Split-Information(Exercise) = (-8/12)*log2(8/12) + (-4/12)*log2(4/12) = 0.9183
Gain-Ratio(Exercise) = Info-Gain(Exercise)/Split-Information(Exercise) = 0.2516/0.9183 = 0.2740

From the above information, the Gain Ratio of Smoking is still the highest and as result it will be the root node for the decision tree. Hence, even if the gain ratio is used as the attribute selection measure, the first level of the decision tree will remain the same.

Solution c)

According to Naïve Bayes theorem,

P(Ck|X=x1,x2…xn) * P(X) = P(Ck) Product(P(xi|Ck) for all i from 1…n

Given X = { Diabetes = “No”, High Blood Pressure =“No”, Smoking = “Non-smoker”, and Exercise = “Yes”}

P(Heart Disease=”Yes”) = 8/12

P(Heart Disease=”No”) = 4/12

P(Diabetes=”No”|Heart Disease=”Yes”) = 5/8

P(High Blood Pressure = “No”|Heart Disease=”Yes”) = 4/8

P(Smoking=”Non-smoker”|Heart Disease=”Yes”) = 2/8

P(Exercise=”Yes”|Heart Disease = “Yes”) = 4/8

Therefore, P(Heart Disease=”Yes”|X) * P(X) = (8/12)(5/8)(4/8)(2/8)(4/8) = 0.026……………….(1)

P(Diabetes=”No”|Heart Disease=”No”) = 2/4

P(High Blood Pressure = “No”|Heart Disease=”No”) = 3/4

P(Smoking=”Non-smoker”|Heart Disease=”No”) = 1/4

P(Exercise=”Yes”|Heart Disease = “No”) = 4/4 = 1

Therefore, P(Heart Disease=”No”|X) * P(X) = (4/12)(2/4)(3/4)(¼)(1) = 0.03125…………………(2)

From (1) & (2) ignoring the factor P(X), since P(Heart Disease=”No”|X)>P(Heart Disease=”Yes”|X) we can conclude that for the given attributes Heart Disease would be a “No”.