ASSIGNMENT 3
CS422 DATA MINING
TA solution

1

1.1 Supermarket dataset: Show the top 4 association rules with Apriori using the default parameters. Discuss what are the main parameters for Apriori that you can modify. Modify support/confidence 3 times: low, medium high, experiment and report the summary of what your learned - explain briefly how your modifications affected the generation of the rules

Supermarket dataset having 217 attributes and 4627 instances.

Apriori Algorithm the TOP 3 Association rules

1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)
2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696 <conf:(0.92)> lift:(1.27) lev:(0.03) [149] conv:(3.28)
3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705 <conf:(0.92)> lift:(1.27) lev:(0.03) [150] conv:(3.27)
4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746 <conf:(0.92)> lift:(1.27) lev:(0.03) [159] conv:(3.26)

- *Customer who buy biscuits, frozen foods, fruit also tend to buy bread and cake*

minimum support and minimum confidence set using default parameters is: -
Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9

Important parameters are
minMetric -- Minimum metric score. Consider only rules with scores higher than this value.If the value is too high the rules generated will be high.
numRules -- Number of rules to find lowerBoundMinSupport -- Lower bound for minimum support.
classIndex -- Index of the class attribute. If set to -1, the last attribute is taken as class attribute.
outputItemSets -- If enabled the itemsets are output as well.
car -- If enabled class association rules are mined instead of (general) association rules.
treatZeroAsMissing -- If enabled, zero (that is, the first value of a nominal) is treated in the same way as a missing value delta -- Iteratively decrease support by this factor. Reduces support until min support is reached or required number of rules has been generated.
upperBoundMinSupport -- Upper bound for minimum support. Start iteratively decreasing minimum support from this value.The value for this parameter must be kept high to consider frequent items
LowerBoundMinSupport- : This value is the least value the support is going to be considered. There wont any rules if we set this value very high hence we must the value to be set based on the dataset and rules to be generated.
metricType – Allows to execute Apriori Algorithm with either confidence or Lift or Leverage or Conviction parameter. But we are supposed to use only confidence metric as of now.

Support: It is set to upperBoundMinSupport and then decreases by delta .Its stops when numRules is reached or when reached to lowerBoundSupport

Confidence	Default 0.9	Low 0.8	Medium 0.86	High 0.93
MinSupport	0.15 with 694 instances	0.13 with 1388 instances	0.2 with 925 instances	0.1 with 463 instances
NumOfCycles	17	14	16	18
NumOfItemsets	6	3	5	7
LargestItemset	L (3): 910	L (2): 69	L (3): 302	L (4): 3950
SmallestItemset	L (6): 1	L (3): 20	L (5): 2	L (7): 20
Top Rule	1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723	1. biscuits=t vegetables=t 1764 ==> bread and cake=t 1487	1. biscuits=t frozen foods=t fruit=t vegetables=t 1039 ==> bread and cake=t 929	1. biscuits=t frozen foods=t party snack foods=t fruit=t vegetables=t total=high 510 ==> bread and cake=t 478
Rule Threshold	<conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)	<conf:(0.84)> lift:(1.17) lev:(0.05) [217] conv:(1.78)	<conf:(0.89)> lift:(1.24) lev:(0.04) [181] conv:(2.62)	<conf:(0.94)> lift:(1.3) lev:(0.02) [110] conv:(4.33)

- To modify confidence, we will change the minMetric as metricType is set to Confidence.
- When confidence value is highest with 0.93 the support is lowest with 0.1 indicating that the itemset generated should have minimum support value of 0.1 which will lead to generating association rules with 0.93 confidence as evident from the rule biscuits=t frozen foods=t party snack foods=t fruit=t vegetables=t total=high 510 ==> bread and cake=t 478, where bread and cake appears 478/510 times when biscuits, frozen foods, party snack foods, fruit, vegetable and total is high.
- The medium value of confidence set to 0.86 leads support to be at 0.2 and generates results that align closely with results of defaults parameters of Apriori algorithm.
- When the value for confidence is set low as 0.8 the support is higher at 0.13, which generates smaller association rule with a smaller number of iterations and smaller itemset.
- the default parameters for Apriori Algorithm are set to nearly medium value of confidence and Support so that optimum association rules and itemset results are achieved.
- No of rules that are generated decreases as we increase the support and confidence also because we are changing the parameters the rules generated are not the same they change inorder to satisfy the parameters.

1.2

Apriori algorithm is run on similar parameters after removing the attributes, no association rules are generated satisfying confidence of 0.9 and MinSupport of 0.1

- To see newly generated association rules the confidence is lowered to 0.8 with support of 0.1
- For the default support of 0.1 we have rules but not along with high confidence of 0.9

1. laundry needs=t margarine=t total=high 613 ==> tissues-paper prd=t 507 <conf:(0.83)> lift:(1.7) lev:(0.05) [209] conv:(2.95)
2. juice-sat-cord-ms=t laundry needs=t total=high 603 ==> tissues-paper prd=t 489 <conf:(0.81)>

lift:(1.67) lev:(0.04) [196] conv:(2.7)
3. sauces-gravy-pkle=t laundry needs=t total=high 621 ==> tissues-paper prd=t 501 <conf:(0.81)>
lift:(1.66) lev:(0.04) [199] conv:(2.64)

- number of rules formed drastically reduces as we remove these main itemsets
- we also see as we have removed the most frequent items there are no rules generated after 0.7

	Low	Medium	Large
Support	0.1	0.7	0.95
Confidence	0.1	0.7	0.95
No of rules	53k	0	0

1.3 Results of association and classifier

```
Best rules found:

 1. adoption-of-the-budget-resolution=y physician-fee-freeze=n 219 ==> Class=democrat 219    <conf:(1)> lift:(1.63) lev:(0.19) [84] conv:(
 2. adoption-of-the-budget-resolution=y physician-fee-freeze=n aid-to-nicaraguan-contras=y 198 ==> Class=democrat 198    <conf:(1)> lift:(
 3. physician-fee-freeze=n aid-to-nicaraguan-contras=y 211 ==> Class=democrat 210    <conf:(1)> lift:(1.62) lev:(0.19) [80] conv:(40.74)
 4. physician-fee-freeze=n education-spending=n 202 ==> Class=democrat 201    <conf:(1)> lift:(1.62) lev:(0.18) [77] conv:(39.01)
 5. physician-fee-freeze=n 247 ==> Class=democrat 245    <conf:(0.99)> lift:(1.62) lev:(0.21) [93] conv:(31.8)
 6. el-salvador-aid=n Class=democrat 200 ==> aid-to-nicaraguan-contras=y 197    <conf:(0.98)> lift:(1.77) lev:(0.2) [85] conv:(22.18)
 7. el-salvador-aid=n 208 ==> aid-to-nicaraguan-contras=y 204    <conf:(0.98)> lift:(1.76) lev:(0.2) [88] conv:(18.46)
 8. adoption-of-the-budget-resolution=y aid-to-nicaraguan-contras=y Class=democrat 203 ==> physician-fee-freeze=n 198    <conf:(0.98)> lift
 9. el-salvador-aid=n aid-to-nicaraguan-contras=y 204 ==> Class=democrat 197    <conf:(0.97)> lift:(1.57) lev:(0.17) [71] conv:(9.85)
10. aid-to-nicaraguan-contras=y Class=democrat 218 ==> physician-fee-freeze=n 210    <conf:(0.96)> lift:(1.7) lev:(0.2) [86] conv:(10.47)
```

```
physician-fee-freeze = n: democrat (253.41/3.75)
physician-fee-freeze = y
| synfuels-corporation-cutback = n: republican (145.71/4.0)
| synfuels-corporation-cutback = y
| | mx-missile = n
| | | adoption-of-the-budget-resolution = n: republican (22.61/3.32)
| | | adoption-of-the-budget-resolution = y
| | | | anti-satellite-test-ban = n: democrat (5.04/0.02)
| | | | anti-satellite-test-ban = y: republican (2.21)
```

- The common attributes considered by both are physician-fee-freeze, and adoption-of the-budget resolution
- physician-fee-freeze is considered in most of the association rules generated by Apriori and is also the root for J48 algorithm for classification.
- the decision tree classifier chooses the attribute that can most precisely classify the main class
- attribute into either democrat or republican.
- The Apriori generates most of the association rules consisting of these two attributes, based on our confidence rule set to 0.9 with support of 0.1, as both these attribute values together determine appropriate class distribution.

- In our previous assignment we saw that physician-fee-freeze was the attribute that classified most of the democrats (proper distribution to distinguish which we can see in the last rule generated here
- We knew that the data contains more democrat and support for democrats is more when compared to republicans hence we see democrat in the RHS of the rules generated
- Attributes with high gain and correlation are part of best rules .As explained in the previous assignment these parameters were the best for classification and also over here they provided best rules with reduced no of cycles and instances

2

2.1

$$GINI(t) = 1 - \sum_{0}^{c-1} [p(i/t)]^2$$

p(i / t) is the fraction of records belonging to the class " i " at a given node t

A Gini index at the root of the decision tree.

	counts
CLASS C0	10
CLASS C1	10

Gini = $1 - (10/20)^2 - (10/20)^2$

= 1- ¼ – ¼

= 0.5

B Gini index for the attribute CustomerId

$Gini_{(1)} = 1 - (1/1)^2 - (0/1)^2 = 0$

$Gini_{(2)} = 1 - (1/1)^2 - (0/1)^2 = 0$

…

...

$Gini_{(20)} = 1 - (1/1)^2 - (0/1)^2 = 0$

Thus, Gini index for Customer ID is 0

c) Calculate the Gini index for the attribute gender.

	Male	Female
CLASS C0	6	4
CLASS C1	4	6

$Gini_{(M)} = 1 - (6/10)^2 - (4/10)^2 = 0.48$

$Gini_{(F)} = 1 - (4/10)^2 - (6/10)^2 = 0.48$

Weighted average

Gini $_{(GENDER)}$ = (n_m/n * Gini$_{(M)}$) + (n_m/n * Gini$_{(M)}$ +)

= (10/20 * 0.48) +(10/20 *.48)

= 0.48

d) Calculate the Gini index for the attribute car type

Gini $_{(Family)}$ = $1 - (1/4)^2 - (3/4)^2 = 0.375$

Gini $_{(Sports)}$ = $1 - (8/8)^2 - (0/8)^2 = 0$

Gini $_{(Luxury)}$ = $1 - (1/8)^2 - (7/8)^2 = 0.21875$

Total =20

Family = 4, Sports = 8, Luxury = 8

Weighted average

Gini $_{(Car\ Type)}$ = [(4/20 * 0.375) + (8/20 * 0) + (8/20 * 0.21875)]

= 0.1625

	Family	Sports	Luxury
Class C0	1	8	1
Class C1	3	0	7

e) Calculate the Gini index for the attribute shirt size

	Small	Medium	Large	Extra Large
Class C0	3	3	2	2
Class C1	2	4	2	2

Gini$_{(small)}$ = $1 - (3/5)^2 - (2/5)^2 = 0.48$

Gini $_{(Medium)}$ = $1 - (3/7)^2 - (4/7)^2 = 0.486$

Gini $_{(Large)}$ = $1 - (2/4)^2 - (2/4)^2 = 0.5$

Gini $_{(ExtraLarge)}$ = $1 - (2/4)^2 - (2/4)^2 = 0.5$

Total = 20

Small = 5, Medium = 7, Large = 4, Extra Large = 4

Gini $_{(shirt\ size)}$ = [(5/20 * 0.48) + (7/20 * 0.489) + (4/20 * 0.5) + (4/20 * 0.5)] = 0.4915

f) Conclusion:

- The Gini coefficient provides an index to measure inequality. The attribute with lowest Gini is preferred as it has less impurity and proper distinct distribution when compared to other attribute.
- Customer ID has 0, car type has Gini of 0.1625, Gender has Gini of 0.48 and shirt size has 0.491.
- The value of Gini of each attribute keeps decreasing with distinction (we can see for customerID and car type they have lesser value than others)
- Based on the Gini index values, we can see that Customer ID has the lowest Gini value and then car type attribute
- Having the lowest Gini doesn't help also because if we look at attribute customer ID it has 0 but cannot be used for splitting. Customer ID should not be used for splitting because each

value is unique for the attribute and will result in large no of nodes.There is no generalization to each new samples as it just provides a new node without any proper rule for classification

- Hence, we can use the second-best attribute, Car Type would be the better attribute as per the Gini index with least impurity and gives you a better classification with lesser nodes compared to customer ID.The leaf nodes have high degree of homogeneity and low impurity and able to distinguish the classes.
- When we look at the distribution tables above, we can see that by using the car type attribute we can classify classes better when compared to other attributes. Attribute shirtsize and gender does not have unique distributes for each class as we can see that each type has nearly equal class distribution (Ex Male 6 C0 and 4 C1, Female 4C0 and 6C1. Gini value for these attribute is also near to 0.5). So, each gender has C0 and C1 in similar proportions. The leaf nodes have low degree of homogeneity and high impurity and are not able to distinguish the classes. Therefore, gender is not a discriminative attribute. On the other hand, Car Type Sports has mostly C0 and other car types mostly C1. Car Type is a more discriminative attribute. The Gini index values confirm this intuition.
- Hence a Gini value of 0.5 shows that the attribute has equal distribution of classes and as the value decreases to 0 (we have better distinct distribution and classification for each classes)
- cartype has value closer to 0 and shirt size, gender closer to 0.5. Gini index reflects the how well you can classify.
- Car Type will be the best attribute.

2.2

a

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	−

Equations used are

Entropy (t) = $- \sum_{i=0}^{c-1} P(i/t) \log_2 P(i/t)$

P(i/t) is relative frequency of class i at a given node t

C= no of classes

The value of entropy must be less .1 meaning impure and 0 means purest (homogeneous node)

Before splitting

	counts
CLASS +	4
CLASS -	5

$E_0 = - [(4/9 * \log_2 (4/9)) + (5/9 * \log_2 (5/9))] = 0.991$

- Entropy means uncertainty, randomness. High entropy means it hard to predicate the next possibility. Entropy is intuitively a measure of how distributed the probabilities are. The more equal is the share for the probability values in all the classes, the higher is the entropy. The

more skewed is the share among the classes, lesser is the entropy. Maximum value of Entropy is 1 (when they have equal distribution in classes) and minimum is 0. It is difficult to classify or predict when the value is high.

- ❖ Entropy is very high (uniformly distributed for each class) and it is difficult to predict weather the instance will belong to class "+" or "-". We need to use attribute that decreases the value of entropy and has better classification of each class in each split.
- ❖ As per the table we don't have proper discrimination among the classes and hence entropy is higher

B

Calculate the information gain of attribute a1 and of attribute a2. Construct the table with the values of a1 per class label and do the same for a2. Explain how you are using that table in your calculation.

Information Gain = Entropy of Parent - Weighted Entropy of Children Nodes
= E (Parent) - Σ ($N_{(v\,i)}$ / N * I(v i))
"N" is the total number of records at the parent node
"k" is the number of attribute values, and
"N(v j)"is the number of records associated with the child node.

- ❖ Information gain gives the value indicating how much better the classifier can do while splitting on an attribute.
- ❖ Information gain must be high as by using the attribute for splitting we get max information. If the Weighted Entropy of Children Nodes is less, then information gain is more
- ❖ If the Weighted sum of the entropy of children is less, we will get high Information gain.

Class	T	F
+	3	1
-	1	4

Entropy a1(T) = -(3/4) * $\log_2$ (3/4) - (1/4) * $\log_2$ (1/4) = 0.8113

Entropy a1(F) = -(1/5) * $\log_2$ (1/5) - (4/5) * $\log_2$ (4/5) = 0.7219

Information gain for a1
= E_0 – entropy of children's
= 0.9911 – [(4/9 * 0.8113) + (5/9 * 0.7219)] = **0.2294**

Class	T	F
+	2	2
-	3	2

Entropy a1(T) = - (2/5) * log 2 (2/5) - (3/5) * log 2 (3/5) = 0.97
Entropy a2(F) = - (2/4) * log 2 (2/4) - (2/4) * log 2 (2/4) = 1
Information gain for a2

$= E_0 - \Sigma (N(v_j) / N * I(v_j))$
$= 0.99 - [(5/9 * 0.97) + (4/9 * 1)] =$ **0.0072**

C

- The entropy of the child nodes will always be less than or equal to the parent node.
- Information gain is how much information we gain by doing the split using an attribute (benefit of splitting using the attribute)
- when we see this above example, we can see that attribute a1 has higher information gain and hence it is better to use this for the split than the attribute a2 as we get better distribution of classes (splitting on a2 attribute gives more of equal distribution and through a1 we get better classification).
- We must use a better attribute that provides more gain (has proper distribution with homogeneity)
- For the complete dataset the labels are uniformly distributed (we can see that we have 4 – class and 5 + class, which is more uniform).
- Hence, we need to find the attribute that provides more information gain or less weighted entropy (attribute that has proper distribution with majority classes like a1 attribute).
- In a1 attribute we can see that for T type there is 3 +ve classes and 1 – classes. F type there is 1 + ve classes and 4 – classes.
- Other attributes have more of uniform distribution, we need attribute having concentrated labels for each class to get more information gain after the split.
- If we use a1 for splitting, any new instance will be classified with small element of surprise (better classification) but if we use a2 attribute we find there is equal distribution and hence equal probability of classification (having high element of surprise, less accurate classification)

2.3.

a)

Consider these 2 decision trees. they are constructed from data with 3 classes and 16 binary attributes. Calculate the cost of each tree
using the MDL principle. What tree should you choose in term of the expected generalization error? Explain briefly but clearly why? Base your answer on the discussions we had in class.
according to MDL principle the total description length is given as

Cost(model, data) = Cost(model) + Cost(data|model)

Cost (model) = cost of encoding each internal node plus each leaf node
Cost (data | model) = cost of classification errors on the training set by the tree
16 attributes, 3 classes:

Cost of encoding each internal node: $\log_2 m$, m=attributes

cost of encoding each attribute is $\log_2 (16) = 4$ bits.

Cost of encoding each leaf: $\log_2 k$, k=classes

the cost for each leaf node is $\log_2 (3) = 2$ bits.

Cost for each misclassification error is $\log_2 (n)$.

For the first decision tree (7 errors)

Cost = (no of internal nodes * cost of encoding) + (no of leaf nodes * cost of encoding) + (no of misclassification *Cost for each misclassification error)
$(2 * 4) + (3 * 2) + (7 * \log_2 n) =$ **$14 + 7\log_2 n$**

For the second decision tree (4 errors)

Cost = (no of internal nodes * cost of encoding) + (no of leaf nodes * cost of encoding) + (no of misclassification *Cost for each misclassification error)

$(4 * 4) + (5 * 2) + (4 * \log_2 n) =$ **$26 + 7\log_2 n$**

According to mdl, the best is the one that comprises the data to the most. If we have 2 models of similar generalization errors, we must prefer simpler model over the more complex as they have a greater of overfitting through errors and does not generalize the new unseen data. Hence by looking we can say that tree a is better than b.

No of training instances	Tree a cost	Tree b cost
8	35	38
16	42	42
32	49	46

N is found by equating both

$14 + 7\log n = 26 + 4\log n$

$3 \log_2 n = 12$

$n = 2^4 = 16$

For n<16 a is better (we can see that if n <16, tree b overfits as generalization error for tree b is more because of the complex model hence we need to prune tree b)

For n >16 b is better.

$e'(T) = e(T) + N \times 0.5$ (N: number of leaf nodes)

for a: $(7 + 3*0.5)/16 = 0.53$

for b: $(4 + 5*0.5)/16 = 0.41$

- based on the generalization error, we can see that we have error for Tree a when compared to hence we can say that Decision tree b is better

b)

Explain how you will use the calculations from (a) in the decision tree post-pruning process. Explain briefly what post-pruning is and why we use it. Outline each step in the decision tree induction process with post-pruning and what will be the resulting tree. You don't have to repeat all steps of the tree induction process, you can mention it as just one step in your outline

Generalization error is a useful tool to measure whether the node needs to be prune or not. If the generalization error in the parent node is smaller than the child node, we will trim the child node.

By using the above generalization error (measure the purity and error of the node). If the parent node has smaller generalization error than the child nodes, then we can prune the child nodes. If the error is same, we can use the MDL principle to make the decision.
For the given node and corresponding subtree decide whether to prune the subtree or not by the mdl criteria (Using the mdl cost for each decision tree we can do post pruning). The MDL principle trades off tree size for training errors.
In 2.3 a we can see that if n <16, tree b overfits as generalization error for tree b is more because of the complex model hence we need to prune tree b

Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances.
P**runing** reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting. Post-pruning gives better results than prepruning as it makes pruning decisions on a fully-grown tree whereas prepruning can suffer from premature termination of tree-growing process.
Post-**pruning** allows the tree to perfectly classify the training set, and then **post prune** the tree rather than stopping the growing tree earlier
steps involved are

- grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom up fashion
- Trimming can be done the most frequently used branch of the subtree or by replacing a subtree with a new leaf node whose class label is determined from the majority class of records affiliated with the subtree. The tree-pruning step terminates when no further improvement is observed
- If generalization error improves after trimming, replace subtree by a leaf node.
- Class label of leaf node is determined from majority class of instances in the subtree

So, for the 2.3a tree b example

After building the complete tree, we try to trim the nodes from bottom up and calculate the generalization error going up from the leaf nodes assuming the nodes to be pruned .We find that the generalization error improves by making the left hand child of root node the leaf node with label c1(Pruning all of the children in the left branch after the first split and making it a leaf node with the majority classes of instances that are present in the subtree ie c1).