Experiment 1.4

Student Name: Nikhil Kumar **UID: 20BCS1817**
Branch: BE-CSE **Section/Group: 20BCS_DM-716 B**
Semester: 6 **Date of Performance: 07/03/23**
Subject Name: DM LAB **Subject Code: 20CSP_376**

AIM :-

Demonstration of FP Group algorithm on supermarket data.

Theory And Output :-

The two primary drawbacks of the Apriori Algorithm are:

1. *At each step, candidate sets have to be built.*

2. *To build the candidate sets, the algorithm has to repeatedly scan the database.*

These two properties inevitably make the algorithm slower. To overcome these redundant steps, a new association-rule mining algorithm was developed named Frequent Pattern Growth Algorithm. It overcomes the disadvantages of the Apriori algorithm by storing all the transactions in a Trie Data Structure. Consider the following data:-

Transaction ID	Items
T1	{E, K, M, N, O, Y}
T2	{D, E, K, N, **O**, Y}
T3	{A, E, K, M}
T4	{C, K, M, U, Y}
T5	{C, E, I, K, O, O}

The above-given data is a hypothetical dataset of transactions with each letter representing an item. The frequency of each individual item is computed:-

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
0	3
U	1
Y	3

Let the minimum support be 3. A Frequent Pattern set is built which will contain all the elements whose frequency is greater than or equal to the minimum support. These elements are stored in

descending order of their respective frequencies. After insertion of the relevant items, the set L looks like this:-

L = {K : 5, E : 4, M : 3, O : 3, Y : 3}

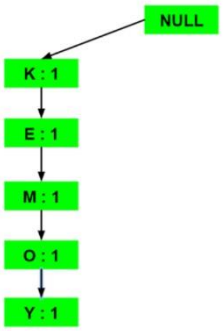
Now, for each transaction, the respective Ordered-Item set is built. It is done by iterating the Frequent Pattern set and checking if the current item is contained in the transaction in question. If the current item is contained, the item is inserted in the Ordered-Item set for the current transaction. The following table is built for all the transactions:

Transaction ID	Items	Ordered-Item Set
T1	{E, K, M, N, O, Y}	{K, E, M, O, Y}
T2	{D, E, K, N, O, Y}	{K, E, O, Y}
T3	{ A, E, K, M}	{K, E, M}
T4	{C, K, M, U, Y}	{K, M, Y}
T5	{C, E, I, K, O, O}	{K, E, O}

Now, all the Ordered-Item sets are inserted into a Trie Data Structure.

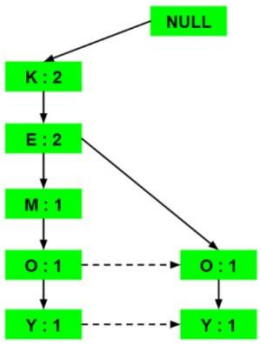
a) Inserting the set {K, E, M, O, Y}:

Here, all the items are simply linked one after the other in the order of occurrence in the set and initialize the support count for each item as 1.



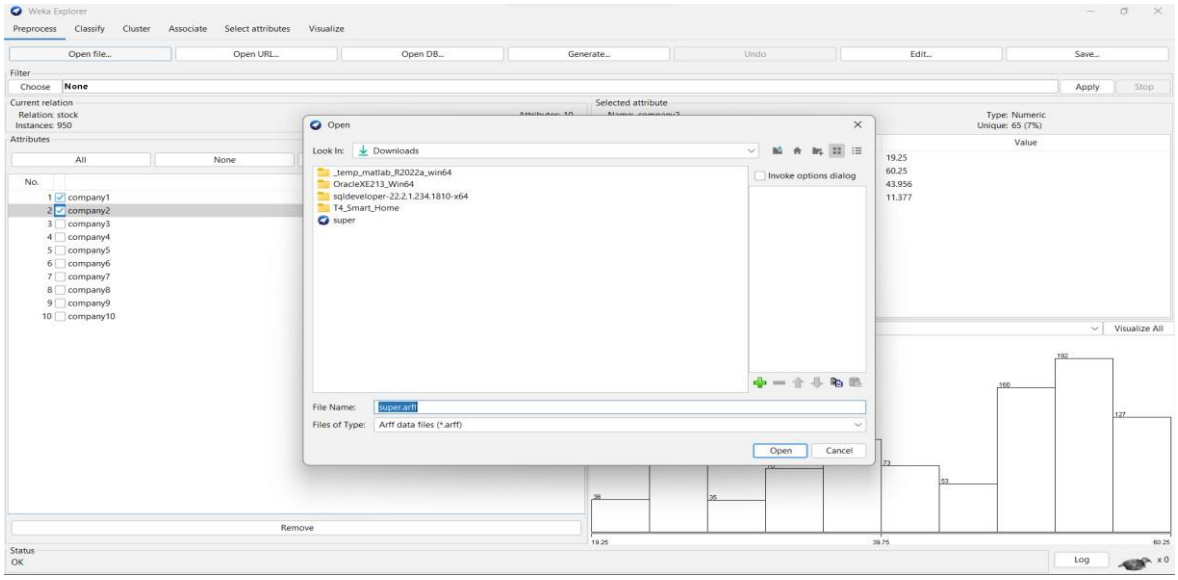
b) **Inserting the set {K, E, O, Y}:**

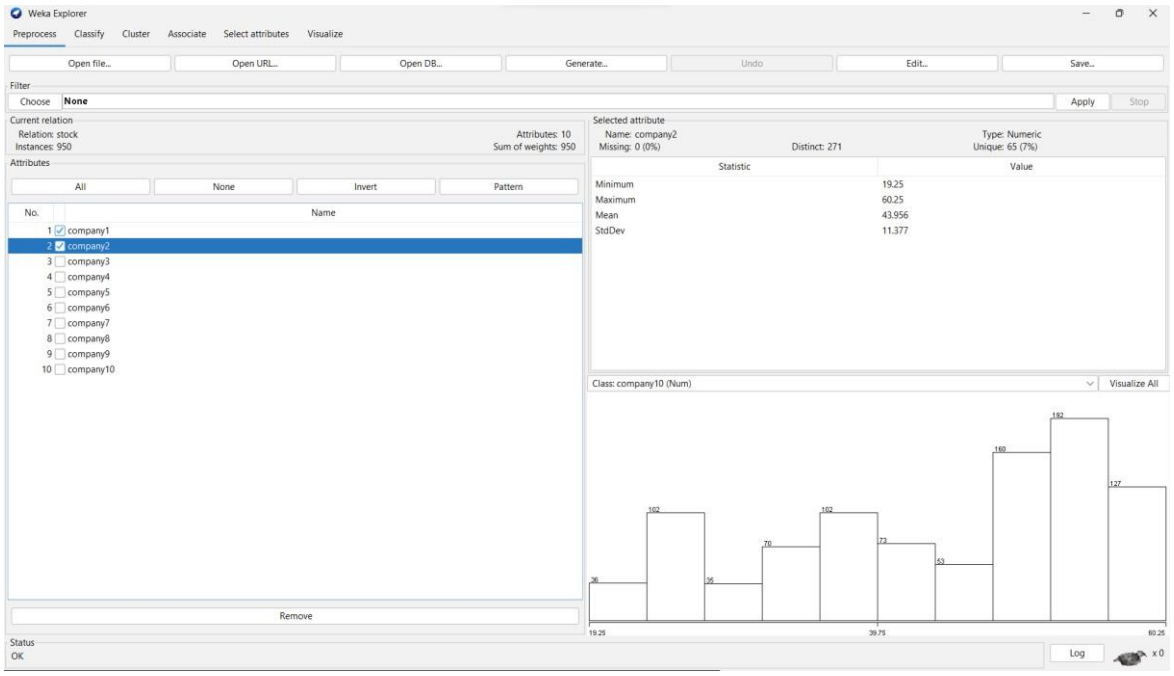
Till the insertion of the elements K and E, simply the support count is increased by 1. On inserting O we can see that there is no direct link between E and O, therefore a new node for the item O is initialized with the support count as 1 and item E is linked to this new node. On inserting Y, we first initialize a new node for the item Y with support count as 1 and link the new node of O with the new node of Y.

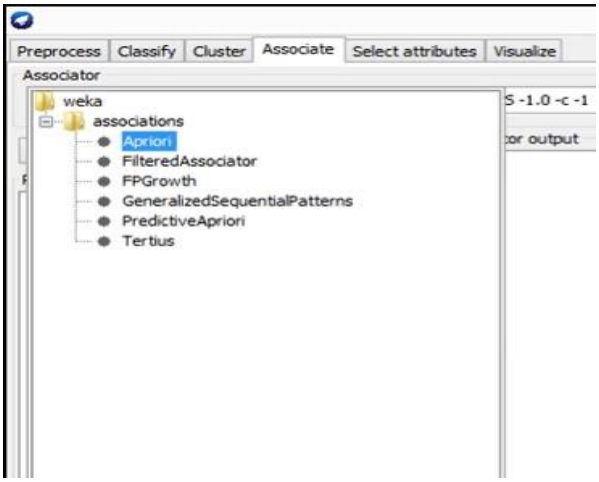


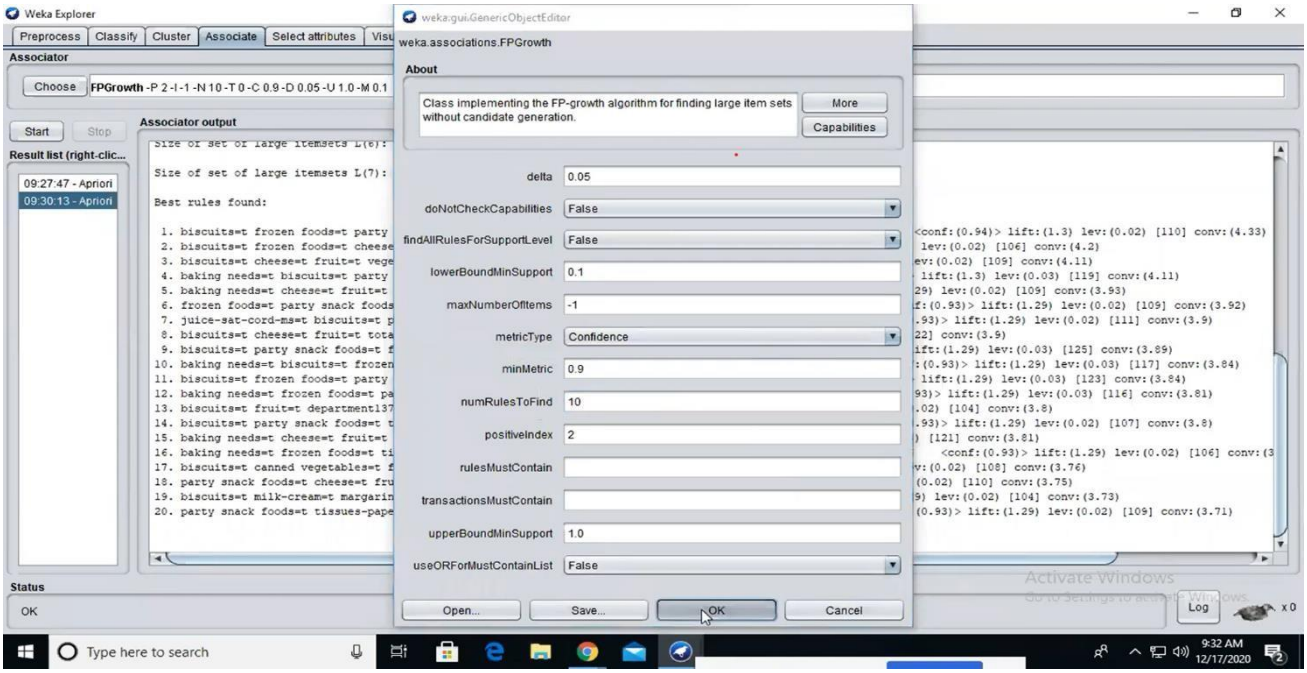
Now, for each item, the Conditional Pattern Base is computed which is path labels of all the paths which lead to any node of the given item in the frequent-pattern tree. Note that the items in the below table are arranged in the ascending order of their frequencies.

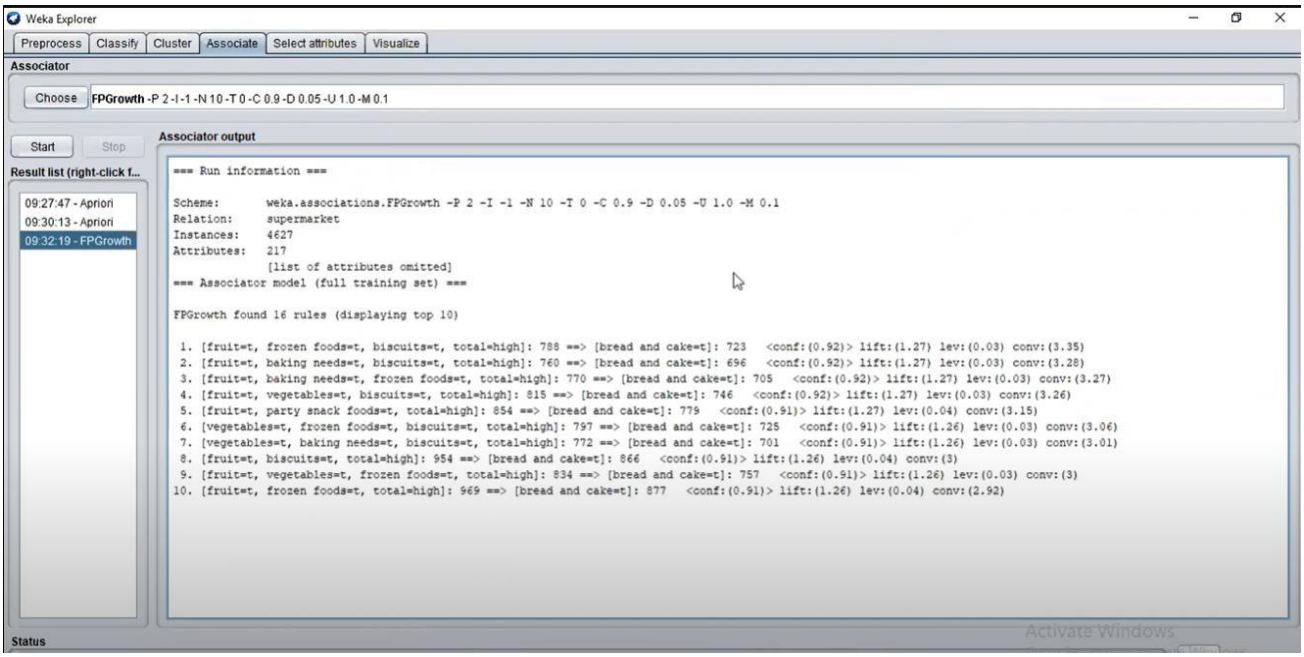
Output :-






Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
weka
associations
Apriori
FilteredAssociator
FPGrowth
GeneralizedSequentialPatterns
PredictiveApriori
Tertius


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Associator
Choose
FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Start
Stop
Result list (right-clic...
09:27:47 - Apriori
09:30:13 - Apriori
Associator output
Size of set of large itemsets L(7):
Best rules found:
1. biscuits=t frozen foods=t party
2. biscuits=t frozen foods=t cheese
3. biscuits=t cheese=t fruit=t vege
4. baking needs=t biscuits=t party
5. baking needs=t cheese=t fruit=t
6. frozen foods=t party snack foods
7. juice-sat-cord-ms=t biscuits=t p
8. biscuits=t cheese=t fruit=t tota
9. biscuits=t party snack foods=t f
10. baking needs=t biscuits=t frozen
11. biscuits=t frozen foods=t party
12. baking needs=t frozen foods=t pa
13. biscuits=t fruit=t department137
14. biscuits=t party snack foods=t t
15. baking needs=t cheese=t fruit=t
16. baking needs=t frozen foods=t ti
17. biscuits=t canned vegetables=t f
18. party snack foods=t cheese=t fru
19. biscuits=t milk-cream=t margarin
20. party snack foods=t tissues-pape
Status
OK
Log
x 0
weka.gui.GenericObjectEditor
weka.associations.FPGrowth
About
Class implementing the FP-growth algorithm for finding large item sets without candidate generation.
More
Capabilities
delta 0.05
doNotCheckCapabilities False
findAllRulesForSupportLevel False
lowerBoundMinSupport 0.1
maxNumberOfItems -1
metricType Confidence
minMetric 0.9
numRulesToFind 10
positiveIndex 2
rulesMustContain
transactionsMustContain
upperBoundMinSupport 1.0
useORForMustContainList False
Open...
Save...
OK
Cancel
Activate Windows
Type here to search
9:32 AM
12/17/2020


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Start
Stop
Result list (right-click f...
09:27:47 - Apriori
09:30:13 - Apriori
09:32:19 - FPGrowth
Associator output
=== Run information ===
Scheme: weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation: supermarket
Instances: 4627
Attributes: 217
[list of attributes omitted]
=== Associator model (full training set) ===
FPGrowth found 16 rules (displaying top 10)
1. [fruit=t, frozen foods=t, biscuits=t, total=high]: 788 ==> [bread and cake=t]: 723 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.35)
2. [fruit=t, baking needs=t, biscuits=t, total=high]: 760 ==> [bread and cake=t]: 696 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.28)
3. [fruit=t, baking needs=t, frozen foods=t, total=high]: 770 ==> [bread and cake=t]: 705 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.27)
4. [fruit=t, vegetables=t, biscuits=t, total=high]: 815 ==> [bread and cake=t]: 746 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.26)
5. [fruit=t, party snack foods=t, total=high]: 854 ==> [bread and cake=t]: 779 <conf:(0.91)> lift:(1.27) lev:(0.04) conv:(3.15)
6. [vegetables=t, frozen foods=t, biscuits=t, total=high]: 797 ==> [bread and cake=t]: 725 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3.06)
7. [vegetables=t, baking needs=t, biscuits=t, total=high]: 772 ==> [bread and cake=t]: 701 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3.01)
8. [fruit=t, biscuits=t, total=high]: 954 ==> [bread and cake=t]: 866 <conf:(0.91)> lift:(1.26) lev:(0.04) conv:(3)
9. [fruit=t, vegetables=t, frozen foods=t, total=high]: 834 ==> [bread and cake=t]: 757 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3)
10. [fruit=t, frozen foods=t, total=high]: 969 ==> [bread and cake=t]: 877 <conf:(0.91)> lift:(1.26) lev:(0.04) conv:(2.92)
Status
Activate Windows