EXPERIMENT 5

Apriori Algorithm

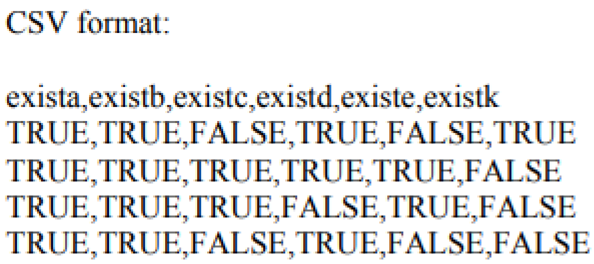
Name: Garima Gupta **UID:** 2019120027

Class: TE EXTC **Subject:** Data Analytics

Aim: To apply the Apriori algoritm to a given dataset Association rule mining with WEKA software

Procedure:

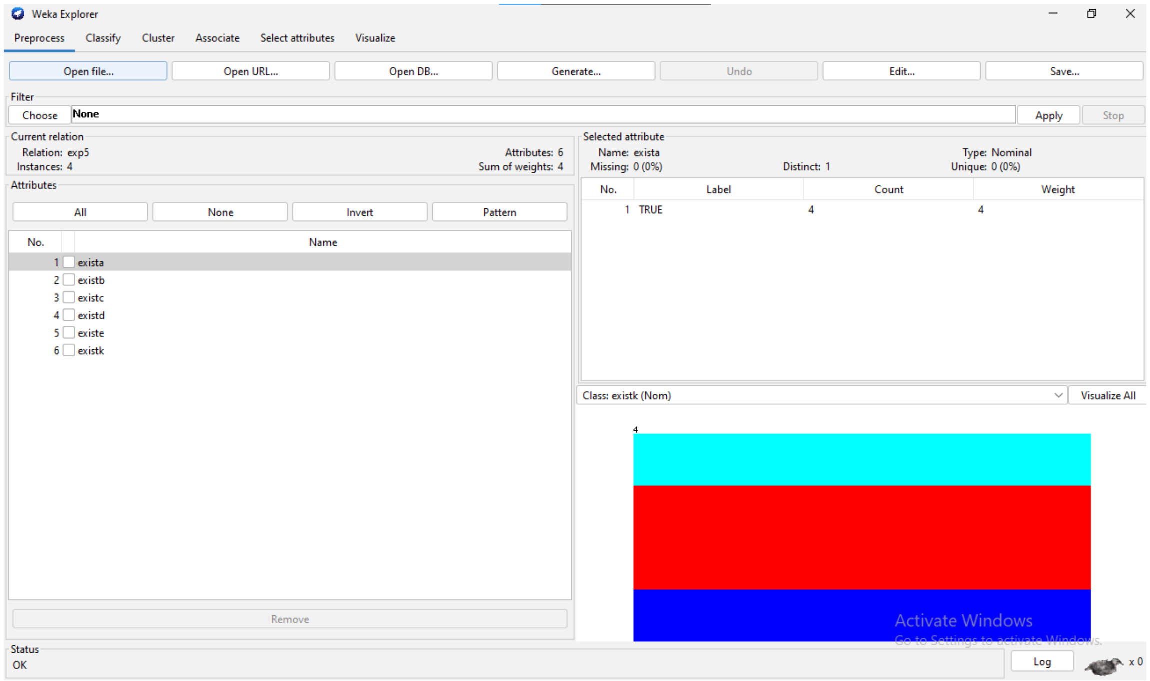
1) Make the CSV file



CSV format:

```
exista,existb,existc,existd,existe,existk
TRUE,TRUE,FALSE,TRUE,FALSE,TRUE
TRUE,TRUE,TRUE,TRUE,TRUE,FALSE
TRUE,TRUE,TRUE,FALSE,TRUE,FALSE
TRUE,TRUE,FALSE,TRUE,FALSE,FALSE
```

2) Open the concerned CSV file in WEKA. It will look like this after opening



Filter: None

Current relation
Relation: exp5
Instances: 4
Attributes: 6
Sum of weights: 4

Attributes

No.	Name
1	□ exista
2	□ existb
3	□ existc
4	□ existd
5	□ existe
6	□ existk

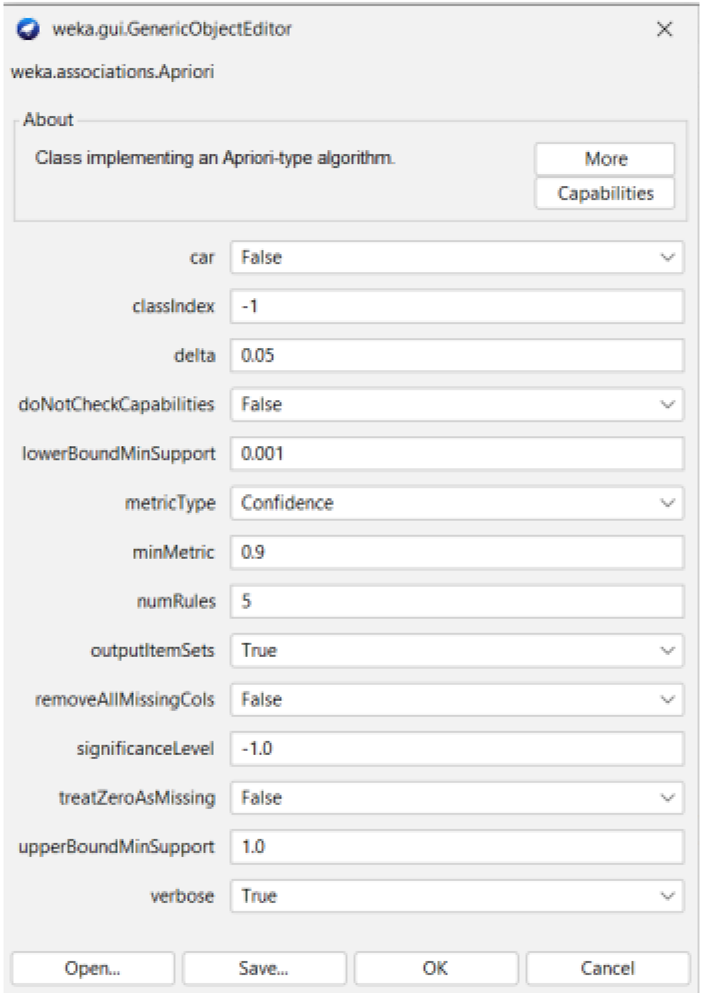
Selected attribute
Name: exista
Type: Nominal
Missing: 0 (0%)
Distinct: 1
Unique: 0 (0%)

No.	Label	Count	Weight
1	TRUE	4	4

Class: existk (Nom)

Status
OK

3) Open Associate and select all the parameters in apriori algoritm.



weka.gui.GenericObjectEditor

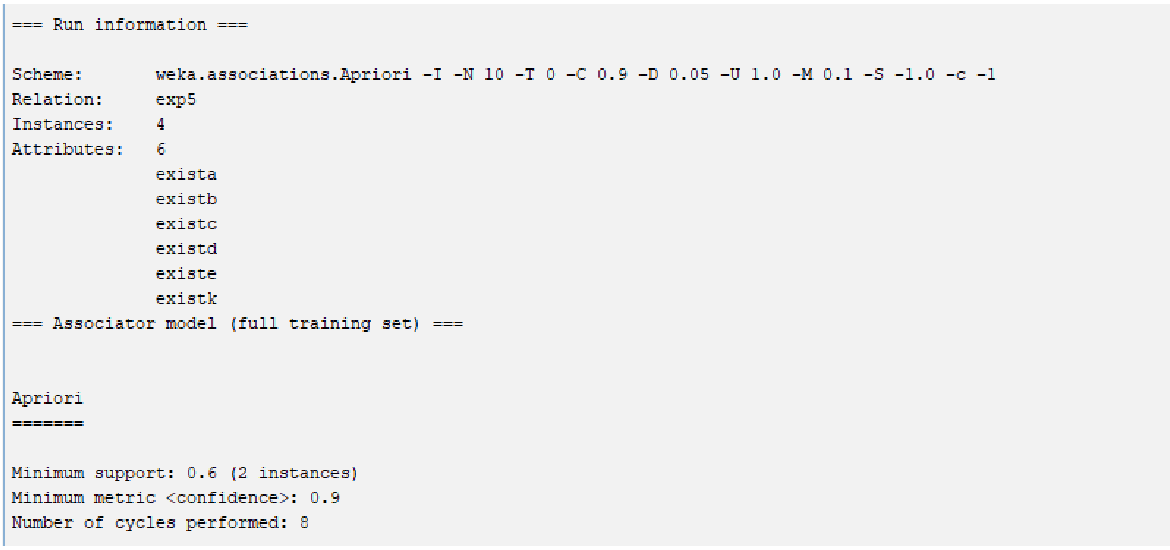
weka.associations.Apriori

About

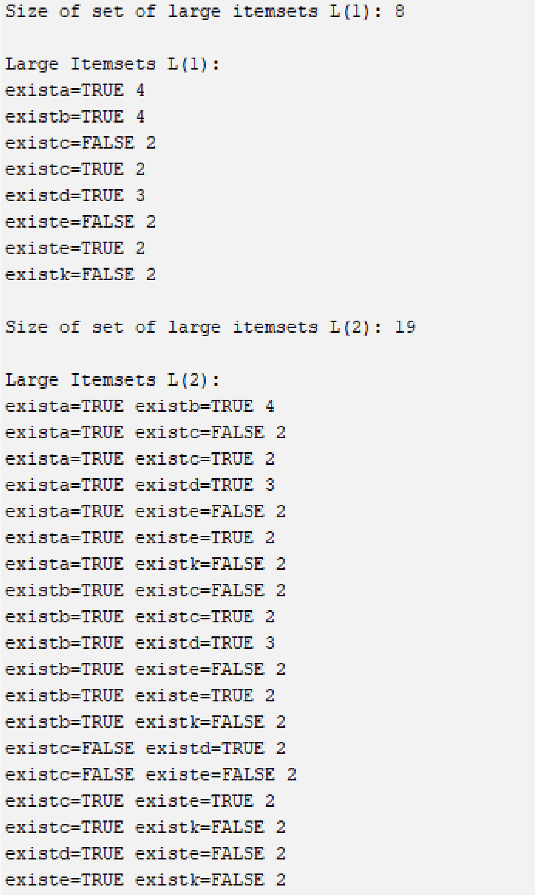
Class implementing an Apriori-type algorithm.

car	False
classIndex	-1
delta	0.05
doNotCheckCapabilities	False
lowerBoundMinSupport	0.001
metricType	Confidence
minMetric	0.9
numRules	5
outputItemSets	True
removeAllMissingCols	False
significanceLevel	-1.0
treatZeroAsMissing	False
upperBoundMinSupport	1.0
verbose	True

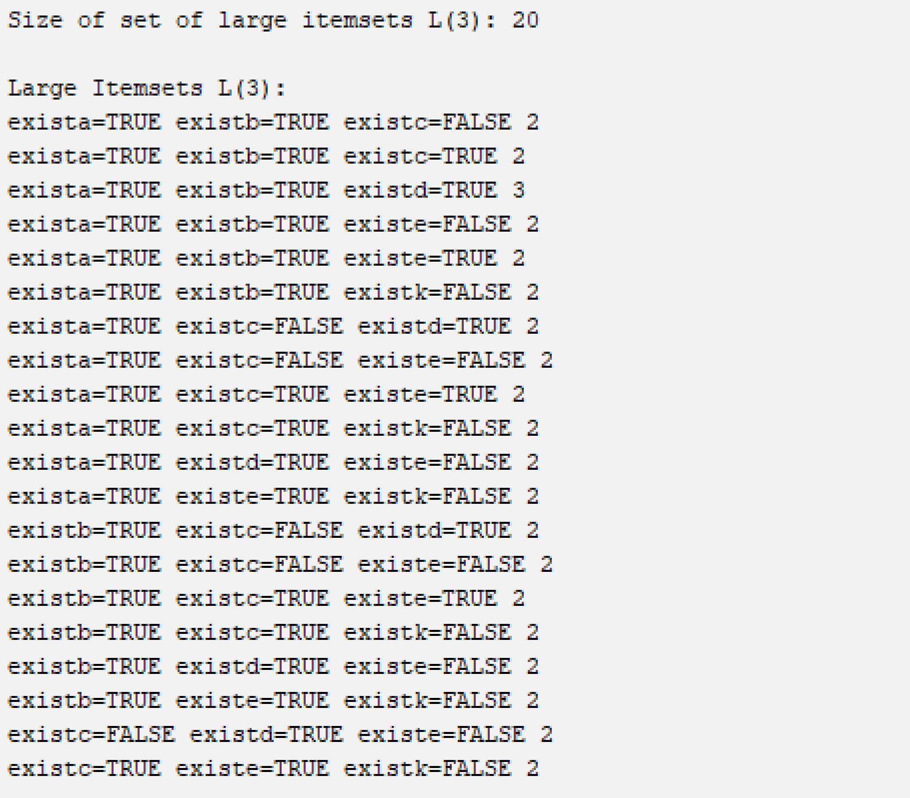
4) Press start so that WEKA can start the Apriori algorithm and find out the best rules. The minimum support is 0.6 and minimum confidence is 0.9.



```
=== Run information ===

Scheme:       weka.associations.Apriori -I -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     exp5
Instances:    4
Attributes:   6
              exista
              existb
              existc
              existd
              existe
              existk
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.6 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 8
```

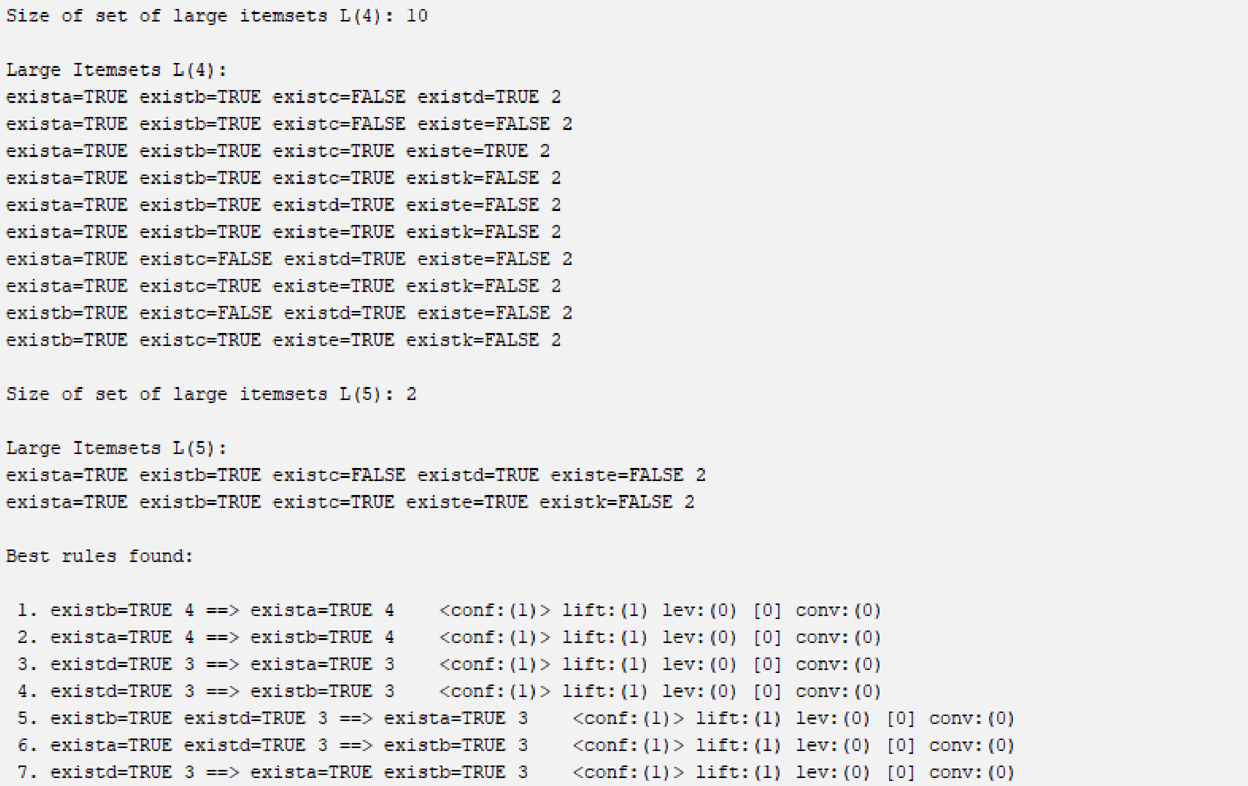


```
Size of set of large itemsets L(1): 8

Large Itemsets L(1):
exista=TRUE 4
existb=TRUE 4
existc=FALSE 2
existc=TRUE 2
existd=TRUE 3
existe=FALSE 2
existe=TRUE 2
existk=FALSE 2

Size of set of large itemsets L(2): 19

Large Itemsets L(2):
exista=TRUE existb=TRUE 4
exista=TRUE existc=FALSE 2
exista=TRUE existc=TRUE 2
exista=TRUE existd=TRUE 3
exista=TRUE existe=FALSE 2
exista=TRUE existe=TRUE 2
exista=TRUE existk=FALSE 2
existb=TRUE existc=FALSE 2
existb=TRUE existc=TRUE 2
existb=TRUE existd=TRUE 3
existb=TRUE existe=FALSE 2
existb=TRUE existe=TRUE 2
existb=TRUE existk=FALSE 2
existc=FALSE existd=TRUE 2
existc=FALSE existe=FALSE 2
existc=TRUE existe=TRUE 2
existc=TRUE existk=FALSE 2
existd=TRUE existe=FALSE 2
existe=TRUE existk=FALSE 2
```



```
Size of set of large itemsets L(3): 20

Large Itemsets L(3):
exista=TRUE existb=TRUE existc=FALSE 2
exista=TRUE existb=TRUE existc=TRUE 2
exista=TRUE existb=TRUE existd=TRUE 3
exista=TRUE existb=TRUE existe=FALSE 2
exista=TRUE existb=TRUE existe=TRUE 2
exista=TRUE existb=TRUE existk=FALSE 2
exista=TRUE existc=FALSE existd=TRUE 2
exista=TRUE existc=FALSE existe=FALSE 2
exista=TRUE existc=TRUE existe=TRUE 2
exista=TRUE existc=TRUE existk=FALSE 2
exista=TRUE existd=TRUE existe=FALSE 2
exista=TRUE existe=TRUE existk=FALSE 2
existb=TRUE existc=FALSE existd=TRUE 2
existb=TRUE existc=FALSE existe=FALSE 2
existb=TRUE existc=TRUE existe=TRUE 2
existb=TRUE existc=TRUE existk=FALSE 2
existb=TRUE existd=TRUE existe=FALSE 2
existb=TRUE existe=TRUE existk=FALSE 2
existc=FALSE existd=TRUE existe=FALSE 2
existc=TRUE existe=TRUE existk=FALSE 2
```

5) After performing all the steps of Apriori we can find out the Best rules



```
Size of set of large itemsets L(4): 10

Large Itemsets L(4):
exista=TRUE existb=TRUE existc=FALSE existd=TRUE 2
exista=TRUE existb=TRUE existc=FALSE existe=FALSE 2
exista=TRUE existb=TRUE existc=TRUE existe=TRUE 2
exista=TRUE existb=TRUE existc=TRUE existk=FALSE 2
exista=TRUE existb=TRUE existd=TRUE existe=FALSE 2
exista=TRUE existb=TRUE existe=TRUE existk=FALSE 2
exista=TRUE existc=FALSE existd=TRUE existe=FALSE 2
exista=TRUE existc=TRUE existe=TRUE existk=FALSE 2
existb=TRUE existc=FALSE existd=TRUE existe=FALSE 2
existb=TRUE existc=TRUE existe=TRUE existk=FALSE 2

Size of set of large itemsets L(5): 2

Large Itemsets L(5):
exista=TRUE existb=TRUE existc=FALSE existd=TRUE existe=FALSE 2
exista=TRUE existb=TRUE existc=TRUE existe=TRUE existk=FALSE 2

Best rules found:

 1. existb=TRUE 4 ==> exista=TRUE 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 2. exista=TRUE 4 ==> existb=TRUE 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 3. existd=TRUE 3 ==> exista=TRUE 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 4. existd=TRUE 3 ==> existb=TRUE 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 5. existb=TRUE existd=TRUE 3 ==> exista=TRUE 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 6. exista=TRUE existd=TRUE 3 ==> existb=TRUE 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 7. existd=TRUE 3 ==> exista=TRUE existb=TRUE 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
```

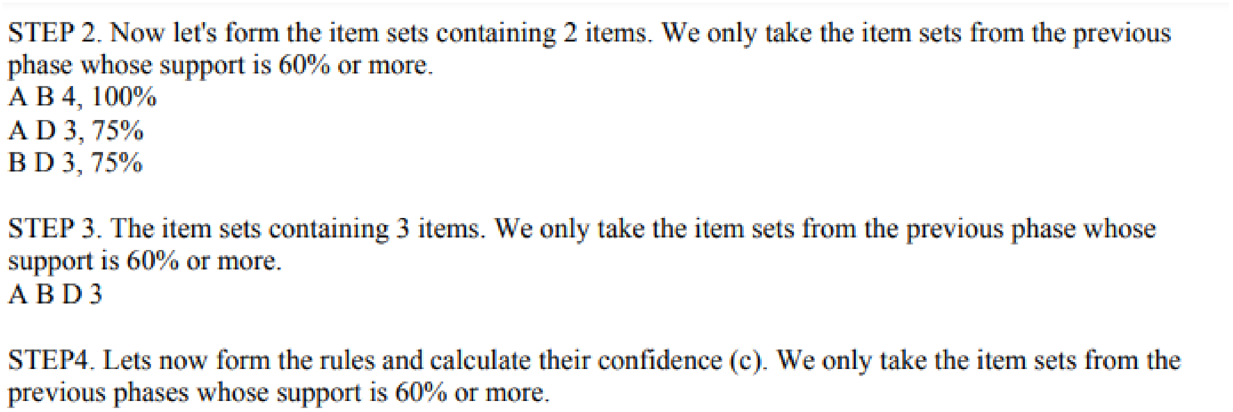
The solution:

Let’s first make a tabular and binary representation of the data:

Transaction	A	B	C	D	E	K
T1	1	1	0	1	0	1
T2	1	1	1	1	1	0
T3	1	1	1	0	1	0
T4	1	1	0	1	0	0

STEP 1. Form the item sets. Let's start by forming the item set containing one item. The number of occurrences and the support of each item set is given after it. In order to reach a minimum support of 60%, the item has to occur in at least 3 transactions.

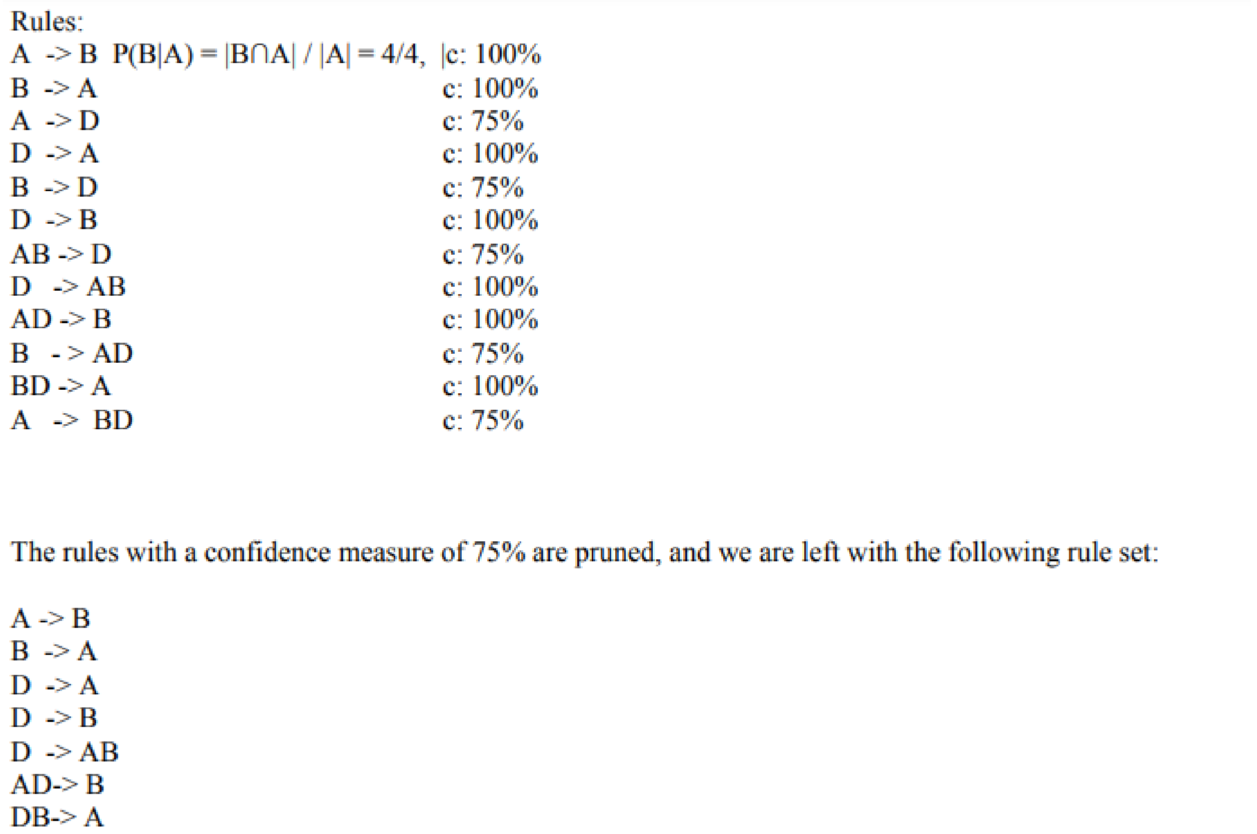
A	4, 100%
B	4, 100%
C	2, 50%
D	3, 75%
E	2, 50%
K	1, 25%



STEP 2. Now let's form the item sets containing 2 items. We only take the item sets from the previous phase whose support is 60% or more.
A B 4, 100%
A D 3, 75%
B D 3, 75%

STEP 3. The item sets containing 3 items. We only take the item sets from the previous phase whose support is 60% or more.
A B D 3

STEP4. Lets now form the rules and calculate their confidence (c). We only take the item sets from the previous phases whose support is 60% or more.



Rules:

A -> B P(B|A) = |B∩A| / |A| = 4/4,	|c: 100%
B -> A	c: 100%
A -> D	c: 75%
D -> A	c: 100%
B -> D	c: 75%
D -> B	c: 100%
AB -> D	c: 75%
D -> AB	c: 100%
AD -> B	c: 100%
B - > AD	c: 75%
BD -> A	c: 100%
A -> BD	c: 75%

The rules with a confidence measure of 75% are pruned, and we are left with the following rule set:

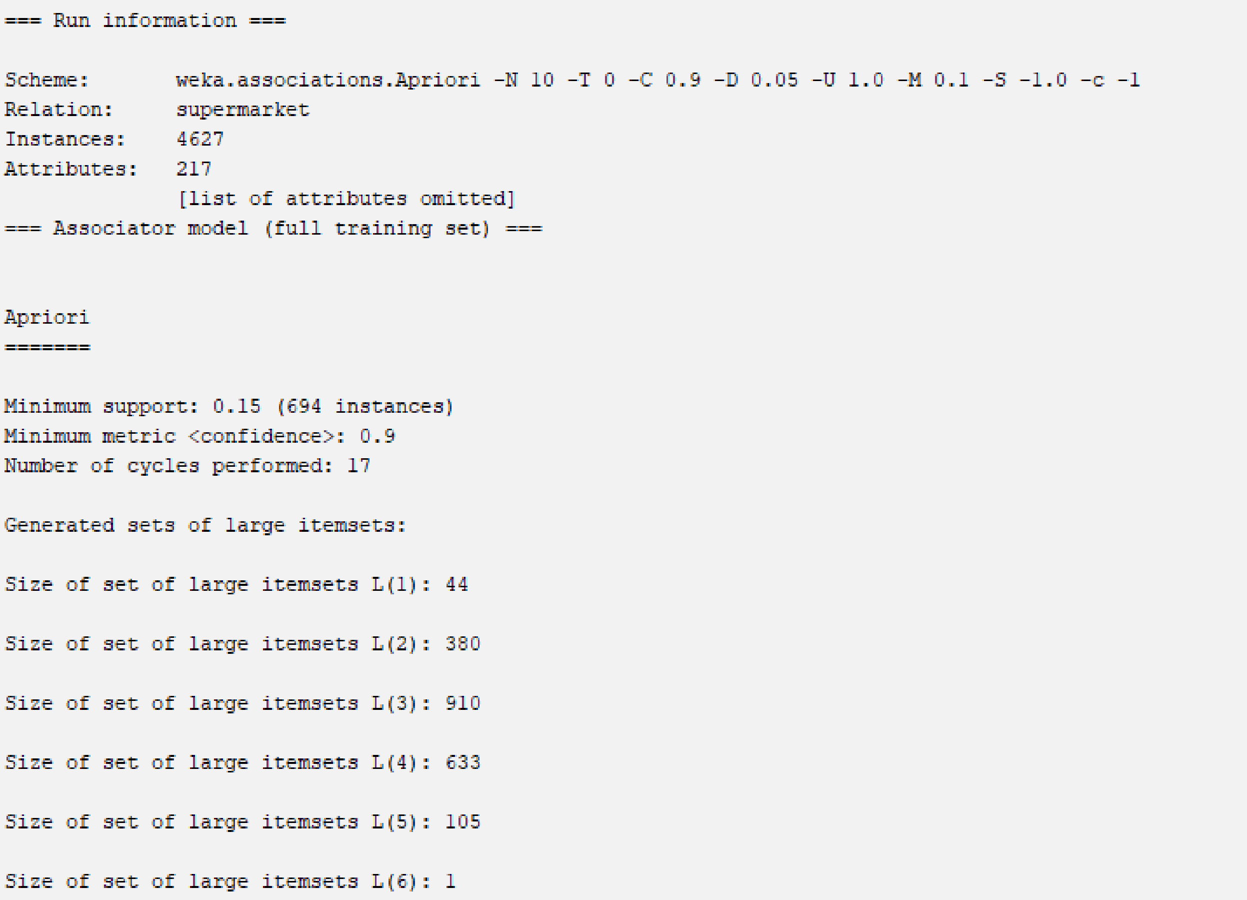
A -> B
B -> A
D -> A
D -> B
D -> AB
AD-> B
DB-> A

Conclusion: We can see that the best rules found from the manual solution and from WEKA are the same. Therefore we can conclude that both are solutions are correct and Apriori has been performed.

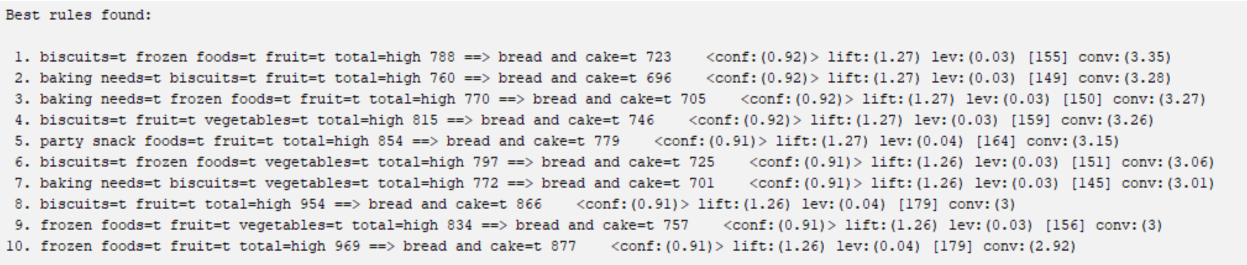
Supermarket.arff

The Apriori Algorithm was run for an inbuilt dataset called supermarket.arff

Case1: The minimum support is 0.15 and confidence is 0.9



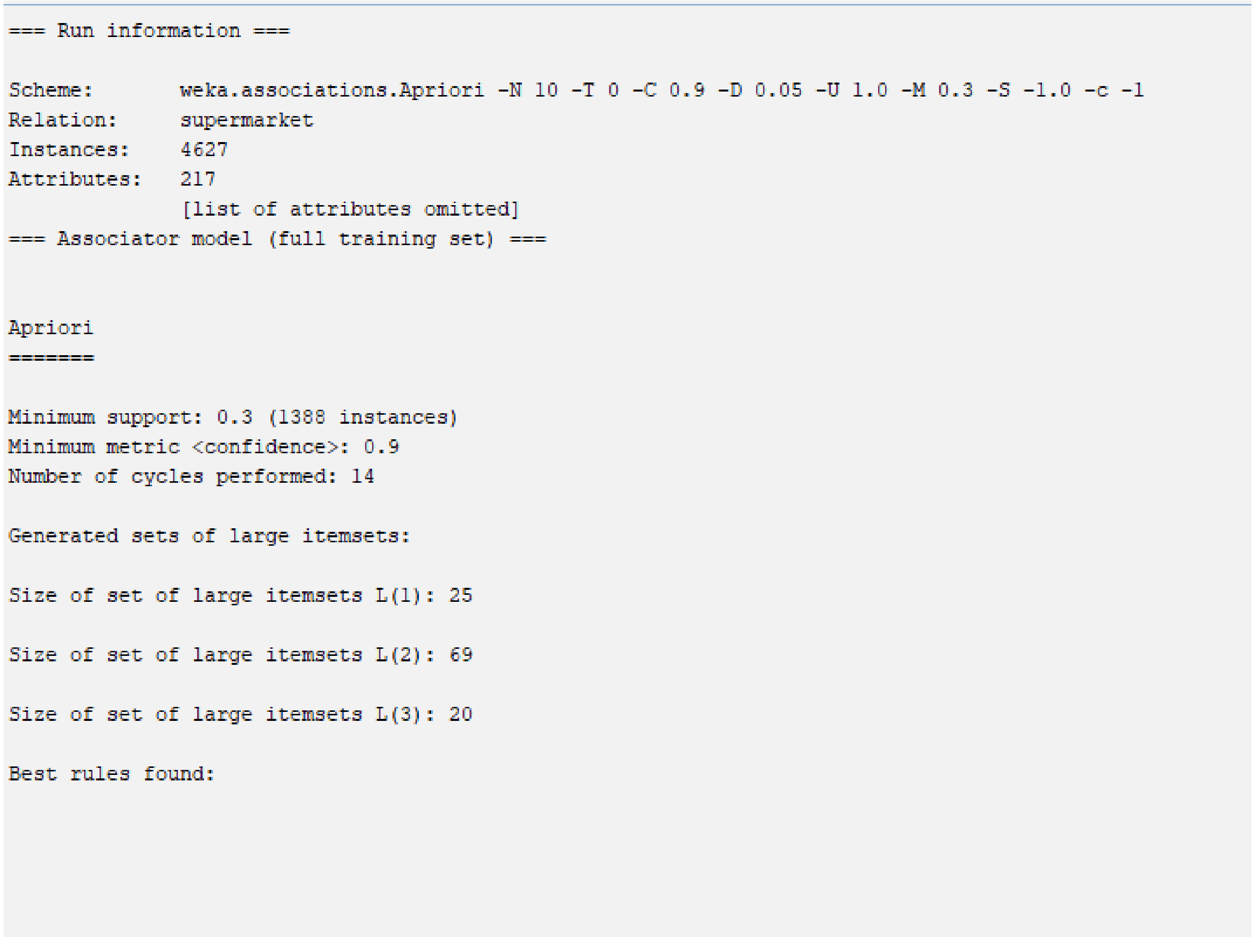
```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     supermarket
Instances:    4627
Attributes:   217
              [list of attributes omitted]
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

Size of set of large itemsets L(6): 1
```



```
Best rules found:

 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)
 5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779    <conf:(0.91)> lift:(1.27) lev:(0.04) [164] conv:(3.15)
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(0.03) [151] conv:(3.06)
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(0.03) [145] conv:(3.01)
 8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(3)
 9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757    <conf:(0.91)> lift:(1.26) lev:(0.03) [156] conv:(3)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(2.92)
```

We can see that in this case 10 rules are generated all with the confidence of 0.9 or higher

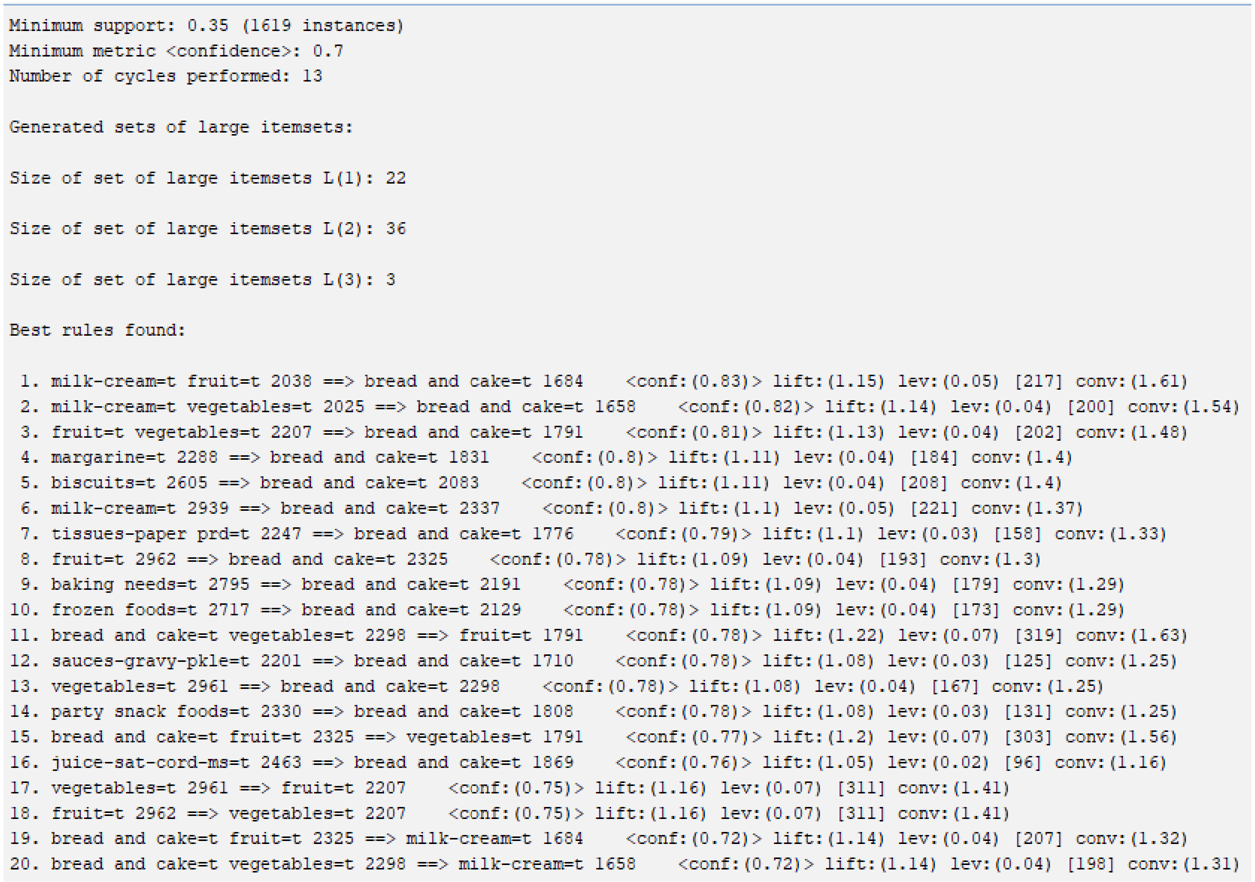
Case 2: The minimum support is 0.3 and confidence is 0.9.



```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.3 -S -1.0 -c -1
Relation:     supermarket
Instances:    4627
Attributes:   217
              [list of attributes omitted]
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.3 (1388 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 14

Generated sets of large itemsets:

Size of set of large itemsets L(1): 25

Size of set of large itemsets L(2): 69

Size of set of large itemsets L(3): 20

Best rules found:
```

In this case we can see that no rule is generated because the minimum support is high.

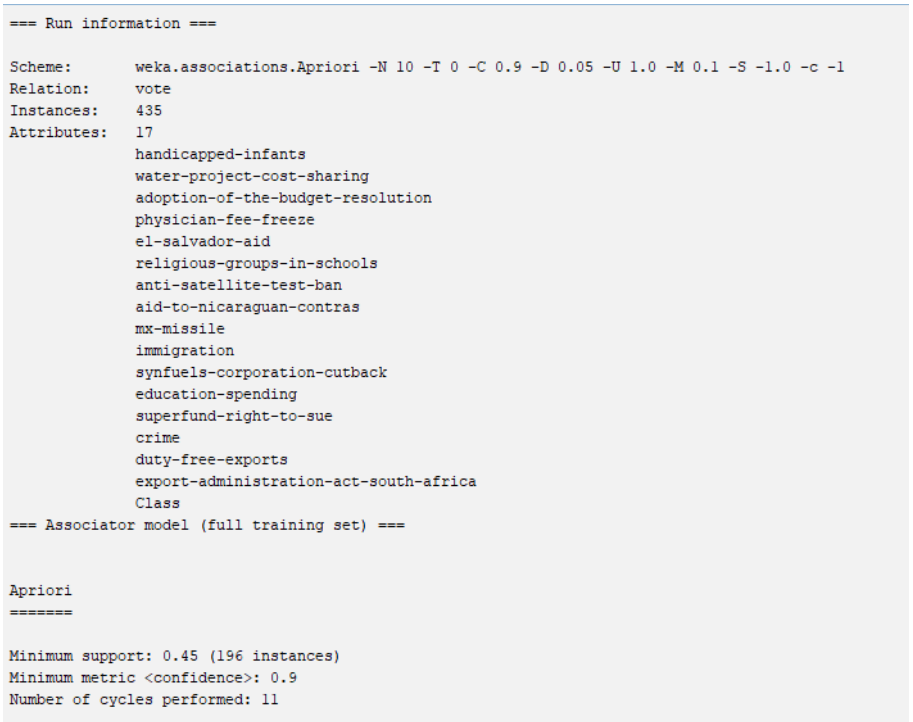
Case 3: The minimum support is 0.3 and confidence is 0.7.



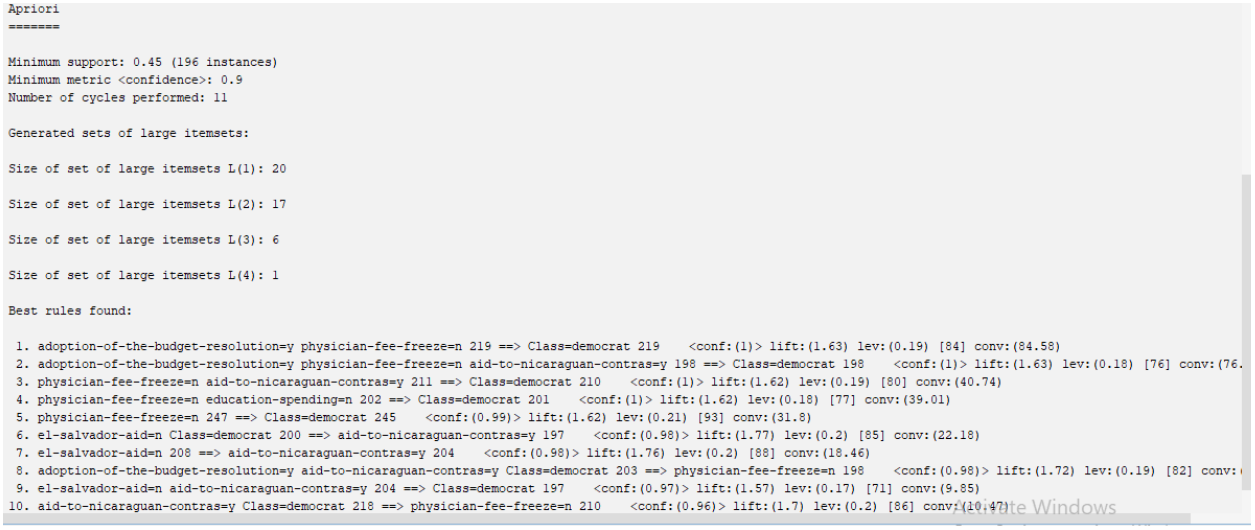
```
Minimum support: 0.35 (1619 instances)
Minimum metric <confidence>: 0.7
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 22

Size of set of large itemsets L(2): 36

Size of set of large itemsets L(3): 3

Best rules found:

 1. milk-cream=t fruit=t 2038 ==> bread and cake=t 1684    <conf:(0.83)> lift:(1.15) lev:(0.05) [217] conv:(1.61)
 2. milk-cream=t vegetables=t 2025 ==> bread and cake=t 1658    <conf:(0.82)> lift:(1.14) lev:(0.04) [200] conv:(1.54)
 3. fruit=t vegetables=t 2207 ==> bread and cake=t 1791    <conf:(0.81)> lift:(1.13) lev:(0.04) [202] conv:(1.48)
 4. margarine=t 2288 ==> bread and cake=t 1831    <conf:(0.8)> lift:(1.11) lev:(0.04) [184] conv:(1.4)
 5. biscuits=t 2605 ==> bread and cake=t 2083    <conf:(0.8)> lift:(1.11) lev:(0.04) [208] conv:(1.4)
 6. milk-cream=t 2939 ==> bread and cake=t 2337    <conf:(0.8)> lift:(1.1) lev:(0.05) [221] conv:(1.37)
 7. tissues-paper prd=t 2247 ==> bread and cake=t 1776    <conf:(0.79)> lift:(1.1) lev:(0.03) [158] conv:(1.33)
 8. fruit=t 2962 ==> bread and cake=t 2325    <conf:(0.78)> lift:(1.09) lev:(0.04) [193] conv:(1.3)
 9. baking needs=t 2795 ==> bread and cake=t 2191    <conf:(0.78)> lift:(1.09) lev:(0.04) [179] conv:(1.29)
10. frozen foods=t 2717 ==> bread and cake=t 2129    <conf:(0.78)> lift:(1.09) lev:(0.04) [173] conv:(1.29)
11. bread and cake=t vegetables=t 2298 ==> fruit=t 1791    <conf:(0.78)> lift:(1.22) lev:(0.07) [319] conv:(1.63)
12. sauces-gravy-pkle=t 2201 ==> bread and cake=t 1710    <conf:(0.78)> lift:(1.08) lev:(0.03) [125] conv:(1.25)
13. vegetables=t 2961 ==> bread and cake=t 2298    <conf:(0.78)> lift:(1.08) lev:(0.04) [167] conv:(1.25)
14. party snack foods=t 2330 ==> bread and cake=t 1808    <conf:(0.78)> lift:(1.08) lev:(0.03) [131] conv:(1.25)
15. bread and cake=t fruit=t 2325 ==> vegetables=t 1791    <conf:(0.77)> lift:(1.2) lev:(0.07) [303] conv:(1.56)
16. juice-sat-cord-ms=t 2463 ==> bread and cake=t 1869    <conf:(0.76)> lift:(1.05) lev:(0.02) [96] conv:(1.16)
17. vegetables=t 2961 ==> fruit=t 2207    <conf:(0.75)> lift:(1.16) lev:(0.07) [311] conv:(1.41)
18. fruit=t 2962 ==> vegetables=t 2207    <conf:(0.75)> lift:(1.16) lev:(0.07) [311] conv:(1.41)
19. bread and cake=t fruit=t 2325 ==> milk-cream=t 1684    <conf:(0.72)> lift:(1.14) lev:(0.04) [207] conv:(1.32)
20. bread and cake=t vegetables=t 2298 ==> milk-cream=t 1658    <conf:(0.72)> lift:(1.14) lev:(0.04) [198] conv:(1.31)
```

In this case we can see that because the confidence is lowered there are 20 rules generated.

The Apriori algorithm was also run on another inbuilt Vote.arff



```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     vote
Instances:    435
Attributes:   17
              handicapped-infants
              water-project-cost-sharing
              adoption-of-the-budget-resolution
              physician-fee-freeze
              el-salvador-aid
              religious-groups-in-schools
              anti-satellite-test-ban
              aid-to-nicaraguan-contras
              mx-missile
              immigration
              synfuels-corporation-cutback
              education-spending
              superfund-right-to-sue
              crime
              duty-free-exports
              export-administration-act-south-africa
              Class
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.45 (196 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 11
```



```
Apriori
=======

Minimum support: 0.45 (196 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 11

Generated sets of large itemsets:

Size of set of large itemsets L(1): 20

Size of set of large itemsets L(2): 17

Size of set of large itemsets L(3): 6

Size of set of large itemsets L(4): 1

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:(84.58)
 2. adoption-of-the-budget-resolution=y physician-fee-freeze=n aid-to-nicaraguan-contras=y 198 ==> Class=democrat 198    <conf:(1)> lift:(1.63) lev:(0.18) [76] conv:(76.
 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:(1.72) lev:(0.19) [82] conv:
 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)
```

The minimum support was 0.45 and confidence was 0.9 and 10 rules were generated. Here all the classes that are associated are the democrats. If we increase the republic party entries in our dataset, we shall see some republican class associated rules as well.

Conclusion

The Apriori algorithm has been performed on WEKA. With the help of parameters like support and confidence we have been able to find the association rules between different variables. By using apriori algorithm we can find essential relationships between different quantities which aids in effective decision making like for example we can decide which grocery items to place together in a market depending on which are being bought together so as to increase sales.