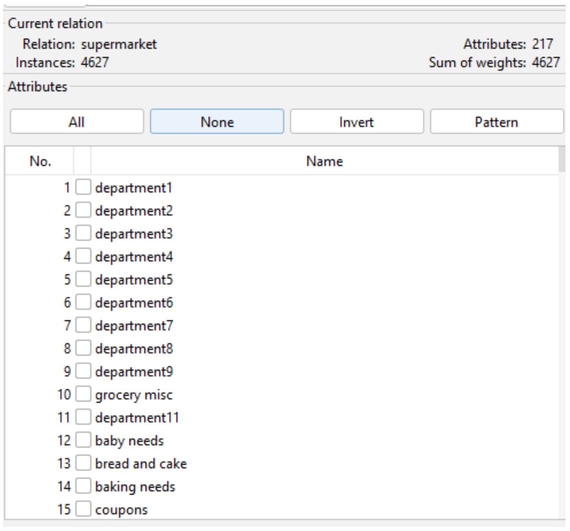
E/17/153 : Part 4 – Association rule learning

1.There are 4627 instances and 217 attributes.



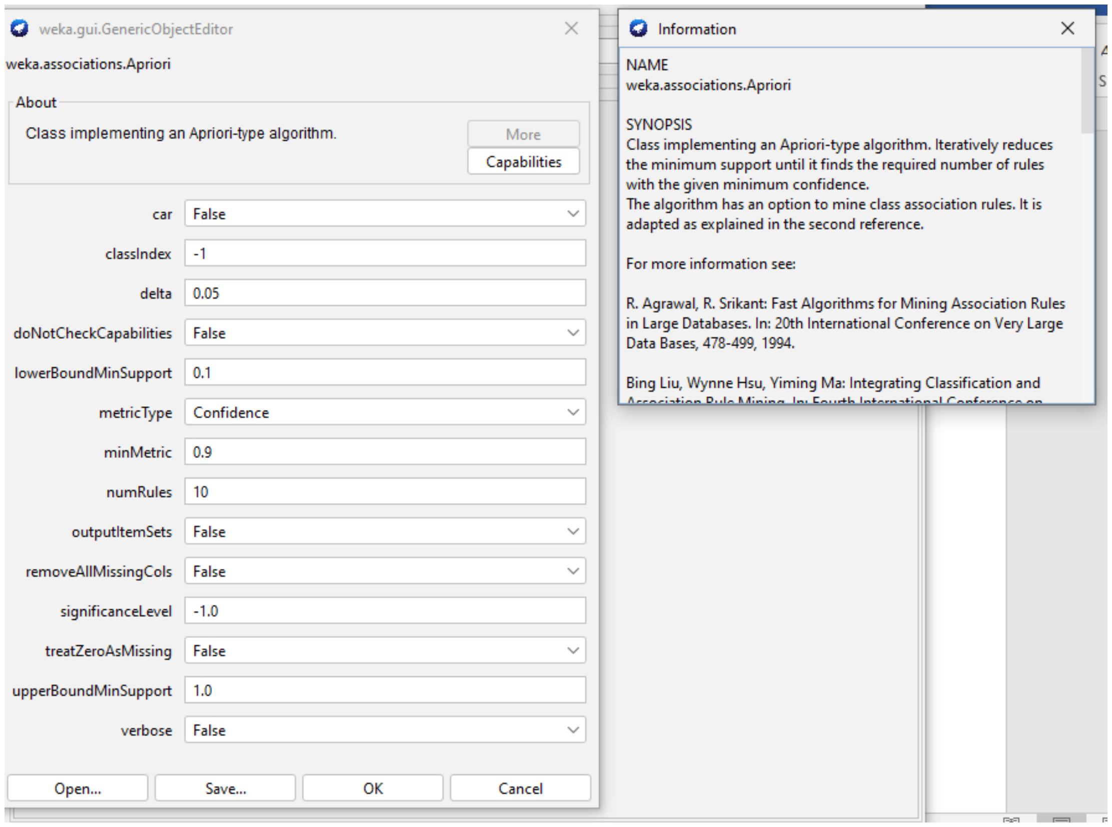
Current relation
Relation: supermarket
Instances: 4627
Attributes: 217
Sum of weights: 4627

Attributes

All | None | Invert | Pattern

No.	Name
1	department1
2	department2
3	department3
4	department4
5	department5
6	department6
7	department7
8	department8
9	department9
10	grocery misc
11	department11
12	baby needs
13	bread and cake
14	baking needs
15	coupons

2.



weka.gui.GenericObjectEditor

weka.associations.Apriori

About

Class implementing an Apriori-type algorithm.

More | Capabilities

car	False
classIndex	-1
delta	0.05
doNotCheckCapabilities	False
lowerBoundMinSupport	0.1
metricType	Confidence
minMetric	0.9
numRules	10
outputItemSets	False
removeAllMissingCols	False
significanceLevel	-1.0
treatZeroAsMissing	False
upperBoundMinSupport	1.0
verbose	False

Open... | Save... | OK | Cancel

Information

NAME
weka.associations.Apriori

SYNOPSIS
Class implementing an Apriori-type algorithm. Iteratively reduces the minimum support until it finds the required number of rules with the given minimum confidence.
The algorithm has an option to mine class association rules. It is adapted as explained in the second reference.

For more information see:

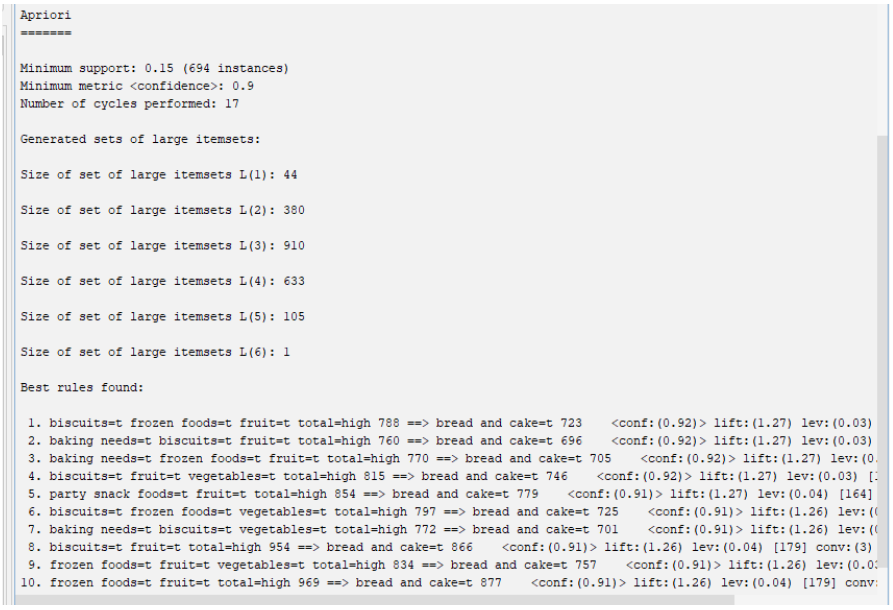
R. Agrawal, R. Srikant: Fast Algorithms for Mining Association Rules in Large Databases. In: 20th International Conference on Very Large Data Bases, 478-499, 1994.

Bing Liu, Wynne Hsu, Yiming Ma: Integrating Classification and

3.

- lowerBoundMinSupport : Lower bound for minimum support
- upperBoundMinSupport : Upper bound for minimum support. Start iteratively decreasing minimum support from this value.
- delta : Iteratively decrease support by this factor. Reduces support until min support is reached or required number of rules has been generated
- numRules : Number of rules to find.
- Confidence metricType : Metric type set the type of metric by which to rank rules. Confidence is the proportion of the examples covered by the premise that are also covered by the consequence (Class association rules can only be mined using confidence)

4.



```
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)
 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)
 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.
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf:(0.92)> lift:(1.27) lev:(0.03) [
 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]
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(
 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:(0.0
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:
```

Algorithm learned and presented 10 rules from the dataset.

Rules are presented in antecedent => consequent format. The number associated with the antecedent is the absolute coverage in the dataset (in this case a number out of a possible total of 4,627). The number next to the consequent is the absolute number of instances that match the antecedent and the consequent. The number in brackets on the end is the support for the rule (number of antecedents divided by the number of matching consequents).

As an example, let's examine the first rule:

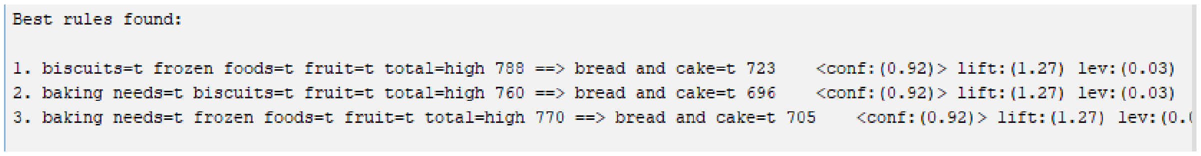
```
1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723    <conf:(0.92)> lift:(1.27) lev:(0.03)
```

It says that when biscuits, frozen foods, and fruits are bought together and the total price is high, it is also very likely that bread and cake are purchased as well. The "biscuits, frozen foods, fruit, total=high" item set appears in 778 transactions, while the "bread, cake" item set appears in 723 transactions. The confidence of this rule is 0.92, meaning that the rule holds true in 92% of transactions where the “biscuits, frozen foods, fruit, total high” itemset is present.

The output also shows several measures as lift, leverage, and conviction, which estimate the accuracy against our initial assumptions, for example, the 3.35 conviction value indicates that the rule would be incorrect 3.35 times as often if the association was purely a random chance. Lift measures the number of times X and Y occur together than expected if they were statistically independent (lift=1).

5.

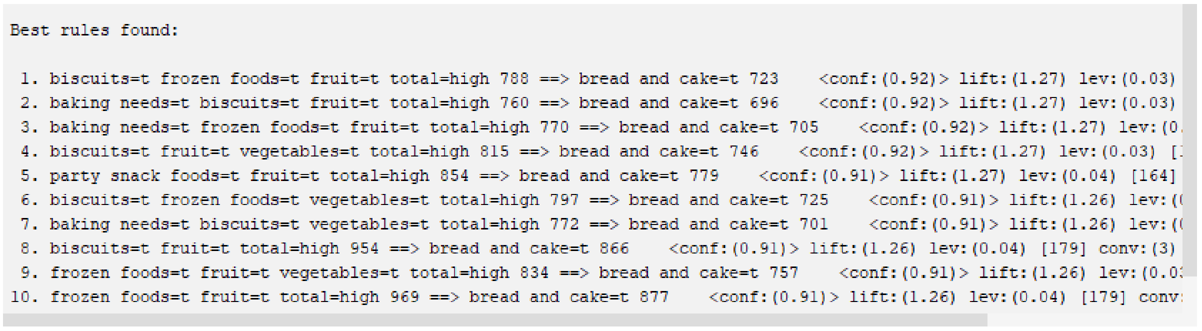
Number of rules = 3



```
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)
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)
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.
```

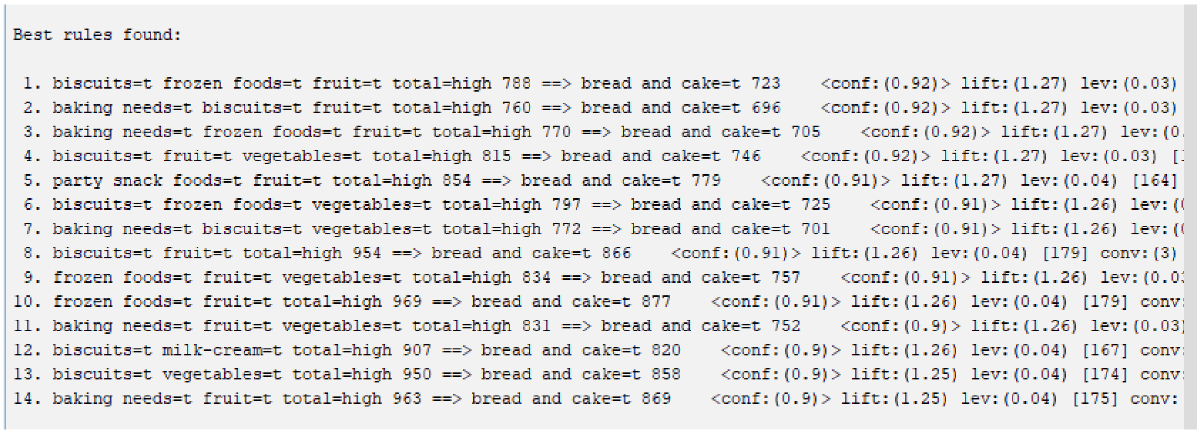
Number of rules = 10



```
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)
 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)
 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.
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf:(0.92)> lift:(1.27) lev:(0.03) [
 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]
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(
 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:(0.0
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:
```

Number of rules = 14



```
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)
 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)
 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.
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf:(0.92)> lift:(1.27) lev:(0.03) [
 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]
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(
 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:(0.0
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:
11. baking needs=t fruit=t vegetables=t total=high 831 ==> bread and cake=t 752    <conf:(0.9)> lift:(1.26) lev:(0.03)
12. biscuits=t milk-cream=t total=high 907 ==> bread and cake=t 820    <conf:(0.9)> lift:(1.26) lev:(0.04) [167] conv:
13. biscuits=t vegetables=t total=high 950 ==> bread and cake=t 858    <conf:(0.9)> lift:(1.25) lev:(0.04) [174] conv:
14. baking needs=t fruit=t total=high 963 ==> bread and cake=t 869    <conf:(0.9)> lift:(1.25) lev:(0.04) [175] conv:
```

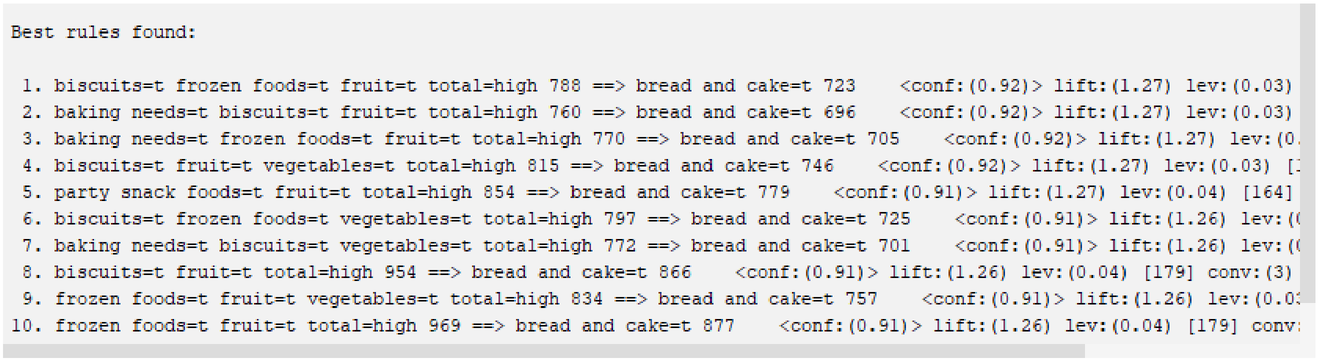
Even the number of rules increased from 3 to 10 and 10 to 14, the first three rules of 10 rule output is same as 3 rule output. Also, first ten rules of 14 rule output is same as the 10 rule output.

By further examining it is observed that the confidence is decreasing in later rules when increasing the number of rules.

6. The supermarket dataset describes the usual shopping habits of supermarket customers. There is one instance per customer. Most of the attributes stand for a particular item group like dairy foods, beef, potatoes and some are for department like department 79, department 81, and so on. The value is ‘t’ if the customer had bought an item and missing otherwise.

Apriori algorithm is a classic algorithm used for frequent pattern mining and association rule learning over transactional. By identifying the frequent individual items in a database and extending them to larger item sets, Apriori can determine the association rules, which highlight general trends about a dataset.

As an example, let’s take the first 10 rules of supermarket dataset.



```
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)
 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)
 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.
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf:(0.92)> lift:(1.27) lev:(0.03) [
 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]
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(
 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:(0.0
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
```

Here we can see that bread and cake are normally go with biscuits, frozen foods, fruits and baking needs. So, if the super market can’t give discounts and increase the selling of bread and cake without lowering their profit, they can give discounts or perform more marketing to other goods that are associated with bread and cake instead. As the rules convey these products do drive the sales of bread and cake.

We can take such indirect decisions by analyzing these rules with the supermarket data set.