Experiment No 6

Aim: Perform data Pre-processing task and Demonstrate performing Classification, Clustering, and Association algorithms on data sets using a data mining tool (WEKA/R tool)

Theory:

Classification:

The term "classification" is usually used when there are exactly two target classes called binary classification. When more than two classes may be predicted, specifically in pattern recognition problems, this is often referred to as multinomial classification. However, multinomial classification is also used for categorical response data, where one wants to predict which category amongst several categories has the instances with the highest probability.

Classification is one of the most important tasks in data mining. It refers to a process of assigning pre-defined class labels to instances based on their attributes. There is a similarity between classification and clustering, it looks similar, but it is different. The major difference between classification and clustering is that classification includes the levelling of items according to their membership in pre-defined groups. Let's understand this concept with the help of an example; suppose you are using a self-organizing map neural network algorithm for image recognition where there are 10 different kinds of objects. If you label each image with one of these 10 classes, the classification task is solved.

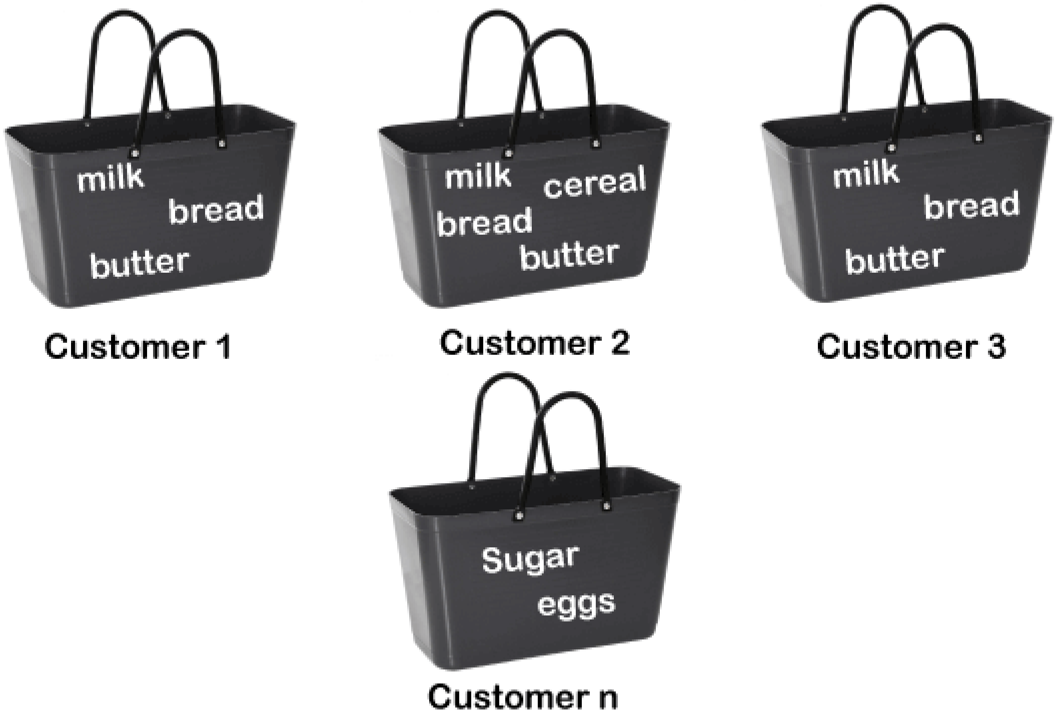
Clustering:

Clustering refers to a technique of grouping objects so that objects with the same functionalities come together and objects with different functionalities go apart. In other words, we can say that clustering is a process of portioning a data set into a set of meaningful subclasses, known as clusters. Clustering is the same as classification in which data is grouped. However, unlike classification, the groups are not previously defined. Instead, the grouping is achieved by determining similarities between data according to characteristics found in the real data. The groups are called Clusters.

Association Rules:

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that

it can be more profitable. It tries to find some interesting relations or associations among the variables of the dataset. It is based on different rules to discover the interesting relations between variables in the database.
Association rule learning is one of the very important concepts of machine learning, and it is employed in **Market Basket analysis, Web usage mining, continuous production, etc.** Here market basket analysis is a technique used by the various big retailers to discover the associations between items. We can understand it by taking the example of a supermarket, as in a supermarket, all products that are purchased together are put together.



Association rule learning can be divided into three types of algorithms:

1. **Apriori**
2. **Eclat**
3. **F-P Growth Algorithm**

Support

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as:

$$\text{Supp(X)=}\frac{Freq(X)}{T}$$

Confidence

Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence=}\frac{Freq(X,Y)}{Freq(X)}$$

Lift

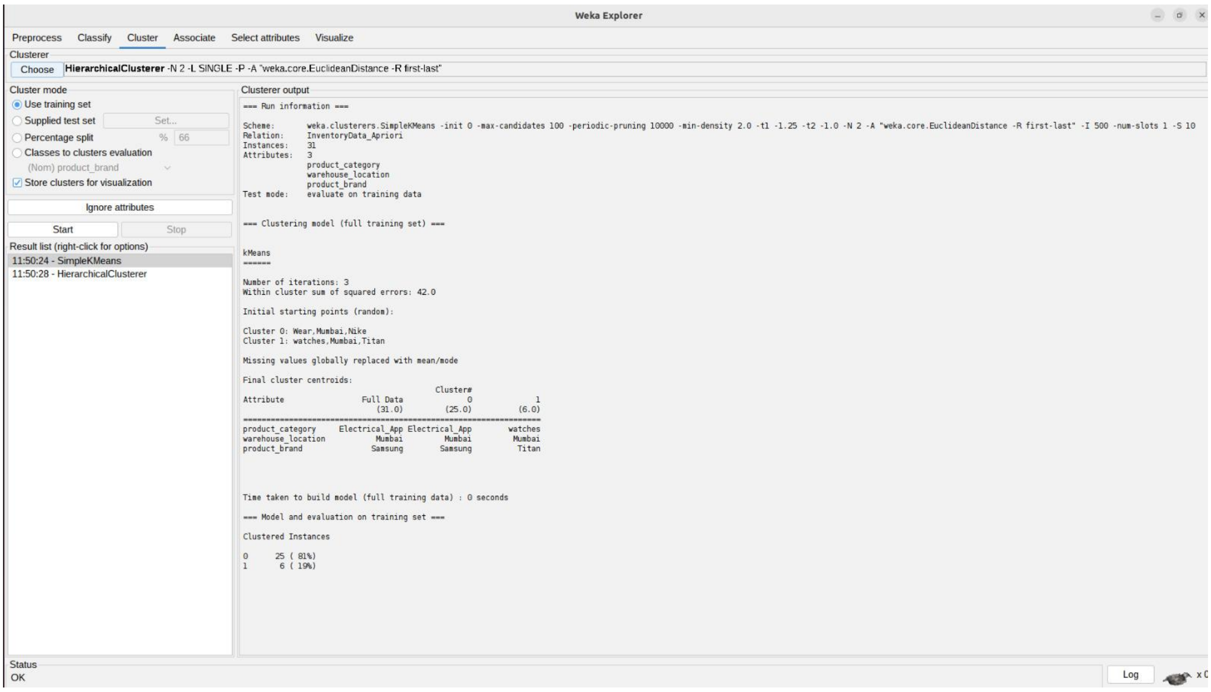
It is the strength of any rule, which can be defined as below formula:

$$\text{Lift=}\frac{Supp(X,Y)}{Supp(X)\times Supp(Y)}$$

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

- If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.
- **Lift>1**: It determines the degree to which the two itemsets are dependent to each other.
- **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

Output:



Clusterer

HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"

Cluster mode

● Use training set

○ Supplied test set

○ Percentage split % 66

○ Classes to clusters evaluation

(Nom) product_brand

☑ Store clusters for visualization

Result list (right-click for options)

11:50:24 - SimpleKMeans

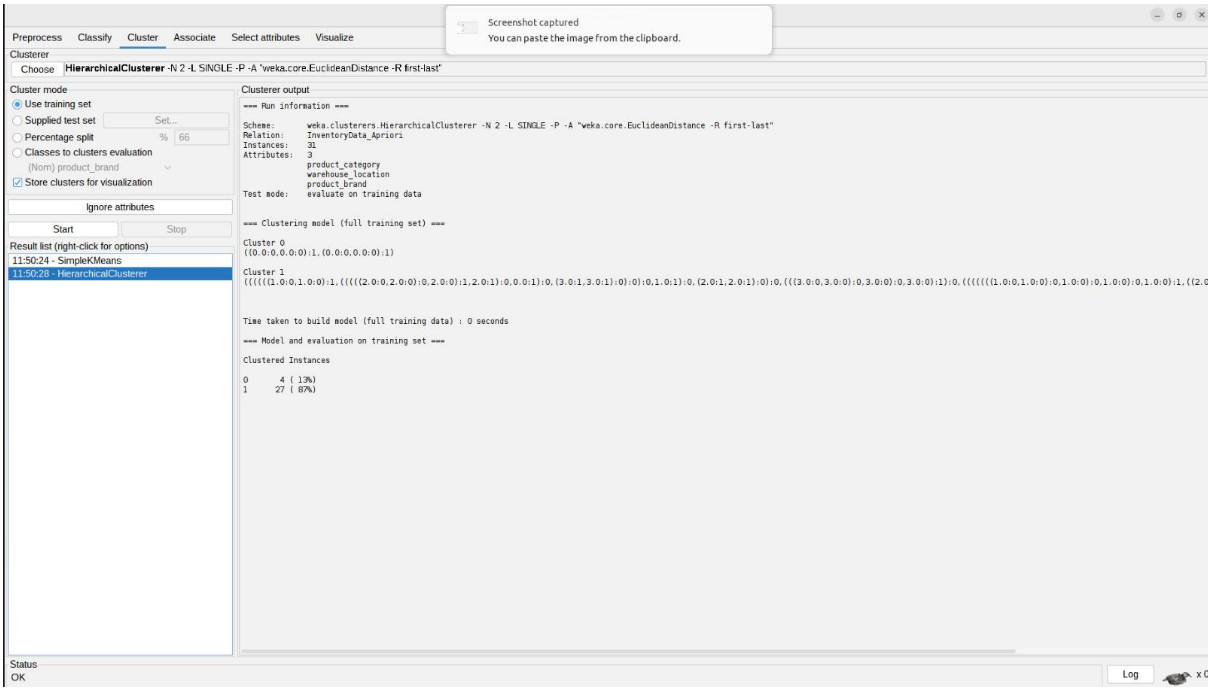
11:50:28 - HierarchicalClusterer

Status

OK

Clusterer output

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Relation:     InventoryData_Apriori
Instances:    31
Attributes:   3
              product_category
              warehouse_location
              product_brand
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 42.0

Initial starting points (random):

Cluster 0: Wear,Mumbai,Nike
Cluster 1: watches,Mumbai,Titan

Missing values globally replaced with mean/mode

Final cluster centroids:
                                       Cluster#
Attribute                Full Data            0            1
                            (31.0)       (25.0)        (6.0)
=============================================================
product_category     Electrical_App Electrical_App      watches
warehouse_location          Mumbai       Mumbai       Mumbai
product_brand              Samsung      Samsung        Titan




Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      25 ( 81%)
1       6 ( 19%)
```



Clusterer

HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"

Cluster mode

● Use training set

○ Supplied test set

○ Percentage split % 66

○ Classes to clusters evaluation

(Nom) product_brand

☑ Store clusters for visualization

Result list (right-click for options)

11:50:24 - SimpleKMeans

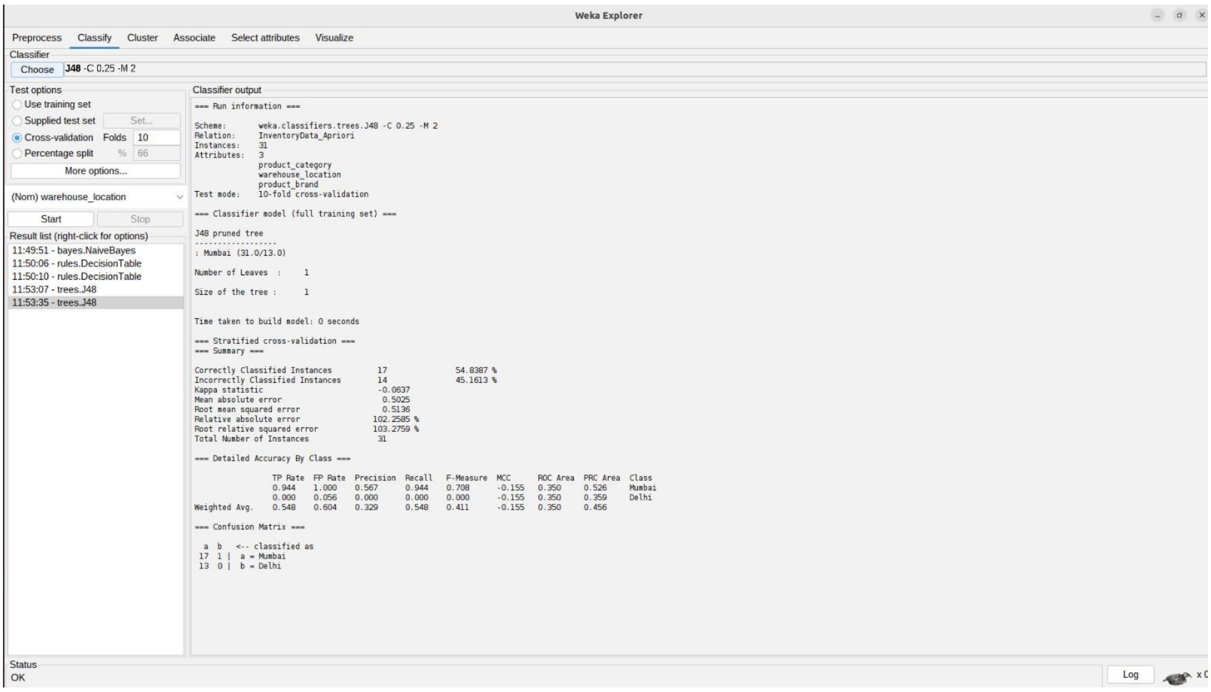
11:50:28 - HierarchicalClusterer

Status

OK

Clusterer output

```
=== Run information ===

Scheme:       weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation:     InventoryData_Apriori
Instances:    31
Attributes:   3
              product_category
              warehouse_location
              product_brand
Test mode:    evaluate on training data


=== Clustering model (full training set) ===

Cluster 0
((0.0:0,0.0:0):1,(0.0:0,0.0:0):1)

Cluster 1
(((((1.0:0,1.0:0):1,(((((2.0:0,2.0:0):0,2.0:0):1,2.0:1):0,0.0:1):0,(3.0:1,3.0:1):0):0):0,1.0:1):0,(2.0:1,2.0:1):0):0,(((3.0:0,3.0:0):0,3.0:0):0,3.0:0):1):0,((((((1.0:0,1.0:0):0,1.0:0):0,1.0:0):0,1.0:0):1,((2.0



Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       4 ( 13%)
1      27 ( 87%)
```



Classifier

Choose J48 -C 0.25 -M 2

Test options

○ Use training set
○ Supplied test set Set...
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) warehouse_location

Result list (right-click for options)

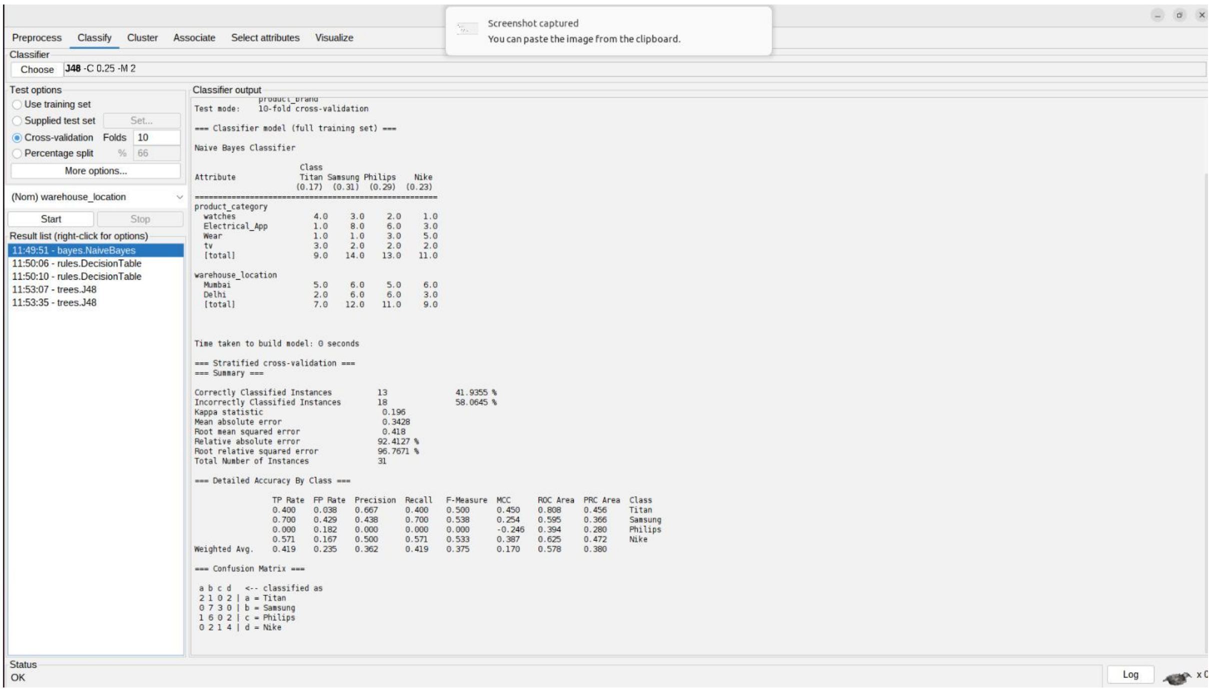
11:49:51 - bayes.NaiveBayes
11:50:06 - rules.DecisionTable
11:50:10 - rules.DecisionTable
11:53:07 - trees.J48
11:53:35 - trees.J48

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     InventoryData_Apriori
Instances:    31
Attributes:   3
              product_category
              warehouse_location
              product_brand
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
: Mumbai (31.0/13.0)

Number of Leaves  :     1

Size of the tree :      1


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          17               54.8387 %
Incorrectly Classified Instances        14               45.1613 %
Kappa statistic                         -0.0637
Mean absolute error                      0.5025
Root mean squared error                  0.5136
Relative absolute error                102.2585 %
Root relative squared error            103.2759 %
Total Number of Instances               31

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.944    1.000    0.567      0.944    0.708      -0.155   0.350     0.526     Mumbai
                 0.000    0.056    0.000      0.000    0.000      -0.155   0.350     0.359     Delhi
Weighted Avg.    0.548    0.604    0.329      0.548    0.411      -0.155   0.350     0.456

=== Confusion Matrix ===

  a  b   <-- classified as
 17  1 |  a = Mumbai
 13  0 |  b = Delhi
```

Status

OK



Screenshot captured

You can paste the image from the clipboard.

Classifier

Choose J48 -C 0.25 -M 2

Test options

○ Use training set
○ Supplied test set Set...
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) warehouse_location

Result list (right-click for options)

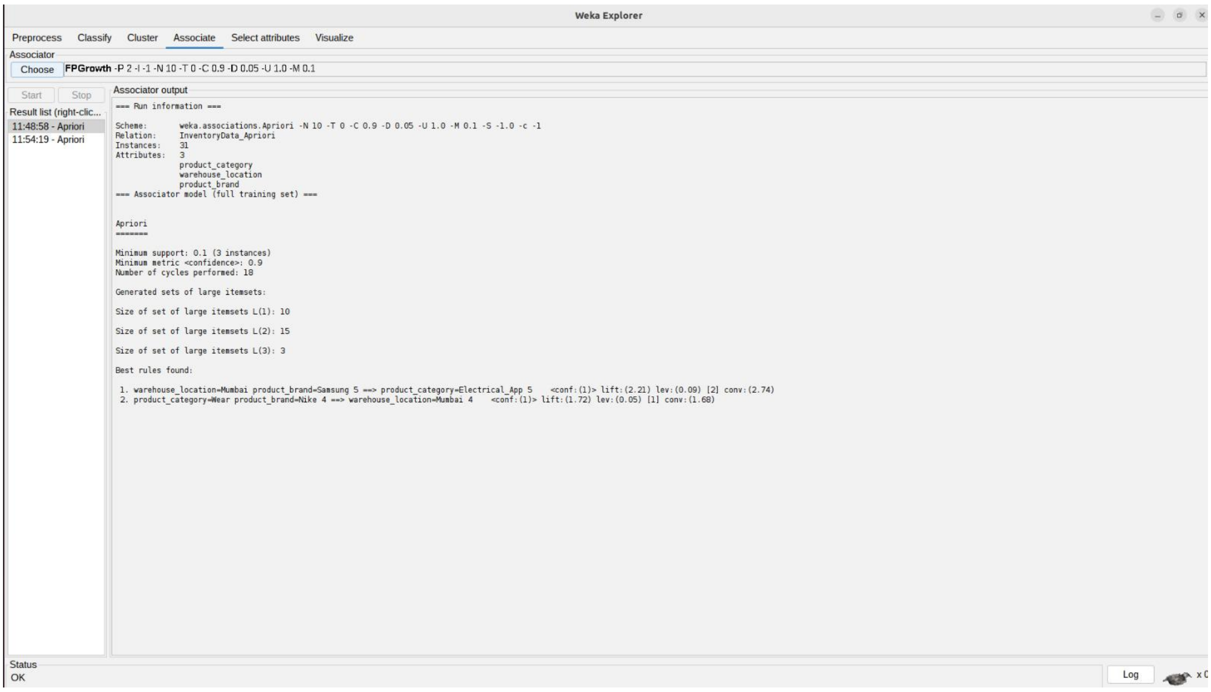
11:49:51 - bayes.NaiveBayes
11:50:06 - rules.DecisionTable
11:50:10 - rules.DecisionTable
11:53:07 - trees.J48
11:53:35 - trees.J48

Classifier output

```
              product_brand
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

                     Class
Attribute            Titan Samsung Philips    Nike
                    (0.17)  (0.31)  (0.29)  (0.23)
===================================================
product_category
  watches              4.0     3.0     2.0     1.0
  Electrical_App       1.0     8.0     6.0     3.0
  Wear                 1.0     1.0     3.0     5.0
  tv                   3.0     2.0     2.0     2.0
  [total]              9.0    14.0    13.0    11.0

warehouse_location
  Mumbai               5.0     6.0     5.0     6.0
  Delhi                2.0     6.0     6.0     3.0
  [total]              7.0    12.0    11.0     9.0



Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          13               41.9355 %
Incorrectly Classified Instances        18               58.0645 %
Kappa statistic                          0.196
Mean absolute error                      0.3428
Root mean squared error                  0.418
Relative absolute error                 92.4127 %
Root relative squared error             96.7671 %
Total Number of Instances               31

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.400    0.038    0.667      0.400    0.500      0.450    0.808     0.456     Titan
                 0.700    0.429    0.438      0.700    0.538      0.254    0.595     0.366     Samsung
                 0.000    0.182    0.000      0.000    0.000      -0.246   0.394     0.280     Philips
                 0.571    0.167    0.500      0.571    0.533      0.387    0.625     0.472     Nike
Weighted Avg.    0.419    0.235    0.362      0.419    0.375      0.170    0.578     0.380

=== Confusion Matrix ===

 a b c d   <-- classified as
 2 1 0 2 | a = Titan
 0 7 3 0 | b = Samsung
 1 6 0 2 | c = Philips
 0 2 1 4 | d = Nike
```

Status

OK



FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1

```
=== 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:     InventoryData_Apriori
Instances:    31
Attributes:   3
              product_category
              warehouse_location
              product_brand
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.1 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 10

Size of set of large itemsets L(2): 15

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

Best rules found:

 1. warehouse_location=Mumbai product_brand=Samsung 5 ==> product_category=Electrical_App 5    <conf:(1)> lift:(2.21) lev:(0.09) [2] conv:(2.74)
 2. product_category=Wear product_brand=Nike 4 ==> warehouse_location=Mumbai 4    <conf:(1)> lift:(1.72) lev:(0.05) [1] conv:(1.68)
```