Author Name : SRAVAN KUMAR VASAM
R & D Project : Machine Learning Association rule learning Apriori and Eclat algorithms
Technologies : R version 4.0.2, Rstudio, Linux
Year of submission : November, 2020

Data source : MarketBasketOptimisation/super market customers 7k observations 20 variables.

Aim: Analysis the summary of data, finding top N products purchased, visualisation of the item frequently, training the sets in two algorithms with minimum support and confidence.

important points to be noted in Association Rule Learning

1) In the market data transactions where customers buy similar products.People who bought also bought. Exp. Super market.
2) movie recommendation
3) support and confidence set in parameters
4) That is what Association Rule Learning will help us figure out!

Now I will show you the step by step output

Implementing the following Association Rule Learning models:

1) Apriori, library(arules)
2) Eclat, library(arules), function : eclat

1. Apriori : There are 5 steps
    1. set a minimum support and confidence
    2. take all the subsets in transactions having higher support than minimum support
    3. take all the rules of these subsets having higher confidence than minimum confidence
    4. sort the rules by decreasing lift

```
install.packages('arules')
library(arules)
dataset = read.csv('Market_Basket_Optimisation.csv', header = FALSE)
dataset = read.transactions('Market_Basket_Optimisation.csv', sep = ',', rm.duplicates = TRUE)
```

```
> library(arules)
> dataset = read.csv('Market_Basket_Optimisation.csv', header = FALS
E)
> dataset = read.transactions('Market_Basket_Optimisation.csv', sep
 = ',', rm.duplicates = TRUE)
distribution of transactions with duplicates:
1
5
```

```
summary(dataset)
```

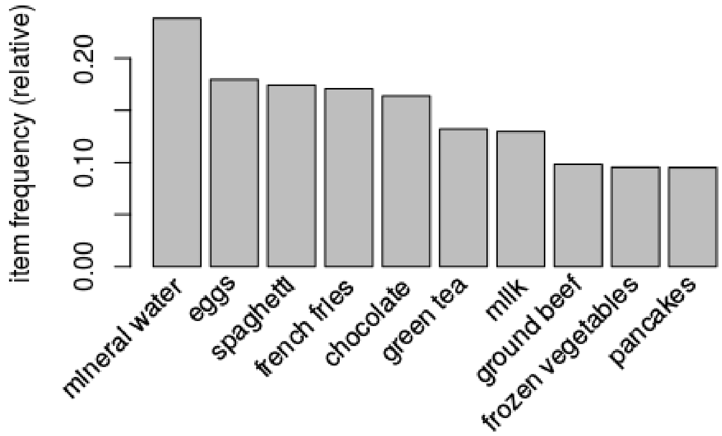
```
> summary(dataset)
transactions as itemMatrix in sparse format with
 7501 rows (elements/itemsets/transactions) and
 119 columns (items) and a density of 0.03288973

most frequent items:
mineral water          eggs     spaghetti  french fries
         1788          1348          1306          1282
    chocolate       (Other)
         1229         22405

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13
1754 1358 1044  816  667  493  391  324  259  139  102   67   40
  14   15   16   18   19   20
  22   17    4    1    2    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   3.914   5.000  20.000

includes extended item information - examples:
            labels
1          almonds
2 antioxydant juice
3        asparagus
```




```
# Training Apriori on the dataset
rules = apriori(data = dataset, parameter = list(support = 0.004, confidence = 0.2))
```

```
> itemFrequencyPlot(dataset, topN = 10)
> # Training Apriori on the dataset
> rules = apriori(data = dataset, parameter = list(support = 0.004,
 confidence = 0.2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime
        0.2    0.1    1 none FALSE            TRUE       5
 support minlen maxlen target   ext
   0.004      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 30

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].
sorting and recoding items ... [114 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.01s].
writing ... [811 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> 
```

```
# Visualising the results
inspect(sort(rules, by = 'lift')[1:10])
```

```
> # visualising the results
> inspect(sort(rules, by = 'lift')[1:10])
     lhs                        rhs              support confidenc
e   coverage     lift count
[1]  {light cream}           => {chicken}       0.004532729  0.290598
3 0.01559792 4.843951    34
[2]  {pasta}                 => {escalope}      0.005865885  0.372881
4 0.01573124 4.700812    44
[3]  {pasta}                 => {shrimp}        0.005065991  0.322033
9 0.01573124 4.506672    38
[4]  {eggs,

      ground beef}           => {herb & pepper} 0.004132782  0.206666
7 0.01999733 4.178455    31
[5]  {whole wheat pasta}     => {olive oil}     0.007998933  0.271493
2 0.02946274 4.122410    60
[6]  {herb & pepper,

      spaghetti}             => {ground beef}   0.006399147  0.393442
6 0.01626450 4.004360    48
[7]  {herb & pepper,

      mineral water}         => {ground beef}   0.006665778  0.390625
0 0.01706439 3.975683    50
[8]  {tomato sauce}          => {ground beef}   0.005332622  0.377358
5 0.01413145 3.840659    40
[9]  {mushroom cream sauce} => {escalope}      0.005732569  0.300699
3 0.01906412 3.790833    43
[10] {frozen vegetables,

      mineral water,

      spaghetti}             => {ground beef}   0.004399413  0.366666
7 0.01199840 3.731841    33
>
```

2. Eclat
    1. set a minimum support
    2. take all the subsets in transactions having higher support than minimum support
    3. sort these subsets by decreasing support
    4. support is only set in parameters. Most frequently used model.

```
# Training Eclat on the dataset
rules = eclat(data = dataset, parameter = list(support = 0.003, minlen = 2)
```

```
> itemFrequencyPlot(dataset, topN = 10)
> rules = eclat(data = dataset, parameter = list(support = 0.003, minlen = 2))
Eclat

parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE   0.003      2     10 frequent itemsets TRUE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

Absolute minimum support count: 22

create itemset ...
set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].
sorting and recoding items ... [115 item(s)] done [0.00s].
creating sparse bit matrix ... [115 row(s), 7501 column(s)] done [0.00s].
writing  ... [1328 set(s)] done [0.02s].
Creating S4 object  ... done [0.00s].
>
```

```
# Visualising the results
```

```
inspect(sort(rules, by = 'support')[1:10])
```

```
> # Visualising the results
> inspect(sort(rules, by = 'support')[1:10])
     items                             support    transIdenticalToItemsets
[1]  {mineral water,spaghetti}         0.05972537 448
[2]  {chocolate,mineral water}         0.05265965 395
[3]  {eggs,mineral water}              0.05092654 382
[4]  {milk,mineral water}              0.04799360 360
[5]  {ground beef,mineral water}       0.04092788 307
[6]  {ground beef,spaghetti}           0.03919477 294
[7]  {chocolate,spaghetti}             0.03919477 294
[8]  {eggs,spaghetti}                  0.03652846 274
[9]  {eggs,french fries}               0.03639515 273
[10] {frozen vegetables,mineral water} 0.03572857 268
     count
[1]  448
[2]  395
[3]  382
[4]  360
[5]  307
[6]  294
[7]  294
[8]  274
[9]  273
[10] 268
>
```