PRACTICAL – 9

Aim: Implementation of Apriori Algorithm.

Theory:

Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemset in a dataset for Boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemset are used to find k+1 itemset.

To improve the efficiency of level-wise generation of frequent itemset, an important property is used called *Apriori property* which helps by reducing the search space.

Apriori Property –

All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that

All subsets of a frequent itemset must be frequent (Apriori property).If an itemset is infrequent, all its supersets will be infrequent.

Components of Apriori algorithm

The given three components comprise the Apriori algorithm.

1. Support
2. Confidence

Support

Support refers to the default popularity of any product. You find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions.

Confidence

Confidence refers to the possibility that the customers bought both biscuits and chocolates together. So, you need to divide the number of transactions that comprise both biscuits and chocolates by the total number of transactions to get the confidence.

Setting working directory.

```
> getwd()
[1] "C:/Users/Suraj/Documents"
>
```

Applying Apriori on sample file named “aprioriData.csv”

```
> setwd("C:/Users/Suraj/Downloads")
> suraj_data <- read.csv("data_apriori.csv")
> trans <- split(suraj_data$Products, suraj_data$Customer_Id,"transactions")
> head(trans)
$`1`
[1] "bread"  "butter" "eggs"   "milk"

$`2`
[1] "beer"   "bread"  "cheese" "chips"  "mayo"   "soda"

$`3`
[1] "bread"   "butter"  "eggs"    "milk"    "oranges"

$`4`
[1] "bread"  "butter" "eggs"   "milk"   "soda"

$`5`
[1] "buns"    "chips"   "beer"    "mustard" "pickels" "soda"

$`6`
[1] "bread"     "butter"    "chocolate" "eggs"      "milk"
```

Loading arules library

```
> install.packages("arules")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate v
sion of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into ‘C:/Users/TARUN/AppData/Local/R/win-library/4.2’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/arules_1.7-5.zip'
Content type 'application/zip' length 2125545 bytes (2.0 MB)
downloaded 2.0 MB

package ‘arules’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\TARUN\AppData\Local\Temp\Rtmpkj8BzN\downloaded_packages
>
```

```
> library(arules)
Loading required package: Matrix

Attaching package: 'arules'

The following objects are masked from 'package:base':

    abbreviate, write

> rules = apriori(trans, parameter=list(support=0.5, confidence=0.9,maxlen=3,minlen=2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.9    0.1    1 none FALSE            TRUE       5     0.5      2      3  rules TRUE

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

Absolute minimum support count: 7

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[15 item(s), 15 transaction(s)] done [0.00s].
sorting and recoding items ... [4 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [11 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
> inspect(rules)
     lhs                rhs      support   confidence coverage  lift     count
[1]  {eggs}          => {milk}   0.6000000 1          0.6000000 1.666667 9
[2]  {milk}          => {eggs}   0.6000000 1          0.6000000 1.666667 9
[3]  {butter}        => {bread}  0.6000000 1          0.6000000 1.250000 9
[4]  {butter, eggs}  => {milk}   0.5333333 1          0.5333333 1.666667 8
[5]  {butter, milk}  => {eggs}   0.5333333 1          0.5333333 1.666667 8
[6]  {bread, eggs}   => {milk}   0.5333333 1          0.5333333 1.666667 8
[7]  {bread, milk}   => {eggs}   0.5333333 1          0.5333333 1.666667 8
[8]  {butter, eggs}  => {bread}  0.5333333 1          0.5333333 1.250000 8
[9]  {bread, eggs}   => {butter} 0.5333333 1          0.5333333 1.666667 8
[10] {butter, milk}  => {bread}  0.5333333 1          0.5333333 1.250000 8
[11] {bread, milk}   => {butter} 0.5333333 1          0.5333333 1.666667 8
>
```

Conclusion:

I have successfully implemented and learned Apriori Algorithm.