AIM: IMPLEMENT APRIORI ALGOEITHM

THEORY:

Apriori Algorithm:

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal and R Srikant and came to be known as Apriori. This algorithm uses two steps "join" and "prune" to reduce the search space. It is an iterative approach to discover the most frequent itemsets.

Apriori says:

The probability that item I is not frequent is if:

- P(I) < minimum support threshold, then I is not frequent.
- P (I+A) < minimum support threshold, then I+A is not frequent, where A also belongs to itemset.
- If an itemset set has value less than minimum support then all of its supersets will also fall below min support, and thus can be ignored. This property is called the Antimonotone property.

The steps followed in the Apriori Algorithm of data mining are:

1. **Join Step** :- This step generates (K+1) itemset from K-itemsets by joining each item with itself.
2. **Prune Step** :- This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.

   Support: Support is the rate of the frequency of an item appears in the total number of items. Like the frequency of burger among all the transactions. Mathematically, for an item A will be given as –

$$\boxed{\text{Support (A)} = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}}$$

   Confidence: Confidence is the conditional probability of occurrence of a consequent (then) providing the occurrence of an antecedent (if). It's kind of testing a rule. Like if a customer buys a burger(antecedent), he is supposed to buy french fries(consequent). Mathematically, the confidence of **B** given **A** will be given as-

$$\boxed{\text{Confidence } (A \longrightarrow B) = \frac{\text{Support}(AUB)}{\text{Support}(A)}}$$

Advantages:

- Easy to understand algorithm.
- Join and Prune steps are easy to implement on large itemsets in large databases

Disadvantages:

- It requires high computation if the itemsets are very large and the minimum support is kept very low.
- The entire database needs to be scanned.

Applications Of Apriori Algorithm :-

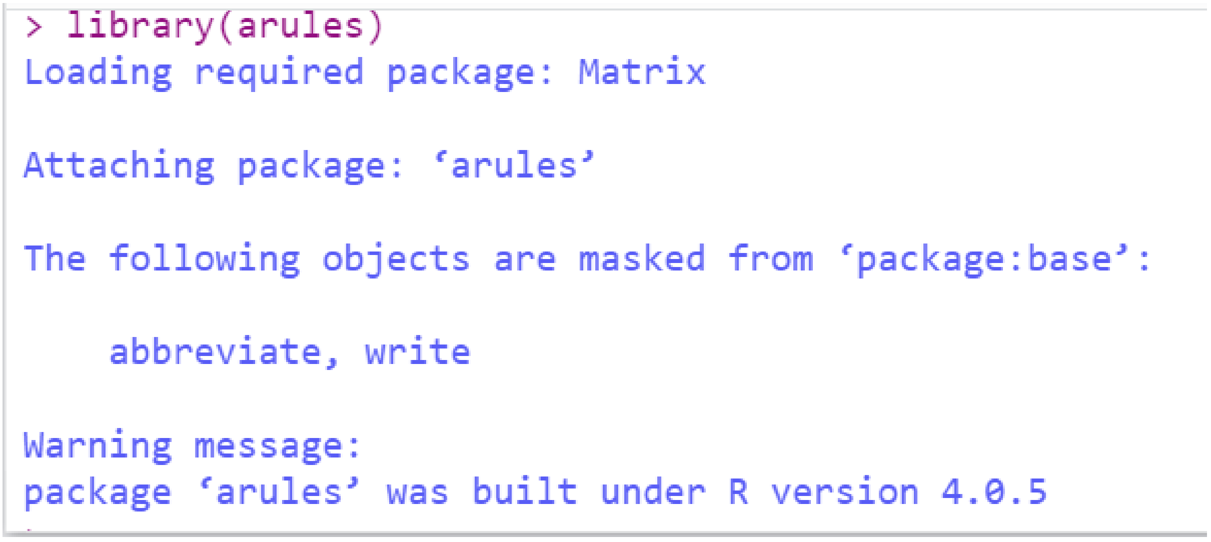
Some fields where Apriori is used:

- **In Education Field:** Extracting association rules in data mining of admitted students through characteristics and specialties.
- **In the Medical field:** For example Analysis of the patient’s database.
- **In Forestry:** Analysis of probability and intensity of forest fire with the forest fire data.
- Apriori is used by many companies like Amazon in the Recommender System and by Google for the auto-complete feature.

1) **INSTALL & LOAD LIBRARY:**

install.packages("arules")

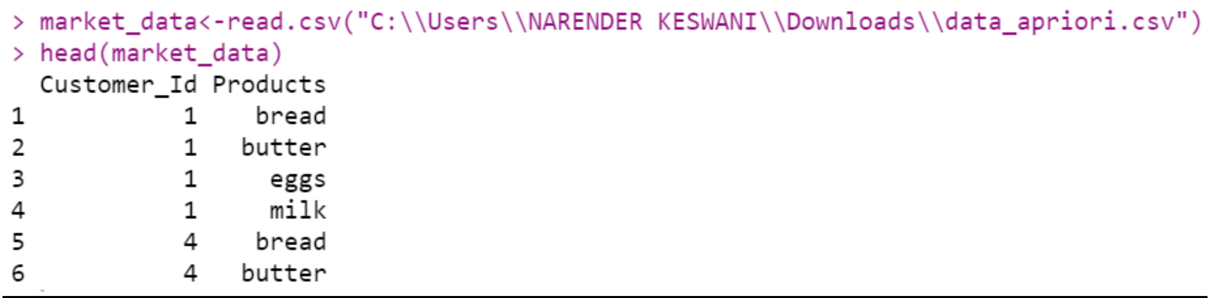
library(arules)



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

Attaching package: ‘arules’

The following objects are masked from ‘package:base’:

    abbreviate, write

Warning message:
package ‘arules’ was built under R version 4.0.5
```

2) **LOAD DATASET & PRINT SOME RECORDS:**

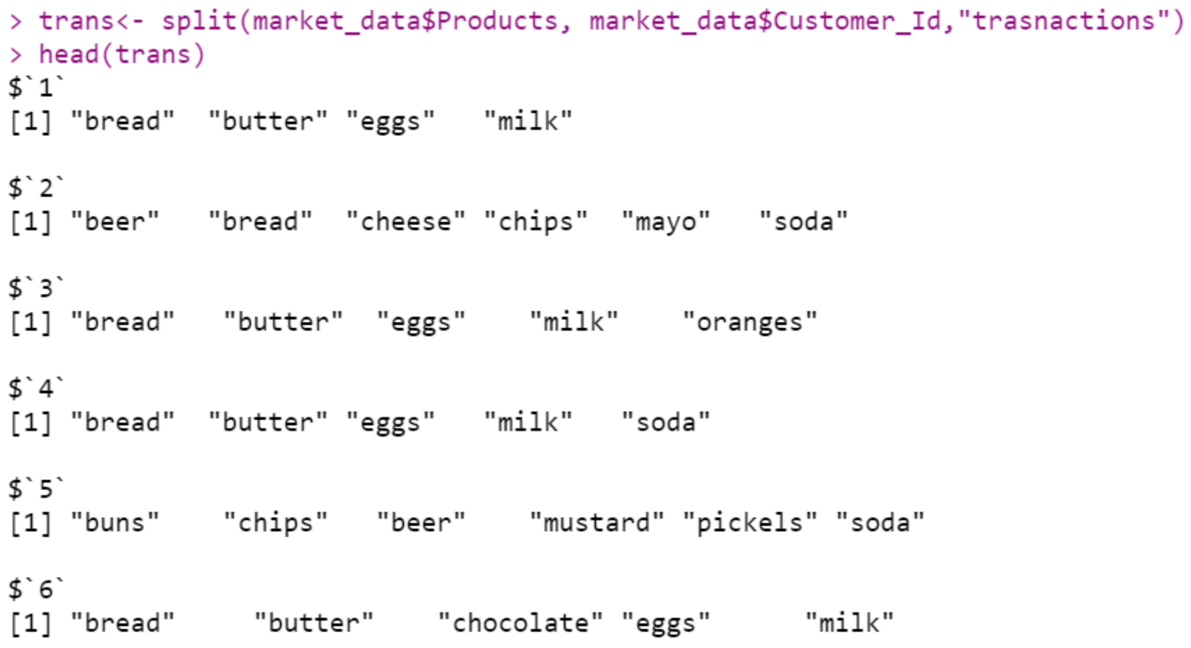
market_data<-read.csv("C:\\Users\\NARENDER KESWANI\\Downloads\\data_apriori.csv")
head(market_data)



```
> market_data<-read.csv("C:\\Users\\NARENDER KESWANI\\Downloads\\data_apriori.csv")
> head(market_data)
  Customer_Id Products
1           1    bread
2           1   butter
3           1     eggs
4           1     milk
5           4    bread
6           4   butter
```

3) DIVIDING DATA INTO GROUPS:

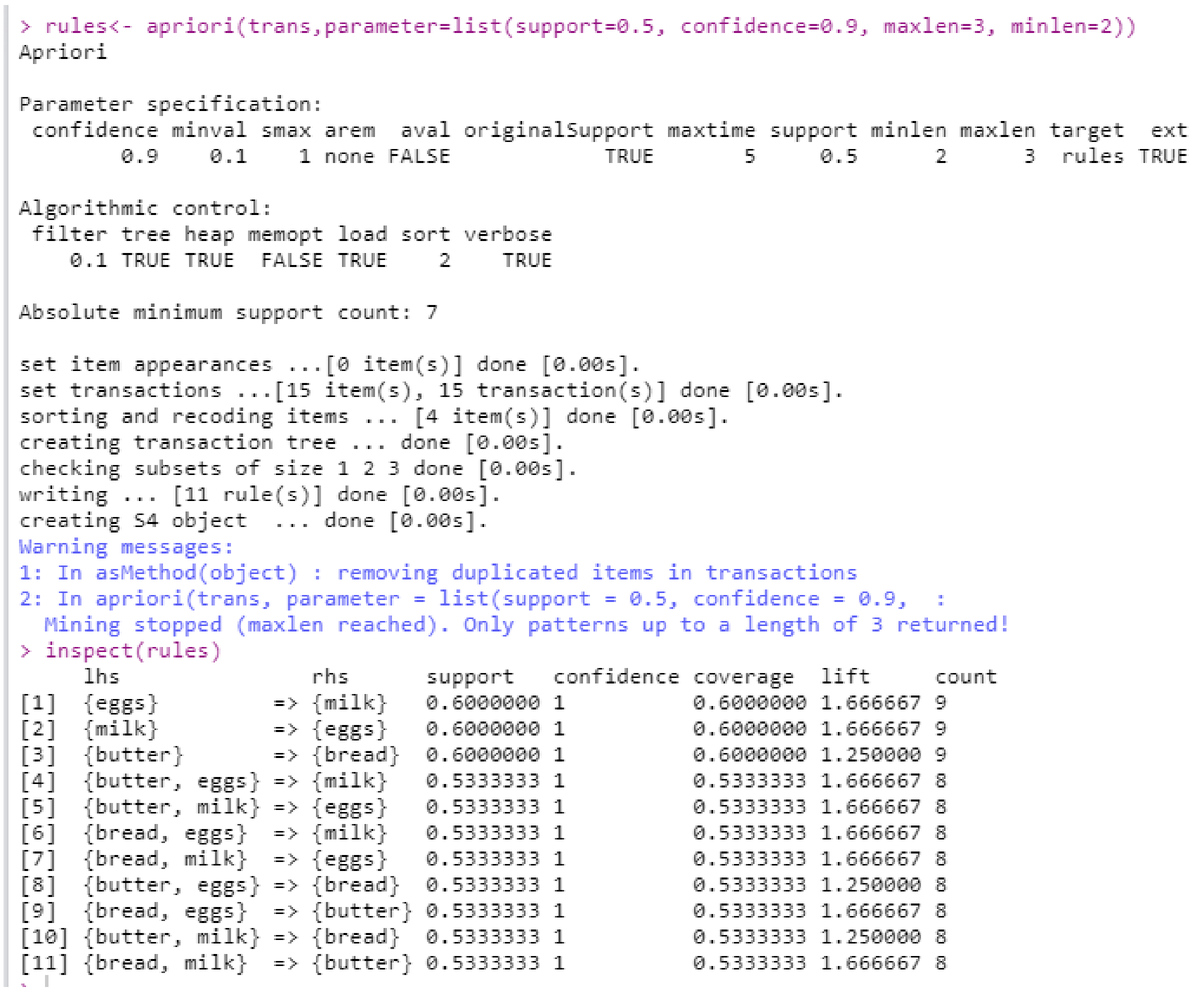
```
trans<- split(market_data$Products, market_data$Customer_Id,"trasnactions")
head(trans)
```



```
> trans<- split(market_data$Products, market_data$Customer_Id,"trasnactions")
> 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"
```

4) APPLYING RULES ON DATASET

rules<- apriori(trans,parameter=list(support=0.5, confidence=0.9, maxlen=3, minlen=2))
inspect(rules)



```
> 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].
Warning messages:
1: In asMethod(object) : removing duplicated items in transactions
2: In apriori(trans, parameter = list(support = 0.5, confidence = 0.9,  :
  Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!
> 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:

From this practical, I have learned how to implement the apriori algorithm in R.