EXPERIMENT NO. 6
Apriori Algorithm

NAME: Rishabh Patil
SAP: 60009200056
BATCH: D12

AIM: Implement program for Apriori Algorithm.

THEORY:

We take an example to understand the concept better. You must have noticed that the Pizza shop seller makes a pizza, soft drink, and breadstick combo together. He also offers a discount to their customers who buy these combos. Do you ever think why does he do so? He thinks that customers who buy pizza also buy soft drinks and breadsticks. However, by making combos, he makes it easy for the customers. At the same time, he also increases his sales performance.

Similarly, you go to Big Bazar, and you will find biscuits, chips, and Chocolate bundled together. It shows that the shopkeeper makes it comfortable for the customers to buy these products in the same place.

The above two examples are the best examples of Association Rules in Data Mining. It helps us to learn the concept of apriori algorithms.

Apriori algorithm refers to an algorithm that is used in mining frequent products sets and relevant association rules. Generally, the apriori algorithm operates on a database containing a huge number of transactions. For example, the items customers but at a Big Bazar.

Apriori algorithm helps the customers to buy their products with ease and increases the sales performance of the particular store.

Components of Apriori algorithm

The given three components comprise the apriori algorithm.

1. Support
2. Confidence
3. Lift

Let's take an example to understand this concept.

We have already discussed above; you need a huge database containing a large no of transactions. Suppose you have 4000 customers transactions in a Big Bazar. You have to calculate the Support, Confidence, and Lift for two products, and you may say Biscuits and Chocolate. This is because customers frequently buy these two items together.

Out of 4000 transactions, 400 contain Biscuits, whereas 600 contain Chocolate, and these 600 transactions include a 200 that includes Biscuits and chocolates. Using this data, we will find out the support, confidence, and lift.

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. Hence, we get

Support (Biscuits) = (Transactions relating biscuits) / (Total transactions)

= 400/4000 = 10 percent.

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.

Hence,

Confidence = (Transactions relating both biscuits and Chocolate) / (Total transactions involving Biscuits)

= 200/400

= 50 percent.

It means that 50 percent of customers who bought biscuits bought chocolates also.

Lift

Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.

Lift = (Confidence (Biscuits - chocolates)/ (Support (Biscuits)

= 50/10 = 5

It means that the probability of people buying both biscuits and chocolates together is five times more than that of purchasing the biscuits alone. If the lift value is below one, it requires that the people are unlikely to buy both the items together. Larger the value, the better is the combination.

CODE WITH OUTPUT:

+ Code

```
[ ] import pandas as pd
```

```
data1 = pd.read_excel('Apriori.xlsx')
data1
```

	Transactions	Items
0	t1	T-shirt,Trousers,Belt
1	t2	T-shirt,Jacket
2	t3	Jacket,Gloves
3	t4	T-shirt,Trousers,Jacket
4	t5	T-shirt,Trousers,Sneakers,Belt,Jacket
5	t6	Trousers,Sneakers,Belt
6	t7	Trousers,Sneakers,Belt

```
[ ] data2 = data1['Items'].str.get_dummies(sep = ',')
```

```
[ ] final_data = pd.concat([data1, data2], axis=1)
```

```
[ ] final_data
```

	Transactions	Items	Belt	Gloves	Jacket	Sneakers
0	t1	T-shirt,Trousers,Belt	1	0	0	0
1	t2	T-shirt,Jacket	0	0	1	0
2	t3	Jacket,Gloves	0	1	1	0
3	t4	T-shirt,Trousers,Jacket	0	0	1	0
4	t5	T-shirt,Trousers,Sneakers,Belt,Jacket	1	0	1	1
5	t6	Trousers,Sneakers,Belt	1	0	0	1
6	t7	Trousers,Sneakers,Belt	1	0	0	1

Consider Support >= 3

```
items = ["T-shirt","Trousers","Sneakers","Belt","Jacket","Gloves"]
items
```

```
['T-shirt', 'Trousers', 'Sneakers', 'Belt', 'Jacket', 'Gloves']
```

Singleton Sets

```
[ ]  singleton_frame = []
     for i in range(len(items)):
       ele = []
       ele.append(items[i])
       ele.append(final_data[items[i]].sum())
       singleton_frame.append(ele)
     singleton_df = pd.DataFrame(singleton_frame, columns=['Item', 'Support_count'])
```

```
singleton_df
```

	Item	Support_count
0	T-shirt	4
1	Trousers	5
2	Sneakers	3
3	Belt	4
4	Jacket	4
5	Gloves	1

```
[ ] singleton_df = singleton_df[singleton_df['Support_count'] >= 3]
    singleton_df
```

	Item	Support_count
0	T-shirt	4
1	Trousers	5
2	Sneakers	3
3	Belt	4
4	Jacket	4

Doubleton

```
doubleton_frame = []
#adding double ton sets
i = 0
j_start = 1
while(i < len(items)-1):
  j = j_start
  while(j<len(items)):
    count = 0
    ele = []
    ele.append(items[i] + ' ' + items[j])
    for k in range(len(final_data)):
      l = final_data.loc[k,"Items"].split(",")
      if items[i] in l:
        if items[j] in l:
          count = count + 1
    ele.append(count)
    doubleton_frame.append(ele)
    j = j+1
  i = i+1
  j_start = j_start+1
```

```
[ ] doubleton_df = pd.DataFrame(doubleton_frame, columns=['Item', 'Support_count'])
    doubleton_df
```

	Item	Support_count
0	T-shirt Trousers	3
1	T-shirt Sneakers	1
2	T-shirt Belt	2
3	T-shirt Jacket	3
4	T-shirt Gloves	0
5	Trousers Sneakers	3
6	Trousers Belt	4
7	Trousers Jacket	2
8	Trousers Gloves	0
9	Sneakers Belt	3
10	Sneakers Jacket	1
11	Sneakers Gloves	0
12	Belt Jacket	1
13	Belt Gloves	0
14	Jacket Gloves	1

```
[ ] doubleton_df = doubleton_df[doubleton_df['Support_count'] >= 3]
    doubleton_df.reset_index(drop=True,inplace=True)
    doubleton_df
```

	Item	Support_count
0	T-shirt Trousers	3
1	T-shirt Jacket	3
2	Trousers Sneakers	3
3	Trousers Belt	4
4	Sneakers Belt	3

TripleTon

```
tripleton_frame = []
#adding triple ton sets
i = 0
j_start = 1
k_start = 2
while(i<len(items)-2):
  j = j_start
  while(j < len(items) - 1):
    k = k_start
    while(k<len(items)):
      count = 0
      ele = []
      ele.append(items[i] + ' ' + items[j] + ' ' + items[k])
      for m in range(len(final_data)):
        l = final_data.loc[m,"Items"].split(",")
        if items[i] in l:
          if items[j] in l:
            if items[k] in l:
              count = count + 1
      ele.append(count)
      tripleton_frame.append(ele)
      k = k + 1
    j = j + 1
    k_start = j + 1
```

```
[ ]  i = i + 1
     j_start = i + 1
     k_start = j_start + 1
```

```
tripleton_df = pd.DataFrame(tripleton_frame, columns=['Item', 'Support_count'])
tripleton_df
```

	Item	Support_count
0	T-shirt Trousers Sneakers	1
1	T-shirt Trousers Belt	2
2	T-shirt Trousers Jacket	2
3	T-shirt Trousers Gloves	0
4	T-shirt Sneakers Belt	1
5	T-shirt Sneakers Jacket	1
6	T-shirt Sneakers Gloves	0
7	T-shirt Belt Jacket	1
8	T-shirt Belt Gloves	0
9	T-shirt Jacket Gloves	0
10	Trousers Sneakers Belt	3

```
[ ]  tripleton_df = tripleton_df[tripleton_df['Support_count'] >= 3]
     tripleton_df.reset_index(drop=True,inplace=True)
     tripleton_df
```

	Item	Support_count
0	Trousers Sneakers Belt	3

Association Rules

For Doubleton

```
[ ]  doubleton_association_rules = []
     for i in range(len(doubleton_df)):
       ele = []
       l = []
       l = doubleton_df.loc[i,"Item"].split(" ")
       ele.append(l[0] + " --> " + l[1])
       index = singleton_df.loc[singleton_df["Item"] == l[0]].index[0]
       conf = doubleton_df.loc[i,"Support_count"] / singleton_df.loc[index,"Support_count"]
       ele.append(conf)
       doubleton_association_rules.append(ele)
```

```
[ ]  doubleton_association_rules = pd.DataFrame(doubleton_association_rules, columns=['Rule', 'Confidence'])
     doubleton_association_rules
```

	Rule	Confidence
0	T-shirt --> Trousers	0.75
1	T-shirt --> Jacket	0.75
2	Trousers --> Sneakers	0.60
3	Trousers --> Belt	0.80
4	Sneakers --> Belt	1.00

For Tripleton

+ Code + Text

```
[ ]  tripleton_association_rules = []
     for i in range(len(tripleton_df)):
       l = tripleton_df.loc[i,"Item"].split(" ")
       for j in range(3):
         ele = []
         c = l.copy()
         c.remove(l[j])
         ele.append(l[j] + "---> " + c[0] + " " + c[1])
         index = singleton_df.loc[singleton_df["Item"] == l[j]].index[0]
         conf = tripleton_df.loc[i,"Support_count"] / singleton_df.loc[index,"Support_count"]
```

```
      ele.append(conf)
      tripleton_association_rules.append(ele)
    for k in range(3):
      n = k+1
      while(n<len(l)):
        ele = []
        v = l.copy()
        v.remove(l[k])
        v.remove(l[n])
        ele.append(l[k] + " " + l[n] + " ---> " + v[0])
        index = doubleton_df.loc[doubleton_df["Item"] == l[k] + " " + l[n]].index[0]
        conf = tripleton_df.loc[i,"Support_count"] / doubleton_df.loc[index,"Support_count"]
        ele.append(conf)
        tripleton_association_rules.append(ele)
        n = n + 1
```

+ Code + Text

```
[ ] tripleton_association_rules = pd.DataFrame(tripleton_association_rules, columns=['Rule', 'Confidence'])
    tripleton_association_rules
```

	Rule	Confidence
0	Trousers---> Sneakers Belt	0.60
1	Sneakers---> Trousers Belt	1.00
2	Belt---> Trousers Sneakers	0.75
3	Trousers Sneakers ---> Belt	1.00

CONCLUSION:
We have successfully implemented program for Apriori Algorithm.