**Ex.no : 4**

# ASSOCIATION MINING

**24/07/2024**

## AIM

To implements programs on forming strong association rules using Apriori andFP Growth algorithms.

## 1. To Form Association Rules based on Apriori Algorithm

### Code :

```
import pandas as pd
from mlxtend.frequent_patterns import apriori,association_rules transactions={
'I1':[1,0,0,1,1,0,1,1,1],
'I2':[1,1,1,1,0,1,0,1,1],
'I3':[0,0,1,0,1,1,1,1,1],
'I4':[0,1,0,1,0,0,0,0,0],
'I5':[1,0,0,0,0,0,0,1,0]
}
df=pd.DataFrame(transactions) print(df)
```

```
# to generate frequent_item sets
frequent_itemsets=apriori(df,min_support=0.2,use_colnames=True) print(frequent_itemsets)
# to form association rules from the obtained frequent itemset
rules=association_rules(frequent_itemsets,metric="confidence",min_threshold=0.2)
print(rules.iloc[:,:6])
```

### Output :

```
     support      itemsets
0   0.666667          (I1)
1   0.777778          (I2)
2   0.666667          (I3)
3   0.222222          (I4)
4   0.222222          (I5)
5   0.444444      (I2, I1)
6   0.444444      (I3, I1)
7   0.222222      (I5, I1)
8   0.444444      (I2, I3)
9   0.222222      (I2, I4)
10  0.222222      (I5, I2)
11  0.222222  (I2, I3, I1)
12  0.222222  (I5, I2, I1)
```

## 2. To Apply Apriori Algorithms on dataset contains list of lists

### Code :

```
from mlxtend.preprocessing import TransactionEncoder data = [
['milk', 'bread', 'eggs'],
['milk', 'bread'],
['milk', 'diapers', 'beer', 'bread'], ['bread', 'butter'],
['milk', 'diapers', 'bread', 'butter'],
['bread', 'butter', 'beer'],
['milk', 'bread', 'butter']
]
te=TransactionEncoder() te_ary=te.fit(data).transform(data) # creates an num array
transactions=pd.DataFrame(te_ary,columns=te.columns_) #converts it into an dataframe
print(transactions)
```

```
# to generate frequent_item sets
frequent_itemsets=apriori(transactions,min_support=0.3,use_colnames=True)
print(frequent_itemsets)
# to form association rules from the obtained frequent itemset
rules=association_rules(frequent_itemsets,metric="confidence",min_threshold=0.4)
print(rules.iloc[:,:6])
```

### Output :

```
  antecedents consequents  antecedent support  consequent support   support
0    (butter)     (bread)            0.571429            1.000000  0.571429
1     (bread)    (butter)            1.000000            0.571429  0.571429
2     (bread)      (milk)            1.000000            0.714286  0.714286
3      (milk)     (bread)            0.714286            1.000000  0.714286

   confidence
0    1.000000
1    0.571429
2    0.714286
3    1.000000
```

## 3. To Apply FP growth Algorithms on find Association rules

### Code :

```
from mlxtend.frequent_patterns
import fpgrowth,association_rules
frequent_itemsets1=fpgrowth(df,min_support=0.2,use_colnames=True)
print(frequent_itemsets1)
rules=association_rules(frequent_itemsets1,metric="confidence",min_threshold=0. 2)
print(rules.iloc[:,:6])
```

## Output :

```
     support       itemsets
0   0.777778           (I2)
1   0.666667           (I1)
2   0.222222           (I5)
3   0.222222           (I4)
4   0.666667           (I3)
5   0.444444       (I2, I1)
6   0.444444       (I3, I1)
7   0.222222   (I2, I3, I1)
8   0.222222       (I5, I1)
9   0.222222       (I5, I2)
10  0.222222   (I5, I2, I1)
11  0.222222       (I2, I4)
12  0.444444       (I2, I3)
```

## 4. To include only Disease Category Items In The Consequent in the Association rule

## Code :

```python
data1 = [
['fever', 'cough', 'sore throat', 'flu'],
['headache', 'nausea', 'migraine'],
['fever', 'rash', 'measles'],
['fever', 'cough', 'sore throat', 'headache', 'flu'], ['nausea', 'vomiting', 'food poisoning'], ['fever', 'rash', 'headache', 'measles'],
['cough', 'sneezing', 'runny nose', 'cold'],
['fever', 'muscle pain', 'fatigue', 'dengue'], ['headache', 'dizziness', 'blurred vision', 'migraine'],
['nausea', 'diarrhea', 'abdominal pain', 'food poisoning']
]
te=TransactionEncoder() te_ary=te.fit(data1).transform(data1) # creates an num array
transactions_d=pd.DataFrame(te_ary,columns=te.columns_) #converts it into an dataframe
print(transactions_d)
```

```python
frequent_itemsets_d=apriori(transactions_d,min_support=0.2,use_colnames=True)
print(frequent_itemsets_d)
# to extract association rules that contains only disease in the consequent part
rules = association_rules(frequent_itemsets_d, metric="confidence", min_threshold=0.5)
diseases = ['flu', 'migraine', 'measles', 'food poisoning', 'cold', 'dengue'] # Filter rules to only include those with diseases in the consequent
disease_rules = rules[rules['consequents'].apply(lambda x: all(item in diseases for item in x))]
print(disease_rules.iloc[:,:6])
```

## Output :

```
          antecedents        consequents
1             (cough)              (flu)
10      (sore throat)              (flu)
12           (nausea)   (food poisoning)
15         (headache)         (migraine)
17             (rash)          (measles)
19     (cough, fever)              (flu)
```

## Result :

Thus, the above association mining algorithms has been applied andsuccessfully produced the strong association rules.