Name: - Abhijit Saha Batch: - B1 SAP ID: - 60004200112 Subject: - Data Mining & Warehousing

<u>Experiment - 6</u>

<u>Aim</u>: - Implementation of Association rule mining Using:

- 1. Apriori Algorithm,
- 2. FPTree

Theory: -

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of the dataset. It is based on different rules to discover the interesting relations between variables in the database. Association rule learning is one of the very important concepts of machine learning, and it is employed in Market Basket analysis, Web usage mining, continuous production, etc. Here market basket analysis is a technique used by various big retailers to discover the associations between items. We can understand it by taking an example of a supermarket, as in a supermarket, all products that are purchased together are put together. For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored on a shelf or mostly nearby.

Association rule learning can be divided into three types of algorithms:

- 1. Apriori
- 2. Eclat
- 3. F-P Growth Algorithm

Association rule learning works on the concept of If and Else Statements, such as if A then B. Here the If the element is called antecedent, then the statement is called as Consequent. These types of relationships where we can find out some association or relation between two items are known as single cardinality. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly. So, to measure the associations between thousands of data items, there are several metrics. These metrics are given below:

- 1. Support
- 2. Confidence
- 3. Lift

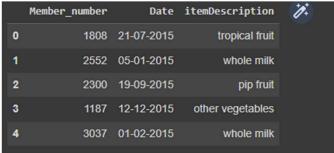
Implementation: -

Connecting to drive from google.colab import drive drive.mount('/content/drive')

Importing and installing required python packages or libraries
!pip install apyori
!pip install mlxtend --upgrade
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from apyori import apriori
from mlxtend.frequent_patterns import fpgrowth, association_rules
from mlxtend.preprocessing import TransactionEncoder

Pre-processing df = pd.read csv('/content/drive/MyDrive/Groceries dataset.csv')

df.head()



df.info()

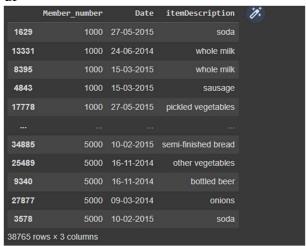
```
for i in df.columns:
```

print(i)

```
Member_number
Date
itemDescription
```

Part A: Apriori

```
# Sorting by members
df.sort_values(by = 'Member_number', inplace = True)
df
```

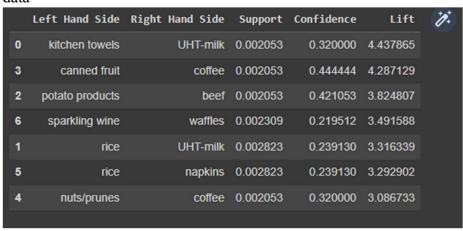


```
 \begin{split} &X = df.iloc[:,[0,2]].values \\ &\# Forming\ transactions \\ &n = 1000 \\ &items = [] \\ &transactions = [] \\ &for\ i\ in\ range(38765): \\ &if(X[i,0] == n): \\ &items.append(X[i,1]) \\ &n = X[i,0] \\ &else: \\ &transactions.append(items) \\ &items = [] \\ &n = X[i,0] \\ &transactions[0] \end{split}
```

Taking only required values

```
['soda',
  'whole milk',
  'whole milk',
  'sausage',
  'pickled vegetables',
  canned beer',
  'yogurt',
  'misc. beverages',
 'salty snack',
 'sausage',
 'semi-finished bread',
 'hygiene articles',
  'pastry']
# Apriori
min sup = float(input('Enter the minimum support: '))
min con = float(input('Enter the minimum confidence: '))
Enter the minimum support: 0.002
Enter the minimum confidence: 0.2
# Forming rules
rules = apriori(transactions = transactions, min support = min sup, min confidence =
min con, min lift = 3, min length = 2, max length = 2)
result = list(rules)
for i in result:
 print(i)
RelationRecord(items=frozenset({'UHT-milk', 'kitchen towels'}),
support=0.002052861175263023,
ordered statistics=[OrderedStatistic(items base=frozenset({'kitchen
lift=4.437864768683275)])
RelationRecord(items=frozenset({'rice', 'UHT-milk'}),
support=0.0028226841159866563,
ordered statistics=[OrderedStatistic(items base=frozenset({'rice'}),
lift=3.3163391613801645)])
RelationRecord(items=frozenset({'beef', 'potato products'}),
support=0.002052861175263023,
ordered statistics=[OrderedStatistic(items base=frozenset({'potato
products'}), items add=frozenset({'beef'}),
confidence=0.4210526315789474, lift=3.8248067721751937)])
RelationRecord(items=frozenset({'canned fruit', 'coffee'}),
support=0.002052861175263023,
ordered statistics=[OrderedStatistic(items base=frozenset({'canned
fruit'}), items add=frozenset({'coffee'}),
confidence=0.4444444444444445, lift=4.287128712871287)])
RelationRecord(items=frozenset({'nuts/prunes', 'coffee'}),
support=0.002052861175263023,
ordered statistics=[OrderedStatistic(items base=frozenset({'nuts/prunes
}), items add=frozenset({'coffee'}), confidence=0.32,
lift=3.086732673267327)])
RelationRecord(items=frozenset({'napkins', 'rice'}),
support=0.0028226841159866563,
```

```
ordered statistics=[OrderedStatistic(items base=frozenset({'rice'}),
items add=frozenset({'napkins'}), confidence=0.2391304347826087,
lift=3.292902135504686)])
RelationRecord(items=frozenset({'waffles', 'sparkling wine'}),
support=0.0023094688221709007,
wine'}), items add=frozenset({'waffles'}),
confidence=0.21951219512195122, lift=3.491587854654057)])
# Inspecting the result
def inspect(result):
 lhs = [tuple(i[2][0][0])[0] for i in result]
 rhs = [tuple(i[2][0][1])[0] \text{ for } i \text{ in result}]
 support = [i[1] \text{ for } i \text{ in result}]
 confidence = [i[2][0][2] for i in result]
 lift = [i[2][0][3] for i in result
 return list(zip(lhs, rhs, support, confidence, lift))
data = pd.DataFrame(inspect(result), columns = ['Left Hand Side', 'Right Hand Side',
'Support', 'Confidence', 'Lift'])
data.sort values(by = 'Lift', ascending = False,inplace = True)
data
```

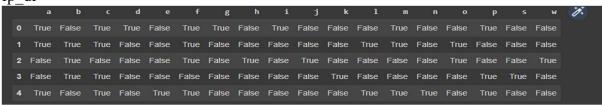


Part B: FP Tree

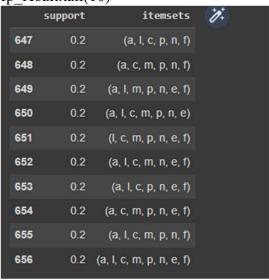
```
# FP Tree

fp_data = [
    ['f', 'a', 'c', 'd', 'g', 'i', 'm', 'p'],
    ['a', 'b', 'c', 'f', 'l', 'm', 'o'],
    ['b', 'f', 'h', 'j', 'o', 'w'],
    ['b', 'c', 'k', 's', 'p'],
    ['a', 'f', 'c', 'e', 'l', 'p', 'm', 'n']
]
```

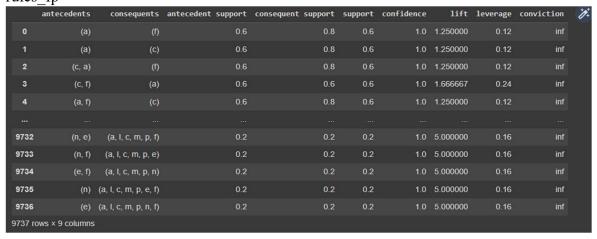
te = TransactionEncoder()
te_array = te.fit(fp_data).transform(fp_data)
fp_df = pd.DataFrame(te_array, columns=te.columns_)
fp_df



fp_result = fpgrowth(fp_df, min_support = min_sup, use_colnames = True)
fp_result.tail(10)



rules_fp = association_rules(fp_result, metric="confidence", min_threshold=0.8) rules_fp



Conclusion: Implemented Apriori and algorithm for a market basket analysis dataset and made an FP Tree for the given dataset.