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

Experiment – 6

Aim: - 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()
```

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Member_number    38765 non-null  int64
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB
```

```
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
```

	Member_number	Date	itemDescription
1629	1000	27-05-2015	soda
13331	1000	24-06-2014	whole milk
8395	1000	15-03-2015	whole milk
4843	1000	15-03-2015	sausage
17778	1000	27-05-2015	pickled vegetables
...	...	...	...
34885	5000	10-02-2015	semi-finished bread
25489	5000	16-11-2014	other vegetables
9340	5000	16-11-2014	bottled beer
27877	5000	09-03-2014	onions
3578	5000	10-02-2015	soda

38765 rows × 3 columns

```
# Taking only required values
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]
```

```
['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
towels'}), items_add=frozenset({'UHT-milk'}), confidence=0.32,
lift=4.437864768683275)])
RelationRecord(items=frozenset({'rice', 'UHT-milk'}),
support=0.002822684115986563,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'rice'}),
items_add=frozenset({'UHT-milk'}), confidence=0.2391304347826087,
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=0.42105263157894 74, 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=0.002822684115986563,
```

```
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,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'sparkling
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] for i in result]
  support = [i[1] for i 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
```

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	kitchen towels	UHT-milk	0.002053	0.320000	4.437865
3	canned fruit	coffee	0.002053	0.444444	4.287129
2	potato products	beef	0.002053	0.421053	3.824807
6	sparkling wine	waffles	0.002309	0.219512	3.491588
1	rice	UHT-milk	0.002823	0.239130	3.316339
5	rice	napkins	0.002823	0.239130	3.292902
4	nuts/prunes	coffee	0.002053	0.320000	3.086733

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
```

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	s	w
0	True	False	True	True	False	True	True	False	True	False	False	False	True	False	False	True	False	False
1	True	True	True	False	False	True	False	False	False	False	False	True	True	False	True	False	False	False
2	False	True	False	False	False	True	False	True	False	True	False	False	False	False	True	False	False	True
3	False	True	True	False	False	False	False	False	False	False	True	False	False	False	False	True	True	False
4	True	False	True	False	True	True	False	False	False	False	False	True	True	True	False	True	False	False

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

	support	itemsets
647	0.2	(a, l, c, p, n, f)
648	0.2	(a, c, m, p, n, f)
649	0.2	(a, l, m, p, n, e, f)
650	0.2	(a, l, c, m, p, n, e)
651	0.2	(l, c, m, p, n, e, f)
652	0.2	(a, l, c, m, n, e, f)
653	0.2	(a, l, c, p, n, e, f)
654	0.2	(a, c, m, p, n, e, f)
655	0.2	(a, l, c, m, p, n, f)
656	0.2	(a, l, c, m, p, n, e, f)

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

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(a)	(f)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
1	(a)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
2	(c, a)	(f)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
3	(c, f)	(a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
4	(a, f)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
...	...	...	...	...	...	...	...	...	...
9732	(n, e)	(a, l, c, m, p, f)	0.2	0.2	0.2	1.0	5.000000	0.16	inf
9733	(n, f)	(a, l, c, m, p, e)	0.2	0.2	0.2	1.0	5.000000	0.16	inf
9734	(e, f)	(a, l, c, m, p, n)	0.2	0.2	0.2	1.0	5.000000	0.16	inf
9735	(n)	(a, l, c, m, p, e, f)	0.2	0.2	0.2	1.0	5.000000	0.16	inf
9736	(e)	(a, l, c, m, p, n, f)	0.2	0.2	0.2	1.0	5.000000	0.16	inf

9737 rows × 9 columns

Conclusion: -

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