Name: Dhruv Bheda **SAPID:** 60004200102

DIV: B/B1

DMW

Exp6

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. Lif

Apriori

1. It is an array based algorithm.
2. It uses Join and Prune technique.
3. Apriori uses a breadth-first search
4. Apriori utilizes a level-wise approach where it generates patterns containing 1 item, then 2 items, then 3 items, and so on.
5. Candidate generation is extremely slow. Runtime increases exponentially depending on the number of different items.
6. Candidate generation is very parallelizable.
7. It requires large memory space due to large number of candidate generation.
8. It scans the database multiple times for generating candidate sets.

FP Growth

1. It is a tree based algorithm.
2. It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy minimum support.
3. FP Growth uses a depth-first search
4. FP Growth utilizes a pattern-growth approach means that, it only considers patterns actually existing in the database.
5. Runtime increases linearly, depending on the number of transactions and items
6. Data are very interdependent, each node needs the root.
7. It requires less memory space due to compact structure and no candidate generation.
8. It scans the database only twice for constructing frequent pattern tree.

Part A:

Read min_support and confidence from the user

Program Apriori algorithm using inbuilt functions.

Print the association rules

Code:

```
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder

print("Enter the minimum support & confidence ")
min_support = float(input("Enter minimum suppport :  "))
min_confidence = float(input("Enter the minimum confidence :  "))
print()
```

```python
# Read the dataset
df = pd.read_csv('/content/GroceryStoreDataSet.csv', names = ['products'], se
p = ',')

# split the data & create list called by data
data = list(df["products"].apply(lambda x:x.split(",") ))

#Let's transform the list, with one-hot encoding
a = TransactionEncoder()
a_data = a.fit(data).transform(data)
df = pd.DataFrame(a_data,columns=a.columns_)
df = df.replace(False,0)
df = df.replace(True,1)

#set a threshold value for the support value and calculate the support value.
df = apriori(df, min_support = min_support, use_colnames = True)

# frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
df['length'] = df['itemsets'].apply(lambda x: len(x))

print(f"Associatives rules with minimum confidence {min_confidence*100}% are
: ")
#Let's view our interpretation values using the Associan rule function.
df_ar = association_rules(df, metric = "confidence", min_threshold = min_conf
idence)
df_ar
```

Output:

```
Enter the minimum support & confidence
Enter minimum suppport :  0.2
Enter the minimum confidence :  0.6

Associatives rules with minimum confidence 60.0% are :
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(MILK)	(BREAD)	0.25	0.65	0.2	0.800000	1.230769	0.0375	1.75
1	(SUGER)	(BREAD)	0.30	0.65	0.2	0.666667	1.025641	0.0050	1.05
2	(CORNFLAKES)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.0800	1.80
3	(SUGER)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.0800	1.80
4	(MAGGI)	(TEA)	0.25	0.35	0.2	0.800000	2.285714	0.1125	3.25

Part B:

Program FP tree using inbuilt functions for the following dataset

TID	*Items bought*
100	{*f, a, c, d, g, i, m, p*}
200	{*a, b, c, f, l, m, o*}
300	{*b, f, h, j, o, w*}
400	{*b, c, k, s, p*}
500	{*a, f, c, e, l, p, m, n*}

Print the frequent patterns generated.

<u>Code:</u>

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns.fpgrowth import fpgrowth
from mlxtend.frequent_patterns import association_rules

print("Enter the minimum support & confidence ")
min_support = float(input("Enter minimum suppport :  "))
min_confidence = float(input("Enter the minimum confidence :  "))

dataset = [['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']]

# Encode the data
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

```
# Fptree and patten table with minimum support of 60%
print(f"\nFP tree with minimum support as {min_support*100}%\n")
pattern = fpgrowth(df, min_support=min_support, use_colnames=True, verbose=2)
 # 3/5 = 60%
print(pattern)
```

```
# Association rules
print("\n\nThe association rules are as follows : ")
rules = association_rules(pattern, metric = "confidence", min_threshold = min
_confidence)
rules
```

Output:

```
Enter the minimum support & confidence
Enter minimum suppport :  0.6
Enter the minimum confidence :  0.8
```

	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 tree with minimum support as 60.0%

6 itemset(s) from tree conditioned on items ()
0 itemset(s) from tree conditioned on items (f)
1 itemset(s) from tree conditioned on items (c)
1 itemset(s) from tree conditioned on items (p)
3 itemset(s) from tree conditioned on items (m)
7 itemset(s) from tree conditioned on items (a)
0 itemset(s) from tree conditioned on items (b)
    support      itemsets
0       0.8           (f)
1       0.8           (c)
2       0.6           (p)
3       0.6           (m)
4       0.6           (a)
5       0.6           (b)
6       0.6        (c, f)
7       0.6        (c, p)
8       0.6        (c, m)
9       0.6        (m, f)
10      0.6     (c, m, f)
11      0.6        (m, a)
12      0.6        (c, a)
13      0.6        (f, a)
14      0.6     (c, m, a)
15      0.6     (m, f, a)
16      0.6     (c, f, a)
17      0.6  (c, m, f, a)
```

The association rules are as follows :

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(p)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
1	(m)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
2	(m)	(f)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
3	(c, m)	(f)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
4	(c, f)	(m)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
5	(m, f)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
6	(m)	(c, f)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
7	(m)	(a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
8	(a)	(m)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
9	(a)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
10	(a)	(f)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
11	(c, m)	(a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
12	(c, a)	(m)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
13	(m, a)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
14	(m)	(c, a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
15	(a)	(c, m)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
16	(m, f)	(a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
17	(m, a)	(f)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
18	(f, a)	(m)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
19	(m)	(f, a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
20	(a)	(m, f)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
21	(c, f)	(a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
22	(c, a)	(f)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
23	(f, a)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
24	(a)	(c, f)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
25	(c, m, f)	(a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
26	(c, m, a)	(f)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
27	(c, f, a)	(m)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
28	(m, f, a)	(c)	0.6	0.8	0.6	1.0	1.250000	0.12	inf
29	(c, m)	(f, a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
30	(c, f)	(m, a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
31	(c, a)	(m, f)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
32	(m, f)	(c, a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
33	(m, a)	(c, f)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
34	(f, a)	(c, m)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
35	(m)	(c, f, a)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
36	(a)	(c, m, f)	0.6	0.6	0.6	1.0	1.666667	0.24	inf

Conclusion:

Implemented Apriori and algorithm for a market basket analysis dataset and made and FP Tree for the given dataset. Apriori is a Join-Based algorithm and FP-Growth is Tree-Based algorithm for frequent itemset mining or frequent pattern mining for market basket analysis.