Name : Meith Navlakha
Comps B1
DWM Practical

Experiment 5

AIM: Implementation of Association rule mining Using
1) Apriori Algorithm
2) FPTree

Part A:

1. Program Apriori from scratch.
2. Read min_support and confidence from the user
3. Print the association rules

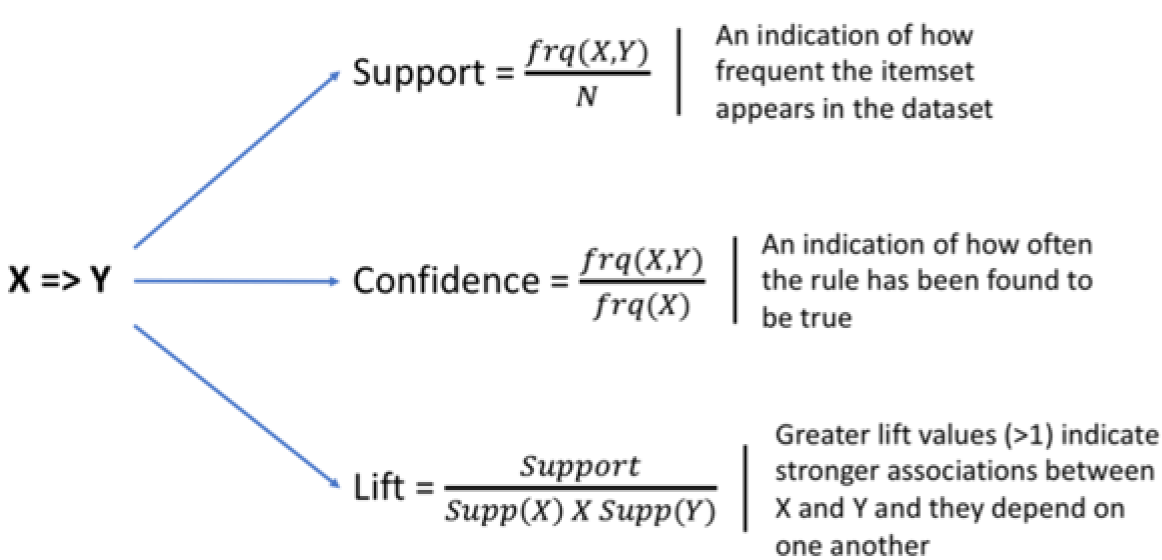
Part B:

1. Program FP tree using inbuilt functions.
2. Print the frequent patterns generated.

THEORY:

Association Rule Algorithm:

Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using some measures of interestingness. The rule shows how frequently an itemset occurs in a transaction.



Name : Meith Navlakha
Comps B1
DWM Practical

Apriori Algorithm:

Apriori algorithm is used for finding frequent itemsets in a dataset for boolean association rule. The name Apriori, is because it uses prior knowledge of frequent itemset properties. It is designed to work on the databases that contain transactions.

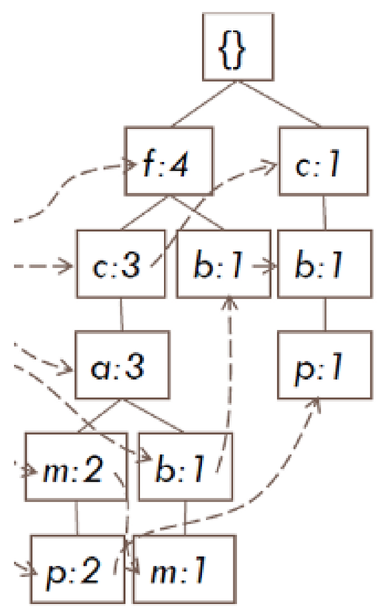
Apriori uses breadth-first search and a Hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length k from item sets of length k-1. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.
Limitations: Time and Space complexity of the algorithm are very high: $O(2^{|D|})$, where |D| is the horizontal width (the total number of items) present in the database.

To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called Apriori property which helps by reducing the search space.

FP Tree Algorithm:

A FP-tree(or Frequent Pattern Tree) is a compact data structure that represents the data set in tree form. Each transaction is read and then mapped onto a path in the FP-tree which is used to maintain associations between the itemsets. This is done until all transactions have been read. Different transactions that have common subsets allow the tree to remain compact because their paths overlap.

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



Name : Meith Navlakha
Comps B1
DWM Practical

CODE:

Part A:
1) Program Apriori from scratch.
2) Read min_support and confidence from the user
3) Print the association rules

```
import pandas as pd
import numpy as np
import math
transaction_df =
pd.read_csv('GroceryStoreDataSet.csv')
print(transaction_df)
transaction_df.index.rename('TID',
inplace=True)

transaction_df.rename(columns={'MILK,BREA
D,BISCUIT' : 'item_list'}, inplace=True)
trans_df = transaction_df.item_list.str.split(',')
def prune(data,supp):

df = data[data.supp_count >= supp] return df

def count_itemset(transaction_df, itemsets):
count_item = {}
for item_set in itemsets: set_A = set(item_set)
for row in trans_df: set_B = set(row)
if set_B.intersection(set_A) == set_A: if
item_set in count_item.keys():
count_item[item_set] += 1
else:
count_item[item_set] = 1 data = pd.DataFrame()
data['item_sets'] = count_item.keys()
data['supp_count'] = count_item.values() return
data
def count_item(trans_items): count_ind_item={}

for row in trans_items: for i in range(len(row)):
        if row[i] in count_ind_item.keys():
        count_ind_item[row[i]] += 1

        else:
        count_ind_item[row[i]] = 1 data =
        pd.DataFrame()
        data['item_sets'] = count_ind_item.keys()
        data['supp_count'] =
```

```
    count_ind_item.values() data =
    data.sort_values('item_sets')

return data

def join(list_of_items): itemsets = []
i = 1
for entry in list_of_items: proceding_items =
list_of_items[i:] for item in proceding_items:
if(type(item) is str): if entry != item:
tuples = (entry, item) itemsets.append(tuples)
else:

if entry[0:-1] == item[0:-1]: tuples =
entry+item[1:] itemsets.append(tuples)
i = i+1

if(len(itemsets) == 0): return None
return itemsets
def apriori(trans_data,supp=3, con=0.5): freq =
pd.DataFrame()
df = count_item(trans_data) while(len(df) != 0):
df = prune(df, supp)
if len(df) > 1 or (len(df) == 1 and
int(df.supp_count >= supp)): freq = df
itemsets = join(df.item_sets) if(itemsets is
None):
return freq

df = count_itemset(trans_data, itemsets) return
df req_item_sets = apriori(trans_df, 4)
freq_item_sets

def calculate_conf(value1, value2):
return round(int(value1)/int(value2) * 100, 2)
def strong_rules(freq_item_sets, threshold):
confidences = {}
for row in freq_item_sets.item_sets:

for i in range(len(row)): for j in range(len(row)):
if i != j:
tuples = (row[i], row[j])
```

Name : Meith Navlakha
Comps B1


```
conf =
calculate_conf(freq_item_sets[freq_item_sets.ite
m_sets == row].supp_count,
count_item(trans_df)[count_item(trans_df).item
_sets == row[i]].supp_count)
confidences[tuples] = conf conf_df =
pd.DataFrame() conf_df['item_set'] =
confidences.keys()
conf_df['confidence'] = confidences.values()
return conf_df[conf_df.confidence >= threshold]
strong_rules(freq_item_sets, 50.0)

from functools import reduce

import operator
def interesting_rules(freq_item_sets): lifts = {}
prob_of_items = []
for row in freq_item_sets.item_sets:
num_of_items = len(row)
prob_all =
freq_item_sets[freq_item_sets.item_sets ==
row].supp_count / 18 for i in
range(num_of_items):

prob_of_items.append(count_item(trans_df)[cou
nt_item(trans_df).item_sets
== row[i]].supp_count / 18)

lifts[row] = round(float(prob_all /
reduce(operator.mul, (np.array(prob_of_items)),
1)), 2)
prob_of_items = [] lifts_df = pd.DataFrame()
lifts_df['Rules'] = lifts.keys() lifts_df['lift'] =
lifts.values() return lifts_df
int_rules = interesting_rules(freq_item_sets)
int_rules
```

Name : Meith Navlakha
Comps B1
DWM Practical

OUTPUT:

	MILK,BREAD,BISCUIT
0	BREAD,MILK,BISCUIT,CORNFLAKES
1	BREAD,TEA,BOURNVITA
2	JAM,MAGGI,BREAD,MILK
3	MAGGI,TEA,BISCUIT
4	BREAD,TEA,BOURNVITA
5	MAGGI,TEA,CORNFLAKES
6	MAGGI,BREAD,TEA,BISCUIT
7	JAM,MAGGI,BREAD,TEA
8	BREAD,MILK
9	COFFEE,COCK,BISCUIT,CORNFLAKES
10	COFFEE,COCK,BISCUIT,CORNFLAKES
11	COFFEE,SUGER,BOURNVITA
12	BREAD,COFFEE,COCK
13	BREAD,SUGER,BISCUIT
14	COFFEE,SUGER,CORNFLAKES

Name : Meith Navlakha
Comps B1
DWM Practical

	item_sets	supp_count
15	(BREAD, SUGER)	4
16	(BREAD, TEA)	4
17	(COFFEE, CORNFLAKES)	4
19	(COFFEE, SUGER)	4
26	(MAGGI, TEA)	4

	item_set	confidence
1	(SUGER, BREAD)	66.67
3	(TEA, BREAD)	57.14
4	(COFFEE, CORNFLAKES)	50.00
5	(CORNFLAKES, COFFEE)	66.67
6	(COFFEE, SUGER)	50.00
7	(SUGER, COFFEE)	66.67
8	(MAGGI, TEA)	80.00
9	(TEA, MAGGI)	57.14

Name : Meith Navlakha
Comps B1
DWM Practical

Part B:

1. **Program FP tree using inbuilt functions.**
2. **Print the frequent patterns generated. Code:**

```
import pandas as pd
import numpy as np
import pyfpgrowth transactions = [
['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']

]

FP = pyfpgrowth.find_frequent_patterns( transactions = transactions , support_threshold=3)
print(FP)

FP_Rules = pyfpgrowth.generate_association_rules( patterns = FrequentPatterns,
confidence_threshold=0.8)
print(FP_Rules)
```

OUTPUT:

```
{('a', 'c'): 3,
 ('a', 'f'): 3,
 ('a', 'm'): 3,
 ('c', 'm'): 3,
 ('a', 'c', 'm'): 3,
 ('f', 'm'): 3,
 ('a', 'f', 'm'): 3,
 ('p',): 3,
 ('c', 'p'): 3,
 ('b',): 3,
 ('f',): 4,
 ('c',): 4}
```

```
{('a', 'c'): (('m',), 1.0),
 ('a', 'm'): (('f',), 1.0),
 ('c', 'm'): (('a',), 1.0),
 ('a', 'f'): (('m',), 1.0),
 ('f', 'm'): (('a',), 1.0),
 ('p',): (('c',), 1.0)}
```

CONCLUSION:

In this experiment, I implemented Association Rule Mining using Apriori Algorithm and FPTree. Apriori Algorithm is used for finding frequent pattern in a dataset for boolean association rule using Breadth-First search and a Hash tree structure. It reduces the size of the itemsets in the database significantly, providing a good performance. On the other hand, FP Tree needs to scan the database only twice as compared to Apriori which scans the transactions for each iteration. Also, the pairing of items is not done which makes it faster than FP Tree. Finally, the FP Tree is an improvisation of Apriori Algorithm used for association rule mining.