Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	T. Y. B.Tech (Computer Engineering)
Course:	Data Mining and Warehouse Laboratory
Course Code:	DJ19CEL501
Experiment No.:	06

AIM: Implementation of Association rule mining Using

1. Apriori Algorithm
2. FP Tree

APRIORI ALGORITHM

CODE:

```
import pandas as pd
import numpy as np
import math

transaction_df = pd.read_csv('GroceryStoreDataSet.csv')
transaction_df
transaction_df.index.rename('TID', inplace=True)
transaction_df.rename(columns={'MILK,BREAD,BISCUIT' : 'item_list'}, inplace=True)
trans_df = transaction_df.item_list.str.split(',') trans_df

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)
supp)):
if len(df) > 1 or (len(df) == 1 and int(df.supp_count >= freq = df
itemsets = join(df.item_sets)
if(itemsets is None): return freq
df = count_itemset(trans_data, itemsets) return df
freq_item_sets = apriori(trans_df, 5) 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]) conf =
calculate_conf(freq_item_sets[freq_item_sets.item_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] confidence_threshold = int(input()) #50
strong_rules(freq_item_sets, threshold=confidence_threshold)
# ### Rules with confidence level >= 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)[count_item(trans_df).ite
m_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
```

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,COKE,BISCUIT,CORNFLAKES
10	COFFEE,COKE,BISCUIT,CORNFLAKES
11	COFFEE,SUGER,BOURNVITA
12	BREAD,COFFEE,COKE
13	BREAD,SUGER,BISCUIT
14	COFFEE,SUGER,CORNFLAKES
15	BREAD,SUGER,BOURNVITA
16	BREAD,COFFEE,SUGER
17	BREAD,COFFEE,SUGER
18	TEA,MILK,COFFEE,CORNFLAKES

	item_set	confidence
0	(BISCUIT, BREAD)	50.00
2	(BISCUIT, CORNFLAKES)	50.00
3	(CORNFLAKES, BISCUIT)	50.00
4	(BOURNVITA, BREAD)	75.00
9	(MAGGI, BREAD)	60.00
11	(MILK, BREAD)	75.00
13	(SUGER, BREAD)	66.67
15	(TEA, BREAD)	57.14
17	(COKE, COFFEE)	100.00
18	(COFFEE, CORNFLAKES)	50.00
19	(CORNFLAKES, COFFEE)	66.67
20	(COFFEE, SUGER)	50.00
21	(SUGER, COFFEE)	66.67
22	(MAGGI, TEA)	80.00
23	(TEA, MAGGI)	57.14

	Rules	lift
0	(BISCUIT, BREAD)	0.75
1	(BISCUIT, CORNFLAKES)	1.50
2	(BOURNVITA, BREAD)	1.12
3	(BREAD, COFFEE)	0.56
4	(BREAD, MAGGI)	0.90
5	(BREAD, MILK)	1.12
6	(BREAD, SUGER)	1.00
7	(BREAD, TEA)	0.86
8	(COFFEE, COKE)	2.25
9	(COFFEE, CORNFLAKES)	1.50
10	(COFFEE, SUGER)	1.50
11	(MAGGI, TEA)	2.06

FP TREE ALGORITHM

CODE:

```
import pandas as pd

from mlxtend.preprocessing import TransactionEncoder from
mlxtend.frequent_patterns import fpgrowth

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']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)

df = pd.DataFrame(te_ary, columns=te.columns_)
df

fpgrowth(df, min_support=0.6, use_colnames=True, verbose=2) # 3/5
= 60%
```

OUTPUT:

	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)

	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