|  |  |
| --- | --- |
| 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()

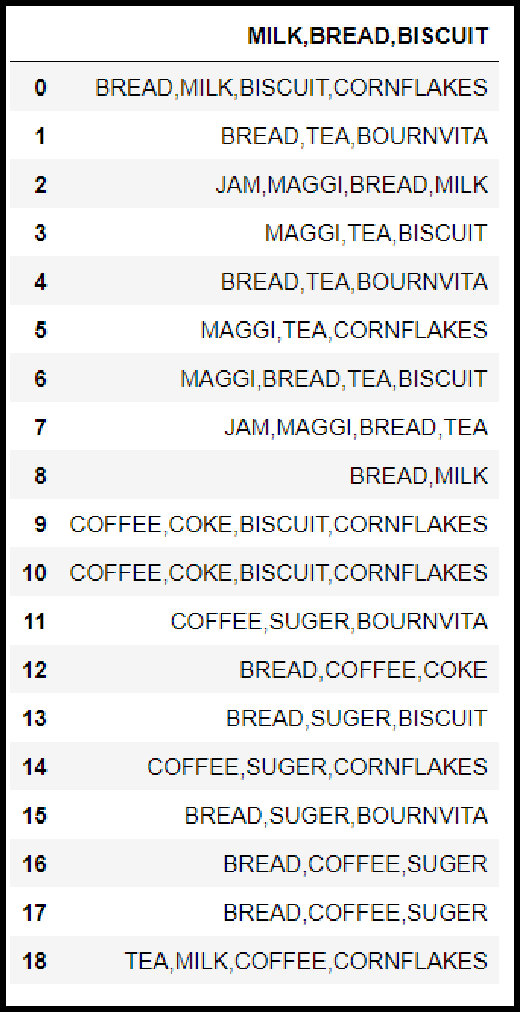
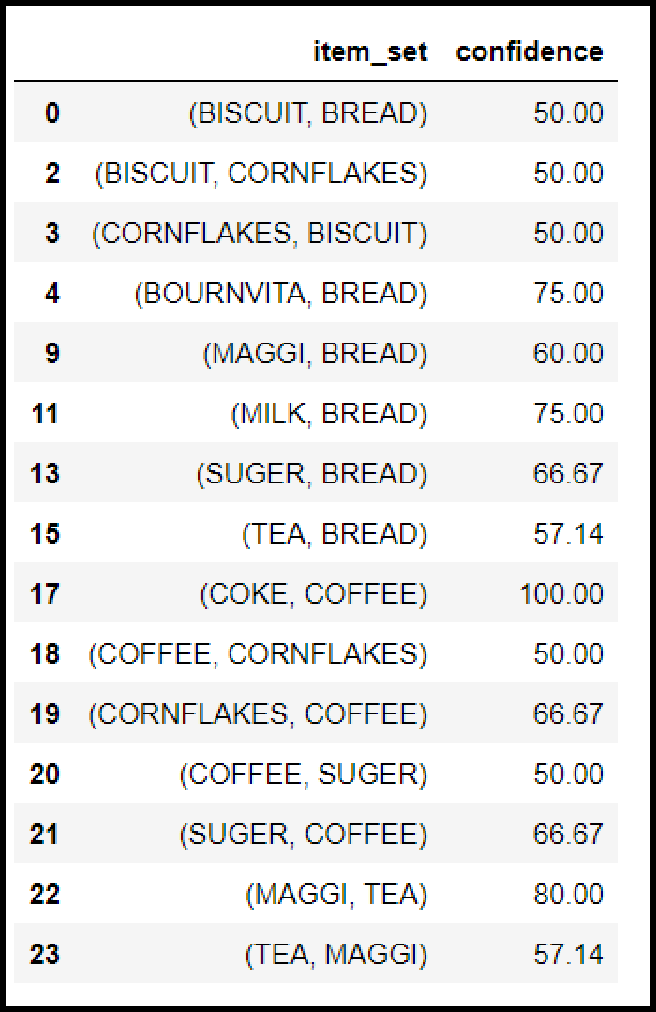
lifts\_df['Rules'] = lifts.keys() lifts\_df['lift'] = lifts.values()

return lifts\_df

int\_rules = interesting\_rules(freq\_item\_sets)

int\_rules

A screenshot of a menu

Description automatically generatedOUTPUT:

A screenshot of a cell phone

Description automatically generatedFP 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%

A screenshot of a computer

Description automatically generatedOUTPUT: