РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

Дисциплина: Интеллектуальный анализ данных

Студент: Гебриал Ибрам Есам Зекри

Группа: НПИбд-01-18

Москва 2021

Вариант № 2

Алгоритм: Apriori

День недели (поле order_dow таблицы orders): “3”

Код департамента (поле department_id таблицы products): “18”

Показатели оценки ассоциативных правил: достоверность (confidence) и рычаг (leverage)

1. При помощи модуля sqlite3 откройте базу данных Instacart в файле instacart.db.

In [36]:

```
import sqlite3
conn = sqlite3.connect('instacart.db')

cursor = conn.cursor()
```

2. При помощи запроса SELECT извлеките из таблицы order_products__train записи, соответствующие указанным в индивидуальном задании дню недели (поле order_dow таблицы orders) и коду департамента (поле department_id таблицы products). Определите количество записей в полученном наборе и определите количество товаров (поле product_id таблицы order_products__train) в транзакциях набора.

In [37]:

```
#for row in cursor.execute
for row in cursor.execute("""
        SELECT order_products__train.order_id, products.product_name,products.department_id
        FROM order_products__train
        JOIN orders on order_products__train.order_id = orders.order_id
        JOIN products  on order_products__train .product_id = products.product_id
        WHERE

            orders.order_dow = 3 AND products.department_id = 18

         """):
    print(row)
```

```
('2885343', 'Simply Clean Baby Wipes', '18', '3')
('450817', 'Sweet Potato Apple Stage 2 Baby Food', '18', '3')
('450817', 'Apple Peach Oatmeal Stage 2', '18', '3')
('1701939', 'Organic Stage 4 Green Bean Pear & Pea Baby Food', '18', '3')
('1214783', 'Stage 1 Apples & Strawberries Organic Pureed Baby Food', '1
8', '3')
('1214783', 'Organic Pears, Peas and Broccoli Puree Stage 1', '18', '3')
('2889501', 'Apple Blueberry Fruit Yogurt Smoothie', '18', '3')
('2889501', 'Organic Fruit Yogurt Smoothie Mixed Berry', '18', '3')
('2889501', 'Organic Strawberry Banana Fruit Yogurt Smoothie', '18', '3')
('2889501', 'Organic Banana Blueberry Baby Food Puree', '18', '3')
('2889501', 'Orange Banana Stage 2 Baby Food', '18', '3')
('2889501', 'Carrot and Broccoli Baby Food Puree', '18', '3')
('1690385', 'Gluten Free SpongeBob Spinach Littles', '18', '3')
('1633791', 'Whole Grain Rice Cereal', '18', '3')
('1409854', 'Oatmilk Calendula Moisturizing Baby Shampoo & Wash', '18',
'3')
('1409854', 'Baby Bee Shampoo and Wash', '18', '3')
('1409854', 'Blueberry & Purple Carrot Teething Wafers', '18', '3')
('1003391', 'Sunny Days Strawberry Snack Bars', '18', '3')
```

In [38]:

```
for row in cursor.execute("""
        SELECT count(*), count(DISTINCT order_products__train.order_id)
        FROM order_products__train
        JOIN orders on order_products__train.order_id = orders.order_id
        JOIN products  on order_products__train.product_id = products.product_id
        WHERE

            orders.order_dow = 3 AND products.department_id = 18

         """):
    print(row)
```

```
(1774, 772)
```

3. Определите количество покупок (транзакций) для пяти наиболее популярных товаров в наборе.

In [39]:

```
for row in cursor.execute("""
    select count(opp.order_id) as counter, p.product_name as popular_product
    from order_products__train opp,orders,products as p
    where p.product_id = opp.product_id and opp.order_id=orders.order_id  and  orders.order
    group by popular_product
    order by counter desc
    limit 5
    """):
    print(row)
```

```
(39, 'Baby Food Stage 2 Blueberry Pear & Purple Carrot')
(38, 'Gluten Free SpongeBob Spinach Littles')
(36, 'Spinach Peas & Pear Stage 2 Baby Food')
(25, 'Peach,  Apricot & Banana Stage 2 Baby Food')
(24, 'Organic Mixed Berry Yogurt & Fruit Snack')
```

4. Постройте транзакционную базу данных для поиск ассоциативных правил из полученного набора записей таблицы order_products__train, используя в качестве идентификатора транзакции поле order_id, а в качестве названий товаров - поле product_name из таблицы products, соответствующее полю product_id.

In [40]:

```
arr=[]
database=dict()

for row in cursor.execute("""
        SELECT orders2.order_id, products.product_name
        FROM order_products__train as orders2, orders, products
        WHERE orders2.order_id=orders.order_id AND orders2.product_id=products.product_id A
        order by orders2.order_id
        """):

    if int(row[0]) in database:
        database[int(row[0])].add(row[1])
    else:
        database[int(row[0])]={row[1]}

for order, products in database.items():
    arr.append([order,products])

print(arr)
```

```
by Food', 'Pumpkin, Sweet Potato and Pear Fruit and Veggie Blend'}], [1516
75, {'Peachy Keen Organic Level 1', 'Grow Well DHA Baby Food', 'Organic St
age 2 Apple Oatmeal With Cinnamon Baby Food', 'Happy Tot Banana, Peach, Pr
une & Coconut Organic Superfoods', 'Grow Well Muscle Baby Food', 'Honest F
ace, Hand, & Baby Wipes', 'Organic Lavenberry Puree Level 2'}], [1523329,
{'Spinach Peas & Pear Stage 2 Baby Food', 'First Prunes'}], [1531450, {'Sp
inach Peas & Pear Stage 2 Baby Food', 'Stage 2 Sweet Potato Corn & Apple B
aby Food', 'Oral Electrolyte Solution Strawberry', 'Peach,  Apricot & Bana
na Stage 2 Baby Food', 'Broccoli & Apple Stage 2 Baby Food', 'Baby Food St
age 2 Blueberry Pear & Purple Carrot'}], [1533868, {'Hot Kid Organic Baby
Mum-Mum Original Rice Rusks', 'Kale & Spinach Superfood Puffs', 'Banana &
Sweet Potato Organic Teething Wafers'}], [1538136, {'Bananas, Raspberries
& Oats Organic Baby Food'}], [1541572, {'Blueberry Oats and Quinoa Whole G
rain Snack', 'Baby Food Stage 2 Pumpkin Banana', 'Organic Stage 2 Apple Oa
tmeal With Cinnamon Baby Food'}], [1545598, {'Organic Sunny Days Apple Sna
ck Bars'}], [1546661, {'Blueberry & Purple Carrot Teething Wafers'}], [156

4874, {'Apple Blueberry Fruit Yogurt Smoothie', 'Organic Fruit Yogurt Smoo
thie Peach Banana', 'Organic Sunny Days Apple Snack Bars'}], [1566261, {'O
rganic Mixed Berry Yogurt & Fruit Snack', 'Banana & Sweet Potato Organic T
eething Wafers', 'Super Morning Organic Apples, Cinnamon, Yogurt & Oats +
```

5. Реализуйте указанный в индивидуальном задании метод построения популярных наборов предметов (3 балла). Протестируйте корректность реализации алгоритма на учебном наборе данных из материалов лекции (лабораторной работы).

In [116]:

```
B_train =[[1,2,5],[2,4],[2,3],[1,2,4],[1,3],[2,3],[1,3],[1,2,3,5],[1,2,3]]
```

In [117]:

```
def createC1(data):
    """
    Create a list of unique items in transaction data.
    Represent each item as a set of length 1.
    """
    C1 = []
    for transaction in data:
        for item in transaction:
            if not [item] in C1:
                C1.append([item])
    C1.sort()

    # create a set for each item in C1
    return [set(x) for x in C1]
```

In [118]:

```
C1=createC1(B_train)
C1
```

Out[118]:

```
[{1}, {2}, {3}, {4}, {5}]
```

In [119]:

```
def selectLk(dataSet,Ck,minSupport): #dataSet raw data set
    scan = {}
    for tid in dataSet:

        for item in Ck:
            if set(item).issubset(tid):
                item = list(map(str, item))
                item = ','.join(item)
                if item not in scan.keys():
                    scan[item] = 1
                else:
                    scan[item] += 1
    numItems = float(len(dataSet))

    Lk = {}
    SupportData = {}
    for key in scan:
        support = scan[key] / numItems
        #supportData[key] = support
        if support >= minSupport:
            Lk[key] = support;

    return Lk
```

In [120]:

```
def createCk(Lk,k):
    Ck = []
    Lk = list(Lk.keys())
    print("Lk:",Lk)
    lenLk = len(Lk)
    for i in range(lenLk):
        for j in range(i+1,lenLk):

            L1 = Lk[i].split(',')
            L1 = list(map(int, L1))
            L1pre = (L1)[:k-2]
            L1pre.sort();
            L2 = Lk[j].split(',')
            L2 = list(map(int, L2))
            L2pre = (L2)[:k-2]
            L2pre.sort()
            if L1pre == L2pre:
                Ck.append(list(set(L1).union(set(L2))))
    print("Ck: ",Ck)
    return Ck
```

In [130]:

```
def apriori(dataSet,minSupport):
    C1 = createC1(dataSet);
    L1 = selectLk(dataSet,C1,minSupport)
    L = [L1]
    k = 2
    while(len(L[k-2]) > 0):
        Ck = createCk(L[k-2],k)
        Lk= selectLk(dataSet,Ck,minSupport)
        L.append(Lk)
        k += 1
    return L
from pprint import pprint
L = apriori(B_train,0.01)
pprint(L)
```

```
Lk: ['1', '2', '5', '4', '3']
Ck:  [[1, 2], [1, 5], [1, 4], [1, 3], [2, 5], [2, 4], [2, 3], [4, 5], [3,
5], [3, 4]]
Lk: ['1,2', '1,5', '2,5', '2,4', '2,3', '1,4', '1,3', '3,5']
Ck:  [[1, 2, 5], [1, 2, 4], [1, 2, 3], [1, 4, 5], [1, 3, 5], [2, 4, 5], [2,
3, 5], [2, 3, 4], [1, 3, 4]]
Lk: ['1,2,5', '1,2,4', '1,2,3', '1,3,5', '2,3,5']
Ck:  [[1, 2, 4, 5], [1, 2, 3, 5], [1, 2, 3, 4]]
Lk: ['1,2,3,5']
Ck:  []
[{'1': 0.6666666666666666,
  '2': 0.7777777777777778,
  '3': 0.6666666666666666,
  '4': 0.2222222222222222,
  '5': 0.2222222222222222},
 {'1,2': 0.4444444444444444,
  '1,3': 0.4444444444444444,
  '1,4': 0.1111111111111111,
  '1,5': 0.2222222222222222,
  '2,3': 0.4444444444444444,
  '2,4': 0.2222222222222222,
  '2,5': 0.2222222222222222,
  '3,5': 0.1111111111111111},
 {'1,2,3': 0.2222222222222222,
  '1,2,4': 0.1111111111111111,
  '1,2,5': 0.2222222222222222,
  '1,3,5': 0.1111111111111111,
  '2,3,5': 0.1111111111111111},
 {'1,2,3,5': 0.1111111111111111},
 {}]
```

6. При помощи указанного в индивидуальном задании метода построения популярного набора предметов или метода BruteForce постройте популярный набор предметов с минимальной поддержкой не менее 3, имеющий максимальную длину. В случае нехватки вычислительных ресурсов (слишком долгой работы программы) при построении популярных наборов предметов сокращайте число записей в наборе данных (например, делая выборку полвины записей набора).

In [66]:

```
query = cursor.execute("""
        SELECT ord.order_id, products.product_name
        FROM order_products__train as ord, orders, products
        WHERE ord.order_id=orders.order_id AND ord.product_id=products.product_id
        AND orders.order_dow='3' AND products.department_id='18'
        order by ord.order_id
        """)
I=set()
D=[]
D_base=dict()
j=0
while j<21:
    row = cursor.fetchone()
    I.add(row[1])
    if int(row[0]) in D_base:
        D_base[int(row[0])].add(row[1])
    else:
        D_base[int(row[0])]={row[1]}
    j=j+1

for order, products in D_base.items():
    D.append([order,products])

D
```

Out[66]:

```
[[1003391, {'Sunny Days Strawberry Snack Bars'}],
 [1009266,
  {'Fiber & Protein Organic Pears, Raspberries, Butternut Squash & Carrots S
nack',
   'Organic Pears, Apples, Peaches, Pumpkin + Cinnamon Fiber & Protein Snac
k'}],
 [10208, {'Kids Sensible Foods Broccoli Littles'}],
 [1021481,
  {'Organic Whole Grain Oatmeal Cereal Baby Food', 'Whole Grain Rice Cerea
l'}],
 [1022467,
  {'Apple Peach Oatmeal Stage 2',
   'Organic Fiber & Protein Pear Blueberry & Spinach Baby Food'}],
 [1030002, {'Gluten Free SpongeBob Spinach Littles'}],
 [1032626,
  {'1st Foods Green Beans',
   '1st Foods Sweet Potatoes',
   'AdvanceCare Oral Electrolyte Solution - Cherry Punch',
   'Oatmeal & Banana Baby Cereal'}],
 [1034515,
  {'Organic Strawberry Banana Fruit Yogurt Smoothie',
   'Sunny Days Strawberry Snack Bars'}],
 [1035897, {'Organic Dairy Iron-Fortified Toddler Formula'}],
 [1050355,
  {'Organic Sunny Days Apple Snack Bars', 'Sunny Days Strawberry Snack Bar
s'}],
 [105944, {'Baby Fresh Pampers Baby Wipes Baby Fresh'}],
 [1073571,
  {'Apple and Blueberry Organic Baby Food',
   'Stage 2 Sweet Potato Corn & Apple Baby Food'}]]
```

In [67]:

```
from itertools import chain, combinations

def powerset(iterable):
    xs = list(iterable)
    # возвращаем итератор, а не список
    return chain.from_iterable(combinations(xs,n) for n in range(len(xs)+1))
```

In [68]:

```
def BruteForce(D, I, minsup):
    F = []
    for X in powerset(I):
        if len(X) > 0:
            supX = ComputeSupport(set( X ), D)
            if supX >= minsup:
                F.append([ X, supX ])
    return F
```

In [69]:

```
def ComputeSupport(X, D):
    supX = 0
    for _,itemset in D:
        if X.issubset(itemset):
            supX += 1
    return supX
```

In [70]:

```
I
```

Out[70]:

```
{'1st Foods Green Beans',
 '1st Foods Sweet Potatoes',
 'AdvanceCare Oral Electrolyte Solution - Cherry Punch',
 'Apple Peach Oatmeal Stage 2',
 'Apple and Blueberry Organic Baby Food',
 'Baby Fresh Pampers Baby Wipes Baby Fresh',
 'Fiber & Protein Organic Pears, Raspberries, Butternut Squash & Carrots Sna
ck',
 'Gluten Free SpongeBob Spinach Littles',
 'Kids Sensible Foods Broccoli Littles',
 'Oatmeal & Banana Baby Cereal',
 'Organic Dairy Iron-Fortified Toddler Formula',
 'Organic Fiber & Protein Pear Blueberry & Spinach Baby Food',
 'Organic Pears, Apples, Peaches, Pumpkin + Cinnamon Fiber & Protein Snack',
 'Organic Strawberry Banana Fruit Yogurt Smoothie',
 'Organic Sunny Days Apple Snack Bars',
 'Organic Whole Grain Oatmeal Cereal Baby Food',
 'Stage 2 Sweet Potato Corn & Apple Baby Food',
 'Sunny Days Strawberry Snack Bars',
 'Whole Grain Rice Cereal'}
```

In [71]:

```
minsup = 3

print("\nПопулярные наборы предметов для minsup =", minsup, ":")

for itemset in BruteForce( D, I, minsup ):
    print( itemset )
```

```
Популярные наборы предметов для minsup = 3 :
[('Sunny Days Strawberry Snack Bars',), 3]
```

7. Для полученного популярного набора предметов постройте набор ассоциативных правил.

In [72]:

```
F_set, _ = BruteForce( D, I, minsup )[-1]
F_set
```

Out[72]:

```
('Sunny Days Strawberry Snack Bars',)
```

In [135]:

```
def powersetk(iterable,k):
    xs = list(iterable)
    # возвращаем итератор, а не список
    return chain.from_iterable(combinations(xs,n) for n in range(k,len(xs)+1))

def AssociationRules(D, Z_set, minconf):
    A_rules = []
    supZ = ComputeSupport(set(Z_set), D)
    A_set = list(powersetk(Z_set,1))[:-1]
    # print("\nA_set:",A_set)
    while len(A_set)>0:
        X_set = A_set[-1]
        #print("\nX_set:",X_set)
        A_set.pop()
        conf = supZ/ComputeSupport(set(X_set), D)
        if conf >= minconf:
            Y_set = sorted(list(set(Z_set)-set(X_set)))
            A_rules.append([X_set, Y_set, supZ, conf])
        else:
            for W_set in powersetk(X_set,1):
                if W_set in A_set:
                    A_set.remove(W_set)
    return A_rules

AssociationRules(D, F_set,0.9)
```

Out[135]:

```
[]
```

8. Для построенного набора ассоциативных правил вычислите показатели (меры) оценки

ассоциативных правил, указанные в индивидуальном задании

In [26]:

```
D_train =[
    [ 1, {"A","B","D","E"} ],
    [ 2, {"B","C","E"} ],
    [ 3, {"A","B","D","E"} ],
    [ 4, {"A","B","C","E"} ],
    [ 5, {"A","B","C","D","E"} ],
    [ 6, {"B","C","D"} ],
]


I = {"A","B","C", "D", "E"}

minsup = 3

print("\nПопулярные наборы предметов для minsup =", minsup, ":")

for itemset in BruteForce( D_train, I, minsup ):
    print( itemset )
```

```
Популярные наборы предметов для minsup = 3 :
[('E',), 5]
[('C',), 4]
[('D',), 4]
[('A',), 4]
[('B',), 6]
[('E', 'C'), 3]
[('E', 'D'), 3]
[('E', 'A'), 4]
[('E', 'B'), 5]
[('C', 'B'), 4]
[('D', 'A'), 3]
[('D', 'B'), 4]
[('A', 'B'), 4]
[('E', 'C', 'B'), 3]
[('E', 'D', 'A'), 3]
[('E', 'D', 'B'), 3]
[('E', 'A', 'B'), 4]
[('D', 'A', 'B'), 3]
[('E', 'D', 'A', 'B'), 3]
```

In [29]:

```
F_set, _ = BruteForce(D_train, I, minsup)[-1]

fn=AssociationRules(D_train, F_set, 0.9)
fn
```

Out[29]:

```
[[('D', 'A', 'B'), ['E'], 3, 1.0],
 [('E', 'D', 'B'), ['A'], 3, 1.0],
 [('E', 'D', 'A'), ['B'], 3, 1.0],
 [('D', 'A'), ['B', 'E'], 3, 1.0],
 [('E', 'D'), ['A', 'B'], 3, 1.0]]
```

confidence

In [33]:

```
for i in range(5):
    confidence=ComputeSupport(set(F_set), D_train)/ComputeSupport(set(fn[i][1]), D_train)
    print(confidence)
```

```
0.6
0.75
0.5
0.6
0.75
```

leverage

In [35]:

```
for i in range(5):
    leverage=(ComputeSupport(set(F_set), D_train)/6)- (ComputeSupport(set(fn[i][0]), D_trai
    print(leverage)
```

```
0.08333333333333331
0.16666666666666669
0.0
0.08333333333333331
0.16666666666666669
```