РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Дисциплина: Интеллектуальный анализ данных

Студент: Леонова Алина

Группа: НФИбд-02-17

Москва 2020

Вариант №3

Постановка задачи:

- При помощи модуля sqlite3 откройте базу данных Instacart в файле instacart.db.
- 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 order_id таблицы order_products**train) в транзакциях набора.
- Определите количество покупок (транзакций) для пяти наиболее популярных товаров в наборе.
- 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.
- Реализуйте указанный в индивидуальном задании метод построения популярных наборов предметов (Apriori/Eclat/Declat) (3 балла) или используйте метод BruteForce (0 баллов). Протестируйте корректность реализации алгоритма на учебном наборе данных из материалов лекции.
- При помощи указанного в индивидуальном задании метода или метода BruteForce постройте популярные наборы товаров с минимальной поддержкой, равной половине среднего количества покупок пяти наиболее популярных товаров. В случае нехватки вычислительных ресурсов для построения популярных наборов товаров оставьте в наборе данных транзакции с 10 наиболее популярными товарами и повторите расчет.
- Для какого-либо из полученных популярных наборов товаров постройте набор ассоциативных правил.
- Для построенного набора ассоциативных правил вычислите показатели: support, confidence, lift, leverage, conviction и выведите на экран.

- Алгоритм: Apriori
- День недели (поле order_dow таблицы orders): “6”
- Код департамента (поле department_id таблицы products): “11” personal care

---

In [ ]:

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

Извлечение из таблицы order_productstrain записей, соответствующих дню недели '6' (поле order_dow таблицы orders) и коду департамента '11' (поле department_id таблицы products). Вывожу первые 10 строк order_productstrain и для проверки правильности работы поля order_dow и department_id

In [2]:

```
for row in cursor.execute("""
    SELECT ord.order_id, ord.product_id, ord.add_to_cart_order,
    ord.reordered, o.order_dow, p.department_id
    FROM order_products__train ord, orders o, products p
    WHERE ord.order_id = o.order_id and ord.product_id = p.product_id
    and o.order_dow = "6" and p.department_id = "11"
    LIMIT 10
    """):
    print(row)
```

```
('2461523', '25997', '10', '1', '6', '11')
('537105', '19513', '3', '0', '6', '11')
('537105', '9047', '1', '0', '6', '11')
('1007973', '22237', '6', '0', '6', '11')
('1007973', '24467', '19', '1', '6', '11')
('1007973', '35339', '8', '0', '6', '11')
('1007973', '39220', '21', '0', '6', '11')
('1007973', '41782', '7', '0', '6', '11')
('1131395', '27544', '8', '1', '6', '11')
('524863', '14398', '2', '1', '6', '11')
```

Определение количества записей в полученном наборе и количества товаров (поле order_id таблицы order_products__train) в транзакциях набора

In [3]:

```
print("(количество записей в полученном наборе, ","количество уникальных товаров)")
for row in cursor.execute("""
    SELECT count(ord.order_id), count(DISTINCT ord.product_id)
    FROM order_products__train ord, orders o, products p
    WHERE ord.order_id = o.order_id and ord.product_id = p.product_id
    and o.order_dow = "6" and p.department_id = "11"
    """):
    print(row)

print("\nКоличество товаров в каждом заказе (первые 10 строк)")
for row in cursor.execute("""
    SELECT ord.order_id, count(ord.product_id)
    FROM order_products__train ord, orders o, products p
    WHERE ord.order_id = o.order_id and ord.product_id = p.product_id
    and o.order_dow = "6" and p.department_id = "11"
    GROUP BY ord.order_id
    LIMIT 10
    """):
    print(row)
```

```
(количество записей в полученном наборе,  количество уникальных товаров)
(3448, 1776)

Количество товаров в каждом заказе (первые 10 строк)
('1000640', 1)
('1000687', 1)
('1002166', 2)
('1003123', 1)
('1003564', 2)
('1005001', 1)
('100578', 2)
('1006311', 1)
('1007973', 5)
('1008188', 2)
```

Определение количества покупок (транзакций) для пяти наиболее популярных товаров в наборе

In [4]:

```
print("(id товара, название, количество покупок)")
for row in cursor.execute("""
    SELECT p.product_id, p.product_name, count(ord.product_id) as num
    FROM order_products__train ord, orders o, products p
    WHERE ord.order_id = o.order_id and ord.product_id = p.product_id
    and o.order_dow = "6" and p.department_id = "11"
    GROUP BY p.product_id
    ORDER BY num desc limit 5
    """):
    print(row)
```

```
(id товара, название, количество покупок)
('33493', 'Cotton Swabs', 36)
('12312', 'Lavender Hand Soap', 35)
('27544', 'Lemon Verbena Hand Soap', 29)
('29418', 'Fluoride-Free Antiplaque & Whitening Peppermint Toothpaste', 2
9)
('9047', 'Premium Epsom Salt', 23)
```

Строю 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 [5]:

```
tr_base = {}
for row in cursor.execute("""
    SELECT ord.order_id, p.product_name
    FROM order_products__train ord, orders o, products p
    WHERE ord.order_id = o.order_id and ord.product_id = p.product_id
    and o.order_dow = "6" and p.department_id = "11"
    """):
    if tr_base.get( int(row[0]) ):
        tr_base[ int(row[0]) ].add(row[1])
    else:
        tr_base[ int(row[0]) ] = {row[1]}
```

In [6]:

```
data = []
for id, products in tr_base.items():
    data.append([id, products])
print("Первые 5 строчек транзакционной базы данных:")
data[:5]
```

```
Первые 5 строчек транзакционной базы данных:
```

Out[6]:

```
[[2461523,
  {"Doctor Formulated Probiotics Once Daily Women's 50 Billion Guaranteed
Vegetarian Capsules"}],
 [537105,
  {'Moroccan Argan Oil + Argan Stem Cell Triple Moisture Conditioner',
   'Premium Epsom Salt'}],
 [1007973,
  {'Body Clear Body Wash',
   'Classic Original Scent Deodorant',
   'Foamy Sensitive Skin Shaving Cream',
   "Fusion Power Men's Razor Blade Refills",
   'Skin Relief Body Wash Fragrance Free'}],
 [1131395, {'Lemon Verbena Hand Soap'}],
 [524863, {'Yerbamate Lemon Energy Shot'}]]
```

In [7]:

```
I = set()
for row in cursor.execute("""
    SELECT DISTINCT p.product_name
    FROM order_products__train ord, orders o, products p
    WHERE ord.order_id = o.order_id and ord.product_id = p.product_id
    and o.order_dow = "6" and p.department_id = "11"
    """):
    I.add(row[0])
print("Первые 5 продуктов из списка всех неповторяющихся продуктов")
list(I)[:5]
```

```
Первые 5 продуктов из списка всех неповторяющихся продуктов
```

Out[7]:

```
['Goats Milk & Chai Soap',
 'Alka-Seltzer Plus Severe Sinus Day & Night Liquid',
 'Fresh Apricot Long Lasting Deodorant',
 'Gentle Skin Cleanser',
 'Organic Grass Fed Whey Protein Unflavored']
```

- Реализуйте метод построения популярных наборов предметов Apriori. Протестируйте корректность реализации алгоритма на учебном наборе данных из материалов лекции.

Алгоритм Apriori

(я пыталась...слишком долго пыталась...)

In [8]:

```
# множестов всех подмножеств
from itertools import chain, combinations
def powerset(iterable):
    xs = list(iterable)
    return chain.from_iterable(combinations(xs,n) for n in range(len(xs)+1))


# расчёт поддержки заданного набора предметов
def ComputeSupport( C, D ):
    supX = 0
    for _,itemset in D:
        for X in itemset:
            if X in C:
                supX += 1
    return supX

'''
# расчёт дерева префиксов
def ExtendPrefixTree( C ):
    Xab = set()
    for Xa in C:
        for Xb in C:
            if Xb >  Xa:
#??????????????????????????????
                if X in C:
            supX += 1
    return C

# Алгоритм Apriori
def Apriori( D, I, minsup ):
    F = []
    C = []
    for i in I:
        C = i
        sup[i] = 0

    k = 1 #уровень

    while C[k] != []:
        sup = ComputeSupport( set(C), D )
        for X in powerset( I ):
            if sup[X] >= minsup:
                F.append( [ X, sup ] )
            else
                C = C - X
        C[k+1] = ExtendPrefixTree(C[k])
        k += 1
return F
'''
```

Out[8]:

```
'\n# расчёт дерева префиксов\ndef ExtendPrefixTree( C ):\n    Xab = set()
\n    for Xa in C:\n        for Xb in C:\n            if Xb >  Xa:\n
#??????????????????????????????\n                if X in C:\n
supX += 1\n    return C\n\n# Алгоритм Apriori\ndef Apriori( D, I, minsup
):\n    F = []\n    C = []\n    for i in I:\n        C = i\n        sup[i]
= 0\n    \n    k = 1 #уровень\n    \n    while C[k] != []:\n        sup =
ComputeSupport( set(C), D )\n        for X in powerset( I ):\n
if sup[X] >= minsup:\n                F.append( [ X, sup ] )\n
else \n                C = C - X\n        C[k+1] = ExtendPrefixTree(C[k])
\n        k += 1\nreturn F\n'
```

Алгоритм Brute Force

In [8]:

```
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))

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

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

- При помощи метода Apriori или Brute Force постройте популярные наборы товаров с минимальной поддержкой, равной половине среднего количества покупок пяти наиболее популярных товаров. В случае нехватки вычислительных ресурсов для построения популярных наборов товаров оставьте в наборе данных транзакции с 10 наиболее популярными товарами и повторите расчет.

Вычисление минимальной поддержки

In [9]:

```
c = []
for row in cursor.execute("""
    SELECT count(ord.product_id) as num
    FROM order_products__train ord, orders o, products p
    WHERE ord.order_id = o.order_id and ord.product_id = p.product_id
    and o.order_dow = "6" and p.department_id = "11"
    GROUP BY p.product_id
    ORDER BY num desc limit 5
    """):
    c += row
print(c)

sum = 0
for i in c:
    sum += i
minsup = (sum / 5) / 2
print("Минимальная поддержка = ", minsup)
```

```
[36, 35, 29, 29, 23]
Минимальная поддержка =  15.2
```

Построение методом Brute Force популярных наборов с минимальной поддержкой, равной половине среднего количества покупок пяти наиболее популярных товаров

In [10]:

```
print("Популярные наборы при minsup = ", minsup)
for itemset in BruteForce( data, I, minsup ):
    print(itemset)
```

```
Популярные наборы при minsup =  15.2
('Fluoride-Free Antiplaque & Whitening Peppermint Toothpaste',) 29
('Deep Moisture Body Wash',) 18
('Makeup Remover Cleansing Towelettes',) 19
('Organic Ground Flaxseed',) 19
('Vanilla Whey Protein Powder',) 16
('Hydrogen Peroxide',) 16
('Cotton Swabs',) 36
('Epsom Salt',) 16
('Lavender Hand Soap',) 35
('Clean Day Basil Hand Soap',) 20
('Premium Epsom Salt',) 23
('Lemon Verbena Hand Soap',) 29

---------------------------------------------------------------------------
-
KeyboardInterrupt                         Traceback (most recent call las
t)
<ipython-input-10-8815da177172> in <module>
      1 print("Популярные наборы при minsup = ", minsup)
----> 2 for itemset in BruteForce( data, I, minsup ):
      3     print(itemset)

<ipython-input-8-d039e63cb960> in BruteForce(D, I, minsup)
     16     for X in powerset( I ):
     17         if len( X ) > 0:
---> 18             supX = ComputeSupport( set( X ), D )
     19             if supX >= minsup:
     20                 print(X, supX)

<ipython-input-8-d039e63cb960> in ComputeSupport(X, D)
      8     supX = 0
      9     for _,itemset in D:
---> 10         if X.issubset( itemset ):
     11             supX += 1
     12     return supX

KeyboardInterrupt:
```

Спустя 30 минут ожидания выполнения алгоритма, оставляю в базе только записи с заказами, в которых присутствует хотябы один из 10 самых популярных продуктов

(Процесс не заканчивается, видно только наборы из одного элемента)

In [11]:

```
print("топ 10 продуктов\n(id товара, название, количество покупок)")
pop = cursor.execute("""
    SELECT p.product_id, p.product_name, count(ord.product_id) as num
    FROM order_products__train ord, orders o, products p
    WHERE ord.order_id = o.order_id and ord.product_id = p.product_id
    and o.order_dow = "6" and p.department_id = "11"
    GROUP BY p.product_id
    ORDER BY num desc limit 10
    """)
ps = set()
for row in pop:
    print(row)
    ps.add(row[1])
```

```
топ 10 продуктов
(id товара, название, количество покупок)
('33493', 'Cotton Swabs', 36)
('12312', 'Lavender Hand Soap', 35)
('27544', 'Lemon Verbena Hand Soap', 29)
('29418', 'Fluoride-Free Antiplaque & Whitening Peppermint Toothpaste', 2
9)
('9047', 'Premium Epsom Salt', 23)
('39162', 'Clean Day Basil Hand Soap', 20)
('2309', 'Organic Ground Flaxseed', 19)
('280', 'Makeup Remover Cleansing Towelettes', 19)
('33439', 'Deep Moisture Body Wash', 18)
('14650', 'Vanilla Whey Protein Powder', 16)
```

In [12]:

```
ndata = []
nI = set()
for id, products in tr_base.items():
    for p in products:
        if p in ps:
            ndata.append([id, products])
            for i in products:
                nI.add(i)
            break

print("Стало ", len(ndata))
print('-----------')
print("Было ", len(tr_base))

list(ndata)[:5]
```

```
Стало  237
-----------
Было  2336
```

Out[12]:

```
[[537105,
  {'Moroccan Argan Oil + Argan Stem Cell Triple Moisture Conditioner',
   'Premium Epsom Salt'}],
 [1131395, {'Lemon Verbena Hand Soap'}],
 [2339119,
  {'Natural Chocolate Flavor Whey Protein Powder',
   'Organic Rice Probiotic Drink Blueberry',
   'Vanilla Whey Protein Powder'}],
 [1668533, {'Makeup Remover Cleansing Towelettes'}],
 [1702162, {'Clean Day Basil Hand Soap'}]]
```

Построение методом Brute Force популярных наборов с минимальной поддержкой, равной половине среднего количества покупок пяти наиболее популярных товаров, на уменьшенных данных

In [13]:

```
print("Популярные наборы при minsup = ", minsup)
for itemset in BruteForce( ndata, nI, minsup ):
    print(itemset)
```

```
Популярные наборы при minsup =  15.2
('Lavender Hand Soap',) 35
('Premium Epsom Salt',) 23
('Cotton Swabs',) 36
('Lemon Verbena Hand Soap',) 29
('Deep Moisture Body Wash',) 18
('Makeup Remover Cleansing Towelettes',) 19
('Organic Ground Flaxseed',) 19
('Clean Day Basil Hand Soap',) 20
('Vanilla Whey Protein Powder',) 16
('Fluoride-Free Antiplaque & Whitening Peppermint Toothpaste',) 29

---------------------------------------------------------------------------
-
KeyboardInterrupt                         Traceback (most recent call las
t)
<ipython-input-13-ad0259d4a7d7> in <module>
      1 print("Популярные наборы при minsup = ", minsup)
----> 2 for itemset in BruteForce( ndata, nI, minsup ):
      3     print(itemset)

<ipython-input-8-d039e63cb960> in BruteForce(D, I, minsup)
     16     for X in powerset( I ):
     17         if len( X ) > 0:
---> 18             supX = ComputeSupport( set( X ), D )
     19             if supX >= minsup:
     20                 print(X, supX)

<ipython-input-8-d039e63cb960> in ComputeSupport(X, D)
      7 def ComputeSupport( X, D ):
      8     supX = 0
----> 9     for _,itemset in D:
     10         if X.issubset( itemset ):
     11             supX += 1

KeyboardInterrupt:
```

—> Вложение в конце

- Для какого-либо из полученных популярных наборов товаров постройте набор ассоциативных правил.

In [14]:

```
#F_bf, _ = BruteForce( data, I, minsup )[-1]
F_bf = ('Cotton Swabs',)
```

In [15]:

```
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( data, F_bf, 0.9)
```

Out[15]:

```
[]
```

В наборе товаров только один элемент, так что набор ассоциативных правил пуст. Сделаю следующий пункт по учебному набору данных.

In [16]:

```
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"} ],
]
F_set = ('A', 'E', 'D', 'B')

rules = AssociationRules(train, F_set, 0.9)
rules
```

Out[16]:

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

In [17]:

```
#оставляю только наборы
r = []
for i in rules:
    r.append([set(i[0]), set(i[1])])
r
```

Out[17]:

```
[[{'B', 'D', 'E'}, {'A'}],
 [{'A', 'B', 'D'}, {'E'}],
 [{'A', 'D', 'E'}, {'B'}],
 [{'D', 'E'}, {'A', 'B'}],
 [{'A', 'D'}, {'B', 'E'}]]
```

Для каждого построенного набора ассоциативных правил вычислияю показатели: support, confidence, lift, leverage, conviction

In [18]:

```
for rules in r:
    print("\nSupport = ", ComputeSupport(rules[0].union(rules[1]), train))

    rsupXY = ComputeSupport(rules[0].union(rules[1]), train)/len(train)
    rsupX = ComputeSupport(rules[0], train)/len(train)
    conf = rsupXY/rsupX
    print("Confidence = ", conf)

    rsupY = ComputeSupport(rules[1], train)/len(train)
    lift = conf/rsupY
    print("Lift = ", lift)

    lev = rsupXY - rsupX * rsupY
    print("Leverage = ", lev)

    conv = (1 - rsupY)/(1 - conf) if conf != 1 else "undefined"
    print("Conviction = ", conv)
```

```
Support =  3
Confidence =  1.0
Lift =  1.5
Leverage =  0.16666666666666669
Conviction =  undefined

Support =  3
Confidence =  1.0
Lift =  1.2
Leverage =  0.08333333333333331
Conviction =  undefined

Support =  3
Confidence =  1.0
Lift =  1.0
Leverage =  0.0
Conviction =  undefined

Support =  3
Confidence =  1.0
Lift =  1.5
Leverage =  0.16666666666666669
Conviction =  undefined

Support =  3
Confidence =  1.0
Lift =  1.2
Leverage =  0.08333333333333331
Conviction =  undefined
```

Пункт расчёта популярных наборов методом Brute Force особенно затратный, я сдалась и перенесла код из Jupyter Notebooks в обычный py файл. Но программа и тут быстро выдала наборы, состоящие из одиночных элементов и молчаливо продолжила думать. На меньшей выборке всё те же 10 самых популярных продуктов, только расположенные в другом порядке.

Я закончила лабораторную лишь ночью, т е уже просрочила сдачу. Так что решила оставить программу работать на ночь в надежде, что позже появятся наборы размером больше одного элемента. Не появились.

Скриншот выполнения. Последний блок — расчёт популярных наборов, остальное дублирует все предшествующие шаги.