РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Дисциплина: Интеллектуальный анализ данных

Студент: Генералов Даниил

Группа: НПИбд-01-21

Москва 2024

Вариант № 28

Алгоритм: Apriori

День недели (поле order_dow таблицы orders): “6”

Код департамента (поле department_id таблицы products): “2”

Запрос: определить список товаров, которые были приобретены ровно два раза

Показатель оценки ассоциативных правил: лифт (lift)

In [1]:
```
TARGET_ORDER_DOW = 6
TARGET_ORDER_DEPT = 2
```

1. открыть базу данных

In [2]:
```
import os
print(os.getcwd())

if not os.getcwd().endswith('hw2'):
    os.chdir('hw2')

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

```
/home/danya/rudn-year4-data-mining/hw2
```

In [3]:
```
c = db.cursor()

c.execute("SELECT name FROM sqlite_master")
print(c.fetchall())
```

```
[('aisles',), ('products',), ('departments',), ('orders',), ('order_products
__train',)]
```

2. загрузить данные из таблицы в dataframe

In [4]:
```
import pandas

departments = pandas.read_sql('SELECT * FROM departments', db)
departments.head()
```

Out[4]:

	department_id	department
0	1	frozen
1	2	other
2	3	bakery
3	4	produce
4	5	alcohol

In [5]:
```
departments.set_index('department_id', inplace=True)
```

In [6]:
```
products = pandas.read_sql('SELECT * FROM products', db)
products.head()
```

Out[6]:

	product_id	product_name	aisle_id	department_id
0	1	Chocolate Sandwich Cookies	61	19
1	2	All-Seasons Salt	104	13
2	3	Robust Golden Unsweetened Oolong Tea	94	7
3	4	Smart Ones Classic Favorites Mini Rigatoni Wit...	38	1
4	5	Green Chile Anytime Sauce	5	13

In [7]:
```
products.set_index('product_id', inplace=True)
```

In [8]:
```
orders = pandas.read_sql(
    'SELECT op.* FROM order_products__train op JOIN orders o ON op.order_id
    db,
    params=(TARGET_ORDER_DOW, TARGET_ORDER_DEPT))
```

In [9]:
```
orders.head()
```

Out[9]:

	order_id	product_id	add_to_cart_order	reordered
0	1447810	26756	18	1
1	112127	1892	10	0
2	2947672	38662	7	0
3	1594674	32115	8	1
4	19097	38662	31	1

In [10]:
```
print("Total rows:", len(orders))
print("Transactions:", len(orders.order_id.unique()))
print("Products:", len(orders.product_id.unique()))
```

```
Total rows: 285
Transactions: 274
Products: 145
```

3. запрос к dataframe

Такие заказы, которые покупают товары, которые присутствуют в ровно двух заказах

In [11]:
```
idxs = orders.groupby('product_id').size().eq(2)
orders[orders.product_id.isin(idxs.index[idxs])]
```

Out[11]:

	order_id	product_id	add_to_cart_order	reordered
14	2806203	30852	13	0
18	3056268	1724	3	1
32	371948	12958	1	1
33	2527067	43985	8	0
44	2231020	21046	6	0
60	1758449	7004	4	1
61	1189708	7004	2	1
65	1536712	44405	2	0
77	46285	20066	4	0
93	1117161	11903	4	1
100	541023	15137	7	1
101	869690	31777	4	0
104	2652915	46507	1	0
106	2731913	8960	5	0
108	250934	28522	4	0
109	2929827	24599	11	0
117	2408539	1724	12	0
118	1492536	11903	3	1
123	2902681	8960	22	0
127	961207	45166	11	1
142	3406661	43985	4	0
146	624792	40301	2	1
148	2376161	28984	1	0
162	1367373	2334	3	0
167	10371	2334	1	0
173	2267310	20599	14	0
174	2611954	30852	1	0
182	3091662	21046	2	0
187	264891	24599	8	0
190	882713	44405	6	0
195	3393984	45166	6	0
208	1568214	46507	21	1
210	2082049	20066	15	0

	order_id	product_id	add_to_cart_order	reordered
214	194281	12958	1	0
223	672415	40301	5	0
224	3363673	31777	4	0
241	1886275	39192	4	1
245	1389485	28522	16	0
257	2404202	15137	7	1
266	1340483	28984	10	0
275	1008640	39192	6	1
281	2949070	20599	6	1

4. транзакционная база

In [42]:
```
# build transactional dataframe, mapping product_id to product name

xacts = orders.merge(products, left_on='product_id', right_index=True).drop(
```

In [43]:
```
xacts
```

Out[43]:
```
order_id
1002424                                              [Rye Flour]
1008640                          [SleepTabs Nighttime Sleep Aid]
1009412    [Moderate Absorbency Long Length Incontinence ...
10371                                    [Organic Garam Masala]
1051349                                [Cherry Vanilla Granola]
                                      ...
954716     [Melatonin, Fast Dissolve, 5 mg, Tablets, Natu...
95948                             [Early Result Pregnancy Test]
959540     [Infants Pain Reliever and Fever Reducer Berry...
961207                                   [French Green Lentils]
972147                                 [Roasted Almond Butter]
Name: product_name, Length: 274, dtype: object
```

In [44]:
```
max_items_xact = xacts.apply(len).idxmax()
max_items_xact
```

Out[44]:
```
'1383780'
```

In [76]:
```
list(xacts[max_items_xact])
```

Out[76]:
```
['Arugula Rocket Salad', 'Parsley Italian Dark Green Flat']
```

5. бинарная транзакционная база

In [47]:
```
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_fit = te.fit(xacts).transform(xacts)
te_fit
```

Out[47]:
```
array([[False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False]])
```

In [48]:
```
binary_df = pandas.DataFrame(te_fit, columns=te.columns_)
binary_df
```

Out[48]:

	100% Pure Eucalyptus	5-HTP 100 Mg Vegetarian Capsules	93/7 Ground Beef	AA Rechargeable Nickel Metal Hydride Batteries	All Purpose Precision Tip 2 Pack	Aromatic Bitters	Aru Ro S
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
269	False	False	False	False	False	False	
270	False	False	False	False	False	False	
271	False	False	False	False	False	False	
272	False	False	False	False	False	False	
273	False	False	False	False	False	False	

274 rows × 145 columns

In [49]:
```
product_counts = binary_df.sum().sort_values(ascending=False)
top_products = product_counts.head(3)
print(f"Top products: \n{top_products}")
```

```
Top products:
Roasted Almond Butter                    29
Light CocoWhip! Coconut Whipped Topping  19
Roasted Unsalted Almonds                 13
dtype: int64
```

6. поиск популярных наборов

In [66]:
```
from mlxtend.frequent_patterns import apriori

absolute_min_support = 3
relative_min_support = absolute_min_support / len(binary_df)

frequent_itemsets = apriori(binary_df, min_support=relative_min_support, use
```

In [67]:
```
frequent_itemsets['itemsets'].apply(len).max()  # == 1!
```

Out[67]:
```
np.int64(1)
```

(значит, нет популярных наборов больше одного элемента)

In [68]:
```
frequent_itemsets
```

Out[68]:

	support	itemsets
0	0.018248	(93/7 Ground Beef)
1	0.010949	(All Purpose Precision Tip 2 Pack)
2	0.010949	(Black Chia Seeds)
3	0.010949	(Cherry Vanilla Granola)
4	0.010949	(Classic Vanilla Coffee Creamer)
5	0.014599	(Coconut Almond Granola)
6	0.018248	(Creamer)
7	0.010949	(Deluxe Nut Mix)
8	0.010949	(Early Result Pregnancy Test)
9	0.010949	(Falafel)
10	0.010949	(Infants Pain Reliever and Fever Reducer Berry...
11	0.029197	(Kiwifruit)
12	0.069343	(Light CocoWhip! Coconut Whipped Topping)
13	0.010949	(Liquid Teething Relief)
14	0.010949	(Oral Electrolyte Powder Assorted Flavors)
15	0.010949	(Organic Coco Whip)
16	0.021898	(Pierogi Potato & Cheese)
17	0.018248	(Pinot Noir Rose)
18	0.105839	(Roasted Almond Butter)
19	0.014599	(Roasted Salted Pistachios)
20	0.047445	(Roasted Unsalted Almonds)
21	0.010949	(SleepGels Nighttime Sleep Aid)
22	0.018248	(Walnuts)
23	0.014599	(Whole Bay Leaves)

7. ассоциативные правила

In [79]:
```
from mlxtend.frequent_patterns import association_rules

rules = association_rules(frequent_itemsets)
```

In [80]:
```
rules
```

Out[80]:

antecedents	consequents	antecedent support	consequent support	support	confidence	li

Из-за того, что в этом датасете довольно мало строк, мы не можем найти никаких ассоциативных правил. Поэтому выбираем случайный другой датасет.

In [86]:
```
import random

TARGET_ORDER_DOW = random.choice(range(7))
TARGET_ORDER_DEPT = random.choice(departments.index)
print(f'{TARGET_ORDER_DOW=}, {TARGET_ORDER_DEPT=}')
orders = pandas.read_sql(
    'SELECT op.* FROM order_products__train op JOIN orders o ON op.order_id
    db,
    params=(TARGET_ORDER_DOW, TARGET_ORDER_DEPT))
print(len(orders))
```

```
TARGET_ORDER_DOW=3, TARGET_ORDER_DEPT='18'
1774
```

In [94]:
```
xacts = orders.merge(products, left_on='product_id', right_index=True).drop(

te = TransactionEncoder()
te_fit = te.fit(xacts).transform(xacts)
binary_df = pandas.DataFrame(te_fit, columns=te.columns_)

product_counts = binary_df.sum().sort_values(ascending=False)
top_products = product_counts.head(3)
print(f"Top products: \n{top_products}")

absolute_min_support = 3
relative_min_support = absolute_min_support / len(binary_df)
frequent_itemsets = apriori(binary_df, min_support=relative_min_support, use
maxidx = frequent_itemsets['itemsets'].apply(len).idxmax()

frequent_itemsets['itemsets'][maxidx]
```

```
Top products:
Baby Food Stage 2 Blueberry Pear & Purple Carrot    39
Gluten Free SpongeBob Spinach Littles               38
Spinach Peas & Pear Stage 2 Baby Food               36
dtype: int64
```

Out[94]:
```
frozenset({'Apple Blueberry Fruit Yogurt Smoothie',
           'Organic Fruit Yogurt Smoothie Mixed Berry',
           'Organic Strawberry Banana Fruit Yogurt Smoothie'})
```

In [100…
```
rules = association_rules(frequent_itemsets, metric='lift')
len(rules)
```

Out[100… 268

Поскольку в задании не задан порог уверенности, его приходится выбрать случайно.

In [108…
```
confidence_threshold = random.choice(rules.confidence)
confidence_threshold
```

Out[108…
```
np.float64(0.75)
```

In [110…
```
rules = association_rules(frequent_itemsets, metric='confidence', min_thresh
rules.head()
```

Out[110…

	antecedents	consequents	antecedent support	consequent support	support	confidence
0	(Baby Food Pears Squash)	(Apples, Pumpkin & Carrots Organic Baby Food)	0.006477	0.009067	0.005181	0.80
1	(Chunky Blend Vegetable Beef Pilaf Baby Food)	(Organic Stage 3 Spaghetti with Cheese 6 Ounce...	0.005181	0.006477	0.003886	0.75
2	(Chunky Blend Vegetable Beef Pilaf Baby Food)	(Tender Chicken & Stars Stage 3)	0.005181	0.010363	0.005181	1.00
3	(Vegetable Chicken Soup Stage 3)	(Chunky Blend Vegetable Beef Pilaf Baby Food)	0.003886	0.005181	0.003886	1.00
4	(Chunky Blend Vegetable Beef Pilaf Baby Food)	(Vegetable Chicken Soup Stage 3)	0.005181	0.003886	0.003886	0.75

8. поиск лучших ассоциативных правил

In [113…
```
rules.sort_values('lift', ascending=False).head(10)
```

Out[113…

	antecedents	consequents	antecedent support	consequent support	support	confidence
3	(Vegetable Chicken Soup Stage 3)	(Chunky Blend Vegetable Beef Pilaf Baby Food)	0.003886	0.005181	0.003886	1.00
4	(Chunky Blend Vegetable Beef Pilaf Baby Food)	(Vegetable Chicken Soup Stage 3)	0.005181	0.003886	0.003886	0.75
28	(Vegetable Chicken Soup Stage 3)	(Tender Chicken & Stars Stage 3, Chunky Blend ...	0.003886	0.005181	0.003886	1.00
27	(Chunky Blend Vegetable Beef Pilaf Baby Food)	(Tender Chicken & Stars Stage 3, Vegetable Chi...	0.005181	0.003886	0.003886	0.75
24	(Tender Chicken & Stars Stage 3, Chunky Blend ...	(Vegetable Chicken Soup Stage 3)	0.005181	0.003886	0.003886	0.75
25	(Tender Chicken & Stars Stage 3, Vegetable Chi...	(Chunky Blend Vegetable Beef Pilaf Baby Food)	0.003886	0.005181	0.003886	1.00
35	(Organic Stage 3 Spaghetti with Cheese 6 Ounce...	(Tender Chicken & Stars Stage 3, Organic Stage...	0.006477	0.005181	0.005181	0.80
32	(Tender Chicken & Stars Stage 3, Organic Stage...	(Organic Stage 3 Spaghetti with Cheese 6 Ounce...	0.005181	0.006477	0.005181	1.00
15	(Apples, Pumpkin & Carrots Organic Baby Food, ...	(Baby Food Pears Squash)	0.005181	0.006477	0.003886	0.75
23	(Chunky Blend Vegetable Beef Pilaf Baby Food)	(Tender Chicken & Stars Stage 3, Organic Stage...	0.005181	0.006477	0.003886	0.75