

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №3**  
**по дисциплине «Машинное обучение»**  
**Тема: Частотный анализ**

Студент гр. 1310

---

Комаров Д. Е.

Преподаватель

---

Жангиров Т.Р.

Санкт-Петербург

2025

## **Постановка задачи**

Цель работы: ознакомление с методами частотного анализа из библиотеки MLxtend.

## **Выполнение лабораторной работы**

### **Загрузка данных**

Для выполнения лабораторной работы используем набор данных «Random Shopping cart». Данный набор содержит данные о 1139 покупателях, покупавших 38 видов товаров.

Сгруппируем товары, принадлежащие одному покупателю, и отбросим такие данные как время покупки и id покупателя. Получим список транзакций. Каждая транзакция представляет собой список товаров, купленных одним покупателем.

### **Подготовка данных**

Поскольку список транзакций непригоден для дальнейшего анализа напрямую, закодируем данные в виде матрицы, строки которой будут являться транзакциями, столбцы – товарами. Если в транзакции будет присутствовать товар, то на пересечении строки транзакции и столбца товара будет 1, иначе 0.

Фрагмент полученной матрицы представлен на рисунке 1.

	all-purpose	aluminum foil	bagels	beef	butter	cereals	...	sugar	toilet paper	paper	tortillas	vegetables	waffles	yogurt
0	True	True	False	True	True	False	...	False	False	False	False	True	False	True
1	False	True	False	False	False	True	...	False	True	True	True	True	True	True
2	False	False	True	False	False	True	...	False	True	False	True	False	False	False
3	True	False	False	False	False	True	...	False	True	False	False	False	False	False
4	True	False	False	False	False	False	...	False	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1134	True	False	False	True	False	True	...	True	False	False	False	False	False	False
1135	False	False	False	False	False	True	...	False	False	False	True	False	False	False
1136	False	False	True	True	False	False	...	True	False	True	True	False	True	True
1137	True	False	False	True	False	False	...	True	True	False	True	True	True	True
1138	False	False	False	False	False	False	...	False	False	False	True	False	False	False

Рисунок 1 – Фрагмент матрицы транзакций

### **Ассоциативный анализ с использованием алгоритма Apriori**

Применим алгоритм Apriori с минимальным уровнем поддержки 0.3. В результате получим список из 51 набора. Данный список представлен на рисунке 2.

	support	itemsets	length
0	0.374890	(all-purpose)	1
1	0.384548	(aluminum foil)	1
2	0.385426	(bagels)	1
3	0.374890	(beef)	1
4	0.367867	(butter)	1
5	0.395961	(cereals)	1
6	0.390694	(cheeses)	1
7	0.379280	(coffee/tea)	1
8	0.388938	(dinner rolls)	1
9	0.388060	(dishwashing liquid/detergent)	1
10	0.389816	(eggs)	1
11	0.352941	(flour)	1
12	0.370500	(fruits)	1
13	0.345917	(hand soap)	1
14	0.398595	(ice cream)	1
15	0.375768	(individual meals)	1
16	0.376646	(juice)	1
17	0.371378	(ketchup)	1
18	0.378402	(laundry detergent)	1
19	0.395083	(lunch meat)	1
20	0.380158	(milk)	1
21	0.375768	(mixes)	1
22	0.362599	(paper towels)	1
23	0.371378	(pasta)	1
24	0.355575	(pork)	1
25	0.421422	(poultry)	1
26	0.367867	(sandwich bags)	1
27	0.349429	(sandwich loaves)	1
28	0.368745	(shampoo)	1
29	0.379280	(soap)	1
30	0.390694	(soda)	1
31	0.373134	(spaghetti sauce)	1
32	0.360843	(sugar)	1
33	0.378402	(toilet paper)	1
34	0.369622	(tortillas)	1
35	0.739245	(vegetables)	1
36	0.394205	(waffles)	1
37	0.384548	(yogurt)	1
38	0.310799	(vegetables, aluminum foil)	2
39	0.300263	(bagels, vegetables)	2
40	0.310799	(cereals, vegetables)	2
41	0.309043	(cheeses, vegetables)	2
42	0.308165	(dinner rolls, vegetables)	2
43	0.306409	(dishwashing liquid/detergent, vegetables)	2
44	0.326602	(eggs, vegetables)	2
45	0.302897	(ice cream, vegetables)	2
46	0.309043	(laundry detergent, vegetables)	2
47	0.311677	(lunch meat, vegetables)	2
48	0.331870	(poultry, vegetables)	2
49	0.305531	(soda, vegetables)	2
50	0.315189	(waffles, vegetables)	2
51	0.319579	(yogurt, vegetables)	2

Рисунок 2 – Список часто встречающихся наборов, полученный алгоритмом Apriori, с минимальным уровнем поддержки 0.3

Как можно увидеть из рисунка 2, количество элементов в полученных наборах варьируется от 1 до 2, а уровень поддержки от 0.3 До 0.74.

Применим, алгоритм Apriori еще раз, задав максимальный размер набора равным 1. В результате получим часть наборов из рисунка 2, длина которых

равна 1. Получилось 37 наборов. Результат выполнения алгоритма Apriori с заданием максимальной длины набора равной 1 представлен на рисунке 3.

	support	itemsets
0	0.374890	(all-purpose)
1	0.384548	(aluminum foil)
2	0.385426	(bagels)
3	0.374890	(beef)
4	0.367867	(butter)
5	0.395961	(cereals)
6	0.390694	(cheeses)
7	0.379280	(coffee/tea)
8	0.388938	(dinner rolls)
9	0.388060	(dishwashing liquid/detergent)
10	0.389816	(eggs)
11	0.352941	(flour)
12	0.370500	(fruits)
13	0.345917	(hand soap)
14	0.398595	(ice cream)
15	0.375768	(individual meals)
16	0.376646	(juice)
17	0.371378	(ketchup)
18	0.378402	(laundry detergent)
19	0.395083	(lunch meat)
20	0.380158	(milk)
21	0.375768	(mixes)
22	0.362599	(paper towels)
23	0.371378	(pasta)
24	0.355575	(pork)
25	0.421422	(poultry)
26	0.367867	(sandwich bags)
27	0.349429	(sandwich loaves)
28	0.368745	(shampoo)
29	0.379280	(soap)
30	0.390694	(soda)
31	0.373134	(spaghetti sauce)
32	0.360843	(sugar)
33	0.378402	(toilet paper)
34	0.369622	(tortillas)
35	0.739245	(vegetables)
36	0.394205	(waffles)
37	0.384548	(yogurt)

Рисунок 3 – Список часто встречающихся наборов с минимальным уровнем поддержки 0.3 и максимальной длиной набора 1

Выведем оставшиеся наборы, а именно те, у которых длина равна 2. Получилось 14 наборов. Наборы длиной 2 представлены на рисунке 4.

	support	itemsets	length
38	0.310799	(vegetables, aluminum foil)	2
39	0.300263	(bagels, vegetables)	2
40	0.310799	(cereals, vegetables)	2
41	0.309043	(cheeses, vegetables)	2
42	0.308165	(dinner rolls, vegetables)	2
43	0.306409	(dishwashing liquid/detergent, vegetables)	2
44	0.326602	(eggs, vegetables)	2
45	0.302897	(ice cream, vegetables)	2
46	0.309043	(laundry detergent, vegetables)	2
47	0.311677	(lunch meat, vegetables)	2
48	0.331870	(poultry, vegetables)	2
49	0.305531	(soda, vegetables)	2
50	0.315189	(waffles, vegetables)	2
51	0.319579	(yogurt, vegetables)	2

Рисунок 4 – Список часто встречающихся наборов с минимальным уровнем поддержки 0.3 и длиной набора 2

Поэкспериментируем алгоритм Apriori для различных минимальных уровней поддержки и построим график зависимости количества часто встречающихся наборов от уровня минимальной поддержки. Ради наглядности график был разделен на 2 части. На рисунке 5 представлен данный график на промежутке [0.05;0.3].

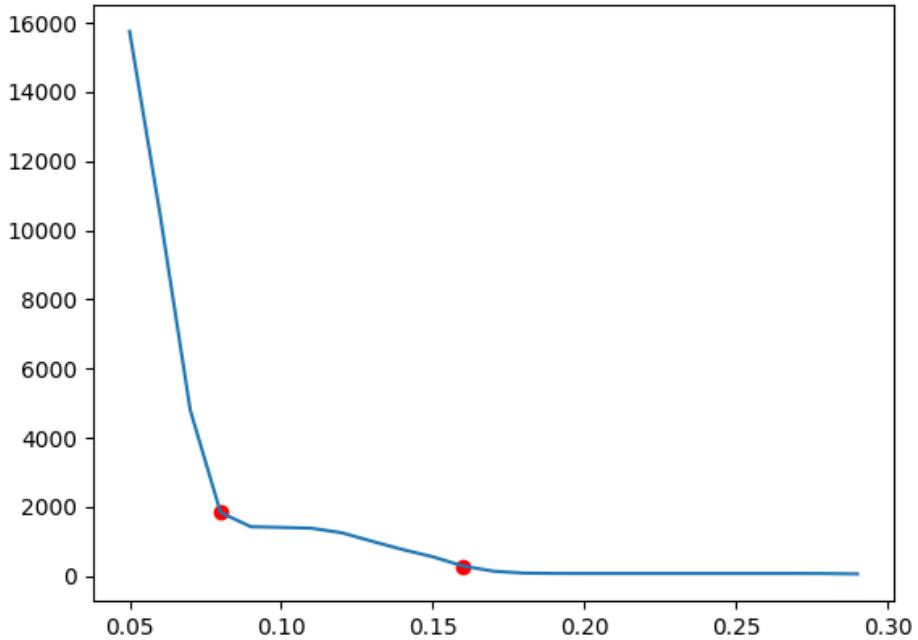


Рисунок 5 – График зависимости количества часто встречающихся наборов от уровня минимальной поддержки на промежутке [0.05;0.3]

График зависимости количества часто встречающихся наборов от уровня минимальной поддержки на промежутке  $[0.3;1]$  представлен на рисунке 6.

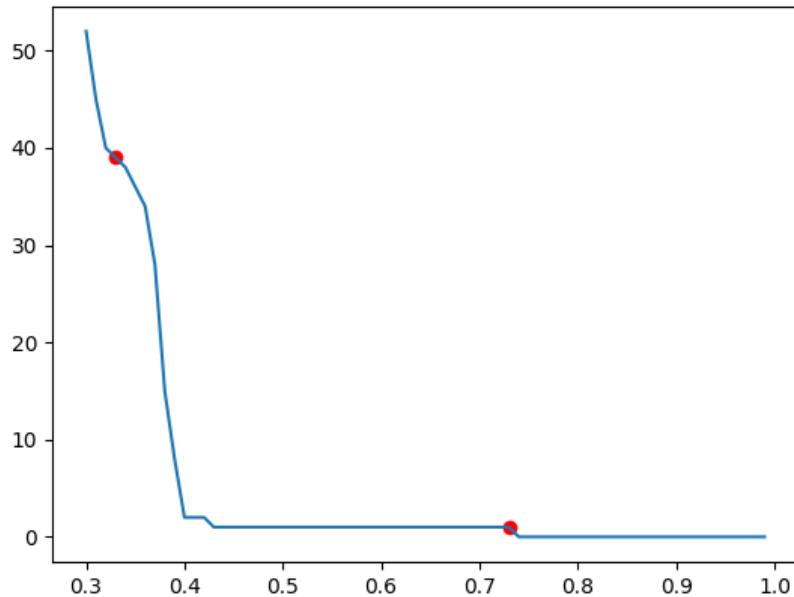


Рисунок 6 – График зависимости количества часто встречающихся наборов от уровня минимальной поддержки на промежутке  $[0.3;1]$

Из рисунков 5 и 6 следует, что, при возрастании минимального уровня поддержки, количество часто встречающихся наборов монотонно убывает. Красными на графиках точками изображены минимальные уровни поддержки, при которых перестают генерироваться наборы определенного размера. Так, при уровне поддержки 0.73 перестают генерироваться наборы длины 1, при уровне поддержки 0.33 – длины 2, при уровне поддержки 0.16 – длины 3 и при уровне поддержки 0.08 – длины 4. Заметим, что именно в этих точках график имеет резкие изломы.

Построим датасет только из тех элементов, которые попадают в наборы размером 1 при уровне поддержки 0.38 и приведем его к формату матрицы, пригодной для дальнейшего анализа. Таких элементов оказалось 15. Проведем ассоциативный анализ при уровне поддержки 0.3 для нового датасета. Было получено 27 наборов. Результат представлен на рисунке 7.

	support	itemsets
0	0.384548	(aluminum foil)
1	0.385426	(bagels)
2	0.395961	(cereals)
3	0.390694	(cheeses)
4	0.388938	(dinner rolls)
5	0.388060	(dishwashing liquid/detergent)
6	0.389816	(eggs)
7	0.398595	(ice cream)
8	0.395083	(lunch meat)
9	0.380158	(milk)
10	0.421422	(poultry)
11	0.390694	(soda)
12	0.739245	(vegetables)
13	0.394285	(waffles)
14	0.384548	(yogurt)
15	0.310799	(vegetables, aluminum foil)
16	0.300263	(bagels, vegetables)
17	0.310799	(cereals, vegetables)
18	0.309043	(vegetables, cheeses)
19	0.308165	(vegetables, dinner rolls)
20	0.306409	(vegetables, dishwashing liquid/detergent)
21	0.326602	(eggs, vegetables)
22	0.302897	(ice cream, vegetables)
23	0.311677	(vegetables, lunch meat)
24	0.331870	(poultry, vegetables)
25	0.305531	(vegetables, soda)
26	0.315189	(vegetables, waffles)
27	0.319579	(yogurt, vegetables)

Рисунок 7 – Список часто встречающихся наборов с минимальным уровнем поддержки 0.3 для нового датасета

Как можно увидеть из рисунка 7, для всех наборов длины 1, уровень поддержки действительно больше или равен 0.38. Для больших наборов уровень поддержки больше или равен 0.3.

Проведем ассоциативный анализ при уровне поддержки 0.15 для нового датасета и выведем все наборы, размер которых больше 1 и в котором есть 'yogurt' или 'waffles'. Результат представлен на рисунке 8.

	support	itemsets
27	0.169447	(aluminum foil, waffles)
28	0.177349	(yogurt, aluminum foil)
40	0.159789	(bagels, waffles)
41	0.162423	(bagels, yogurt)
52	0.160667	(cereals, waffles)
53	0.172081	(yogurt, cereals)
63	0.172959	(cheeses, waffles)
64	0.172081	(cheeses, yogurt)
73	0.169447	(dinner rolls, waffles)
74	0.166813	(yogurt, dinner rolls)
82	0.175593	(dishwashing liquid/detergent, waffles)
83	0.158033	(yogurt, dishwashing liquid/detergent)
90	0.169447	(eggs, waffles)
91	0.174715	(eggs, yogurt)
97	0.172959	(ice cream, waffles)
98	0.156277	(ice cream, yogurt)
103	0.184372	(waffles, lunch meat)
104	0.161545	(yogurt, lunch meat)
108	0.167691	(milk, yogurt)
111	0.166813	(poultry, waffles)
112	0.180860	(poultry, yogurt)
114	0.177349	(soda, waffles)
115	0.167691	(soda, yogurt)
116	0.315189	(vegetables, waffles)
117	0.319579	(vegetables, yogurt)
118	0.173837	(yogurt, waffles)
119	0.152766	(vegetables, yogurt, aluminum foil)
128	0.157155	(eggs, vegetables, yogurt)
130	0.157155	(vegetables, waffles, lunch meat)
131	0.152766	(poultry, vegetables, yogurt)

Рисунок 8 – Список часто встречающихся наборов длиной больше 1 и содержащих 'yogurt' или 'waffles' с минимальным уровнем поддержки 0.15 для нового датасета

Как можно увидеть из рисунка 8, все наборы действительно имеют размер больше 1 и содержат 'yogurt' или 'waffles'.

Построим датасет только из тех элементов, которые не попадают в наборы размером 1 при уровне поддержки 0.38 и приведем его к формату матрицы, пригодной для дальнейшего анализа. Таких элементов оказалось 23. Проведем ассоциативный анализ при уровне поддержки 0.3 для нового датасета. Получилось 22 набора. Результат представлен на рисунке 9.

	support	itemsets
0	0.374890	(all-purpose)
1	0.374890	(beef)
2	0.367867	(butter)
3	0.379280	(coffee/tea)
4	0.352941	(flour)
5	0.370500	(fruits)
6	0.345917	(hand soap)
7	0.375768	(individual meals)
8	0.376646	(juice)
9	0.371378	(ketchup)
10	0.378402	(laundry detergent)
11	0.375768	(mixes)
12	0.362599	(paper towels)
13	0.371378	(pasta)
14	0.355575	(pork)
15	0.367867	(sandwich bags)
16	0.349429	(sandwich loaves)
17	0.368745	(shampoo)
18	0.379280	(soap)
19	0.373134	(spaghetti sauce)
20	0.360843	(sugar)
21	0.378402	(toilet paper)
22	0.369622	(tortillas)

Рисунок 9 – Список часто встречающихся наборов с минимальным уровнем поддержки 0.3 для нового датасета

Вернемся к исходному датасету.

Напишите правило, для вывода всех наборов, в которых хотя бы два элемента начинаются на 's' и применим его к исходному датасету при минимальном уровне поддержки 0.15. Результат представлен на рисунке 10.

	support	itemsets
492	0.158911	(soap, sandwich bags)
493	0.162423	(sandwich bags, soda)
498	0.150132	(sandwich loaves, shampoo)
499	0.158033	(sandwich loaves, soap)
500	0.150132	(sandwich loaves, spaghetti sauce)
503	0.151010	(soap, shampoo)
504	0.150132	(shampoo, soda)
509	0.174715	(soap, soda)
510	0.160667	(soap, spaghetti sauce)
511	0.154522	(sugar, soap)
516	0.167691	(spaghetti sauce, soda)
517	0.162423	(sugar, soda)

Рисунок 10 – Список наборов, в которых хотя бы два элемента начинаются на 's', с уровнем поддержки больше 0.15

Как можно увидеть из рисунка 10, каждый набор действительно имеет хотя бы два элемента, начинающихся на 's'.

Напишем правило, для вывода всех наборов, для которых уровень поддержки изменяется от 0.1 до 0.25. В результате был получен 1331 набор. Список полученных наборов представлен на рисунке 11.

	support	itemsets
38	0.157155	(all-purpose, aluminum foil)
39	0.150132	(bagels, all-purpose)
40	0.144864	(beef, all-purpose)
41	0.147498	(butter, all-purpose)
42	0.151010	(all-purpose, cereals)
...	...	...
1401	0.135206	(waffles, vegetables, toilet paper)
1402	0.130817	(yogurt, vegetables, toilet paper)
1403	0.121159	(waffles, vegetables, tortillas)
1404	0.130817	(yogurt, vegetables, tortillas)
1405	0.146620	(waffles, yogurt, vegetables)

Рисунок 11 – Список наборов, для которых уровень поддержки изменяется от 0.1 до 0.25

Из рисунка 11 можно увидеть, что уровни поддержки наборов действительно лежат в промежутке от 0.1 до 0.25.

## Выводы

В ходе выполнения лабораторной работы было проведено ознакомление с методами частотного анализа из библиотеки MLxtend. Был проведен ассоциативный анализ с использованием алгоритма Apriori набора данных, содержащего списки покупок, для различных уровней минимальной

поддержки. Также были освоены методы фильтрации полученных наборов как по длине, так и по содержащимся в них элементах. Была проанализирована зависимость количества часто встречающихся наборов от уровня минимальной поддержки и построен ее график.

Код программы, написанной для выполнения лабораторной работы представлен в приложении А.

## ПРИЛОЖЕНИЕ А

### КОД ПРОГРАММЫ

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import mlxtend.preprocessing
import mlxtend.frequent_patterns

all_data = pd.read_csv('dataset_group.csv', header=None)
print(all_data)
unique_id = list(set(all_data[1]))
print(len(unique_id))

items = list(set(all_data[2]))
print(len(items))

dataset = [[elem for elem in all_data[all_data[1] == id][2] if elem in items]
for id in unique_id]
te = mlxtend.preprocessing.TransactionEncoder()
te_ary = te.fit_transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
print(df)

results      = mlxtend.frequent_patterns.apriori(df,           min_support=0.3,
use_colnames=True)
results['length'] = results['itemsets'].apply(lambda x: len(x))
print(results)

results      = mlxtend.frequent_patterns.apriori(df,           min_support=0.3,
use_colnames=True, max_len=1)
print(results)

results      = mlxtend.frequent_patterns.apriori(df,           min_support=0.3,
use_colnames=True)
results['length'] = results['itemsets'].apply(lambda x: len(x))
results = results[results['length'] == 2]
print(results)
print('\nCount of result itemsets = ', len(results))

x=[xx/100 for xx in range(5,100)]
y=[]
lenx=[]
leny=[]
lastmaxlen=0
x.reverse()
for xx in x:
    results      = mlxtend.frequent_patterns.apriori(df,   min_support=xx,
use_colnames=True)
    if len(results)!=0:
        ml=len(results['itemsets'].to_list()[-1])
        if ml!=lastmaxlen:
            lenx.append(xx)
            leny.append(len(results))
            lastmaxlen=ml
    y.append(len(results))
plt.plot(x[70:],y[70:])
plt.scatter(lenx[2:],leny[2:],c='red')
plt.show()
plt.plot(x[:70],y[:70])
plt.scatter(lenx[:2],leny[:2],c='red')
plt.show()
```

```

for n in range(len(lenx)):
    print(lenx[n])

results      =      mlxtend.frequent_patterns.apriori(df,      min_support=0.38,
use_colnames=True, max_len=1)
new_items = [ list(elem)[0] for elem in results['itemsets']]
new_dataset = [[elem for elem in all_data[all_data[1] == id][2] if elem in
new_items] for id in unique_id]
te_ary = te.fit_transform(new_dataset)
df2 = pd.DataFrame(te_ary, columns=te.columns_)

results      =      mlxtend.frequent_patterns.apriori(df2,      min_support=0.3,
use_colnames=True)
print(results)

results      =      mlxtend.frequent_patterns.apriori(df2,      min_support=0.15,
use_colnames=True)
results=results[results['itemsets'].apply(lambda x: ('yogurt' in x) or
('waffles' in x)and (len(x)>1))]
print(results)

results      =      mlxtend.frequent_patterns.apriori(df,      min_support=0.38,
use_colnames=True, max_len=1)
new_items = [ list(elem)[0] for elem in results['itemsets']]
new_dataset = [[elem for elem in all_data[all_data[1] == id][2] if not(elem in
new_items)] for id in unique_id]
te_ary = te.fit_transform(new_dataset)
df2 = pd.DataFrame(te_ary, columns=te.columns_)

results      =      mlxtend.frequent_patterns.apriori(df2,      min_support=0.3,
use_colnames=True)
print(results)

results      =      mlxtend.frequent_patterns.apriori(df,      min_support=0.15,
use_colnames=True)
results=results[results['itemsets'].apply(lambda x: [str[0]for str in
x].count('s')>=2)]
print(results)

results      =      mlxtend.frequent_patterns.apriori(df,      min_support=0.1,
use_colnames=True)
results=results[results['support'].apply(lambda x: x<=0.25)]
print(results)

```