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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Машинное обучение»

Тема: Частотный анализ

Студент гр. 8304 _____ Кирьянов Д.И.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Ознакомиться с методами частотного анализа из библиотеки MLxtend.

Ход работы.

1. Загрузка данных

1.1. Создан Python скрипт. Загружены данные в датафрейм.

1.2. Получены списки всех id покупателей и всех товаров, которые есть в файле.

```
1139
38
```

2. Подготовка данных

Закодированы данные при помощи TransactionEncoder. Выведен полученный dataframe.

	all- purpose	aluminum foil	bagels	...	vegetables	waffles	yogurt
0	True	True	False	...	True	False	True
1	False	True	False	...	True	True	True
2	False	False	True	...	True	False	False
3	True	False	False	...	False	False	False
4	True	False	False	...	True	True	True
...	...	...	...	...	...	...	...
1134	True	False	False	...	False	False	False
1135	False	False	False	...	True	False	False
1136	False	False	True	...	True	False	True
1137	True	False	False	...	True	True	True
1138	False	False	False	...	True	False	False

[1139 rows x 38 columns]

Данные стали представляться таким образом, что для каждого id покупателя формируется булевый список товаров.

3. Ассоциативный анализ с использованием алгоритма Apriori

3.1.Применён алгоритм apriori с уровнем поддержки 0.3.

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

3.2. Применен алгоритм apriori с тем же уровнем поддержки, но ограничим максимальный размер набора единицей.

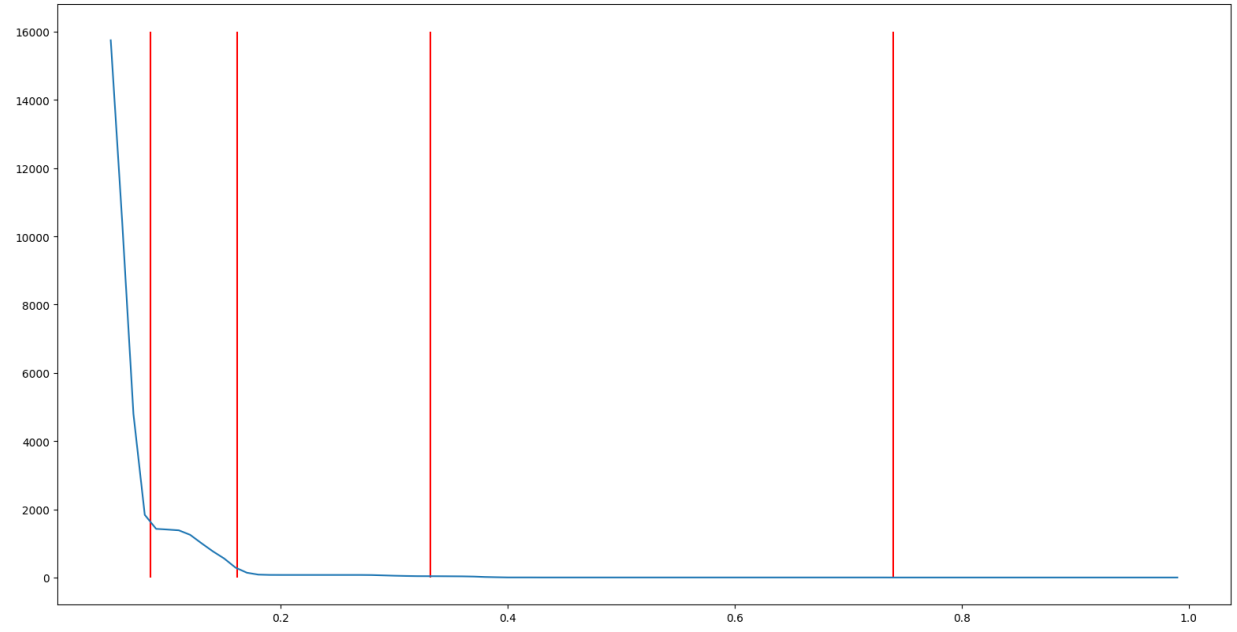
	support	itemsets
0	0.374890	(all- purpose)
1	0.384548	(aluminum foil)
2	0.385426	(bagels)
3	0.374890	(beef)
4	0.367867	(butter)
5	0.395961	(cereals)
6	0.390694	(cheeses)
7	0.379280	(coffee/tea)
8	0.388938	(dinner rolls)
9	0.388060	(dishwashing liquid/detergent)
10	0.389816	(eggs)
11	0.352941	(flour)
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)

3.3. Применен алгоритм apriori и выведены только те наборы, которые имеют размер 2, а также количество таких наборов.

	support	itemsets	length
38	0.310799	(aluminum foil, vegetables)	2
39	0.300263	(vegetables, bagels)	2
40	0.310799	(cereals, vegetables)	2
41	0.309043	(cheeses, vegetables)	2
42	0.308165	(vegetables, dinner rolls)	2
43	0.306409	(vegetables, dishwashing liquid/detergent)	2
44	0.326602	(eggs, vegetables)	2
45	0.302897	(ice cream, vegetables)	2
46	0.309043	(vegetables, laundry detergent)	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

Count of result itemstes = 14

3.4. Построен график зависимости количества наборов от уровня поддержки. Определены значения уровня поддержки при котором перестают генерироваться наборы размера 1,2,3, и т.д. Полученные уровни отмечены красным цветом.



Для набора размером 1 значение = 0.7392449517120281

Для набора размером 2 значение = 0.33187006145741876

Для набора размером 3 значение = 0.16154521510096576

Для набора размером 4 значение = 0.08516242317822652

3.5. Построен датасет, в котором оставлены только те элементы, которые попадают в наборы размеров 1 при уровне поддержки 0.38. Полученный датасет приведен к формату, который можно обработать.

	aluminum foil	bagels	cereals	...	vegetables	waffles	yogurt
0	True	False	False	...	True	False	True
1	True	False	True	...	True	True	True
2	False	True	True	...	True	False	False
3	False	False	True	...	False	False	False
4	False	False	False	...	True	True	True
...	...	...	...	...	...	...	...
1134	False	False	True	...	False	False	False
1135	False	False	True	...	True	False	False
1136	False	True	False	...	True	False	True
1137	False	False	False	...	True	True	True
1138	False	False	False	...	True	False	False

[1139 rows x 15 columns]

3.6. Проведен ассоциативный анализ при уровне поддержки 0.3 для нового датасета.

	support	itemsets	length
0	0.384548	(aluminum foil)	1
1	0.385426	(bagels)	1
2	0.395961	(cereals)	1
3	0.390694	(cheeses)	1
4	0.388938	(dinner rolls)	1
5	0.388060	(dishwashing liquid/detergent)	1
6	0.389816	(eggs)	1
7	0.398595	(ice cream)	1
8	0.395083	(lunch meat)	1
9	0.380158	(milk)	1
10	0.421422	(poultry)	1
11	0.390694	(soda)	1
12	0.739245	(vegetables)	1
13	0.394205	(waffles)	1
14	0.384548	(yogurt)	1
15	0.310799	(vegetables, aluminum foil)	2
16	0.300263	(vegetables, bagels)	2
17	0.310799	(cereals, vegetables)	2
18	0.309043	(cheeses, vegetables)	2
19	0.308165	(dinner rolls, vegetables)	2
20	0.306409	(dishwashing liquid/detergent, vegetables)	2
21	0.326602	(eggs, vegetables)	2
22	0.302897	(ice cream, vegetables)	2
23	0.311677	(lunch meat, vegetables)	2
24	0.331870	(vegetables, poultry)	2
25	0.305531	(soda, vegetables)	2
26	0.315189	(waffles, vegetables)	2
27	0.319579	(yogurt, vegetables)	2

У наборов длины 1 минимальный уровень поддержки стал 0.38.

3.7. Проведен ассоциативный анализ при уровне поддержки 0.15 для нового датасета. Выведены все наборы, размер которых больше 1 и в котором есть 'yogurt' или 'waffles'.

	support	itemsets	length
27	0.169447	(waffles, aluminum foil)	2
28	0.177349	(yogurt, aluminum foil)	2
40	0.159789	(waffles, bagels)	2
41	0.162423	(yogurt, bagels)	2
52	0.160667	(waffles, cereals)	2
53	0.172081	(cereals, yogurt)	2
63	0.172959	(waffles, cheeses)	2
64	0.172081	(yogurt, cheeses)	2
73	0.169447	(waffles, dinner rolls)	2
74	0.166813	(dinner rolls, yogurt)	2
82	0.175593	(waffles, dishwashing liquid/detergent)	2
83	0.158033	(dishwashing liquid/detergent, yogurt)	2
90	0.169447	(waffles, eggs)	2
91	0.174715	(eggs, yogurt)	2
97	0.172959	(waffles, ice cream)	2
98	0.156277	(yogurt, ice cream)	2
103	0.184372	(lunch meat, waffles)	2
104	0.161545	(lunch meat, yogurt)	2
108	0.167691	(yogurt, milk)	2
111	0.166813	(poultry, waffles)	2
112	0.180860	(poultry, yogurt)	2
114	0.177349	(waffles, soda)	2
115	0.167691	(yogurt, soda)	2
116	0.315189	(waffles, vegetables)	2
117	0.319579	(vegetables, yogurt)	2
118	0.173837	(waffles, yogurt)	2
119	0.152766	(vegetables, yogurt, aluminum foil)	3
128	0.157155	(eggs, vegetables, yogurt)	3
130	0.157155	(lunch meat, vegetables, waffles)	3
131	0.152766	(poultry, vegetables, yogurt)	3

3.8. Составлен ещё один датасет из элементов, не попавших в датасет п. 3.3 задания.

	all- purpose	beef	butter	...	sugar	toilet paper	tortillas
0	True	True	True	...	False	False	False
1	False	False	False	...	False	True	True
2	False	False	False	...	False	True	False
3	True	False	False	...	False	True	False
4	True	False	False	...	False	True	True
...	...	...	...	...	...	...	...
1134	True	True	False	...	True	False	False
1135	False	False	False	...	False	False	False
1136	False	True	False	...	True	False	True
1137	True	True	False	...	True	True	False
1138	False	False	False	...	False	False	False

[1139 rows x 23 columns]

3.9. Проведен анализ aprioti для полученного датасета.

	support	itemsets	length
0	0.374890	(all- purpose)	1
1	0.374890	(beef)	1
2	0.367867	(butter)	1
3	0.379280	(coffee/tea)	1
4	0.352941	(flour)	1
..	...	...	...
280	0.080773	(ketchup, coffee/tea, soap)	3
281	0.081651	(juice, ketchup, spaghetti sauce)	3
282	0.081651	(juice, laundry detergent, spaghetti sauce)	3
283	0.081651	(ketchup, sandwich bags, soap)	3
284	0.081651	(tortillas, sandwich bags, mixes)	3

3.10. Написано правило для вывода всех наборов, в которых хотя бы два элемента начинаются на 's'.

```
['sandwich loaves', 'sandwich bags']
['sandwich bags', 'shampoo']
['sandwich bags', 'soap']
['sandwich bags', 'spaghetti sauce']
['sandwich bags', 'sugar']
['sandwich loaves', 'shampoo']
['sandwich loaves', 'soap']
['sandwich loaves', 'spaghetti sauce']
['sandwich loaves', 'sugar']
['shampoo', 'soap']
['shampoo', 'spaghetti sauce']
['sugar', 'shampoo']
['spaghetti sauce', 'soap']
['sugar', 'soap']
['sugar', 'spaghetti sauce']
['ketchup', 'sandwich bags', 'soap']
```

3.11. Выбраны все наборы, для которых уровень поддержки лежит в промежутке [0.1, 0.25].

	support	itemsets	length
23	0.144864	(all- purpose, beef)	2
24	0.147498	(all- purpose, butter)	2
25	0.146620	(all- purpose, coffee/tea)	2
26	0.142230	(all- purpose, flour)	2
27	0.150132	(all- purpose, fruits)	2
..	...	...	...
271	0.151888	(toilet paper, spaghetti sauce)	2
272	0.148376	(tortillas, spaghetti sauce)	2
273	0.151888	(toilet paper, sugar)	2
274	0.147498	(tortillas, sugar)	2
275	0.156277	(tortillas, toilet paper)	2

4. Выводы

Ознакомились с методами частотного анализа из библиотеки MLxtend. При большем уровне поддержки уменьшается количество наборов. При этом сначала перестают генерироваться наборы большего размера.

ПРИЛОЖЕНИЕ А

Исходный код программы

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

all_data = pd.read_csv('dataset_group.csv', header=None)
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 = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
print(df)

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

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

results = 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 itemstes = ',len(results), '\n')
res = []
for minSup in range(5, 100):
    results = apriori(df, min_support=minSup/100, use_colnames=True)
    res.append(len(results))
plt.plot([i/100 for i in range(5, 100)], res)
results = apriori(df, min_support=0.05, use_colnames=True)
results['length'] = results['itemsets'].apply(lambda x: len(x))
for i in range(1, 5):
    res = results[results['length'] == i]
    print(max(res['support']))
    plt.vlines(max(res['support']), 0, 16000, color='r')
plt.show()

results = 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 = TransactionEncoder()
te_ary = te.fit(new_dataset).transform(new_dataset)
new_df = pd.DataFrame(te_ary, columns=te.columns_)
print(new_df)

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

results = apriori(new_df, min_support=0.15, use_colnames=True)
```

```
results['length'] = results['itemsets'].apply(lambda x: len(x))
results = results[results['length'] > 1]
print(results[results['itemsets'].apply(lambda x: ('yogurt' in x) or
('waffles' in x))])

results = 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 not
in new_items_] for id in unique_id]
te = TransactionEncoder()
te_ary = te.fit(new_dataset_).transform(new_dataset_)
new_df_ = pd.DataFrame(te_ary, columns=te.columns_)
print(new_df_)


results = apriori(new_df_, min_support=0.08, use_colnames=True)
results['length'] = results['itemsets'].apply(lambda x: len(x))
print(results)

y = results['itemsets'].apply(lambda x: list(x))
for i in y:
    count = 0
    for j in i:
        if j[0] == 's':
            count += 1

    if count > 1:
        print(i)


results = results[results['support'] <= 0.25]
print(results[results['support'] >= 0.1])
```