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. 6304 _____ Ястребков А. С.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2020

Цель работы:

Ознакомиться с методами частотного анализа из библиотеки Mlxtend.

Ход работы

Загрузка данных.

Загружен требуемый датасет, на рис. 1 приведён фрагмент исходных данных.

	0	1	2
0	2000-01-01	1	yogurt
1	2000-01-01	1	pork
2	2000-01-01	1	sandwich bags
3	2000-01-01	1	lunch meat
4	2000-01-01	1	all- purpose
5	2000-01-01	1	flour
6	2000-01-01	1	soda

Рис. 1. Фрагмент исходных данных.

Из датасета были извлечены массивы уникальных ID покупателей (всего 1139) и список уникальных товаров (38 позиций). Данные были сведены в датасет и преобразованы с помощью TransactionEncoder из библиотеки Mlxtend. Полученный датасет представляет из себя матрицу булевых значений размера 1139x38, в которой значение True, если покупатель приобретал данный товар, и False, если не приобретал. Код для первичной обработки данных приведён в листинге 1. Фрагмент датасета представлен на рис. 2.

Листинг 1. Загрузка и подготовка данных для ассоциативного анализа.

```
all_data = pd.read_csv('dataset_group.csv', header=None)
unique_id = list(set(all_data[1]))
items = list(set(all_data[2]))
dataset = [[elem for elem in all_data[all_data[1] == id]
[2] if elem in items] for id in unique_id]
te = TransactionEncoder()
te_array = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_array, columns=te.columns_)
```

	all-purpose	aluminum foil	bagels	beef	butter	cereals	cheeses	coffee/tea	dinner rolls	dishwashing liquid/detergent	...	shampoo	soap	soda	spaghetti sauce	sugar	toilet paper	tort
0	True	True	False	True	True	False	False	False	True	False	...	True	True	True	False	False	False	F
1	False	True	False	False	False	True	True	False	False	True	...	True	False	False	False	False	True	
2	False	False	True	False	False	True	True	False	True	False	...	True	True	True	True	False	True	F
3	True	False	False	False	False	True	False	False	False	False	...	False	False	True	False	False	True	F
4	True	False	False	False	False	False	False	False	True	False	...	False	False	True	True	False	True	

Рис. 2. Фрагмент датасета после преобразования TransactionEncoder.

Ассоциативный анализ с использованием алгоритма Apriori.

1. Датасет был обработан с помощью алгоритма Apriori из библиотеки Mlxtend. Данный алгоритм ищет часто встречающиеся наборы элементов в заданном датасете. Он изначально разрабатывался с прицелом на базы данных, подобные имеющемуся датасету, то есть, содержащие информацию о транзакциях за определённый промежуток времени. В первом эксперименте был установлен уровень поддержки 0.3, то есть, алгоритм нашёл все наборы данных, которые встречаются не менее чем в 30% выборки (такие наборы товаров, которые покупали не менее 30% покупателей). При выбранном уровне поддержки получено **52 набора** длиной 1 или 2 элемента. Первые и последние 5 наборов представлены в таблице 1.

Таблица 1. Часть полученных наборов при уровне поддержки 0.3.

	уровень поддержки	элементы	длина набора
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
...	...	...	...
47	0.311677	(lunch meat, vegetables)	2
48	0.331870	(poultry, vegetables)	2
49	0.305531	(soda, vegetables)	2
50	0.315189	(vegetables, waffles)	2
51	0.319579	(yogurt, vegetables)	2

В таблице 2 приведены только наборы, содержащие 2 элемента. Всего таких наборов 14.

Таблица 2. Наборы длиной 2 при уровне поддержки 0.3.

	уровень поддержки	элементы	длина набора
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	(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	(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	(vegetables, waffles)	2
51	0.319579	(yogurt, vegetables)	2

2. На рис. 3 показан график зависимости количества генерируемых наборов от уровня поддержки. Точками отмечены те уровни поддержки, при которых перестают генерироваться наборы длиной 4 (зелёная, уровень поддержки — 0.09), 3 (жёлтая, уровень поддержки — 0.17), 2 (пурпурная, уровень поддержки — 0.34), 1 (красная, уровень поддержки — 0.74).

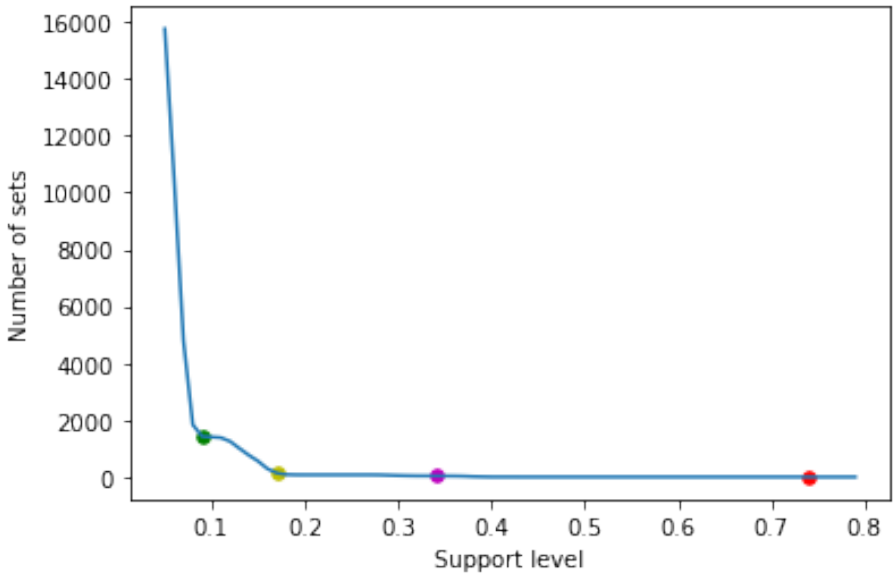

Рис. 3. График зависимости количества наборов от уровня поддержки. Точками отмечены уровни поддержки, при которых не генерируются наборы длиной 4 (зелёная), 3 (жёлтая), 2 (пурпурная), 1 (красная).

3. Был построен датасет только из тех элементов, которые попадают в наборы длиной 1 при уровне поддержки 0.38. Для нового датасета был проведён анализ с уровнем поддержки 0.3. В результате, число сгенерированных наборов уменьшилось с 52 до 28 (15 наборов с одним элементом против 38; 13 наборов с 2 элементами против 14). Результат содержит те же наборы, которые были получены после анализа исходного датасета, за исключением отфильтрованных. Уровень поддержки также не изменился, что показано в таблице 3.

Таблица 3. Уровень поддержки для некоторых наборов в исходном и отфильтрованном датасете.

длина	элементы	уровень поддержки в исходном датасете	уровень поддержки в отфильтрованном датасете
1	(aluminum foil)	0.384548	0.384548
1	(dishwashing liquid/detergent)	0.388060	0.388060
1	(vegetables)	0.739245	0.739245
2	(aluminum foil, vegetables)	0.310799	0.310799

2	(lunch meat, vegetables)	0.311677	0.311677
2	(yogurt, vegetables)	0.319579	0.319579

4. Для нового датасета с помощью кода в листинге 2 был проведён ассоциативный анализ при уровне поддержки 0.15, результат был отфильтрован так, чтобы остались наборы длиной более 1 , содержащие позиции yogurt или waffles. Результат сведён в таблицу 4.

Листинг 2. Фильтрация наборов длиной более 2, содержащих элементы yogurt или waffles.

```
low_support = apriori(new_df, min_support=0.15,
use_colnames=True)
low_support['length'] =
low_support['itemsets'].apply(lambda x: len(x))

only_yogurt_waffles = lambda x: x['length'] > 1 and
('yogurt' in x['itemsets'] or 'waffles' in x['itemsets'])

print(low_support[low_support.apply(only_yogurt_waffles,
axis=1)])
```

Таблица 4. Наборы с длиной больше 1, содержащие yogurt или waffles.

	уровень поддержки	элементы	длина набора
27	0.169447	(aluminum foil, waffles)	2
28	0.177349	(aluminum foil, yogurt)	2
40	0.159789	(waffles, bagels)	2
41	0.162423	(yogurt, bagels)	2
52	0.160667	(cereals, waffles)	2
53	0.172081	(yogurt, cereals)	2
63	0.172959	(cheeses, waffles)	2
64	0.172081	(yogurt, cheeses)	2
73	0.169447	(dinner rolls, waffles)	2
74	0.166813	(yogurt, dinner rolls)	2
82	0.175593	(dishwashing liquid/detergent, waffles)	2
83	0.158033	(yogurt, dishwashing liquid/detergent)	2

90	0.169447	(eggs, waffles)	2
91	0.174715	(yogurt, eggs)	2
97	0.172959	(ice cream, waffles)	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	(milk, yogurt)	2
111	0.166813	(poultry, waffles)	2
112	0.180860	(poultry, yogurt)	2
114	0.177349	(soda, waffles)	2
115	0.167691	(soda, yogurt)	2
116	0.315189	(vegetables, waffles)	2
117	0.319579	(yogurt, vegetables)	2
118	0.173837	(yogurt, waffles)	2
119	0.152766	(aluminum foil, yogurt, vegetables)	3
128	0.157155	(yogurt, eggs, vegetables)	3
130	0.157155	(lunch meat, vegetables, waffles)	3
131	0.152766	(poultry, yogurt, vegetables)	3

5. Был проведён ассоциативный анализ при уровне поддержки 0.3 для тех элементов, которые не попали в датасет в п. 3. В результате, получены наборы длиной только 1 с уровнем поддержки менее 0.38, как и ожидалось. Код приведён в листинге 3, результат сведён в таблицу 5.

Листинг 3. Анализ элементов, не вошедших в датасет в п. 3.

```
other_items = [item for item in items if item not in
new_items]

other_dataset = [[elem for elem in all_data[all_data[1]
== id][2] if elem in other_items] for id in unique_id]

te_3 = TransactionEncoder()
te_array =
te_3.fit(other_dataset).transform(other_dataset)
other_df = pd.DataFrame(te_array, columns=te_3.columns_)
```

```
other_results = apriori(other_df, min_support=0.3,
use_colnames=True)

print(other_results)
```

Таблица 5. Результат ассоциативного анализа для элементов, не попавших в датасет в п. 3.

	уровень поддержки	элементы	длина набора
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
5	0.370500	(fruits)	1
6	0.345917	(hand soap)	1
7	0.375768	(individual meals)	1
8	0.376646	(juice)	1
9	0.371378	(ketchup)	1
10	0.378402	(laundry detergent)	1
11	0.375768	(mixes)	1
12	0.362599	(paper towels)	1
13	0.371378	(pasta)	1
14	0.355575	(pork)	1
15	0.367867	(sandwich bags)	1
16	0.349429	(sandwich loaves)	1
17	0.368745	(shampoo)	1
18	0.379280	(soap)	1
19	0.373134	(spaghetti sauce)	1
20	0.360843	(sugar)	1
21	0.378402	(toilet paper)	1
22	0.369622	(tortillas)	1

6. Было написано правило, которое позволяет отфильтровать наборы так, чтобы остались такие, где хотя бы два элемента начинаются на „s“. Для тестирования

был сформирован набор данных при уровне поддержки 0.15, код приведён в листинге 4, результат работы правила приведён на рис. 4.

Листинг 4. Фильтрация по элементам, начинающимся на 's'.

```
def two_s_elements(row):
    s_elements = 0
    for item in row['itemsets']:
        if str(item).startswith('s'):
            s_elements = s_elements + 1
        if s_elements == 2:
            return True
    return False

results_for_rules = apriori(df, min_support=0.15,
use_colnames=True)
results_for_rules['length'] =
results_for_rules['itemsets'].apply(lambda x: len(x))

print(results_for_rules[results_for_rules.apply(two_s_ele
ments, axis=1)])
```

	support	itemsets	length
492	0.158911	(soap, sandwich bags)	2
493	0.162423	(soda, sandwich bags)	2
498	0.150132	(shampoo, sandwich loaves)	2
499	0.158033	(soap, sandwich loaves)	2
500	0.150132	(spaghetti sauce, sandwich loaves)	2
503	0.151010	(soap, shampoo)	2
504	0.150132	(soda, shampoo)	2
509	0.174715	(soap, soda)	2
510	0.160667	(soap, spaghetti sauce)	2
511	0.154522	(sugar, soap)	2
516	0.167691	(soda, spaghetti sauce)	2
517	0.162423	(sugar, soda)	2

Рис. 4. Датафрейм после применения правила.

7. Было создано правило, которое оставляет только наборы с уровнем поддержки от 0.1 до 0.25 (листинг 5). Фрагмент полученного датафрейма представлен на рис. 5.

Листинг 5. Фильтрация наборов с уровнем поддержки между 0.1 и 0.25.

```
results_for_rules = apriori(df, min_support=0.07,
use_colnames=True)
results_for_rules['length'] =
results_for_rules['itemsets'].apply(lambda x: len(x))

support_filter = lambda df: (df['support'] > 0.1) &
(df['support'] < 0.25)

print(results_for_rules[support_filter])
```

	support	itemsets	length
38	0.157155	(aluminum foil, all- purpose)	2
39	0.150132	(all- purpose, bagels)	2
40	0.144864	(all- purpose, beef)	2
41	0.147498	(all- purpose, butter)	2
42	0.151010	(cereals, all- purpose)	2
...	...	...	...
4458	0.135206	(vegetables, waffles, toilet paper)	3
4459	0.130817	(vegetables, yogurt, toilet paper)	3
4461	0.121159	(tortillas, vegetables, waffles)	3
4462	0.130817	(tortillas, vegetables, yogurt)	3
4463	0.146620	(vegetables, yogurt, waffles)	3

Рис. 5. Результат применения правила фильтрации по уровню поддержки.

Вывод:

В результате выполнения работы был изучен алгоритм ассоциативного анализа Apriori из библиотеки Mlxtend. Было установлено, что алгоритм разработан с прицелом на обработку транзакционных баз данных, и его задача — поиск наиболее часто встречающихся наборов значений в таких данных. Основным параметром работы алгоритма является уровень поддержки, который определяет, насколько часто должен встречаться набор данных, чтобы попасть в итоговую выборку. Очевидно и подтверждено экспериментами то, что чем выше значение минимального уровня поддержки, тем меньше наборов генерирует алгоритм.