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 _____ Антонов С.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы:

Ознакомиться с методами частотного анализа из библиотеки MLxtend.

Ход работы:

Загрузка данных

1. На данном этапе был скачан и загружен датасет в датафрейм.

```
import pandas as pd
import numpy as np

all_data = pd.read_csv('dataset_group.csv',header=None)
```

	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
...	...	...	...
22338	2002-02-26	1139	soda
22339	2002-02-26	1139	laundry detergent
22340	2002-02-26	1139	vegetables
22341	2002-02-26	1139	shampoo
22342	2002-02-26	1139	vegetables

Рисунок 1 Загруженный датасет

2. Получен список всех id покупателей, которые есть в датасете, таких всего 1139.

```
unique_id = all_data[1].unique()
print(unique_id.size)
```

3. Получен список всех товаров, которые есть в датасете.

```
items = all_data[2].unique()
print(items.size)
```

4. Для частотного был сформирован отдельный датасет.

```
dataset = [[elem for elem in all_data[all_data[1] == id][2] if elem in items]
for id in unique_id]
```

Подготовка данных

1. Полученный датасет был закодирован в виде матрицы с использованием TransactionEncoder.

```
te = TransactionEncoder()
te_ary = te.fit_transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
print(df)
```

```
      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
```

Рисунок 2 Часть датасета после преобразования TransactionEncoder.

Полученный датасет представляет из себя матрицу булевых значений размера 1139x38, в которой значение True, если покупатель приобретал данный товар, и False, если не приобретал.

Ассоциативный анализ с использованием алгоритма Apriori.

1. К полученному датасету был применен алгоритм ассоциативного анализа Apriori из библиотеки MLxtend. Примененный алгоритм ищет часто встречающиеся наборы элементов в заданном наборе данных. Изначально такой алгоритм разрабатывался для работы с базами данных, содержащих информацию о транзакциях за разный промежуток времени. В первом эксперименте был установлен уровень поддержки 0,3. Это означает, что алгоритм нашел 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 длинной 1 и 2 элемента. Для удобства представим первые и последние 5 элементов.

Таблица1. Часть полученных наборов данных 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

2. Применим алгоритм apriori с тем же уровнем поддержки, но ограничим максимальный размер набора единицей.

Таблица 2. Часть полученных наборов данных при уровне поддержки 0,3 с ограниченным максимальным размером набора, равным 1.

	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
…	…	…	…
33	0.378402	(toilet paper)	1
34	0.369622	(tortillas)	1

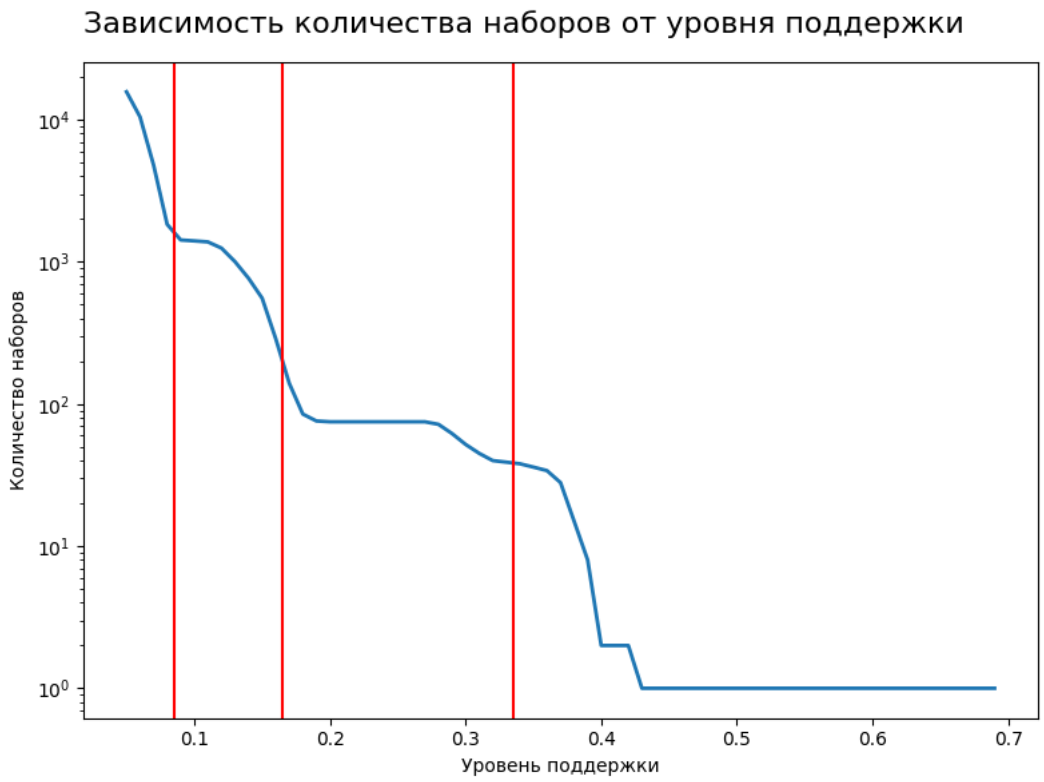
35	0.739245	(vegetables)	1
36	0.394205	(waffles)	1
37	0.384548	(yogurt)	1

3. Применим алгоритм apriori и выведем только те наборы, которые имеют размер 2, а также количество таких наборов. Всего таких наборов 14.

	support	itemsets	length
38	0.310799	(aluminum foil, vegetables)	2
39	0.300263	(bagels, vegetables)	2
40	0.310799	(vegetables, cereals)	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 3 Наборы длиной 2 при уровне поддержки 0,3

4. Приведем график зависимости количества генерируемых наборов от уровня поддержки.



5. Создан датасет только из элементов, которые попадают в наборы размером 1 при уровне поддержки 0,38. Для нового датасета был проведен анализ с уровнем поддержки 0,3. В результате, число сгенерированных наборов уменьшилось с 52 до 28 (15 наборов с 1 элементов против 38, 13 наборов с 2 элементами против 14). Результат содержит те же наборы данных, которые были получены после анализа исходного датасета, а исключением отфильтрованных. Уровень поддержки не изменился, что можно увидеть на рисунке 4.

Листинг программы для создания нового датасета:

```
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]
```

	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	(aluminum foil, vegetables)	2
16	0.300263	(bagels, vegetables)	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	(poultry, vegetables)	2
25	0.305531	(soda, vegetables)	2
26	0.315189	(waffles, vegetables)	2
27	0.319579	(yogurt, vegetables)	2

Рисунок 4 Новые полученные наборы.

6. Проведен ассоциативный анализ для нового датасета при уровне поддержки 0,15. Результат был отфильтрован так, что остались только наборы длиной более 1, содержащие yogurt или waffles.

	support	itemsets	length
27	0.169447	(waffles, aluminum foil)	2
28	0.177349	(aluminum foil, yogurt)	2
40	0.159789	(waffles, bagels)	2
41	0.162423	(yogurt, bagels)	2
52	0.160667	(waffles, cereals)	2
53	0.172081	(yogurt, cereals)	2
63	0.172959	(cheeses, waffles)	2
64	0.172081	(cheeses, yogurt)	2
73	0.169447	(waffles, dinner rolls)	2
74	0.166813	(yogurt, dinner rolls)	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	(waffles, lunch meat)	2
104	0.161545	(yogurt, lunch meat)	2
108	0.167691	(yogurt, milk)	2
111	0.166813	(waffles, poultry)	2
112	0.180860	(yogurt, poultry)	2
114	0.177349	(soda, waffles)	2
115	0.167691	(soda, yogurt)	2
116	0.315189	(waffles, vegetables)	2
117	0.319579	(yogurt, vegetables)	2
118	0.173837	(waffles, yogurt)	2
119	0.152766	(yogurt, aluminum foil, vegetables)	3
128	0.157155	(yogurt, eggs, vegetables)	3
130	0.157155	(waffles, vegetables, lunch meat)	3
131	0.152766	(yogurt, poultry, vegetables)	3

Рисунок 5 Наборы длиной большей 1 и содержащие элементы yogurt или waffles.

7. Был сформирован датасет из элементов, которые не попали в датасет в п. 5, после чего датасет был приведен к удобному для анализа виду.

```
diff = set(list(df)) - set(list(df_new))
diff_items = [ list(elem)[0] for elem in results['itemsets']]
diff_dataset = [[elem for elem in all_data[all_data[1] == id][2] if elem not
in diff_items] for id in unique_id]
te = TransactionEncoder()
te_ary = te.fit_transform(diff_dataset)
df_new = pd.DataFrame(te_ary, columns=te.columns_)
```

Приведем часть получившегося датасета:

	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)

Рисунок 6 Часть нового датасета

8. Было написано правило, для вывода всех наборов, в которых хотя бы 2 элемента начинаются 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_elements, axis=1)])
```

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

Рисунок 7 Датафрейм после применения правила элементов начинающихся хотя бы с 2 ‘s’

9. Было создано правило, которое оставляет только наборы с уровнем поддержки от 0.1 до 0.25.

```
subset_10_25 = lambda df: df[np.logical_and(df.support>=0.1, df.support <=
0.25)]
print(subset_10_25(apriori_results[0]))
```

	support	itemsets
38	0.157155	(all- purpose, aluminum foil)
39	0.150132	(all- purpose, bagels)
40	0.144864	(all- purpose, beef)
41	0.147498	(butter, all- purpose)
42	0.151010	(cereals, all- purpose)
...	...	...
9136	0.135206	(vegetables, toilet paper, waffles)
9137	0.130817	(vegetables, yogurt, toilet paper)
9139	0.121159	(tortillas, waffles, vegetables)
9140	0.130817	(tortillas, yogurt, vegetables)
9142	0.146620	(vegetables, yogurt, waffles)

Рисунок 8 Датафрейм после применения правила фильтрации по уровню поддержки

Выводы:

В результате выполнения лабораторной работы был изучен алгоритм Apriori из библиотеки MLxtend. В ходе работы было установлено, что данный алгоритм был разработан для работы с транзакциями баз данных, его задача – поиск наиболее часто встречающихся наборов значений в ких данных. Основным параметром работы алгоритма является уровень поддержки,

который определяет, насколько часто должен встречается набор данных, чтобы попасть в итоговую выборку. Также было выяснено, что чем выше значение минимального уровня поддержки, тем меньше наборов генерирует лгоритм.

.