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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Машинное обучение»

ТЕМА: Частотный анализ

Студент гр. 6302 _____ Барбарич И.Г.

Руководитель _____ Жангиров Т. Р.

Санкт-Петербург

2020

Цель работы

Ознакомиться с методами частотного анализа из библиотеки MLxtend.

Загрузка данных

1. Загрузить датасет по ссылке.

2. Создать Python скрипт. Загрузить данные в датафрейм.

```
import pandas as pd
import numpy as np

all_data = pd.read_csv('dataset_group.csv', header=None)
#В файле нет строки с названием столбцов, поэтому параметр header равен None.
#Интерес представляет информация об id покупателя - столбец с названием 1
#Название купленного товара хранится в столбце с названием 2
```

Рисунок 1. Загрузка данных.

3. Получим список всех id покупателей, которые есть в файле.

```
unique_id = list(set(all_data[1]))
print(len(unique_id)) #Выведем количество id
```

```
main
C:\Users\Lion\PycharmProjects\pythonProject3\venv\Scripts\python
1139
```

Рисунок 2. Вывод кол-во id покупателей.

4. Получим список всех товаров, которые есть в файле.

```
unique_id = list(set(all_data[1]))
print(len(unique_id)) #Выведем количество id

items = list(set(all_data[2]))
print(len(items)) #Выведем количество товаров
```

```
main
C:\Users\Lion\PycharmProjects\pythonProject3\venv\Scripts\python.exe
1139
38
```

Рисунок 3. Вывод кол-во всех товаров.

5. Далее необходимо сформировать датасет подходящий для частотного анализа. Для этого надо слить все товары одного покупателя в один список. Для дальнейшего частотного анализа id покупателя будет не нужен

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

Рисунок 4.

Подготовка данных

1. Так как полученные датасет не пригоден для анализа напрямую, так как каждый список пользователя может содержать разное количество товаров. Поэтому данные надо закодировать так, чтобы их можно было представить в виде матрицы. Для кодированния данных используем TransactionEncoder.

```
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
```

Рисунок 5.

2. Выведите полученный dataframe и объясните, как стали представляться данные.

```
      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]
```

Рисунок 6. Полученный dataframe.

Ассоциативный анализ с использованием алгоритма Apriori

1. Применим алгоритм apriori с минимальным уровнем поддержки 0.3

Объясните полученный результат

	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	(vegetables, bagels)	2
37	0.384548	(yogurt)	1
38	0.310799	(vegetables, aluminum foil)	2
39	0.300263	(vegetables, bagels)	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	(vegetables, waffles)	2
51	0.319579	(vegetables, yogurt)	2

Рисунок 7. Работа алгоритма apriori с минимальным уровнем поддержки 0.3

В результате работы алгоритма получим список всех комбинаций товаров, который встречаются у более 30% покупателей.

2. Применим 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)

Рисунок 8. Работа алгоритма apriori с минимальным уровнем поддержки 0.3 с ограничением максимального размера набора единицей.

3. Применим алгоритм apriori и выведем 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	(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	(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

Рисунок 8. Работа алгоритма.

4. Посчитайте количество наборов при различных уровнях поддержки. Начальное значение поддержки 0.05, шаг 0.01. Постройте график зависимости количества наборов от уровня поддержки.

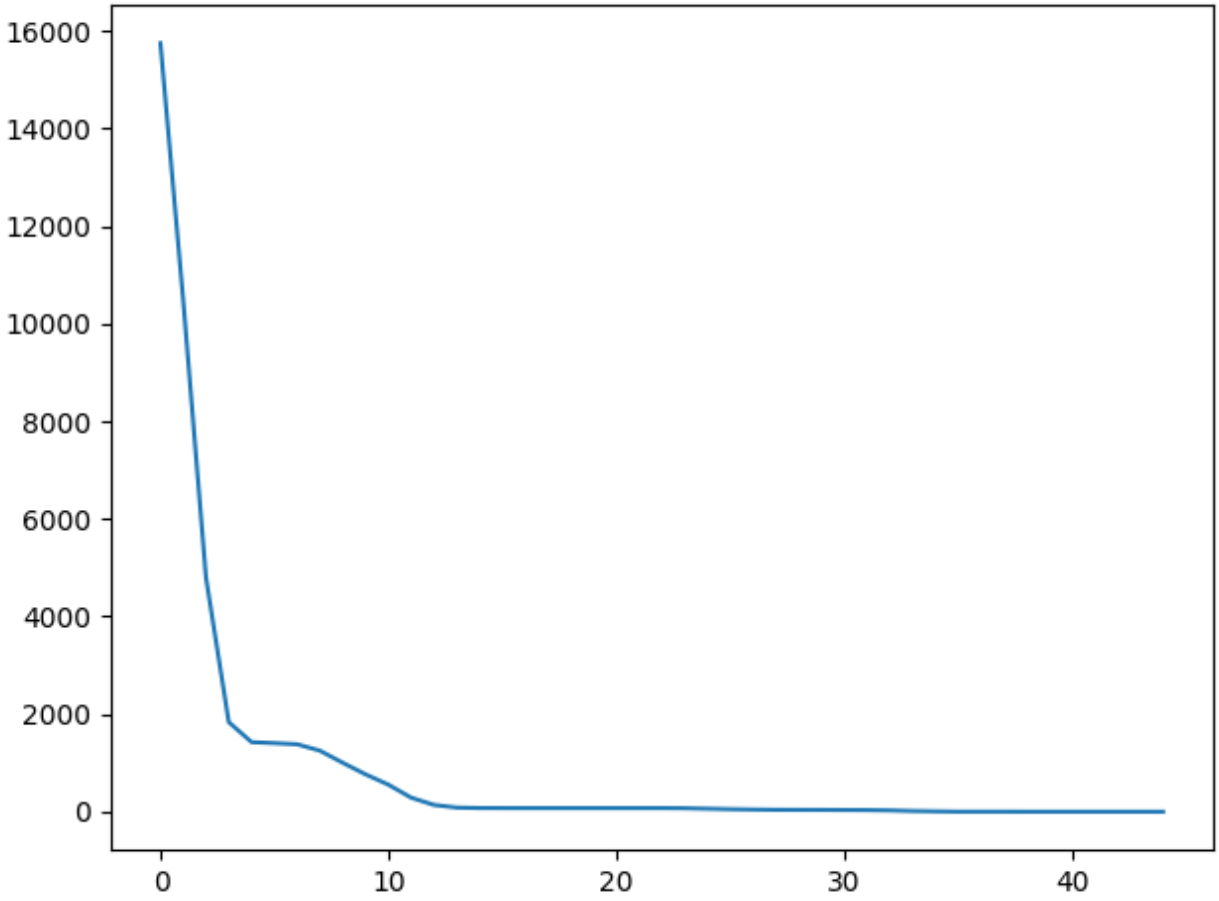


Рисунок 9. График кол-ва наборов от уровня поддержки.

5. Определите значение уровня поддержки при котором перестают генерироваться наборы размера 1,2,3, и.т.д. Отметьте полученные уровне поддержки на графике построенном в пункте 4.

```
min_support_range = np.arange(0.05, 0.5, 0.01)
lens_apriori = []

max_level = None
results_point = []
results_min_level = []
for current_support in min_support_range:
    results = apriori(df, min_support=current_support, use_colnames=True)
    lens_apriori.append(len(results))
    results['length'] = results['itemsets'].apply(lambda x: len(x))

    if max_level is None:
        max_level = results['length'].max()
    else:
        while max_level > 0 and len(results[results['length'] == max_level]) == 0:
            # print(len(results))
            results_point.append(len(results))
            results_min_level.append(current_support)
            max_level -= 1
```

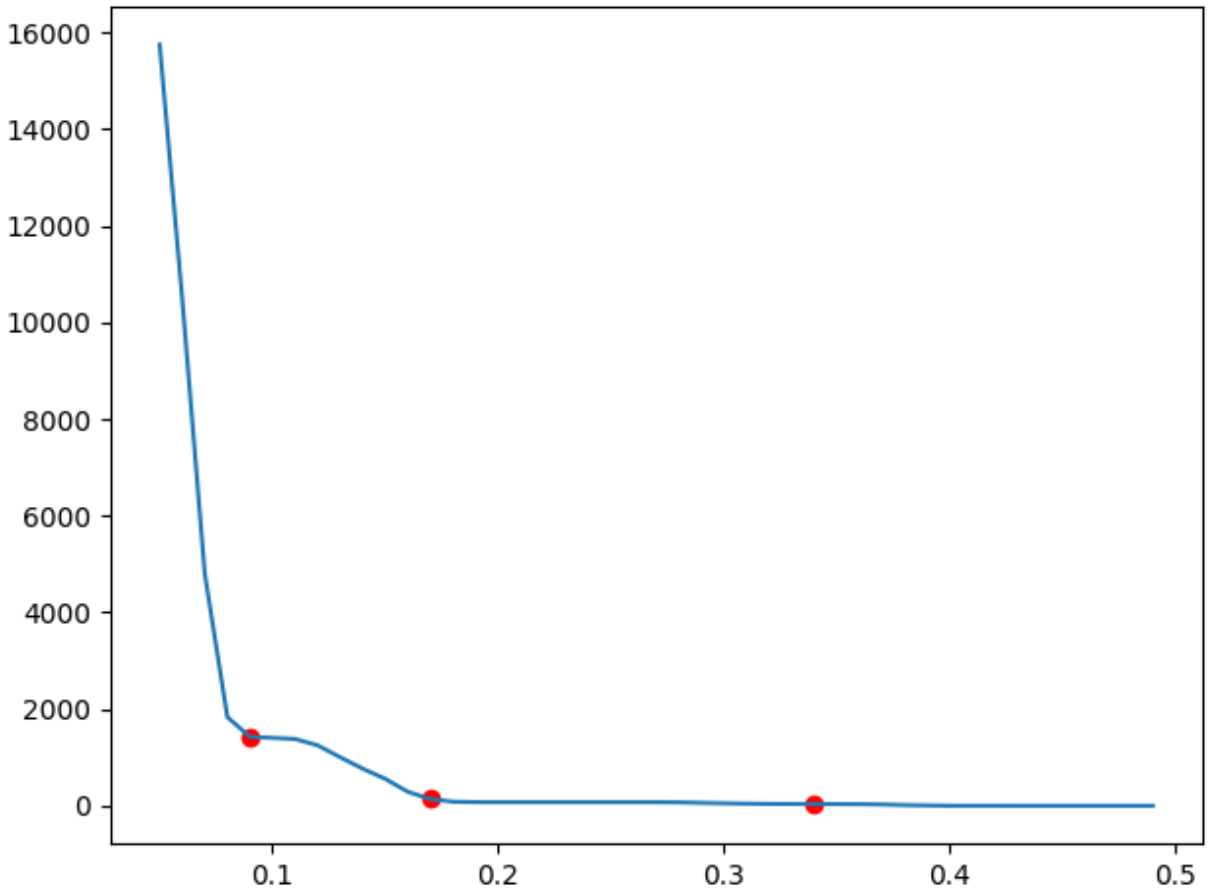


Рисунок 10.

6. Построим датасет только из тех элементов, которые попадают в наборы размером 1 при уровне поддержки 0.38.

7. Приведите полученный датасет к формату, который можно обработать.

```
te = TransactionEncoder()
te_ary = te.fit(new_dataset).transform(new_dataset)
df_new = pd.DataFrame(te_ary, columns=te.columns_)
```

8. Проведите ассоциативный анализ при уровне поддержки 0.3 для нового датасета. Опишите в чем сходства и различия.

```
# 6. Построим датасет только из тех элементов, которые попадают в наборы размером 1 при
# уровне поддержки 0.38
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)
df_new = pd.DataFrame(te_ary, columns=te.columns_)

results = apriori(df_new, min_support=0.3, use_colnames=True)
results['length'] = results['itemsets'].apply(lambda x: len(x))
# print(results)
# print('\nCount of result itemstes = ', len(results))
```

	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	(vegetables, dishwashing liquid/detergent)	2
21	0.326602	(vegetables, eggs)	2
22	0.302897	(vegetables, ice cream)	2
23	0.311677	(vegetables, lunch meat)	2
24	0.331870	(vegetables, poultry)	2
25	0.305531	(soda, vegetables)	2
26	0.315189	(vegetables, waffles)	2
27	0.319579	(vegetables, yogurt)	2

Count of result itemstes = 28

Рисунок 11.

Уменьилась выборка данных, вместо 52 – 28.

9. Проведите ассоциативный анализ при уровне поддержки 0.15 для нового датасета.Выведите все наборы размер которых больше 1 и в котором есть 'yogurt' или 'waffles'

```
# < 15%
results = apriori(df_new, min_support=0.15, use_colnames=True)
results['length'] = results['itemsets'].apply(lambda x: len(x))  # добавление размера набора
results = results[results['length'] > 1]
results = results[results['itemsets'].apply(lambda x: ('yogurt' in x) or ('waffles' in x))]
```

	support	itemsets	length
27	0.169447	(waffles, aluminum foil)	2
28	0.177349	(yogurt, aluminum foil)	2
40	0.159789	(bagels, waffles)	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	(waffles, dishwashing liquid/detergent)	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	(ice cream, yogurt)	2
103	0.184372	(lunch meat, waffles)	2
104	0.161545	(yogurt, lunch meat)	2
108	0.167691	(yogurt, milk)	2
111	0.166813	(poultry, waffles)	2
112	0.180860	(yogurt, poultry)	2
114	0.177349	(soda, waffles)	2
115	0.167691	(yogurt, soda)	2
116	0.315189	(vegetables, waffles)	2
117	0.319579	(vegetables, yogurt)	2
118	0.173837	(yogurt, waffles)	2
119	0.152766	(vegetables, yogurt, aluminum foil)	3
128	0.157155	(vegetables, yogurt, eggs)	3
130	0.157155	(vegetables, lunch meat, waffles)	3
131	0.152766	(vegetables, yogurt, poultry)	3

Count of result itemstes = 30

Рисунок 12.

10. Постройте датасет, из тех элементов, которые не попали в датасет в п. 6 и приведите его к удобному для анализа виду.

```python
# Постройте датасет, из тех элементов, которые не попали в датасет в п. 6 и приведите его к
# удобному для анализа виду
new_dataset_2 = [[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_2).transform(new_dataset)
df_new = pd.DataFrame(te_ary, columns=te.columns_)
```

11. Проведите анализ aprioti для полученного датасета

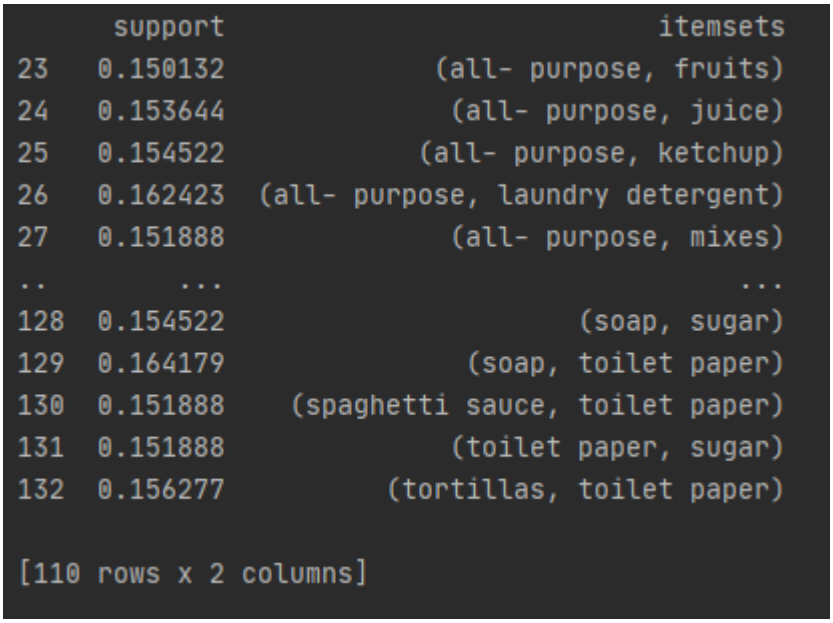
```python
# 11. Проведите анализ aprioti для полученного датасета
results = apriori(df_new, min_support=0.15, use_colnames=True)
results['length'] = results['itemsets'].apply(lambda x: len(x))  # добавление размера набора
print(results)
print('\nCount of result itemstes = ', len(results))
```

```
      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
..        ...                            ...     ...
128  0.154522                (sugar, soap)       2
129  0.164179           (toilet paper, soap)       2
130  0.151888  (toilet paper, spaghetti sauce)       2
131  0.151888           (sugar, toilet paper)       2
132  0.156277       (tortillas, toilet paper)       2

[133 rows x 3 columns]
```

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

```
      support                             itemsets
120  0.158911             (sandwich bags, soap)
122  0.150132        (sandwich loaves, shampoo)
123  0.158033           (sandwich loaves, soap)
124  0.150132  (sandwich loaves, spaghetti sauce)
125  0.151010                   (shampoo, soap)
127  0.160667           (soap, spaghetti sauce)
128  0.154522                     (sugar, soap)

Count of result itemstes =  7
```

13. Напишите правило, для вывода всех наборов, для которых уровень поддержки изменяется от 0.1 до 0.25



	support	itemsets
23	0.150132	(all- purpose, fruits)
24	0.153644	(all- purpose, juice)
25	0.154522	(all- purpose, ketchup)
26	0.162423	(all- purpose, laundry detergent)
27	0.151888	(all- purpose, mixes)
..	...	...
128	0.154522	(soap, sugar)
129	0.164179	(soap, toilet paper)
130	0.151888	(spaghetti sauce, toilet paper)
131	0.151888	(toilet paper, sugar)
132	0.156277	(tortillas, toilet paper)

[110 rows x 2 columns]

Вывод

В результате работы ознакомился с методами частотного анализа из библиотеки MLxtend**.**