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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Машинное обучение»

Студенты гр. 6304 _____ Тимофеев А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Лабораторная работа №3

Загрузка данных

In [10]:
```python
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.rcParams['figure.dpi'] = 200

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

Out[10]:

	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

22343 rows × 3 columns

In [29]:
```python
unique_id = list(set(all_data[1]))
print('Количество уникальных покупателей : {}'.format(len(unique_id)))
```

```
Количество уникальных покупателей : 1139
```

In [28]:
```python
items = list(set(all_data[2]))
print('Количество уникальных товаров: {}'.format(len(items)))
```

```
Количество уникальных товаров: 38
```

In [31]:
```python
dataset = [[elem for elem in all_data[all_data[1] == id][2] if elem in items] for id in unique_id]
```

Товары были распределены по покупателям

Подготовка данных

Был получен датасет, где строки представляют собой списки покупок отдельных клиентов. Если клиент купил соответствующий товар, в столбце стоит Ttrue, иначе - False.

In [14]:
```python
from mlxtend.preprocessing import TransactionEncoder

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

Out[14]:

	all-purpose	aluminum foil	bagels	beef	butter	cereals	cheeses	coffee/tea	dinner rolls	dishwashing liquid/detergent	...	shampoo	soap	soda	spaghe sau
0	True	True	False	True	True	False	False	False	True	False	...	True	True	True	Fa
1	False	True	False	False	False	True	True	False	True	True	...	True	False	False	Fa
2	False	False	True	False	False	True	True	False	True	False	...	True	True	True	T
3	True	False	False	False	False	True	False	False	False	True	...	False	False	True	Fa
4	True	False	False	False	False	False	False	False	True	False	...	False	False	True	Tr
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1134	True	False	False	True	False	True	True	True	True	True	...	True	True	False	Fa
1135	False	False	False	True	False	True	True	True	True	True	...	False	True	False	T
1136	False	False	True	True	False	False	False	False	False	True	...	True	True	False	Fa
1137	True	False	False	True	False	False	True	False	False	False	...	False	True	True	T
1138	False	False	False	False	False	False	False	False	False	False	...	True	False	True	Fa

1139 rows × 38 columns

Ассоциативный анализ с использованием алгоритма Apriori

Были получены наборы с уровнем поддержки выше 0.3. Данные наборы были куплены не менее 30% клиентами.

In [15]:
```python
from mlxtend.frequent_patterns import apriori

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

```
     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
40  0.310799                          (vegetables, cereals)       2
41  0.309043                          (vegetables, cheeses)       2
42  0.308165                     (vegetables, dinner rolls)       2
43  0.306409     (vegetables, dishwashing liquid/detergent)       2
44  0.326602                             (vegetables, eggs)       2
45  0.302897                        (ice cream, vegetables)       2
46  0.309043                (laundry detergent, vegetables)       2
47  0.311677                       (vegetables, lunch meat)       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
```

Были получены наборы с уровнем поддержки выше 0.3 размером 1.

In [16]:
```python
results = apriori(df, min_support=0.3, use_colnames=True, max_len=1)
print(results)
```

```
     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)
```

Были получены наборы с уровнем поддержки выше 0.3 размером 2, а также количество таких наборов.

In [17]:
```python
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))
```

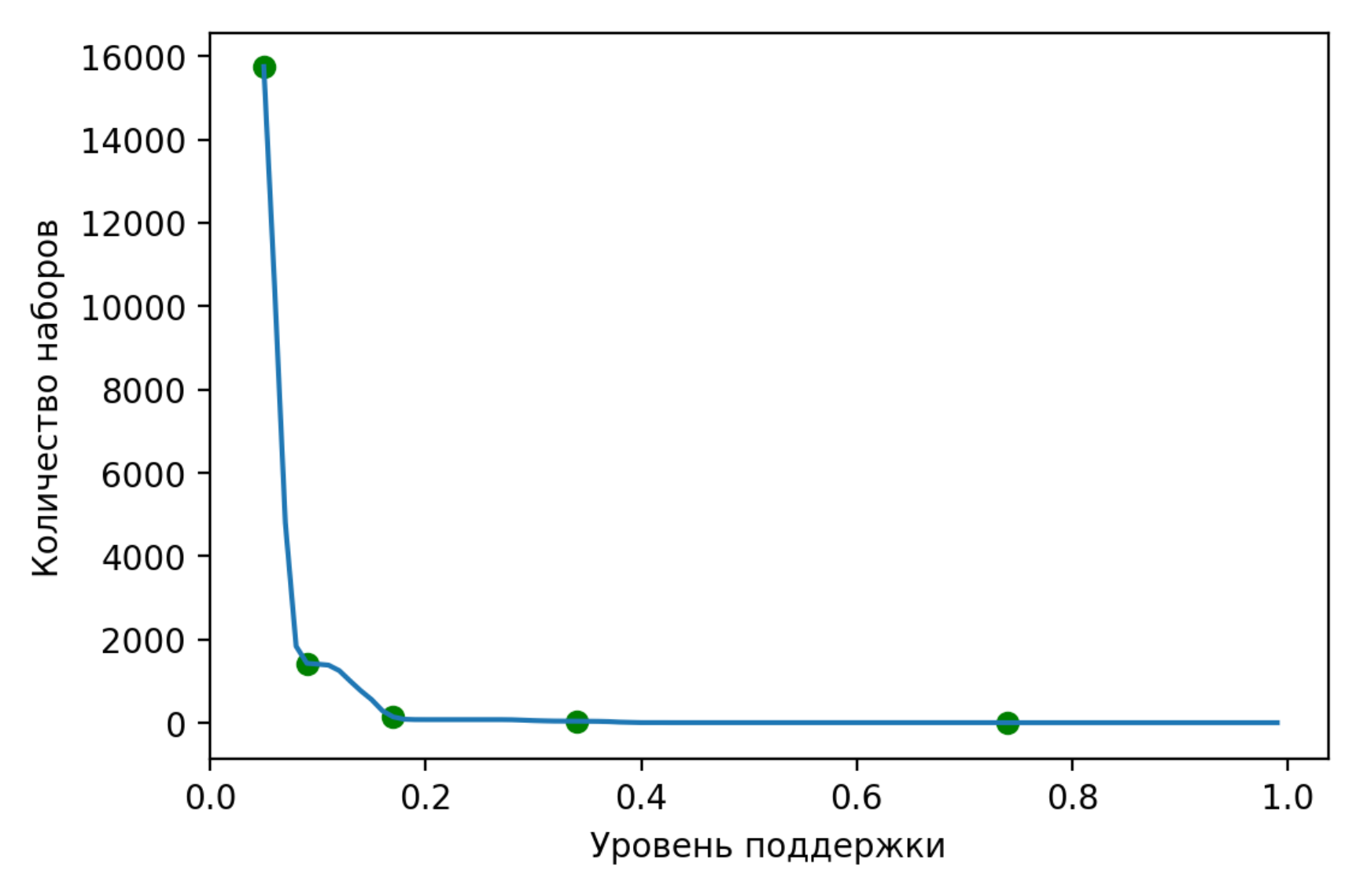
```
     support                                      itemsets  length
38  0.310799                    (vegetables, aluminum foil)       2
39  0.300263                           (vegetables, bagels)       2
40  0.310799                          (vegetables, cereals)       2
41  0.309043                          (vegetables, cheeses)       2
42  0.308165                     (vegetables, dinner rolls)       2
43  0.306409     (vegetables, dishwashing liquid/detergent)       2
44  0.326602                             (vegetables, eggs)       2
45  0.302897                        (ice cream, vegetables)       2
46  0.309043                (laundry detergent, vegetables)       2
47  0.311677                       (vegetables, lunch meat)       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

Count of result itemstes =  14
```

Построен график зависимости количества наборов от уровня поддержки, а также определены значения уровня поддержки при которых перестают генерироваться наборы размера 1,2,3, и.т.д

In [37]:
```python
supports = np.arange(0.05, 1, 0.01)
num_of_sets = []
max_nums = []
max_num = None
for sup in supports:
    sup_results = apriori(df, min_support=sup, use_colnames=True)
    sup_results['length'] = sup_results['itemsets'].apply(lambda x: len(x))
    num = len(sup_results)
    current_max_num = np.max(sup_results['length'])
    if max_num is None:
        max_num = current_max_num
        max_nums.append(sup.round(2))
        plt.scatter(sup, num, c='g')
    elif current_max_num < max_num:
        max_num = current_max_num
        max_nums.append(sup.round(2))
        plt.scatter(sup, num, c='g')
    elif np.isnan(current_max_num) and not np.isnan(max_num):
        max_num = current_max_num
        max_nums.append(sup.round(2))
        plt.scatter(sup, num, c='g')
    num_of_sets.append(num)
print('max_nums: {}'.format(max_nums))
plt.plot(supports, num_of_sets)
plt.xlabel('Уровень поддержки')
plt.ylabel('Количество наборов')
plt.show()
```

```
max_nums: [0.05, 0.09, 0.17, 0.34, 0.74]
```



Был построен датасет только из тех элементов, которые попадают в наборы размером 1 при уровне поддержки 0.38 и приведен к формату, который можно обработать.

In [19]:
```python
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_ary = te.fit(new_dataset).transform(new_dataset)
ndf = pd.DataFrame(te_ary, columns=te.columns_)
ndf
```

Out[19]:

	aluminum foil	bagels	cereals	cheeses	dinner rolls	dishwashing liquid/detergent	eggs	ice cream	lunch meat	milk	poultry	soda	vegetables	waffles	yogurt
0	True	False	False	False	True	False	False	True	True	False	False	True	True	False	True
1	True	False	True	True	False	True	False	False	False	True	False	False	True	True	True
2	False	True	True	True	True	False	True	True	True	True	True	True	True	False	False
3	False	False	True	False	False	False	False	False	True	False	False	True	False	False	False
4	False	False	False	False	True	False	True	False	False	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1134	False	False	True	True	True	True	False	True	False	False	True	False	False	False	False
1135	False	False	True	True	True	True	True	False	True	True	True	False	True	False	False
1136	False	True	False	False	True	True	True	False	True	False	True	False	True	False	True
1137	False	False	False	True	False	False	False	False	False	True	True	True	True	True	True
1138	False	False	False	False	False	False	False	False	False	False	False	True	True	False	False

1139 rows × 15 columns

Для нового датасета был проведен анализ с уровнем поддержки 0.3, в результатах отсутствуют наборы, чей уровень поддержки в исходном датасете был ниже 0.38.

In [20]:
```python
results = apriori(ndf, min_support=0.3, use_colnames=True)
print(results)
print('\nCount of result itemstes = ',len(results))
```

```
     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  0.394205                                      (waffles)
14  0.384548                                       (yogurt)
15  0.310799                    (vegetables, aluminum foil)
16  0.300263                           (vegetables, bagels)
17  0.310799                          (vegetables, cereals)
18  0.309043                          (vegetables, cheeses)
19  0.308165                     (vegetables, dinner rolls)
20  0.306409     (vegetables, dishwashing liquid/detergent)
21  0.326602                             (vegetables, eggs)
22  0.302897                        (ice cream, vegetables)
23  0.311677                       (vegetables, lunch meat)
24  0.331870                          (poultry, vegetables)
25  0.305531                             (soda, vegetables)
26  0.315189                          (vegetables, waffles)
27  0.319579                           (vegetables, yogurt)

Count of result itemstes =  28
```

Был проведен ассоциативный анализ при уровне поддержки 0.15 для нового датасета, а также выведены все наборы, размер которых больше 1 и в которых есть 'yogurt' или 'waffles'.

In [21]:
```python
results = apriori(ndf, min_support=0.15, use_colnames=True)
results['res'] = results['itemsets'].apply(lambda x: len(x) > 1 and ('yogurt' in x or 'waffles' in x))
results = results[results['res']]
print(results)
print('\nCount of result itemstes = ', len(results))
```

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

Count of result itemstes =  30
```

Был построен датасет, из тех элементов, которые не попали в датасет в п. 6, и приведен к удобному для анализа виду.

In [22]:
```python
difference = set(list(df)) - set(list(ndf))
one_more_df = [[elem for elem in all_data[all_data[1] == id][2] if elem in difference] for id in unique
_id]
te = TransactionEncoder()
te_ary = te.fit_transform(one_more_df)
omdf = pd.DataFrame(te_ary, columns=te.columns_)
omdf
```

Out[22]:

	all-purpose	beef	butter	coffee/tea	flour	fruits	hand soap	individual meals	juice	ketchup	...	pasta	pork	sandwich bags	sandwich loaves	shampoo
0	True	True	True	False	True	False	False	False	False	False	...	False	True	True	False	True
1	False	False	False	False	False	False	True	True	False	False	...	False	False	True	False	True
2	False	False	False	False	False	False	False	False	False	True	...	False	True	False	True	True
3	True	False	False	False	False	False	False	False	True	False	...	False	False	False	False	False
4	True	False	False	False	True	False	False	True	False	False	...	True	True	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1134	True	True	False	True	False	True	True	False	True	False	...	False	True	True	False	True
1135	False	False	False	True	False	False	True	True	False	False	...	True	False	False	False	False
1136	False	True	False	False	False	False	True	True	True	False	...	False	True	False	False	True
1137	True	True	False	False	False	False	False	False	False	True	...	False	False	True	False	False
1138	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	True

1139 rows × 23 columns

Был проведен анализ apriori для полученного датасета.

In [23]:
```python
results = apriori(omdf, min_support=0.3, use_colnames=True)
results
```

Out[23]:

	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)

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

In [24]:
```python
results = apriori(df, min_support=0.1, use_colnames=True)
results['res'] = results['itemsets'].apply(lambda x: len([item for item in x if item.startswith('s')])
>= 2)
results = results[results['res']]
print(results)
print('\nCount of result itemstes = ', len(results))
```

```
       support                                       itemsets   res
675   0.137840                (sandwich loaves, sandwich bags)  True
676   0.146620                        (shampoo, sandwich bags)  True
677   0.158911                           (sandwich bags, soap)  True
678   0.162423                           (soda, sandwich bags)  True
679   0.147498                (spaghetti sauce, sandwich bags)  True
680   0.131694                          (sugar, sandwich bags)  True
686   0.150132                      (shampoo, sandwich loaves)  True
687   0.158033                         (sandwich loaves, soap)  True
688   0.141352                         (soda, sandwich loaves)  True
689   0.150132              (spaghetti sauce, sandwich loaves)  True
690   0.136962                        (sugar, sandwich loaves)  True
696   0.151010                                 (shampoo, soap)  True
697   0.150132                                 (soda, shampoo)  True
698   0.139596                      (shampoo, spaghetti sauce)  True
699   0.147498                                (sugar, shampoo)  True
705   0.174715                                    (soda, soap)  True
706   0.160667                         (spaghetti sauce, soap)  True
707   0.154522                                   (sugar, soap)  True
713   0.167691                         (soda, spaghetti sauce)  True
714   0.162423                                   (soda, sugar)  True
720   0.144864                        (sugar, spaghetti sauce)  True
1351  0.115013   (vegetables, sandwich loaves, sandwich bags)  True
1352  0.122915           (vegetables, shampoo, sandwich bags)  True
1353  0.129939              (vegetables, sandwich bags, soap)  True
1354  0.129061              (soda, vegetables, sandwich bags)  True
1355  0.123793   (vegetables, spaghetti sauce, sandwich bags)  True
1356  0.113257             (vegetables, sugar, sandwich bags)  True
1361  0.129061         (vegetables, shampoo, sandwich loaves)  True
1362  0.132572            (vegetables, sandwich loaves, soap)  True
1363  0.121159            (soda, vegetables, sandwich loaves)  True
1364  0.122915 (vegetables, spaghetti sauce, sandwich loaves)  True
1365  0.121159           (vegetables, sugar, sandwich loaves)  True
1370  0.124671                   (vegetables, shampoo, soap)  True
1371  0.128183                   (soda, vegetables, shampoo)  True
1372  0.117647        (vegetables, shampoo, spaghetti sauce)  True
1373  0.122037                  (vegetables, sugar, shampoo)  True
1378  0.141352                      (soda, vegetables, soap)  True
1379  0.136962           (vegetables, spaghetti sauce, soap)  True
1380  0.127305                     (vegetables, sugar, soap)  True
1385  0.138718           (soda, vegetables, spaghetti sauce)  True
1386  0.136084                     (soda, vegetables, sugar)  True
1391  0.124671          (vegetables, sugar, spaghetti sauce)  True

Count of result itemstes =  42
```

Было написано правило, для вывода всех наборов, для которых уровень поддержки изменяется от 0.1 до 0.25.

In [27]:
```python
results = apriori(df, min_support=0.1, use_colnames=True)
results['res'] = results['support'].apply(lambda support: support > 0.1 and support < 0.25)
results = results[results['res']]
results
```

Out[27]:

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

1331 rows × 3 columns

Выводы

В ходе выполнения данной лабораторной работы было произведено знакомство с методами частотного анализа из библиотеки MLxtend.