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. 6307 _____ Золотухин М. А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2020

Загрузка и подготовка данных

Загрузим и подготовим данные к обработке.

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

plt.rcParams["figure.figsize"] = (20, 10)
plt.rcParams['figure.facecolor'] = 'white'

#%%

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

#%%

unique_id = list(set(all_data[1]))
print(f'Buyers: {len(unique_id)}') # Выведем количество id

#%%

items = list(set(all_data[2]))
print(f'Items: {len(items)}') # Выведем количество товаров

#%%

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_)
```

	all-purpose	aluminum foil	bagels	beef	butter	cereals	cheeses	coffee/tea	dinner rolls	dishwashing liquid/detergent	...	shampoo	soap	soda	spaghetti sauce	sugar	toilet paper	tortillas
0	True	True	False	True	True	False	False	False	True	False	...	True	True	True	False	False	False	False
1	False	True	False	False	False	True	True	False	False	True	...	True	False	False	False	False	True	True
2	False	False	True	False	False	True	True	False	True	False	...	True	True	True	True	False	True	False
3	True	False	False	False	False	True	False	False	False	False	...	False	False	True	False	False	True	False
4	True	False	False	False	False	False	False	False	True	False	...	False	False	True	True	False	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1134	True	False	False	True	False	True	True	True	True	True	...	True	True	False	False	True	False	False
1135	False	False	False	False	False	True	True	True	True	True	...	False	True	False	True	False	False	False
1136	False	False	True	True	False	False	False	False	True	True	...	True	True	False	False	True	False	True
1137	True	False	False	True	False	False	True	False	False	False	...	False	True	True	True	True	True	False
1138	False	False	False	False	False	False	False	False	False	False	...	True	False	True	False	False	False	False

Исходные данные были представлены в виде таблицы с колонками: дата, id покупателя и наименование покупки.

Теперь данные представлены так, что для каждого покупателя (наименование ряда) существуют колонки, каждая из которых представляет собой один из продуктов, а значения для каждого покупателя — купил он этот продукт или нет. Проще говоря получилась "тепловая карта" продуктов для покупателей.

Ассоциативный анализ с использованием алгоритма Apriori

Применим алгоритм apriori с минимальным уровнем поддержки 0.3.

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

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

	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)

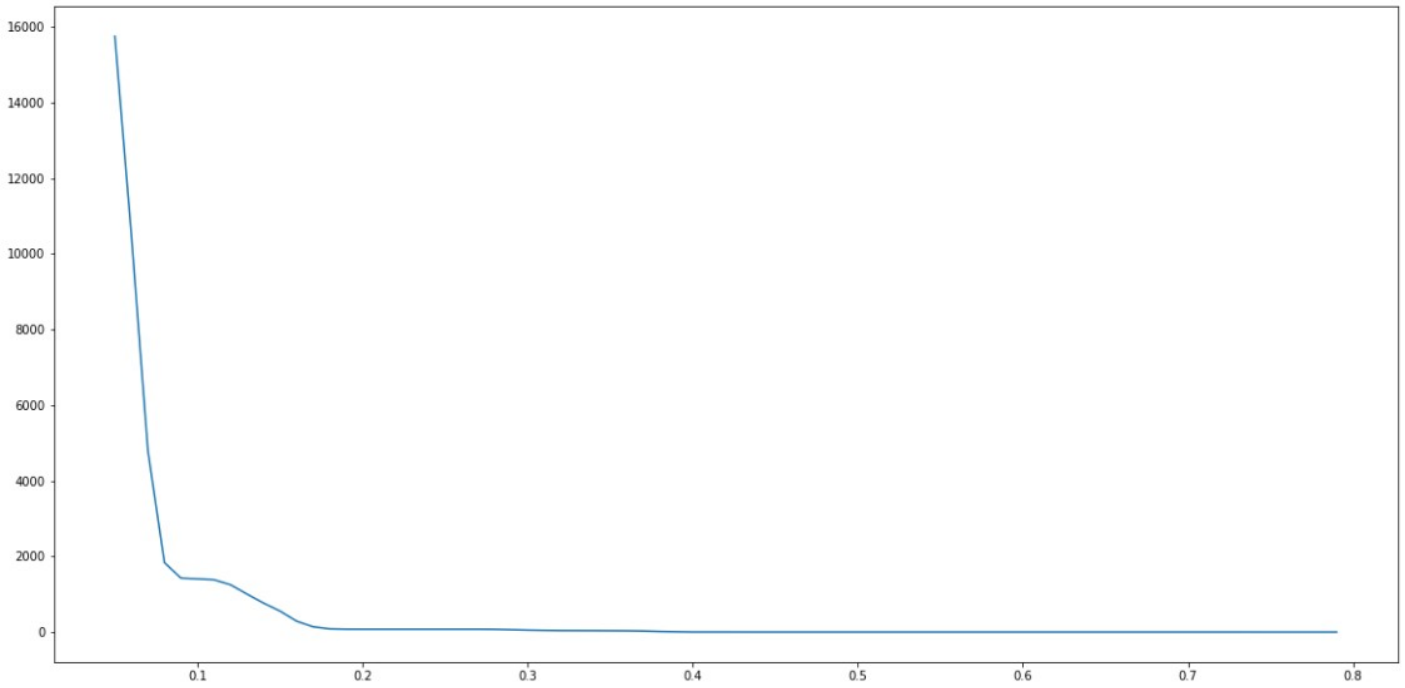
Применим алгоритм apriori и выведем только те наборы, которые имеют размер 2, а также количество таких наборов.

	support	itemsets	length
15	0.310799	(vegetables, aluminum foil)	2
16	0.300263	(bagels, vegetables)	2
17	0.310799	(vegetables, cereals)	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	(vegetables, lunch meat)	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

Посчитаем количество наборов при различных уровнях поддержки. Начальное значение поддержки 0.05, шаг 0.01. Построим график зависимости количества наборов от уровня поддержки.

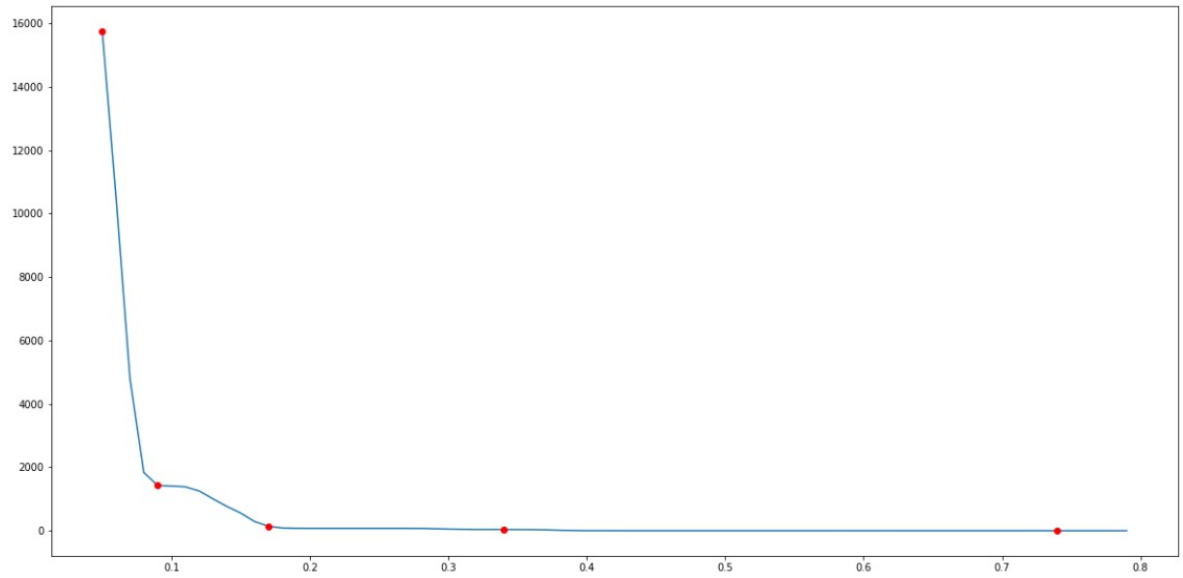
```
min_support_range = np.arange(0.05, 0.8, 0.01)

itemsets_lengths = []
for min_support in min_support_range:
    results = apriori(df, min_support=min_support, use_colnames=True)
    itemsets_lengths.append(len(results))
```

```
plt.figure()
plt.plot(min_support_range.tolist(), itemsets_lengths)
plt.show()
```



Как видно из графика количество наборов значительно уменьшается с повышением границы, что логично.

Определим значения уровня поддержки при котором перестают генерироваться наборы размера 1,2,3, и.т.д. Отметим полученные уровни поддержки на графике построенном выше.



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]
```

Приведём полученный датасет к формату, который можно обработать.

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

	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

Проведём ассоциативный анализ при уровне поддержки 0.3 для нового датасета.

	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)

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

Как видно в подборке присутствуют лишь 28 элементов, вместо 52 ранее, потому что из датасета были исключены элементы с частотой менее 0.38 и размером набора 1. Т.е. элементы, что были получены присутствуют также в старой подборке.

Проведём ассоциативный анализ при уровне поддержки 0.15 для нового датасета.

	support	itemsets
0	0.384548	(aluminum foil)
1	0.385426	(bagels)
2	0.395961	(cereals)
3	0.390694	(cheeses)
4	0.388938	(dinner rolls)
...	...	...
127	0.158033	(eggs, soda, vegetables)
128	0.157155	(eggs, yogurt, vegetables)
129	0.158033	(vegetables, lunch meat, poultry)
130	0.157155	(waffles, vegetables, lunch meat)
131	0.152766	(yogurt, vegetables, poultry)

Выведем все наборы размер которых больше 1 и в котором есть 'yogurt' или 'waffles'.

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

	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	(yogurt, cereals)	2
63	0.172959	(waffles, cheeses)	2
64	0.172081	(yogurt, cheeses)	2
73	0.169447	(dinner rolls, waffles)	2
74	0.166813	(dinner rolls, yogurt)	2
82	0.175593	(waffles, dishwashing liquid/detergent)	2
83	0.158033	(yogurt, dishwashing liquid/detergent)	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	(waffles, soda)	2
115	0.167691	(yogurt, soda)	2
116	0.315189	(waffles, vegetables)	2
117	0.319579	(yogurt, vegetables)	2
118	0.173837	(waffles, yogurt)	2
119	0.152766	(yogurt, vegetables, aluminum foil)	3
128	0.157155	(eggs, yogurt, vegetables)	3
130	0.157155	(waffles, vegetables, lunch meat)	3
131	0.152766	(yogurt, vegetables, poultry)	3

Построим датасет, из тех элементов, которые не попали в датасет в п. 6 и приведём его к удобному для анализа виду.

```
# All data dataframe
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
all_df = pd.DataFrame(te_ary, columns=te.columns_)

diff_df = all_df[all_df.columns.difference(new_items)]
```

```
diff_df
```

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

Проведём анализ apriori для полученного датасета для уровней поддержки 0.3 и 0.15.

Результат для 0.3:

	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)

Для 0.15:

	support	itemsets
0	0.374890	(all- purpose)
1	0.374890	(beef)
2	0.367867	(butter)
3	0.379280	(coffee/tea)

	support	itemsets
4	0.352941	(flour)
...	...	...
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)

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

```
# All data dataframe
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
all_df = pd.DataFrame(te_ary, columns=te.columns_)

results = apriori(all_df, min_support=0.1, use_colnames=True)
# print(results)
results = results[results['itemsets'].apply(
    lambda x: np.fromiter(
        map(lambda y: y.startswith('s'), x), dtype=bool
    ).sum() ⩾ 2
)]
results
```

	support	itemsets
675	0.137840	(sandwich bags, sandwich loaves)
676	0.146620	(sandwich bags, shampoo)
677	0.158911	(sandwich bags, soap)
678	0.162423	(sandwich bags, soda)
679	0.147498	(sandwich bags, spaghetti sauce)
680	0.131694	(sandwich bags, sugar)
686	0.150132	(sandwich loaves, shampoo)
687	0.158033	(soap, sandwich loaves)
688	0.141352	(soda, sandwich loaves)
689	0.150132	(spaghetti sauce, sandwich loaves)
690	0.136962	(sandwich loaves, sugar)
696	0.151010	(soap, shampoo)
697	0.150132	(soda, shampoo)
698	0.139596	(spaghetti sauce, shampoo)
699	0.147498	(shampoo, sugar)
705	0.174715	(soap, soda)
706	0.160667	(soap, spaghetti sauce)
707	0.154522	(soap, sugar)
713	0.167691	(spaghetti sauce, soda)
714	0.162423	(soda, sugar)
720	0.144864	(spaghetti sauce, sugar)
1351	0.115013	(sandwich bags, sandwich loaves, vegetables)
1352	0.122915	(sandwich bags, shampoo, vegetables)

	support	itemsets
1353	0.129939	(sandwich bags, soap, vegetables)
1354	0.129061	(sandwich bags, soda, vegetables)
1355	0.123793	(sandwich bags, spaghetti sauce, vegetables)
1356	0.113257	(vegetables, sandwich bags, sugar)
1361	0.129061	(sandwich loaves, shampoo, vegetables)
1362	0.132572	(soap, sandwich loaves, vegetables)
1363	0.121159	(soda, sandwich loaves, vegetables)
1364	0.122915	(spaghetti sauce, sandwich loaves, vegetables)
1365	0.121159	(vegetables, sandwich loaves, sugar)
1370	0.124671	(soap, shampoo, vegetables)
1371	0.128183	(soda, shampoo, vegetables)
1372	0.117647	(spaghetti sauce, shampoo, vegetables)
1373	0.122037	(vegetables, shampoo, sugar)
1378	0.141352	(soap, soda, vegetables)
1379	0.136962	(soap, spaghetti sauce, vegetables)
1380	0.127305	(vegetables, soap, sugar)
1385	0.138718	(spaghetti sauce, soda, vegetables)
1386	0.136084	(vegetables, soda, sugar)
1391	0.124671	(vegetables, spaghetti sauce, sugar)

Напишем правило, для вывода всех наборов, для которых уровень поддержки изменяется от 0.1 до 0.25

```
# All data dataframe
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
all_df = pd.DataFrame(te_ary, columns=te.columns_)

results = apriori(all_df, min_support=0.1, use_colnames=True)

results = results[np.logical_and(results.support ⩽ 0.25, results.support ⩾ 0.1)]
results
```

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