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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Машинное обучение»

Студенты гр. 6304 _____ Тимофеев А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы

Ознакомиться с методами ассоциативного анализа из библиотеки MLxtend

Ход работы

Загрузка данных

1. Был создан датафрейм Pandas на основе загруженного датасета (https://www.kaggle.com/irfanasrullah/groceries)
2. Из датафрейма были убраны значения NaN, получен список уникальных товаров и их количество.

```
169 {'sweet spreads', 'newspapers', 'cocoa drinks', 'chewing gum', 'soap', 'canned beer', 'brown bread', 'dishes', 'UHT-milk', 'potato products', 'beverages', 'bottled water', 'fro
zen vegetables', 'packaged fruit/vegetables', 'instant coffee', 'tea', 'pork', 'tidbits', 'sound storage medium', 'canned vegetables', 'pastry', 'onions', 'liquor (appetizer)', 'ba
gs', 'syrup', 'seasonal products', 'liqueur', 'popcorn', 'other vegetables', 'ham', 'frozen fruits', 'butter milk', 'sparkling wine', 'citrus fruit', 'detergent', 'mayonnaise', 'ro
lls/buns', 'decalcifier', 'sugar', 'margarine', 'root vegetables', 'baby food', 'cookware', 'make up remover', 'chocolate marshmallow', 'prosecco', 'ready soups', 'domestic eggs',
'cake bar', 'male cosmetics', 'hygiene articles', 'salad dressing', 'flour', 'specialty cheese', 'cling film/bags', 'frozen dessert', 'coffee', 'abrasive cleaner', 'pet care', 'pre
servation products', 'sliced cheese', 'flower (seeds)', 'frankfurter', 'vinegar', 'toilet cleaner', 'fruit/vegetable juice', 'grapes', 'frozen fish', 'potted plants', 'honey', 'ski
n care', 'cream', 'whole milk', 'pasta', 'curd', 'liver loaf', 'light bulbs', 'hair spray', 'salty snack', 'semi-finished bread', 'condensed milk', 'curd cheese', 'baby cosmetics',
'kitchen towels', 'baking powder', 'cat food', 'candles', 'dish cleaner', 'mustard', 'kitchen utensil', 'snack products', 'spread cheese', 'artif. sweetener', 'oil', 'butter', 'des
sert', 'soft cheese', 'pip fruit', 'nut snack', 'beef', 'hard cheese', 'white wine', 'chicken', 'liquor', 'waffles', 'whipped/sour cream', 'jam', 'herbs', 'specialty bar', 'dog foo
d', 'rice', 'turkey', 'processed cheese', 'tropical fruit', 'misc. beverages', 'napkins', 'frozen potato products', 'bathroom cleaner', 'zwieback', 'house keeping products', 'spice
s', 'hamburger meat', 'red/blush wine', 'sausage', 'white bread', 'frozen meals', 'yogurt', 'meat', 'cream cheese', 'roll products', 'pickled vegetables', 'ice cream', 'fish', 'sau
ces', 'shopping bags', 'ketchup', 'female sanitary products', 'flower soil/fertilizer', 'photo/film', 'specialty vegetables', 'berries', 'softener', 'cooking chocolate', 'organic s
ausage', 'long life bakery product', 'nuts/prunes', 'Instant food products', 'specialty chocolate', 'pudding powder', 'rum', 'bottled beer', 'brandy', 'finished products', 'dental
care', 'cereals', 'soda', 'organic products', 'meat spreads', 'chocolate', 'candy', 'rubbing alcohol', 'salt', 'cleaner', 'soups', 'frozen chicken', 'canned fish', 'specialty fat',
'whisky', 'canned fruit'}
```

Рисунок 1 – Количество товаров и их список

FPGrowth и FPMax

1. Данные были преобразованы к виду пригодному для анализа, был проведен ассоциативный анализ с помощью алгоритма FPGrowth при уровне поддержки 0.3.

	support	itemsets
0	0.082766	(citrus fruit)
1	0.058566	(margarine)
2	0.139502	(yogurt)
3	0.104931	(tropical fruit)
4	0.058058	(coffee)
...	...	...
58	0.033249	(whole milk, pastry)
59	0.047382	(root vegetables, other vegetables)
60	0.048907	(root vegetables, whole milk)
61	0.030605	(rolls/buns, sausage)
62	0.032232	(whole milk, whipped/sour cream)

Рисунок 2 – Результаты ассоциативного анализа (FPGrowth)

2. Аналогичный анализ был проведен с применением алгоритма FPMax.

	support	itemsets
0	0.030402	(specialty chocolate)
1	0.031012	(onions)
2	0.032944	(hygiene articles)
3	0.033249	(berries)
4	0.033249	(hamburger meat)
5	0.033452	(UHT-milk)
6	0.033859	(sugar)
7	0.037112	(dessert)
8	0.037417	(long life bakery product)
9	0.037824	(salty snack)
10	0.038434	(waffles)
11	0.039654	(cream cheese)
12	0.042095	(white bread)
13	0.042908	(chicken)
14	0.048094	(frozen vegetables)
15	0.049619	(chocolate)
16	0.052364	(napkins)
17	0.052466	(beef)
18	0.053279	(curd)
19	0.055414	(butter)
20	0.057651	(pork)
21	0.058058	(coffee)
22	0.058566	(margarine)
23	0.058973	(frankfurter)
24	0.063447	(domestic eggs)
25	0.064870	(brown bread)
26	0.032232	(whole milk, whipped/sour cream)
27	0.072293	(fruit/vegetable juice)
28	0.030097	(whole milk, pip fruit)
29	0.077682	(canned beer)
30	0.079817	(newspapers)
31	0.080529	(bottled beer)
32	0.030503	(whole milk, citrus fruit)
33	0.033249	(whole milk, pastry)
34	0.030605	(rolls/buns, sausage)
35	0.098526	(shopping bags)
36	0.035892	(tropical fruit, other vegetables)
37	0.042298	(tropical fruit, whole milk)
38	0.047382	(root vegetables, other vegetables)
39	0.048907	(root vegetables, whole milk)
40	0.034367	(whole milk, bottled water)
41	0.034367	(rolls/buns, yogurt)
42	0.043416	(other vegetables, yogurt)
43	0.056024	(whole milk, yogurt)
44	0.032740	(soda, other vegetables)
45	0.038332	(rolls/buns, soda)
46	0.040061	(whole milk, soda)
47	0.042603	(rolls/buns, other vegetables)
48	0.056634	(rolls/buns, whole milk)
49	0.074835	(whole milk, other vegetables)

Рисунок 3 – Результаты ассоциативного анализа (FPMax)

3. Были определены минимальные и максимальные уровни поддержки соответственно для наборов различного размера. Результаты представлены в таблице 1.

Таблица 1 – Минимальные и максимальные уровни поддержки

Алгоритм	Размер набора	Minsup	Maxsup
FPGrowth	1	0.030	0.256
	2	0.030	0.075
FPMax	1	0.030	0.099
	2	0.030	0.075

Так как в алгоритме FPMax наборы меньшей длины входят в наборы большей, различия наблюдаются лишь в максимальном уровне поддержки для наборов размера 1.

4. Были построены гистограммы для 10 наиболее часто встречающихся товаров и наборов для каждого алгоритма.

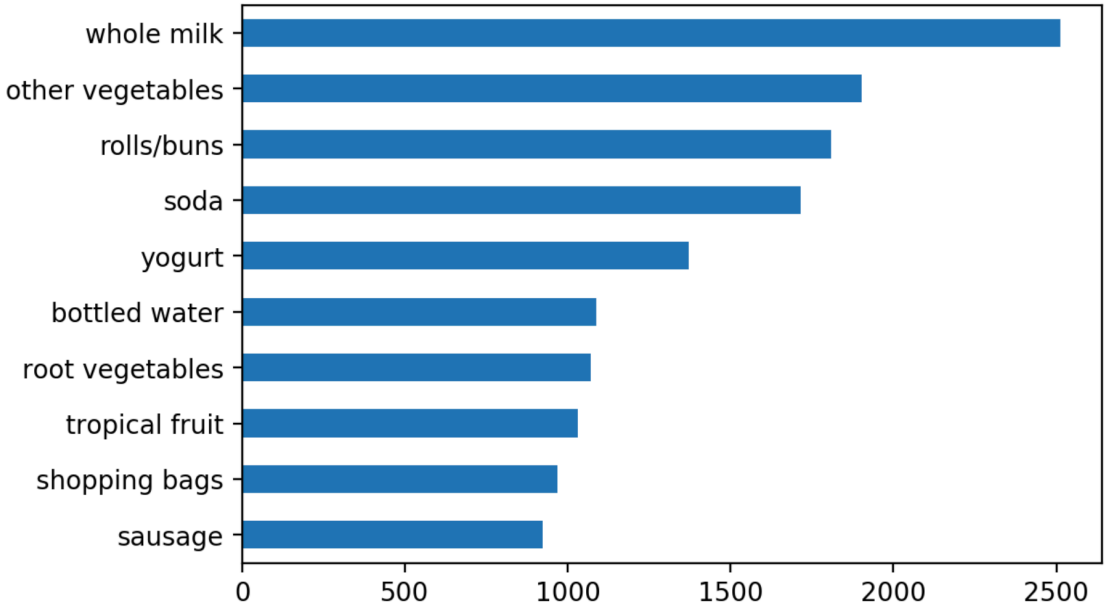

Рисунок 4 – 10 наиболее часто встречающихся товаров

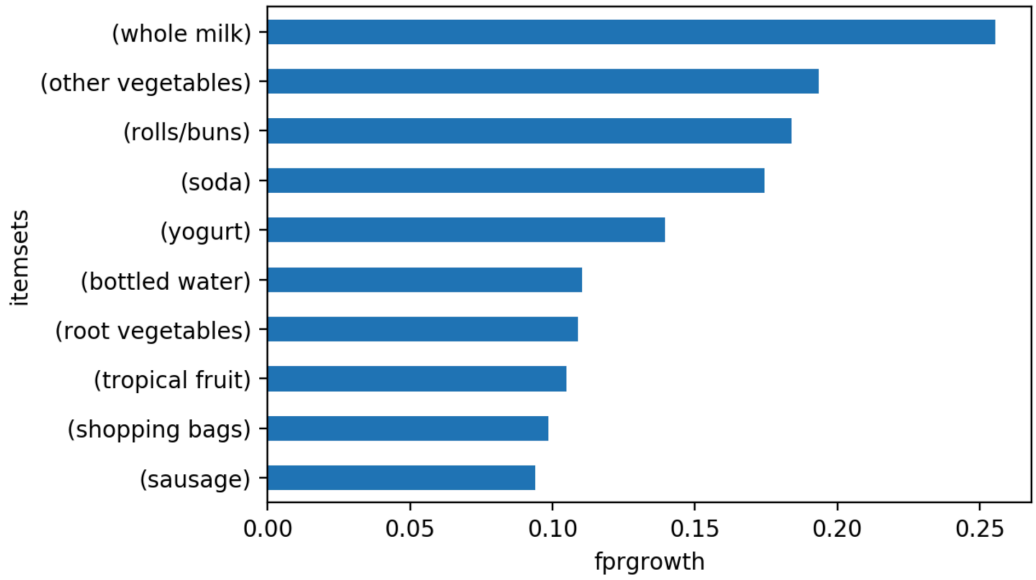

Рисунок 5 – 10 наиболее часто встречающихся наборов (FPGrowth)

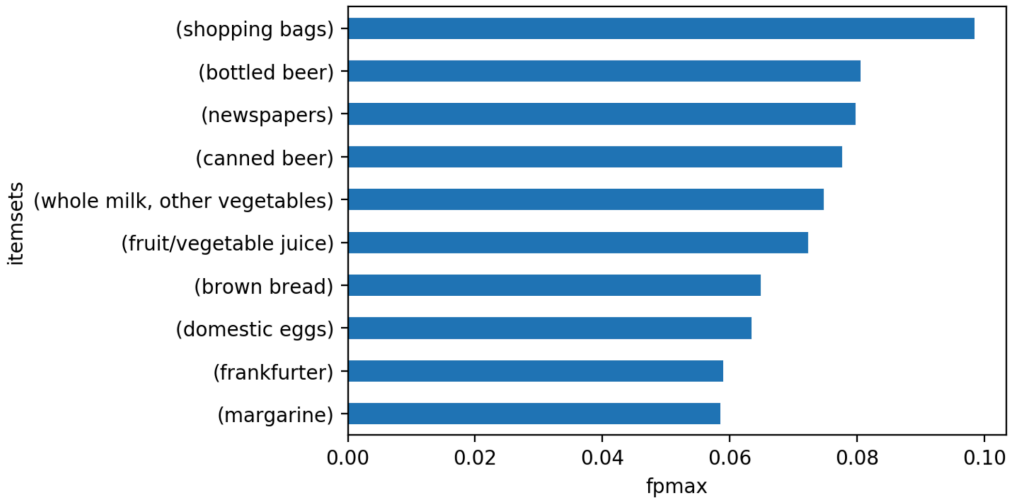

Рисунок 6 – 10 наиболее часто встречающихся наборов (FPMax)

5. Было произведено ограничение товаров, содержащихся в выборке и проведены анализы описанными ранее методами на новой выборке. Минимальные и максимальные уровни поддержки представлены в таблице 2.

	support	itemsets
0	0.082766	(citrus fruit)
1	0.139502	(yogurt)
2	0.104931	(tropical fruit)
3	0.255516	(whole milk)
4	0.193493	(other vegetables)
5	0.183935	(rolls/buns)
6	0.080529	(bottled beer)
7	0.110524	(bottled water)
8	0.174377	(soda)
9	0.088968	(pastry)
10	0.108998	(root vegetables)
11	0.077682	(canned beer)
12	0.093950	(sausage)
13	0.098526	(shopping bags)
14	0.071683	(whipped/sour cream)
15	0.057651	(pork)
16	0.030503	(whole milk, citrus fruit)
17	0.056024	(whole milk, yogurt)
18	0.034367	(rolls/buns, yogurt)
19	0.043416	(other vegetables, yogurt)
20	0.035892	(tropical fruit, other vegetables)
21	0.042298	(tropical fruit, whole milk)
22	0.074835	(whole milk, other vegetables)
23	0.042603	(rolls/buns, other vegetables)
24	0.056634	(rolls/buns, whole milk)
25	0.034367	(whole milk, bottled water)
26	0.038332	(rolls/buns, soda)
27	0.040061	(whole milk, soda)
28	0.032740	(soda, other vegetables)
29	0.033249	(whole milk, pastry)
30	0.047382	(root vegetables, other vegetables)
31	0.048907	(root vegetables, whole milk)
32	0.030605	(rolls/buns, sausage)
33	0.032232	(whole milk, whipped/sour cream)

Рисунок 7 – Результаты анализа на новой выборке (FPGrowth)

	support	itemsets
0	0.057651	(pork)
1	0.032232	(whole milk, whipped/sour cream)
2	0.077682	(canned beer)
3	0.080529	(bottled beer)
4	0.030503	(whole milk, citrus fruit)
5	0.033249	(whole milk, pastry)
6	0.030605	(rolls/buns, sausage)
7	0.098526	(shopping bags)
8	0.035892	(tropical fruit, other vegetables)
9	0.042298	(tropical fruit, whole milk)
10	0.047382	(root vegetables, other vegetables)
11	0.048907	(root vegetables, whole milk)
12	0.034367	(whole milk, bottled water)
13	0.034367	(rolls/buns, yogurt)
14	0.043416	(other vegetables, yogurt)
15	0.056024	(whole milk, yogurt)
16	0.032740	(soda, other vegetables)
17	0.038332	(rolls/buns, soda)
18	0.040061	(whole milk, soda)
19	0.042603	(rolls/buns, other vegetables)
20	0.056634	(rolls/buns, whole milk)
21	0.074835	(whole milk, other vegetables)

Рисунок 8 – Результаты анализа на новой выборке (FPMax)

Таблица 2 – Минимальные и максимальные уровни поддержки

Алгоритм	Размер набора	Minsup	Maxsup
FPGrowth	1	0.058	0.256
	2	0.031	0.075
FPMax	1	0.058	0.099
	2	0.030	0.075

Так как из выборки были удалены наборы с низким уровнем поддержки, изменения коснулись только значений наименьшего уровня поддержки для обоих алгоритмов.

6. Был построен график зависимости количества правил от уровня поддержки.

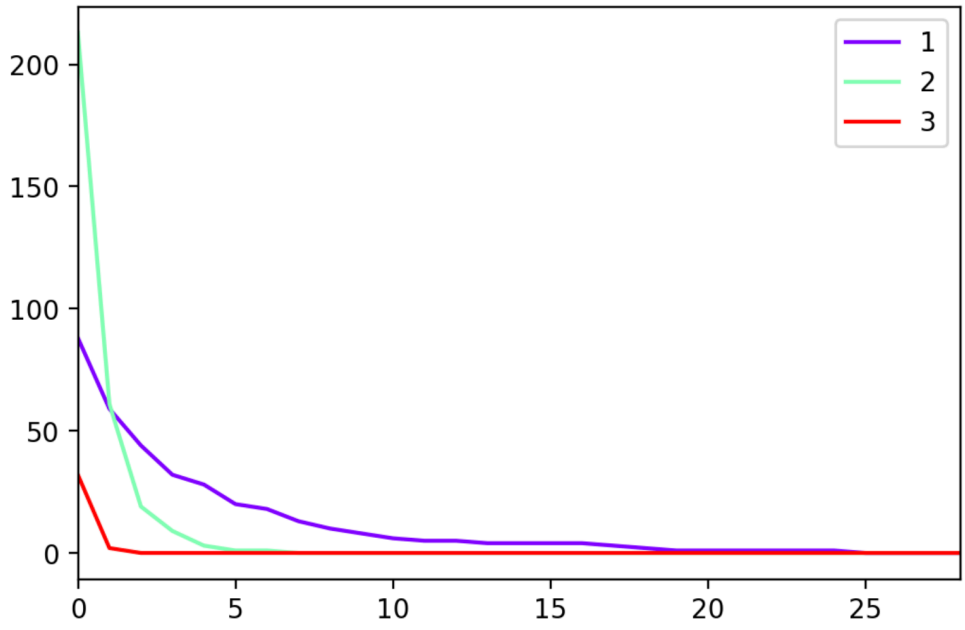

Рисунок 9 – График зависимости количества правил от уровня поддержки

Из графика видна обратная зависимость размера наборов от уровня поддержки.

Ассоциативные правила

1. Была сформирована выборка из определенных товаров с минимальным размером 2.
2. Были получены частоты наборов и проведен ассоциативный анализ.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(yogurt)	(whole milk)	0.241240	0.421869	0.110954	0.459933	1.090228	0.009183	1.070481
1	(yogurt)	(other vegetables)	0.241240	0.335079	0.085985	0.356427	1.063713	0.005150	1.033172
2	(tropical fruit)	(yogurt)	0.185864	0.241240	0.057994	0.312026	1.293423	0.013156	1.102890
3	(tropical fruit)	(other vegetables)	0.185864	0.335079	0.071083	0.382449	1.141370	0.008804	1.076706
4	(tropical fruit)	(whole milk)	0.185864	0.421869	0.083770	0.450704	1.068352	0.005359	1.052495
5	(whole milk)	(other vegetables)	0.421869	0.335079	0.148208	0.351313	1.048449	0.006849	1.025026
6	(other vegetables)	(whole milk)	0.335079	0.421869	0.148208	0.442308	1.048449	0.006849	1.036649
7	(rolls/buns)	(whole milk)	0.296214	0.421869	0.112163	0.378654	0.897564	-0.012801	0.930450
8	(bottled water)	(whole milk)	0.185461	0.421869	0.068063	0.366992	0.869921	-0.010177	0.913309
9	(bottled water)	(soda)	0.185461	0.267217	0.057390	0.309446	1.158033	0.007832	1.061153
10	(citrus fruit)	(whole milk)	0.146395	0.421869	0.060411	0.412655	0.978159	-0.001349	0.984313
11	(citrus fruit)	(other vegetables)	0.146395	0.335079	0.057189	0.390646	1.165836	0.008135	1.091192
12	(root vegetables)	(other vegetables)	0.196335	0.335079	0.093838	0.477949	1.426378	0.028050	1.273671
13	(root vegetables)	(whole milk)	0.196335	0.421869	0.096859	0.493333	1.169400	0.014031	1.141049
14	(sausage)	(rolls/buns)	0.167539	0.296214	0.060612	0.361779	1.221342	0.010985	1.102730
15	(sausage)	(whole milk)	0.167539	0.421869	0.059203	0.353365	0.837619	-0.011477	0.894062
16	(sausage)	(other vegetables)	0.167539	0.335079	0.053363	0.318510	0.950552	-0.002776	0.975687
17	(whipped/sour cream)	(whole milk)	0.124245	0.421869	0.063834	0.513776	1.217858	0.011419	1.189023
18	(whipped/sour cream)	(other vegetables)	0.124245	0.335079	0.057189	0.460292	1.373683	0.015557	1.232002
19	(pastry)	(whole milk)	0.150624	0.421869	0.065848	0.437166	1.036260	0.002304	1.027179

Рисунок 10 – Результаты анализа

Расчет был произведен на основе метрики *confidence*. Список метрик, их значения, формулы и диапазон представлены в таблице 3.

Таблица 3 – Метрики ассоциативного анализа

Метрика	Значение	Формула	Диапазон
antecedent support	Поддержка A	$\sup(A)$	[0,1]
consequent support	Поддержка C	$\sup(C)$	[0,1]
support	Поддержка набора из A и C	$\sup(A \rightarrow C) = \sup(A \cup C)$	[0,1]
confidence	Вероятность увидеть C в транзакции, содержащей A. Confidence 1, если A и C всегда находятся вместе в транзакциях.	$\text{confidence}(A \rightarrow C) = \frac{\sup(A \rightarrow C)}{\sup(A)}$	[0,1]
lift	Как часто A и C возникают вместе, чем если бы они были статистически независимы. Если A и C независимы, то lift = 1.	$\text{lift}(A \rightarrow C) = \frac{\text{confidence}(A \rightarrow C)}{\sup(C)}$	$[0, \infty]$
leverage	Разница между частотой появления A и C вместе с частотой, при независимых A и C. Если A и C независимы, то leverage = 0.	$\text{leverage}(A \rightarrow C) = \sup(A \rightarrow C) - \sup(A) \times \sup(C)$	[-1,-1]
conviction	Большое значение означает, что C сильно зависит от A. Если A и C независимы, то conviction	$\text{conviction}(A \rightarrow C) = \frac{1 - \sup(C)}{1 - \text{confidence}(A \rightarrow C)}$	$[0, \infty]$

	= 1. При абсолютной зависимости conviction=∞		

3. Были построены ассоциативные правила для различных метрик. Полученные значения среднего, медианы и СКО представлены в таблице 4.

   Таблица 4 – Статистические меры метрик

Метрика	Среднее	Медиана	СКО
support	0.074685	0.066955	0.022549
lift	1.042997	1.056081	0.183264
leverage	0.015533	0.013594	0.006063
conviction	1.017200	1.022851	0.083993
confidence	0.289579	0.264439	0.103683

4. Для полученных правил для метрики *confidence* был построен граф, ориентированный от антецедента к консеквенту, с шириной ребер, отображающих уровень поддержки. Граф представлен на рисунке 11.

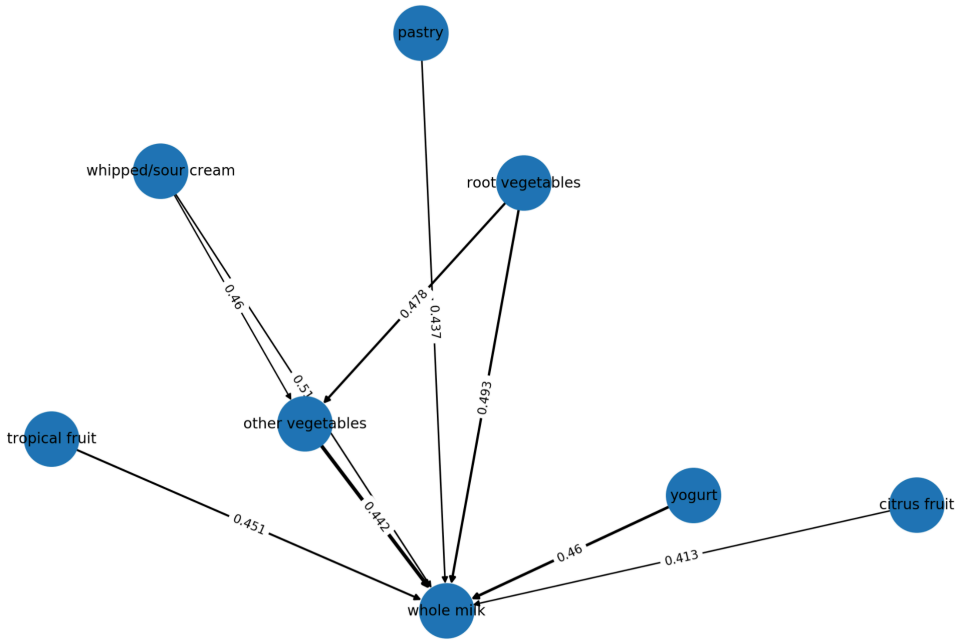

Рисунок 11 – Полученный график

5. В качестве иного метода визуализации правил была использована тепловая карта, представленная на рисунке 12.

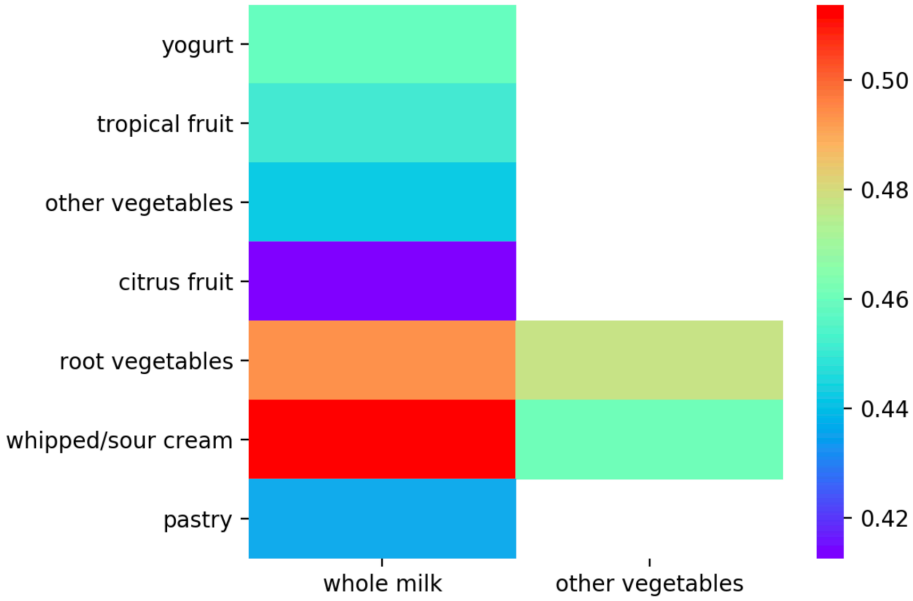

Рисунок 12 – Полученная тепловая карта

Выводы

В ходе данной лабораторной работы изучены методы ассоциативного анализа из библиотеки *MLxtend*, определены их различия.

ПРИЛОЖЕНИЕ А

Исходный код

```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth, fpmax, association_rules
import networkx as nx
import seaborn as sns

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


all_data = pd.read_csv('groceries - groceries.csv')
print(all_data)

np_data = all_data.to_numpy()
np_data = [[elem for elem in row[1:] if isinstance(elem,str)] for row in np_data]

unique_items = set()
items_with_count = {}
for row in np_data:
    for elem in row:
        unique_items.add(elem)
print(len(unique_items), unique_items)

te = TransactionEncoder()
te_ary = te.fit(np_data).transform(np_data)
data = pd.DataFrame(te_ary, columns=te.columns_)
data

result_fpgrowth = fpgrowth(data, min_support=0.03, use_colnames = True)
result_fpgrowth

def printMinMaxSupport(res):
    curr_len = 1
    supports = res[res['itemsets'].apply(lambda x: len(x) == curr_len)]['support']
    while len(supports):
        print('Количество элементов: {}, minsup: {:.3f}, maxsup:
{:.3f}'.format(curr_len, np.min(supports), np.max(supports)))
        curr_len += 1
        supports = res[res['itemsets'].apply(lambda x: len(x) ==
curr_len)]['support']

printMinMaxSupport(result_fpgrowth)

result_fpmax = fpmax(data, min_support=0.03, use_colnames = True)
result_fpmax

printMinMaxSupport(result_fpmax)

largest = data.sum().nlargest(10)
print(largest, largest.shape)
largest.sort_values().plot.barh()

plt.xlabel('fprgrowth')
result_fpgrowth.set_index('itemsets')['support'].nlargest(10).sort_values().plot.barh
()
```

```
plt.xlabel('fpmax')
result_fpmax.set_index('itemsets')['support'].nlargest(10).sort_values().plot.barh()

items = ['whole milk', 'yogurt', 'soda', 'tropical fruit', 'shopping bags',
'sausage', 'whipped/sour cream', 'rolls/buns', 'other vegetables', 'root vegetables',
'pork', 'bottled water', 'pastry', 'citrus fruit', 'canned beer',
'bottled beer']
np_data = all_data.to_numpy()
np_data = [[elem for elem in row[1:] if isinstance(elem,str) and elem in items] for
row in np_data]

te = TransactionEncoder()
te_ary = te.fit(np_data).transform(np_data)
new_data = pd.DataFrame(te_ary, columns=te.columns_)
new_data

result_fpgrowth_new = fpgrowth(new_data, min_support=0.03, use_colnames = True)
result_fpgrowth_new

result_fpmax_new = fpmax(new_data, min_support=0.03, use_colnames = True)
result_fpmax_new

printMinMaxSupport(result_fpgrowth_new)

printMinMaxSupport(result_fpmax_new)

min_supports = np.arange(0.01, 0.3, 0.01)
fpg_results = []
for min_sup in min_supports:
        fpg_res = fpgrowth(data, min_support=min_sup, use_colnames = True)
        fpg_res['length'] = fpg_res['itemsets'].apply(lambda x: len(x))
        fpg_count = fpg_res.groupby('length').count().apply(list)['itemsets']
        fpg_np_count = np.array(fpg_count)
        if not np.all(np.isnan(fpg_np_count)):
            fpg_results.append(fpg_np_count)
        else:
            fpg_results.append([0, 0, 0])

fpg_df = pd.DataFrame(fpg_results).fillna(value=0)
fpg_df.columns += 1
fpg_df.plot(colormap='rainbow')

np_data = all_data.to_numpy()
np_data = [[elem for elem in row[1:] if isinstance(elem,str) and elem in items] for
row in np_data]
np_data = [row for row in np_data if len(row) > 1]
te = TransactionEncoder()
te_array = te.fit_transform(np_data)
data = pd.DataFrame(te_array, columns=te.columns_)
data

result = fpgrowth(data, min_support=0.05, use_colnames = True)
result

rules = association_rules(result, min_threshold = 0.3)
rules

rules_c = {}
metrics = ['support', 'lift', 'leverage', 'conviction', 'confidence']
for metric in metrics:
    rules_c[metric] = association_rules(result, min_threshold = 0.01, metric=metric)
```

```
print('mean: \n{}\n median: \n{}\n std: \n{}\n\n'.format(rules_c['support'].mean(),
rules_c['support'].median(), rules_c['support'].std()))
rules_c['support']

print('mean: \n{}\n median: \n{}\n std: \n{}\n\n'.format(rules_c['lift'].mean(),
rules_c['lift'].median(), rules_c['lift'].std()))
rules_c['lift']

print('mean: \n{}\n median: \n{}\n std: \n{}\n\n'.format(rules_c['leverage'].mean(),
rules_c['leverage'].median(), rules_c['leverage'].std()))
rules_c['leverage']

print('mean: \n{}\n median: \n{}\n std:
\n{}\n\n'.format(rules_c['conviction'].mean(), rules_c['conviction'].median(),
rules_c['conviction'].std()))
rules_c['conviction']

print('mean: \n{}\n median: \n{}\n std:
\n{}\n\n'.format(rules_c['confidence'].mean(), rules_c['confidence'].median(),
rules_c['confidence'].std()))
rules_c['confidence']

rules_g = association_rules(result, min_threshold = 0.4, metric='confidence')

digraph = nx.DiGraph()
for rule in rules_g.itertuples(index=False):
    digraph.add_edge(rule.antecedents, rule.consequents, weight=rule.support,
label=round(rule.confidence, 3))

plt.figure(figsize=(12, 8))
pos = nx.spring_layout(digraph)
nx.draw(digraph, pos,
    labels={node: '\n'.join(node) for node in digraph.nodes()},
    width=[digraph[u][v]['weight']*20 for u,v in digraph.edges()],
    node_size=2000
)

nx.draw_networkx_edge_labels(digraph, pos,
edge_labels=nx.get_edge_attributes(digraph, 'label'))
plt.axis('off')
plt.show()

rules_hm = rules_g.pivot(index='antecedents', columns='consequents',
values='confidence')
rules_hm.index = ['\n'.join(ind) for ind in rules_hm.index]
rules_hm.columns = ['\n'.join(col) for col in rules_hm.columns]
sns.heatmap(rules_hm, cmap='rainbow')
plt.tight_layout()
plt.show()
```