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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Машинное обучение»

Тема: Ассоциативный анализ

Студент гр. 8304 _____ Кирьянов Д.И

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2021

Цель работы

Ознакомиться с методами ассоциативного анализа из библиотеки MLxtend.

Ход работы

Загрузка данных

Создан Python скрипт. Загружены данные в датафрейм

	Item(s)	Item 1	Item 2	...	Item 30	Item 31	Item 32
0	4	citrus fruit	semi-finished bread	...	NaN	NaN	NaN
1	3	tropical fruit	yogurt	...	NaN	NaN	NaN
2	1	whole milk	NaN	...	NaN	NaN	NaN
3	4	pip fruit	yogurt	...	NaN	NaN	NaN
4	4	other vegetables	whole milk	...	NaN	NaN	NaN
...	...	...	...	...	...	...	...
9830	17	sausage	chicken	...	NaN	NaN	NaN
9831	1	cooking chocolate	NaN	...	NaN	NaN	NaN
9832	10	chicken	citrus fruit	...	NaN	NaN	NaN
9833	4	semi-finished bread	bottled water	...	NaN	NaN	NaN
9834	5	chicken	tropical fruit	...	NaN	NaN	NaN

Рис 1 – Загруженные данные

Получен список всех уникальных товаров. Выведен список товаров, а также их количество.

```
{'cake bar', 'grapes', 'rolls/buns', 'oil',
169
```

Рис 2 – Список товаров и их количество

Полный список: {'cake bar', 'grapes', 'rolls/buns', 'oil', 'specialty cheese', 'decalcifier', 'spread cheese', 'sound storage medium', 'salad dressing', 'sausage', 'bathroom cleaner', 'whole milk', 'cleaner', 'pickled vegetables', 'toilet cleaner', 'pasta', 'specialty fat', 'flower (seeds)', 'soft cheese', 'berries', 'specialty chocolate', 'tea', 'other vegetables', 'detergent', 'cookware', 'bags', 'rum', 'waffles', 'honey', 'curd', 'artif. sweetener', 'pip fruit', 'liquor', 'syrup', 'cat food', 'meat', 'organic products', 'bottled beer', 'salt', 'candy', 'brown bread', 'abrasive cleaner', 'canned fish', 'whisky', 'citrus fruit', 'hard cheese', 'candles', 'frozen vegetables', 'canned beer', 'pastry', 'female sanitary products', 'butter milk', 'flower soil/fertilizer', 'curd cheese', 'rice', 'baby cosmetics', 'finished products', 'jam', 'ketchup', 'popcorn', 'instant coffee', 'coffee', 'frozen chicken', 'dessert', 'soap', 'bottled water',

'pudding powder', 'meat spreads', 'canned vegetables', 'house keeping products', 'ready soups', 'brandy', 'sliced cheese', 'hamburger meat', 'yogurt', 'white bread', 'beef', 'chicken', 'dental care', 'frankfurter', 'specialty vegetables', 'skin care', 'prosecco', 'pork', 'beverages', 'margarine', 'fish', 'rubbing alcohol', 'condensed milk', 'whipped/sour cream', 'shopping bags', 'onions', 'dog food', 'potato products', 'hygiene articles', 'kitchen towels', 'cream cheese', 'frozen fish', 'Instant food products', 'liqueur', 'tidbits', 'canned fruit', 'mustard', 'UHT-milk', 'male cosmetics', 'white wine', 'turkey', 'cooking chocolate', 'frozen fruits', 'herbs', 'liver loaf', 'organic sausage', 'kitchen utensil', 'spices', 'seasonal products', 'butter', 'dishes', 'processed cheese', 'light bulbs', 'ice cream', 'softener', 'soups', 'soda', 'sweet spreads', 'nuts/prunes', 'napkins', 'domestic eggs', 'potted plants', 'frozen dessert', 'frozen potato products', 'make up remover', 'semi-finished bread', 'packaged fruit/vegetables', 'sauces', 'specialty bar', 'cream', 'cling film/bags', 'dish cleaner', 'vinegar', 'red/blush wine', 'baking powder', 'zwieback', 'cereals', 'pet care', 'chocolate', 'misc. beverages', 'newspapers', 'chocolate marshmallow', 'photo/film', 'chewing gum', 'sparkling wine', 'root vegetables', 'fruit/vegetable juice', 'roll products', 'ham', 'preservation products', 'hair spray', 'mayonnaise', 'tropical fruit', 'flour', 'sugar', 'baby food', 'liquor (appetizer)', 'cocoa drinks', 'salty snack', 'nut snack', 'long life bakery product', 'snack products', 'frozen meals'}

FPGrowth и FPMax

Преобразованы данные к виду, удобному для анализа:

	Instant food products	UHT-milk	...	yogurt	zwieback
0	False	False	...	False	False
1	False	False	...	True	False
2	False	False	...	False	False
3	False	False	...	True	False
4	False	False	...	False	False
...	...	...	...	...	...
9830	False	False	...	False	False
9831	False	False	...	False	False
9832	False	False	...	True	False
9833	False	False	...	False	False
9834	False	False	...	False	False

Рис 3 – Преобразованные данные

Проведен ассоциативный анализ используя алгоритм FPGrowth при уровне поддержки 0.03:

	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	(pastry, whole milk)
59	0.047382	(root vegetables, other vegetables)
60	0.048907	(root vegetables, whole milk)
61	0.030605	(sausage, rolls/buns)
62	0.032232	(whipped/sour cream, whole milk)

Рис 4 – Результат FPGrowth при minSup=0.03

Проанализированы получившиеся варианты. Определены минимальное и максимальное значения для уровня поддержки для набора из 1,2:

```
len 1 min: 0.03040162684290798
len 1 max: 0.25551601423487547
len 2 min: 0.030096593797661414
len 2 max: 0.07483477376715811
```

Рис 5 – Анализ полученных данных

Проведен аналогичный анализ используя алгоритм 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	(whipped/sour cream, whole milk)
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	(citrus fruit, whole milk)
33	0.033249	(pastry, whole milk)
34	0.030605	(sausage, rolls/buns)
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	(bottled water, whole milk)
41	0.034367	(yogurt, rolls/buns)
42	0.043416	(yogurt, other vegetables)
43	0.056024	(yogurt, whole milk)
44	0.032740	(soda, other vegetables)
45	0.038332	(soda, rolls/buns)
46	0.040061	(soda, whole milk)
47	0.042603	(rolls/buns, other vegetables)
48	0.056634	(rolls/buns, whole milk)
49	0.074835	(whole milk, other vegetables)

Рис 6 – Результат FPMax

Был проведен анализ полученных данных:

```
len 1 min: 0.03040162684290798
len 1 max: 0.09852567361464158
len 2 min: 0.030096593797661414
len 2 max: 0.07483477376715811
```

Рис 7 – Анализ полученных данных

Результаты для наборов длины 2 не изменились. FPMax – это вариант FPGrowth, в который входят только максимальные наборы элементов. Набор элементов максимальный, если он является частым и не встречается такого частого супер-шаблона, содержащего его.

Построена гистограмму для каждого товара. Отображен результат только для 10 самых встречаемых товаров:

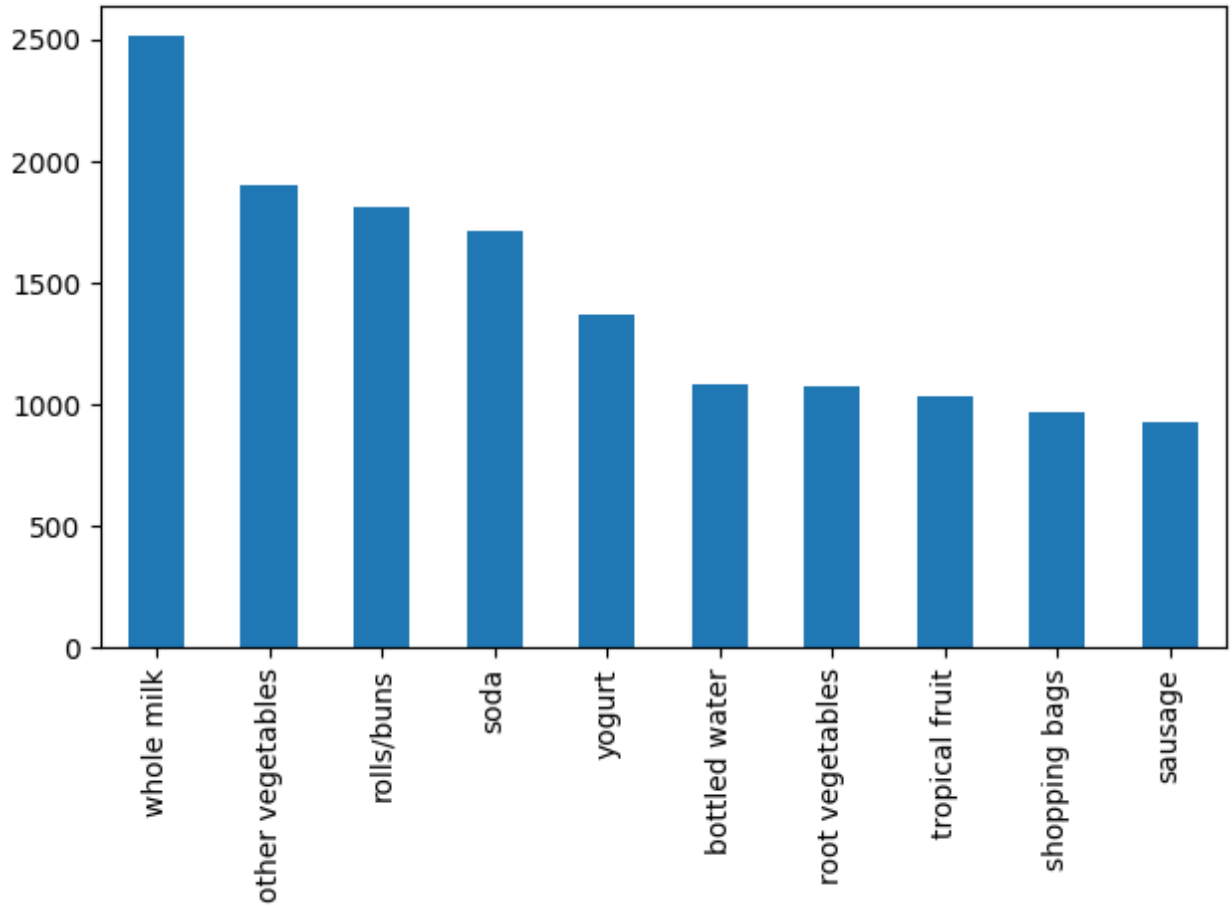

Рис 8 – Гистограмма для каждого товара

Проведен анализ FPGrowth и FPMax для нового набора данных:

	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	(other vegetables, whole milk)
23	0.042603	(other vegetables, rolls/buns)
24	0.056634	(rolls/buns, whole milk)
25	0.034367	(whole milk, bottled water)
26	0.038332	(rolls/buns, soda)
27	0.040061	(soda, whole milk)
28	0.032740	(other vegetables, soda)
29	0.033249	(pastry, whole milk)
30	0.047382	(root vegetables, other vegetables)
31	0.048907	(root vegetables, whole milk)
32	0.030605	(rolls/buns, sausage)
33	0.032232	(whipped/sour cream, whole milk)

Рис 9 – FPGrowth для нового набора данных

```
len 1 min: 0.05765124555160142
len 1 max: 0.25551601423487547
len 2 min: 0.030503304524656837
len 2 max: 0.07483477376715811
```

Рис 10 – Анализ полученных данных

	support	itemsets
0	0.057651	(pork)
1	0.032232	(whipped/sour cream, whole milk)
2	0.077682	(canned beer)
3	0.080529	(bottled beer)
4	0.030503	(whole milk, citrus fruit)
5	0.033249	(pastry, whole milk)
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	(other vegetables, soda)
17	0.038332	(rolls/buns, soda)
18	0.040061	(soda, whole milk)
19	0.042603	(other vegetables, rolls/buns)
20	0.056634	(rolls/buns, whole milk)
21	0.074835	(other vegetables, whole milk)

Рис 11 – FPMax для нового набора данных

```
len 1 min: 0.05765124555160142
len 1 max: 0.09852567361464158
len 2 min: 0.030503304524656837
len 2 max: 0.07483477376715811
```

Рис 12 – Анализ полученных данных

Построен график изменения количества получаемых правил от уровня поддержки.

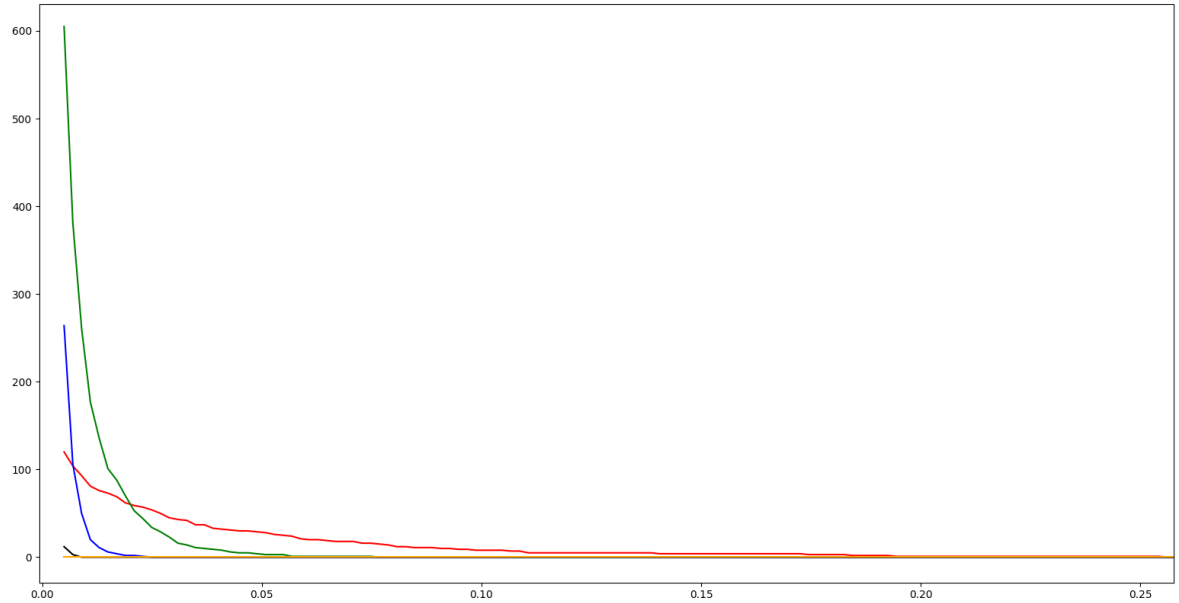

Рис 13 – График изменения количества правил от поддержки для разных длин наборов

1 - красный, 2 - зеленый, 3 - синий, 4 - черный, 5 - оранженый.

Ассоциативные правила

Проведен ассоциативный анализ:

	antecedents	consequents	...	leverage	conviction
0	(yogurt)	(whole milk)	...	0.009183	1.070481
1	(yogurt)	(other vegetables)	...	0.005150	1.033172
2	(tropical fruit)	(yogurt)	...	0.013156	1.102890
3	(tropical fruit)	(other vegetables)	...	0.008804	1.076706
4	(tropical fruit)	(whole milk)	...	0.005359	1.052495
5	(other vegetables)	(whole milk)	...	0.006849	1.036649
6	(whole milk)	(other vegetables)	...	0.006849	1.025026
7	(rolls/buns)	(whole milk)	...	-0.012801	0.930450
8	(bottled water)	(whole milk)	...	-0.010177	0.913309
9	(bottled water)	(soda)	...	0.007832	1.061153
10	(citrus fruit)	(whole milk)	...	-0.001349	0.984313
11	(citrus fruit)	(other vegetables)	...	0.008135	1.091192
12	(root vegetables)	(other vegetables)	...	0.028050	1.273671
13	(root vegetables)	(whole milk)	...	0.014031	1.141049
14	(sausage)	(rolls/buns)	...	0.010985	1.102730
15	(sausage)	(whole milk)	...	-0.011477	0.894062
16	(sausage)	(other vegetables)	...	-0.002776	0.975687
17	(whipped/sour cream)	(whole milk)	...	0.011419	1.189023
18	(whipped/sour cream)	(other vegetables)	...	0.015557	1.232002
19	(pastry)	(whole milk)	...	0.002304	1.027179

Рис 14 – Результаты анализа

Расчет проводится на основе метрики “confidence”

confidence - $\text{confidence}(A \rightarrow B) = \frac{\text{support}(A \rightarrow B)}{\text{support}(A)}$. Вероятность увидеть консеквент в транзакции при условии, что оно также содержит антецедент. Метрика не является симметричной и направленной. Значение 1 – максимальное для правила $A \rightarrow B$, если консеквент и антецедент всегда встречаются вместе.

lift - $\text{lift}(A \rightarrow B) = \frac{\text{confidence}(A \rightarrow B)}{\text{support}(B)}$. Насколько чаще предшествующее и последующее действие правила $A \rightarrow B$ встречается вместе, чем ожидалось, если бы они были статически независимыми. Lift = 1, если A и B независимы.

leverage - $\text{leverage}(A \rightarrow B) = \text{support}(A \rightarrow B) - \text{support}(A) \times \text{support}(B)$. Разница между наблюдаемой частотой появления A и B вместе и частотой, которую можно ожидать, если A и B были бы независимыми. При leverage = 0 предметы независимы.

conviction - $\text{conviction}(A \rightarrow B) = \frac{1 - \text{support}(B)}{1 - \text{confidence}(A \rightarrow B)}$. Насколько консеквент сильно зависит от антецедента. При conviction = 1 предметы независимы.

Рассчитаны среднее значение, медиана и СКО для каждой метрики:

```
support
Mean: 0.07468477957805385
Median: 0.06695529601288763
Std: 0.022549155077774167
confidence
Mean: 0.2895787148360594
Median: 0.2644391408114558
Std: 0.10368277946597466
lift
Mean: 1.0429974993487907
Median: 1.0560808232246663
Std: 0.1832639607239429
leverage
Mean: 0.015533118831358378
Median: 0.01359374637587954
Std: 0.006063340549426105
conviction
Mean: 1.0171997189136908
Median: 1.022850930928648
Std: 0.08399289113954451
```

Рис 15 – Результаты вычислений

Построен граф для следующего анализа:

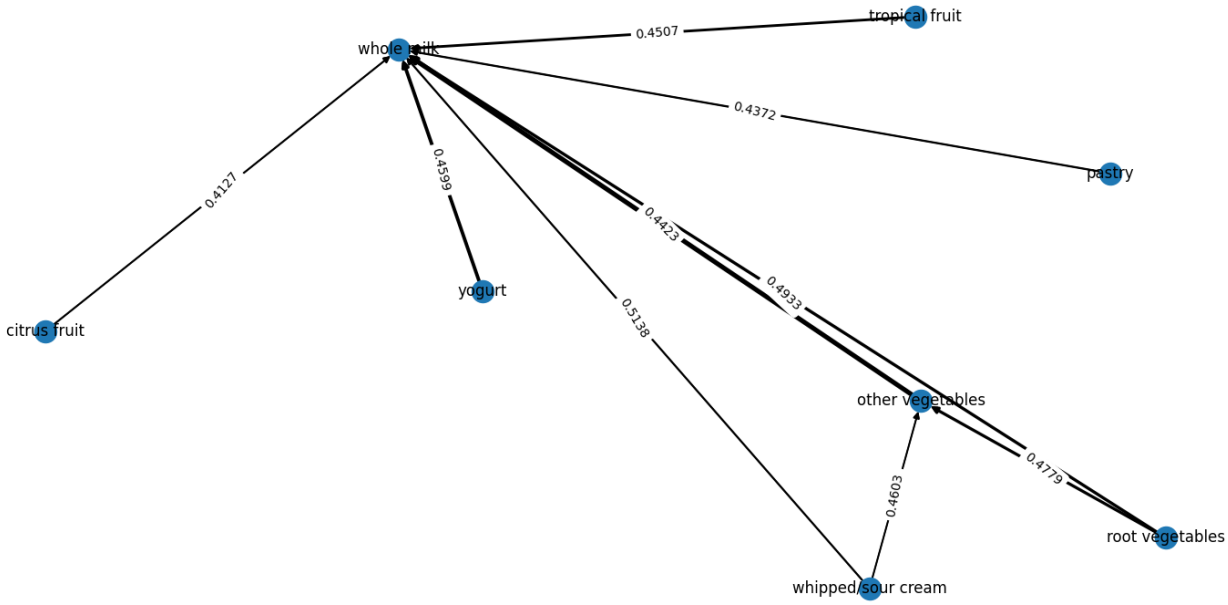

Рис 16 – Граф анализа

Вывод

Ознакомились с ассоциативного частотного анализа из библиотеки MLxtend.

ПРИЛОЖЕНИЕ А

Исходный код программы

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

all_data = pd.read_csv('groceries - groceries.csv')
print(all_data) #Видно, что датафрейм содержит NaN значения

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()
for row in np_data:
    for elem in row:
        unique_items.add(elem)

print(unique_items)
print(len(unique_items))

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

result = fpgrowth(data, min_support=0.03, use_colnames = True)
print(result)
result['length'] = result['itemsets'].apply(lambda x: len(x))
result_1 = result[result['length'] == 1]
print("len 1 min:", min(result_1['support']))
print("len 1 max:", max(result_1['support']))
result_2 = result[result['length'] == 2]
print("len 2 min:", min(result_2['support']))
print("len 2 max:", max(result_2['support']))

result = fpmax(data, min_support=0.03, use_colnames=True)
print(result)
result['length'] = result['itemsets'].apply(lambda x: len(x))
result_1 = result[result['length'] == 1]
print("len 1 min:", min(result_1['support']))
print("len 1 max:", max(result_1['support']))
result_2 = result[result['length'] == 2]
print("len 2 min:", min(result_2['support']))
print("len 2 max:", max(result_2['support']))

count_of_items = data.sum()
count_of_items.nlargest(10).plot.bar()
plt.tight_layout()
plt.show()

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)
data_ = pd.DataFrame(te_ary, columns=te.columns_)
print(data_)
result = fpgrowth(data_, min_support=0.03, use_colnames = True)
print(result)
result['length'] = result['itemsets'].apply(lambda x: len(x))
result_1 = result[result['length'] == 1]
print("len 1 min:", min(result_1['support']))
print("len 1 max:", max(result_1['support']))
result_2 = result[result['length'] == 2]
print("len 2 min:", min(result_2['support']))
print("len 2 max:", max(result_2['support']))

result = fpmax(data_, min_support=0.03, use_colnames=True)
print(result)
result['length'] = result['itemsets'].apply(lambda x: len(x))
result_1 = result[result['length'] == 1]
print("len 1 min:", min(result_1['support']))
print("len 1 max:", max(result_1['support']))
result_2 = result[result['length'] == 2]
print("len 2 min:", min(result_2['support']))
print("len 2 max:", max(result_2['support']))

colors = ['r', 'g', 'b', 'black', 'orange']
for i in range(1, 6):
    arr = []
    for minSup in np.linspace(0.005, 1.0, 50):
        results = fpgrowth(data, min_support=minSup, use_colnames=True,
max_len=i)
        results['length'] = results['itemsets'].apply(lambda x: len(x))
        results = results[results['length'] == i]
        arr.append(len(results))
    plt.plot(np.linspace(0.005, 1, 50), arr, colors[i - 1])
plt.show()

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_ary = te.fit(np_data).transform(np_data)
data = pd.DataFrame(te_ary, columns=te.columns_)
result = fpgrowth(data, min_support=0.05, use_colnames = True)
print(result)
rules = association_rules(result, min_threshold = 0.3)
print(rules)
metrics = ["support", "confidence", "lift", "leverage", "conviction"]
for i in metrics:
    print(i)
    rules = association_rules(result, min_threshold=0.01, metric=i)
    print("Mean:", rules[i].mean())
    print("Median:", rules[i].median())
    print("Std:", rules[i].std())
rules = association_rules(result, min_threshold = 0.4, metric='confidence')
G = nx.DiGraph()

for index, row in rules.iterrows():
    l = list(row['antecedents'])[0]
    r = list(row['consequents'])[0]
    w = row['support'] * 25
    label = round(row['confidence'], 4)
```

```
    G.add_edge(l, r, label=label, weight=w)
pos = nx.spring_layout(G)
plt.figure()
nx.draw_networkx(G, pos, with_labels=True)
nx.draw_networkx_edges(G, pos, width=list([G[n1][n2]['weight'] for n1, n2 in
G.edges]))
nx.draw_networkx_edge_labels(G, pos,edge_labels=dict([((n1, n2),
f'{G[n1][n2]["label"]}') for n1, n2 in G.edges]), font_color='black')
plt.show()
```