**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Машинное обучение»**

Тема: Частотный анализ

|  |  |  |
| --- | --- | --- |
| Студент гр. 8304 |  | Кирьянов Д.И. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2021

**Цель работы.**

Ознакомиться с методами частотного анализа из библиотеки MLxtend.

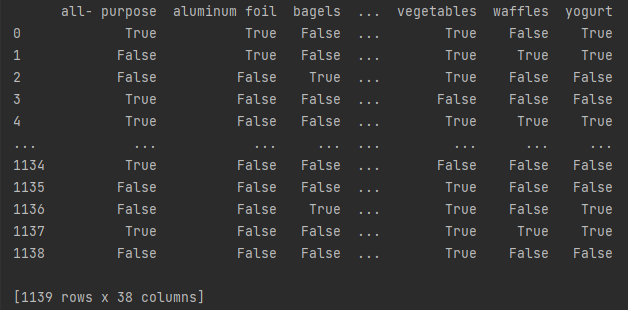
**Ход работы.**

1. **Загрузка данных**
   1. Создан Python скрипт. Загружены данные в датафрейм.
   2. Получены списки всех id покупателей и всех товаров, которые есть в файле.



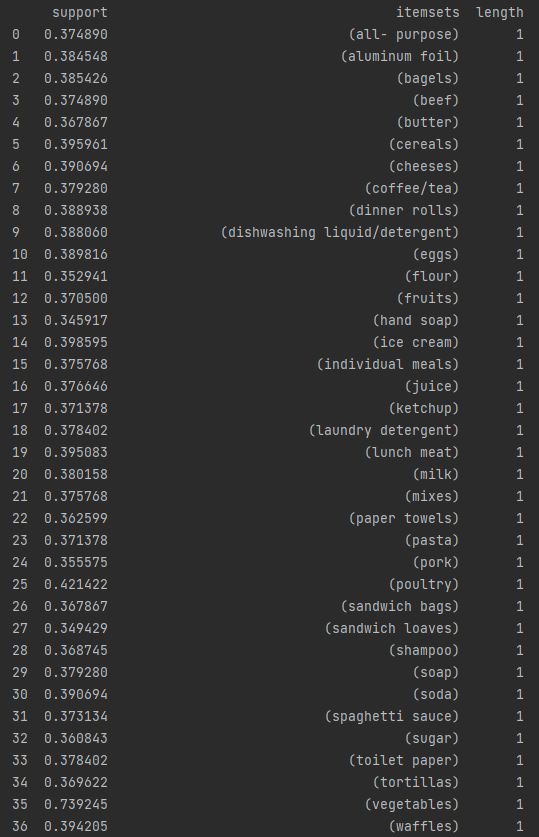
1. **Подготовка данных**

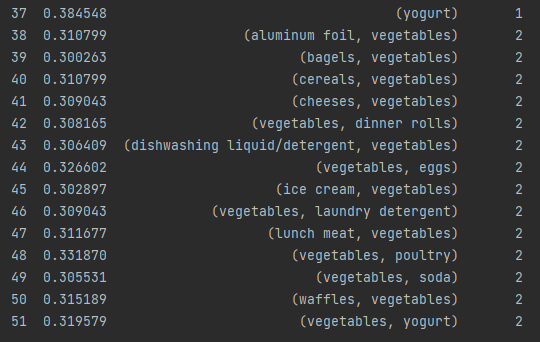
Закодированы данные при помощи TransactionEncoder. Выведен полученный dataframe.



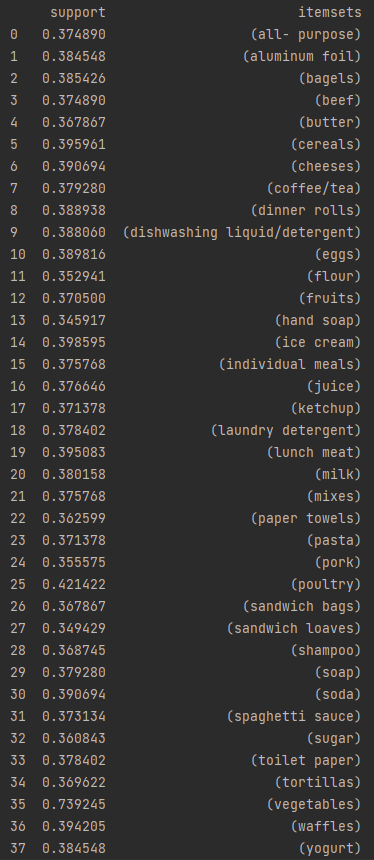
Данные стали представляться таким образом, что для каждого id покупателя формируется булевый список товаров.

1. **Ассоциативный анализ с использованием алгоритма Apriori**
   1. Применён алгоритм apriori с уровнем поддержки 0.3.

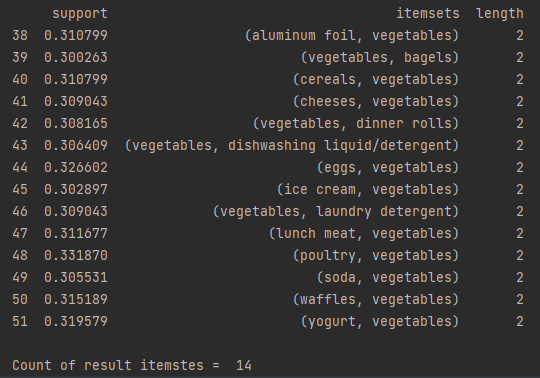




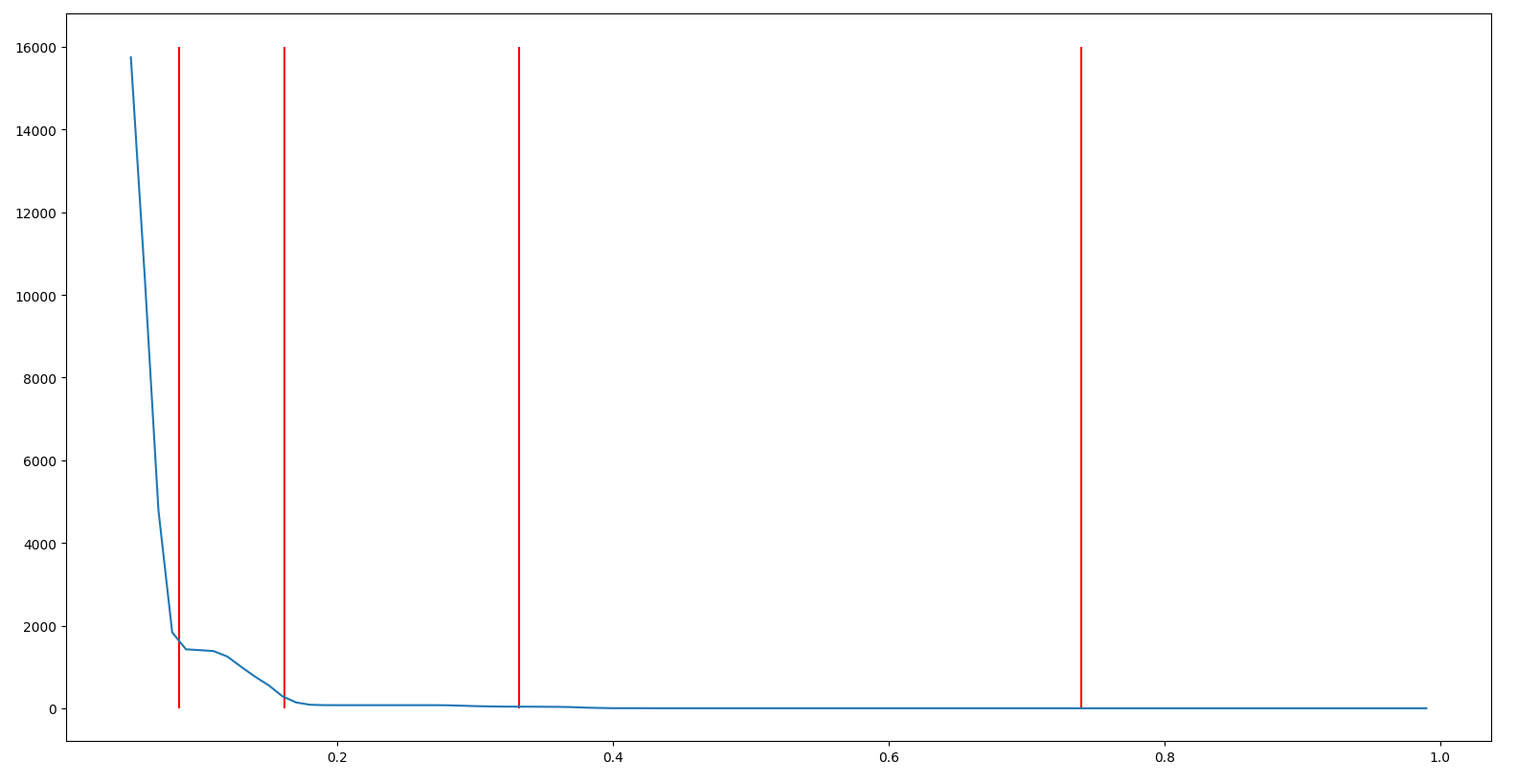
* 1. Применен алгоритм apriori с тем же уровнем поддержки, но ограничим максимальный размер набора единицей.



* 1. Применен алгоритм apriori и выведены только те наборы, которые имеют размер 2, а также количество таких наборов.



* 1. Построен график зависимости количества наборов от уровня поддержки. Определены значения уровня поддержки при котором перестают генерироваться наборы размера 1,2,3, и т.д. Полученные уровни отмечены красным цветом.



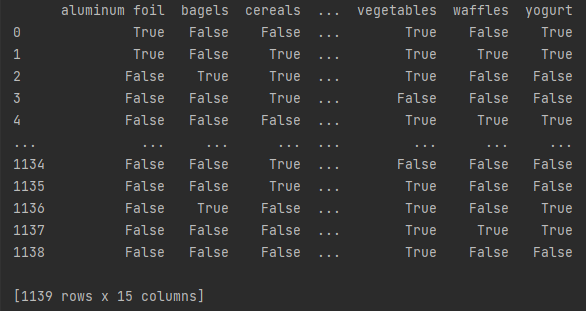
Для набора размером 1 значение = 0.7392449517120281

Для набора размером 2 значение = 0.33187006145741876

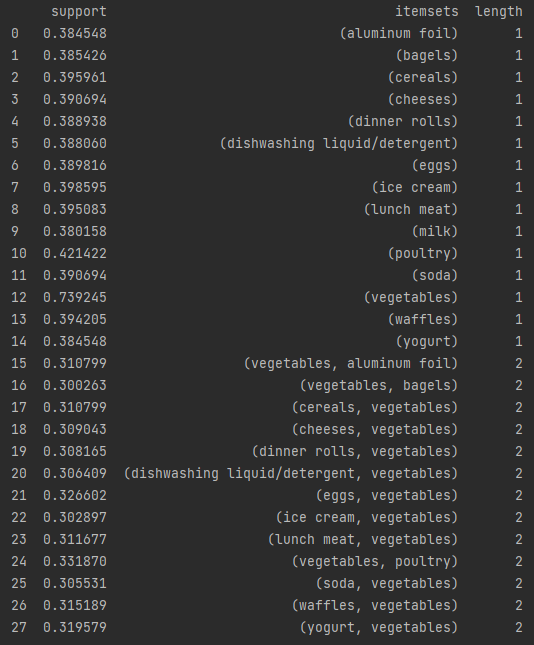
Для набора размером 3 значение = 0.16154521510096576

Для набора размером 4 значение = 0.08516242317822652

* 1. Построен датасет, в котором оставлены только те элементы, которые попадают в наборы размеров 1 при уровне поддержки 0.38. Полученный датасет приведен к формату, который можно обработать.

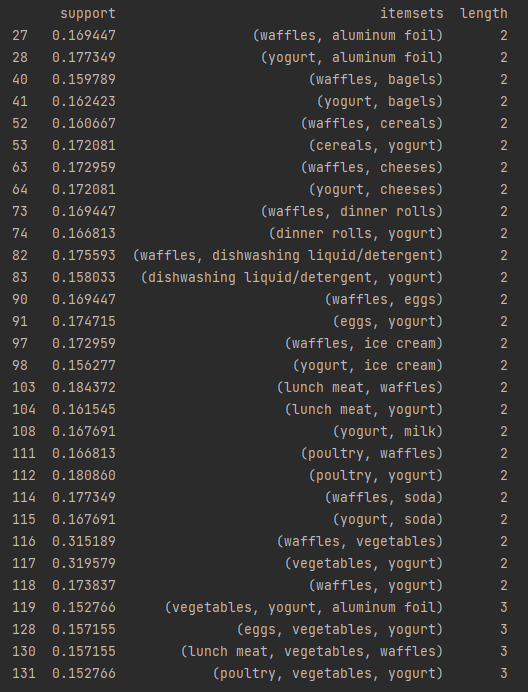


* 1. Проведен ассоциативный анализ при уровне поддержки 0.3 для нового датасета.

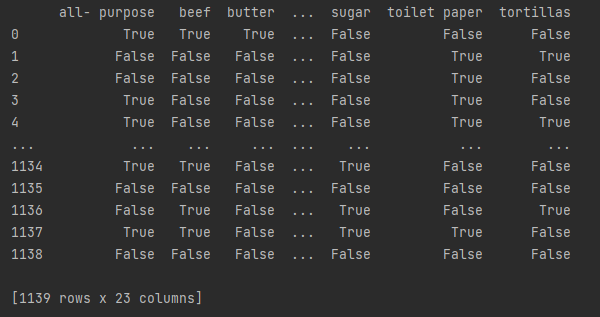


У наборов длины 1 минимальный уровень поддержки стал 0.38.

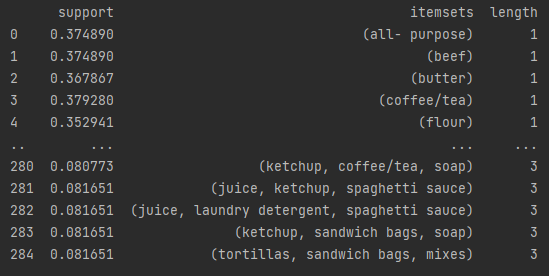
* 1. Проведен ассоциативный анализ при уровне поддержки 0.15 для нового датасета. Выведены все наборы, размер которых больше 1 и в котором есть 'yogurt' или 'waffles'.



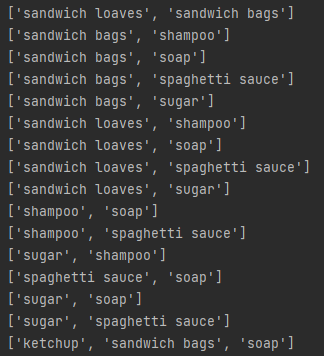
* 1. Составлен ещё один датасет из элементов, не попавших в датасет п. 3.3 задания.



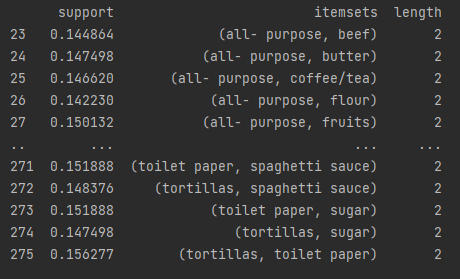
* 1. Проведен анализ aprioti для полученного датасета.



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



* 1. Выбраны все наборы, для которых уровень поддержки лежит в промежутке [0.1, 0.25].



1. **Выводы**

Ознакомились с методами частотного анализа из библиотеки MLxtend. При большем уровне поддержки уменьшается количество наборов. При этом сначала перестают генерироваться наборы большего размера.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

import pandas as pd  
import numpy as np  
from mlxtend.preprocessing import TransactionEncoder  
from mlxtend.frequent\_patterns import apriori  
import matplotlib.pyplot as plt  
  
all\_data = pd.read\_csv('dataset\_group.csv', header=None)  
unique\_id = list(set(all\_data[1]))  
print(len(unique\_id))  
items = list(set(all\_data[2]))  
print(len(items))  
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\_)  
print(df)  
  
results = apriori(df, min\_support=0.3, use\_colnames=True)  
results['length'] = results['itemsets'].apply(lambda x: len(x))  
print(results)  
  
results = apriori(df, min\_support=0.3, use\_colnames=True, max\_len=1)  
print(results)  
  
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), '\n')  
res = []  
for minSup in range(5, 100):  
 results = apriori(df, min\_support=minSup/100, use\_colnames=True)  
 res.append(len(results))  
plt.plot([i/100 for i in range(5, 100)], res)  
results = apriori(df, min\_support=0.05, use\_colnames=True)  
results['length'] = results['itemsets'].apply(lambda x: len(x))  
for i in range(1, 5):  
 res = results[results['length'] == i]  
 print(max(res['support']))  
 plt.vlines(max(res['support']), 0, 16000, color='r')  
plt.show()  
  
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)  
new\_df = pd.DataFrame(te\_ary, columns=te.columns\_)  
print(new\_df)  
  
results = apriori(new\_df, min\_support=0.3, use\_colnames=True)  
results['length'] = results['itemsets'].apply(lambda x: len(x))  
print(results)  
  
results = apriori(new\_df, min\_support=0.15, use\_colnames=True)  
results['length'] = results['itemsets'].apply(lambda x: len(x))  
results = results[results['length'] > 1]  
print(results[results['itemsets'].apply(lambda x: ('yogurt' in x) or ('waffles' in x))])  
  
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 not in new\_items\_] for id in unique\_id]  
te = TransactionEncoder()  
te\_ary = te.fit(new\_dataset\_).transform(new\_dataset\_)  
new\_df\_ = pd.DataFrame(te\_ary, columns=te.columns\_)  
print(new\_df\_)  
  
  
results = apriori(new\_df\_, min\_support=0.08, use\_colnames=True)  
results['length'] = results['itemsets'].apply(lambda x: len(x))  
print(results)  
  
y = results['itemsets'].apply(lambda x: list(x))  
for i in y:  
 count = 0  
 for j in i:  
 if j[0] == 's':  
 count += 1  
  
 if count > 1:  
 print(i)  
  
  
results = results[results['support'] <= 0.25]  
print(results[results['support'] >= 0.1])