|  |  |  |
| --- | --- | --- |
|  | |  |
| МИНОБРНАУКИ РОССИИ | |  |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** | |  |
|  | Институт информационных технологий (ИТ) | |
|  | Кафедра прикладной математики | |

|  |  |
| --- | --- |
| **Отчет по практической работе №8.** | |
| **по дисциплине** | |
| **«Технологии и инструментарий анализа больших данных»** | |
|  | |
| Выполнил студент группы ИКБО- 22-20 | Гологузов К.А. |
| Принял | Парамонов А.А. |

Москва 2023

Решение

1. Загрузить данные Market\_Basket\_Optimisation.csv.

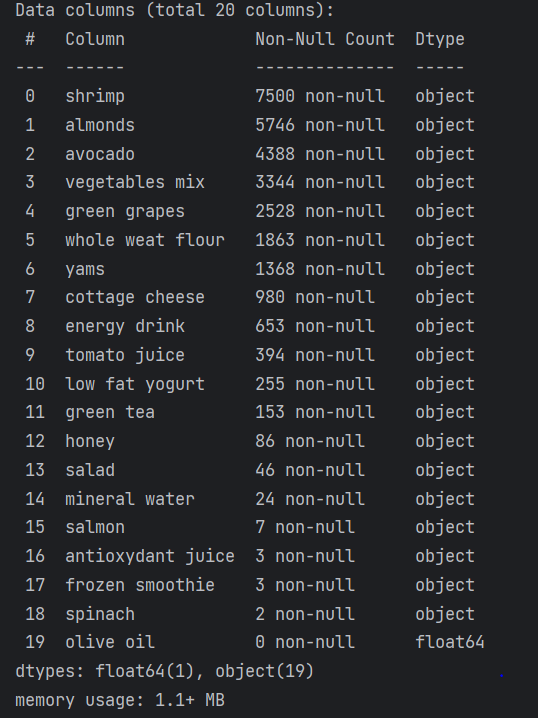


Рис.1 – Загруженные даные

2. Визуализировать данные (отразить на гистограммах относительную и фактическую частоту встречаемости для 20 наиболее популярных товаров).

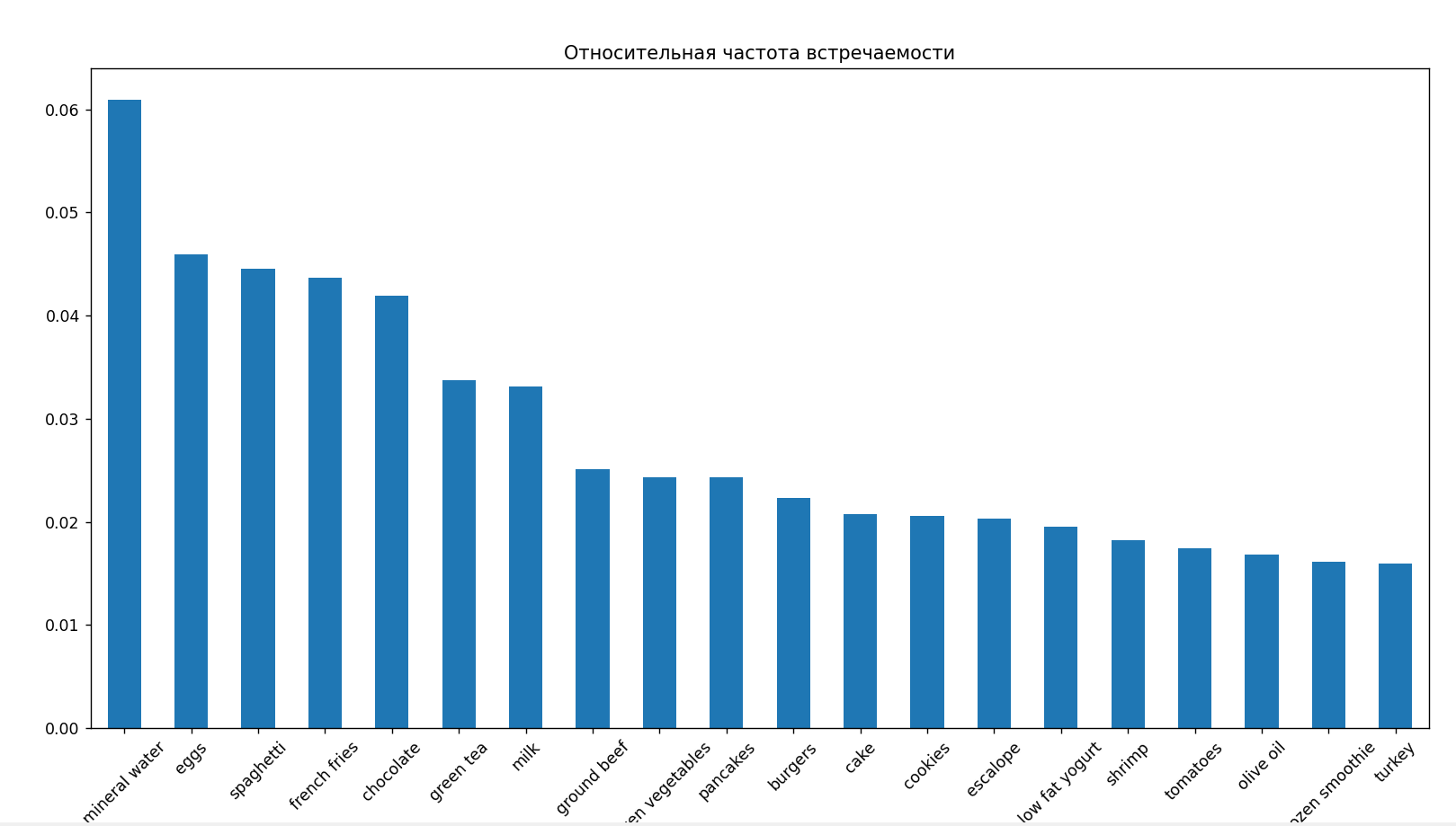


Рис.2 – Относительная частота встречаемости

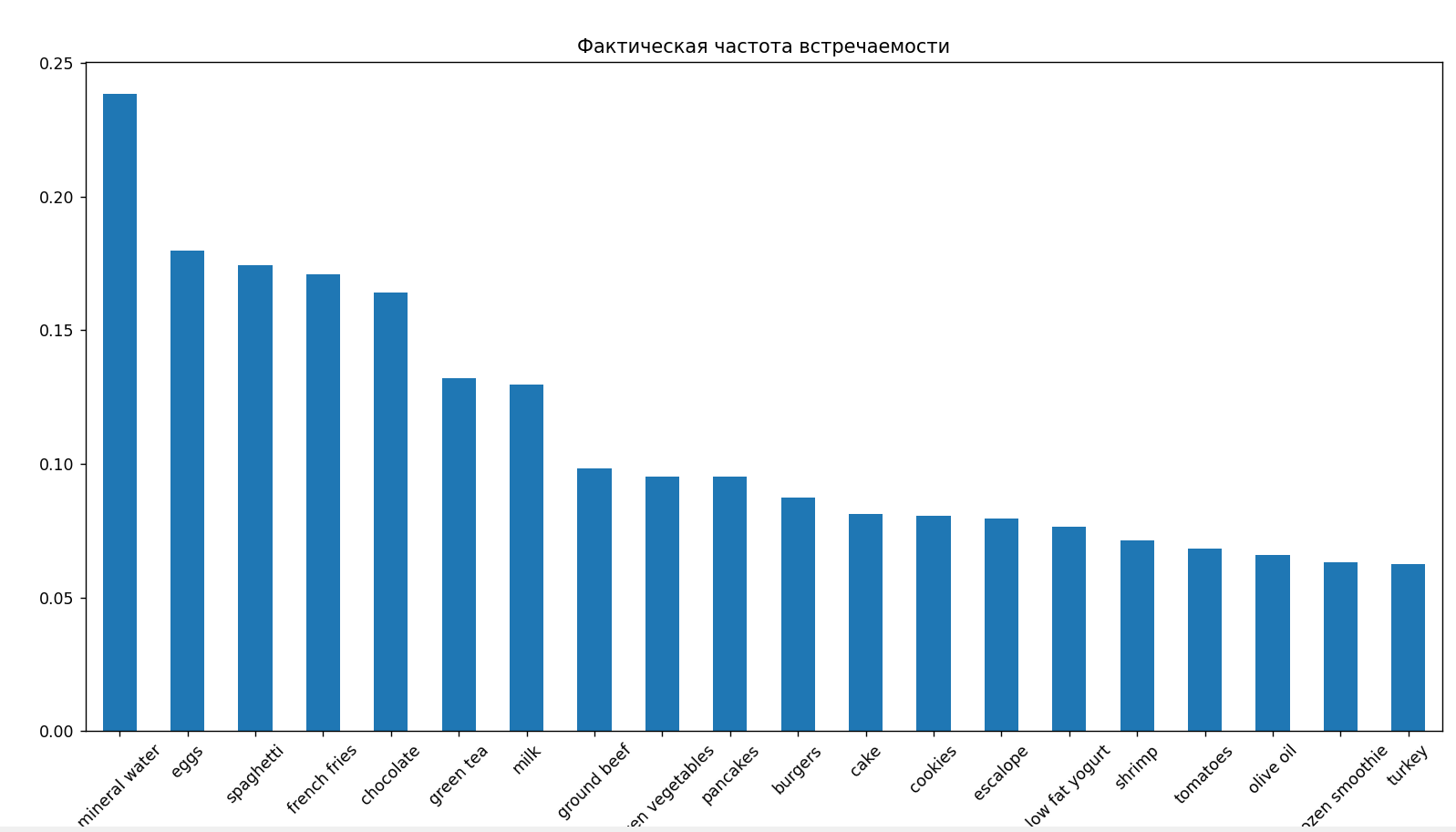
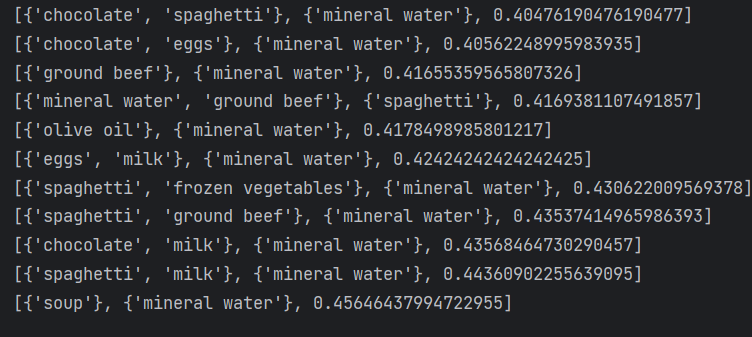


Рис.3 – Фактическая частота встречаемости

3. Применить алгоритм Apriori, используя 3 разные библиотеки (apriori\_python, apyori, efficient\_apriori). Подобрать гиперпараметры для алгоритмов так, чтобы выводилось порядка 10 наилучших правил.

Рис.4 – Результат библиотеки apriori\_python

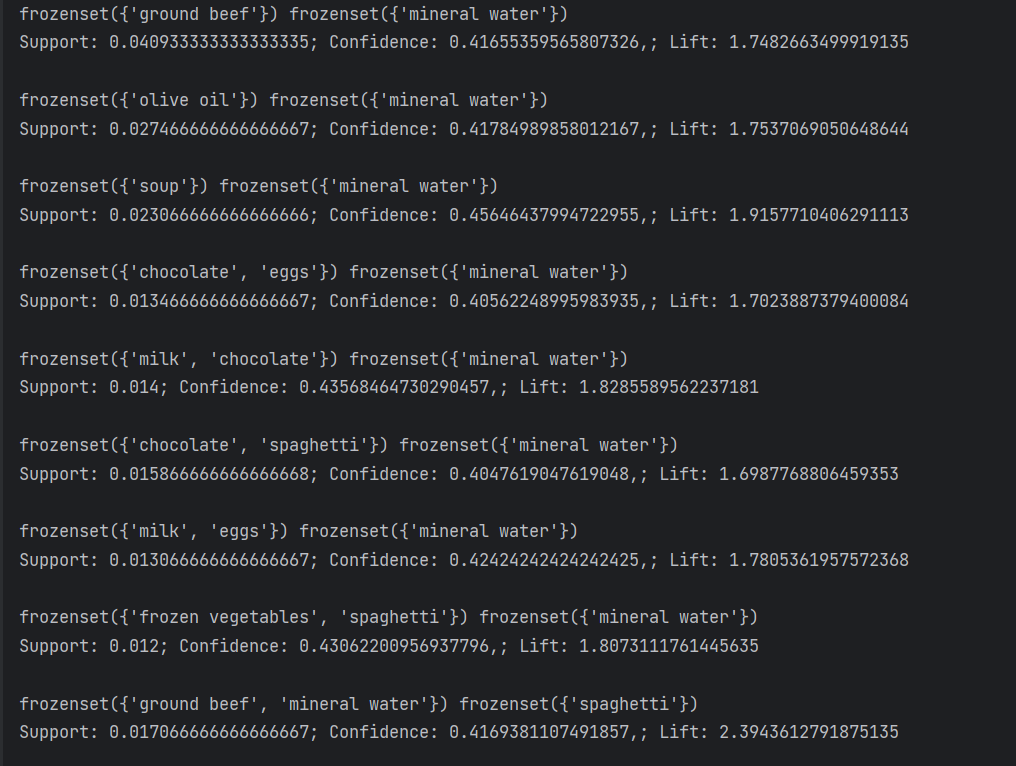


Рис.5 – Результат библиотеки apyori

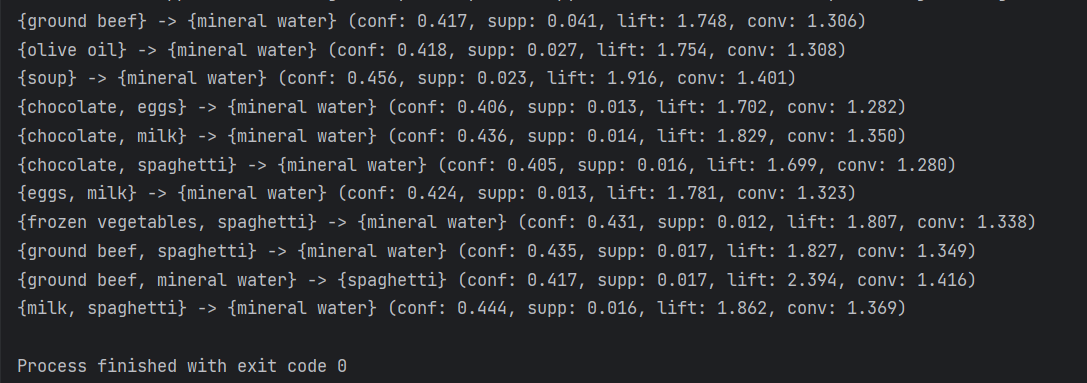


Рис.6 – Результат библиотеки efficient\_apriori

4. Применить алгоритм FP-Growth из библиотеки fpgrowth\_py. Подобрать гиперпараметры для алгоритма так, чтобы выводилось порядка 10 наилучших правил.

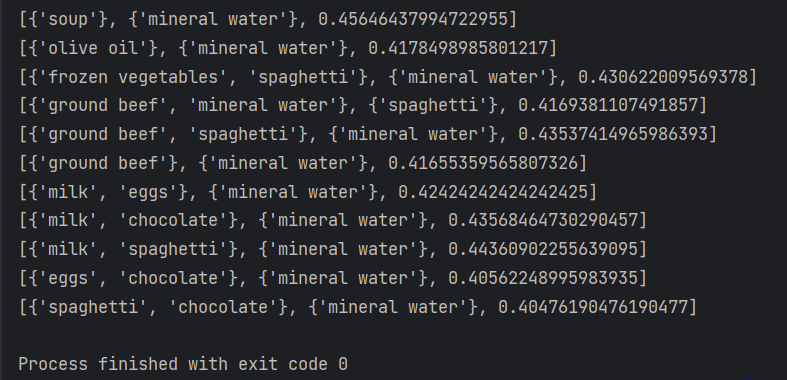


Рис. 7 – Результат библиотеки fpgrowth\_py

5. Сравнить время выполнения всех алгоритмов и построить гистограмму.

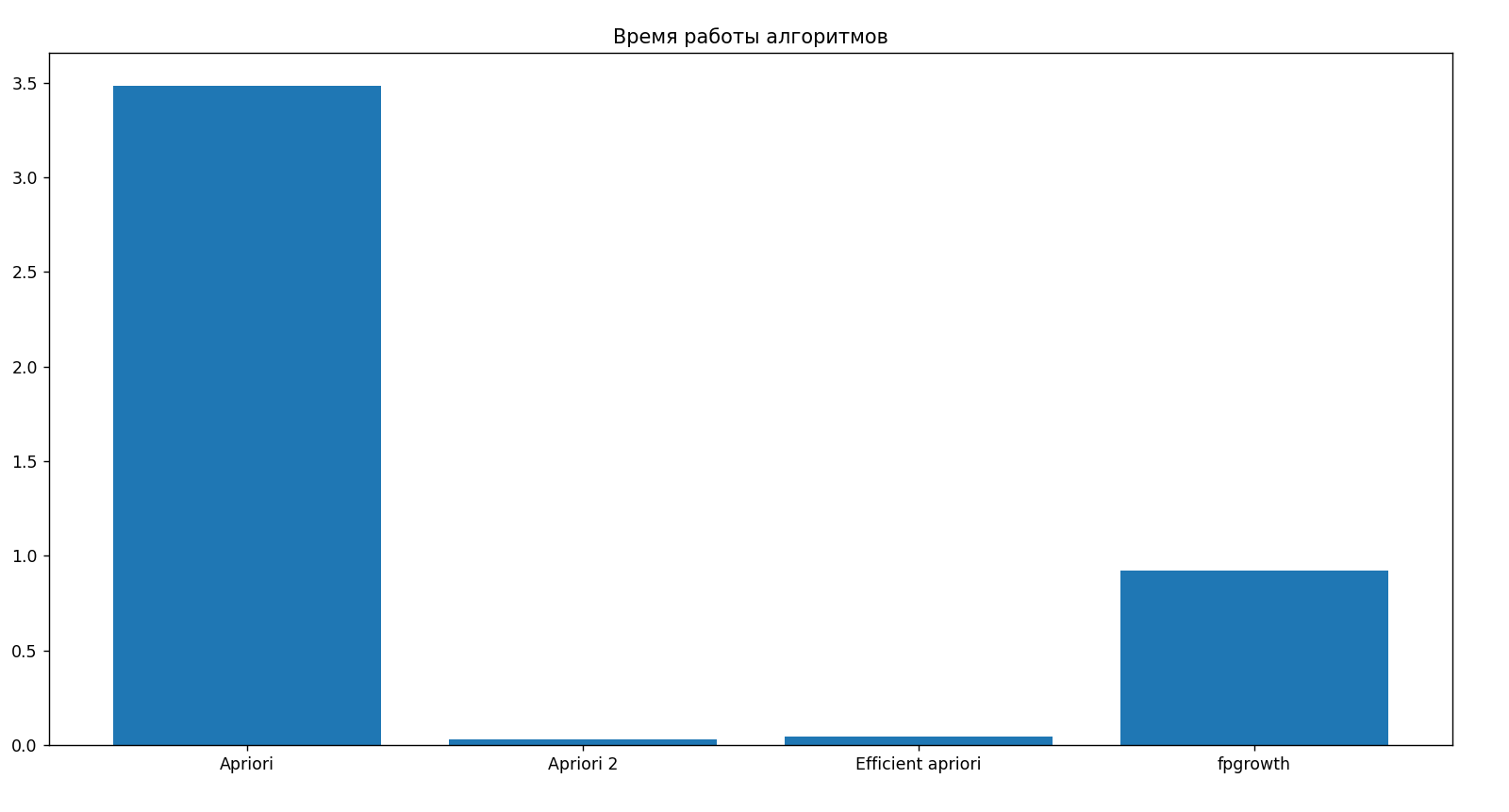


Рис.8 – Сравнение времени работы алгоритмов

6. Загрузить данные data.csv.

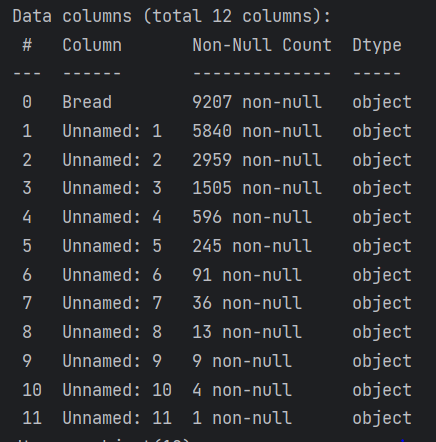


Рис.9 – data.csv

7. Визуализировать данные (отразить на гистограммах относительную и фактическую частоту встречаемости для 20 наиболее популярных товаров).

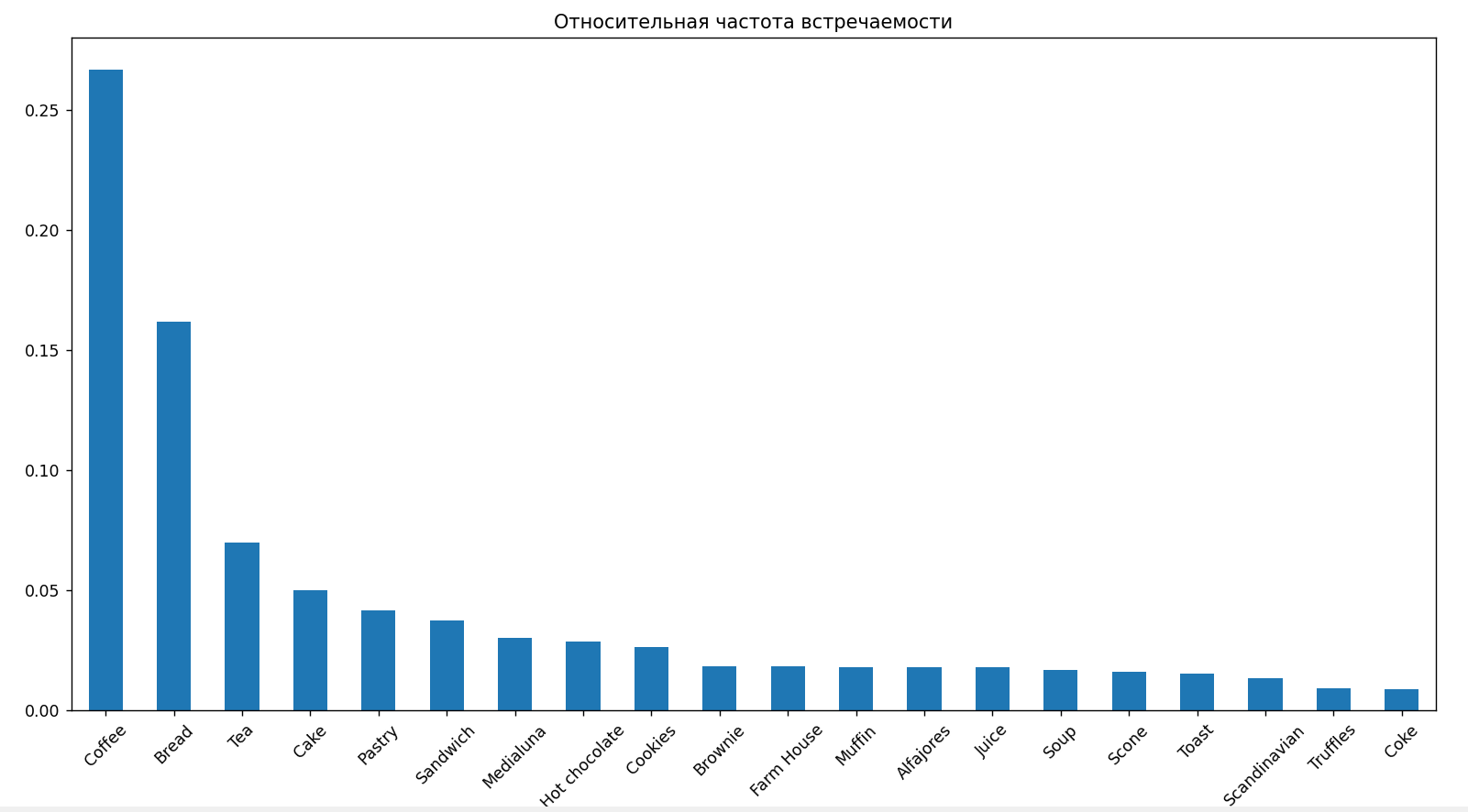


Рис.10 – Относительная частота встречаемости

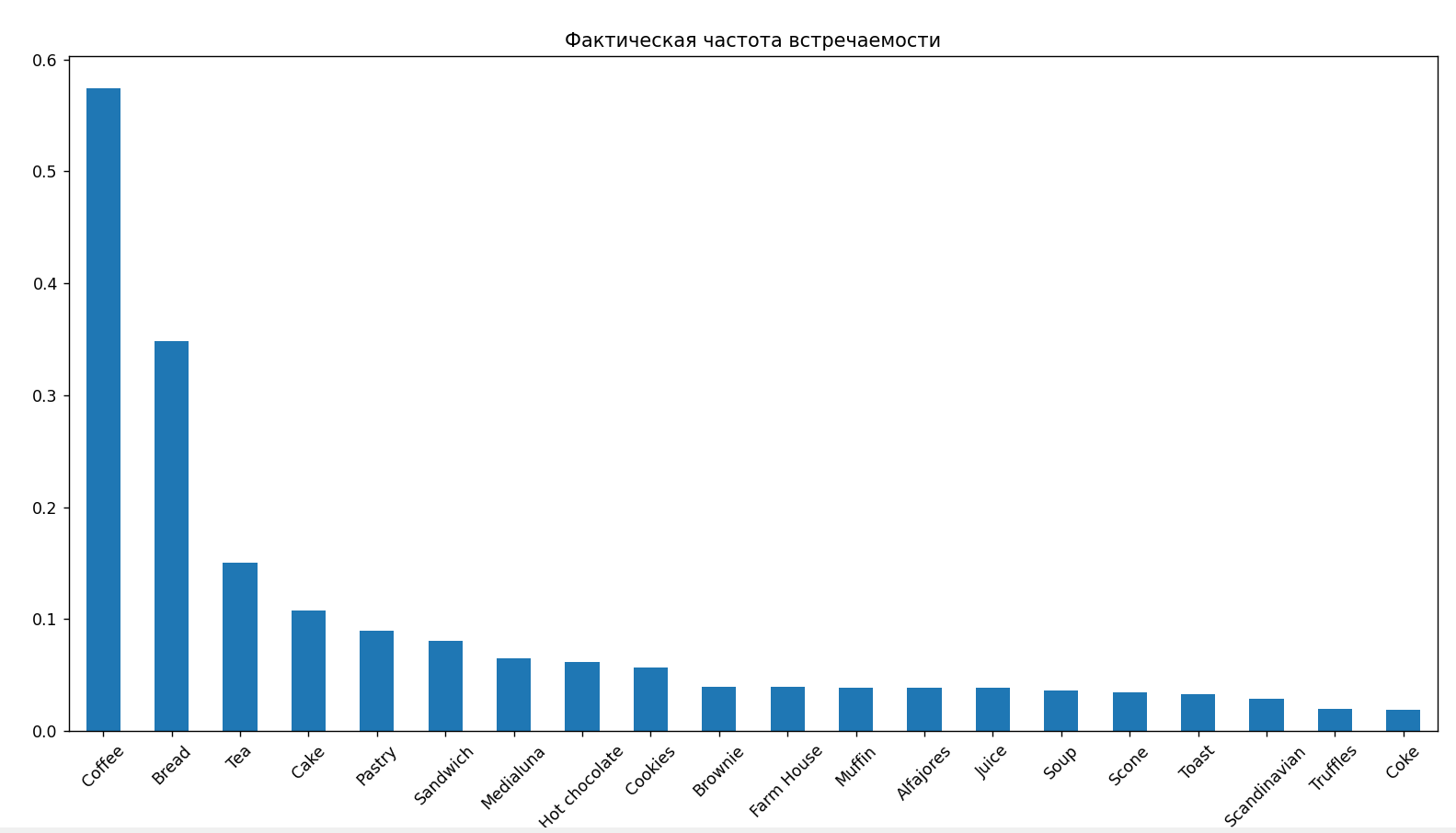


Рис.11 – Фактическая частота встречаемости

8. Применить алгоритм Apriori, используя 3 разные библиотеки (apriori\_python, apyori, efficient\_apriori). Подобрать гиперпараметры для алгоритмов так, чтобы выводилось порядка 10 наилучших правил.

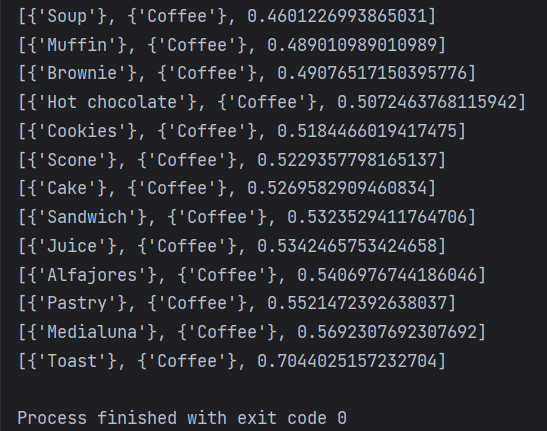


Рис.12 – Результат библиотеки apriori\_python

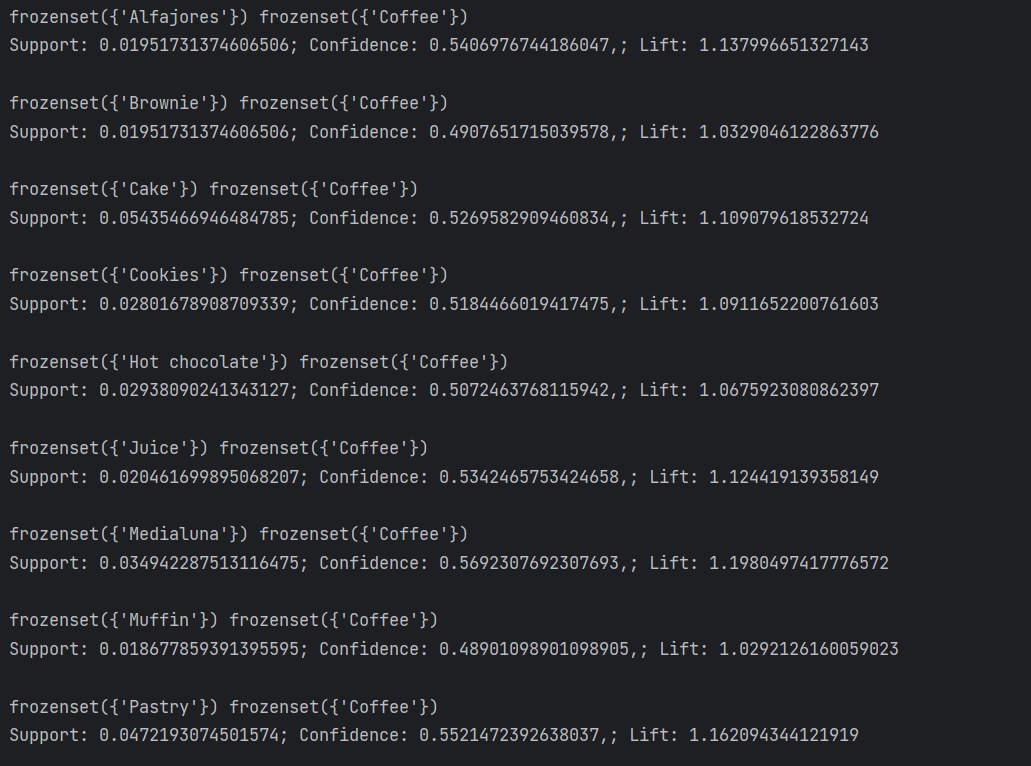


Рис.13 – Результат библиотеки apyori

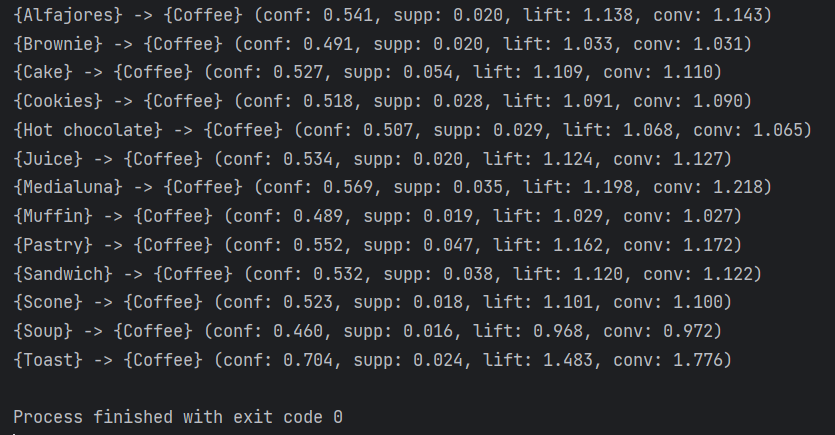


Рис.14 – Результат библиотеки efficient\_apriori

9. Применить алгоритм FP-Growth из библиотеки fpgrowth\_py. Подобрать гиперпараметры для алгоритма так, чтобы выводилось порядка 10 наилучших правил.

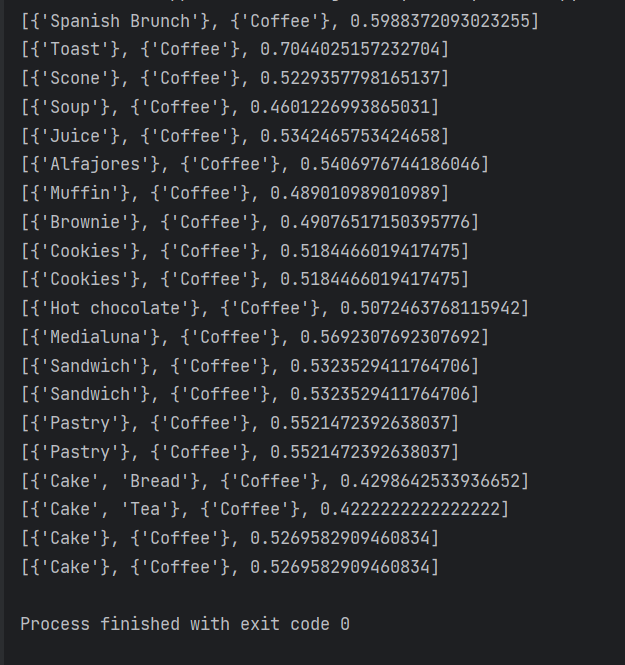


Рис.15 – Результат работы fpgrowth\_py

10.Сравнить время выполнения всех алгоритмов и построить гистограмму.

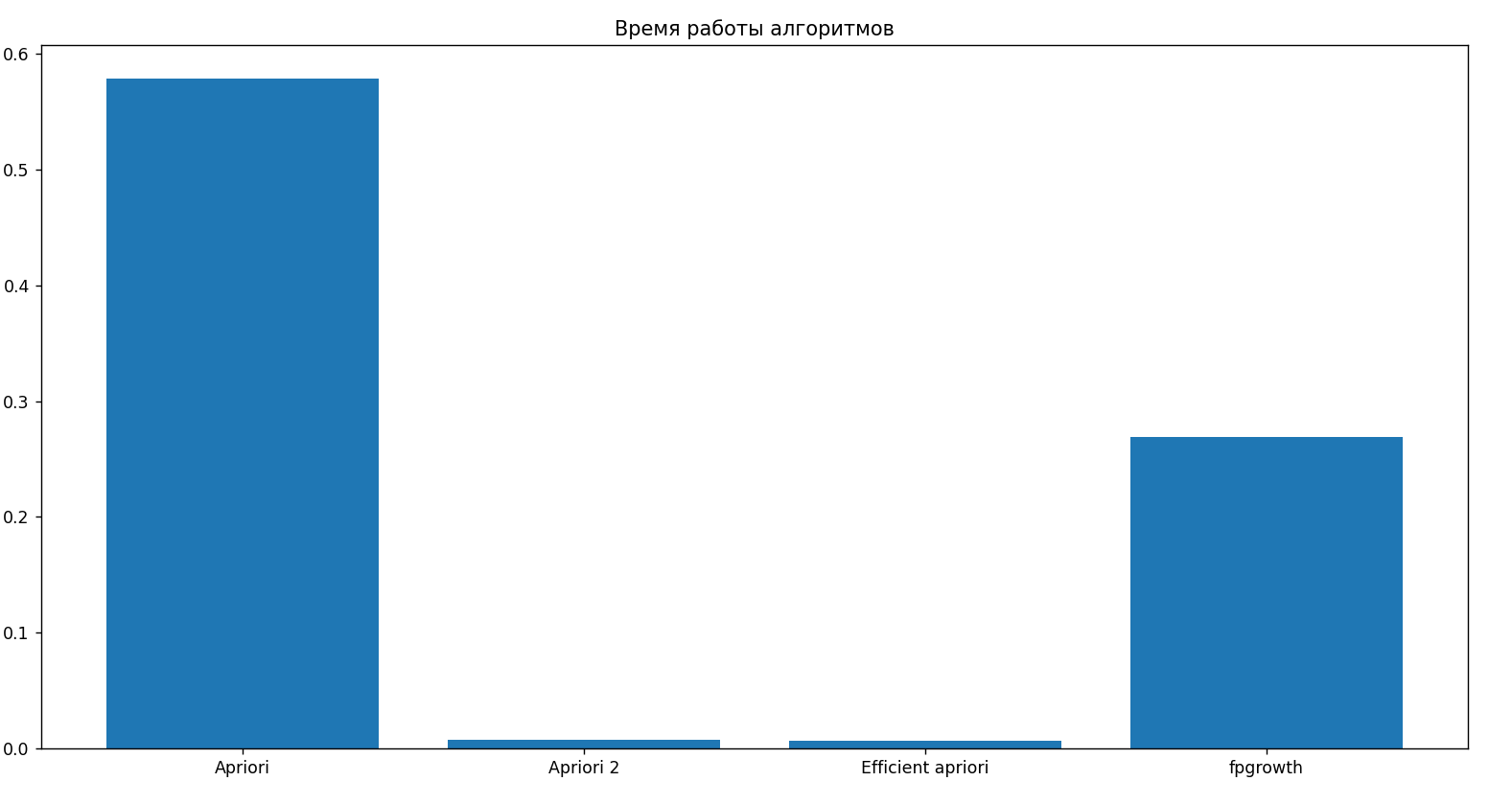


Рис. 16 – Время работы алгоритмов

11.Сформулировать выводы и сделать отчет.

Благодаря проделанной работе мы можем сделать вывод, что библиотеки apyori и efficient\_apriori работают значительно быстрее аналогов.

Для поиска ассоциативных правил обоих датасетов были использованы следующие гиперпараметры:

Минимальная поддержка – 0.012

Минимальная достоверность – 0.4

ПРИЛОЖЕНИЕ

Приложение 1 – 1.py

import pandas as pd

import matplotlib.pyplot as plt

from apriori\_python import apriori as a1

from apyori import apriori as a2

from efficient\_apriori import apriori

from fpgrowth\_py import fpgrowth

import time

data = pd.read\_csv("data.csv")

# 1

# print(data.info())

# 2

# print(data.stack().value\_counts())

# print(data.stack().value\_counts(normalize=True))

# data.stack().value\_counts(normalize=True)[0:20].plot(kind="bar")

# plt.title("Относительная частота встречаемости")

# data.stack().value\_counts().apply(lambda item: item / data.shape[0])[0:20].plot(kind="bar")

# plt.title("Фактическая частота встречаемости")

# plt.xticks(rotation=45)

# plt.show()

# 3

transactions = []

for i in range(data.shape[0]):

row = data.iloc[i].dropna().tolist()

transactions.append(row)

t = []

# 3.1

start = time.perf\_counter()

t1, rules = a1(transactions, minSup=0.012, minConf= 0.4) # s=0.012, c=0.4

time1 = time.perf\_counter()-start

t.append(time1)

# for rule in rules:

# print(rule)

# 3.2

start = time.perf\_counter()

rules = a2(transactions=transactions,

min\_support=0.012,

min\_confidence=0.4,

min\_lift=1.0001)

results = list(rules)

time2 = time.perf\_counter() - start

t.append(time2)

# for r in results:

# for subset in r[2]:

# print(subset[0], subset[1])

# print("Support: {0}; Confidence: {1},; Lift: {2}".format(r[1], subset[2], subset[3]))

# print()

# 3.3

start = time.perf\_counter()

itemsets, rules = apriori(transactions, min\_support=0.012, min\_confidence=0.4)

time3 = time.perf\_counter() – start

Продолжение – Приложение 1

t.append(time3)

# for i in range(len(rules)):

# print(rules[i])

# 4

start = time.perf\_counter()

itemsets, rules = fpgrowth(transactions, minSupRatio=0.012, minConf=0.4)

time4 = time.perf\_counter() - start

t.append(time4)

#

# for i in range(len(rules)):

# print(rules[i])

# 5

print("Apriori", t[0], "\n")

print("Apriori 2", t[1], "\n")

print("Efficient apriori", t[2], "\n")

print("fpgrowth", t[3], "\n")

plt.bar(["Apriori", "Apriori 2", "Efficient apriori", "fpgrowth"], t)

plt.title("Время работы алгоритмов")

plt.show()