|  |  |  |
| --- | --- | --- |
|  | |  |
| 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 | Никулин К.В. |
| Принял | Парамонов А.А. |

Москва 2023

Решение

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

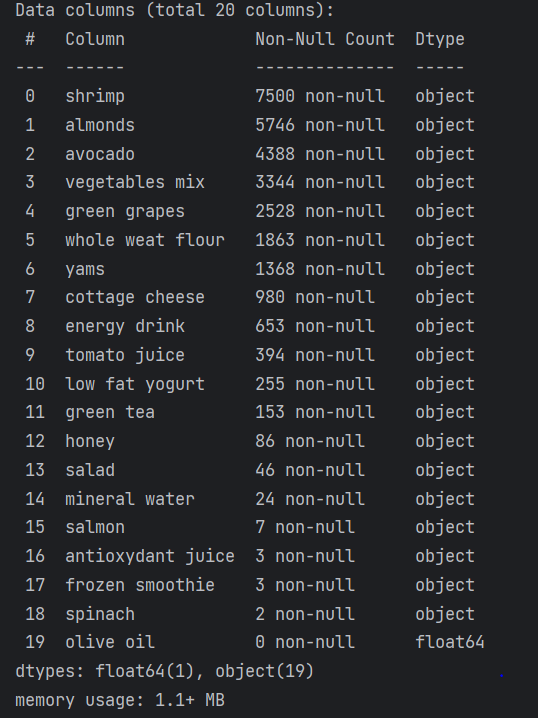


Рис.1 – Загруженные данные

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

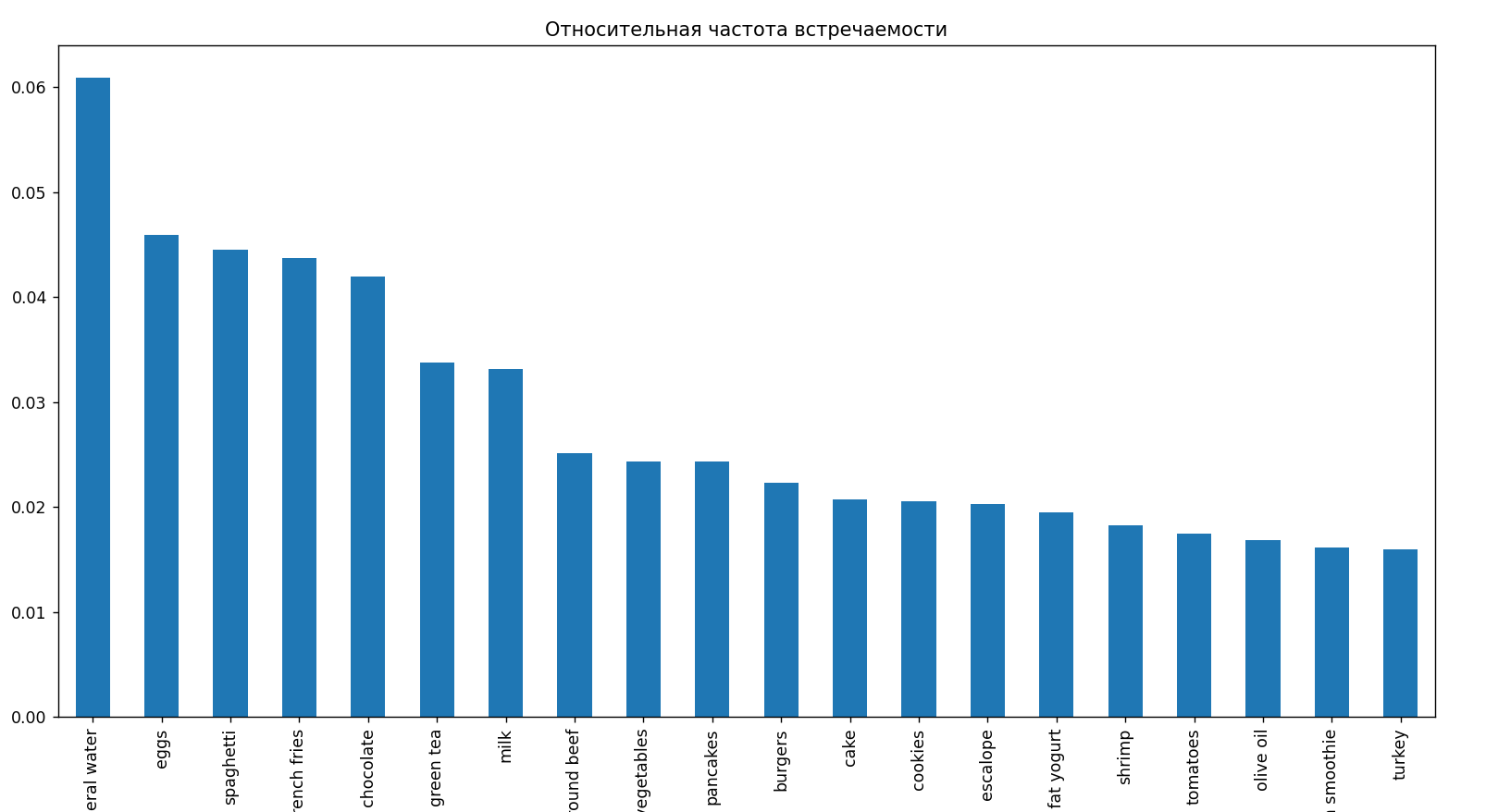


Рис.2 – Относительная частота встречаемости

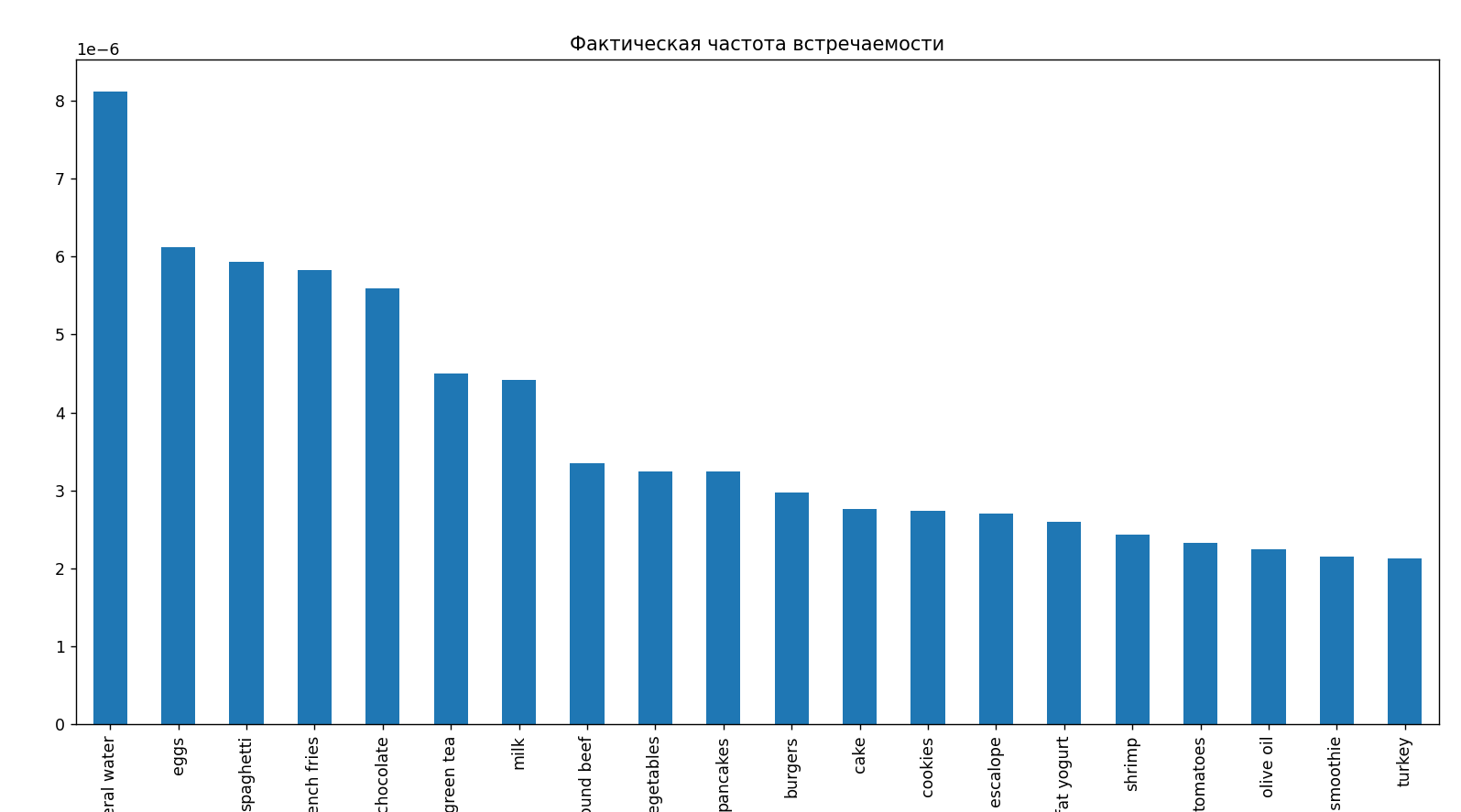
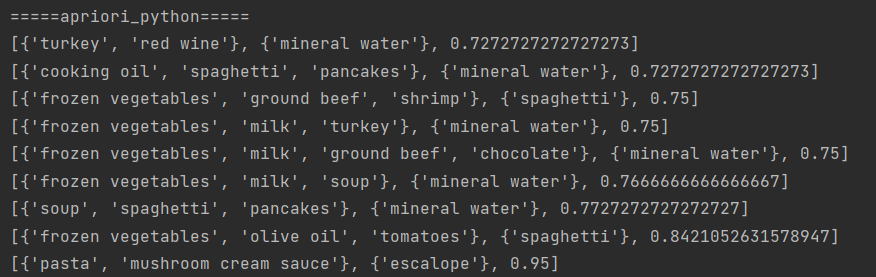


Рис.3 – Фактическая частота встречаемости

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

Для всех алгоритмов были выбраны гиперпараметры (Support=0.002, Confidence=0.72)

Рис.4 – Результат библиотеки apriori\_python

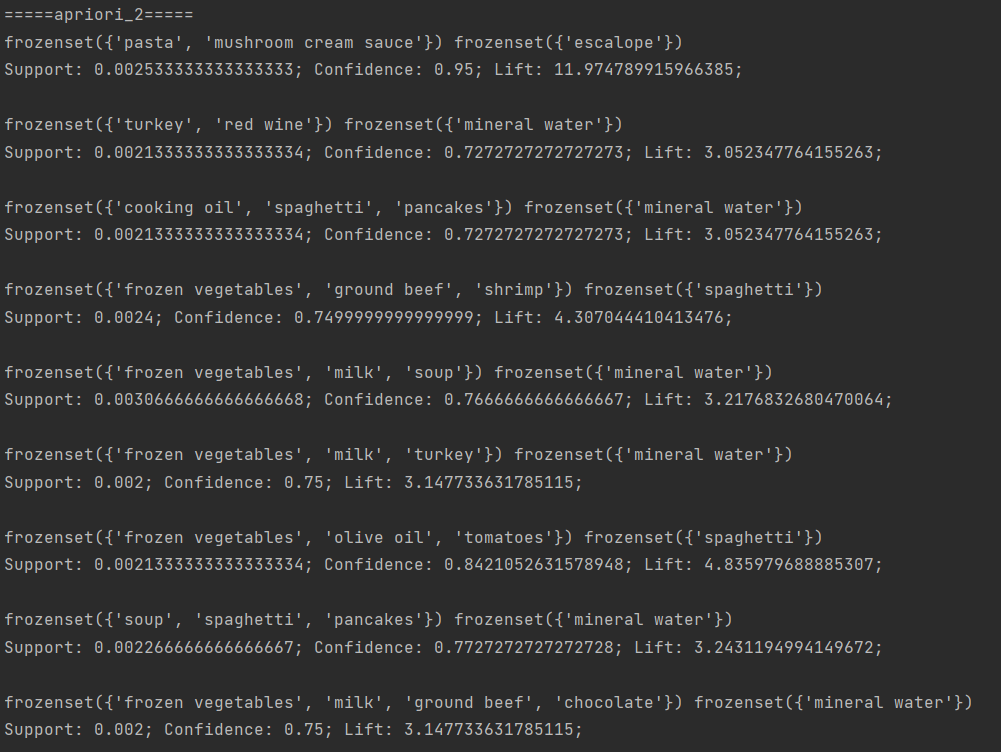


Рис.5 – Результат библиотеки apyori

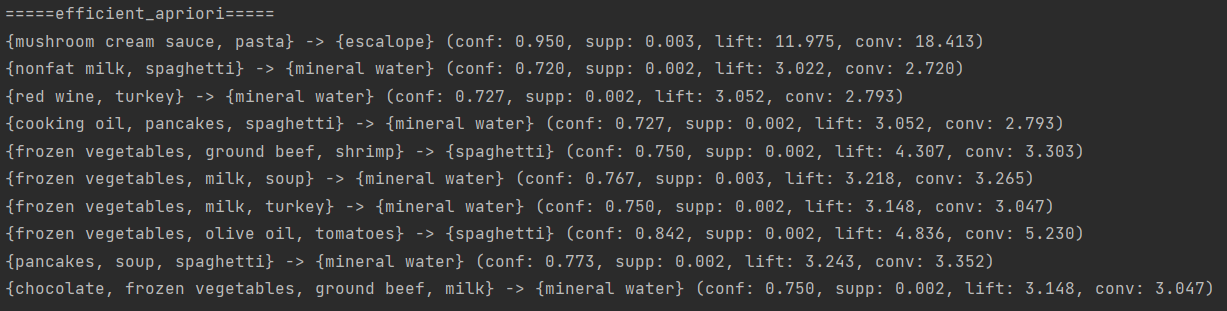


Рис.6 – Результат библиотеки efficient\_apriori

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

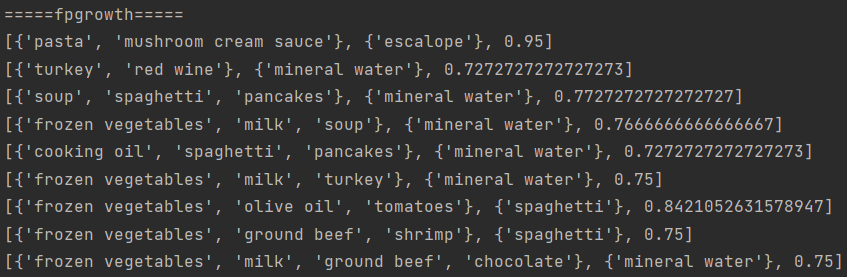


Рис. 7 – Результат библиотеки fpgrowth\_py

5. Сравнить время выполнения всех алгоритмов и построить гистограмму.

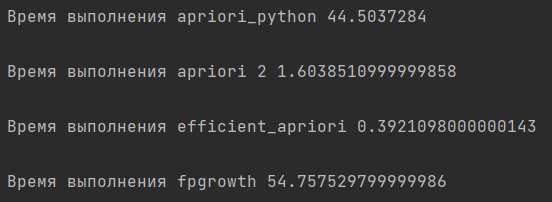


Рис.8 – Сравнение времени работы алгоритмов

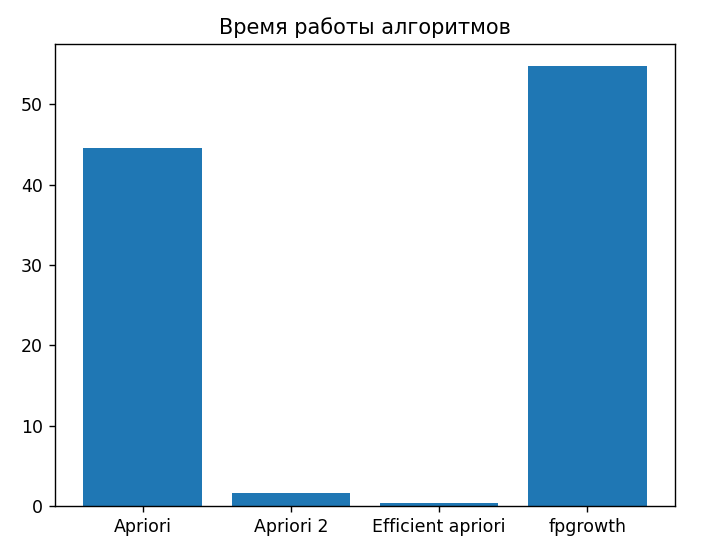


Рис.9 – Гистограмма сравнения времени работы алгоритмов

6. Загрузить данные data.csv.

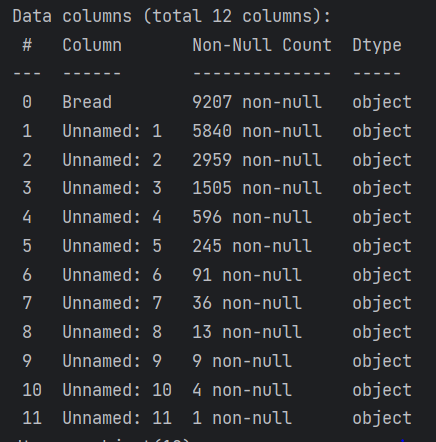


Рис.9 – data.csv

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

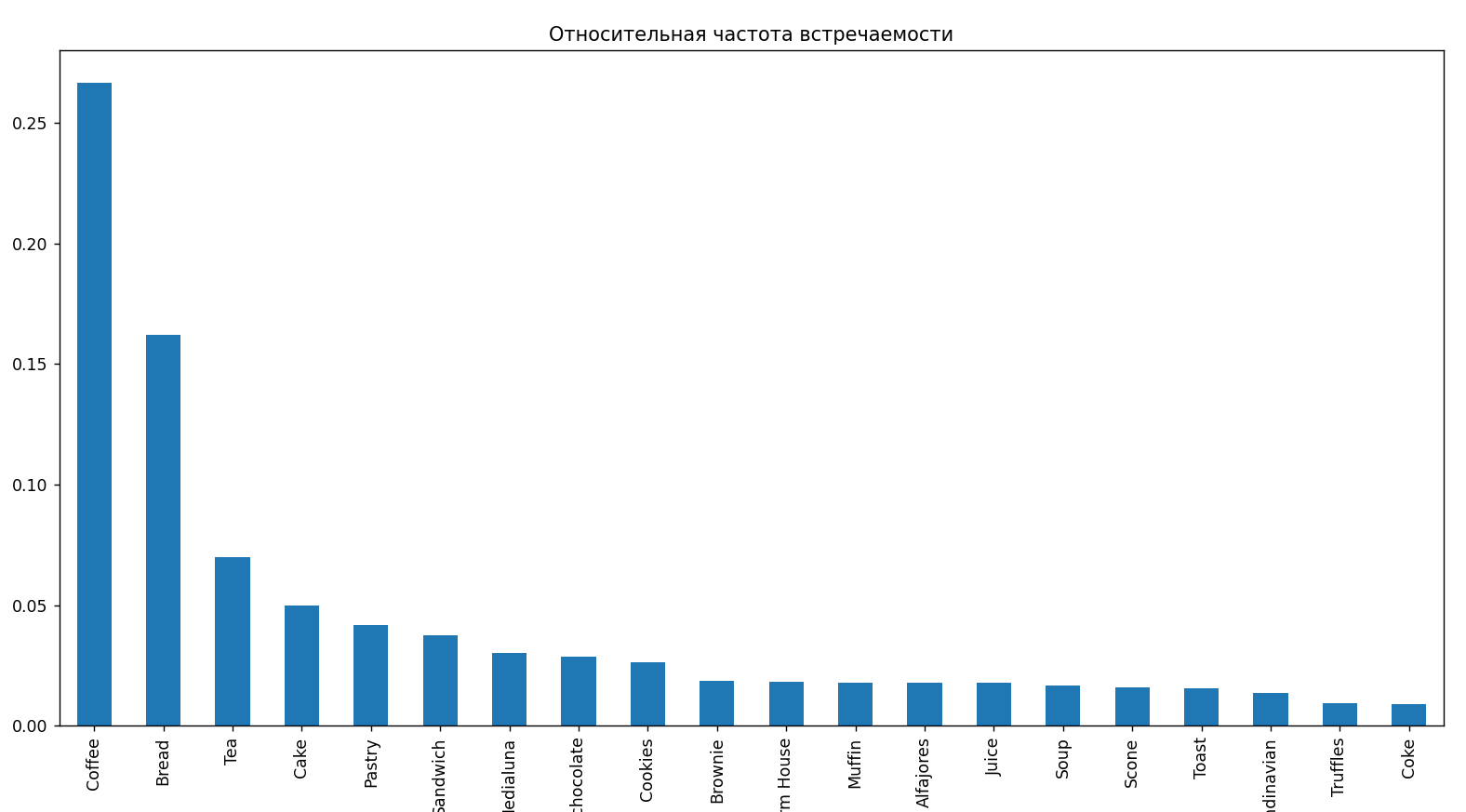


Рис.10 – Относительная частота встречаемости

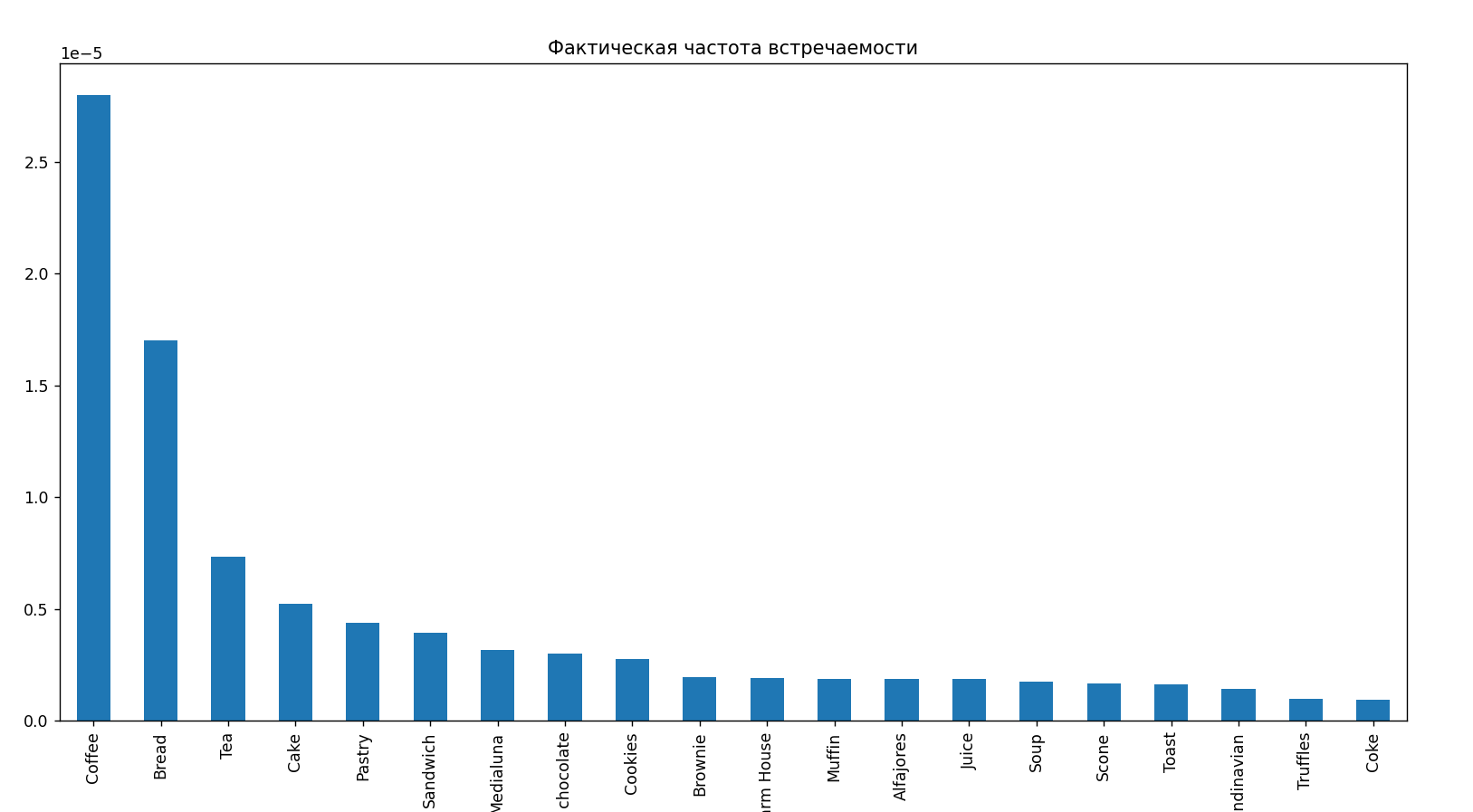


Рис.11 – Фактическая частота встречаемости

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

Для всех алгоритмов были выбраны гиперпараметры (Support=0.015, Confidence=0.5)

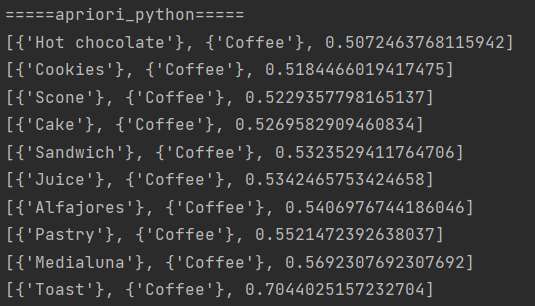


Рис.12 – Результат библиотеки apriori\_python

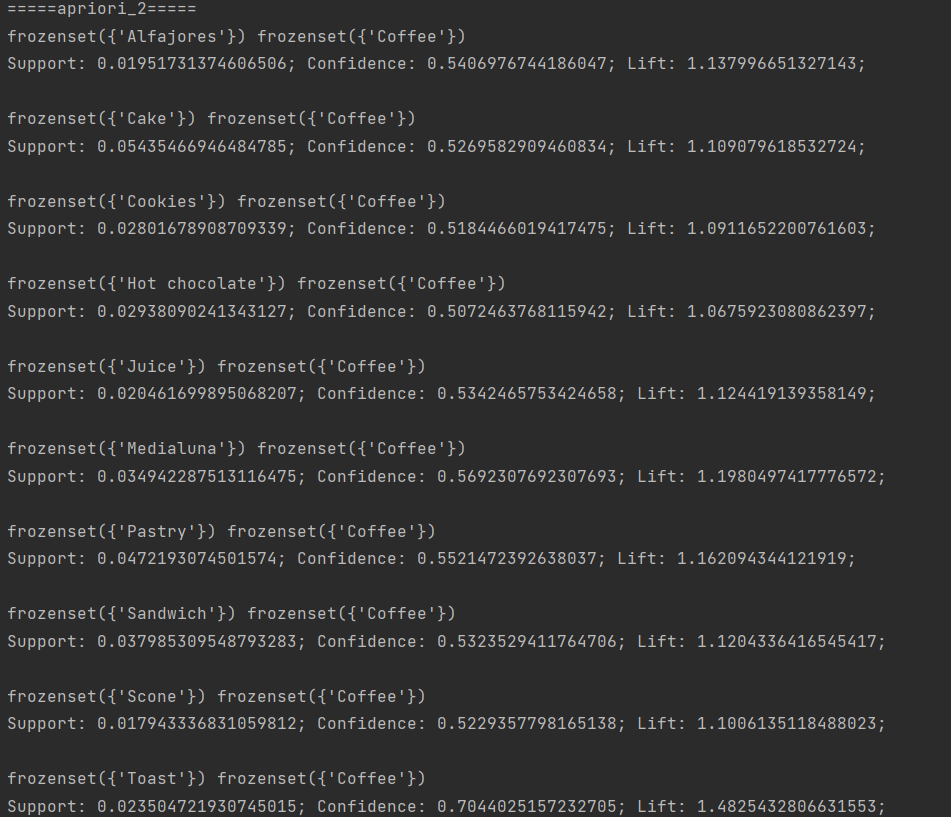


Рис.13 – Результат библиотеки apyori

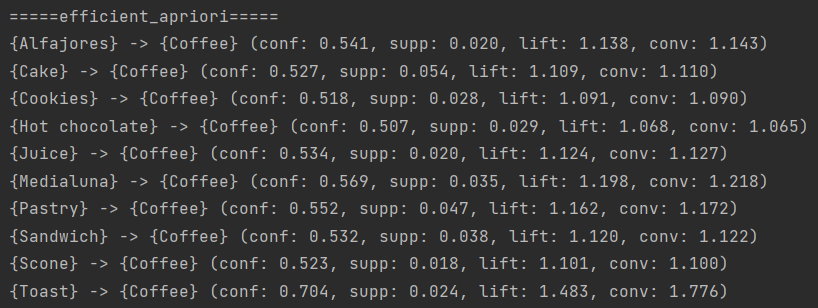


Рис.14 – Результат библиотеки efficient\_apriori

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

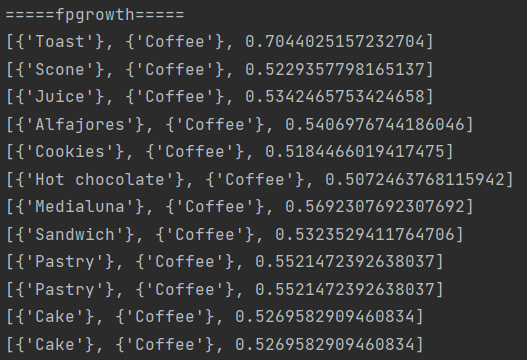


Рис.15 – Результат работы fpgrowth\_py

10.Сравнить время выполнения всех алгоритмов и построить гистограмму.

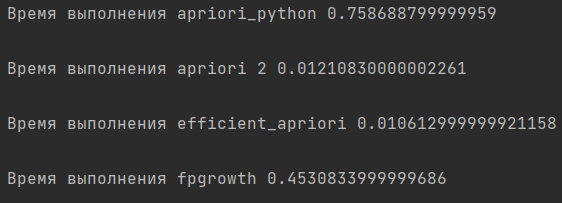


Рис. 16 – Сравнение времени работы алгоритмов

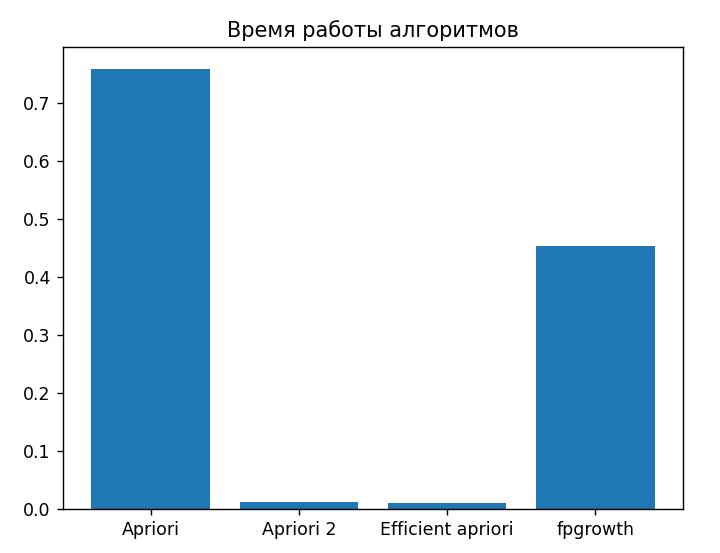


Рис. 17 – Гистограмма сравнения времени работы алгоритмов

11.Сформулировать выводы и сделать отчет.

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

ПРИЛОЖЕНИЕ

Приложение 1 – pr8.py

import time  
import pandas as pd  
import matplotlib.pyplot as plt  
from apriori\_python import apriori as apr1  
from apyori import apriori as apr2  
from efficient\_apriori import apriori as apr3  
from fpgrowth\_py import fpgrowth  
  
  
def apriori1(df):  
 transactions = []  
 for i in range(df.shape[0]):  
 row = df.iloc[i].dropna().tolist()  
 transactions.append(row)  
 t = []  
 start = time.perf\_counter()  
 t1, rule = apr1(transactions, minSup=0.015, minConf=0.5)  
 time1 = time.perf\_counter()-start  
 t.append(time1)  
 print('=====apriori\_python=====')  
 for item in rule:  
 print(item)  
  
 start = time.perf\_counter()  
 rules = apr2(transactions=transactions, min\_support=0.015, min\_confidence=0.5, min\_lift=1.0001)  
 results = list(rules)  
 time2 = (time.perf\_counter()-start)  
 t.append(time2)  
 print('=====apriori\_2=====')  
 for result in results:  
 for subset in result[2]:  
 print(subset[0], subset[1])  
 print("Support: {0}; Confidence: {1}; Lift: {2};".format(result[1], subset[2], subset[3]))  
 print()  
  
 start = time.perf\_counter()  
 itemsets, rules = apr3(transactions, min\_support=0.015, min\_confidence=0.5)  
 time3 = time.perf\_counter()-start  
 t.append(time3)  
 print('=====efficient\_apriori=====')  
 for i in range(len(rules)):  
 print(rules[i])  
  
 start = time.perf\_counter()  
 itemsets, rules = fpgrowth(transactions, minSupRatio=0.015, minConf=0.5)  
 time4 = time.perf\_counter()-start  
 t.append(time4)  
 print('=====fpgrowth=====')  
 for i in range(len(rules)):  
 print(rules[i])  
  
 print("Время выполнения apriori\_python", t[0], "\n")  
 print("Время выполнения apriori 2", t[1], "\n")  
 print("Время выполнения efficient\_apriori", t[2], "\n")  
 print("Время выполнения fpgrowth", t[3], "\n")  
 plt.bar(["Apriori", "Apriori 2", "Efficient apriori", "fpgrowth"], t)  
 plt.title("Время работы алгоритмов")  
 plt.show()  
 return  
  
  
def main():  
 df = pd.read\_csv("data.csv")  
 print(df.info())  
 df.stack().value\_counts(normalize=True)[0:20].plot(kind='bar', title='Относительная частота встречаемости')  
 plt.show()  
 df.stack().value\_counts(normalize=True).apply(lambda item: item / df.shape[0])[0:20].plot(kind='bar'  
 ,title='Фактическая частота встречаемости')  
 plt.show()  
 apriori1(df)  
 return  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()