МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий
Кафедра прикладной математики

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8

«Ассоциативные правила»

по дисциплине

«Технологии и инструментарий анализа больших данных»

Выполнил студент группы ИВБО-03-21 — Лазарев А. В.

Принял преподаватель кафедры прикладной математики — Тетерин Н.Н.

Практическая работа выполнена « __ » _____ 2024 г.

«Зачтено» « __ » _____ 2024 г.

Москва 2024

СОДЕРЖАНИЕ

1 РЕШЕНИЕ ЗАДАЧ

1.1 Задача №1

Решение программы представлено на Рисунке 1.1.

```
[2] url = 'https://raw.githubusercontent.com/InspectorJelly/BigDataMirea/refs/heads/main/datasets/Market_Basket_Optimisation.csv'
    data = pd.read_csv(url)
```

Рисунок 1.1 – Программа

1.2 Задача №2

Решение и результат программы представлены на Рисунке 1.2, 1.3.

```
all_items = data.values.flatten()
all_items = pd.Series(all_items).dropna()
item_counts = all_items.value_counts()
top_20_items = item_counts.head(20)
relative_frequencies = top_20_items / top_20_items.sum()

plt.figure(figsize=(14, 6))
plt.bar(top_20_items.index, top_20_items.values, color='skyblue')
plt.title('Топ-20 товаров по фактической частоте', fontsize=16)
plt.ylabel('Частота', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.tight_layout()
plt.show()

plt.figure(figsize=(14, 6))
plt.bar(relative_frequencies.index, relative_frequencies.values, color='salmon')
plt.title('Топ-20 товаров по относительной частоте', fontsize=16)
plt.ylabel('Относительная частота', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.tight_layout()
plt.show()
```

Рисунок 1.2 – Программа

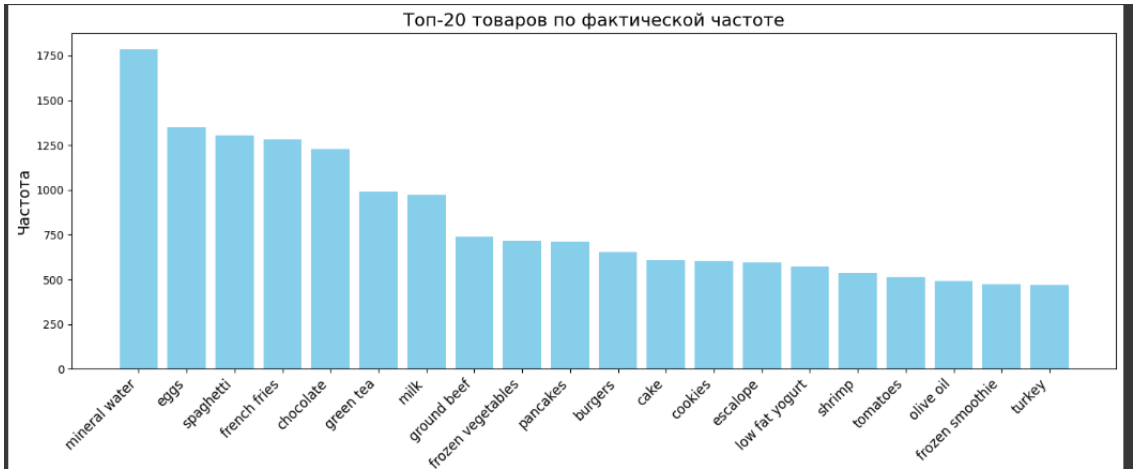

Рисунок 1.3 – Результат работы программы, фактическая частота

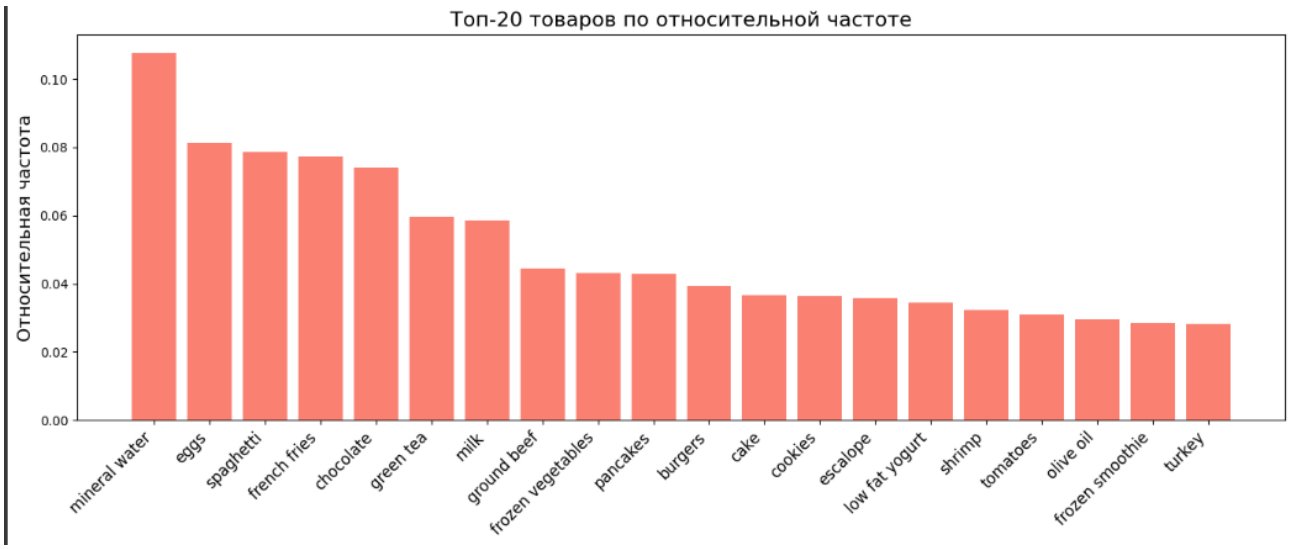

Рисунок 1.4 – Результат выполнения программы, относительная частота

1.3 Задача №3

Решение и результат программы представлены на Рисунке 1.3.

```
transactions = data.apply(lambda row: row.dropna().tolist(), axis=1).tolist()

#apyori
start = time.time()
print("Results with apyori:")
apyori_results = list(apyori_apriori(transactions, min_support=0.005, min_confidence=0.3))
for rule in apyori_results[:10]:
    print(rule)
apyori_time = time.time() - start

#efficient_apriori
start = time.time()
print("\nResults with efficient_apriori:")
itemsets, rules = efficient_apriori(transactions, min_support=0.005, min_confidence=0.3)
for rule in sorted(rules, key=lambda x: x.lift, reverse=True)[:10]:
    print(rule)
efficient_apriori_time = time.time() - start

#apriori_python
start = time.time()
print("\nResults with apriori_python:")
freq_itemsets, rules = apriori_python(transactions, minSup=0.005, minConf=0.3)


# Вывод правил
print("Rules (apriori_python):")
for rule in rules[:10]:
    antecedent, consequent, confidence = rule
    print(f"Rule: {set(antecedent)} -> {set(consequent)}, Confidence: {confidence}")
apriori_python_time = time.time() - start
```

Рисунок 1.5 – Программа

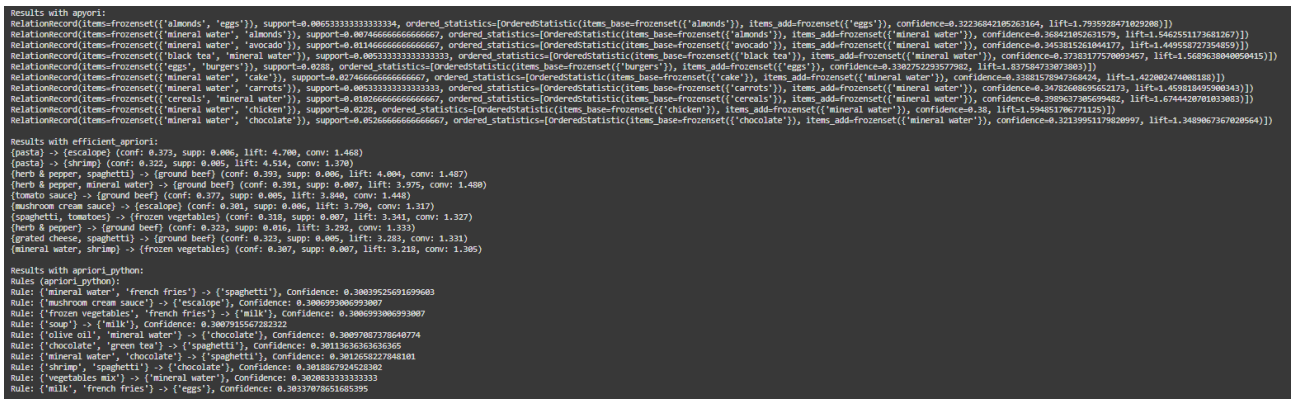

Рисунок 1.6 – Результат выполнения программы

1.4 Задача №4

Решение и результат программы представлены на Рисунках 1.4.

```
[32] start = time.time()
     freqItemSet, rules = fpgrowth(transactions, minSupRatio=0.01, minConf=0.2)
     best_fp_rules = sorted(rules, key=lambda rule: rule[2], reverse=True)[:10]
     for rule in best_fp_rules:
         print(f"Rule: {rule[0]} -> {rule[1]}, confidence: {rule[2]}")
     fp_time = time.time() - start
```

Рисунок 1.7 – Программа

```
Rule: {'eggs', 'ground beef'} -> {'mineral water'}, confidence: 0.5066666666666667
Rule: {'milk', 'ground beef'} -> {'mineral water'}, confidence: 0.503030303030303
Rule: {'ground beef', 'chocolate'} -> {'mineral water'}, confidence: 0.47398843930635837
Rule: {'milk', 'frozen vegetables'} -> {'mineral water'}, confidence: 0.4689265536723164
Rule: {'soup'} -> {'mineral water'}, confidence: 0.45646437994722955
Rule: {'spaghetti', 'pancakes'} -> {'mineral water'}, confidence: 0.455026455026455
Rule: {'olive oil', 'spaghetti'} -> {'mineral water'}, confidence: 0.4476744186046512
Rule: {'milk', 'spaghetti'} -> {'mineral water'}, confidence: 0.443609022556390 95
Rule: {'milk', 'chocolate'} -> {'mineral water'}, confidence: 0.43568464730290457
Rule: {'spaghetti', 'ground beef'} -> {'mineral water'}, confidence: 0.43537414965986393
```

Рисунок 1.8 – Результат выполнения программы

1.5 Задача №5

```
transactions = data.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
execution_times = {}
execution_times['apyori'] = apyori_time
execution_times['efficient_apriori'] = efficient_apriori_time
execution_times['apriori_python'] = apriori_python_time
execution_times['fpgrowth'] = fp_time

algorithms = list(execution_times.keys())
times = list(execution_times.values())

plt.figure(figsize=(8, 6))
plt.bar(algorithms, times)
plt.title('Сравнение Времени Выполнения Различных Алгоритмов', fontsize=14)
plt.xlabel('Алгоритм', fontsize=12)
plt.ylabel('Время выполнения (секунды)', fontsize=12)
plt.show()
for algorithm, exec_time in execution_times.items():
    print(f"Time for {algorithm}: {exec_time:.4f} seconds")
```

Рисунок 1.9 – Программа

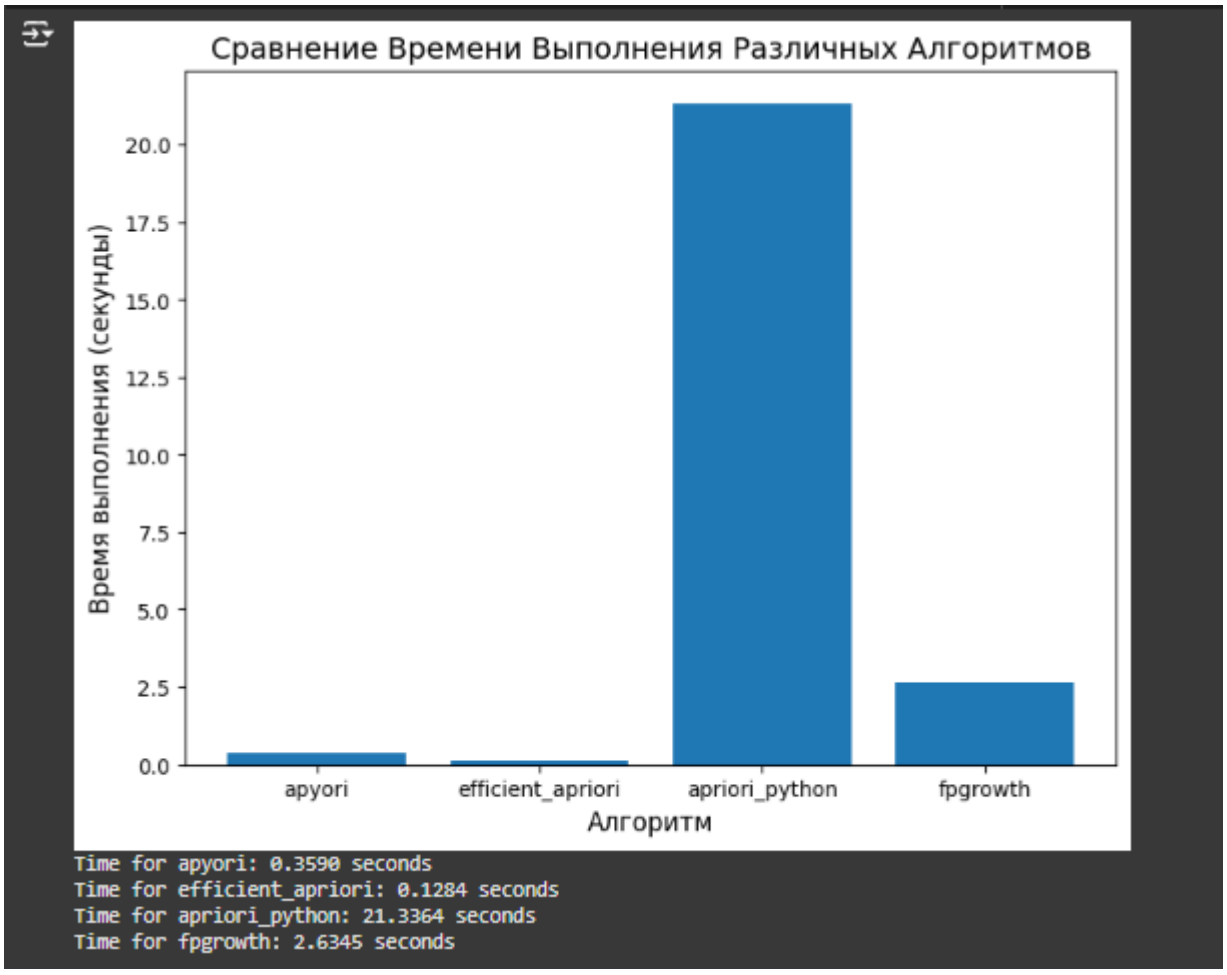

Рисунок 1.10 – Результат выполнения программы

1.6 Задача №6

```
url = 'https://raw.githubusercontent.com/InspectorJelly/BigDataMirea/refs/heads/main/datasets/data.csv'
data = pd.read_csv(url)
```

Рисунок 1.11 – Программа

1.7 Задача №7

```
all_items = data.values.flatten()
all_items = pd.Series(all_items).dropna()
item_counts = all_items.value_counts()
top_20_items = item_counts.head(20)
relative_frequencies = top_20_items / top_20_items.sum()

plt.figure(figsize=(14, 6))
plt.bar(top_20_items.index, top_20_items.values, color='skyblue')
plt.title('Топ-20 товаров по фактической частоте', fontsize=16)
plt.ylabel('Частота', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.tight_layout()
plt.show()

plt.figure(figsize=(14, 6))
plt.bar(relative_frequencies.index, relative_frequencies.values, color='salmon')
plt.title('Топ-20 товаров по относительной частоте', fontsize=16)
plt.ylabel('Относительная частота', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.tight_layout()
plt.show()
```

Рисунок 1.12 – Программа

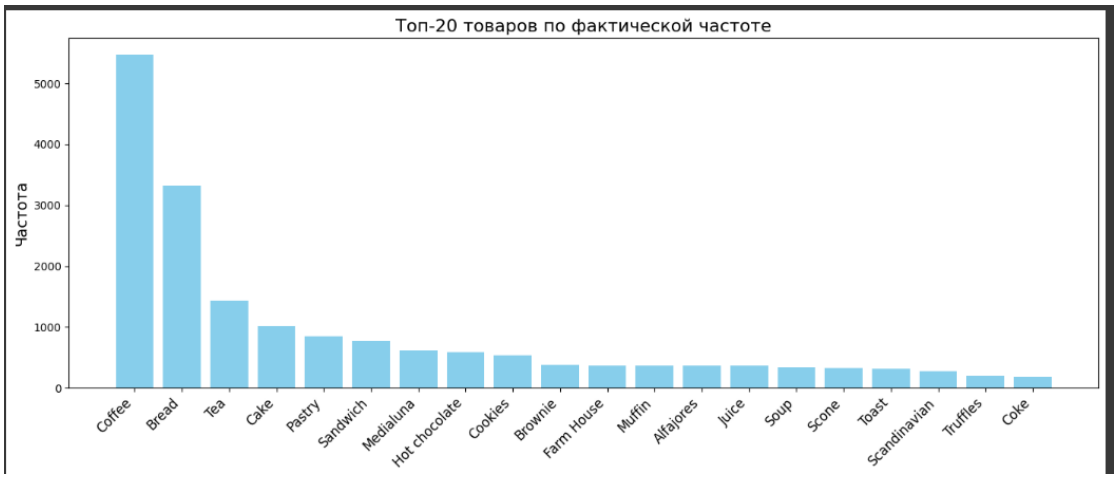

Рисунок 1.13 – Результат выполнения программы, фактическая частота

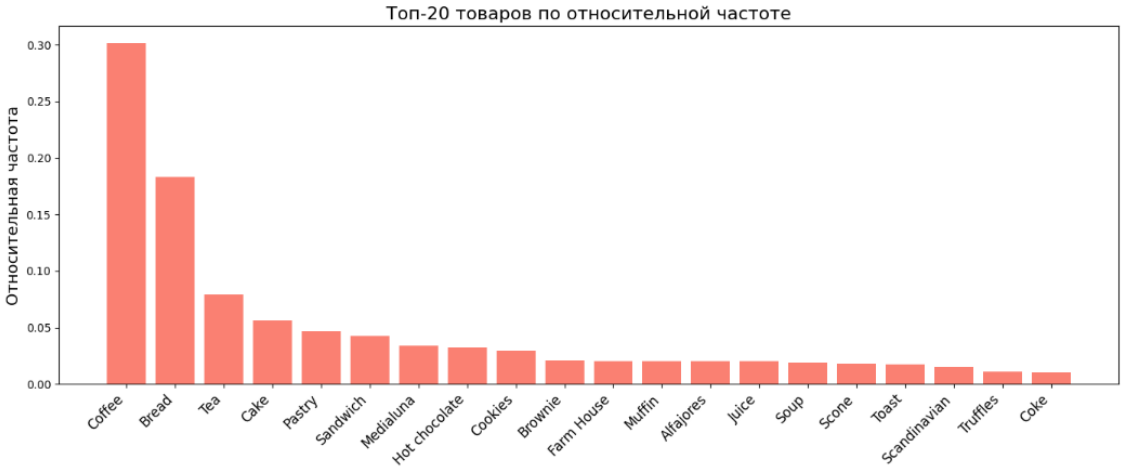

Рисунок 1.14 – Результат выполнения программы, относительная частота

1.8 Задача №8

```
transactions = data.apply(lambda row: row.dropna().tolist(), axis=1).tolist()

#apyori
start = time.time()
print("Results with apyori:")
apyori_results = list(apyori_apriori(transactions, min_support=0.005, min_confidence=0.3))
for rule in apyori_results[:10]:
    print(rule)
apyori_time = time.time() - start

#efficient_apriori
start = time.time()
print("\nResults with efficient_apriori:")
itemsets, rules = efficient_apriori(transactions, min_support=0.005, min_confidence=0.3)
for rule in sorted(rules, key=lambda x: x.lift, reverse=True)[:10]:
    print(rule)
efficient_apriori_time = time.time() - start

#apriori_python
start = time.time()
print("\nResults with apriori_python:")
freq_itemsets, rules = apriori_python(transactions, minSup=0.005, minConf=0.3)


# Вывод правил
print("Rules (apriori_python):")
for rule in rules[:10]:
    antecedent, consequent, confidence = rule
    print(f"Rule: {set(antecedent)} -> {set(consequent)}, Confidence: {confidence}")
apriori_python_time = time.time() - start
```

Рисунок 1.15 – Программа

Рисунок 1.16 – Результат выполнения программы

1.9 Задача №9

```
[39] start = time.time()
     freqItemSet, rules = fpgrowth(transactions, minSupRatio=0.01, minConf=0.2)
     best_fp_rules = sorted(rules, key=lambda rule: rule[2], reverse=True)[:10]
     for rule in best_fp_rules:
         print(f"Rule: {rule[0]} -> {rule[1]}, confidence: {rule[2]}")
     fp_time = time.time() - start
```

Рисунок 1.17 – Программа

```
Rule: {'Toast'} -> {'Coffee'}, confidence: 0.7044025157232704
Rule: {'Spanish Brunch'} -> {'Coffee'}, confidence: 0.5988372093023255
Rule: {'Medialuna'} -> {'Coffee'}, confidence: 0.5692307692307692
Rule: {'Medialuna'} -> {'Coffee'}, confidence: 0.5692307692307692
Rule: {'Pastry'} -> {'Coffee'}, confidence: 0.5521472392638037
Rule: {'Pastry'} -> {'Coffee'}, confidence: 0.5521472392638037
Rule: {'Tiffin'} -> {'Coffee'}, confidence: 0.547945205479452
Rule: {'Alfajores'} -> {'Coffee'}, confidence: 0.5406976744186046
Rule: {'Juice'} -> {'Coffee'}, confidence: 0.5342465753424658
Rule: {'Sandwich'} -> {'Coffee'}, confidence: 0.5323529411764706
```

Рисунок 1.18 – Результат выполнения программы

1.10 Задача №10

```
transactions = data.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
execution_times = {}
execution_times['apyori'] = apyori_time
execution_times['efficient_apriori'] = efficient_apriori_time
execution_times['apriori_python'] = apriori_python_time
execution_times['fpgrowth'] = fp_time

algorithms = list(execution_times.keys())
times = list(execution_times.values())

plt.figure(figsize=(8, 6))
plt.bar(algorithms, times)
plt.title('Сравнение Времени Выполнения Различных Алгоритмов', fontsize=14)
plt.xlabel('Алгоритм', fontsize=12)
plt.ylabel('Время выполнения (секунды)', fontsize=12)
plt.show()
for algorithm, exec_time in execution_times.items():
    print(f"Time for {algorithm}: {exec_time:.4f} seconds")
```

Рисунок 1.19 – Программа

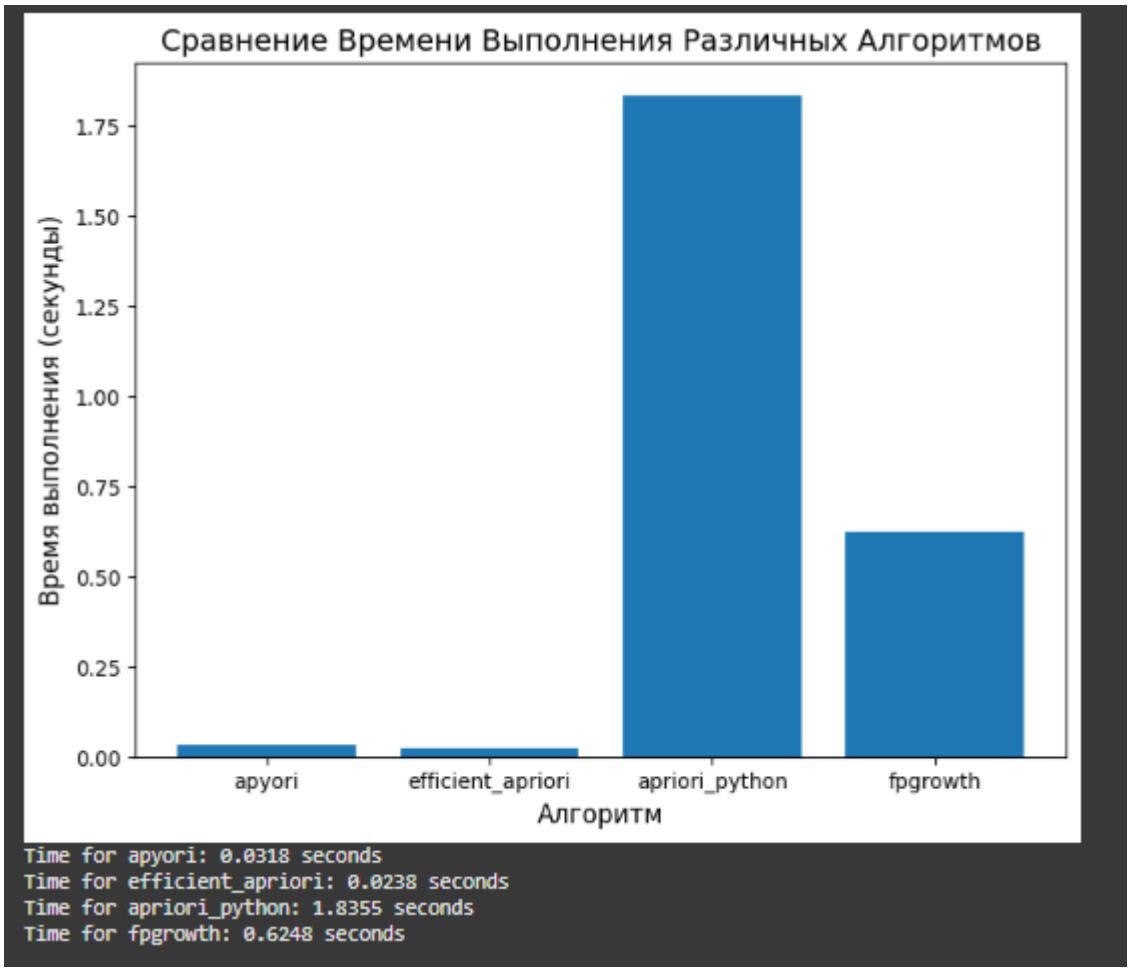

Рисунок 1.20 – Результат выполнения программы