



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

Лазарев А. В.

ИББО-03-21

Принял преподаватель кафедры прикладной
математики

Тетерин Н.Н.

Практическая работа выполнена

« __ » _____ 2024 г.

«Зачтено»

« __ » _____ 2024 г.

Москва 2024

СОДЕРЖАНИЕ

1 РЕШЕНИЕ ЗАДАЧ	3
1.1 Задача №1	3
1.2 Задача №2	3
1.3 Задача №3	4
1.4 Задача №4	5

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

```

Results with apriori:
RelationRecord(items=frozenset({'almonds', 'eggs'}), support=0.006333333333333334, ordered_statistics=[OrderedStatistic(items_base=frozenset({'almonds'}), items_add=frozenset({'eggs'}), confidence=0.32236042105263164, lift=1.7935920471829208)])
RelationRecord(items=frozenset({'mineral water', 'almonds'}), support=0.007466666666666667, ordered_statistics=[OrderedStatistic(items_base=frozenset({'almonds'}), items_add=frozenset({'mineral water'}), confidence=0.368421052631579, lift=1.5462551173681267)])
RelationRecord(items=frozenset({'mineral water', 'avocado'}), support=0.011466666666666667, ordered_statistics=[OrderedStatistic(items_base=frozenset({'avocado'}), items_add=frozenset({'mineral water'}), confidence=0.345381526184617, lift=1.44045927278489)])
RelationRecord(items=frozenset({'black tea', 'mineral water'}), support=0.006333333333333333, ordered_statistics=[OrderedStatistic(items_base=frozenset({'black tea'}), items_add=frozenset({'mineral water'}), confidence=0.37383177570093457, lift=1.5689638040050415)])
RelationRecord(items=frozenset({'eggs', 'burgers'}), support=0.0288, ordered_statistics=[OrderedStatistic(items_base=frozenset({'burgers'}), items_add=frozenset({'eggs'}), confidence=0.33027223577982, lift=1.83758473307388)])
RelationRecord(items=frozenset({'mineral water', 'cake'}), support=0.027466666666666667, ordered_statistics=[OrderedStatistic(items_base=frozenset({'cake'}), items_add=frozenset({'mineral water'}), confidence=0.3388357094786406, lift=1.422002474006188)])
RelationRecord(items=frozenset({'mineral water', 'carrots'}), support=0.006333333333333333, ordered_statistics=[OrderedStatistic(items_base=frozenset({'carrots'}), items_add=frozenset({'mineral water'}), confidence=0.34782608695652173, lift=1.4591818495900343)])
RelationRecord(items=frozenset({'cereals', 'mineral water'}), support=0.010266666666666667, ordered_statistics=[OrderedStatistic(items_base=frozenset({'cereals'}), items_add=frozenset({'mineral water'}), confidence=0.390637305629482, lift=1.674442070103303)])
RelationRecord(items=frozenset({'mineral water', 'chicken'}), support=0.0228, ordered_statistics=[OrderedStatistic(items_base=frozenset({'chicken'}), items_add=frozenset({'mineral water'}), confidence=0.38, lift=1.594831706773125)])
RelationRecord(items=frozenset({'mineral water', 'chocolate'}), support=0.05266666666666667, ordered_statistics=[OrderedStatistic(items_base=frozenset({'chocolate'}), items_add=frozenset({'mineral water'}), confidence=0.32139951179820997, lift=1.3489067367820564)])

Results with efficient_apriori:
(pasta) -> {escalope} (conf: 0.373, supp: 0.006, lift: 4.700, conv: 1.468)
(pasta) -> {shrimp} (conf: 0.322, supp: 0.006, lift: 4.514, conv: 1.370)
(herb & pepper, spaghetti) -> {ground beef} (conf: 0.393, supp: 0.006, lift: 4.004, conv: 1.487)
(herb & pepper, mineral water) -> {ground beef} (conf: 0.391, supp: 0.007, lift: 3.575, conv: 1.400)
(tomato sauce) -> {ground beef} (conf: 0.377, supp: 0.005, lift: 3.840, conv: 1.448)
(mushroom cream sauce) -> {escalope} (conf: 0.301, supp: 0.006, lift: 3.790, conv: 1.317)
(spaghetti, tomatoes) -> {frozen vegetables} (conf: 0.318, supp: 0.007, lift: 3.341, conv: 1.327)
(herb & pepper) -> {ground beef} (conf: 0.323, supp: 0.016, lift: 3.292, conv: 1.333)
(grated cheese, spaghetti) -> {ground beef} (conf: 0.323, supp: 0.006, lift: 3.284, conv: 1.331)
(mineral water, shrimp) -> {frozen vegetables} (conf: 0.307, supp: 0.007, lift: 3.218, conv: 1.305)

Results with apriori_python:
Rules (apriori_python):
Rules: {'mineral water', 'french fries'} -> {'spaghetti'}, Confidence: 0.30039525091699603
Rules: {'mushroom cream sauce'} -> {'escalope'}, Confidence: 0.30009930009930007
Rules: {'frozen vegetables', 'french fries'} -> {'milk'}, Confidence: 0.30009930009930007
Rules: {'soup'} -> {'milk'}, Confidence: 0.3007915567282322
Rules: {'olive oil', 'mineral water'} -> {'chocolate'}, Confidence: 0.30097087377064074
Rules: {'chocolate', 'green tea'} -> {'spaghetti'}, Confidence: 0.30113636363636365
Rules: {'mineral water', 'chocolate'} -> {'spaghetti'}, Confidence: 0.3012658227848101
Rules: {'shrimp', 'spaghetti'} -> {'chocolate'}, Confidence: 0.301880923623302
Rules: {'vegetables mix'} -> {'mineral water'}, Confidence: 0.3020833333333333
Rules: {'milk', 'french fries'} -> {'eggs'}, Confidence: 0.3033707051685395

```

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: 0.44360902255639095
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

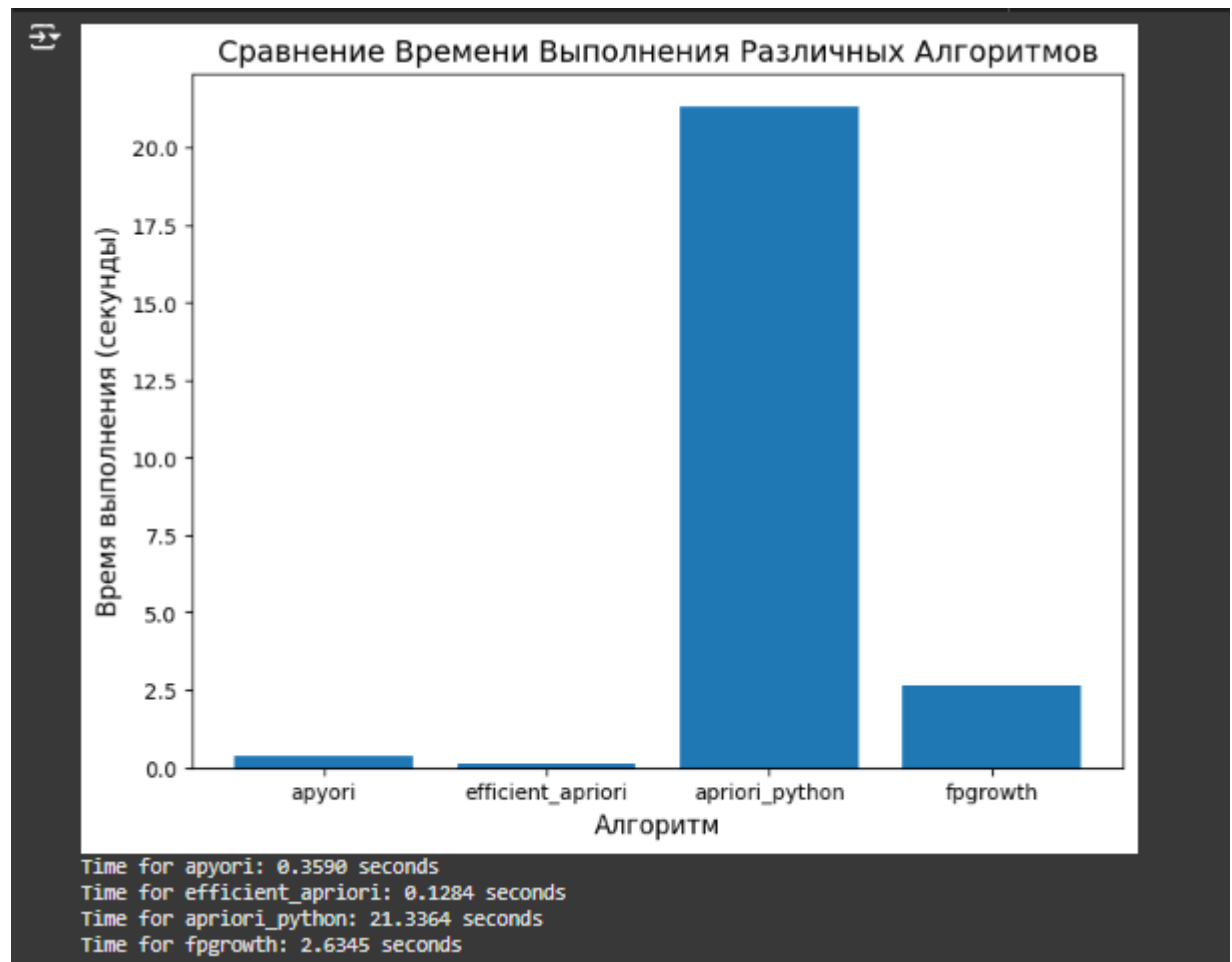


Рисунок 1.10 – Результат выполнения программы

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

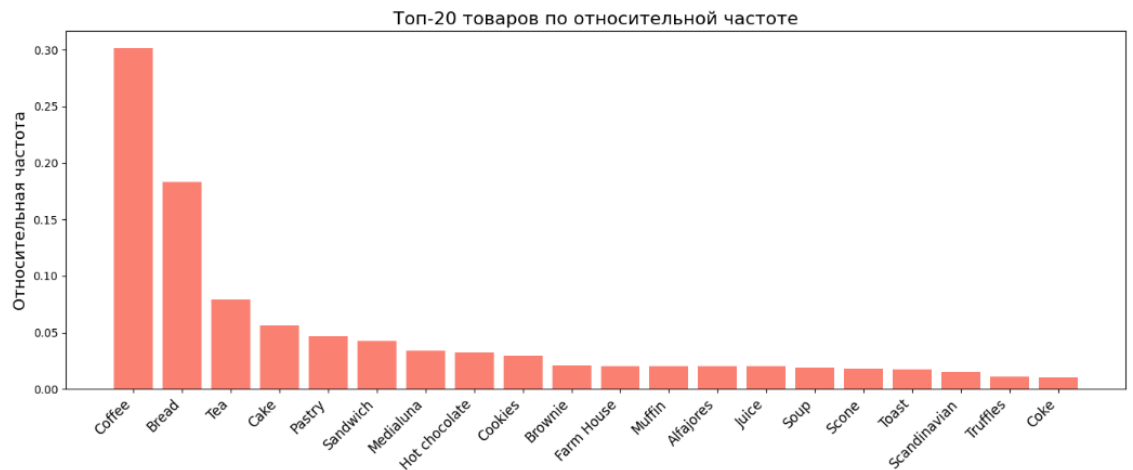


Рисунок 1.14 – Результат выполнения программы, относительная частота

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


```

Results with apriori:
relationccord(items=frozenset({'bread'}), support=0.324868352578029, ordered_statistics=[orderedStatistic(items_base=frozenset(), items_add=frozenset({'bread'}), confidence=0.324868352578029, lift=1.0)])
relationccord(items=frozenset({'coffee'}), support=0.475111647429171, ordered_statistics=[orderedStatistic(items_base=frozenset(), items_add=frozenset({'coffee'}), confidence=0.475111647429171, lift=1.0)])
relationccord(items=frozenset({'alfajores', 'coffee'}), support=0.431517317960506, ordered_statistics=[orderedStatistic(items_base=frozenset({'alfajores'}), items_add=frozenset({'coffee'}), confidence=0.5486976744186047, lift=1.137996651327143)])
relationccord(items=frozenset({'jam', 'bread'}), support=0.0050367261218816789, ordered_statistics=[orderedStatistic(items_base=frozenset({'jam'}), items_add=frozenset({'bread'}), confidence=0.338021638148845, lift=1.048696605246286)])
relationccord(items=frozenset({'bread', 'pastry'}), support=0.42308111523468537, ordered_statistics=[orderedStatistic(items_base=frozenset({'pastry'}), items_add=frozenset({'bread'}), confidence=0.336550261246462, lift=1.0424214423568745)])
relationccord(items=frozenset({'brownie', 'coffee'}), support=0.419617317960506, ordered_statistics=[orderedStatistic(items_base=frozenset({'brownie'}), items_add=frozenset({'coffee'}), confidence=0.4087651715839578, lift=1.032004612286376)])
relationccord(items=frozenset({'coffee', 'cake'}), support=0.405394660429478, ordered_statistics=[orderedStatistic(items_base=frozenset({'cake'}), items_add=frozenset({'coffee'}), confidence=0.5209612994940434, lift=1.1890798131032724)])
relationccord(items=frozenset({'chicken stew', 'coffee'}), support=0.0054145792259847, ordered_statistics=[orderedStatistic(items_base=frozenset({'chicken stew'}), items_add=frozenset({'coffee'}), confidence=0.3983738237198374, lift=0.8338505443994389)])
relationccord(items=frozenset({'coffee', 'coke'}), support=0.0064088394543546695, ordered_statistics=[orderedStatistic(items_base=frozenset({'coke'}), items_add=frozenset({'coffee'}), confidence=0.3315217391384376, lift=0.6977478229277923)])
relationccord(items=frozenset({'coffee', 'cookies'}), support=0.42308111523468537, ordered_statistics=[orderedStatistic(items_base=frozenset({'cookies'}), items_add=frozenset({'coffee'}), confidence=0.5104466812417405, lift=1.0911652280761602)])

Results with efficient_apriori:
(keeping 11 local) -> {coffee} (conf: 0.810, supp: 0.005, lift: 1.704, conv: 2.750)
{toast} -> {coffee} (conf: 0.704, supp: 0.404, lift: 1.483, conv: 1.770)
{salad} -> {coffee} (conf: 0.420, supp: 0.007, lift: 1.318, conv: 1.404)
{cake, hot chocolate} -> {coffee} (conf: 0.682, supp: 0.007, lift: 1.207, conv: 1.318)
{spanish brunch} -> {coffee} (conf: 0.599, supp: 0.411, lift: 1.206, conv: 1.300)
{medialuna} -> {coffee} (conf: 0.569, supp: 0.430, lift: 1.158, conv: 1.210)
{pastry} -> {coffee} (conf: 0.552, supp: 0.007, lift: 1.162, conv: 1.172)
{tiffin} -> {coffee} (conf: 0.540, supp: 0.006, lift: 1.152, conv: 1.103)
{alfajores} -> {coffee} (conf: 0.541, supp: 0.408, lift: 1.138, conv: 1.143)
{hearty & seasonal} -> {coffee} (conf: 0.504, supp: 0.006, lift: 1.137, conv: 1.143)

Results with apriori_python:
rules (apriori_python):
Rule: {'coke'} -> {'coffee'}, Confidence: 0.3315217391384376
Rule: {'jam'} -> {'bread'}, Confidence: 0.338021638148845
Rule: {'pastry'} -> {'bread'}, Confidence: 0.338021638148845
Rule: {'tea'} -> {'coffee'}, Confidence: 0.3696262629629296
Rule: {'bread', 'cookies'} -> {'coffee'}, Confidence: 0.3696262629629296
Rule: {'truffles'} -> {'coffee'}, Confidence: 0.3697916666666667
Rule: {'tea', 'sandwich'} -> {'coffee'}, Confidence: 0.375
Rule: {'bread', 'pastry'} -> {'coffee'}, Confidence: 0.38462797181449274
Rule: {'mineral water'} -> {'coffee'}, Confidence: 0.39552238805970147
Rule: {'chicken stew'} -> {'coffee'}, Confidence: 0.3983738237198374

```

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

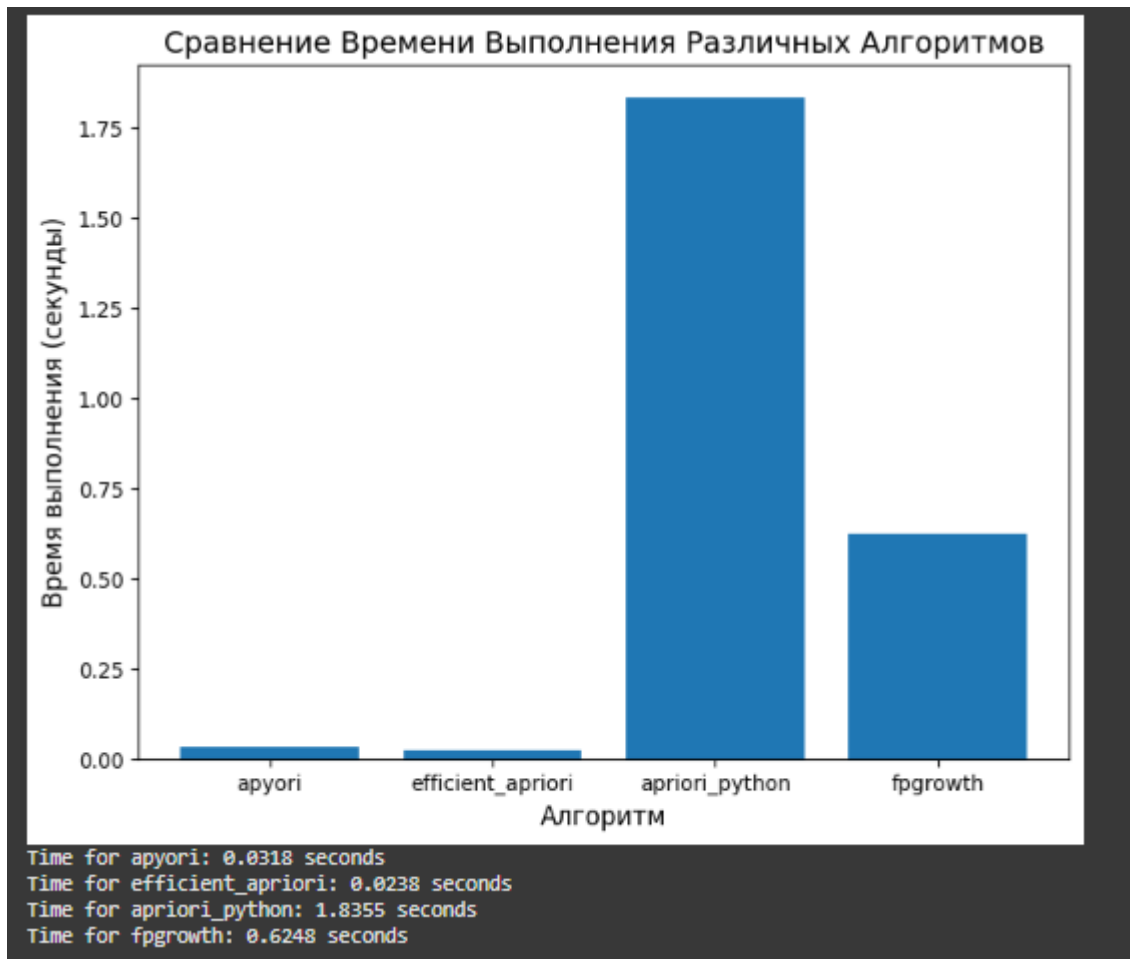


Рисунок 1.20 – Результат выполнения программы