МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра прикладной математики

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8

по дисциплине «Технологии и инструментарий анализа больших данных»

Выполнил студент группы ИКБО-20-21 Фомичев Р.А.

Проверил ассистент кафедры ПМ ИИТ Тетерин Н.Н.

Москва 2024

Задание 1-2

Загрузить данные Market_Basket_Optimisation.csv. Визуализировать данные (отразить на гистограммах относительную и фактическую частоту встречаемости для 20 наиболее популярных товаров). Код программы с выводом представлены на рисунке 1.

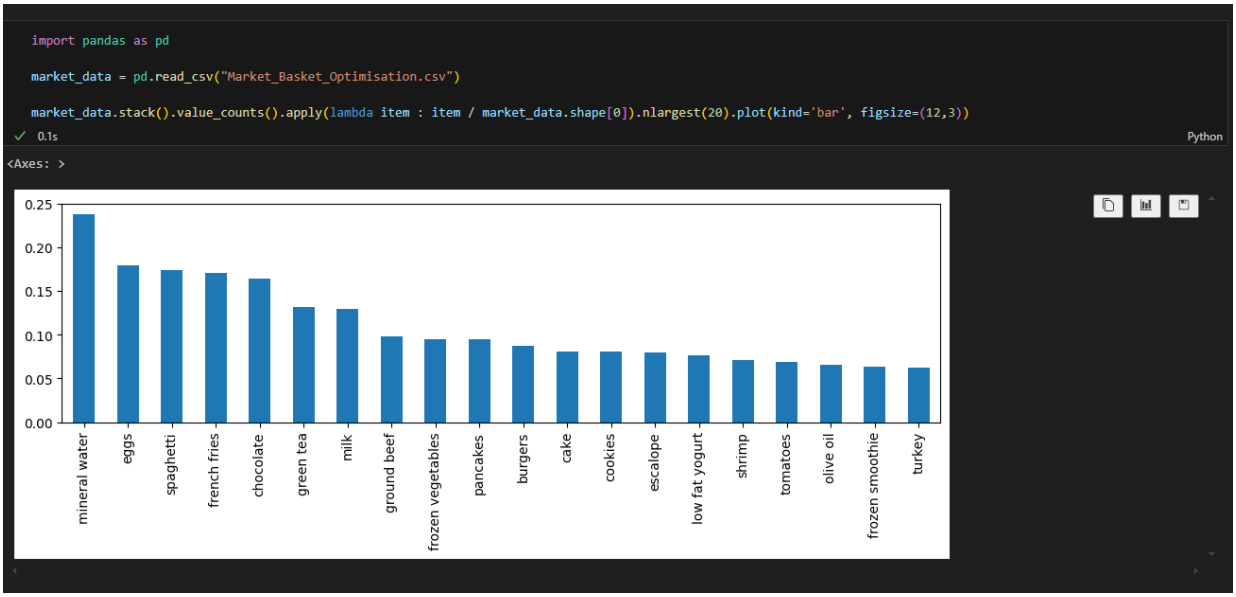


Рисунок 1 – Код программы с выводом

Задание 3

Применить алгоритм Apriori, используя 3 разные библиотеки (apriori_python, apyori, efficient_apriori). Подобрать гиперпараметры для алгоритмов так, чтобы выводилось порядка 10 наилучших правил. Код программы с выводом представлены на рисунках 2 - 5.

```python
print ("1. apriori_python \n")

transactions = []
for i in range(market_data.shape[0]):
    row = market_data.iloc[i].dropna().tolist()
    transactions.append(row)

start = time.time()
t1, rules = apr_1(transactions, minSup=0.01, minConf=0.43)
end = time.time()

total_time_1 = end - start
rules
```

✓ 5.6s

```
1. apriori_python


[[{'frozen vegetables', 'spaghetti'}, {'mineral water'}, 0.430622009569378],
 [{'ground beef', 'spaghetti'}, {'mineral water'}, 0.43537414965986393],
 [{'chocolate', 'milk'}, {'mineral water'}, 0.43568464730290457],
 [{'milk', 'spaghetti'}, {'mineral water'}, 0.44360902255639095],
 [{'olive oil', 'spaghetti'}, {'mineral water'}, 0.4476744186046512],
 [{'pancakes', 'spaghetti'}, {'mineral water'}, 0.455026455026455],
 [{'soup'}, {'mineral water'}, 0.45646437994722955],
 [{'frozen vegetables', 'milk'}, {'mineral water'}, 0.4689265536723164],
 [{'chocolate', 'ground beef'}, {'mineral water'}, 0.47398843930635837],
 [{'ground beef', 'milk'}, {'mineral water'}, 0.503030303030303],
 [{'eggs', 'ground beef'}, {'mineral water'}, 0.5066666666666667]]
```

Рисунок 2 – Код программы с выводом

```python
print ("2. apyori \n\n\n")

start = time.time()
rules = apr_2(transactions=transactions, min_support=0.01, min_confidence=0.43, min_lift=1.0001)
end = time.time()

total_time_2 = end - start

result = list(rules)
for row in result:
    for subset in row[2]:
        print(*subset[0], "|-|", *subset[1])
        print(f"Support: {row[1]:.2f}; Confidence: {subset[2]:.2f}; Lift: {subset[3]:.2f}.\n")
```

✓ 0.0s

```
2. apyori



soup |-| mineral water
Support: 0.02; Confidence: 0.46; Lift: 1.92.

chocolate ground beef |-| mineral water
Support: 0.01; Confidence: 0.47; Lift: 1.99.

chocolate milk |-| mineral water
Support: 0.01; Confidence: 0.44; Lift: 1.83.

ground beef eggs |-| mineral water
Support: 0.01; Confidence: 0.51; Lift: 2.13.
```

Рисунок 3 – Код программы с выводом

```
ground beef eggs |-| mineral water
Support: 0.01; Confidence: 0.51; Lift: 2.13.

milk frozen vegetables |-| mineral water
Support: 0.01; Confidence: 0.47; Lift: 1.97.

frozen vegetables spaghetti |-| mineral water
Support: 0.01; Confidence: 0.43; Lift: 1.81.

ground beef milk |-| mineral water
Support: 0.01; Confidence: 0.50; Lift: 2.11.

ground beef spaghetti |-| mineral water
Support: 0.02; Confidence: 0.44; Lift: 1.83.

milk spaghetti |-| mineral water
Support: 0.02; Confidence: 0.44; Lift: 1.86.

olive oil spaghetti |-| mineral water
Support: 0.01; Confidence: 0.45; Lift: 1.88.

pancakes spaghetti |-| mineral water
Support: 0.01; Confidence: 0.46; Lift: 1.91.
```

Рисунок 4 – Вывод программы

```python
print ("3. efficient_apriori \n\n\n")

start = time.time()
itemstst, rules = apr_3(transactions, min_support=0.01, min_confidence=0.43)
end = time.time()

total_time_3 = end - start

for row in rules:
    print(row)
```

✓ 0.0s

```
3. efficient_apriori


{soup} -> {mineral water} (conf: 0.456, supp: 0.023, lift: 1.916, conv: 1.401)
{chocolate, ground beef} -> {mineral water} (conf: 0.474, supp: 0.011, lift: 1.989, conv: 1.448)
{chocolate, milk} -> {mineral water} (conf: 0.436, supp: 0.014, lift: 1.829, conv: 1.350)
{eggs, ground beef} -> {mineral water} (conf: 0.507, supp: 0.010, lift: 2.126, conv: 1.544)
{frozen vegetables, milk} -> {mineral water} (conf: 0.469, supp: 0.011, lift: 1.968, conv: 1.434)
{frozen vegetables, spaghetti} -> {mineral water} (conf: 0.431, supp: 0.012, lift: 1.807, conv: 1.338)
{ground beef, milk} -> {mineral water} (conf: 0.503, supp: 0.011, lift: 2.111, conv: 1.533)
{ground beef, spaghetti} -> {mineral water} (conf: 0.435, supp: 0.017, lift: 1.827, conv: 1.349)
{milk, spaghetti} -> {mineral water} (conf: 0.444, supp: 0.016, lift: 1.862, conv: 1.369)
{olive oil, spaghetti} -> {mineral water} (conf: 0.448, supp: 0.010, lift: 1.879, conv: 1.379)
{pancakes, spaghetti} -> {mineral water} (conf: 0.455, supp: 0.011, lift: 1.910, conv: 1.398)
```

Рисунок 5 – Код программы с выводом

Задание 4

Применить алгоритм FP-Growth из библиотеки fpgrowth_py. Подобрать гиперпараметры для алгоритма так, чтобы выводилось порядка 10 наилучших правил. Код программы с выводом представлены на рисунке 6.

```
print ("4. fpgrowth_py \n\n\n")

start = time.time()
itemsets, rules = fpgrowth(transactions, minSupRatio=0.01, minConf=0.43)
end = time.time()

total_time_4 = end - start

for row in rules:
    print(row)
```

✓ 0.4s

```
4. fpgrowth_py



[{'Tiffin'}, {'Coffee'}, 0.547945205479452]
[{'Spanish Brunch'}, {'Coffee'}, 0.5988372093023255]
[{'Toast'}, {'Coffee'}, 0.7044025157232704]
[{'Scone'}, {'Coffee'}, 0.5229357798165137]
[{'Soup'}, {'Coffee'}, 0.4601226993865031]
[{'Juice'}, {'Coffee'}, 0.5342465753424658]
[{'Alfajores'}, {'Coffee'}, 0.5406976744186046]
[{'Muffin'}, {'Coffee'}, 0.489010989010989]
[{'Brownie'}, {'Coffee'}, 0.49076517150395776]
[{'Cookies'}, {'Coffee'}, 0.5184466019417475]
[{'Cookies'}, {'Coffee'}, 0.5184466019417475]
[{'Hot chocolate'}, {'Coffee'}, 0.5072463768115942]
[{'Medialuna'}, {'Coffee'}, 0.5692307692307692]
[{'Medialuna'}, {'Coffee'}, 0.5692307692307692]
[{'Sandwich'}, {'Coffee'}, 0.5323529411764706]
[{'Sandwich'}, {'Coffee'}, 0.5323529411764706]
[{'Pastry'}, {'Coffee'}, 0.5521472392638037]
[{'Pastry'}, {'Coffee'}, 0.5521472392638037]
[{'Cake'}, {'Coffee'}, 0.5269582909460834]
[{'Cake'}, {'Coffee'}, 0.5269582909460834]
```

+ Code

Рисунок 6 – Код программы с выводом

Задание 5

Сравнить время выполнения всех алгоритмов и построить гистограмму. Код программы с выводом представлены на рисунке 7.

```python
import matplotlib.pyplot as plt

print(f"apriori_python: {total_time_1:.2f} c.")
print(f"apyori: {total_time_2:.4f} c.")
print(f"efficient_apriori: {total_time_3:.2f} c.")
print(f"fpgrowth_py: {total_time_4:.2f} c.")

x = ["apriori_python", "apyori", "efficient apriori", "fpgrowth_py"]
y = [total_time_1, total_time_2 ,total_time_3, total_time_4]

plt.figure(figsize=(10, 3))

plt.subplot(1, 2, 1)
plt.title("Время выполнения")
plt.bar(x, y)

plt.subplot(1, 2, 2)
plt.title("Время выполнения (log)")
plt.yscale("log")
plt.bar(x, y)

plt.show()
```

✓ 0.2s

```
apriori_python: 5.07 c.
apyori: 0.0000 c.
efficient_apriori: 0.04 c.
fpgrowth_py: 1.51 c.
```

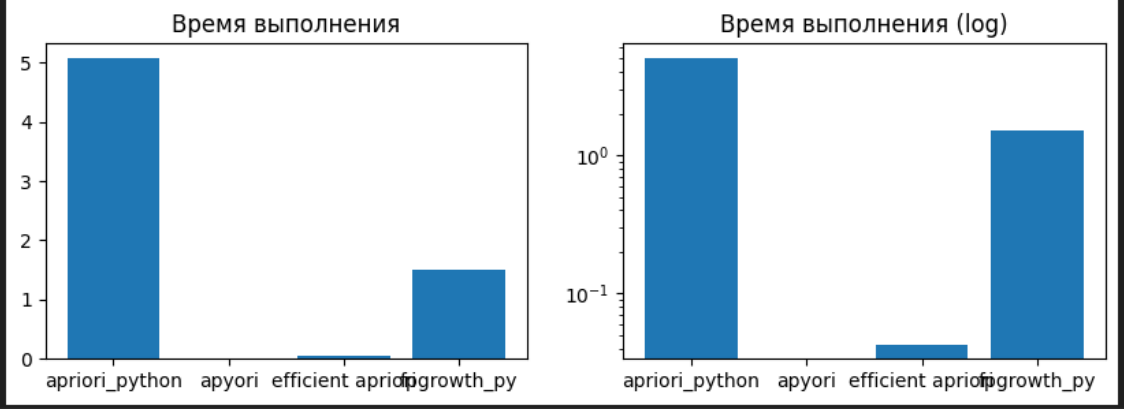


Рисунок 7 – Код программы с выводом

Задание 6-7

Загрузить данные data.csv. Визуализировать данные (отразить на гистограммах относительную и фактическую частоту встречаемости для 20 наиболее популярных товаров). Код программы с выводом представлены на рисунке 8.

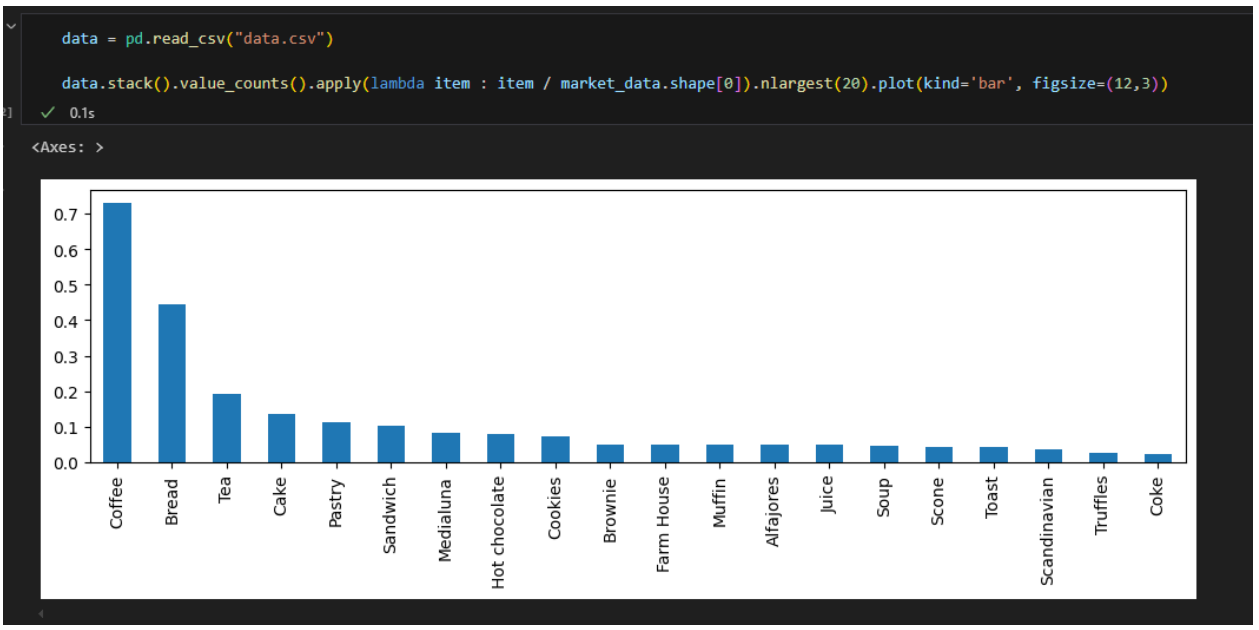
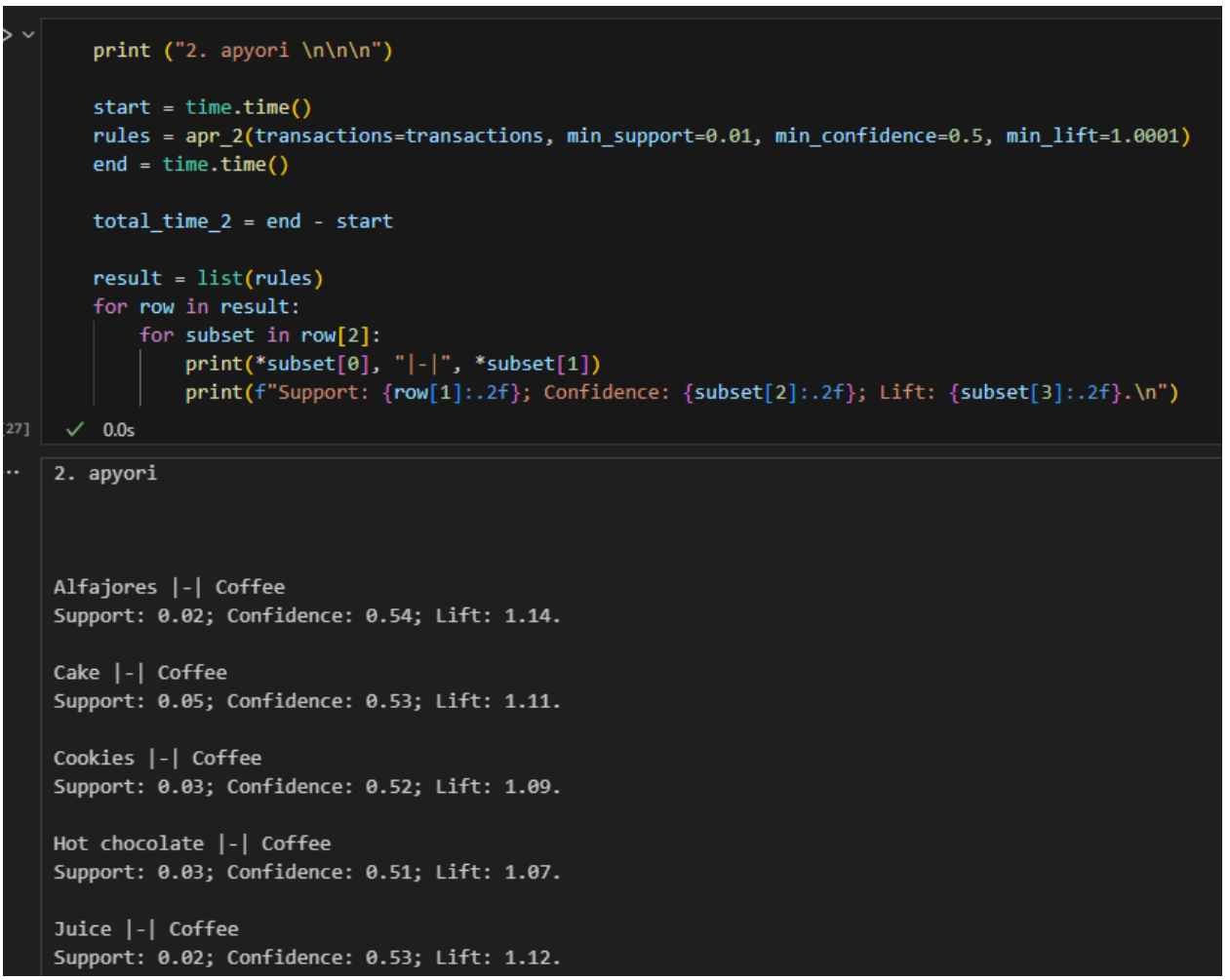


Рисунок 8 – Код программы с выводом

Задание 8

Применить алгоритм Apriori, используя 3 разные библиотеки (apriori_python, apyori, efficient_apriori). Подобрать гиперпараметры для алгоритмов так, чтобы выводилось порядка 10 наилучших правил. Код программы с выводом представлены на рисунках 9 – 12.



```
print ("2. apyori \n\n\n")

start = time.time()
rules = apr_2(transactions=transactions, min_support=0.01, min_confidence=0.5, min_lift=1.0001)
end = time.time()

total_time_2 = end - start

result = list(rules)
for row in result:
    for subset in row[2]:
        print(*subset[0], "|-|", *subset[1])
        print(f"Support: {row[1]:.2f}; Confidence: {subset[2]:.2f}; Lift: {subset[3]:.2f}.\n")
```

27] ✓ 0.0s

```
2. apyori



Alfajores |-| Coffee
Support: 0.02; Confidence: 0.54; Lift: 1.14.

Cake |-| Coffee
Support: 0.05; Confidence: 0.53; Lift: 1.11.

Cookies |-| Coffee
Support: 0.03; Confidence: 0.52; Lift: 1.09.

Hot chocolate |-| Coffee
Support: 0.03; Confidence: 0.51; Lift: 1.07.

Juice |-| Coffee
Support: 0.02; Confidence: 0.53; Lift: 1.12.
```

Рисунок 9 – Код программы с выводом

```python
print ("2. apyori \n\n\n")

start = time.time()
rules = apr_2(transactions=transactions, min_support=0.01, min_confidence=0.5, min_lift=1.0001)
end = time.time()

total_time_2 = end - start

result = list(rules)
for row in result:
    for subset in row[2]:
        print(*subset[0], "|-|", *subset[1])
        print(f"Support: {row[1]:.2f}; Confidence: {subset[2]:.2f}; Lift: {subset[3]:.2f}.\n")
```

✓ 0.0s

```
2. apyori



Alfajores |-| Coffee
Support: 0.02; Confidence: 0.54; Lift: 1.14.

Cake |-| Coffee
Support: 0.05; Confidence: 0.53; Lift: 1.11.

Cookies |-| Coffee
Support: 0.03; Confidence: 0.52; Lift: 1.09.

Hot chocolate |-| Coffee
Support: 0.03; Confidence: 0.51; Lift: 1.07.

Juice |-| Coffee
Support: 0.02; Confidence: 0.53; Lift: 1.12.

Medialuna |-| Coffee
Support: 0.03; Confidence: 0.57; Lift: 1.20.

Pastry |-| Coffee
Support: 0.05; Confidence: 0.55; Lift: 1.16.
```

Рисунок 10 – Код программы с выводом

```
Juice |-| Coffee
Support: 0.02; Confidence: 0.53; Lift: 1.12.

Medialuna |-| Coffee
Support: 0.03; Confidence: 0.57; Lift: 1.20.

Pastry |-| Coffee
Support: 0.05; Confidence: 0.55; Lift: 1.16.

Sandwich |-| Coffee
Support: 0.04; Confidence: 0.53; Lift: 1.12.

Scone |-| Coffee
Support: 0.02; Confidence: 0.52; Lift: 1.10.

Spanish Brunch |-| Coffee
Support: 0.01; Confidence: 0.60; Lift: 1.26.

Toast |-| Coffee
Support: 0.02; Confidence: 0.70; Lift: 1.48.
```

Рисунок 11 – Код программы с выводом

```
print ("3. efficient_apriori \n\n\n")

start = time.time()
itemstst, rules = apr_3(transactions, min_support=0.01, min_confidence=0.5)
end = time.time()

total_time_3 = end - start

for row in rules:
    print(row)
```

28] ✓ 0.0s

```
3. efficient_apriori


{Alfajores} -> {Coffee} (conf: 0.541, supp: 0.020, lift: 1.138, conv: 1.143)
{Cake} -> {Coffee} (conf: 0.527, supp: 0.054, lift: 1.109, conv: 1.110)
{Cookies} -> {Coffee} (conf: 0.518, supp: 0.028, lift: 1.091, conv: 1.090)
{Hot chocolate} -> {Coffee} (conf: 0.507, supp: 0.029, lift: 1.068, conv: 1.065)
{Juice} -> {Coffee} (conf: 0.534, supp: 0.020, lift: 1.124, conv: 1.127)
{Medialuna} -> {Coffee} (conf: 0.569, supp: 0.035, lift: 1.198, conv: 1.218)
{Pastry} -> {Coffee} (conf: 0.552, supp: 0.047, lift: 1.162, conv: 1.172)
{Sandwich} -> {Coffee} (conf: 0.532, supp: 0.038, lift: 1.120, conv: 1.122)
{Scone} -> {Coffee} (conf: 0.523, supp: 0.018, lift: 1.101, conv: 1.100)
{Spanish Brunch} -> {Coffee} (conf: 0.599, supp: 0.011, lift: 1.260, conv: 1.308)
{Toast} -> {Coffee} (conf: 0.704, supp: 0.024, lift: 1.483, conv: 1.776)
```

Рисунок 12 – Код программы с выводом

Задание 9

Применить алгоритм FP-Growth из библиотеки fpgrowth_py. Подобрать гиперпараметры для алгоритма так, чтобы выводилось порядка 10 наилучших правил. Код программы с выводом представлен на рисунке 13.

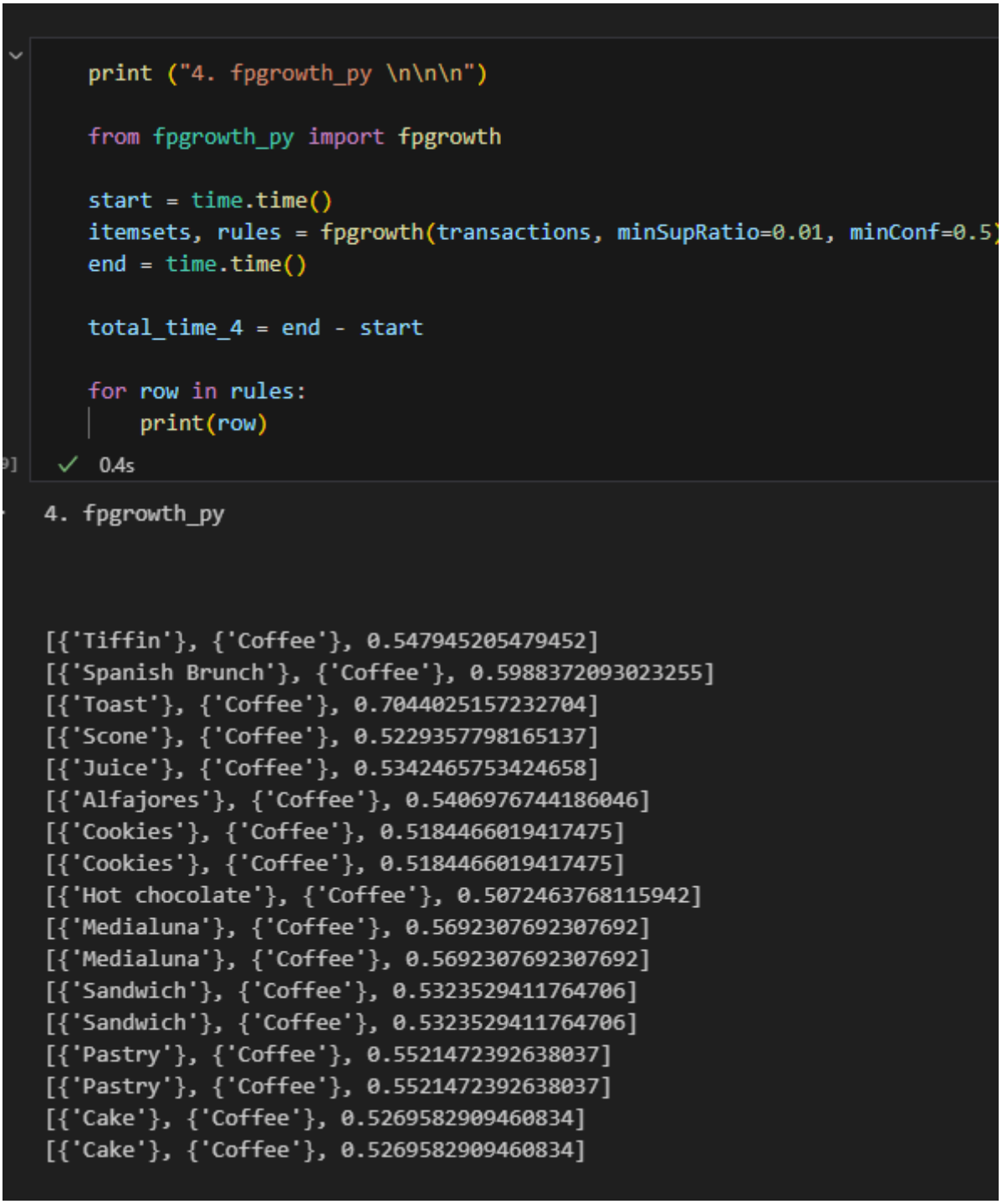


Рисунок 13 – Код программы с выводом

Задание 10

Сравнить время выполнения всех алгоритмов и построить гистограмму. Код программы с выводом представлены на рисунке 14.

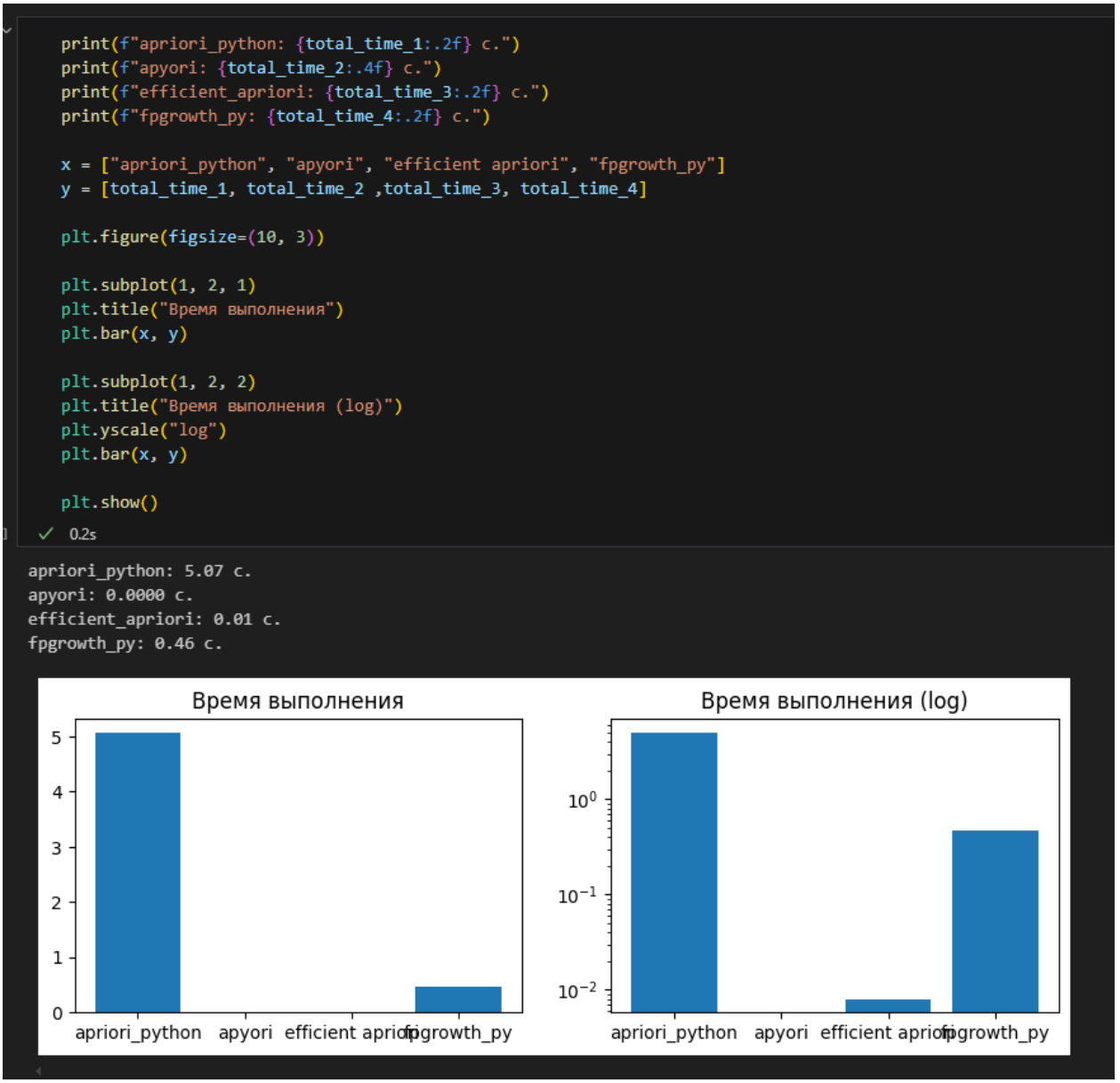


Рисунок 14 – Код программы с выводом