Assignment DMW-3

Title: Apriori Algorithm

Problem statement:

Apply Apriori algorithm to find frequently occuring items from given data & generate strong association using support & confidence thresholds.

Objectives:

- Understanding the concepts of association rules.
- Creating association rules to derive recommendations depending on the confidences of the rules.

Outcomes:

Students will be able to:

- Understand the concept of association rules.
- Create association rules to derive recommendations depending on the confidences of the rules.

S/w & H/w Requirements:

- Fedora 20 / Windows 10
- Jupyter Notebooks / Google Colab.

Theory:

Apriori Algorithm:

It is used for finding frequent itemsets in a dataset for boolean association rule. It uses prior knowledge of frequent itemset properties - we apply an iterative approach or level-wise search where k frequent itemsets are used to find k+1 itemsets.

Apriori Property:

All non-empty subsets of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that:

All subsets of a frequent itemset must be frequent.

If an item is infrequent, all its supersets will be infrequent.

Important Definitions:

1. Support: It is one of the measure of interestingness. This tells about the usefulness & certainity of rules. 5% support means 5% of transactions in database follow the rule.

   Support $(A \rightarrow B)$ = Support_count $(A \cup B)$

2. Confidence: A confidence of 60% means that 60% of the customers who purchased milk & butter also brought bread.

   Confidence $(A \rightarrow B)$ = Support_count$(A \cup B)$ / Support count (A)

If a rule satisfies both minimum support & minimum confidence it is a strong rule.

3. Support_count (x): No. of transactions in which X appears.

   If x is $(A \cup B)$ then it's the no. of transactions in which A & B.

4. Maximal Itemset: An itemset is maximal frequent if none of its supersets are frequent.

5. Closed itemset: An itemset is closed if none of its immediate supersets have some support count same as Itemset.

6 k-itemset: Itemset which contains k items is a k-itemset. So it can be said that an itemset is frequent if the corresponding support count is greater than minimum support count.

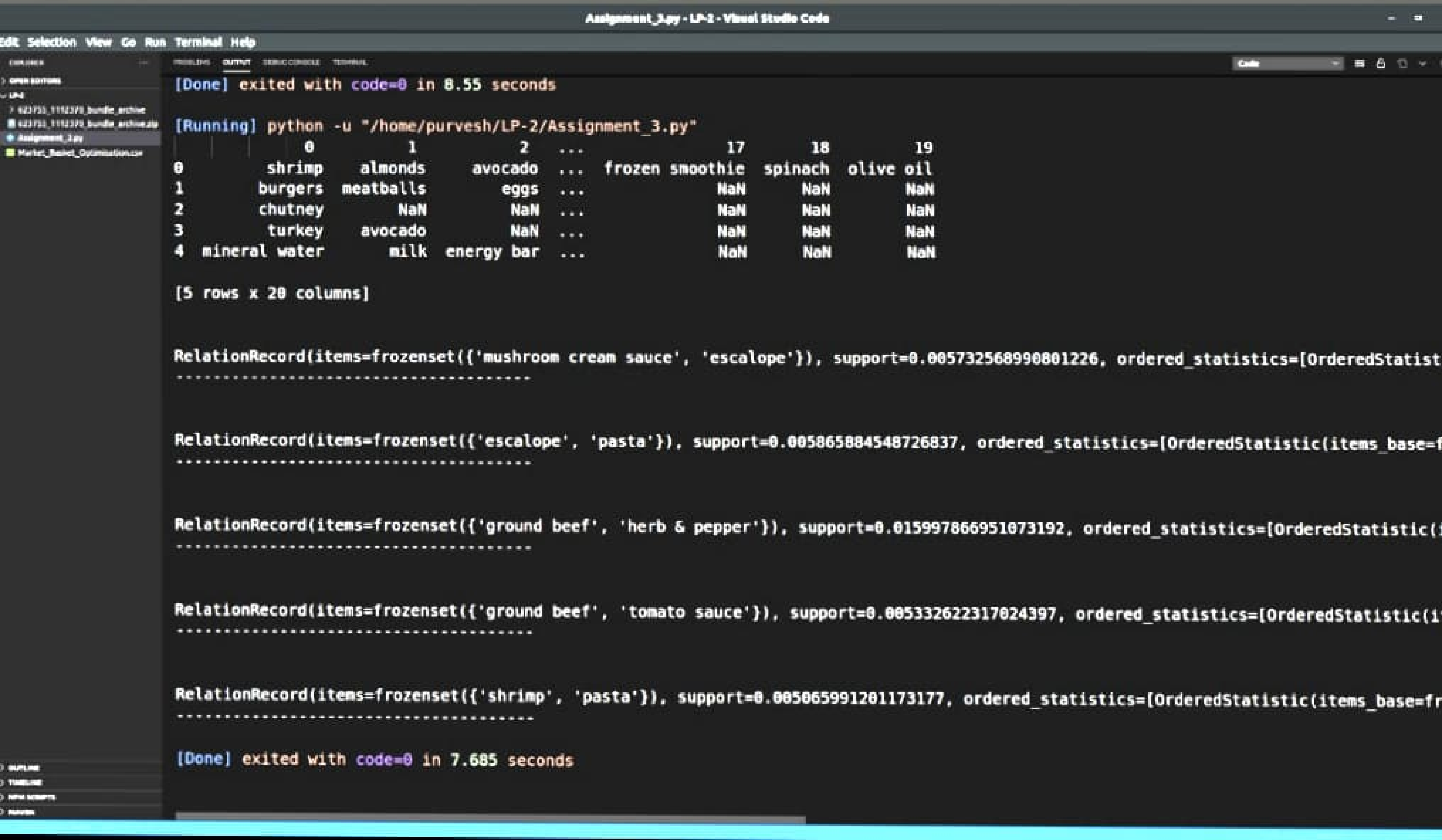
Limitations:

1. Computationally expensive: Even though the apriori algorithm reduces the number of candidate itemsets to consider this number could still be huge when store inventories are large or when the support threshold is low. However, using ~~both~~ hash tables, we can sort candidate itemsets more efficiently.

2. Spurious associations: Analysis of large inventories would involve more itemset configurations and the support threshold, might have to be lowered to detect certain associations. However lowering the support threshold might also increase the number of Spurious associations detected.

Conclusion:

We have successfully applied a-priori algorithm to find frequently Occuring items from given data & generated strong association rules using support & confidence thresholds.



Assignment_3.py - LP-2 - Visual Studio Code
Edit Selection View Go Run Terminal Help
Assignment_3.py
Market_Basket_Optimisation.csv
OUTPUT
Code

```
[Done] exited with code=0 in 8.55 seconds

[Running] python -u "/home/purvesh/LP-2/Assignment_3.py"
              0          1           2  ...              17       18         19
0        shrimp    almonds     avocado  ...  frozen smoothie  spinach  olive oil
1       burgers  meatballs        eggs  ...              NaN      NaN        NaN
2       chutney        NaN         NaN  ...              NaN      NaN        NaN
3        turkey    avocado         NaN  ...              NaN      NaN        NaN
4  mineral water      milk  energy bar  ...              NaN      NaN        NaN

[5 rows x 20 columns]


RelationRecord(items=frozenset({'mushroom cream sauce', 'escalope'}), support=0.005732568990801226, ordered_statistics=[OrderedStatist
----------------------------------------


RelationRecord(items=frozenset({'escalope', 'pasta'}), support=0.005865884548726837, ordered_statistics=[OrderedStatistic(items_base=f
----------------------------------------


RelationRecord(items=frozenset({'ground beef', 'herb & pepper'}), support=0.015997866951073192, ordered_statistics=[OrderedStatistic(i
----------------------------------------


RelationRecord(items=frozenset({'ground beef', 'tomato sauce'}), support=0.005332622317024397, ordered_statistics=[OrderedStatistic(i
----------------------------------------


RelationRecord(items=frozenset({'shrimp', 'pasta'}), support=0.005065991201173177, ordered_statistics=[OrderedStatistic(items_base=fr
----------------------------------------

[Done] exited with code=0 in 7.685 seconds
```

```python
import numpy as np
import pandas as pd
from apyori import apriori

dataset = pd.read_csv('Market_Basket_Optimisation.csv', header = None)

print(dataset.head())

transactions = []
for i in range(0, 7501):
    transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

rule_list = apriori(transactions, min_support = 0.003, min_confidence = 0.3, min_lift = 3, min_length = 2)

results = list(rule_list)
for i in results[0:5]:
    print('\n')
    print(i)
    print('--------------------------------------')
```