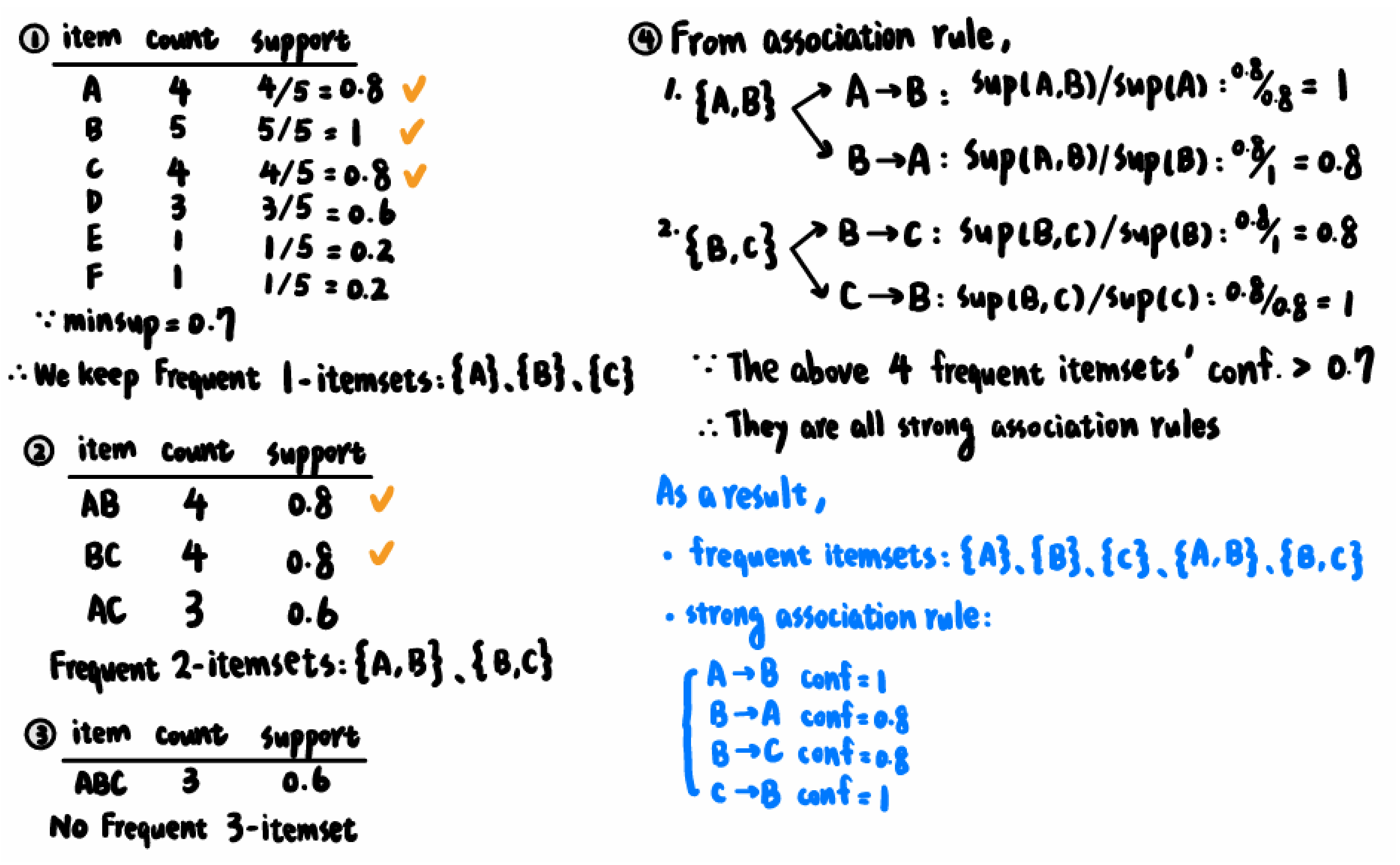
A1 FP growth

111511245 朱昱安 2025.03.20

Part I.



①

item	count	support	
A	4	4/5 = 0.8	✓
B	5	5/5 = 1	✓
C	4	4/5 = 0.8	✓
D	3	3/5 = 0.6	
E	1	1/5 = 0.2	
F	1	1/5 = 0.2	

∵ minsup = 0.7

∴ We keep Frequent 1-itemsets: {A}、{B}、{C}

②

item	count	support	
AB	4	0.8	✓
BC	4	0.8	✓
AC	3	0.6	

Frequent 2-itemsets: {A,B}、{B,C}

③

item	count	support
ABC	3	0.6

No Frequent 3-itemset

④ From association rule,

1. {A,B}
   - A→B : sup(A,B)/sup(A) : $0.8/0.8 = 1$
   - B→A : sup(A,B)/sup(B) : $0.8/1 = 0.8$
2. {B,C}
   - B→C : sup(B,C)/sup(B) : $0.8/1 = 0.8$
   - C→B : sup(B,C)/sup(C) : $0.8/0.8 = 1$

∵ The above 4 frequent itemsets' conf. > 0.7

∴ They are all strong association rules

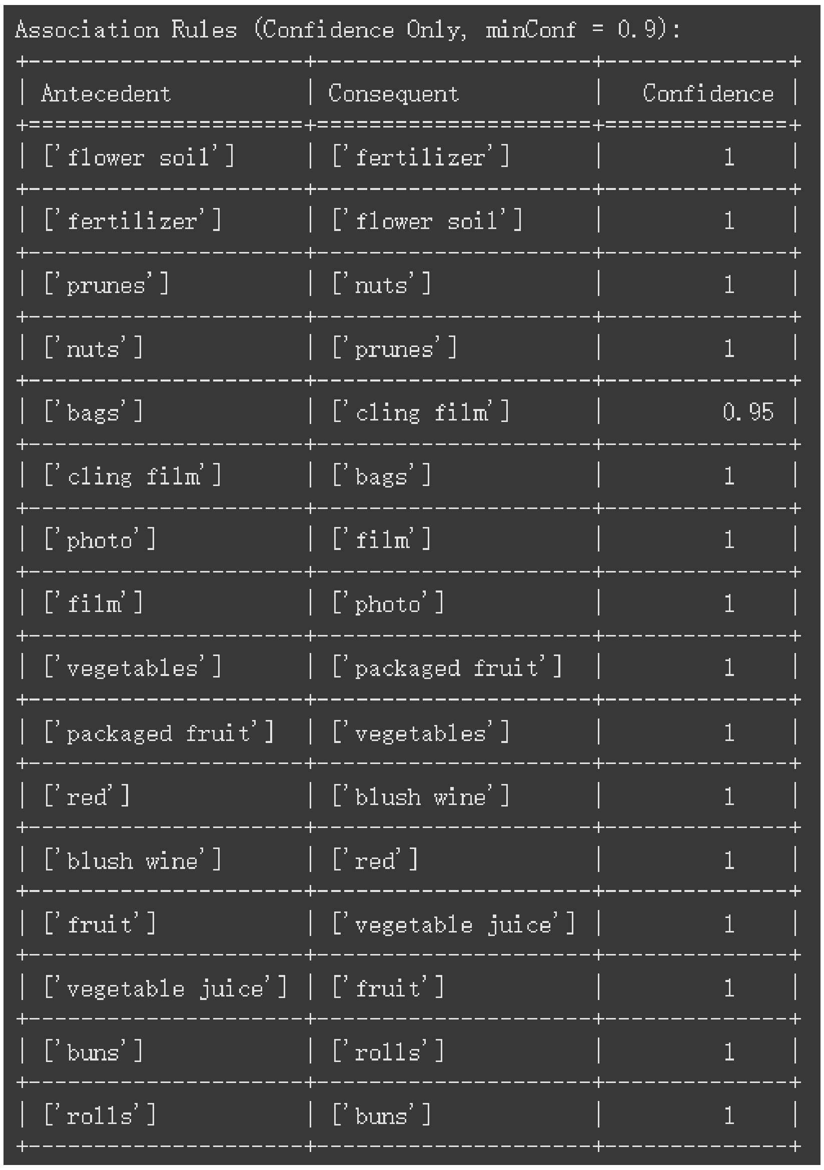
As a result,

- frequent itemsets: {A}、{B}、{C}、{A,B}、{B,C}
- strong association rule:
  - A→B conf = 1
  - B→A conf = 0.8
  - B→C conf = 0.8
  - C→B conf = 1

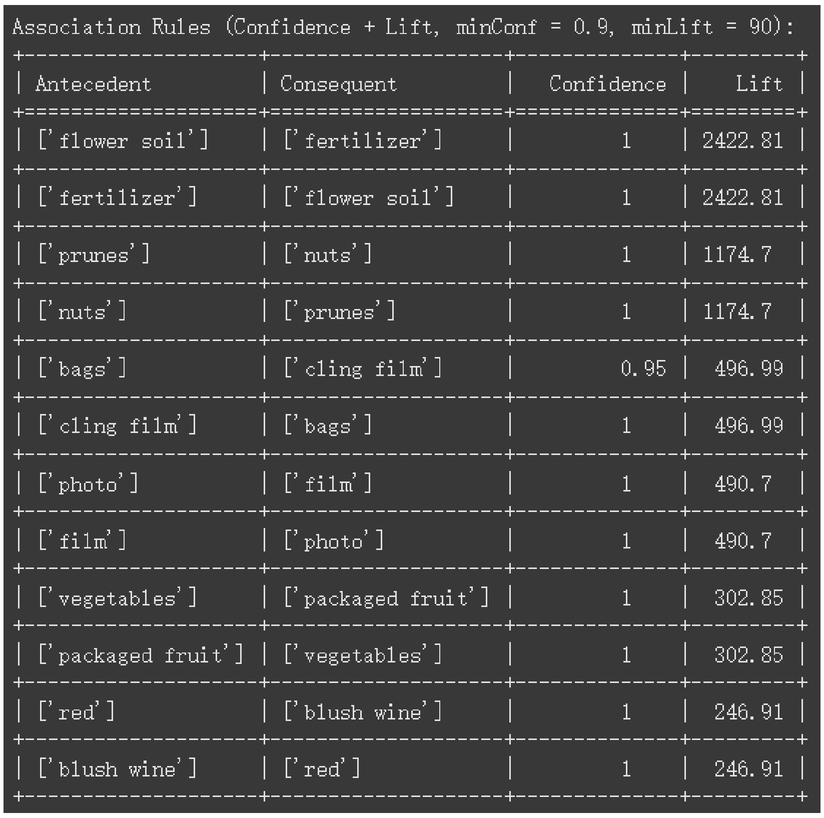
Part II.

1 & 2、

The following two pictures are the results of these two questions. The upper one is the association rules can be identified in the given dataset when using a minimum support of 0.01% and a minimum confidence of 90%, and the other one is the association rules can be identified in the given dataset when using a minimum support of 0.01% and the minimum lift value of 90.



```
Association Rules (Confidence Only, minConf = 0.9):
+---------------------+---------------------+--------------+
| Antecedent          | Consequent          |   Confidence |
+=====================+=====================+==============+
| ['flower soil']     | ['fertilizer']      |         1    |
+---------------------+---------------------+--------------+
| ['fertilizer']      | ['flower soil']     |         1    |
+---------------------+---------------------+--------------+
| ['prunes']          | ['nuts']            |         1    |
+---------------------+---------------------+--------------+
| ['nuts']            | ['prunes']          |         1    |
+---------------------+---------------------+--------------+
| ['bags']            | ['cling film']      |         0.95 |
+---------------------+---------------------+--------------+
| ['cling film']      | ['bags']            |         1    |
+---------------------+---------------------+--------------+
| ['photo']           | ['film']            |         1    |
+---------------------+---------------------+--------------+
| ['film']            | ['photo']           |         1    |
+---------------------+---------------------+--------------+
| ['vegetables']      | ['packaged fruit']  |         1    |
+---------------------+---------------------+--------------+
| ['packaged fruit']  | ['vegetables']      |         1    |
+---------------------+---------------------+--------------+
| ['red']             | ['blush wine']      |         1    |
+---------------------+---------------------+--------------+
| ['blush wine']      | ['red']             |         1    |
+---------------------+---------------------+--------------+
| ['fruit']           | ['vegetable juice'] |         1    |
+---------------------+---------------------+--------------+
| ['vegetable juice'] | ['fruit']           |         1    |
+---------------------+---------------------+--------------+
| ['buns']            | ['rolls']           |         1    |
+---------------------+---------------------+--------------+
| ['rolls']           | ['buns']            |         1    |
+---------------------+---------------------+--------------+
```

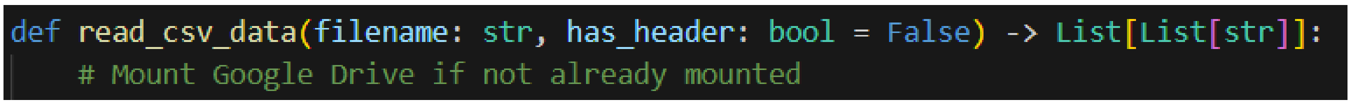


```
Association Rules (Confidence + Lift, minConf = 0.9, minLift = 90):
+--------------------+--------------------+--------------+---------+
| Antecedent         | Consequent         |   Confidence |    Lift |
+====================+====================+==============+=========+
| ['flower soil']    | ['fertilizer']     |         1    | 2422.81 |
+--------------------+--------------------+--------------+---------+
| ['fertilizer']     | ['flower soil']    |         1    | 2422.81 |
+--------------------+--------------------+--------------+---------+
| ['prunes']         | ['nuts']           |         1    | 1174.7  |
+--------------------+--------------------+--------------+---------+
| ['nuts']           | ['prunes']         |         1    | 1174.7  |
+--------------------+--------------------+--------------+---------+
| ['bags']           | ['cling film']     |         0.95 |  496.99 |
+--------------------+--------------------+--------------+---------+
| ['cling film']     | ['bags']           |         1    |  496.99 |
+--------------------+--------------------+--------------+---------+
| ['photo']          | ['film']           |         1    |  490.7  |
+--------------------+--------------------+--------------+---------+
| ['film']           | ['photo']          |         1    |  490.7  |
+--------------------+--------------------+--------------+---------+
| ['vegetables']     | ['packaged fruit'] |         1    |  302.85 |
+--------------------+--------------------+--------------+---------+
| ['packaged fruit'] | ['vegetables']     |         1    |  302.85 |
+--------------------+--------------------+--------------+---------+
| ['red']            | ['blush wine']     |         1    |  246.91 |
+--------------------+--------------------+--------------+---------+
| ['blush wine']     | ['red']            |         1    |  246.91 |
+--------------------+--------------------+--------------+---------+
```

3、

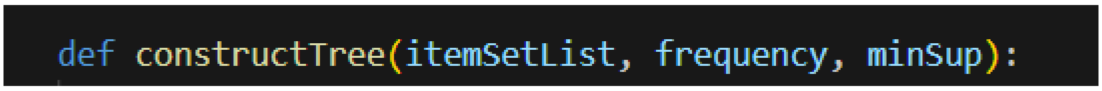
I’ve attached my main modifications in my ipynb file, and I will explain these modifications here again.

First, I add a function named *read_csv_data* to read csv from Google Drive, so I can get the needed file. Besides, I add a function to determine whether there is a header in the file, and I can also know that whether there are some problems with the input file.



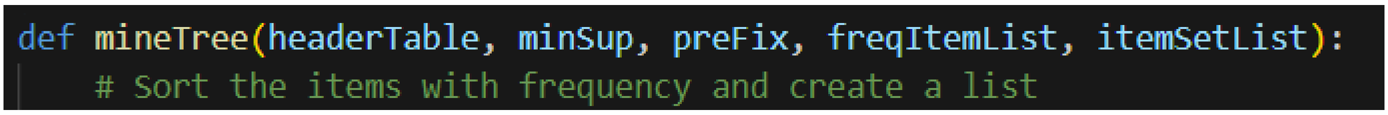
```
def read_csv_data(filename: str, has_header: bool = False) -> List[List[str]]:
    # Mount Google Drive if not already mounted
```

Secondly, I modified *constructTree*. Since the original header Table was sorted by frequency in ascending order (reverse=False), but I have been modified to sort in descending order (reverse=True). This change ensures that high-frequency items are prioritized during the construction process. The *constructTree* function builds the foundational data structure for the FP-growth algorithm, transforming raw transactions into a tree and table that enable efficient frequent pattern mining. It filters out infrequent items, sorts them for optimal structure, and constructs a tree that captures co-occurrence relationships.



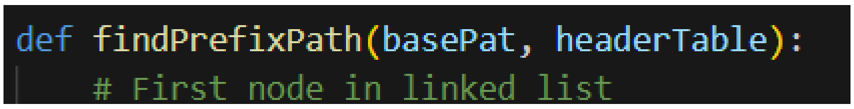
```
def constructTree(itemSetList, frequency, minSup):
```

Then, I modified the *mineTree* function, which is a recursive component of the FP-growth algorithm designed to mine frequent itemsets from an FP-tree. It explores the tree by generating conditional pattern bases and recursively constructing conditional FP-trees to extract all frequent patterns that meet a minimum support threshold (minSup). My modification explicitly verifies each newFreqSet’s support against itemSetList, ensuring only truly frequent itemsets are included. More robust, as it guards against potential inconsistencies between the FP-tree and the dataset. In conclusion , it is slower but more precisely guarantee accuracy by cross-checking with itemSetList, making the resulting frequent itemsets and subsequent rules more reliable. It produces more reliable rules, potentially fewer in number.



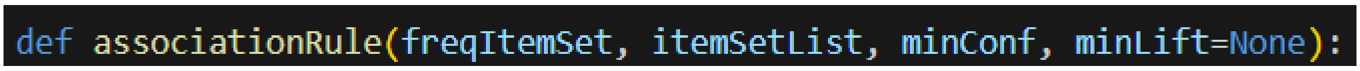
```
def mineTree(headerTable, minSup, preFix, freqItemList, itemSetList):
    # Sort the items with frequency and create a list
```

Next, *FindPrefixPath* is also modified because it generates shorter prefix paths, excluding the immediate parent of the base pattern in some interpretations, depending on how it’s used downstream. It aligns with a stricter interpretation of FP-growth, where the conditional pattern base excludes the base pattern and its immediate context more explicitly from the start. This matches the corrected intent, but there’s no functional impact on rules



```
def findPrefixPath(basePat, headerTable):
    # First node in linked list
```

Finally, I modified the *associationRule* because it can ensure only frequent itemsets are processed, with robust rule generation, producing accurate and high-quality rules. Also, we can get fewer itemsets and rules due to stricter filtering, but all are valid and optionally refined by lift. In conclusion, it is lower but more precise. Adding lift, identifying stronger associations.



```python
def associationRule(freqItemSet, itemSetList, minConf, minLift=None):
```

Besides, I adjust the *fpgrowth* to be *fpgrowth_from_csv*. Since we need to read from csv.

4、

In my code, the anti-monotonicity property is maintained and impacts the search space as follows:

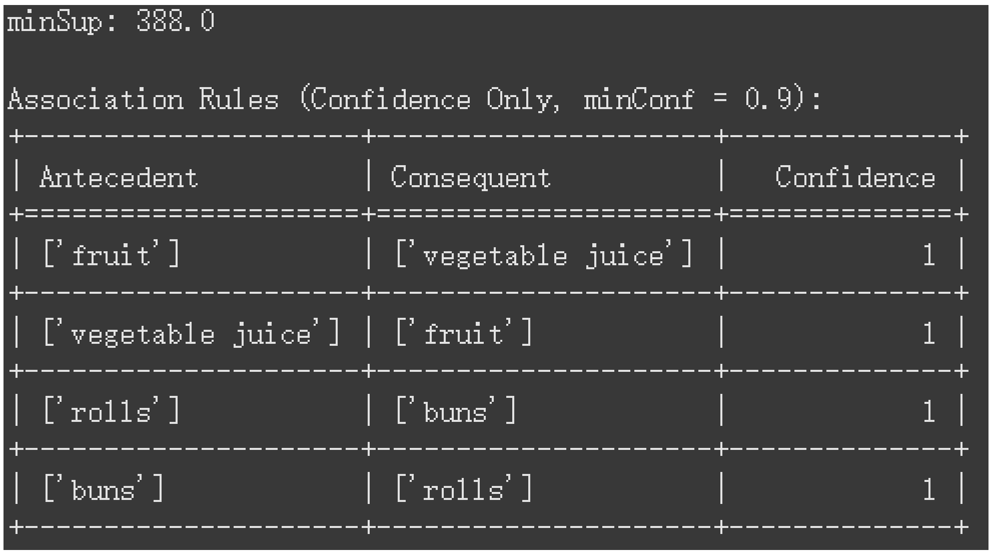
- Enforced in *constructTree* (responsible for initial filtering), reinforced in *mineTree* (support checks and conditional trees).
- Prunes infrequent items and their supersets at every stage, from tree construction to recursive mining.
- Reduces the search space from exponential (2^n) to a manageable subset of frequent patterns, making FP-growth efficient while ensuring correctness for downstream association rule generation.

This property is what allows my implementation to scale effectively, especially with large datasets like those read from CSV files in *fpgrowth_from_csv*. The *associationRule* function only processes *freqItemSet* from *mineTree*. Since anti-monotonicity ensures *freqItemSet* contains only frequent itemsets, the rule generation focuses on valid candidates, further leveraging the property implicitly. The integration ensures that anti-monotonicity propagates from the initial tree construction through recursive mining, consistently reducing the space to only frequent patterns.

5、

I've tried two kind of changes, and I focus mainly on *minSup* and *minConf*.

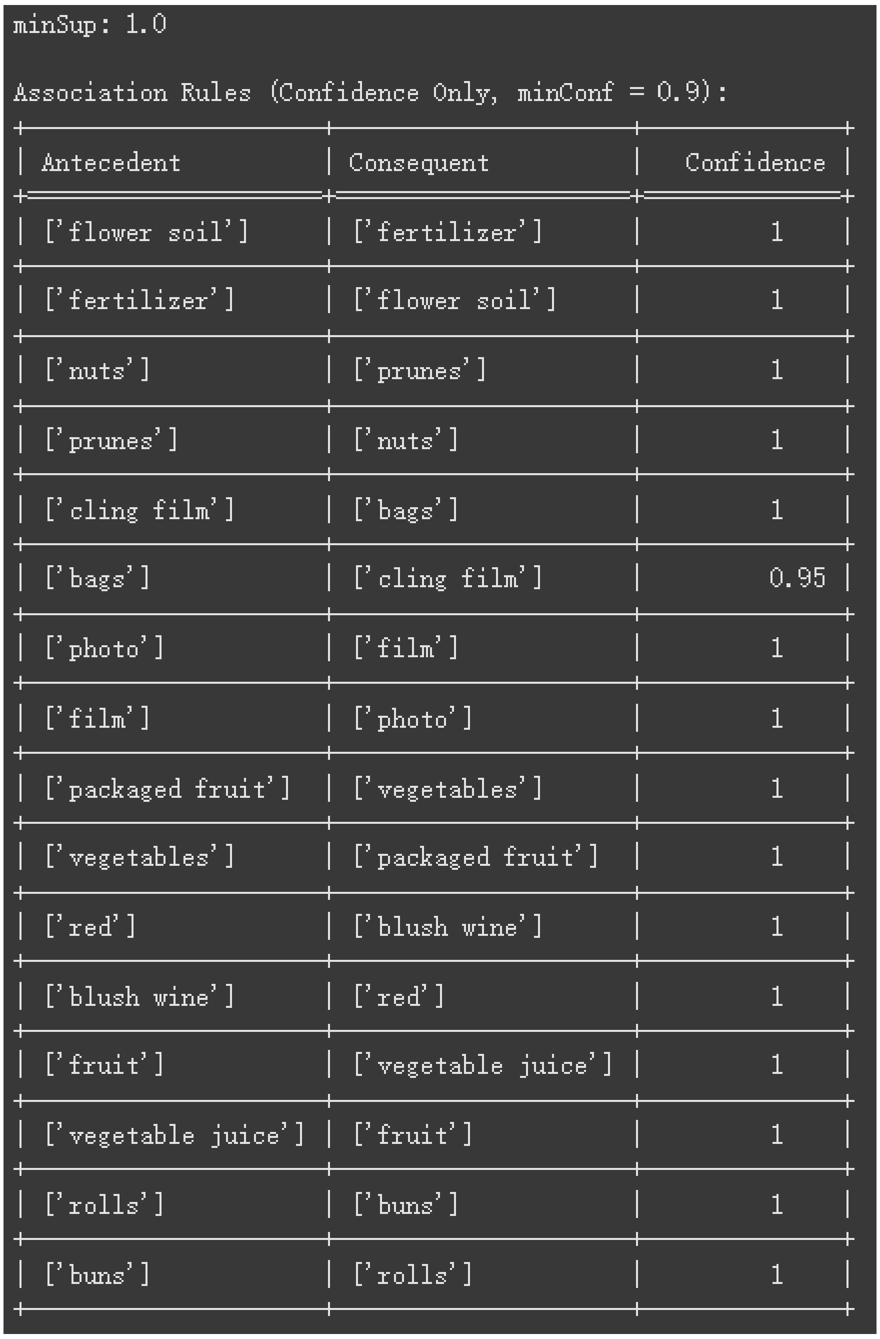
First, if we lower the *minSup*, *associationRules* will increase since we can have more itemsets kept. Otherwise, it will decrease since we drop more itemsets. The result shows that by a direct observation about the relationship between the minimum support threshold (*minSup*) and the number of association rules generated in the FP-growth algorithm. When we lower *minSup*, more itemsets meet the support threshold, increasing the size of *freqItemSet* (the output of *mineTree*). Otherwise, more frequent itemsets provide more candidates for *associationRule*, typically leading to an increase in the number of association rules.



```
minSup: 388.0

Association Rules (Confidence Only, minConf = 0.9):
+----------------------+----------------------+---------------+
| Antecedent           | Consequent           |   Confidence  |
+======================+======================+===============+
| ['fruit']            | ['vegetable juice']  |             1 |
+----------------------+----------------------+---------------+
| ['vegetable juice']  | ['fruit']            |             1 |
+----------------------+----------------------+---------------+
| ['rolls']            | ['buns']             |             1 |
+----------------------+----------------------+---------------+
| ['buns']             | ['rolls']            |             1 |
+----------------------+----------------------+---------------+
```

(*minSup* = 0.01)

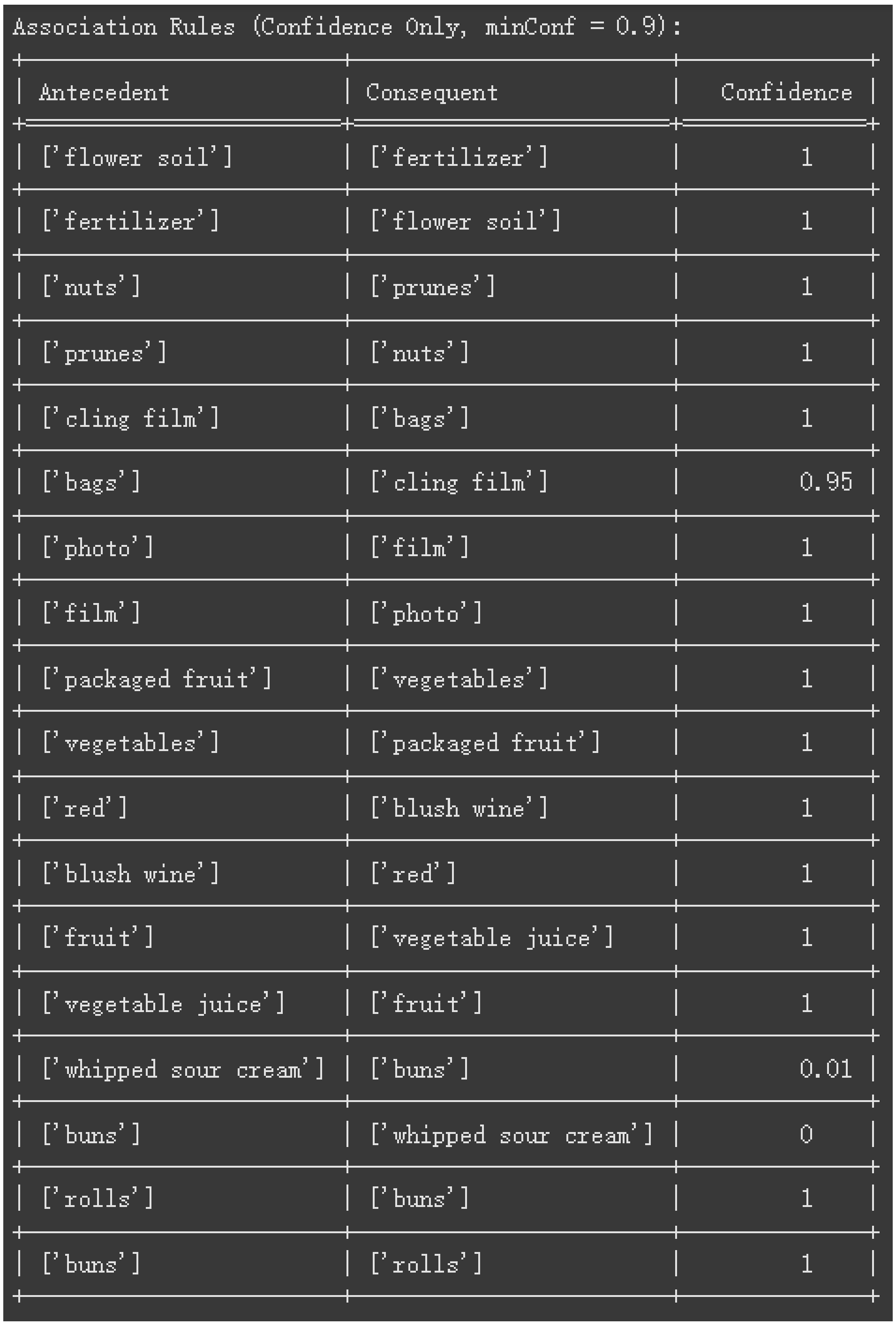


```
minSup: 1.0

Association Rules (Confidence Only, minConf = 0.9):
+----------------------+----------------------+--------------+
| Antecedent           | Consequent           |  Confidence  |
+======================+======================+==============+
| ['flower soil']      | ['fertilizer']       |       1      |
+----------------------+----------------------+--------------+
| ['fertilizer']       | ['flower soil']      |       1      |
+----------------------+----------------------+--------------+
| ['nuts']             | ['prunes']           |       1      |
+----------------------+----------------------+--------------+
| ['prunes']           | ['nuts']             |       1      |
+----------------------+----------------------+--------------+
| ['cling film']       | ['bags']             |       1      |
+----------------------+----------------------+--------------+
| ['bags']             | ['cling film']       |         0.95 |
+----------------------+----------------------+--------------+
| ['photo']            | ['film']             |       1      |
+----------------------+----------------------+--------------+
| ['film']             | ['photo']            |       1      |
+----------------------+----------------------+--------------+
| ['packaged fruit']   | ['vegetables']       |       1      |
+----------------------+----------------------+--------------+
| ['vegetables']       | ['packaged fruit']   |       1      |
+----------------------+----------------------+--------------+
| ['red']              | ['blush wine']       |       1      |
+----------------------+----------------------+--------------+
| ['blush wine']       | ['red']              |       1      |
+----------------------+----------------------+--------------+
| ['fruit']            | ['vegetable juice']  |       1      |
+----------------------+----------------------+--------------+
| ['vegetable juice']  | ['fruit']            |       1      |
+----------------------+----------------------+--------------+
| ['rolls']            | ['buns']             |       1      |
+----------------------+----------------------+--------------+
| ['buns']             | ['rolls']            |       1      |
+----------------------+----------------------+--------------+
```

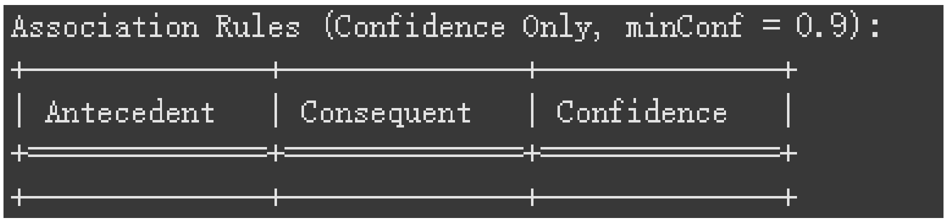
(*minSup* = 0.000001)

Secondly, if we adjust the minConf to be very high or low, there are some interesting results：



```
Association Rules (Confidence Only, minConf = 0.9):
+-------------------------+-------------------------+------------+
| Antecedent              | Consequent              | Confidence |
+=========================+=========================+============+
| ['flower soil']         | ['fertilizer']          |     1      |
+-------------------------+-------------------------+------------+
| ['fertilizer']          | ['flower soil']         |     1      |
+-------------------------+-------------------------+------------+
| ['nuts']                | ['prunes']              |     1      |
+-------------------------+-------------------------+------------+
| ['prunes']              | ['nuts']                |     1      |
+-------------------------+-------------------------+------------+
| ['cling film']          | ['bags']                |     1      |
+-------------------------+-------------------------+------------+
| ['bags']                | ['cling film']          |       0.95 |
+-------------------------+-------------------------+------------+
| ['photo']               | ['film']                |     1      |
+-------------------------+-------------------------+------------+
| ['film']                | ['photo']               |     1      |
+-------------------------+-------------------------+------------+
| ['packaged fruit']      | ['vegetables']          |     1      |
+-------------------------+-------------------------+------------+
| ['vegetables']          | ['packaged fruit']      |     1      |
+-------------------------+-------------------------+------------+
| ['red']                 | ['blush wine']          |     1      |
+-------------------------+-------------------------+------------+
| ['blush wine']          | ['red']                 |     1      |
+-------------------------+-------------------------+------------+
| ['fruit']               | ['vegetable juice']     |     1      |
+-------------------------+-------------------------+------------+
| ['vegetable juice']     | ['fruit']               |     1      |
+-------------------------+-------------------------+------------+
| ['whipped sour cream']  | ['buns']                |       0.01 |
+-------------------------+-------------------------+------------+
| ['buns']                | ['whipped sour cream']  |     0      |
+-------------------------+-------------------------+------------+
| ['rolls']               | ['buns']                |     1      |
+-------------------------+-------------------------+------------+
| ['buns']                | ['rolls']               |     1      |
+-------------------------+-------------------------+------------+
```

(*minConf* = 0.0001)



```
Association Rules (Confidence Only, minConf = 0.9):
+--------------+--------------+--------------+
| Antecedent   | Consequent   | Confidence   |
+==============+==============+==============+
+--------------+--------------+--------------+
```

(*minConf* = 100)

When we have higher *minConf*, there will be fewer rules meet the threshold because only those with strong predictive power (high confidence) are retained, which have smaller *rules_conf_only* and *rules_conf_lift*, focusing on highly reliable rules. Otherwise, when we have lower *minConf*, there are more rules qualified, including those with weaker relationships between antecedent and consequent. As a result, larger *rules_conf_only* and *rules_conf_lift*, capturing more associations, including less certain ones.

In conclusion, if we have higher *minConf*, rules are more trustworthy (e.g., A => B is more likely true), but some valid weaker rules might be missed. As for lower *minConf*, including more speculative rules, increasing noise but potentially revealing subtle patterns. *minConf* operates on the output of *mineTree* (*freqItemSet*), so its effect is independent of *minSup's* pruning but depends on the itemsets available. A high *minSup* might limit the itemsets, indirectly reducing rule candidates regardless of *minConf*.

Finally , there are three things that I found it interesting :

- Causal Relationship：It proves that *minSup* has a direct causal effect on the number of association rules via the number of frequent itemsets, modulated by *minConf*.
- Algorithm Design：It validates the FP-growth algorithm's reliance on anti-monotonicity to control output size, showing how *minSup* acts as a tuning knob for granularity.
- Practical Implication：It demonstrates that adjusting *minSup* is a primary lever for managing the trade-off between discovering more patterns (low *minSup*) and focusing on dominant patterns (high *minSup*), with *minConf* fine-tuning rule selection.

In essence, it proves the tunability and correctness of the above implementation in leveraging support thresholds to influence rule generation, a hallmark of efficient frequent pattern mining.