Frequent Itemset Mining Jiyun Xu

Overview:

1. Data cleaning

- I get rid of both 'fnlwgt' and 'education-num' attributes because they have no contribution for any purposes of this project other than making troubles.
- '?'-Unknown data are ignored and deleted from the database because the total number of unknown data is non-trivial and can occur in the result of frequent itemset mining.

2. Binning method

Because 'age' describes continuous data, if we do not apply binning method to them, they can hardly appear in frequent itemset. After carefully comparison, I choose to apply equal-width binning method which has a gap of 10.

```
xujiyundeMBP:project1 xujiyun$ python3 binCheck.py
{'11-20': 2410, '21-30': 8162, '31-40': 8546, '41-50': 6983, '51-60': 4128, '61-70': 1792, '71-80': 441, '81-90': 99,
```

As shown above, for example, in the whole database, there are 8162 records contain people within 21 to 30 years old which increase the probability of occurrence of 'age' catalog during frequent itemset mining when I want to choose a relatively large minimum support count. (This is where the recommended minimum support comes from when running my code.)

3. Evaluation of my code:

I'm confident that my code gets to the right answer for frequent itemset mining since when I choose different methods to analysis with the same minimum support, it always gives the same answer and the calculation time is reasonable. It also prints out the total number of frequent itemsets it finds and the calculation time it uses.

Please check 'README' and have fun with it!

Experiments and comparison:

1. Giant Comparison Form(With 32561 records in database):

min_sup_count	1000	2000	4000	8000	12000
Total Number of Frequent Itemsets	20478	6638	1895	391	140
Apriori	359.2s	59.4s	16.3s	4.1s	1.6s
Apriori-Improved	299.8s	42.6s	9.4s	2.2s	0.9s
Efficiency Improvement (towards Apriori)	16.5%	28.3%	42.3%	46.3%	43.8%
FP-Growth	21.5s	9.8s	2.8s	0.5s	0.2s
Efficiency Improvement (towards Apriori-Improved)	92.8%	77.0%	70.2%	77.3%	77.8%

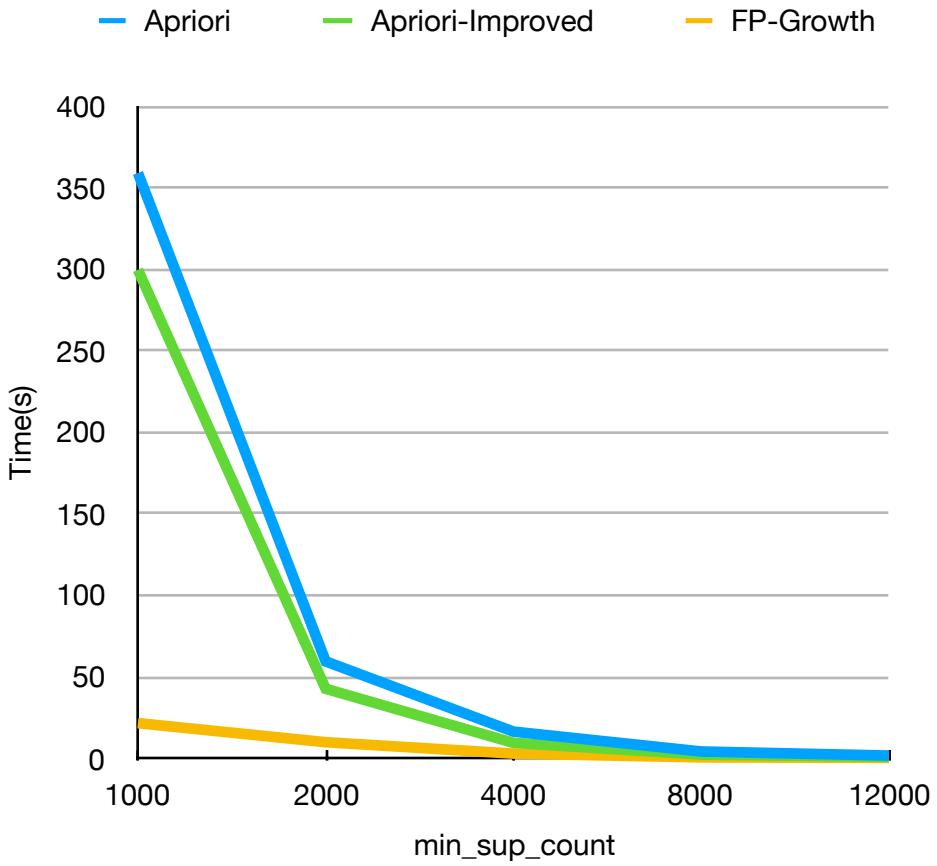
1. Apriori and Apriori-Improved:

After data cleaning and binning, I find there are many duplicate records in database. Because Apriori method is going to scan database plenties of times, the reduction of size of database can significantly improve the efficiency of Apriori method. So, the database now becomes a dictionary where each transaction is treated as a frozenset which becomes a key of the dictionary and the value of it would be how many times each records appears in the database. After doing so, the number of records of database is reduced from **32561** to **19008**.
As we can see in the comparison form, the improvement method is doing great with a relatively big min_sup_count which yield an efficiency improvement around **45%** , however it is considered less helpful as the min_sup_count becomes relatively small and the efficiency improvement is shrinking to around **20%**.

2. FP-Growth and Apriori-Improved:

I apply the improvement method to both Apriori-Improvement and FP-Growth, so it is more reasonable to compare the efficiency of FP-Growth with the improved one rather than the original one. We can yield a stable **77%** percent efficiency improvement when the min_sup_count is relatively big and when it becomes smaller and smaller, FP-growth can do even better, yielding a efficiency improvement around **93%**.

3. Time Consuming Chart:



It is obvious that FP-Growth is doing a far better job than Apriori and the variant of Apriori, thus, it is worth putting some efforts to develop a FP-Growth algorithm to operate Frequent Itemset Mining with a relatively large database and relatively small minimum support count.
FP-Growth is doing better because it only needs to scan the database two times, however, Apriori algorithm needs to do it during every iteration. Also, FP-Growth

algorithm doesn't contain candidates generation process which makes it much more efficient.

Appendix:

The result running code could look like:

```
{'Female', '<=50K', '11-20', 'No-capital-gain'}
{'United-States', '<=50K', '11-20', 'No-capital-gain', 'Female'}
{'<=50K', '11-20', 'No-capital-gain', 'Female', 'No-capital-loss'}
{'Female', '<=50K', '11-20', 'United-States', 'No-capital-loss', 'No-capital-gain'}
{'Female', '<=50K', '11-20', 'Never-married'}
{'United-States', '<=50K', '11-20', 'Never-married', 'Female'}
{'<=50K', '11-20', 'Never-married', 'Female', 'No-capital-loss'}
{'Female', '<=50K', '11-20', 'United-States', 'No-capital-loss', 'Never-married'}
{'<=50K', '11-20', 'Never-married', 'No-capital-gain', 'Female'}
{'Female', '<=50K', '11-20', 'United-States', 'Never-married', 'No-capital-gain'}
{'Female', '<=50K', '11-20', 'No-capital-loss', 'Never-married', 'No-capital-gain'}
{'Female', '<=50K', '11-20', 'United-States', 'No-capital-loss', 'Never-married', 'No-capital-gain'}
20478 Frequent Itemsets Total
Calculated in 21.5s
```