Data Science

Assignment 1 Apriori

Prof. Sangwook Kim  
TAs. Jangwan Koo, Yeonchang Lee

2013011424  
Yedarm Seong  
Division of Computer Science and Engineering

1. Environment

OS: Windows 10  
Language: Python 3.5.2

2. Summary of algorithm

(1) Creating a Candidate Item Set

For each transaction in tran the DB:  
 For each candidate itemset can:  
 if can is a subset of tran:  
 increment the count of can  
 For each candidate itemset:  
 if the support meets the min, keep this item  
return frequent itemsets list

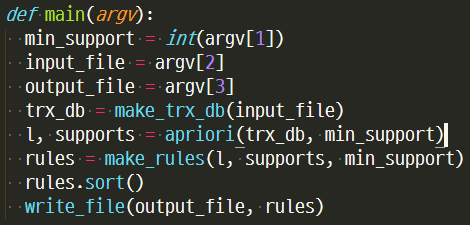
(2) Apply Apirori algorithm

While the number of items in the set is greater than 0:  
 Create a list of candidate itemsets of length k  
 Scan the dataset to see if each itemset is frequent  
 Save frequent itemsets to create itemsets of length k + 1

(3) Find Association rules  
For frequent sets with length 1 <= i <= length:  
 For frequent sets of length i < j <= length:  
 If any frequent set of length i is a subset of any frequent set of length j:  
 Find association rules and calculate support and confidence

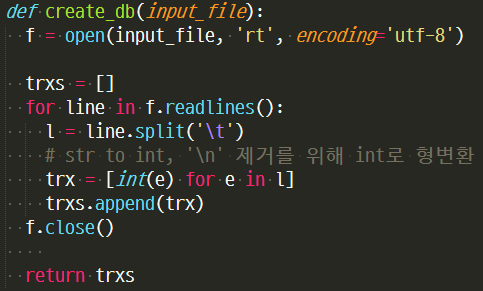
3. Detailed description of codes (for each function)

Main Function



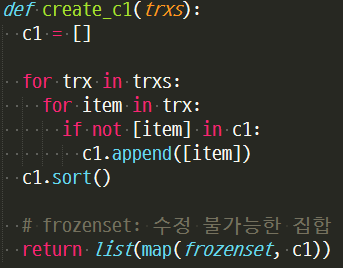
Get the minimum support, input file, and output file by parsing the arguments.  
Create a transaction database from the input file.  
We use a database and minimum support to create list of all frequent set and support dictionary of all frequent set.  
The dictionary key is a frequent set.  
Find association rules using frequent sets, support dictionary, and minimum support.  
Write rules to file for assignment type

A Functions that read a file and return a transaction list



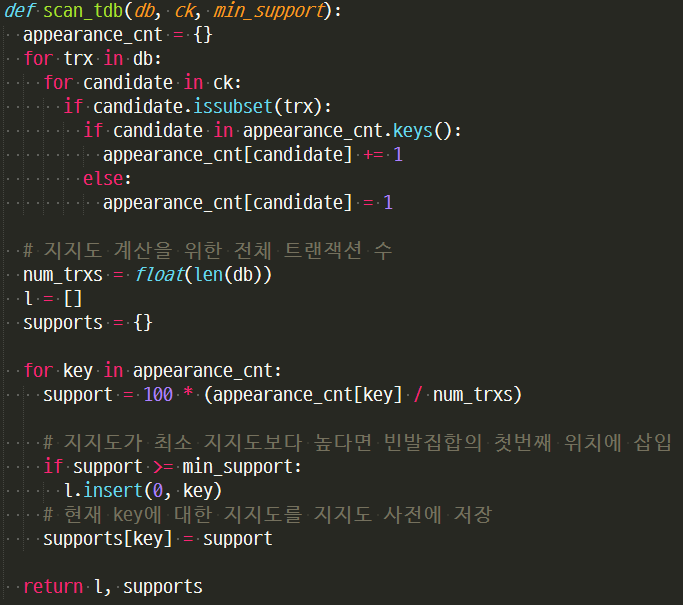
Reads and parses the file and returns it as a list.

A function that creates a candidate set of length 1.  
The reason for using frozenset is to use c1 as dictionary key.  
(List, set can not be key in python3)



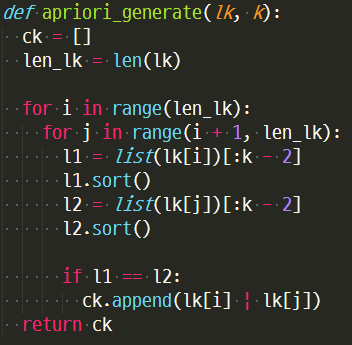
Repeats all transactions and generates a candidate set of length 1.

Create a frequent set and support dictionary with length k



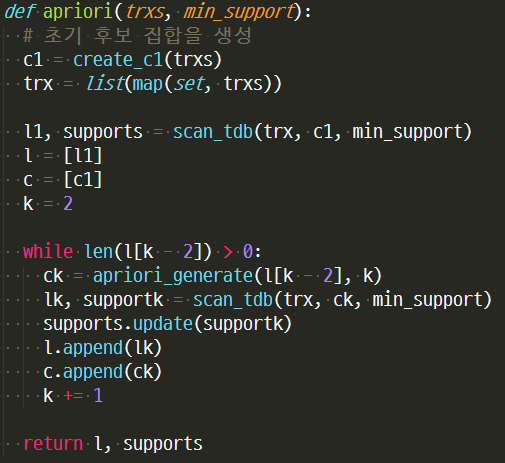
Each transaction and candidate set in the database is repeated.  
If the candidate set is a subset of the transaction, the number of occurrences of the candidate set is increased by one.  
We use the expression of the candidate set and the total number of transactions to generate a frequent set and support dictionary that is higher than the minimum support.

A function that creates a candidate set of length k.



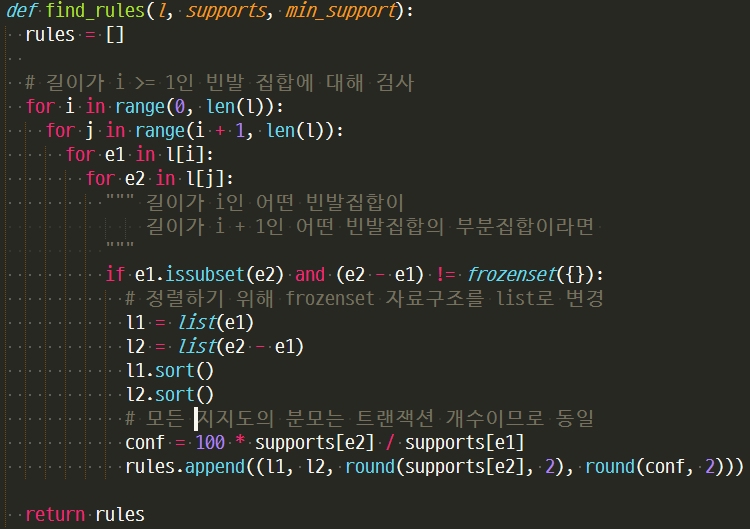
Generates a candidate set of length k + 1 with frequent sets of length k.

Apriori algorithm to generate a frequent set and support dictionary.



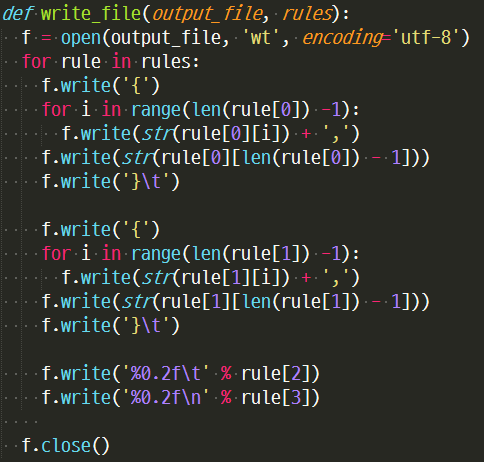
Generate C1 and L1 from the initial database, and use it to generate Ck and Lk of length k and calculate the support.  
Computes and returns all frequent sets and support dictionary.

A functions that find association rules



e1 (in frequent set with length i> = 1) and e2(in a frequent set with length j> i).  
If e1 is a subset of e2 and e2 - e1 is not an empty set, find association rules and calculate support and confidence

Writing association rule results to a file



Write the file according to the assignment type.

4. Instructions for compiling source codes at TA's computer (e.g. screenshot) (*Important!!*)

**Python does not need a compile process**

If you already install pyinstaller for python3, in the directory where the apriori.py file is located, type the following command:

Windows:  
pyinstaller -F apriori.py

Ubuntu:  
pyinstaller -F apriori.py

Then the dist folder is created in the directory where the command is executed, and there is ‘apriori.exe’ file in it.(For linux, the ‘apriori’ file)

Usage: apriori.exe [minimum support] [input file] [output file]

After using pyinstaller, there is an exe file in the dist folder:  
ex) dist/apriori.exe 5 input.txt output.txt

If you move the exe file to the same location as input.txt, you can use it like this:  
ex) apriori.exe 5 input.txt ouput.txt

5. Any other specification of implementation and testing

Another execution method:  
Running as a Python file

ex) apriori.py [minimum support] [input file] [output file]