Assignment 2

Mining association rules (Python ver)

The objectives of this assignment are:

1) To understand the impact of the parameters on the number of frequent itemsets and rules extracted from the data.
2) To generate an interpretation of the rules and filter them, to identify the most relevant ones.

To this aim, we will use data from the AOL (America On Line) search engine, as described in the paper *A Picture of Search*, presented by G. Pass, A. Chowdhury, and C. Torgeson at the First International Conference on Scalable Information Systems, Hong Kong, June, 2006. In particular, for a subset of anonymous users in the original data we have listed the queries they submitted to the search engine for a period of three months, in 2006.

Preliminary of Language/Software: You can choose Pycharm or Jupiter Notebook (or others that you have been familiar with) as your Python IDE. To install packages, use command `pip install [package name]` in the Windows' or Mac's terminal.

TASK 1: Warm Up) Read the www.csv data via python and preprocess it. This file can be found on Studium. Each row represents a search query. The first field is the anonymous user id, the second field contains the search query, and each of the following columns corresponds to a frequent keyword and takes the values **t**(rue) or **f**(alse) depending on the presence of that keyword in the search query. HINT: Use `import csv` and `csv.reader(open('www.csv', mode ='r'))` to read the csv file. You may need to use function `split` twice to extract the second column (i.e. the search query) as a list of list of strings. Besides, count the number of records and the number of attributes (excluding User and Query).

TASK 2: Generation of Frequent Itemsets) As you know, you can split the Association Rule Mining task into two parts. First, you should find the frequent itemsets. HINT: You can choose to implement your own code or utilize relevant packages (like `pyfpgrowth:` https://pypi.org/project/pyfpgrowth/, and the function `find_frequent_patterns)`

2.1) Set min-support so that an itemset must be present in at least 100 search queries to be considered frequent.
[Reminder: the *support* of an itemset is the fraction of records containing the items in the itemset (in this case, the keywords), and *min-support* is the minimum value of support to consider an itemset *frequent*]
2.2) Execute the process and examine the results.

TASK 3: Effect of Support) Execute the previous process varying the min-support (by setting `a for loop)`. Choose different values in the range [0.001, 0.01] and for each of these check:

3.1) the number of the found frequent itemsets, and
3.2) the maximum size of the found frequent itemsets.

3.3) From this analysis, choose a value for min-support leading to the discovery of around 150 frequent itemsets and use this value in the next tasks.

TASK 4: Generating the Rules) Taking the frequent itemsets as input, generate the association rules. For now, set a minimum confidence of 80%. HINT: You can choose to implement your own code or utilize relevant packages (like `pyfpgrowth`, and its function `generate_association_rules`)

4.1) Execute the process and inspect the result.
4.2) Find, among the identified rules, an example of a high-confidence rule X -> Y where Y -> X does not have high confidence. Why is the confidence of Y -> X lower?

TASK 5: Impact of Confidence) Execute again the previous process varying the min-confidence (by setting `a for loop`). Choose 5 different values of min-confidence in the range [0.1, 1] and for each of these indicate the number of generated rules. See how the number of rules changes.

TASK 6: Rule Interpretation) In this activity, you should test different evaluation functions with respect to their ability to highlight the *relevant* rules. Therefore, you are requested to provide your own subjective judgment on the identified rules and compare this against the values assigned by the algorithm:

6.1) Execute the process with a very low support, to extract a larger number of rules.
6.2) Choose up to 3 most **interesting** rules (notice that this is a subjective measure: you decide what is interesting). (Can you find 3?)
6.3) Choose up to 3 most **unexpected** rules (notice that this is also a subjective measure). (Can you find 3?)
6.4) For each of these 6 rules, provide their ranking according to confidence and lift (that is, at which position do the rules that you have selected appear, when rules are sorted by confidence/lift?).
6.5) Describe (if possible) the features characterizing the relevant (interesting and/or unexpected) rules, and update your process so that irrelevant rules are no longer identified (still preserving the interesting ones).
HINT: If you implement your own code, the filter(s) can be embedded in the code of **TASK 4**; if you use package (like `pyfpgrowth`), you can add a filter after the code of **TASK 4**.