Homework DMV

Introduction and rules

The goal of this homework is to analyze a real dataset using data mining techniques. Visualization will also be used both to progress in the analysis and to present some of the results.

The deadline for submitting your homework is **Sunday the 12th of November at 23:59** (no extension). Homework submitted after that date will not be reviewed, and the grade will be 0.

- The homework has to be submitted on Moodle a zip file. If the zip file is big and Moodle rejects it, provide a link to an online storage resource in a text file (the link should be available at least until 1st of March 2024).
- The homework must be done **alone**.
- The expected output is:
    - A report in PDF answering the different tasks.
    - The code written to perform the tasks.

We strongly recommend you to work in a "notebook" environment: either Jupyter + Python, or R markdown (available in RStudio for example). In this case, your PDF report can be an export of your notebook, and the code can be the notebook file (ipynb or Rmd).

If you choose to work with other tools (ex: code in Java), then the report has to detail precisely what you did, include pseudo-code of the pre-processing you did (do NOT copy paste hundreds of lines of Java in the report!!!), and contain appropriate figures. Such report can be generated either with LaTeX or with Word/LibreOffice.

Last, in your report we ask you to try to justify your choices and results with data visualizations whenever possible (be it asked by the question or not).

Dataset

The dataset that we will be analyzing is a real dataset of retail transactions from the company Instacart. They have provided this data in the context of a Kaggle challenge:

https://www.kaggle.com/c/instacart-market-basket-analysis

You can recover a dump of the data on the DMV course home page: (216 MB)

http://people.irisa.fr/Alexandre.Termier/dmv/instacart_basket_data.zip

login: *dmvstudent*

password: *patmin7102*

The task of interest for the challenge was to determine “reorders” of customers, a prediction task.

In this homework, we are not interested in prediction. Our goals will be:

- Learn how to prepare data for data mining tasks, with the help of visualization ;
- Understand how pattern mining algorithms behave on real data ;
- Try to discover new knowledge in the data with pattern mining and visualization techniques.

The data is described here : https://www.kaggle.com/c/instacart-market-basket-analysis/data

We are interested in the orders of customers. Each order is a transaction, i.e. a list of products, as we have seen in the course of frequent itemset mining.

Tasks

Task 0: warming up

You can find example notebooks for simple exploratory analysis:

- In Python: https://www.kaggle.com/sudalairajkumar/simple-exploration-notebook-instacart
- In R: https://www.kaggle.com/philippsp/exploratory-analysis-instacart

These notebooks are also copied in the zip file you received.

Your first task is to look at these notebooks in order to get a first understanding of the data: how is it organized, how to load it and make first simple analysis task. Choose any of the languages (Python, R or something else), and based on the provided notebooks, load the data and compute some statistics on the **training** dataset (`order_products_train.csv`): number of different orders, number of different products, average/min/max products per order.

Task 1: Frequent itemsets

The goal of this task is to analyze the orders of the **training** dataset with frequent itemset mining.

Question 1.1: We will start by using Apriori. You can find various implementations of Apriori:

- In Python: I provide you a simple notebook with a straightforward Apriori code (file `Apriori.ipynb`)
- In R: use the arules package
- In Java: an implementation of Apriori can be found in the SPMF library: http://www.philippe-fournier-viger.com/spmf/

Understand the input expected by the Apriori of your choice, and transform the training dataset in that format.

The report must contain an explanation of the transformation, as well as code (Python/R) or pseudo-code (other language).

Note: for those using R, in the arules package there are several options for input, but most of them fail on the Instacart data (too big). The solution is to use `read.transactions()`, this requires some work to put the data in the correct format…

Question 1.2: By a simple analysis of the data, indicate what is the **maximum** support that an itemset can have. This will help you set you minimum support threshold in Apriori.

Question 1.3: Run Apriori with an appropriate support threshold. If too few itemsets are returned, decrease the threshold, if there are too many, increase the threshold.

Report the running time (note: don't let the algorithm run more than 10 minutes) and the number of frequent itemests found.

At some point, the speed Apriori will become a problem to get results quickly. It would thus be relevant to analyze only one portion of the dataset. Select a relevant portion of the dataset, using visualization techniques to justify the interest of the portion selected. Apply Apriori on that portion of the dataset.

Skim through the returned frequent itemsets (either in all data or in the selected portion), find some of them that seem of particular interest, and give an interpretation of your findings.

Question 1.4: We now want to see if *closed* frequent itemsets would help. This time, use the LCM algorithm. You can use the LCM algorithm provided in the scikit-mine library (Python):

- Link to the library: https://github.com/scikit-mine/scikit-mine
- Installation information: https://scikit-mine.github.io/scikit-mine/installation.html
- Example use of LCM : https://scikit-mine.github.io/scikit-mine/tutorials/itemsets/LCM_on_chess.html

1.4 a) Answer the same question as for Apriori: bring the (complete) dataset to the input format of scikit-mine's LCM, execute it with an appropriate support and present some results.

1.4 b) Compare the running time and output size of Apriori (previous question) and LCM in your report. This comparison should span over several minimum support values and would be best presented with graphics (see examples in the DMV course). Don't forget to comment your graphics!

Question 1.5: The closed frequent itemsets tell you which products are sold together. Lets suppose that Instacart wants to build a physical shop, and that it wants to place the aisles of products that sell well together far apart. Determine the aisles that must be put far apart. Use a graph visualization to present the results, where the nodes are the aisles. The semantics of the edges is up to you (this is the interesting part of the question!).

Question 1.6: Using either of the implementations above, compute association rules. For ease of interpretation, write a simple post-processing code that selects rules concluding either on a single product that you are interested in, or only on products from a specific aisle/department.

Write an interpretation for 2-3 rules that you found.

Question 1.7: To conclude on frequent itemsets, use the SLIM algorithm from the scikit-mine library to mine itemsets from your data.

Example notebook: https://scikit-mine.github.io/scikit-mine/tutorials/itemsets/SLIM.html

SLIM is a one-step, faster variant of KRIMP that you studied in the DMV course.

- The parameters of the SLIM implementation are non-standard (https://scikit-mine.github.io/scikit-mine/reference/itemsets.html#skmine.itemsets.SLIM). Study them, try several values, and explain your final choice of parameters
- Compare the itemsets returned by SLIM with those of LCM. In your opinion, did SLIM correctly capture important structures in the data?

Task 2: Sequential patterns

The data provided by Instacart can be used to build sequences of orders for the customers (this time, one has to use the `order_products_prior.csv` dataset). The transformation of the given data to sequences of purchases is not trivial: we did it for you and provide a dataset of sequences for 20000 clients in the file `transactions_seq.txt`.

The format of each line is the following:

- Column 1: user id
- Column 2: order number (not an order id, but a number in the sequence of orders)
- Column 3: size of order
- Column 4: list of products in text format (without spaces), separated by a comma

You can observe that the first 10 lines of the line give you all 10 orders placed by user 1. You don't know the exact date of the order, but you know the sequencing of orders.

Question 2.1: We want to mine frequent sequences of itemsets from the dataset of sequences of orders. Use either the PrefixSpan or the CloSpan algorithm (CloSpan = PrefixSpan + closure) from the SPMF library (link before).

- Transform the data in the input format of the chosen algorithm
- Mine with an appropriate threshold
- Write a simple postprocessing function that gives a pretty printing of the found frequent sequences.

In your results, select 2 or 3 patterns of interest and write an interpretation.

NB: if you really look at the provided data and at your results, you may notice issues in the preprocessing. Can you spot them and find solutions to avoid having problems in your results?

Bonus points if you can provide a new transactions_seq.txt *file without these issues!*

Question 2.2: Select one of the sequential patterns with at least 3 itemsets. Find all of its occurrences in the data. Use visualization techniques to present a detailed analysis of these occurrences.

The suggestion is to focus on the "gap" between each element of the pattern: it will vary in the different occurrences. It can be interesting to show the variations of this gap graphically, allowing a finer understanding of the customer exhibiting this pattern.

Task 3 (optional): Discriminative patterns

Unfortunately, the major libraries do not provide code to mine discriminative patterns. We thus propose a simplified analysis.

Question 3.1: Divide the training dataset into two partitions, and justify your choice of partitioning. Some ideas: people that buy organic / others, people that are vegan / others,…

Question 3.2: Mine frequent itemsets in both partitions with a standard frequent itemset mining algorithm. Then choose a discriminative measure, and write a postprocessing code that takes as input the frequent itemsets of the two partitions, and returns discriminative patterns according to the selected measure (ex: GrowthRate, OddsRatio,…).

Select few discriminative patterns and write an interpretation. If possible, propose a visualization of (some of) your results.