CS5228 Assignment 1

Data Preprocessing, Clustering, and Association Rules

Due date: 2 October 2020 11.59pm

Credits: See Kiong Ng, Ziwei Xu, Yiwei Wang

Instructions and Submission

A Jupyter Notebook file, `answer_sheet.ipynb`, is provided. For a tutorial to using Jupyter notebooks, see https://jupyterlab.readthedocs.io/en/latest/ for JupyterLab, or https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/ for Jupyter Notebook. You can also watch the recordings of the class tutorial on Python Data Analysis (which can be found in LumiNUS Conferencing -> Expired).

You are expected to write all your codes and answers within the indicated spaces in the Jupyter notebook (answers to the conceptual questions can be embedded in the Notebook as markdown cells). Submit a single Jupyter notebook with the name "`YourNameInLumiNUS_YourNUSNETID.ipynb`" to the submission folder in LumiNUS.

To get started, you will need to install the following software packages:

- Python (version 3.6 or newer)
- Jupyter Notebook or Jupyter Lab
- Common python modules: pandas, numpy, matplotlib
- efficient-apriori (https://pypi.org/project/efficient-apriori/)

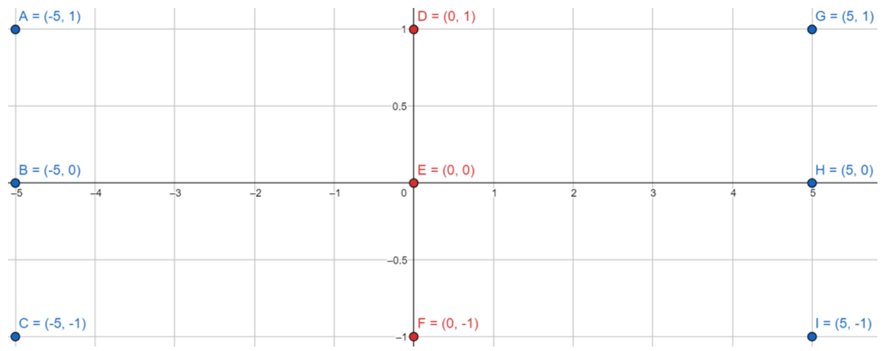
You may use any Python libraries / APIs to complete the assignment. (The focus is on understanding the properties of various algorithms and interpreting their output, not implementation; you don't have to implement them from scratch).

Here are some very useful webpages to find out more about the packages that you will be using for this assignment:

- https://pandas.pydata.org/pandas-docs/stable/indexing.html
- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.groupby.html
- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.apply.html
- https://stackoverflow.com/questions/39922986/pandas-group-by-and-sum

If you have further questions, you can email Yiwei Wang (e0409763@u.nus.edu).

1. Clustering and Initialization (20 points)



1) Consider the nine data points (A, B, C, D, E, F, G, H, I) in Fig. 1. Taking the points D, E, and F as the initial cluster centers, apply the K-Means algorithm on the data, with the number of clusters K = 3. (You can either run K-Means by hand, which is not tedious for this example, or use a Python library. You do not need to implement the K-Means algorithm yourself.) At the end of each iteration, list the positions of the cluster centers, as well as the set of points belonging to each cluster. Do you think this clustering result is satisfactory? (10 points)

2) Initialization is important for K-means. Consider the following heuristic method[1] for selecting the initial cluster center positions:

- Choose the first center c_1 as the point A.
- For $k = 2, \dots, K$, set $c_k = \arg\max_{x \in X}(\min_{i=1,\dots,k-1} \|x - c_i\|_2)$, where X is the set of data points.

Apply this heuristic to the data points in Fig. 1. Show the computed cluster centers for K = 3. Next, run the K-means algorithm with the obtained cluster centers. At the end of each iteration, list the positions of the cluster centers, as well as the set of points belonging to each cluster. (10 points)

2. Selecting the Number of Clusters (10 points)

1) Here, we will explore how to select the number of clusters. Using Python 3.6, load the attached data file ‘assignment1.data’ using the following commands:

```
import joblib
X = joblib.load(‘assignment1.data‘)
```

This results in X, which is a 400 by 2 matrix, where each row is a single sample, in 2 dimensions. Apply K-means on these samples with K ranging from 1 to 10. Plot a figure, where the y-axis is the Within Cluster Sum of Squares (WCSS) after convergence, and the x-axis is K from 1 to 10:

$$WCSS = \sum_{k=1}^{K} \sum_{x \in c_i} \|x - c_i\|_2^2$$

[1] This heuristic can be seen as a deterministic version of the K-means++ initialization we talked about in class. Both methods consider points based on measuring their distance to the closest cluster centers, preferring points with larger such distance, but K-means++ is randomized, while this version is deterministic.

Select a value of K that you think is appropriate for clustering this dataset, and explain the reason.

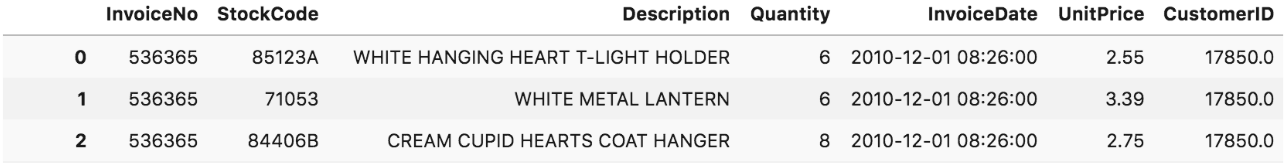
3. Data Cleaning and Exploration (20 points)

Imagine you are a data analyst for an online shopping company. Using the company's sales records, you would like to derive insights that can help to develop new strategies to improve sales.

The dataset can be found in `record.csv`. It contains the following columns:

Column Name	**Explanation**
InvoiceNo	The ID of the transaction
StockCode[2]	The ID of the item
Description	The name of the good
Quantity	The number of an good bought in the transaction
InvoiceDate	The date of the transaction
UnitPrice	The unit price of the good
CustomerID	The ID of customer

For example, the following records indicate that customer 17850 bought six "WHITE HANGING HEART T-LIGHT HOLDER", which has stock code 85123A, on 1st Dec 2010. In the same transaction 536365, the customer also bought items 71053, 84406B etc.



	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0

1) Before continuing, let us examine the dataset for "dirty" records to do some data cleaning. Remove the records with negative values of the Quantity variable, and the records with NaN values of the CustomerID variable. Report how many records were removed in total. (6 points)

After removing the "dirty" records, let us explore the dataset by getting "quick facts" such as those listed in the table below. Please provide the answers to the questions listed in the table. (8 points)

	Question	**Answer**
2)	Starting date of the dataset?	`(YYYY-MM-DD)`
3)	Ending date of the dataset?	`(YYYY-MM-DD)`
4)	Number of customers?	`(Integer)`
5)	Number of transactions?	`(Integer)`
6)	Number of different kind of goods?	`(Integer)`

[2] In the rest of this assignment, "item ID" is synonymous with StockCode, and should be used as the identifier for distinguishing items.

7)	Number of transactions customer ID 17850 have made?	`(Integer)`
8)	Which customer (ID) have made the most transactions?	`(Integer)`
9)	What is the item ID (i.e. StockCode) of the best-seller? We define “best-seller” as the item with the highest sales volume.	`(Integer)`

10) Next, let us get some general understanding about the transactions. Please make a histogram of the number of unique items per transaction (as described in the footnote) (3 points)[3] and describe one insight that you can observe from the plot. (3 points)[4]

4. Mining Association Rules (30 points)

After taking some efforts to explore the dataset to gain a good degree of familiarity with the data, you are now ready to mine the dataset for frequent patterns and association rules.

1) Let us first consider whether the “brute-force” counting method (i.e. counting all possible itemsets) is feasible. Suppose we can count 2^{36} itemsets per second. Will we complete the counting before the sun burns out (the sun has another $5 \times 10^9 < 2^{33}$ years to burn)? (4 points)

2) Run efficient-apriori in python with **min_support**=0.025, **min_confidence**=0.2, max_length=4. Write down the number of rules found and the rule with the highest lift. (4 points)

3) Run efficient-apriori in python with **min_support**=0.02, **min_confidence**=0.2, max_length=4. Write down the number of rules found and the rule with the highest lift. (4 points)

4) Run efficient-apriori in python with **min_support**=0.025, **min_confidence**=0.4, max_length=4. Write down the number of rules found and the rule with the highest lift. (4 points)

5) Compare the first two cases. How do they differ in the time taken for the algorithm to run, the number of rules found, and the lift of the highest lift rule? Briefly explain why each of these findings occur, based on the effects of changing **min_support**. (8 points)

6) Compare the first and third case. How do they differ in the number of rules found? Briefly explain why this finding occurs, based on the effects of changing **min_confidence**. (4 points).

7) Report the descriptions of the items associated with the highest lift rule you found in the three queries (you may use any method to do this). Does the rule make sense? (2 points).

[3] You can plot this histogram by running matplotlib.hist() with 200 bins on the sequence of values $n_1, \ldots, n_N$, where n_i is the number of unique items in transaction i. This produces a histogram with “number of unique items in transaction” in the x-axis, and “count” in the y-axis, i.e. each bar counts how many transactions fall into the corresponding bucket.

[4] It is sufficient to comment on the general shape of the curve and what it implies about the data; it’s fine if the insight does not seem especially interesting.