CS5228 Assignment 1

Data Preprocessing, Clustering, and Association Rules

Due date: 13 March 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://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/>.

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, seaborn
* efficient-apriori (<https://pypi.org/project/efficient-apriori/>)

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)](mailto:e0409763@u.nus.edu)).

# Clustering and Initialization (20 points) Assignment1/fig_1.png

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. 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 for selecting the initial cluster center positions:

* Choose the first center as the point A.
* For , set , where 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)

# 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:

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

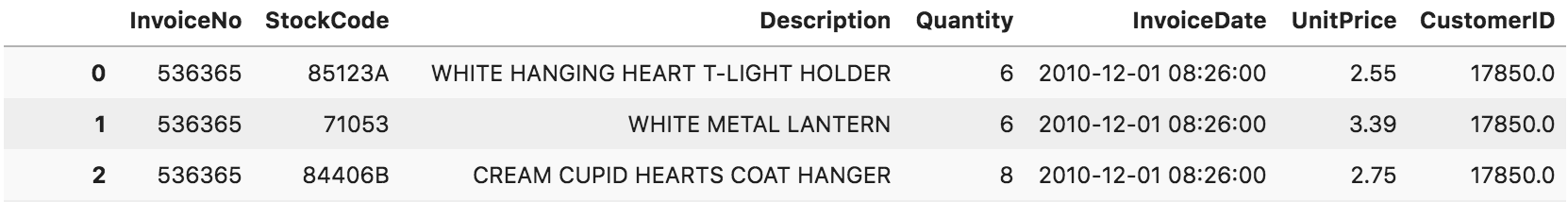
# Data Cleaning and Exploration (20 points)

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 | 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.



Before continuing, let us examine the dataset for “dirty” records to do some data cleaning. There are at least two types of “dirty” records in the dataset.  Please provide a description of each of the types of “dirty” records that you can find in the dataset, as well as the corresponding number of such records that are to be removed the dataset. (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** |
| 1) | Starting date of the dataset? | (YYYY-MM-DD) |
| 2) | Ending date of the dataset? | (YYYY-MM-DD) |
| 3) | Number of customers? | (Integer) |
| 4) | Number of transactions? | (Integer) |
| 5) | Number of different kind of goods? | (Integer) |
| 6) | Number of transactions customer ID 17850 have made? | (Integer) |
| 7) | Which customer (ID) have made the most transactions? | (Integer) |
| 8) | What is the item ID of the best-seller? We define “best-seller” as the item with the highest sales volume. | (Integer) |

9) Next, let us get some general understanding about the transactions. Please make a histogram of the number of unique items per transaction (3 points) and describe one insight that you can observe from the plot, and explain why you find it interesting. (3 points)

# 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.

Please note that questions (2)~(4) below can require fairly long computation time to complete.

1) Let us first consider whether the “brute-force” counting method (i.e. counting all possible itemsets) is feasible. Suppose we can count itemsets per second. Will we complete the counting before the sun burns out (the sun has another years to burn)? (4 points)

2) Run efficient-apriori in python with **min\_support**=0.5%, **min\_confidence**=20%, max\_length=4. Write down the rule with the highest lift (denoted as ). (5 points)

3) Run efficient-apriori in python with **min\_support**=1%, **min\_confidence**=20%, max\_length=4. Write down the rule with the highest lift (denoted as ). (4 points)

4) Run efficient-apriori in python with **min\_support**=0.5%, **min\_confidence**=40%, max\_length=4. Write down the rule with the highest lift (denoted as ). (4 points)

5) You must have noticed numerous differences between the two runs in (2) and (3). List at least 3 differences you’ve found. You may want to consider the elapsed time and the quality of the results. (3 points)

6) Which one of and do you think is better? Explain your answer. (4 points)

7) From your observation, what are the effects of increasing/reducing **min\_support** and **min\_confidence**? Support your answer with evidence. (6 points)