COMPUTATION II – ALGORITHMS & DATA STRUCTURES

Group Project Guidelines
(Spring semester — AY 2023/2024)

Joao Fonseca
jpfonseca@novaims.unl.pt

1 General Overview

You are expected to apply your knowledge about algorithms and data structures to analyze a dataset.

- All the communication will be done via email to: jpfonseca@novaims.unl.pt.
- Every student must be enrolled into a group in order to be evaluated. The enrolment activity is open on the course's Moodle page under the tab "Group Project". Submissions from students not enrolled in any group will not be accepted.
- Groups must be composed of 3 members. If you are having trouble finding a group, please send me an email as soon as possible.
- You are required to use the dataset assigned to your group. Therefore, you should download the dataset `sales_dataset_group_<XX>.pkl` from the `Datasets` folder on Moodle, where `<XX>` is your group's number.
- You are required to use the Jupyter Notebook provided on Moodle for implementing your code. You are expected to add clear and concise comments to your code to enhance readability and understanding, and create well documented functions to address each part of the project.
- The least hard-coded implementations (*i.e.*, generalizable code), the better. The more flexible the implementations (*e.g.*, having a single function that you can ensure would work in other contexts), the better.
- You may only use Python built-in libraries, `pickle`, `numpy`, `matplotlib`, `timeit` and the function `spearmanr` from the `scipy.stats` module to develop the project. No other libraries, software or programming languages are allowed.
- The deadline is June 2nd. There will be a penalty of 10% for each day of delay. See Section 3 for more information on the evaluation guidelines.

2 Project description

You have been given a file (stored as a `pickle` file) containing a dataset with sales information from a hypothetical students' union[1]. Each group's dataset is unique, which means your project will have different results from the rest of the groups.

You are expected to find out how to read the dataset file using Python on your own. Note that the dataset is likely (*i.e.*, almost certain) to have some problems. You are expected to detect these problems and fix them within reason. General guidelines to deal with these problems: (1) do not remove entries from the dataset unless you have a good reason to do so, (2) do not manually edit the dataset using any other type of software, (3) do not hard-code edits in the dataset (instead, write a function and define rule sets to deal with these problems). You may ask me for general guidance on addressing these problems, but don't expect me to explain how to perform the actual implementation of possible solutions. In addition, you are expected to write the code on your own, do not share code with other groups. However, you are allowed to search for solutions online and use code from online sources, as long as you provide the source (a URL to the source as a comment in the code is sufficient).

2.1 Dataset description

The dataset consists of a list of dictionaries, each dictionary representing a single purchase containing the following information:

- **Name:** The name of the student.
  The values associated with these keys are strings containing the names of the students. Assuming that each person has a unique name (although, in reality, this is rarely the case), there will inevitably be cases where multiple purchases are associated with the same name in the dataset. This is to be expected, as it is logical that a single student may make multiple purchases on different days, or even on the same day.

- **Date:** Date of the purchase.
  The value associated with the 'Date' key is a string that contains a date in the format of "DD-MM-YYYY". The first element corresponds to the day of the purchase, the second to the month, and the third to the year.

- **Item:** The name of the purchased item.
  The key represents the name of the item that was purchased from the Students' Union. This refers to the specific product that was bought from the range of items offered by the SU.

- **Unit price:** The item's unit price.
  The "Unit Price" key represents the price of the item purchased, expressed in Euros as a floating-point number. Please note that this value represents the price per unit of the item, rather than the total value of the purchase.

- **Quantity:** Amount of units purchased.
  The "Quantity" key represents the number of items that were purchased, expressed as an integer value.

[1]The union and the data are entirely fictitious. No identification with actual people, institutions and products is intended or should be inferred.

2.2 Information Extraction

You will extract and save information from the dataset for each customer. As you read through the tasks, it may be useful to consider which data structure you will use to store the extracted information. One option is to create a new dataset that includes customer names and the extracted information, while another option is to extend the existing dataset with additional fields for the extracted information.

Recency

Using the provided dataset, calculate and save the number of days since each student's/customer's latest purchase. You can use March 1st as the current date for this calculation. This will provide us with information about the recency of each student's last purchase, indicating how recently they made a purchase from the SU.

Frequency

Employing the dataset that has been given, calculate and save the total number of purchases made by each student throughout the given date interval. This information will tell us how many times each student bought something from the SU.

Monetary Value

Using the provided dataset, calculate and save the total amount spent by each student on purchases throughout the year. This information will tell us the total value of all the items purchased by each student from the SU at the given date interval.

Plot histograms for recency, frequency and monetary value of the dataset. You can make use of `matplotlib` to do this.

2.3 Sorting

In this section, you will sort the student names based on their calculated **recency**, **frequency**, and **monetary value**. It is important to use an efficient and suitable sorting algorithm for this task.

First RFM Score (RFM)

The three independent steps for assigning recency, frequency, and monetary value scores to each student are as follows:

- Sort the students based on their calculated recency, with the student who made the most recent purchase appearing at the top/beginning of the structure.
  You need to divide the sorted students into three equal parts based on when they made their last

purchase. Then, assign a recency score to each part, with a score of 3 for the most recent third of customers, a score of 2 for the middle third, and a score of 1 for the least recent third.

- Re-sort the students based on their calculated frequency values, highest frequency being on top / beginning. Assign a frequency score of 3 to the top one-third of the list, 2 to the second one-third, and 1 to the remaining one-third.
- Re-sort the customer list based on their calculated monetary values, largest value being on top / beginning. Assign a monetary score of 3 to the top one-third of the list, 2 to the second one-third, and 1 to the remaining one-third.

After these steps, we will have assigned an RFM score (recency, frequency, monetary value) to each student.

Using the name of the students and their RFM score, perform another sorting of the data:

- Sort the names using their RFM score in descending order. In the end, we are expecting to see the students with RFM score 333 at the top/beginning of the list and 111 at the bottom/end of the list.

Can you determine which students are most valuable to our SU, considering that our organization is unfair and values students solely based on their purchase history?

Second RFM Score (RFM')

Recalculate the recency, frequency and monetary value scores for each student as follows:

- Follow the first step of the First RFM Score calculation. Rank the students by their recency and create 3 buckets with scores from 3 to 1.
- Within each bucket of recency, rank students by frequency and create 3 buckets. Assign a frequency score of 3 to the top one-third of the list, 2 to the second onethird, and 1 to the remaining one-third.
- Within each bucket of frequency, rank customers by monetary and create 3 buckets. Assign a monetary score of 3 to the top one-third of the list, 2 to the second one-third, and 1 to the remaining one-third.

Use `scipy.stats.spearmanr` to correlate RFM and RFM'. This function calculates the Spearman Rank Correlation which is used to measure the correlation between two ranked variables.

2.4 Searching

- Using linear search, find the RFM scores of your group members (assuming the dataset contains names of everyone, otherwise use another name).

Which is the best student/customer among your group?

- Sort the students in alphabetical order (hint: you may assign a numerical value to indicate the position of a letter in the alphabet using a dictionary).
- Search for your group members using binary search and find their RFM scores.

How much time, in seconds, did it take to find a name using linear search and binary search? Compare the results (you can use the timeit module).

3 Evaluation Guidelines

A minimum grade of 9.5 is required for both evaluation components. The grade of the project will be the same in the first and second examination epochs. Grade of this group project will be given by the following formula:

$$\text{Project Grade} = 0.5 * \text{Notebook Grade} + 0.5 * \text{Defense Grade} \tag{1}$$

The following list provides details about each major criterion:

- The code must be clean, able to run without errors, and well commented. Clarity, synthesis, and objectiveness are essential for good quality programming.
- Ability to explore, prepare and visualize the data.
- Ability to correctly perform sorting.
- Ability to correctly perform searching.
- Ability to use suitable data structures.
- Ability to justify your decisions with a priori reasoning and comment upon the obtained results.

Important notes regarding grading:

- Elements from the same group might end up having different grades, depending on the quality of the discussion held by each individual student.
- An absence in the project discussion will result in a discussion grade of zero for that student (the remaining members of the group will not be affected).
- If you are a repeating student, you may use the code you developed last year, **only if all members in the group are repeating students**. Therefore, **repeating students may not team up with students taking the course for the first time.** In addition, note that your final grade might not necessarily be the same as last year (the evaluation criteria will be different and your project grade is still dependent on the discussion). **If a student taking the course for the**

first time forms a group with a repeating student, the new student will get a zero in the project grade.

3.1 Deadline

The last day to deliver the project is June 2nd, Sunday night at 23:59. You can resubmit your project on Moodle as many times as you want until the deadline. There will be a penalty of 10% for each day of delay.

3.2 Defense

There will be a mandatory presential defense where all the group members must participate. Different members of the same group can receive different grades. There is no need for any kind of slides or presentations for the defense, it will be in Q&A format. The absence of a group element implies 0 on their grade of defense. The schedule of the defenses will be published closer to the date.