CITS1401 Computational Thinking with Python

Project 2, Semester 1, 2024

Submission deadline: **24th May 2024, 6:00 PM**

Total Marks: **30 (Value: 20%)**

Project description

You should construct a Python 3 program containing your solution to the following problem and submit your program electronically on Moodle. The name of the file containing your code should be your student ID e.g., 12345678.py. No other method of submission is allowed. **Please note that this is an individual project**. Your program will be automatically run on Moodle for some sample test cases provided in the project sheet if you click the "check" link. However, this **does not test** all required criteria and your submission will be **manually** tested thoroughly for grading purposes after the due date. Remember you need to submit the program as a single file and copy-paste the same program in the provided text box. You have only one attempt to submit, so do not submit until you are satisfied with your attempt. All open submissions at the time of the deadline will be automatically submitted. Once your attempt is submitted, there is no way in the system to open/reverse/modify it.

You are expected to have read and understood the University's guidelines on academic conduct. In accordance with this policy, you may discuss with other students the general principles required to understand this project, but the work you submit must be the result of your own effort. Plagiarism detection, and other systems for detecting potential malpractice, will therefore be used. Besides, if what you submit is not your own work then you will have learned little and will therefore, likely, fail the final exam.

You must submit your project before the deadline mentioned above. Following UWA policy, a late penalty of 5% will be deducted for each day i.e., 24 hours after the deadline, that the assignment is submitted. No submissions will be allowed after 7 days following the deadline except approved special consideration cases.

Project Overview

This project is also based on the social media dataset. In today's digital age, social media has become an integral part of our daily lives, shaping how we connect, communicate, and consume information. With millions of users engaging across various platforms, understanding user behaviours and trends on social media has become increasingly important. The aim of this project is to analyse the data to better understand social media usage and its impact.

The dataset for this project comprises several columns, including age, gender, time spent on social media (in hours), engagement score, platform, interests, country, demographics, profession, income, debt status, etc.

You are required to write a Python 3 program that will read a CSV file. After reading the file, your program is required to perform statistical analysis described below and return them as outputs.

Input

Your program must define the function `main` with the following syntax:

```
def main(csvfile):
```

The input arguments for this function is:

- `csvfile`: The name of the CSV file (as string) containing the record of the social media and their users' details. The first row of the CSV file will contain the headings of the columns. A sample CSV file "social_media.csv" is provided with the project sheet on LMS and Moodle.

Output

We expect 4 outputs in the order below.

i. **OP1** = list of two dictionary items: A list containing two dictionary items: the first dictionary item containing the record of user whose profession is "student" while the second dictionary item containing the records of users who are not students. The key for both dictionary items is ID of the user and value is a list containing `age`, `time_spent_hour`, and `engagement_score`, in the order mentioned.

ii. **OP2** = dictionary item: A dictionary where key is a platform and value is a list, containing the total, average and standard deviation of engagement time of users using that platform.

iii. **OP3** = list of cosine similarity scores: A list of two numeric values calculating the cosine similarity between the `age` and `income` for users whose profession is "student" and for the users whose profession is not student.

iv. **OP4** = Cohen's d test: A numeric value for Cohen's d test for engagement time to find the difference between two groups: one group is of the users whose profession is "student" while the other group is of users whose profession is not student.

All required formulae are provided at the end of the project sheet. All returned numeric values (in dictionaries, lists and individual) must contain values rounded to four decimal places (if required to be rounded off). Do not round the values during calculations. Instead, round them only at the time when you save them into the final output variables.

Requirements

1) **You are not allowed to import any external or internal module in python**. While use of many of these modules, e.g., `csv` or `math` is a perfectly sensible thing to do in a production setting, it takes away much of the point of different aspects of the project, which is about getting practice opening text files, processing text file data, and use of basic Python structures, in this case lists and loops.
2) Ensure your program does NOT call the `input()` function at any time. Calling the `input()` function will cause your program to hang, waiting for input that the automated testing system will not provide (in fact, what will happen is that if the marking program detects the call(s), it will not test your code at all which may result in zero grade).
3) Your program should also not call `print()` function at any time except for the case of graceful termination (if needed). If your program has encountered an error state and is exiting gracefully then your program needs to return zero for numerical

values, empty for non-numerical items such as, "" for string, [] for lists and {} for dictionary items, and print an appropriate message. At no point should you print the program's outputs instead of (or in addition to) returning them or provide a printout of the program's progress in calculating such outputs.

4) Do not assume that the input file names will end in `.csv`. File name suffixes such as `.csv` and `.txt` are not mandatory in systems other than Microsoft Windows. Do not enforce that within your program that the file must end with a `.csv` or any other extension (or try to add an extension onto the provided csv file argument), doing so can easily lead to syntax error and loosing marks.

Examples

Download `social_media.csv` file from the folder of Project 2 on LMS or Moodle. An example of how you can call your program from the Python shell and examine the results it returns, is provided below:

```
>> OP1, OP2, OP3, OP4 = main('social_media.csv')
```

Few details of the returned output variables are:

```
>> len(OP1[0])

23

>> OP1[0]['95q155']

[16, 8, 27.95]

>> len(OP2)

3

>> OP2['youtube']

[40.5571, 0.5878, 1.1939]

>> OP3

[0.8651, 0.9722]

>> OP4

3.1625
```

Assumptions

Your program can assume the following:

- The order of columns can be different than the order provided in the sample file. Also there can be extra columns in the CSV file. Moreover, rows can be in random order except the first row containing the headings.

- All string data in the file is case-insensitive, which means “Asia” is same as “ASIA”. You program needs to handle the situation to consider both to be the same. The string data for outputs (including keys of dictionary items) must be in lower case.
- There can be missing or invalid data, for instance, age cannot be negative, user ids must be alphanumeric, engagement score cannot be non-numeric, etc. There cannot be two identical records (rows) or multiple records (rows) for single user. You need to think of other cases yourself. If there is any invalid data then entire row(s) should be ignored.
- Number of different social medial platform, professions, countries, demographics will vary, so do not hardcode.
- The main function may not be provided with valid input parameters.
- The necessary formulae are provided at the end of this document.

Important grading instruction

Note that you have not been asked to write specific functions. The task has been left to you. However, it is essential that your program defines the top-level function `main(csvfile)` (hereafter referred to as “main()” in the project document to save space when writing it. Note that when `main()` is written, it still implies that it is defined with its input argument). The idea is that within `main()`, the program calls the other functions. Of course, these functions may then call further functions. This is important because when your code is tested on Moodle, the testing program will call your `main()` function. So if you fail to define `main()`, the testing program will not be able to test your code and your submission will be graded zero. Don’t forget the submission guidelines provided at the start of this document.

Marking rubric

Your program will be marked out of 30 (later scaled to be out of 20% of the final mark). 22 out of 30 marks will be awarded automatically based on how well your program completes several tests, reflecting normal use of the program, and also how the program handles various states including, but not limited to, different numbers of rows in the input file and / or any error or corner states/cases. You need to think creatively what your program may face. Your submission will be graded by data files other than the provided data file. Therefore, you need to be creative to investigate corner or worst cases. I have provided few guidelines from ACS Accreditation manual at the end of the project sheet which will help you to understand the expectations.

8 out of 30 marks will be awarded on style (5/8) “the code is clear to read” and efficiency (3/8) “your program is well constructed and run efficiently”. For style, think about use of proper comments, function docstrings, sensible variable names, your name and student ID at the top of the program, etc. (Please watch the lectures where this is discussed).

Style Rubric:

0	Gibberish, impossible to understand.
1-2	Style is really poor or fair.
3-4	Style is good or very good, with small lapses.
5	Excellent style, really easy to read and follow.

Your program will be traversing text files of various sizes (possibly including large csv files), so you need to minimise the number of times your program looks at the same data items.

Efficiency rubric:

0	Code too complicated to judge efficiency or wrong problem tackled.
1	Very poor efficiency, additional loops, inappropriate use of `readline()`,etc.
2	Acceptable or good efficiency with some lapses.
3	Excellent efficiency, should have no problem on large files, etc.

Automated testing is being used so that all submitted programs are being tested the same way. Sometimes it happens that there is one mistake in the program that means that no tests are passed. If the marker can spot the cause and fix it readily, then they are allowed to do that and your - now fixed - program will score whatever it scores from the tests, minus 4 marks per intervention, because other students will not have had the benefit of marker intervention. Still, that's way better than getting zero. On the other hand, if the bug is hard to fix, the marker needs to move on to other submissions.

Extract from Australian Computing Society Accreditation manual 2019:

As per Seoul Accord section D, a complex computing problem will normally have some or all of the following criteria:

- involves wide-ranging or conflicting technical, computing, and other issues;
- has no obvious solution, and requires conceptual thinking and innovative analysis to formulate suitable abstract models;
- a solution requires the use of in-depth computing or domain knowledge and an analytical approach that is based on well-founded principles;
- involves infrequently encountered issues;
- is outside problems encompassed by standards and standard practice for professional computing;
- involves diverse groups of stakeholders with widely varying needs;
- has significant consequences in a range of contexts;
- is a high-level problem possibly including many component parts or sub-problems;
- identification of a requirement or the cause of a problem is ill defined or unknown.

Necessary formulae

i. **Cosine Similarity**

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

where A_l and B_i are the ith components of vectors A and B, respectively. You can find more details at https://en.wikipedia.org/wiki/Cosine_similarity.

ii. **Standard deviation:**

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}$$

where x_1, x_2, x_3 … … x_n are observed values in the sample data. $\bar{x}$ is the average value of observations and N is the number of observations.

iii. **Cohen's d test:**

A Cohen's d test is a statistical method used to determine the differences between two group means. It is defined as:

$$d = \frac{\mu_1 - \mu_2}{s} \qquad s = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

where μ_1 and μ_2 are the averages of Group 1 (student) and Group 2 (other users) respectively, s is the pooled standard deviation, s_1 and s_2 are the standard deviations of Group 1 and Group 2 respectively, and n_1 and n_2 are the sample sizes of Group 1 and Group 2, respectively.

iv. **Engagement time:**

Engagement time is calculated as:

Engagement time = (time_spent_hour x engagement_score) / 100