AAE 875: Fundamentals of OOP and Data Analytics

Summer 2019

Final Program (FP)

<< Implementing a Data Analyst chatbot >>

In this assignment, you will develop a computer program that simulates a Data Analyst chatbot, named **Freddie**, capable of processing user input and returning desired output following the rules and directions in the script. Note that Freddie's level of intelligence is directly related to your instructions. Edits must be made directly to the script in order to change the manner by which the program operates.

You are a Data Scientist working for Google's Healthcare Division (Verily) and you have recently been assigned the task of analyzing hospital discharge data from the state of New York. The data consists of **more than 3 million** de-identified patient level **observations** between 2014 - 2016 (yes, we are talking about big data!). The deadline is just around the corner and you are asking **Freddie**, one of the (imaginary) Data Analysts in your team, to help with this task.

Here is a sample run of the work Freddie can do for you (interleaved input and output).

As shown in the sample above, Freddie can help you identify your computer's operating system (OS), set the input and output paths, read input data stored into the memory of your computer, provide descriptive statistics for key variables in the analysis, and finally, run a linear regression model of your choice. If **Freddie** is implemented correctly, your only task as a Data Scientist is to interpret the results of the regression and report these to the team.

The data can be directly downloaded from health.data.ny.gov for the years 2014, 2015, and 2016. I recommend downloading the comma separated files.

Grading Scheme [100 points]:

Each part of the program (module / main script) is worth 20 points, as follows:

- 5 points for folder structure, code style and comments (please follow instructions provided in class; set editor margins to 80).
- 15 points for code execution (your program should execute without any modifications by TA or instructor; Adam will test your code on a Mac machine, I will test on a Windows machine; we will only review .py files (please do not submit .ipynb files).

Files: main.py, path.py, input.py, cleaning.py, statistics.py, regression.py

- main.py – Script that contains all the functions (and stubs) necessary for Freddie to process user input and return desired output.
- path.py – Module that contains all the functions (and stubs) that are needed to set up your OS, the drive letter, and input and output directories.

- input.py – Module that contains all the functions (and stubs) needed to read data stored in the internal memory of your computer.
- cleaning.py – module that contains all the functions (and stubs) to perform any necessary data cleaning
- statistics.py – Module that contains all the functions (and stubs) to perform data visualization and descriptive statistics.
- regression.py – Module that contains all the functions (and stubs) to perform regression analysis.

Detailed function headers are provided for you in the main script, path and input modules. The implementation of each function must meet the detailed descriptions in the function headers. You are responsible to create the cleaning, statistics and regression modules following the same idea.

The input and output paths are defined as the paths to local directories in your computer where raw data will be stored (input) and cleaned data will be saved (output).

Tasks:

1. You will start by implementing the **welcome_prompt()** and **grettings()** functions in the *hospitalsNY script*.

2. The next step is to set the OS, the drive letter corresponding to your computer's OS, and finally the input and output paths. For this you will implement the **os_and_dtive_letter()**, **input_path()** and **output_path()** functions. All these functions rely on the correct implementation of the *path.py module*.

3. Assuming the paths are correctly set, you are now ready to read the raw .csv data. For this, you will implement the **data_input()** function and its associated *input.py module*.

For tasks 1-3 above if user input is not recognized by Freddie, then user should be prompted to enter input again. This means that your program should not end unless I see the regression results.

4. Once the data is correctly read in the memory of the computer, you can move to the implementation of the **data_cleaning()** function and its associated *cleaning.py module.* You can choose the data structure that you want to work with (e.g. nested list, dataframe). Make sure there are (1) no observations with missing values and that (2) variable names are consistent across time.

5. Assuming the data is clean and ready to visualize, implement the **summary_stats()** function and its associated *statistics.py module.* At this step you can choose to write tables and/or to plot graphs. I expect to see at least 5 tables and/or graphs summarizing 10 variables in the dataset. The variables you choose should be related to your regression specification (see point 6 below).

6. You are almost done! The final task is to implement the **linear_model()** function and its associated *regression.py module*. The data you have in hand is super rich (more than 3 mil observations and more than 30 variables!) so I expect from you to come up with a research question that estimates the casual relationship between two variables in the dataset (hint: do not forget to control for other covariates as well).

To submit your work, create a PRIVATE repository in GitHub and add Adam (adamtheising) and I (corneliailin) as contributors. We will download your submissions from here after the deadline.

Please follow the **folder structure** we discussed in class (re: Lecture 4). Your Literature folder should contain (a) pdf versions of all the online sources you used to create your code, and (b) a README.md file with the contribution of each person that helped you with this task + a summary with links to the pdf files mentioned in (a).

Finally, **use Git and GitHub to version control** your folder (re: Lab 3, [clone, pull, push]). Make sure to push your code periodically (I recommend doing this at least once per day). We will review your progress during this period and provide feedback as needed.