Final Project, Deliverable 3

Submission. For your submission, each group will upload their presentation, a link to their GitHub repository containing their Python project (as a comment in Canvas), and any other supporting materials if those were not already represented in the project itself. I should be able to reproduce all the results that you describe in your presentation.

The culminating activity in this course is the final project. In groups of three to four students, you will design a project that establishes a research question and executes some type of experimental procedure around it. This work involves two key components: a software component and a presentation component.

The **software component** will be a GitHub repository containing the Python package your team has created for your project. Your software should follow Python best practices, leveraging the ideas we've been developing throughout the semester. In particular, your software should at least:

- follow modern project structuring described in our "A Second Look at Python Packages" labs—that is, use the `pyproject.toml` format instead of the `setup.py` format
- use object-oriented programming when appropriate (remember, part of expertise is knowing when to apply what you know and when not to)
- feature a carefully-designed testing suite that tests the code you've written **using `pytest`**
- **support logging using the `logging` module at sensible places**
- allow the end-user to access and use the package with an easy-to-understand API (in other words, it should be simple for me to reproduce your experiments)
- discuss the high-level aim of the package in the README, including a demonstration of how to use key features and a graphic of a diagram illustrating how the pieces of your package connect to one another. In addition, you must use draw.io or some other diagram software to create an image showing your program architecture. The point is, someone who has never looked at your code should be able to easily ground themselves in the high-level structure of what you've done by reading your documentation. **In the README, put the full names of each team member.**
- include all other project-structuring code, such as an `environment.yml` file (which will likely differ from our class's configuration if you use additional libraries)
- **Data should not be pushed to GitHub. Instead, you should use GitHub LFS for large data files, or provide a link to the cloud (e.g., Google Drive), where the data can be downloaded using your code. Then, you can preprocess the data, or store preprocessed data in the cloud instead (e.g., if preprocessing takes a long time).**
- **Your team is expected to collaborate on GitHub. One team member should create a Git repository for the project, and the other team members should pull remote versions of that repository for their own local copies. Each member is expected to make contributions, which will be reflected in the commit history of the repository. Gaining some comfort with this process is important for what real team-based software development looks like.**
- every function, method, and class should have a docstring
- be thoughtfully-designed and organized
- use a linting software like `pylint` or an opinionated, PEP 8 formatter like black or ruff to clean your code. It is possible to have such software run every time you save changes; for example, you can install a formatter or linter as a VS Code plugin.

Your **team's presentation** should follow the best practices we outlined when thinking about effective presentations, in addition to whatever flare your team brings during showtime. Your goal, of course, is to tell a memorable story—have fun with it. Your presentation should be no longer than a fixed time limit, which we will calculate in class based on the number of groups. In addition to a slide deck, you may want to provide a live demo.