Final Project, Deliverable 3

Submission. For your submission, each group will upload their presentation, a link to their GitHub repository containing the below contents, 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. I should be able to stand up your application and faithfully reproduce your work.**

The **software component** of this deliverable will be a GitHub repository containing:

- the Python package your team has created containing your team's analytics
- frontend logic
- Dockerfiles and any containerization logic
- all files relevant for cloud configuration and connection

Your software should follow best practices, leveraging the ideas we've been developing throughout the semester. In particular, your software should at least:

- ❒ **At the top of your README, put the full names of each team member.**
- ❒ 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 and describe the **architecture diagram of your entire system**. 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.
- ❒ 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)
- ❒ be stylistically consistent: it should be non-obvious for the reader to tell what parts of the code were written by different authors
- ❒ be thoughtfully-designed and organized
- ❒ use object-oriented programming when appropriate (remember, part of expertise is knowing when to apply what you know and when not to)
- ❒ every function, method, and class should have a docstring
- ❒ static-site documentation rendered via MkDocs or sphinx
- ❒ **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.**
- ❒ **use issue templates and track progress through a robust Git collaborative workflow, as described in Assignment 2**
- ❒ use a linting software like `pylint` or an opinionated, PEP 8 formatter like ruff to clean your code
- ❒ follow modern project structuring described in our Modern Python Packaging lab—that is, use the `pyproject.toml` format
- ❒ feature a carefully-designed testing suite that tests the code you've written **using `pytest`**
- ❒ **support logging using the `logging` module at sensible places**
- ❒ **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).**
- ❒ use GitHub Actions to run standard CI steps
  - ❒ define an operating system
  - ❒ install Python
  - ❒ access the repository
  - ❒ install `poetry`
  - ❒ install your package
  - ❒ run few tests
- ❒ follow a microservices design pattern, where individual components of your work are separate Docker container images, connected via Docker Compose. As described in Assignment 3, this means making good design decisions about the software you'll use for a user-facing UI, API layer, backend database logic, analytics, and anything else relevant to the functioning of your application
- ❒ use experiment tracking and monitoring tools such as Evidently, Data Version Control, and MLFlow
- ❒ deploy to a cloud platform. You are allowed to deploy to Hugging Face Spaces for the final project, and you can deploy to GCP, Azure, AWS, or something similar.
- ❒ **be very clear about your baseline methods, how you improved over them, and how you measured that improvement**
- ❒ Include all of your analysis in a PDF in your repository called `discussion.pdf`. The `discussion.pdf` should be 3 pages, excluding references, and it will include your comments, results, and overall analysis. Your general structure might be something like the below. You should use Times New Roman, 12-point font, single-space, with one-inch margins. Provide citations to all techniques and literature cited in a reference section, which will not be included in the required page count. Discuss:
  - ❒ exploratory data analysis over your dataset (e.g., collection size, collection characteristics, …)
  - ❒ details of engineering decisions such as preprocessing (e.g., discussing how you preprocess documents, did you use techniques like deduplication or other forms of preprocessing, …)
  - ❒ details of your algorithms, describing the baseline method and improvements on it
  - ❒ how you evaluated how well your approaches performed at your task of interest (some tasks may have annotated data associated to them and some will not)
  - ❒ descriptions of how your team adapted to the feedback you received from students and the instructor

- ❒ visualizations, results, and overall analysis
- ❒ challenges you encountered and conclusions

❒ **The upshot is that the final deliverable will be a Dockerized pipeline of microservices comprising a complex frontend and backend that is deployed to a cloud service.**

Your **team's presentation** should follow the best practices we will outline when we think 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 length will be determined by the number of groups in the classroom. In addition to a slide deck, you may want to provide a live demo.