DS 5010 Projects

Overview

The purpose of these projects is to gain hands-on experience working on a non-trivial programming project in a collaborative team. The project task is to implement a simple Python package designed around a specific purpose of your choice.

Projects are to be completed in teams of 2 - 4 class members. The project is worth a total of 30% of the class grade, and includes a written project proposal, a written report, and a package repository. Each team should work independently on their own projects, but may collaborate on technical questions via Piazza and Github. *Make sure to proofread your proposal and report – it should appear polished and professional.*

Projects may utilize existing external libraries *as long as there is no significant functional overlap with the project package.* If the project makes significant use of external libraries, then the proposal and report must clearly state which aspects of the project are the teams' novel effort. Projects with a very high degree of similarity with other past or current students' work, or with other existing work (e.g., found online), which are not clearly cited, will be considered plagiarism. In such instance, all team members will receive a zero on the project, and will be reported to the university.

Each project is expected to be hosted on a public Github repository owned by one of the team members. This is to encourage openness and reproducibility, by allowing your code to be easily reviewed by instructors and other students. Please do not include large datasets or large binary files in your repositories.

Suggestions

You are encouraged to be creative in your package's functionality and design. Good projects will have a clearly-stated purpose, with modules, classes, and functions all organized around achieving that purpose. You are encouraged to talk to the instructor about project ideas before submitting the proposal.

Your package's functionality does not need to be unique or novel (i.e., it may duplicate functionality already provided by other external libraries), but the implementation must be the teams' own work. The package should avoid duplicating functionality from built-in libraries.

Some possible ideas for project packages:

- Array, matrix, and/or sparse matrix library
- Tools for importing and manipulating a genre of data (e.g., CSV, XML, images, etc.)
- Library for a specific type of data analysis or modeling (e.g., SVM, PCA, etc.)
- Graphical tools for plotting a specific type of data
- Automation tools (e.g., bulk file renaming, manipulation, etc.)
- Implement an advanced data structure or data storage interface

You are strongly encouraged to brainstorm ideas beyond these examples. Ideally, the package should be useful in some aspect of a data science workflow, but this is not strictly required.

Your package **must** have a unifying purpose. It should not simply be a collection of unrelated utility functions. However, make sure to consider what additional convenience functionality you should provide to ensure your users get the most use of your package.

Teams

Please form teams of 2 - 4 class members who will work together on the project. You may reach out on Piazza when seeking out other project team members. Exceptions may be made for individual projects in extenuating circumstances.

Proposal

The project proposal be no more than 1 page containing the following:

1. **Title:** A descriptive title of the project
2. **Authors:** List all team members' full names
3. **Summary:** 1-2 paragraphs summarizing the overall purpose of your package.
4. **Proposed design:** 2-3 paragraphs describing in detail the modules, classes, and functions you will need to implement to provide the package's intended functionality. Describe any external libraries you may need to use, and what aspects of the implementation will be your team's work. Describe any potential challenges you foresee in your implementation.

The project proposal should be submitted as a PDF on Canvas.

Report

The project report should be no more than 2-3 pages containing the following:

1. **Title:** A descriptive title of the project
2. **Authors:** List all team members' full names
3. **Github:** A link to a Github repository where the project's source code will be deposited.
4. **Summary:** Summarize the overall purpose of the package (i.e., what problem does it solved), any related work or similar libraries, and a brief, non-technical description of any notable modules, classes, or functions.
5. **Design:** A technical description of the modules, classes, and functions implemented in the package. This does not need to be exhaustive, and should only cover the most important aspects of the package for a user to understand. The description should be clear and concise so that a user can understand the overall design and organization of the package.
6. **Usage:** Description and examples of how the package should be used, how it fits into a workflow. This section may include code snippets, but it should primarily be a written description.
7. **Discussion:** Discuss how the package compares to related libraries, and how it fits into the overall ecosystem. Why should people use this package? How could the package be improved?
8. **Statement of contributions:** List the full names of the authors and how each member contributed to the completion of the project.
9. **References:** Cite any external libraries used by the project, and any sources that were used as a reference. Use a consistent format and numbering scheme

The project report should be submitted as a PDF on Canvas.

Package repository

The complete code for each project hosted on a public Github repository owned by one of the team members. The repository should include at least the following:

1. **README.md:** A Markdown-formatted description of the package's overall purpose, the organization of the package code in the repository, and some simple examples of usage.
2. **___init___.py**: Initialization script marking the repository as a Python package.
3. **Modules:** Modules ending in ".py" containing Python code defining classes and functions.
4. **Test:** Test scripts ending in ".py" containing unit tests covering most of the package's functionality.

A link to the repository should be submitted on Canvas.