CMSE 202 Final Project

Requirements and Grading Rubric

Overall requirements

Each presentation should be **no more than 12 minutes** long and should include appropriate **visual aids**. There will be **~3 minutes** allocated for questions from the rest of the class. The presentation should be **divided roughly evenly between all group members** (i.e. one person shouldn't do all of the talking). Your presentation should address:

- The **question(s)** you set out to answer.
- The **model**, expressed in broad mathematical terms, that you applied to your chosen data or chosen topic.
- The **computational techniques** that you used to analyze your data or run your model. This should include an explanation of any pre-existing python-based libraries or packages that you used to aid in your project.
- The **answer(s)** you arrived at.
- The **difficulties or complications** you ran into with your project and what you did to try to cover overcome these difficulties.

Talk slides

It is recommended that you create your slides using Google Slides so that you can more easily collaborate on the slides with your group members, but you are allowed to use another format if there is one that you prefer. If you use Google Slides, you should share your slides with your instructor using their `@msu.edu` email address when you've completed them. Once you've completed your slides, you should also upload a copy of them to D2L using the "Project Presentation" dropbox. You should aim for having a reasonable number of slides -- a good rule of thumb is ~1 min of time per slide. You should avoid having more than 15 slides for a presentation of this length. In addition, make sure you have enough slides to support your presentation. If you have too few slides you might either have too much content per slide or you don't have enough content in total. **The slides should**:

- Address the above points in a logical order.
- Include a title slide with your project title, name, and course number (with section number).
- Effectively communicate information through a combination of text and figures. Text and figures should be

legible (even from the back of the room!).
- Only include text and figures that support the presentation (avoid including content "just because").
- Avoid large paragraphs of text, use bullet points instead.

Code requirements

All code used for the project should be committed to a **private** GitHub repository that is shared with the instructors. The code should be well-documented and include a README.md file in the GitHub repository that explains how to run the code and a description for what each group member contributed to the overall project. Well-documented code should include comments to explain what the code is doing and docstrings for functions and classes. You should also make sure that the variable names are chosen so that the instructor can make sense of your code.

Grading

The oral presentation, slides, and project code should address all of the points outlined in the "Overall Requirements" above. The project will be graded as follows:

Presentation

- 30% of the grade (**30 points**) comes from the oral presentation and slides. Is it clear and well-structured? Does the student effectively communicate the key ideas about their results? Do the slides complement their oral presentation, and conform to the guidelines?

> #### Examples of presentation performance
>
> - *Excellent presentation*: student speaks clearly and in a logical manner, and makes their key points clear. They use eye contact and minimally use notes. Slides conform to specifications in final presentation document, including number of slides, adequately addressing all points, legibility. Slides complement oral presentation. (**30 points**)
> - *Good presentation*: student speaks clearly and logically and conveys key points, but presentation is somewhat lacking (heavy use of notes; moderate eye contact, etc.) Slides deviate from specifications in some minor way: too many or too few slides, not all points addressed, some slides hard to understand (poor graphics, too much or too little text, etc.) (**22 points**)
> - *Fair presentation*: student's oral presentation is substantially lacking: little eye contact, speaking too quietly to be heard or with little inflection, clarity/logical flow in speaking is sub-par, etc. Slides deviate from specifications in some substantial way: too many or too few slides, half or fewer of points addressed, most slides hard to understand (poor graphics, too much or too little text, etc.) (**14 points**)

> - *Poor presentation*: student's oral presentation is completely lacking: little to no eye contact, cannot be heard, there is no logical progression to the presentation. Slides do not conform to specifications in any meaningful way. (**6 points**)

Demonstration of content knowledge and computational skills

- 30% of the grade (**30 points**) comes from the content of the student's presentation and in the code developed. Did the students approach a specfic question in a scientific way? Did the students compare and contrast two or three different but appropriate techniques? Are the results (positive or negative!) presented in a clear and convincing manner? This section is subdivided into two categories: computational techniques and visualization.
    - Computational Tools (**20 pts**): Which tool(s) did you use to solve your problem? Did you use the correct tool to answer your scienctific question? Your choice of tools should be justified in your presentation.
    - Visualization (**10 pts**): Are your results clearly displayed? Are the visualizations easy to digest for someone not familiar with the project? Do the visualizations effectively demonstrate some point?

> ### Examples of content knowledge and skills demonstration
>
> - *Excellent presentation*: the students pose a specific question with a measureable outcome and use a few different techniques to try to answer the question. The students explain why the chosen techniques are appropriate for answering the proposed question, and compare how the two techniques performed. The graphs are well labeled and helper text is added to the figures to highlight important parts of the figure. (**30 points**)
> - *Good presentation*: the students pose a specific question with an unclear metric and a few different techniques to try to answer the question, but one of the techniques is not appropriate for the problem. The students briefly explain why they chose the techniques, but do not compare the techniques. The graphs have labeled axes and legends. (**22 points**)
> - *Fair presentation*: the students use a single technique to try to answer a question that isn't well defined. The students briefly explain why they chose the technique. The graphs simply show raw data or are not labeled. (**14 points**)
> - *Poor presentation*: the students forego computational tools and try to answer a poorly-defined question using graphs and intuition. (**6 points**)

Collaboration

- 20% of the grade (**20 points**) comes from working together effectively. This grade is individual to each team member. Team members that worked well with the group will get a much higher grade than students who refused to meet or take on tasks.

    - Balancing the workload: Was the work effectively distributed among all team members? Did each team member contribute meaningfully?
    - Use of collaborative code development tools: Did you leverage version control software to aid in code development between group members? Are your commits descriptive? Did you manage merge conflicts successfully? Does each team member have at least a few commits (not necessarily in equal quantity)?

Good coding practices

- 20% of the grade (**20 points**) comes from the quality of the code that is written. In addition, your instructor should be able to use the README to understand how to appropriately run your code.

    - Does the code have adequate comments? Do the variable names make sense? Do function and classes have Docstrings? Your instructor should be able to read through your code and understand what it is doing and why. Is the data managed effectively and efficiency such that it is easy to understand how the data is being manipulated? Are variables clearly identified? Are appropriate data types used (dictionaries vs. numpy arrays etc.) ? Look at past examples of code to get a sense for the level of detail you should include.