What We Need To Do For Final Project Next Week & In The Future:

Code Review:

- Code reviews are done to improve code, find bugs. Ultimate goal is help someone improve their code, learn from it (both sides).
- Should describe what application does and give you the opportunity to describe what your code does
- **Mainly, Comments are given on:**
    a. Choice of variable and function names
        i. Don't pick variable names like "l".
    b. Readability of code
        i. Many ways to code, but not are equivalent. Code should be easy to read and not require lots of time to decipher.
    c. How to improve reuse and efficiency of code
        i. Ex: Creating functions to reduce reuse of code over and over again.
    d. How to use existing python packages
        i. Ex: stats.model would be a better idea here than scikit learn, etc. or this package wasn't mean to originally do this, you should try this other one instead.
- A Model of approaching code reviews is a **compliment sandwich:**
    - Compliment, critique, compliment. Be positive.

Technology Review:

- Give enough background so that Stephanie and Dave can understand why we are looking at a technology.
    - For example: Goal of project is to estimate boiling points, so using machine learning technologies on the properties of different molecules and their respective boiling points is needed to predict that.
    - Quick, concise. 30 seconds
- Discuss 2 to 3 pieces of software you can import (like tensorflow, etc.), how they work, what your appeal was for using these packages (why use it), what are the drawbacks of using the package?
    - Recommended: create some test cases to determine appeals and drawbacks for different packages.
    - When evaluating a package, look for examples on github. For example, for Folium, if there is an example on Github that does exactly what you want it to, then it is very appealing and advantageous to use this package.
    - Evaluate the issues of the issues on the Github to determine how people use the package, relevance of package, etc.
    - Maximum 5 minutes per team ***DUE THURSDAY 2/15/2021*****.**

Standups:

- Towards the end of the quarter
- 1-2 minute standups.
- Off the cuff, quick and dirty updates on project.
    - Progress made since last update and how that compares with plan.
    - How to compensate to reach deadlines if progress has not met planned progress.
- Deliverables that you want to shoot for for the next period.
    - What are your goals for next period?

- Challenges to making of next deliverables for next period
    - Technological uncertainties / hiccups
        - Ex: Can't get decision tree to predict anything other than the mean.

Software Design:

- Feels like this is basically an outline of the goal of our software and the code we need to accomplish this.
- Use case pieces and component pieces.
- Design outlines the use cases and reviewed the components that are necessary to realize those use cases.
- Design means something you leave behind to describe the goals of the software and what set of primitives are necessary to achieve those goals.
- Good designs have few components and clear roles that are outlined.
    - Good designs also have few interactions between components (1-2, not 4 - 5).
- Creating a software design allows us to approach our complex goal
    - Helps finds potential bugs before we code
    - Enables team to work in parallel and separate portions of work
    - Promotes testability

Steps in Design:

<u>Draft of this due Friday 2/19/2021:</u>

1. Functional Design:
    a. Describe what system does (use cases)
2. Component Design:
    a. Specify the components that are necessary to accommodate that use case
    b. These can be things that we don't actually write such as a database, a web server, etc.
    c. Each use case has a top level component
        i. For example compute pairwise correlation from homework.
            1. Top level use case is pairwise correlation. Each use case has a top level component.
            2. Inside pairwise correlation, other functions are called such as row_i_to_row_j. So that is another component.
    d. Goal is to flesh out components and sub-components organically. **Not all of your code should be in a gigantic component or else it will be impossible to test efficiently.**
        i. We want many components that are common across use cases
3. Specification of Components:
    a. Components need to be described with sufficient detail so that someone with only modest knowledge of our project can implement the code for the component.
        i. Name
        ii. What the component does
        iii. Inputs (with type information!)
        iv. Outputs (also with type information)
        v. How does the component interact with the other components in the system and what other components does the component being described use? (For example, a function calling another function).
            - Something we could talk about here are those external databases that this component refers to. For example, for the ATM use, a database of information of customers must be referred to in order to verify the user's pin and card number. This can be an external component database.

        1. Level of detail that we describe these external components is based on level of control we have over them.

    vi. Write pseudo code of components
    vii. Helps layout inputs, outputs, etc. Clarifies components.

Use web sequence diagrams to draw a flow diagram with interactions out.
sequencediagram.org