AI Technician Capstone
Cohort 3

June 5, 2023

Contents

Phase 0

Overview

Phases

The AI Technician capstone will be conducted in 4 phases over 18 weeks. Each phase will begin with a 1-hour kickoff meeting that will orient you to the phase and discuss important considerations for the phase. The phases include:

Phase 1: Exploratory Data Analysis (2 weeks)

Phase 2: Machine Learning Modeling (6 weeks)

Phase 3: Web Application Development (8 weeks)

Phase 4: Final presentation (2 weeks)

Capstone Deliverables

The deliverables from this capstone will be:

1. 1x Design document

2. 1x Hosted web application

3. 2-3x AI models (1 per team member)

4. 1x Presentation

Workload

You will have at least one full day per week to work on your capstone. Each week has an associated deliverable that must be sent to your AI Professional mentor by **Friday at 1200**.

Mentorship and Progress Checks

Each capstone team will be assigned an AI Professional mentor. The mentor will formally meet with the team at least once every 3 weeks to review progress and provide guidance. Additionally, the deliverables specified for each week will be delivered to the team mentor for weekly review.

Seeking Help

Your assigned AI Professional mentor will be available to provide help on an as-needed basis. When meeting with the mentor, for each question you have, bring a written document that answers the following questions:

1. What is the issue?

2. Why is it not working?

3. What could be the problem?

    4. What have you tried to solve the problem?

The purpose of answering these questions is to facilitate troubleshooting on your own and to help clearly identify the problem you are encountering.

Individual and Group Tasks

In each phase, there are individual and group tasks. The individual tasks are denoted by `[Ind]`, and the group tasks are denoted by `[Grp]`.

Teams

Capstone teams will consist of 2-3 AI Technicians and are based on current portfolio assignments.

O&I	Mentor
CW2 Gonzalez SPC Lacovara SPC Fortuna	CPT Nelson

I&P 1	Mentor
CW2 Vickers SFC Schulze SPC Morales	CPT Kuehnert

I&P 2	Mentor
SFC Thames SSG Malagon SPC Tarila	CPT Leinweber

Sustainment	Mentor
1LT Cheng-Bustamante SSG Nguyen SGT Mangual	CPT Kuehnert

ATR & Soldier Performance	Mentor
1LT McGrath 1LT Owens SFC Thompson SSG Kuewa	CPT Leinweber

XVIII ABC 1	Mentor
SFC Belyayev SSG Shields SSG Prickett	CPT Wilkins

XVIII ABC 2	Mentor
1LT Robinson SSG Ballew	CPT Wilkins

Phase 1

Exploratory Data Analysis

Learning Objectives

LO1. Get familiar with a given dataset

LO2. Produce useful visualizations

LO3. Practice cleaning and formatting data for a data science task

Scenario

You are a data scientist at a small bank that is trying to get more customers to subscribe to term deposits. In an effort to increase the number of subscriptions, the bank has been running a telemarketing campaign. You have been asked to analyze the data collected so far in the campaign and use it to inform which customers the bank should call next week to maximize the number of subscriptions.

Introduction and Deliverables

In this phase, you will analyze the provided Bank Marketing dataset. The data was collected from direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. The classification goal is to predict if a client will subscribe to a term deposit (this is denoted by a "yes" or a "no" in the last column of the dataset). The bank wants clients to subscribe to their term deposit.

Week	Task	Deliverable
1	Download and clean the dataset, understand the features, produce visualizations of the data, identify key features, construct new features	Jupyter notebook
2	Write-up of project overview and Phase I and review with AI Professional mentor	Design document

Table 1.1: Deliverables in Phase I. Progress review weeks are highlighted.

Tasks

Instructions:

1. **[Ind]** Download the dataset from the AI2C GitHub here. You can clone the entire repo using:
   `$ git clone https://github.com/AFC-AI2C/ai-tech-capstone`

   (a) Read the `README.md` file describing the dataset you just downloaded.

   (b) For this phase, you will be using the `train.csv` data.

2. **[Ind]** Open a Jupyter notebook and use Pandas to read in the dataset. Print the head of the dataset to confirm that is was read in correctly.

   (a) If you need to install Pandas, read this documentation.

3. **[Ind]** Produce summary statistics for the dataset (reference: here)

   (a) Analyze the summary statistics. How many columns are in the summary statistics DataFrame? Why are there fewer columns in the summary statistics DataFrame than in the training dataset?

   (b) What were the ages of the youngest and oldest people called?

   (c) What was the average duration of the calls?

4. **[Ind]** Produce histograms for the `age`, `job`, `marital`, and `education` features (reference: here).

   (a) To start, try to use this Pandas histogram command. This should work for the `age` column, but not for the `job`, `marital`, and `education` columns. Why is this? Figure out another method for producing the histograms for these columns.

   (b) Based on visually inspecting the histograms, what is roughly the ratio of married customers to single customers?

   (c) What is the most common job among the customers?

5. **[Ind]** What percentage of customers subscribed to the term deposit?

6. **[Ind]** Print the record at index 10000.

7. **[Grp]** Answer the following questions in your design document (write the code used to obtain these answers in your Jupyter notebook):

   (a) The labels are the rightmost column of the dataset. What is the data type of the labels (ex. integer, float, boolean)?

   (b) How many records are in the dataset?

   (c) How many features are in the dataset?

   (d) Name 5 features that you think might be most useful in your analysis. What are the data types of the features you identified?

   (e) Are there any missing entries? If so, how are you going to handle the missing entries?

   (f) What is the age, job, marital status, and education level of the record at index 10000?

   (g) Look at the `contact` feature. How many calls were made by telephone? How many calls were made by cellular?

   (h) What month corresponds to the highest success rate for getting clients to subscribe? How did you determine this?

   (i) What day of the week corresponds to the highest success rate for getting clients to subscribe? How did you determine this?

8. **[Ind]** Drop the following columns: `duration`, `contact`, `month`, and `day_of_week`.

9. **[Ind]** How many single people younger than 40 were called? (reference: here)

   (a) Of these customers, how many of them subscribed to the term deposit?

   (b) Does this demographic subscribe to the term deposit more or less than the average for all of the customers?

10. **[Ind]** How many customers are single, older than 40, and have a university degree?

   (a) Of these customers, how many of them subscribed to the term deposit?

   (b) Does this demographic subscribe to the term deposit more or less than the average for all of the customers?

11. **[Ind]** Convert categorical variables to one-hot encodings.

12. **[Grp]** Answer these administration questions in your design document:

   (a) Who is in your team?

   (b) Who is your AI Professional mentor?

   (c) What day of the week is your "capstone day"?

   (d) Where are you going to work on your capstone?

   (e) What skills are you most comfortable with after the AI Technician course?

   (f) What skills are you least comfortable with after the AI Technician course?

13. **[Grp]** Answer these reflection questions in your design document:

   (a) Do you believe you accomplished the learning objectives?

   (b) What tasks were most helpful for accomplishing these learning objectives?

    (c) What challenged you during the phase? How did you solve this? If nothing challenged you, what could be added to this phase to make it more interesting/challenging?

14. [Grp] Write these sections in your design document:

    (a) Administration - see Question 12

    (b) Introduction - what problem are you working on?

    (c) EDA Findings - what did you learn during the EDA? This is where you will put the answers to the questions listed above. This section should also include figures that support your analysis.

    (d) Phase Reflection - see Question 13

Phase 2

Machine Learning Prediction

Learning Objectives

LO1. Become familiar with machine learning capabilities and tasks

LO2. Tune hyperparameters to improve model performance

LO3. Perform model selection under constraints for an applied problem

Scenario

After analyzing the telemarketing campaign data, you have been asked to use the data to help the bank prioritize the customers to call next week. The `train.csv` data is the historical data so far in the campaign, and `test_X.csv` are data on potential customers to call. You will build a machine learning tool to decide in which order the bank should call those potential customers to maximize the number of term subscriptions gathered by the end of the week. The bank has only allocated 120 call-hours for telemarketing next week.

Introduction and Deliverables

Now that you have prepared the data, you are going to use it to provide predictive analytics for the bank telemarketing campaign. There are many questions we could try to answer with the dataset, but we are going to focus on predicting whether or not someone is going to subscribe to a term deposit. Specifically, the bank has a list of approximately 8,000 clients and wants to know in what order they should be called given that there are a limited number of labor-hours available for the telemarketing campaign.

This phase is scheduled to take 6 weeks with tasks and deliverables as shown in Table 2.1.

There are three main deliverables for this portion of the capstone.

1. An organized Jupyter notebook with all specified tasks. This notebook should contain all the steps needed to train your models and provide inferences on the test set for submission.

2. Inferences on the test set. The inferences should be a single column in the same order as the test data set. You will submit your results to your group mentor in CSV format.

3. A well-structured write-up of the phase. This document will include the questions you answer throughout the Phase as well as intermediate results such as plots.

Week	Task	Deliverable
3	Build model 0	Jupyter notebook
4	Fine-tune hyperparameters of model 0 and submit inference results	Inference results
5	Review with AI Professional mentor	Document
6	Build candidate models with hyperparameter tuning	Jupyter notebook
7	Model selection and final inference results	Inference Results
8	Write-up of Phase II and review with AI Professional mentor	Document

Table 2.1: Deliverables in Phase II. Progress review weeks are highlighted.

Week 3 - Building a Simple Model

As a first step towards answering the bank's question, you are going to build a simple predictive model. This is meant to familiarize you with the scikit-learn library and making predictions with data.

Load the Data

The first step is to load data to train the model and load the test data features to provide predictions on unseen data.

1. **[Ind]** Use Pandas to read the training data into a data frame. This training data should be the output of your work in Phase I.

2. **[Ind]** Split the training data into a data frame without the column "y". These are the features you will use to predict the target variable "y".

3. **[Ind]** Make a one-column data frame of just the column "y" and encode it so that "yes" has the value 1 and "no" has the value 0. You will build a model to predict whether an observation is 1 or 0.

4. **[Ind]** Randomly split the training data into a training and validation set so that 25% of the data is held-out for validation.

5. **[Ind]** Use Pandas to read the test data into a data frame. Note that this data does not have the "y" feature value. You will provide predictions on these features to evaluate how well your model works on data that was not seen during training. Make sure that you apply the preprocessing that you did in Phase I on this data (e.g. one-hot encoding).

6. **[Ind]** Rebalance the training data so that there are equal observations in each class (equal "yes" and "no" y values). There are a few ways to do this, but we recommend using SMOTE. You should have seen in Phase I that there is a large class imbalance with only about 10% of observations falling into the "yes" class. This can be problematic when training ML models on the data because a naïve model that always predicts the majority class can score very well — 90% accuracy in this case!

Fit a Logistic Regression

The first model you will use to classify the data is an unregularized logistic regression. This is a popular model for binary classification tasks like this one, and it serves as a good baseline for a comparison against more sophisticated models. This model makes strong parametric assumptions that the log-odds of the dependent variable is a linear combination of the independent variables.

1. **[Ind]** Create a logistic regression model object with no penalty. We will discuss and use penalties next week.

2. **[Ind]** Fit the model using the training data.

3. **[Ind]** Make predictions on the test data.

4. **[Grp]** Report the accuracy of the model on the training data. Do you think that this accuracy will be similar to the accuracy on the test data? Explain your reasoning.

5. **[Grp]** Now report the accuracy of the model on the validation data. Do you think that this accuracy will be similar to the accuracy on the test data? Explain your reasoning.

6. **[Grp]** Write a function that takes a model, training data, and validation data and prints out a summary of the model performance on each set. At a minimum you should display accuracy and AUC-ROC, but you might want to consider including the built-in classification report. Evaluate your model with the function and add the result to your document.

Week 4 - Hyperparameter Tuning

This week you are going to build off-of the logistic regression model to produce a *regularized* logistic regression. In logistic regression, you can tune the bias and variance of your estimator by penalizing the coefficient weights. This allows you to tune the model to your specific data problem to achieve a lower expected prediction error. The typical ways to penalize the coefficients are to use the L1 norm (LASSO), L2 norm (Ridge), or a linear combination of L1 and L2 norms (elastic net). For this problem, you are going to use an L2 penalty.

Now that you are using a regularized logistic regression, you have to tune the regularization hyperparameter λ to the data. This parameter determines the strength of the penalty term, and when $\lambda = 0$ the model is unregularized. By changing λ, you will change the bias and variance of your estimator which will impact your predictive performance. You will use the training data to determine which value of λ works best. This process is called *hyperparameter tuning*.

In order to tune the λ hyperparameter, you need a way to estimate the prediction error on unseen data. Two common ways to do this are bootstrapping and cross-validation. In this exercise, we will be using k-fold cross-validation to choose our hyperparameters. If you are interested in learning more about cross-validation we recommend reading the *scikit-learn* documentation on the topic.

Grid Search

1. **[Ind]** Instantiate a standard scaler to scale each feature in the data to a standard normal distribution. This is essential in regularized logistic regressions because they are *not* scale invariant. If we do not do this, then features with larger values will be more heavily penalized by the regularization term.

2. **[Ind]** Instantiate a new SMOTE object.

3. **[Ind]** Instantiate a logistic regression object like in the previous week, but this time assign it an L2 penalty.

4. **[Grp]** Combine the SMOTE, scaler, and logistic regression objects (in that order) into a pipeline. Explain why we need to do this rather than scaling and resampling all of the training data before our cross-validation.

5. **[Ind]** Run a grid search cross-validation to evaluate different hyperparameter choices for C in the logistic regression, where C is the multiplicative inverse of the λ parameter we discussed earlier. You will need to decide what values of C to try, the number of folds k, and the scoring metric. We recommend talking to your group mentor about considerations for these choices.

Evaluate Cross-Validation Results

After you run the grid search cross-validation, you should interpret the results. You are going to create a figure to visualize the results like the one shown in Figure 2.1.
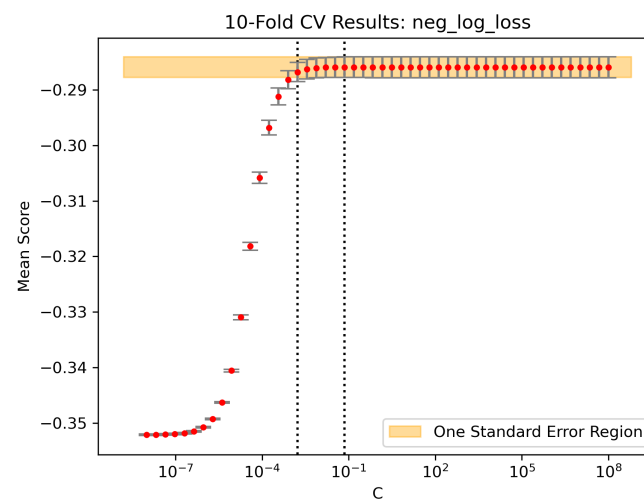

Figure 2.1: Cross-validation results from the logistic regression model. Error bars indicate one standard error of the mean. Note: the scoring metric used to produce this plot was neg_log_loss.

1. **[Ind]** Write a function that takes the `GridSearchCV.cv_results_` from your grid search as an argument.

2. `[Ind]` In the function, plot the mean score as a function of C.

3. `[Ind]` Now have the function return the value for C that achieves the maximum score. If there are ties, choose the C with the smallest value (most regularized).

At this point, you have enough information to pick a good hyperparameter; however, you have only used the mean of the score without incorporating the variance of the score across the folds. Over the next few steps, you will incorporate the variance of the cross-validation results to apply a common heuristic in logistic regression called the one standard error rule.

1. `[Ind]` In the same function as before, calculate the standard error of the mean score by dividing the standard error of the scores by the square root of the number of folds.[1]

2. `[Ind]` In the plot, add error bars that show the standard error of the mean scores.

3. `[Ind]` In the function, in addition to the C that corresponds to the best mean score, return the smallest C that is within one standard error of the optimal C. This choice of C gets you a model that is more regularized than the model with the best mean score but at a statistically similar performance. This is a heuristic called the one standard error rule that chooses to have a simpler model at the cost of slightly worse performance.

4. `[Ind]` In the plot, add a band that shows the region for the one standard error rule (see Figure 2.1). Pyplot's fill_between may be useful.

5. `[Ind]` Add the resulting plot to your document and write a few sentences interpreting the plot. You might want to think about how regularization relates to performance and which C you would choose (one standard-error vs maximum).

Inference on the Test Set

Now that you have done hyperparameter selection, you will fit a final model and provide predictions on the test set.

1. `[Grp]` Pick which C you want to use from your grid search. This should be one of the two values that you return from the function you wrote.

2. `[Grp]` Fit the model to all the training data with your chosen C and do not forget to scale the data! Explain why you should fit your model to all the training data at this step.

3. `[Grp]` Report the performance of the model on the validation data using the function you wrote previously.

4. `[Grp]` Now use the model to predict the labels on the test data and submit the results to your group mentor. Make sure to keep the original order of the test data in your submission.

---

[1]There are a couple caveats here that should be addressed. First of all, this calculation of standard error is not entirely correct because it assumes that the fold scores are independent; however, due to the overlap in training data across the folds, this assumption is incorrect. Secondly, if you use the "std_test_score" from `cv_results_` then you will be using a biased estimate of the sample standard deviation that underestimates the true population standard deviation. You can partially address this with Bessel's correction.

Week 5 - Review with AI Professional Mentor

Review the work so far with your group mentor. You should have an established baseline model with tuned hyperparameters. You should be able to use this model to provide inferences on the test data. Once these requirements have been verified, discuss your responses to the questions asked throughout the instructions.

Week 6 - Build Candidate Models

Over the previous two weeks, you built a good baseline model for prediction. Now you have a point of comparison for more sophisticated approaches which we expect to provide better performance, though often at the cost of higher computational complexity. This week, each member of the capstone team is going to build their own classifier with a different algorithm. Use the previous weeks' instruction to guide you in the process.

Here are some options to get you started.

- Naive Bayes

- K-Nearest Neighbor

- Decision Tree

- Random Forest

- Gradient Boosting

- XGBoost

At this point you have a candidate model for predicting whether someone will subscribe to the term deposit. Now you need to use the data to tune hyperparameters with that model. Follow the same process that you used to tune the L2 Logistic Regression with grid search and k-fold cross-validation; however, your models might have multiple parameters that you should consider tuning. Be careful with adding too many parameters to the grid search because it will search every combination of parameter value that you provide. If you decide to search over multiple parameters and many values for each parameter, consider using a random search.

Note: Hyperparameters are not constrained to arguments that you explicitly pass to your model. They also include data pre-processing, resampling (e.g. SMOTE), and any other transformation you make on the data. We recommend that you treate resampling as a hyperparameter. To simplify this process, produce a tuned candidate model with and without resampling to be evaluated in model selection next week.

Week 7 - Model Selection

At this stage, your group has multiple candidate models, but how should you decide which one is the best? This is something that you will encounter any time you are trying to solve a problem with machine learning. You need to be able to quantify the performance of models for the particular task. In this case, you have been given a list of ~8,000 potential customers and you need to rank them in the order that they should be called to get the most subscriptions with the constraint that the bank only has 120 call-hours. For this problem, assume that these 120 hours only apply to time on the phone with the potential customer and not on time between calls or waiting for the customer to pick-up.

Exploratory Analysis

In order to measure predictive performance with constrained call-hours, you will need information about the call durations. As a first step, you will plot the call durations in the training data set.

1. [Grp] Make a kernel density plot of call duration for customers that subscribed to the product.

2. [Grp] Make a kernel density plot of call duration for customers that did not subsribe to the product.

3. [Grp] Overlay the two figures and add it to your document.

4. [Grp] What can you conclude from the figure about the call durations? Why do you think the distributions are different?

5. [Grp] How do the call distributions affect the performance measures that you care about in a model? How harmful are false positives? False negatives?

Custom Scoring

Now you are going to write a custom scoring function that calculates the number of subscriptions a model would have produced given a specified call-hour constraint.

1. The function will have the following parameters.

   - Model
   - Feature matrix X
   - Target vector y
   - Call duration vector
   - Call-hour limit

2. Have the model predict the probability that each observation will subscribe to the product.

3. Sort the target and call duration vectors by the predicted class probabilities from largest to smallest.

4. Iterate through the sorted predictions and call durations to determine how many subscriptions you would have generated in the given call-hour limit.

Model Selection

Now you are going to use your scoring function on the validation set to perform model selection.

1. [Grp] Score the baseline model.

2. [Grp] Score all candidate models.

3. [Grp] Which model did best? How does it compare to the baseline model?

4. [Grp] In addition to a simple baseline model like a logistic regression, we often want to compare against a model that randomly guesses. Make a class called "RandomClassifier" that produces random guesses for binary classification (0 or 1) based on the training data class proportions. Make sure that the class implements the "predict_proba" method to be compatible with the scoring function. How do your models compare to the random classifier?

Final Model

You should now have a model that you selected as performing the best on the validation set. If you wanted, you could safely put this model into production and get reasonable predictions, but remember that you have a set of validation data that was never used to train the model. If you want to give your model the best chance to succeed on unseen data, you should refit on *all* the data.

1. `[Grp]` Rerun your winning model's entire training pipeline, including hyperparameter selection, on the entire data set.

2. `[Grp]` Use the model to predict class probabilities on the test data.

3. `[Grp]` Use the predicted class probabilities to assign the rank order (starting at 0) to the test data. These are what you will submit to evaluate your model on the test data.

Week 8 - Write-Up

At this point, you have a machine learning model trained to rank-order customers for the telemarketing campaign and are ready to put it into production. Take this week to clean up your Jupyter notebook, verify that your models are properly trained, finish your write-up, and discuss the write-up with your group mentor. This week each team will brief LTC Anderson on their results from Phase 1 and Phase 2 of the capstone.

Presentation Requirements

1. `[Grp]` Each team will sign up for a 15-minute presentation slot with LTC Anderson using the following link.

2. `[Grp]` Prepare a 15-minute presentation that includes discussion of the following (note: your presentation will discuss your work done in each week of the capstone so far):

   (a) Problem Description/Motivation (Review material from Weeks 1-2)

   (b) EDA Findings (Review material from Weeks 1-2)

   (c) Logistic Regression Model (Review material from Week 3)

   (d) Hyperparameter Tuning (Review material from Weeks 4-5)

   (e) Custom-made Model(s) (Review material from Weeks 6-7)

   (f) Discussion of how you selected your group's model (Review material from Weeks 7-8)

   (g) Lessons learned during Phases 1 and 2 (Review material from Weeks 1-8)

3. `[Ind]` Meet with your team mentor to review your progress and get caught up as necessary. All individuals should be caught up through Week 8 by the end of this week in order to start Phase 3 on time on Monday June 5, 2023.

4. `[Grp]` Deliver your 15-minute presentation to LTC Anderson on Thursday June 1, 2023.

Phase 3

Web Application Development

Scenario

Introduction and Deliverables

In this phase, you will build a cloud native, AI-enabled web application. You will deploy the models you developed in Phase 2 using simple Flask applications for online predictions, build a Django application to display your results, and instrument all of these using open source tools. This pattern of separate, containerized applications gives you and your team development flexibility and allows you to update and scale components individually.

Week	Task	Deliverable
9	Create a Flask application (one per team member) to respond to HTTP requests with prediction results	3 Flask apps
10	Create a Django web application	Django App
11	Containerize your applications	5 Container Images
12	Instrument your applications using Prometheus and visualize with Grafana	Grafana Dashboard
13	Deploy your web application in LTAC	Deployed Application
14	Web user interface development	Landing Page
15	Flex week: Catch up, or add a new feature to Django application	Django Application
16	Flex week: Catch up, or add a React.js component to your web page	React Component

Table 3.1: Deliverables in Phase III. Progress review weeks are highlighted.

Setting Up Your Development Environment

Python, Git, and Directories

Before you start creating your application, there a few things that will make it easier and faster for you to develop as an individual and member of a team.

1. Directories: Computers and spaces don't go well together. Many of the steps that we will guide you through in the upcoming weeks use the terminal (Mac), shell (Linux), or PowerShell (Windows). If you have used spaces in the directory path it will be much harder to navigate your file system in the terminal and some development tools will be more challenging to configure. With that in mind, a few recommendations to make it easier for you to work in the terminal. If you prefer Windows but struggle with PowerShell you can alternatively consider Git Bash to enjoy many of the features of working in a Mac or Linux environment on a Windows OS.

    i. lowercase everything. usernames, filenames, directory names are all easier to type if you avoid capital letters. (There are exceptions to this, and even some languages where it isn't the standard practice, but you should use those by exception only)

    ii. single word. Name your directories and files with a single word. Generally, try to pick ones that differ in the first few characters. It will make it faster for you to type and you can use tab completion to avoid mistakes and speed up your navigation.

    iii. snake case. If you must use multiple words in a file or directory name, separate them with underscores, these interrupt your ability to read less than hyphens. Never use spaces.

2. Editors: We recommend you use Visual Studio Code (VSCode) as your code editor. There are many editors available and the choice of which you want to use is highly preference based. Until you are experienced in the language you are working in, we **do not** recommend you use a fully featured Integrated Development Environment (IDE) like PyCharm. A more simple text editor is better for learning the fundamentals. VSCode can grow with you as you can add extensions that speed up your development but at its core, it is just a text editor.

3. Python: There are many versions of Python available and currently supported. For most of what you do, the version will not matter so you should pick the one that is available in your **deployment** environment. If it isn't specified, generally use the latest stable release. If you are using Mac or Linux, use a package manager like Homebrew, apt, or yum.

4. Virtual Environments: Virtual environments are a critical tool for Python developers. The language comes with a builtin virtual environment creation tool – venv – we recommend you use for all your Python work. In week 9 we will provide you with the basic commands to get started in a virtual environment, but for now a few first principles.

    i. environment name. Name your virtual environment the same thing in every project, '.venv' is the standard naming convention.

    ii. create once, initialize always. Only create your virtual environment when you start a new project, initialize it everytime you work on it. Some editors (like VSCode) will do this for you automatically if properly configured.

5. Source Control: To be an effective member of a team that builds software (and realize all machine learning resides in software) you must be familiar with Git. There are books written on the topic and we will not replicate that here. We highly recommend you read, at minimum, Chapters 1 & 2 in Pro Git. We will provide you with commands to guide you in working with this tool over the coming weeks, but you will need to expand your knowledge of this tool to be an effective contributor to your team.

Getting Started

In Phase 3, your team will work together on a shared Git repository to complete each week's assignments. Only one of you needs to follow steps 1-9 below to create your team's git repository. The remaining team members should complete steps 10-13. (Note: All team members will need to execute step 5 upon cloning the repository, and before they begin working on this phase of the project)

1. Create a new directory on your computer named **capstone**. We recommend putting this in an easy to access location on your computer (i.e. `/Users/<username>/Documents/computing/capstone`)

   `mkdir capstone && cd capstone`

2. In your newly created directory, initialize a git repository. If you have not already, you will need to install git.

   `git init --initial-branch=dev`

   Here we specify dev as the initial or default branch. We use dev to align the terminology between deployments/environments and the branch names in our source code repositories.

3. Open this directory in Visual Studio Code.

4. Create a file in this directory named `README.md`. In that file, add the following pieces of information and save the file.

   i. A Team Name
   ii. Your name

5. Create a Python Virtual Environment. In the integrated VSCode terminal window run:

   `python -m venv .venv`

   Once created, you can **activate** this virtual environment as follows:

   (Linux/Mac) `source .venv/bin/activate`

   (Windows) `.venv\Scripts\activate`

6. Create a new file in your directory named `.gitignore`. In this file, add the following line:

   `.venv`

   This tells git to recursively ignore any files in the .venv directory.

7. Commit these files to your git repository. Before doing so, check the status of changes in your working directory.

   `git status`

   You should see something like the following in your terminal. If not, revisit the previous steps.
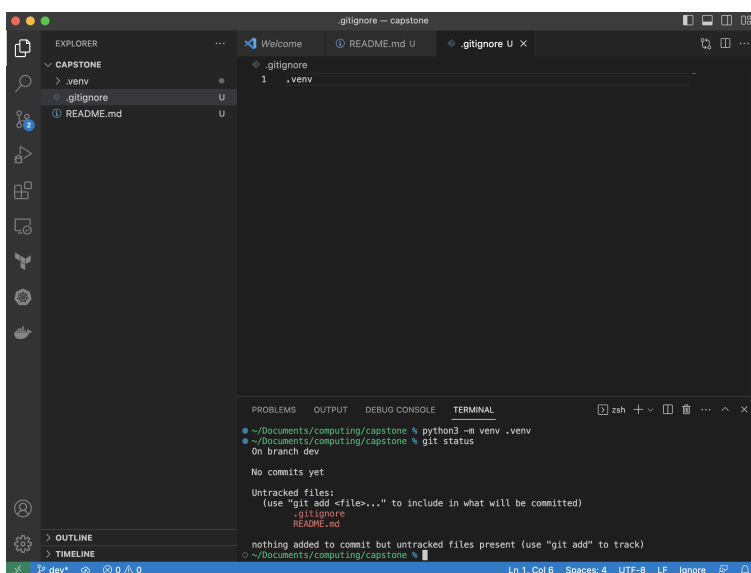

Figure 3.1: git status output after creating new files

If your output looked like the screenshot above, run the following to "stage" your changes.

```
git add README.md
```

```
git add .gitignore
```

Now that you have added the file to your 'staged' changes, when you run `git status` again you will see a different outcome, verify that now:
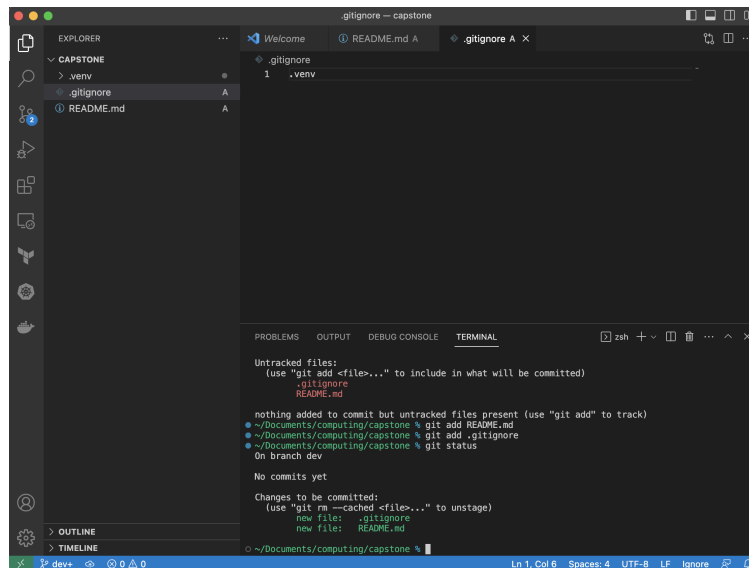
```
git status
```


Figure 3.2: git status output after staging your work

If your output looked like the image above, commit your changes.

```
git commit
```

Note: if you have not used git before, this command will fail and you will need to configure your settings as follows before you re-run the above command.

```
git config --global user.name <Your GitHub username here>
```

```
git config --global user.email <Your GitHub email here>
```

Note: A commit message or "gist" should follow the following format. First, there should be a short, single line summary which will serve as the commit title. Then, after an empty line, a sequence of short descriptions should describe every change (there should only be a small number included in each commit) included in the commit. Example below.

    Initialize Project

    Created our team's initial README.md

As you can see, this commit message is succinct, but clearly describes what was changed in the repository in this commit. Beyond serving as an esoteric reference of past work, this makes reviewing different versions of work and tracking down future bugs faster and more reliable on your team.

8. Create a remote repository. While you would typically do this yourself (and select the repository location best suited for your team's access and development needs) we have created a remote repository for you in the AFC-AI2C GitHub. Insert the link below where specified in the below command. This command specifies a remote repository that your newly created git repository will 'track' at the specified URL.

```
git remote add origin <remote repo link here>
```

   i. `https://github.com/AFC-AI2C/capstone-team-1.git` (CW2 Gonzalez, SPC Lacovara, SPC Fortuna)
  ii. `https://github.com/AFC-AI2C/capstone-team-2.git` (CW2 Vickers, SFC Schulze, SPC Morales)
 iii. `https://github.com/AFC-AI2C/capstone-team-3.git` (SFC Thames, SSG Malagon, SPC Tarila)
  iv. `https://github.com/AFC-AI2C/capstone-team-4.git` (1LT Cheng, SSG Nguyen, SSG Mangual)
   v. `https://github.com/AFC-AI2C/capstone-team-5.git` (1LT Owens, SFC Thompson, SSG Kuewa)
  vi. `https://github.com/AFC-AI2C/capstone-team-6.git` (SFC Belyayev, SSG Shields, SSG Prickett)

vii. `https://github.com/AFC-AI2C/capstone-team-7.git` (1LT Robinson, SSG Ballew)

9. Push to the remote repository. This command pushes your changes to the specified remote repository, setting the local branch called `dev` to "track" the remote branch with the same name in the remote repository named `origin`.

   `git push --set-upstream origin dev`

10. Now that you have a repository setup, each additional team member needs to setup the directory by "cloning" this repository as follows.

    In a shell, command prompt, or terminal, navigate to the directory you want to store Phase 3 work in (suggested: `/Users/<username>/Documents/computing`) and run the command below. You can find the repo link in your team's link above under the green "Clone" button.
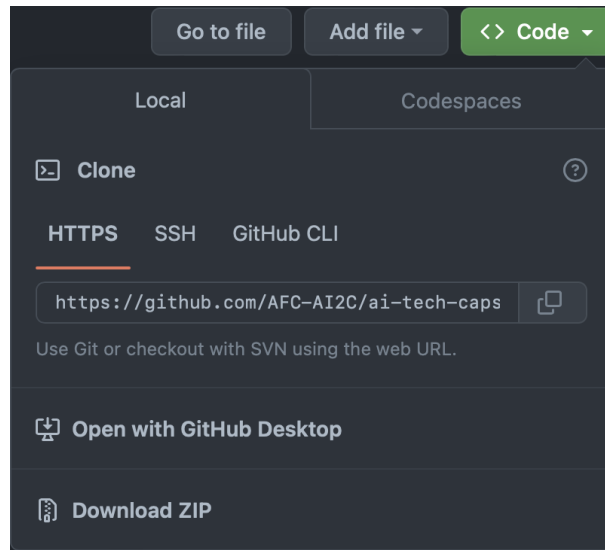

Figure 3.3: GitHub repository clone button

    `git clone <remote repo link here> capstone`

    This will create a directory called "capstone" that will store the contents of your team's remote repository. Open this directory in VSCode.

11. Create a new branch. Open the integrated terminal in VSCode and run:

    `git checkout -b add-<your name>`

    You have now created a place you can make changes to your team's source code safely and in parallel with your other teammates.

12. Open the README.md file and add your name to the list of names. Save the file.

13. Run the following sequence of commands to stage, commit, and push your work.

    i. `git status`
    ii. `git add README.md`
    iii. `git commit`
    iv. `git push -u origin add-<your name>`

14. Open your team's repository in GitHub and create a pull request from your branch to dev.

15. After another team member reviews your work, merge your work into dev. Once you, or anyone else on your team starts to work on the project again, start first by running `git pull` to fetch the latest changes to your base branch, dev, before you create and checkout a new branch.

Best Practices for working with Git

1. Commit and push your changes whenever you either a: finish a major change/feature or b: finish working for the day.

2. Have a team mate to review your work; another set of eyes and a different perspective are always helpful.

3. Be explicit in actions. Only add files you are sure you want to add. Use `git status` to check yourself and your work before you commit.

Git Command Sequence Reference

The following references all assume you are working in the terminal and have navigated to the base directory for this project `capstone`.

Create and Checkout a New Branch

Over the next two months of capstone, we will start each task telling you to "Create and Checkout a New Branch". We summarize that process below for your reference. This assumes you have cloned your team's git repository and that you have navigated to your 'capstone' directory before you run any of the commands.
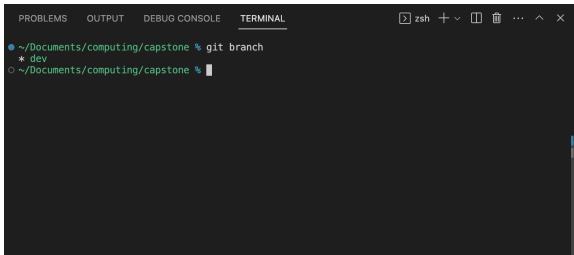
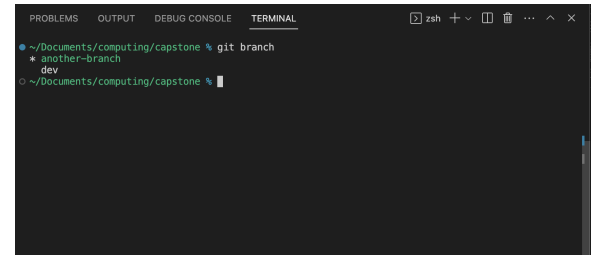i. Verify you are on the branch, 'dev'.

`git branch`

Figure 3.4: good

Figure 3.5: bad

If you were not on the branch dev, checkout dev now.

`git checkout dev`

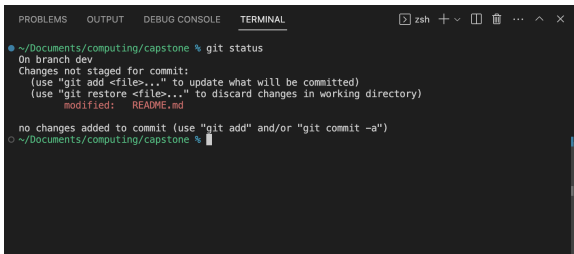ii. Verify you do not have any unsaved work.
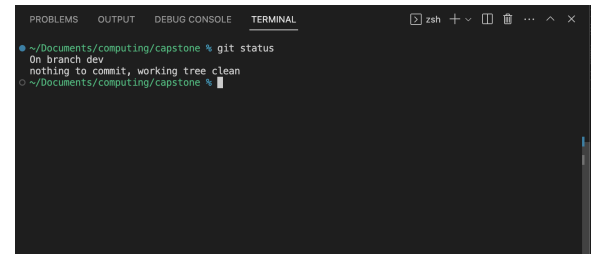
`git status`

Figure 3.6: good

Figure 3.7: bad

If you do not have any changes present, then proceed to pull the latest changes. If you do, resolve this by saving your work.

iii. Pull the latest changes from your remote repository.

`git pull origin dev`

iv. Create and Checkout a New Branch.

`git checkout -b <new_branch_name>`

v. Begin working on your new changes

Stage, Commit, and Push your Changes

Below is the sequence of commands we refer to when we say to "stage, commit, and push your changes".

i. Verify which files you have changed.

ii. Optionally inspect which changes are present with git diff

   iii. Add the files you wish to check in to your repository with git add ¡filename¿

   iv. Run git commit and add a descriptive commit message

   v. Run git push (if it is the first time you have pushed from this branch, add the –set-upstream flag)

   vi. Continue working on your branch.

Open a Pull Request and Merge when Ready

When we say to "open a pull request and merge when ready", we are referring to the following steps.

   i. Verify you have pushed all the lastest changes from your branch to your remote repository.

   ii. Open your team's repository in GitHub

   iii. Open the pull requests page

   iv. Select "New Pull Request"

   v. Set the base branch to dev and specify your newly created branch to merge into it

   vi. Add a teammate as a reviewer and let them know you have opened a PR

   vii. Once a teammate reviews your code, merge using the GitHub gui or manually in the terminal

   viii. Resolve any merge conflicts, should they arrise

Week 9 - Online Prediction

In previous weeks your team built multiple models and by now have identified the best performing one for your team. After sharing your initial results, you've been asked to make this model available for recurring use and future predictions. While there are many ways to share models, a common approach is to build a simple web app using a tool like Flask. These applications expose model inferencing abilities behind a web interface where an HTTP request is used to trigger a prediction from your model and returns an HTTP response with the prediction result. This week, your team will build three Flask applications (one per team member) that each host a model. One model should be the best one from Phase 2 and the other two should use the two starter notebooks provided to you in the starter code for this week.

Learning Objectives

LO1. Prepare a machine learning model for inference

LO2. Identify model input requirements

LO3. Compare HTTP methods and their utility for online inference

LO4. Define and run a Flask application responding to an HTTP request

LO5. Define an API end point using one of the above request types

Tasks

Instructions:

1. **[Ind]** Each member of your team should follow the following steps to prepare your environment for this week's tasks.

    i. Open your capstone directory in VSCode. Ensure you activate your team's virtual environment as follows: (Linux/Mac) `source .venv/bin/activate` (Windows) `.venv\Scripts\activate`

    ii. Create and checkout a new branch

    iii. Create a new directory for a Flask application using the naming convention specified below based on the model prediction service you are building for your team.

        a. Best Phase 2 Model - `mkdir marketing_pred && cd marketing_pred`
        b. Computer Vision Model - `mkdir cv_pred && cd cv_pred`
        c. Natural Language Model - `mkdir nlp_pred && cd nlp_pred`

    iv. In that directory, create a new file titled `README.md`. You will answer a number of questions from this and future weeks in this README file.

    v. Open the integrated terminal window and run: `pip install Flask==2.3.2`

    vi. Create a new file named `requirements.txt` in your project's top level directory (`capstone`)

    vii. In this new file, add the following line: `Flask==2.3.2`

    viii. Verify your project directory looks like this (substituting marketing_pred with the directory name you created above):

    ```
    capstone/
    ├── .venv/
    ├── .gitignore
    ├── requirements.txt
    ├── README.md
    ├── marketing_pred/
        └── README.md
    ```

    ix. Stage, commit, and push your changes to your team's directory

2. **[Ind]** Prepare a model for inference

    For each of three models (one per team member), serialize your model. Note: Each of you should name your model something unique.

    i. **[Ind]** Best Model from Phase 2. In the notebook you used to train your model import joblib and use the `dump` method to serialize your model.

    ii. [Ind] Computer Vision Model. Open the notebook titled, `cv.ipynb`. At the end of the notebook, import joblib and use the `dump` method to serialize your model.

    iii. [Ind] Time Series Model. Open the notebook titled, `nlp.ipynb`. At the end of the notebook, import joblib and use the `dump` method to serialize your model.

3. [Ind] Save the serialized model in your newly created directory.

4. [Ind] Stage, commit, and push your changes to your team's repository. Your newly created directory should now have two files (README.md and your model file.)

5. [Ind] Create a minimal Flask application.

    i. Create a new file in your new directory titled `app.py`.

    ii. Import Flask and define a route that listens for a `GET` request at the path `heartbeat` and responds with a success message as its response.

    iii. Run your Flask application to test its functionality. Add a screenshot of your test in the README.md file in your app's directory. (Hint: You can test a web server using curl in your terminal or shell.)

    iv. Once your app is successfully responding to HTTP requests at the heartbeat endpoint, stage, commit, and push your changes to your team's repository.

6. [Ind] Identify your model's input requirements. Review your team's (or the provided notebook if you are working on the computer vision or natural language processing models) notebook and identify the required inputs for your model to perform an inference. Describe those requirements in the README.md file in your app's directory. Be sure to include the following pieces of information:

    i. What method do you call for your model to perform a prediction?

    ii. What is the input type of that method?

    iii. What limitations exist for that input?

    iv. What does the input represent?

    v. Is the input pre-processed before it is passed as a paramter to the model's prediction method?

7. [Ind] With the answers identified above in mind, create a new endpoint in your Flask application. It must respond to HTTP requests at the path, `predict`. Use this new endpoint to expose your model's prediction method. A few things to keep in mind as you create this endpoint.

    i. You will need to dynamically access the input you identified in the previous list of questions. In Flask you do this by accessing the request object.

    ii. This may require you to adjust which HTTP method your `predict` endpoint responds to. You can learn more about the methods available at the Mozilla Developer Network (MDN) website. (Hint: the two methods relevant for this project are the GET and POST methods.) In the README.md file in your app's directory, write a short paragraph comparing and contrasting the GET and POST methods and their use cases. Include which one you are using for your app and why you chose it.

    iii. Since you deserialized your model previously using joblib, you will now need to import joblib in your `app.py` file and install it in your virtual environment. When you do, be sure to update your project's `requirements.txt` file as well.

    iv. Once you have created an endpoint that responds to the proper method at the correct path, you will need to consider how to load your model into memory when your application starts or when a prediction request is made. In your app's README.md file, include a short paragraph describing why you chose to load your model at prediction time or when your app starts. Compare and contrast the benefits of each approach. (Hint: Speed and memory are generally considered part of a tradeoff space when you make system design decisions like this.)

    v. Start your Flask application to test its functionality. Include a screenshot of your test in the README.md file in your app's directory. (Hint: curl can be configured with options to send files with a request or to use a specific request method)

    vi. Once your application is successfully responding to prediction requests at the `predict` endpoint, stage, commit, and push your changes to your team's repository.

8. [Ind] Now that you have finished creating a prediction application, navigate to your team's remote git repository on GitHub and open a pull request. When you do, follow the following guidelines.

    i. Request for one of your teammates to "review" your changes. Once they have checked your work, you can merge your changes into dev.

    ii. If you have properly completed all the previous steps for this week, after the first one of you merges your changes you will find your next merge requests are blocked due to a merge conflict. You will need to resolve the merge conflict present in your `requirements.txt` file before you can proceed to merge all your team's work. While they should be rare in this project, merge conflicts can occur and it is good practice for you to resolve them. Be sure when you do that all packages your project explicitly depends on (all those you "import" in your source files) are listed in your project's `requirements.txt` file.

9. by the end of this week, your team's repository should look like the following. If you have additional files, clean up your repository by removing them before starting on week 10.

```
capstone/
├── .venv/
├── .gitignore
├── requirements.txt
├── README.md
├── marketing_pred/
│   ├── README.md
│   ├── app.py
│   └── <your best Phase 2 model>
├── cv_pred/
│   ├── README.md
│   ├── app.py
│   └── <your serialized cv model>
└── nlp_pred/
    ├── README.md
    ├── app.py
    └── <your serialized nlp model>
```

Week 10 - Django Web App

Learning Objectives

LO1. Practice creating a simple web application

LO2. Add model inference functionality to a web application

LO3. Understand how to read package documentation

Tasks

Instructions:

1. **[Grp]** Make sure you have Django installed. To do this, you can run the following command in a shell prompt:
   `$ python -m django --version`

   (a) **[Grp]** If Django is installed, the version number will be displayed in the shell.

   (b) If Django is not installed, you will get an error saying "No module named django." If this is the case, install Django by following these instructions: link

2. **[Grp]** Read the Django documentation to get familiar with creating a Django project, the development server, and creating an app within your project (here).

   (a) Create a new folder to store your Django app.

   (b) `cd` into that folder from a shell prompt

   (c) Create a project named "lastname_capstone":
   `$ django-admin startproject lastname_capstone`

      i. This command should have created a folder named "lastname_capstone" in your current directory.

      ii. If this step did not work, you may be able to find troubleshooting steps here.

   (d) Verify that you have the following file structure:

   ```
   └── lastname_capstone/
       ├── manage.py
       └── lastname_capstone/
           ├── __init__.py
           ├── setings.py
           ├── urls.py
           ├── asgi.py
           └── wsgi.py
   ```

   (e) Verify that the Django webserver works.

      i. Make sure you are currently in the outer "lastname_capstone" directory. Type `$ ls` into the shell prompt, and you should see `manage.py` in the directory contents.

      ii. Run the following:
      `$ python manage.py runserver`

      iii. Visit the following page in your web browser: `http://127.0.0.1:8000/`. You should see "Congratulations!" on the page with a rocketship taking off.

3. **[Ind]** Read the following page to learn how to integrate a machine learning model into a Django web app: link.

   (a) If you are a team of two: 1 person will add the model developed in Phase 2 to the Django app, and 1 person will add an off-the-shelf computer vision model to the Django app.

   (b) If you are a team of three: 1 person will add the model developed in Phase 2 to the Django app, 1 person will add an off-the-shelf computer vision model to the Django app, and 1 person will add an off-the-shelf time series model to the Django app.

Week 11 - Containerization

Learning Objectives

LO1. Understand how, why, and when to containerize web applications

LO2. Understand how to monitor an application in production

Week 12 - Instrumentation & Metrics

Learning Objectives

LO1. Understand how, why, and when to instrument an application

Week 13 - LTAC

Learning Objectives

LO1. Understand the capabilities offered by LTAC

LO2. Deploy your web application using Helm and monitor using k8s

Week 14 - Web UI

Learning Objectives

LO1. Understand the fundamental components of web sites, HTML, CSS, & Javascript

LO2. Improve the look and feel of your web application

LO3. Present an analytic using web technologies

LO4. Practice adding functionality to deployed web applications

Week 15 - Web Analytics

Learning Objectives

LO1. Add a feature to a web application

LO2. Practice redeploying a containerized application

Week 16 - React.js

Learning Objectives

LO1. Improve the look and feel of your web application with reactive components

Phase 4

Final Presentation

Learning Objectives

LO1. Practice synthesizing your work into a coherent presentation

LO2. Practice presenting your work

LO3. Practice learning from other teams' presentations

Scenario

Introduction and Deliverables

Week	Task	Deliverable
17	Prepare presentation and review with AI Professional mentor	Presentation
18	Final presentation	Presentation

Table 4.1: Deliverables in Phase IV. Progress review weeks are highlighted.

Tasks

1. [Grp] Create a presentation summarizing your work during the capstone. Your presentation should cover the following:

   (a) Introduction
      i. What did you do during the capstone?
   (b) Motivation
      i. What makes the project or your analysis interesting?
   (c) EDA Outcomes
      i. What techniques did you use during EDA?
      ii. What visualizations did you produce during EDA?
      iii. How did EDA inform your understanding of the dataset?
      iv. What did you learn during the EDA phase?
      v. Show figures and outcomes from your Jupyter notebook.
   (d) Modeling Outcomes
      i. What models did you examine?
      ii. How did each of your models perform?
      iii. How did you measure how your models performed?
      iv. How did you select which model to use?
      v. What did you learn during the modeling phase?
   (e) Web Application Development
      i. What analytics did you develop?

        ii. How did you design your web application?

       iii. Demonstrate the functionality of your web application

          A. Demonstrate performing live inference with your web app

          B. Demonstrate the functionality of your database

          C. Demonstrate the analytics you developed

       iv. What did you learn during the web application development phase?

   (f) Reflection

        i. What did you learn during the capstone?

       ii. What skills were new to you? What skills were review?

2. `[Grp]` Schedule a rehearsal with your AI Professional mentor

3. `[Grp]` Give your presentation to an audience including AI2C leadership and CMU faculty