Capstone Project Guidelines

Machine Learning Engineering Career Track

Overview

As you think about your capstone project, keep in mind that it's better to pick a relatively straightforward, "boring" project that you can deliver well than to pick a very complex, shiny idea that'll give you trouble. Here are the things that will make your capstone project a success:

- Focus your project around a realistic client and problem.
- Use your mentor as a resource, a sounding board, and a filter.
- Use the course TAs and the community for feedback throughout the course.

Springboard

Introduction

How to Pick a Good Capstone Project

When you are working as a machine learning engineer in the industry, you have to deliver a "good enough" solution that's ready for production in a limited amount of time. Unlike a typical course project, you don't have the luxury of taking the time to find the optimal or best approach, and you have to ensure that your work is production-ready. It's very important to have a sense of the various tradeoffs between different approaches and pick one that's well-suited to the problem and resources you have.

How do you pick a capstone project that reflects this mindset? Here are some general guidelines:

- **Is this a real problem that someone would care about?**
    - Ideally, your project could be a real application that others can try out or something to add to your portfolio to highlight your skills.
- **Is there a real data set available?**
    - You want to work on a project that has real-world data, not an academic/teaching dataset.
- **Is the data set easy enough to acquire and clean?**
    - Pick a dataset that's relatively clean. As a rule of thumb, if you have to spend more than two weeks cleaning your data, you may want to select another dataset.

To paraphrase Einstein, *keep it as simple as possible, but no simpler.* :-) Your mentor will help you decide if your project idea meets these guidelines.

Some students choose to work with a project topic that leverages specific domain expertise they have from any prior work experience, while others want to showcase their skills in an industry or domain they would like to enter. In both cases, work closely with your mentor to choose projects that are the right balance of challenging and attainable given your current skill set.

We recommend reading through this entire document before starting on your capstone project.

Good luck!!!

Springboard

What Is A Good ML Project?

A project walkthrough is an important part of the interview. You need ONE beautiful project, that showed iteration, with trial and error.

1. Consider What Skills You Want To Demonstrate and Develop

In unit 2, you have learned about different roles in the MLE job family. Capstone is a great chance to show your strength. Shape your capstone in a way that helps you shine.

2. Pick An Area of Focus

Although all students need to go through the milestones, we Introduce 4 areas of focus: data, models, engineering, and product.

A. Data-Focused Engineer

In this area of focus you will spend 50-60% of your time working with data, 20-30% of your time working with models, and 20-30% on architecture and MLOps. You will demonstrate a compelling solution to the problem you have selected where data is the star of the show. If you want to be a Data-Focused Engineer, you may:

- Spend more time engineering your data pipeline: acquisition, cleaning, processing, etc.
- Build a fully automated pipeline for data ingestion
- Drive the increased success and accuracy of your model through better data processing. High data quality is the best way to increase the accuracy of most models. The alternative to this is to increase accuracy by focusing more modeling and tuning, to circumvent flaws in the data. If you would prefer to focus on this aspect of machine learning, one of the other areas of focus might be preferable.
- Build lots of tests to ensure data sanity. This might involve implementing anomaly detection feeding the data into the ML model.
- Use a real-time/streaming solution vs the tradition batch processing approach, and potentially build an online training approach as a result
- Develop and display your ability to collect and handle a very large amount of data

B. Modeling-Focused Engineer

In this area of focus you will spend 25% of your time working with data, 50% of your time working with models, and 25% on architecture and MLOps. You will demonstrate a compelling

solution to the problem you have selected where the model is is the star of the show. If you want to be a Modeling-Focused Engineer, you may:

- Spend a considerable amount of time on feature engineering.
- Try many different models and carefully review and compare results to balance accuracy, generalization, overfitting, success metrics, and more.
- Perform hyperparameter tuning for every model.
- Go further than using a simple “out of the box” solution. You may achieve that by blending multiple models together, stacking, or any technique that will use a more elaborate approach. That might include testing a new deep neural network (DNN) structure for the problem at hand.
- Use some of the latest research and tools, and implement or adapt them to your problem.

C. Architecture-Focused Engineer

In this area of focus you will spend 30-40% of your time working with data, 20-30% of your time working with models, and 40% on architecture and MLOps. You will demonstrate a compelling solution to the problem you have selected where the engineering and software architecture is the star of the show. If you want to be a Architecture-Focused Engineer:

- Your primary goal will be a well-engineered solution from the data perspective, from model training to deployment and especially the post-deployment phase. This may require monitoring, retraining, and iterating on your solution based on performance.
- You will likely not use an out of the box cloud solution, but instead focus on deploying your own from scratch, writing your own API, building a Docker container, and deploying it on Kubernetes or another cloud solution.
- You will have a detailed workflow diagram for each logical area - data, modeling, and in production - as well as operational diagrams.
- You might use Airflow or other orchestration solution to trigger retraining or redeployment of your model.
- You will integrate many open source platforms/solutions together to create an enterprise-like-solution.
- Your deliverable will be operating live, and you will have developer-oriented documentation available for others to use or integrate your work.
- You will know exactly how long it takes to train your model, and how many inferences/s you are able to achieve, and demonstrate you can scale.

D. Product-driven Engineer

In this area of focus you will spend 30% of your time working with data, 20% of your time working with models, and 50% on architecture and packaging your model as a product. You will

demonstrate a compelling solution to the problem you have selected where the product or app is the star of the show. The utility of your model is the major focus of your project, and the engineering work might be written up in a white paper on how you used your data and tools. If you want to be a ML-Driven Product Engineer, you'll:

- Show the clear utility of your solution to end-users
- Demonstrate how the product works to anybody
- Have a website that shows off the product, why you built it, how you built it, and which problem it solves
- Still spend a generous amount of time on data, modeling, and deployment of your project.
- Spend extra time in testing the product, even potentially building a feedback loop into your product in order to improve the original trained model
- Show both the synthetic and real life accuracy of your model
- Recognize what kind of biases your system might have, and how to address it

You do not have to make a decision now and can change as you know more about what datasets and models you would want to work with. This focus will help you to manage your time and direct you to allocate energy wisely throughout the course.

3. Is Your Project Too Simple Or Complex?

This goal of this course is to give you the skills to design and create an ML/DL application, and the skills to deploy that application to production using the latest engineering tools and techniques.

ML/DL contains a smorgasbord of techniques. There is no way to apply everything ML technique to the problem you have chosen, nor should you. The more complex the techniques you use, the more attractive your project will be to employers. However, it's important to balance complexity with a practical approach that produces a real, scalable application. Some questions to ask yourself:

- What technique best applies to this problem?
- How easily can this technique be deployed to production?
- Are the results of this technique "good enough" to meet the business requirements of the solution?

In the real world, it **can sometimes be better** to use a simple logistic regression approach that's accurate enough, highly performant, and easy to deploy as a production application, than a intricate deep learning approach that's intensive to deploy or maintain. You'll have to make similar decisions and tradeoffs about your capstone project. When you're interviewing for your MLE role, you'll be expected to justify these decisions and tradeoffs.

Springboard

Work with your mentor to determine a problem and approach that meets the requirements of this course and makes you both feel confident that you'll be able to do well.

Project Milestones

Here is a list of milestones that you will need to submit to finish your capstone project. They are interwoven with the technical units. Understand that the work you do at each step is not necessarily over once you have completed it. This is an open-ended project and depending on your idea and it's execution, some of these steps may take longer and you may need to revisit them. This is your awesome idea that you are creating, so the circumstances surrounding it's development might necessitate constant tweaks to some of the steps throughout the project. These steps are useful to help you make a great project. In later units, you will be advised how to allocate your time given different focus areas.

Step no.	**What you will do**	**Unit(s) no. & name of deliverable**	**Estimated time**
1	Start Planning for Your Capstone	Unit 2	0.5 -1 Hours
2	Collect Your Data	Unit 3	2-4 Hours
3	Benchmark Your Model	Unit 8	2-4 Hours
4	Project Proposal	Unit 8	2-4 Hours
5	Data Wrangling & Exploration	Units 12	15-20 Hours
6	Try Various Models	Unit 14	3-4 Hours
7	Build Your Machine Learning (or Deep Learning) Prototype	Unit 20	25-30 Hours
8	Scale Your Prototype with Large-Scale Data	Unit 20	10-15 Hours

9	Study Optional units	Unit 30~33	20-25 Hours
10	Pick your deployment method	Unit 21	0.5-1 Hours
11	Design Your Deployment Solution Architecture	Unit 23	2-4 Hours
12	Run Your Code End-to-End with Logging and Testing	Unit 24	5-10 Hours
13	Deployment Implementation	Unit 24	12-15 Hours
14	Share your project with the world	Unit 27	2-4 Hours
15	Final Submission	Unit 34	

Phase 1: Build a Working Prototype

Step 1. Start Planning for Your Capstone

For each of your project idea:

- Include a short blurb for each of your ideas.
    - The blurb should, at a high level, describe the problem and the data you'll be using to solve it. At this point, there's no need to talk about specific methods and techniques.
- Post your idea (title and blurb) on the community and solicit feedback from both mentors and other students. Pick one idea to work on based on the feedback you get. Discuss the idea with your mentor to ensure they're on board.

Please note: The goal of a project is NOT to do something novel — it's to demonstrate your competence as a machine learning engineer. It's perfectly acceptable to work on a dataset that's been worked on before and even answer a question that's been answered before, as long as the work is your own.

Examples For Inspiration

This is one of the project ideas by one of our alumni, Siri Surab.

Here's a few more ideas to spark inspiration for your capstone project. Many of these ideas come from natural language processing or computer vision, since they're two of the hottest fields in AI. Your project doesn't need to be in one of these specializations.

- **Inventory tracking and compliance using object recognition:** A company wants to automatically track inventory in its warehouses using a camera with an object recognition algorithm. A similar model could be used in a home application, such as a smart fridge which recognizes what's placed in it.
- **Language translation:** Also called machine translation, this technique uses AI to translate one human language to another, in text or speech. You can work between the two formats like speech-to-text transcription or text-to-speech generation.
- **QA systems and chatbots:** Increasingly, companies are using automated chatbots to address their customer service workloads. These bots produce human-like responses to questions and are getting better every day.
- **Text summarization:** Imagine an application that can digest the daily news and produce a coherent summary tailored to a user. You can apply summarization to different domains, such as an application that can automatically produce a personalized summary for a student who's trying to research a large amount of material.
- **Fraud/spam detection:** Detect "bad" transactions or items in a dataset. This could take the form of detecting fraudulent credit card transactions, fake news on social media, spam in email, doctored images or video, or abusive behavior on Twitter. Depending on the problem, you can use a variety of techniques, ranging from "traditional" machine learning to the latest in deep learning.

Former Capstone Project

This is a project by one of our alumni, Siri Surab. She used the Quora Duplicate Questions dataset from Kaggle, and applied NLP techniques from both "old-school" Machine Learning as well as Deep Learning to identify duplicate questions on Quora.

- Ideation

- Blog Post
- GitHub repository

We don't expect you to understand all of these techniques at the beginning of the course, but we've presented this here as an example of what your final project will look like. You'll go much deeper into this specific project in a later unit on NLP.

Plan For Your Project Based On Your Focus Area

In total, you would spend ~120 hours on your project, including studying for some advanced topics if you choose to. In the last resource, you just learned about four areas of focus. Be aware that each of these focuses would influence how you spend time on each of the milestone steps of your capstone project.

1.) Think about which area that you want to focus on based on your strengths and what you want to showcase for your future employers.
2.) Here is a potential plan for you. It contains a breakdown of suggested time allocation for each main part based on your focus, and some areas where you need to budget extra time. This is only a reference, please discuss concrete details with your mentors. Don't feel that you need to commit to this plan. Your plan may likely change as you proceed through the course.

	Data Collection & Processing	**Models Prototype & Scaling**	**Deployment/ Engineering Architecture**
	Step 1~5	Step 6~9	Step 10~12
Data -focused	50%~60% You may spend extra time on data collection, clean and processing. And if you need to process real-time data, you will need to build ETL infrastructure.	20~30%	20~30%
Model -focused	25%	50% You will try different models and make ensemble models to enhance performance. If you are developing DL models, please budget	25% If you are short on time, feel free to leverage existing deployment methods like Algorithmia.

		extra time.	
Architecture -focused	30~40% You can shine by building automation in the data pipeline to support analysis at scale.	20~30%	40%% You can build automation to train/test data or design your own engineering architecture. We suggest this focus for students with a strong SWE engineer background.
Product -focused	30%	20%	50% You might spend extra time on developing the app UI/UX and streaming data.

1. Write a description of your three capstone project ideas.
Your ideas can be broad and high-level. The descriptions should address the problem and identify the data you'll use to solve it. You do not need to talk about specific methods and techniques.

- Write at least 3 to 4 sentences explaining your idea and identifying the data you'd use to solve it.

2. Submit a Google Doc link to the submission box.

- Remember to enable sharing permissions to "comment."
- Please do not submit .pdfs, .ppts, or markdowns.

3. Review your ideas and your tentative decision for your focus area with your mentor during your next call.

4. Post your idea (including a title and description) to your student community to receive peer feedback.

Step 2. Collect Your Data

The first step in your capstone project is to collect data. In some cases, it can be as simple as downloading a dataset in a zip file or a tarball. Or, it can require extracting data using a publicly available API or scraping a website. We urge you to work closely with your mentor to ensure that

the data collection process is not too challenging. Also, **if your data collection requires you to write code, START EARLY. Refer to 3.5.1 What Data To Collect for detailed tips.**

At the end of this step, you'll submit a link to your GitHub repository that contains the following:

1. Code for how you collected the data, if applicable
2. The actual dataset: If your dataset is small enough to fit in a CSV, then include it in the repository. If it's a big dataset or has a lot of binary files (graphics, audio), consider using the Git Large File Storage extension.

Step 3. Benchmark Your Model (Optional)

Summary

Time Estimate: 2 - 4 Hours

This is an optional step for students who may not have a straight forward ability to plug their dataset into an existing service. The only way you can figure that out is by trying. You can use the results of running your data through an AI service as a benchmark, or reference point, to see how well other models you may experiment with are performing. In this unit, you learned about many different AI services like AWS and Azure. If one of these tools will work well with your dataset, go ahead and plug it in. Get the initial results and discuss them with your mentor as to whether these results provide a good baseline to work against when developing your own model.

While you are to get very quick results with minimum effort, be aware of the danger of using this black-box solution without knowing how it truly works under the hood. Because you don't know what is really happening to your data, you can't totally understand your results as an engineer. Take a look both at where such a service shines or fails on some of the data input you feed it. Consider how you might address such issues in your own capstone project.

Please note, you may have to do additional wrangling in order for the services to process the data properly.

Step 4. Project Proposal

Once you've decided on your final capstone project idea, we'd like you to write a proposal. A project proposal is a short (1-2 page) document that answers the following questions:

1. What is the problem you want to solve? Why is it an interesting problem?
2. What data are you going to use to solve this problem? How will you acquire this data?

3. In brief, outline your approach to solving this problem. You might not know everything in advance, and this approach may change later. This might include information like:
    a. Is this a supervised or unsupervised problem?
    b. If supervised, is it a classification or regression problem?
    c. What are you trying to predict?
    d. What will you use as predictors?
    e. Will you try a more "traditional" machine learning approach, a deep learning approach, or both?
4. What will be your final deliverable? Will it be an application deployed as a web service with an API or a more robust web/mobile app.
5. What computational resources would you need at a minimum to do this project? *You may not have a very clear sense now, but work with your mentor to come to an estimate. In real industry applications, you'll often be called upon to provide resource estimates at the beginning of a project.*
    a. Processing power (CPU)
    b. Memory
    c. Specialized hardware such as GPUs

The proposal will be part of a GitHub repository for your project. All code and further documentation you write will be added to this repository.

Step 5. Clean and Wrangle Your Data

In the course, you'll apply some of the data wrangling techniques you have learned to your capstone dataset. As you're working in your Jupyter notebook, document the steps you undertook to clean your dataset.

Consider the following as you take notes:

- What kind of cleaning steps did you perform?
- How did you deal with missing values, if any?
- Were there outliers, and how did you decide to handle them?
- If your dataset is too large to work with, does it make sense to build your prototype on a smaller subset of the data?

After you have obtained and wrangled your dataset, you will perform preliminary exploration. This exploratory data analysis (EDA) uses a combination of inferential statistics and data visualization to find interesting trends and significant features. For example:

- Are there variables that are particularly significant in terms of explaining the answer to your project question?
- Are there strong correlations between pairs of independent variables or between an independent and a dependent variable?

At the end of this step, you'll submit a link to your Jupyter notebook in your GitHub repository that shows how you cleaned, wrangled, and (optionally) explored the data.

Step 6. Experiment With Various Algorithms

The purpose of this step is for you to rigorously test how to build the best model for analyzing the patterns found in your dataset. Perform some of the following activities:

- Build an automated process to test many modeling techniques and ML algorithms with your data to see which one yields the best results
- Define the performance metric(s) best applied to your problem (accuracy, F1, RSME, LOC, etc.)
- Test various loss functions across models to see which one yields the best result
- Perform tuning of one or more model, across one or multiple hyperparameters
- Build a robust cross-validation process for your problem
- Ensemble multiple models together, and demonstrate the superior results
- Analyze the prediction results to confirm how some of your models ended up properly generalizing or overfitting the data
- Present your best model(s)

Step 7. Build Your Machine Learning (or Deep Learning) Prototype

The goal of this step is to find a machine learning or deep learning approach that works for your problem, and show that it's a viable one. Since the application has not been deployed to production yet, we'll call it a *prototype*.

You'll build your prototype in a Jupyter notebook. Depending on your problem, you'll be using a more "traditional" machine learning (ML) technique or a deep learning (DL) one. Your goal is to come up with a working implementation in a Jupyter notebook. This prototype could work on a subset of the data, but demonstrates that your approach to solving the problem is a viable one based on the following criteria:

- The data has been reasonably split into training, validation, and test sets
- You have used the correct metric(s) to evaluate the performance of your algorithm

- The performance of your algorithm is “good enough” as determined by your mentor

At this point, you’ll submit a link to your Jupyter notebook with the ML/DL algorithm coded and your results well-documented in a way that your mentor (or a potential employer) can easily follow.

Step 8. Scale Your Prototype with Large-Scale Data

In this step, your goal is to ensure that your ML/DL approach, which has proved to be viable, can work with large volumes of data. You can work with your mentor to determine what that means for your problem.

Using scikit-learn, SparkML, Keras, TensorFlow, PyTorch or another technology you have learned, implement your prototype at scale.

In case your earlier prototype was working with a subset, ensure that this scaled-up prototype can handle your complete dataset.

Think about what your capstone problem would look like in the real world.

- How much data would you need to handle?
- Can you scale your prototype to handle that volume of data using the approach and tools you have selected?

In a Jupyter notebook, implement the scaled version of your prototype and document what trade-offs and implementation decisions you have to make to scale your algorithm. Submit the GitHub link to this notebook.

Step 9. Study Advanced Units

We have three optional units in this course that will provide you with in-depth knowledge in Deep Learning, Natural Language Processing, or Computer Vision (this is actually two units, one conceptual and one hands-on). These units are here to help you succeed in your capstone and your career, if you wish to specialize in one of these topics.

- If you pick Deep Learning/NLP/Computer Vision, you will need to do additional study. We have created units on each of these topics and will share some advanced, state-of-art knowledge.
- Refer to FAQ if you are undecided about pursuing one of the topics.
- You will still need knowledge of the basic algorithms taught throughout the course

1. If you are pursuing one of these specializations, please pick one of the four optional units (unit 30-33) to study.
2. After you finish studying these units, write a few paragraphs on how you have leveraged the more advanced technique (if it applies to them) in a notebook, and compare such solutions with previous results with your mentor.
3. Then please mark this project step as complete, after you finish studying the optional units.

Phase 2: Deploy Your Prototype to Production

Congratulations on creating your machine learning solution prototype! As a machine learning engineer, your work does not end here. It's your job to also deploy this solution to production.

In this section, we've outlined a few steps you're likely to follow as you deploy your application, based on input from MLEs who work in the industry.

Please note: You may not always follow these steps in the order stated, and you're welcome to work with your mentor to figure out the best deployment strategy for your project.

Step 10. Pick Your Deployment Methods

In this step, you will pick how you would like to deploy your project. The right deployment method will vary depending on the scope of your project.

- There are three majors ways to deploy your projects: as an API, through a library on an existing framework, or through a shipping product (which can be either a web site end users can access for example, or a mobile app)
- Refer to some of the project examples
- Be aware of your time management
- Please refer to Unit 23 teaches basic tools and unit 24 have more advanced options
- Please submit a short list of your next steps regarding deployment and engineering. Please discuss with your mentor to decide deployment methods.

Step 11. Design Your Deployment Solution Architecture

The first step in deploying a prototype is to determine what the deployment would look like in production. What are the various pieces of the deployment, and how do they fit in? Here are some specific questions to think about:

- What are the major components of your system? What are the inputs and outputs?
- Where and how will the data be stored?
- How will data get from one component of the system to another?
- What is the lifecycle of your ML/DL model?
    - How frequently do you need to retrain your model? Is it at fixed intervals when you collect a certain amount of new data or when some other conditions are met?
    - What kind of data do you need for retraining? How will you store and manage it?
    - How do you know if the retrained model is good enough to deploy?
    - How will the retrained model be deployed?
    - How will the retrained model be stored as an artifact?
- How will the system be monitored? How will you debug it if there are problems?
- How will your system respond to unexpected errors or outages?
- What are the specific tools/technologies you'll use to build this system?
- What is the estimated implementation cost in terms of resources, time, and money as applicable?

Submit a link to a document (Google Docs or PDF) containing the following:

1. A sketch or diagram of what your deployment architecture would look like. You're welcome to draw it by hand and take a picture to upload to your document.
2. A 1-2 page write-up explaining your design decisions, covering why you chose to design your system in a certain way and why you have chosen to use certain technologies. Also, include an estimate of the cost of the system (e.g. computing resources, time, money).
3. A pre-deployment checklist. Please refer to 20.4.1 (link) to the list of questions that you should think through.

Your mentor will review your document and discuss it with you in your next call. Based on the feedback from your mentor, you may need to rework and resubmit this architecture design.

Step 12. Run Your Code End-to-End with Logging and Testing

So far, your code has been running in a Jupyter notebook. However, that's not the way production code works. The very first step in deploying your prototype to production is to ensure that your code runs independently from the command line and is well-tested. So, now is the time for you to apply all of the best practices that you've learned about software engineering!

1. Create a new repository for your production code by cloning the repository where your prototype was created. We will call this new repository the *production repo*.

2. Refactor the code in the production repo. This might include the following steps as applicable:
    a. Get your code out of your Jupyter notebooks into Python files. You'll edit the Python files in a code editor or an IDE such as PyCharm, Eclipse or Atom. Pick your favorite editor or IDE here, no restrictions! Revisit the pre-work to refresh your memory on setting up IDEs.
    b. Create Python modules, classes, and functions, as appropriate, so your code is well-organized
    c. Remove unnecessary print and debugging statements, along with interactive features such as visualizations
    d. Add useful logging statements at critical points of the code.
3. Add unit tests to test your various functions and methods. Aim for at least 60% of code coverage in your tests. Think about how to cover edge cases or erroneous inputs.
4. Ensure that your code runs end-to-end from the command line with appropriate command-line options
5. Create a README at the top level of your repository documenting exactly how to run your code.
6. Submit a link to your production repo containing the refactored and tested code.

Step 13. Deploy Your Application to Production

Now that you've refactored your code and designed an architecture for your system, it's time to actually implement the architecture and deploy it to production.

The exact steps you follow here depends on you and your mentor, and your specific project. However, in general, there are a few common steps that MLEs might follow:

1. **Step one**: Implement your data pipeline. This is where all of the engineering skills you've learned will come in handy.
    a. Implement the right systems for storing your data, loading it for training your model, and making predictions with your model.
    b. Make sure you log necessary and sufficient information for monitoring and debugging your system.
    c. Test each component and subsystem as much as possible as you build it out. It's much easier to find bugs in smaller systems than larger ones
2. **Step two**: Design an API using a tool such as Swagger. If you choose to use a managed ML service (e.g. Domino, AWS SageMaker, Google Cloud ML), they might provide out-of-the-box API calls. That's completely fine!
    a. As you design your API, try to make sure your API follows good design principles.
    b. Use a tool such as Flask to create a simple user interface for your API

3. **Step three**: Package your application as a Docker container. This makes it easy for you or anyone else to "spin up" your application without worrying about installing the right tools or libraries. Some of the platforms you'll use might provide out-of-the-box support for containers; you're welcome to use that.
4. **Step four**: Test your API using tools such as Postman or Swagger. Make sure you test both read and write functions, and various corner cases.
5. **Step five**: Document your API. Using a tool such as Sphinx lets you automatically create beautiful API documentation from annotations in your code. Add your documentation to your GitHub repository.
6. **Step six**: Add instructions at the very beginning of the README for your GitHub repository on how to access and use your application, and submit a link to your GitHub repository

Step 14. Share Your Project With The World

The final deliverables for your project, such as your code and application (please see the project evaluation section below), are part of your portfolio, which means that you'll be sharing them with potential employers. We require students to share their project on Piazza to receive feedback from peers and on GitHub to highlight their skills to potential employers.

When you and your mentor agree on a stopping point for the project, you should have the following deliverables ready *before asking your student advisor* to start the completion process.

Mandatory deliverables

- **Code** for your project, well-documented on GitHub. We've provided some guidelines for what a good GitHub should look like below.
- **An application** that is deployed on a cloud platform. This application must be accessible using an API or a web service. Your GitHub README should contain instructions on how to access and use the application.

Optional deliverables

- **A frontend web interface** for your application. This is optional but can make the visualization of your work more compelling. However, if you don't have any web development skills, we recommend you focus on your core MLE skills and not spend time on this task.
- **A blog post** summarizing your project. We love distributing our students work. It helps students get exposure to potential employers and sets you up as a voice in the community. If you write a blog post about your project, our content team will help you polish it up and share it on a variety of platforms.

  Blog posts are a great way to generate awareness of your work and your brand as a

machine learning engineer. Here are Springboard's guidelines for blog posts. If you'd like your blog post to be considered for the Springboard blog, please write a draft according to the guidelines and share it with your student advisor. You're also welcome to post on your own blog, Medium, LinkedIn, or other platforms. Please share the link of your post with your student advisor.

Springboard

Guidelines for Strong Portfolio

Your portfolio consists of all of your projects, including the code and documentation, usually in your GitHub account. Typically, a hiring manager who looks at your portfolio wants to see evidence of both technical and communication skills. It is your responsibility to ensure that your portfolio is clear and easy to navigate.

Tips to Help Your Portfolio Stand Out

1. Every project should ideally be in a separate repository that is clearly named.
2. For each project:
    a. Have a README page:
        i. The README should provide an executive summary of the project (i.e. summarizes the problem, approach, and results), along with instructions, if needed, on how to run and access the application.
        ii. The README should also include a list of the important files that the reader should view. The files should be clearly named and organized.
    b. Clean up your code and document:
        i. Your approach and methodology are clear to any technical reader. You do not need to document every line of your code, but have comments or text explaining important decision points and why you chose them.
    c. Include any other documents that you have created (e.g. a report or slide deck in the same repository as the code).
        i. Make sure the README points them out to the reader.
3. Ensure that your portfolio is not cluttered with “junk” (i.e. repositories or folders that are incomplete, irrelevant, or undocumented).

Overall, try to put yourself in the shoes of an experienced machine learning engineer or hiring manager who has a limited amount of time (5-7 minutes) to look at your portfolio. How can you ensure that you make it easy for them to get a good idea of your skills and abilities? Your course TA, mentor, and the community should also be able to provide good feedback on your portfolio, so please use them as resources.

Project Evaluation

For Springboard to consider your workshop complete and issue a certificate of completion, your mentor needs to approve your final project submissions based on the rubric listed below. If the project is not approved, please discuss the feedback from your mentor and resubmit in case improvements are necessary. Your student advisor will not be able to process your workshop completion until your project is approved by your mentor!

Capstone Project Rubric

We use the following rubric for evaluating final capstone projects. Please take a good look at it and make sure you discuss with your mentor to agree on success criteria.

View the Capstone Project Rubric here.

Your capstone project is evaluated based on two criteria: completion and process/understanding. Within each criterion, specific benchmarks and expectations are listed to help ensure the overall quality and mentor approval of your capstone project.

Because your mentor approves each step and later the entirety of your capstone, it's vital that you work with your mentor throughout the course to determine and agree what the bar is for each criterion and to incorporate timely feedback during the intermediate stages.

Good Luck with your Capstone Project!