Welcome

Machine learning models are great, but unless you know how to put them into production, it's hard to get them to create the maximum amount of possible value.

Some of challenges of taking a ML model to production:

- **Concept Drift or Data drift:** sometimes the data you receive for prediction is different (drifting) from the data you trained the model on.
- **It takes a lot more than ML code:** ML code only covers the $x \rightarrow y$ mapping. But, the code for a ML production system is much larger.

- Usually the ML code is only about 5-10% of an ML production system code. So, after developing the ML model, there's still a lot of work to go from the initial PoC (proof of concept) to a production system. It's referred to as **PoC to Production Gap**.
- **So, what's all the other stuff?** Based on the Sculley et. al.'s paper, it's the following:

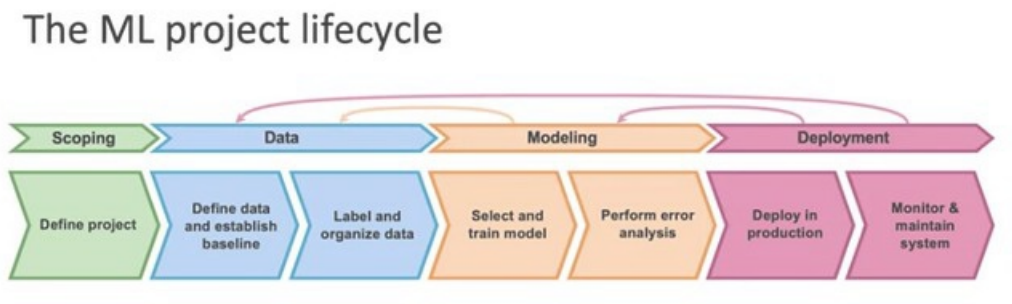
[D. Sculley et. al. NIPS 2015: Hidden Technical Debt in Machine Learning Systems]

Steps of an ML Project

When building a ML project, thinking through a ML project lifecycle is an effective way to plan out all the steps that need to work on.

These are the major steps of a ML project:

1. Scoping
2. Data
3. Modeling
4. Deployment



Case Study: Speech Recognition

What's needed to build a valuable production deployment speech recognition system?

1. **Scoping**
    - Decide to work on speech recognition for voice search
    - Decide on the key metrics:
        - Accuracy, latency, throughput
        - Note that metrics are very problem-dependent.
    - Estimate resources and timeline
2. **Data**
    - Is the data labeled consistently?
        - Example: you might have a simple audio transcribed in this three ways:
            - "Um, today's weather"
            - "Um… today's weather"
            - "Today's weather"
        - Essentially, one can pick any of the above way of transcribing. The problem is that sometimes you have 1/3 of your data labeled the first way, 1/3 the second way, and 1/3 the third way.
        - This makes your data inconsistent and confusing for the learning algorithm. How is the learning algorithm supposed to guess which one of these conventions specific transcription has happened to be used for an audio clip.
        - In such cases, one possible solution is to standardize on one way of labeling.

- How much silence before/after each clip?
- How to perform volume normalization?
- And other possible “data definition” cases.
- In production systems, the data definition is usually is not fixed.

3. **Modeling**
   - Three main contributors of an ML model:
     - Code (algorithm/model)
     - Hyperparameters
     - Data
   - In ***research/academia***, usually they keep the data fixed and vary model algorithm or hyperparameters in order to get good performance.
   - In contrast, in ***product teams*** where your main goal is to just build and deploy a working valuable ML system, it’d be more valuable to hold the “code” fixed and instead focus on optimizing the data and maybe the hyperparameters.
   - **ML system = Code + Data**
   - **Error analysis** can tell you where your model still falls short.
   - You can use error analysis to tell you how to *systematically* improve your data (and maybe improve your code too).
   - Error analysis can help you ***be more targeted in exactly what data to collect***. You don’t want to always collect more data (it can be expensive too).
4. **Deployment**
   - This is how you might deploy a speech recognition system:
     - You have mobile phone (**edge device**), with software running *locally* on your phone.
     - The software taps into the microphone to record what someone’s saying.
     - You’d use a **VAD (Voice Activity Detection)** module to select out just the audio of someone speaking.
     - Then, you send that audio clip to a “**prediction server**” (which is usually in the cloud).
     - The prediction server return both the transcript + search results.
     - Then, the returns will show on the frontend on your phone.