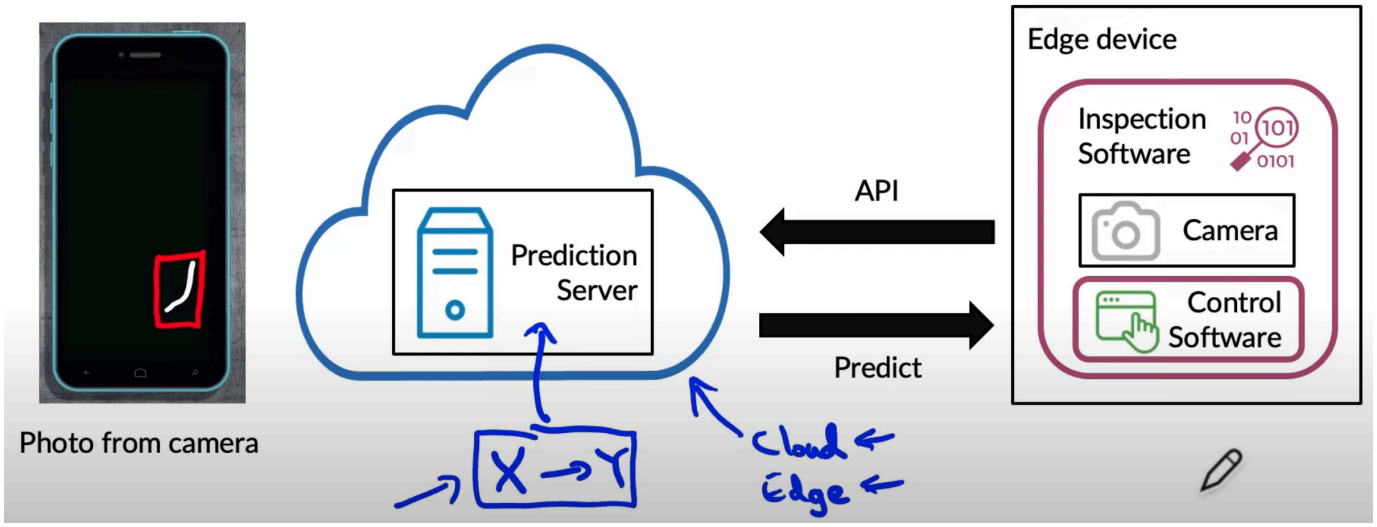
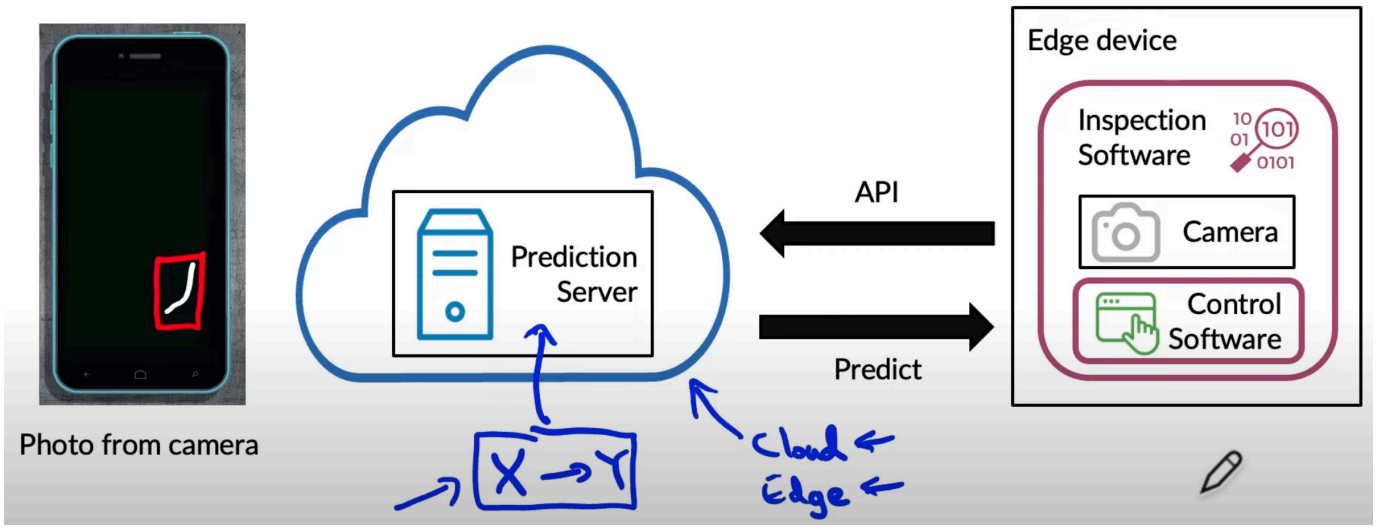
Week 1: Overview of the ML Lifecycle and Deployment

1. Overview

- **Deployment Example** (Visual Inspection of Phones)





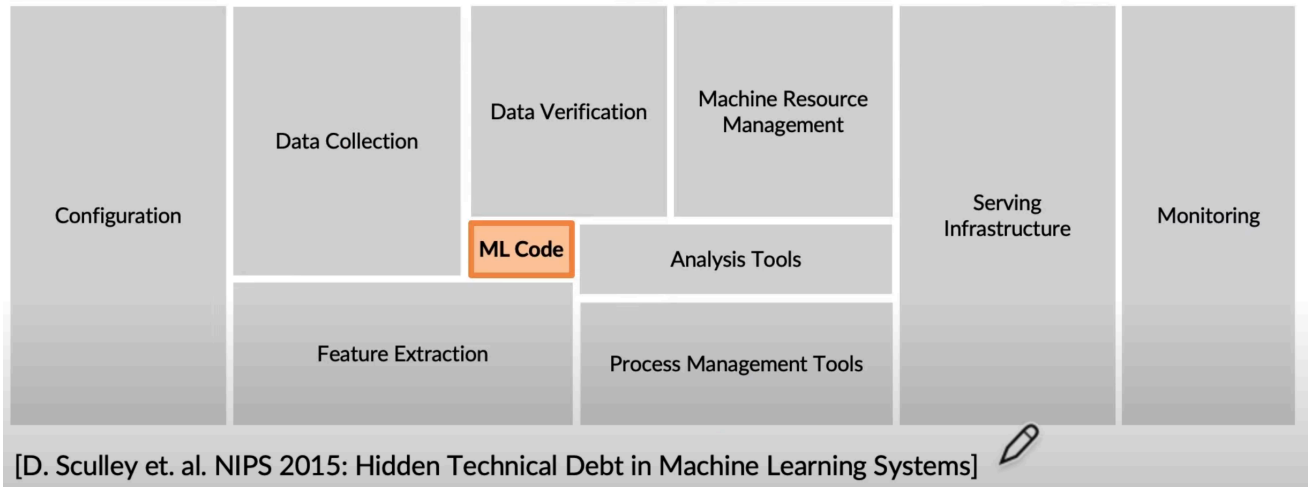
<An ***edge device*** is configured via local access and also has a port to connect it to the internet and cloud. >

--> after successfully training a machine learning model on Jupyter Notebook, it's likely that the actual dataset changes because of the environment conditions (lighting...) --> a machine learning engineer needs to adjust the data distribution

- **ML in Production**

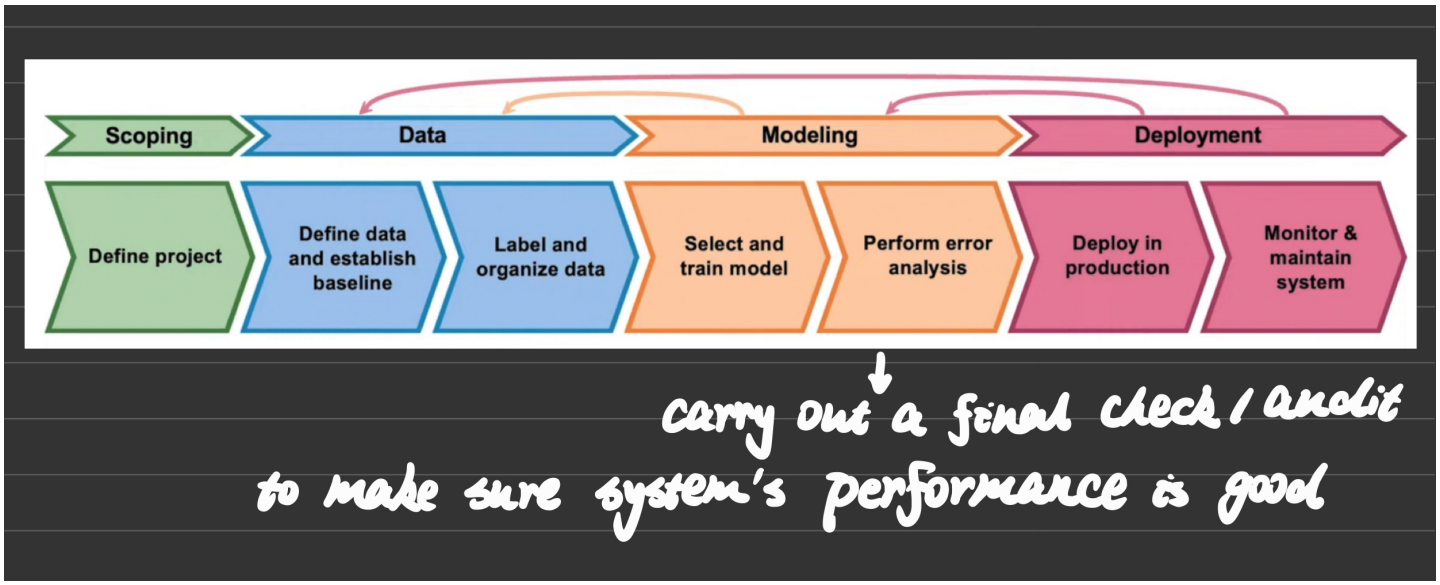
  --> in the production process, only 5-10% of code is for ML model

  - The requirements surrounding ML infrastructure



Key point: to systematically plan out the life cycle of a machine learning project

2. Steps of an ML Project



- **Case Study: Speech Recognition**
  - Step 1: Scoping
    - Decide to work on speech recognition for voice search
    - Decide on key **metrics**: accuracy, latent, throughput (how many queries per second could we handle)
    - Estimate resources and timeline needed
  - Step 2: Data
    - Define data: **[Make sure to have high quality data]**
      - Is the data labeled consistently?

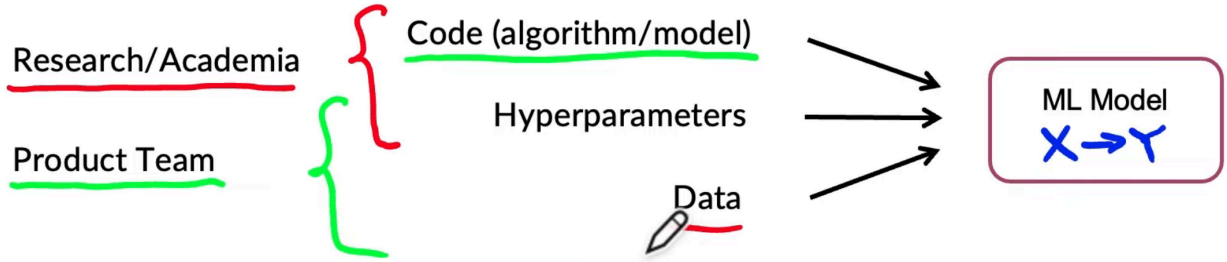
"Um, today's weather"
"Um... today's weather"
"Today's weather"

--> choose the first and second translation (the third translation is inconsistent and may be confusing for learning algorithms)

--> **Set up a standard at the beginning**

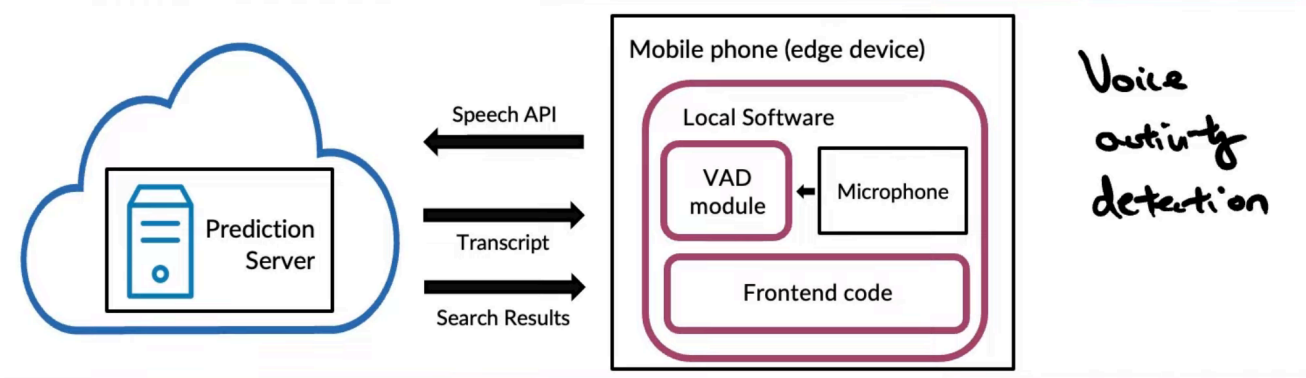
- How much silence before/after each clip?
- How to perform volume normalization?

- Step 3: Modeling
  - Three key inputs:



--> In production, tend to fix the code and change data and hyperparameters instead (maybe download code from GitHub and focus on optimizing data --> efficient and cheaper to get higher accuracy) --> **Data-centric Approach**

- Step 4: Deployment

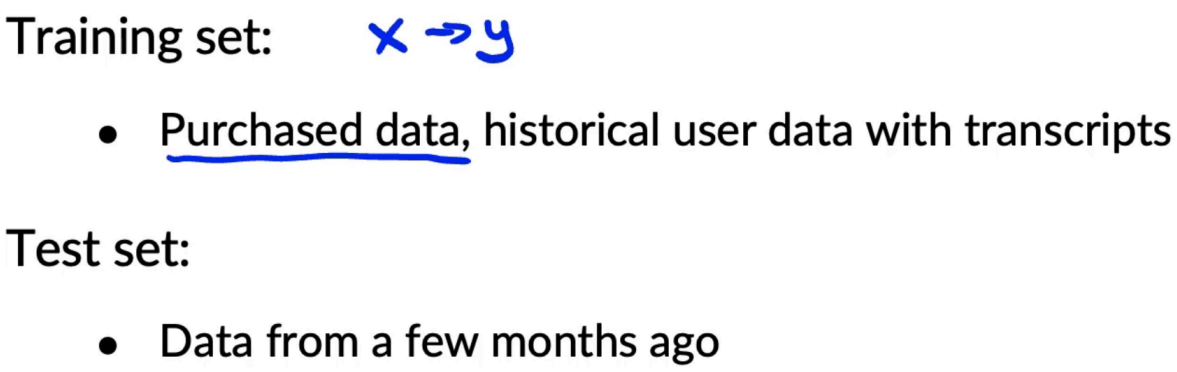


  - Key challenge: **Concept/Data Drift** when data distribution changes

- Other Import Discipline:
  - MLOps (Machine Learning Operations): software tools and principles to support progress through the ML project lifecycle

3. Key Challenges

- **1. Concept/Data Drift [Handled by monitoring and maintaining the system]**



- Concept drift: the mapping ($x \rightarrow y$) changes

  Data drift: the input data distribution of x changes

- Data may have changed because of language, time...
  - Gradual change along time
  - Sudden change (due to big change like COVID-19) --> collet new data and retrain systems to adapt to new data distribution

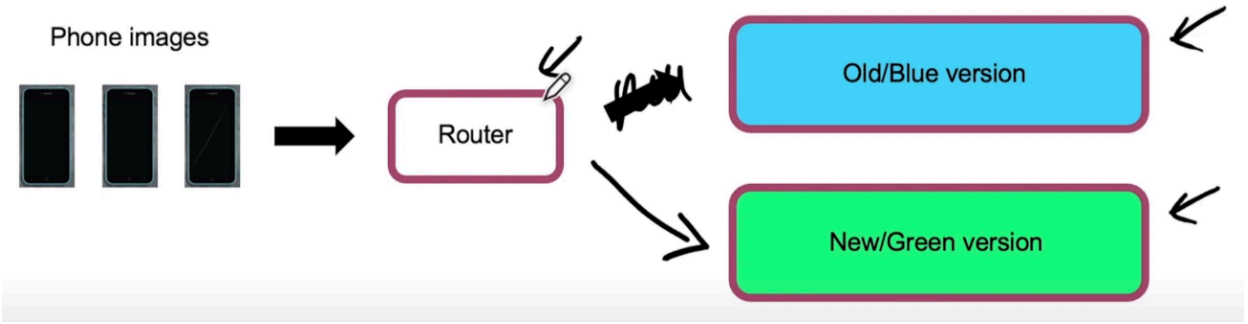
- **2. Software Engineering Issues [Handled by deploying in production]**
  - Checklist of Questions --> make the appropriate software engineering choices when implementing prediction services
    - **Realtime or Batch** predictions
    - Predictions run on **Cloud** (for large computing resources) **vs. Edge/Browser** (mobile and more controlled...)
    - **Compute resources(CPU/GPU/memory)** --> choose the right software based on resources provided
    - **Latency, throughput(QPS)**
    - **Logging** --> may be useful to log as much of the data as possible for analysis and review + provide more data for retraining the learning algorithm
    - **Security and Privacy**

4. Deployment Patterns

- **Common Deployment Cases**
  - New product/capability: start with a small amount of traffic and then gradually increase
    - **[Canary Deployment]**: the practice of making staged releases --> to spot problems early on
      - Roll out a software update to a small part of the users initially, so they may test it and provide feedback.
      - Monitor system and ramp up traffic gradually
  - Automate/Assist with manual task: with previous knowledge and experience
    - **[Shadow Mode] is usually used**: ML shadows the human and ***run in parallel*** + ML system's output ***not used for any decisions*** during this phase

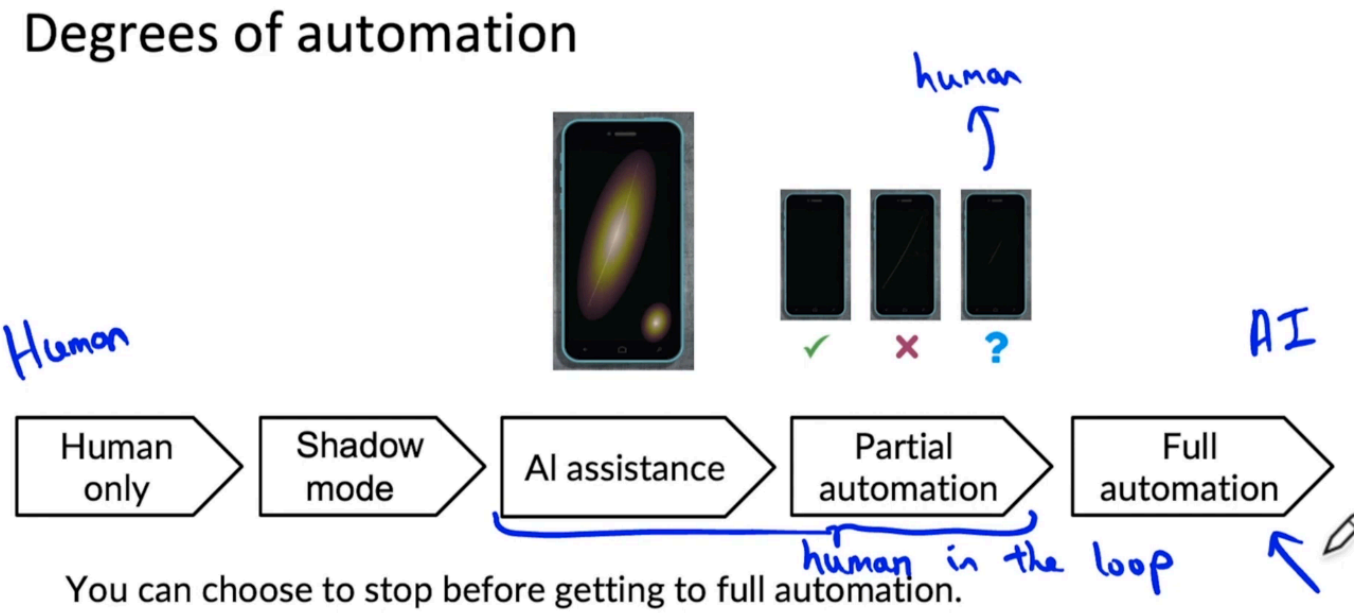
--> gather data of how the algorithm performs and how that compares to the human judgements

- Replace previous ML system
  - **[Blue-Green Deployment]**: let the router switch over the traffic between old and new versions



    - Easy way to enable rollback
- **Key Ideas:**
  - Gradual ramp up with monitoring --> rather than sending tons of traffic
  - Rollback: if the new algorithm is not working, can revert back to the previous system

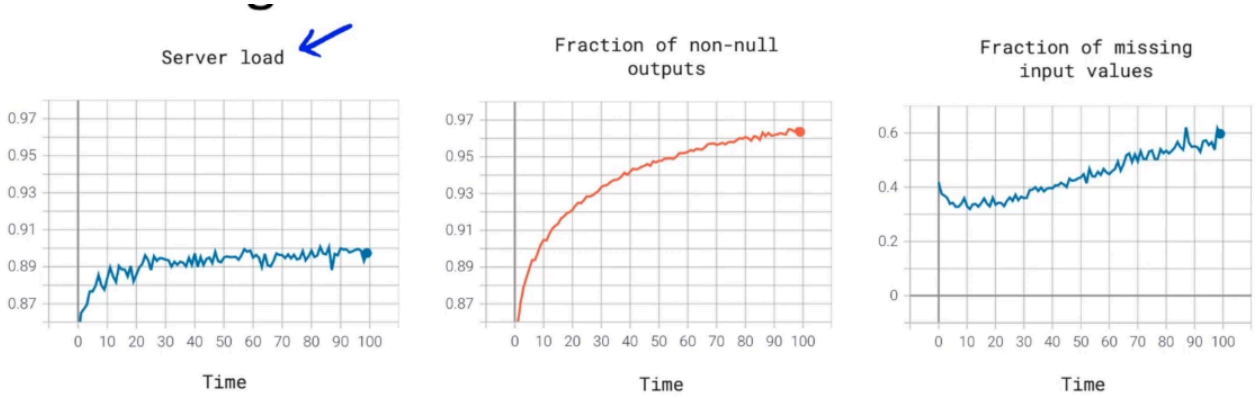
- **Import Framework: Degrees of Automation**



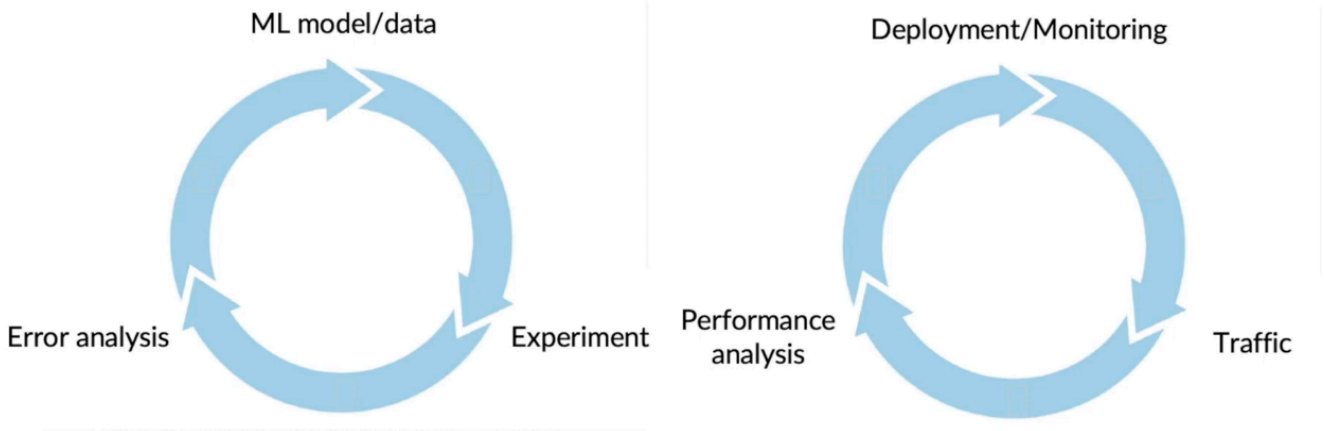
5. Monitoring

- **Monitoring Dashboard**

  --> The most common way to monitor a machine learning system to track performance over time



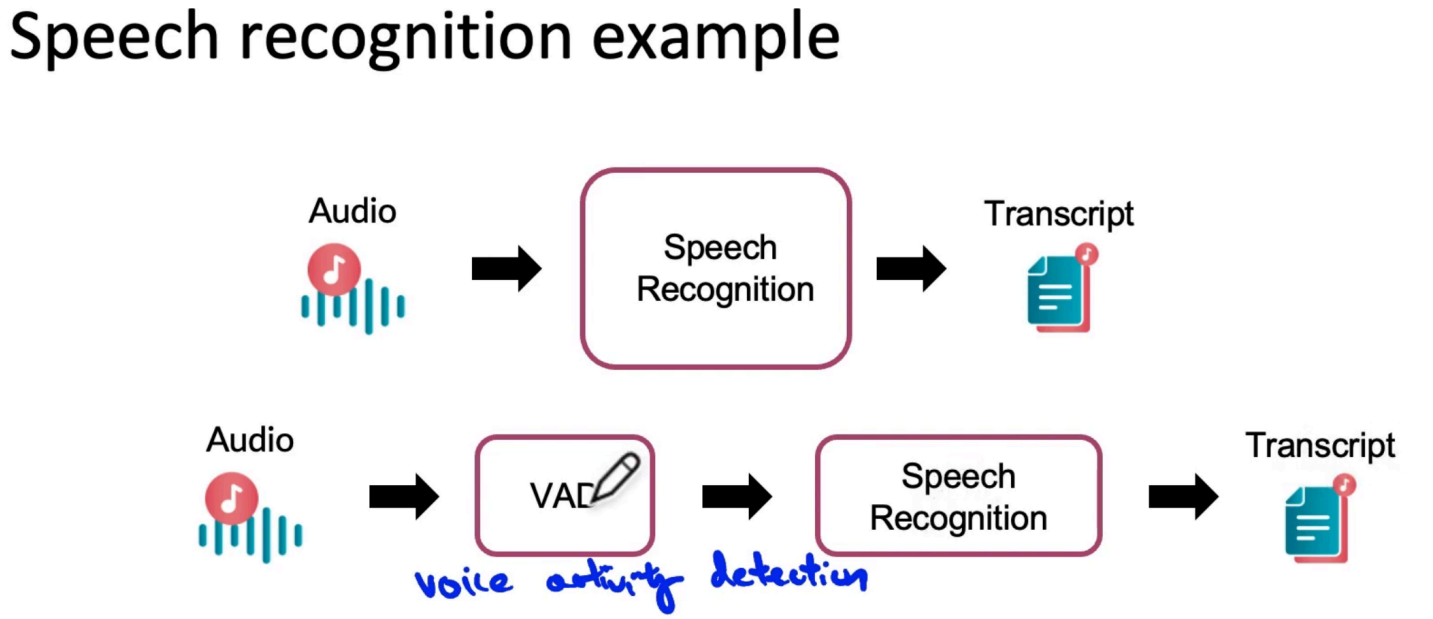
  - Brainstorm the things that could go wrong.
  - Brainstorm a few statistics/metrics that will detect the problem.
  - It is ok to use many metrics initially and gradually remove the ones you find not useful.
  - Set thresholds for alarms
  - Adapt metrics and thresholds over time
- **Metric Examples to Track**
  - Software Metrics: Memory, compute, latency, throughput, server load
  - Input Metrics (x): Average input length, average input volume, number of missing values (very common with structured data), average image brightness **[for each component]**
  - Output Metrics (y): number of times return `''/None`; number of times the user redoes search (may misrecognize the user's query the first time around); number of times the user swithes to typing; CTR (click-through rate) **[spot problems]**
- **Deployment is also Iterative as well as Modeling**



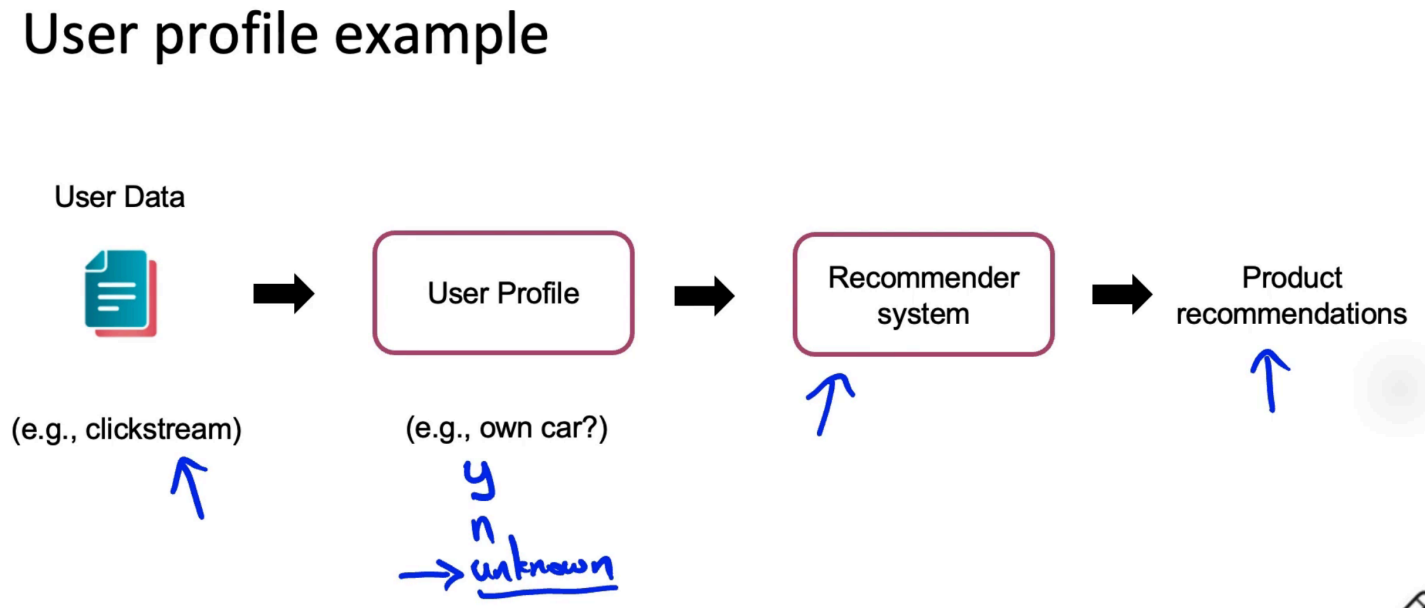
- **Model Maintenance**
  - Manual Retraining (more common)

- Automatic Retraining --> happen more in consumer software services

6. Pipeline Monitoring



--> the second example has two modules of learning algorithms, and the performance of the first module might affect the performance of the second module



How quickly do they change?

- User data generally has slower drift.
- Enterprise data (B2B applications) can shift fast.