THE MLOPS Process:

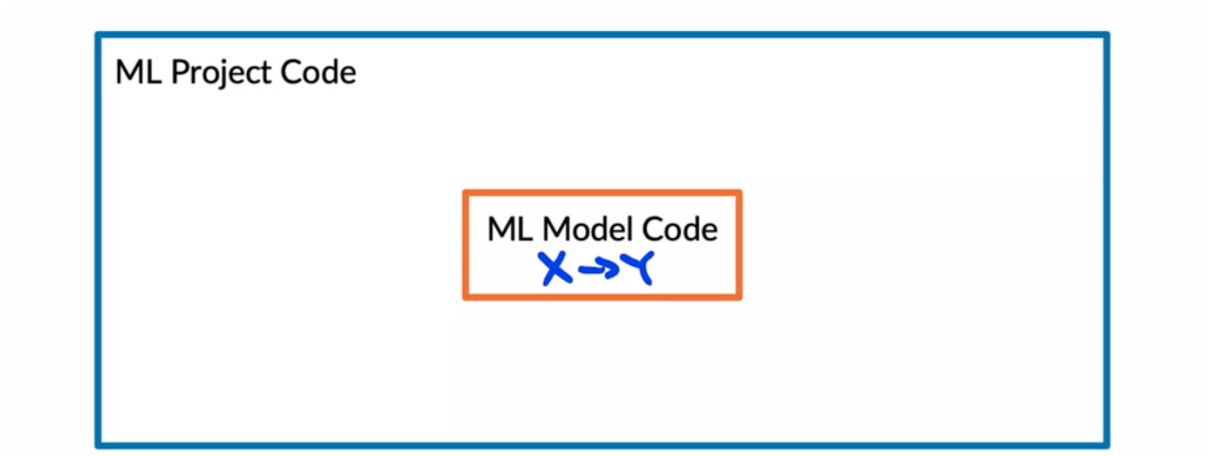
- Designing the ML solution
    - Establishing functional and non-functional requirements
- ML Experimentation and Development
    - Delivering a stable MML model to be run in production.
- ML Operations
    - Testing, Versioning, Continuous delivery, and monitoring.

Issues in a production system:

- Data drift: The distribution of data in the real world shifts or changes in respective of the data that the model was trained on, so the model needs to be aware of that.

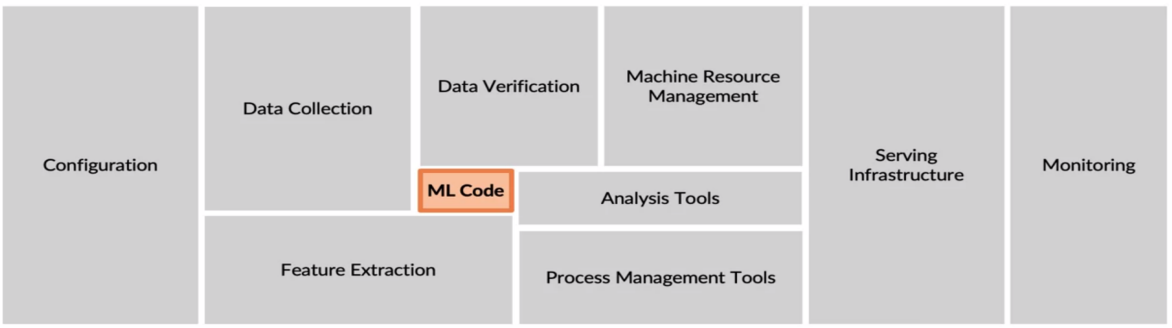
Experience in both machine learning and software to be successful, the final product is not a single result, is a service or product that can operate continuously.

ML code in production:



Machine learning in production is a lot more than machine learning code. 5%-10% is machine learning code. That's why POC in jupyter notebook to actual Deployment POC can still do a lot of work. **POC to Production GAP.**

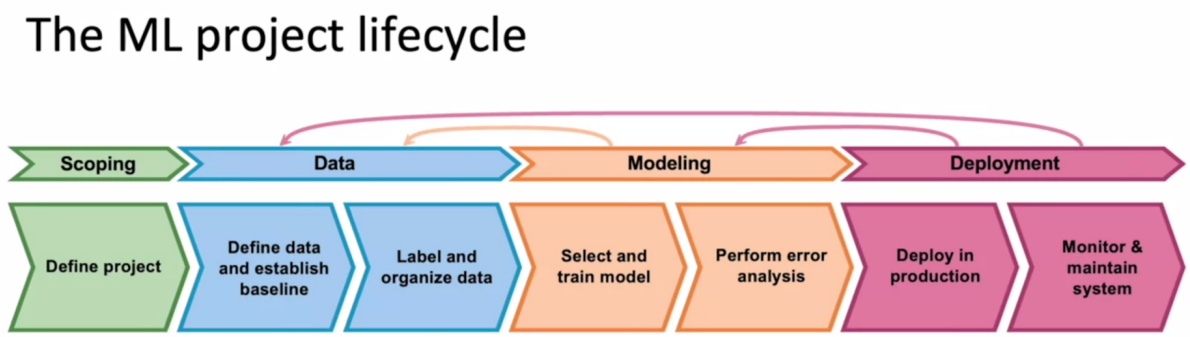
The requirements surrounding ML infrastructure:



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

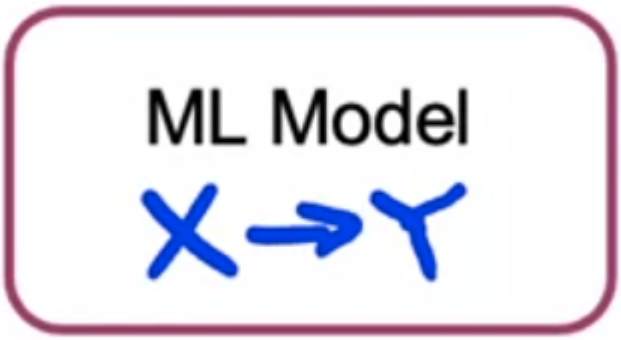
Steps of an ML Project (LifeCycle):

- Scoping: We define the project, What exactly we want to apply machine learning to?, What is $X -> Y$?
- Collect Data: Defining the data and establish baseline, labeling, and organize data.
- Modeling: Research, Select and train the model, also perform error analysis. (This part you probably will go back and iterate between annotate more data o change the model).
- Deployment: Deploy in production, Monitor & Maintain system. (Monitor for data distribution changes), probably after some time, you will need to update your data, retrain the model, and then update the deployment model.



Case Study: Speech Recognition application

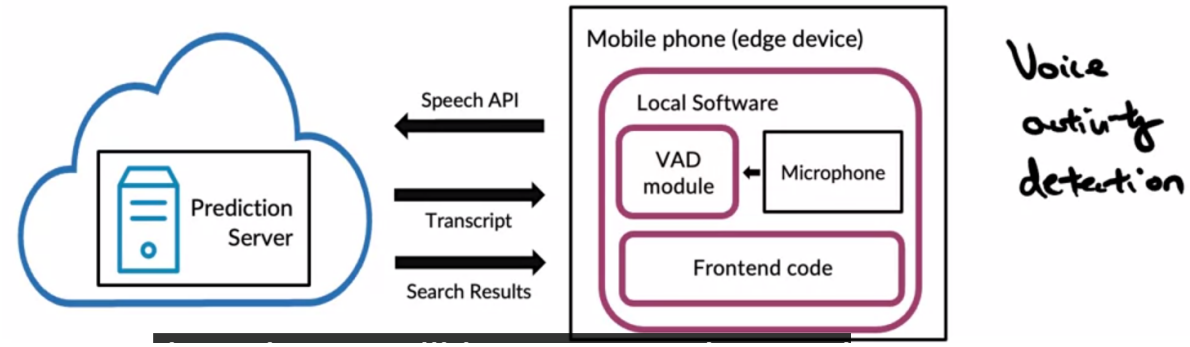
- Scoping: Decide to work on speech recognition for voice search
- Decide on key metrics:
  - Accuracy: How accurate is the speech recognition system?
  - Latency: How long does the system take to transcribe speech into text?
  - Throughput: How many queries per second can we handle?
- Estimate resources and timeline: Budget, Time, resources, and so on.
- Define data: is the data labeled consistently? consistently conventions in the annotation.
  - How much silence before/after each clip?
  - How to perform volume normalization?
  - Don't have to keep the dataset fix! Making sure you have quality data.
- Modeling: the code (algorithm/model), Hyperparameters, Data



For production and deployment, usually, the tendency is to keep the code fix and optimize the data and hyperparameters. In a research environment, usually, the tendency is to keep the data fixed and optimize the learning algorithm and the hyperparameters.

Use an open-source implementation - and optimize your data and the code too if it is necessary. Error analysis will help you to see how to optimize the data.

- Deployment:



Key challenge: Concept drift/data drift in production, Monitors to spot those problems and fix them!

We will learn the life cycle backward :)

MLOPS: the key idea of MLOPS is those systematic ways to think about scoping, data, modeling, and deployment, and also software tools to support the best practices.

Deployment Week 1:

challenges:

- Machine learning or statistical issues
- Software engine issues

Concept Drift or data drift:
What if your data changes after your system have already been deployed?

- **Gradual changes:** Sutil changes in our data, but very slowly.
- **Sudden shock:** Big changes. For example the start of the covid global pandemic.

concept-drift: change the desired mapping.
data-drift: change the X (train data).

- Detect and manage any changes.

Software engineering issues:

Checklist of questions:

- Realtime or Batch.
- Cloud vs Edge/Browser
- Compute resources (CPU/GPU/memory)
- Latency, Throughputs (QPS - query time per seconds)
- Logging (logging all the new data entries for new retrain)
- Security and privacy.

Deploying systems require two sets abroad of tasks:

- Writing the software to enable you to deploy the system in production
- Monitoring your system in production to handle concept/data drift

The first deployment is just half the way to finish, after this first deploy you need to maintain your system

Deployment patterns:

1. New product/capability: Start with a small amount of traffic and gradually increase it.
2. Automate/assist with the manual task: Shadow mode deployment.
3. Replace previous ML system:

    Key Ideas:

    - Gradual ramp up with monitoring
    - Rollback (If the algorithm isn't working, roll back to a previous system)

Shadow mode deployment:

The ML system shadows the human and runs in parallel. The ML system's output is not used for any decisions during this phase.

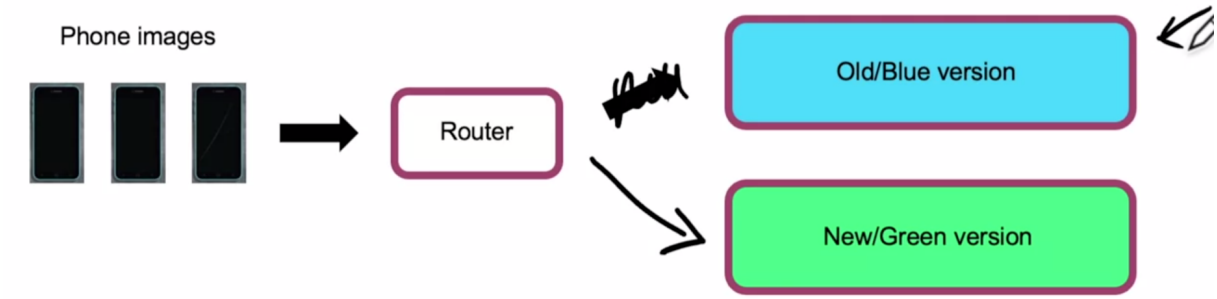
The purpose of a shadow mode allows you to gather data about the algorithm performance and how that compares to the human judgment, and verify if the algorithm is accurate and decide whether or not to allow the system to make real decisions.

Canary deployment:
When the learning algorithm is ready to make real predictions.

- Roll out to a small fraction (say 5%) of traffic initially.
- Monitor system and ramp up traffic gradually.

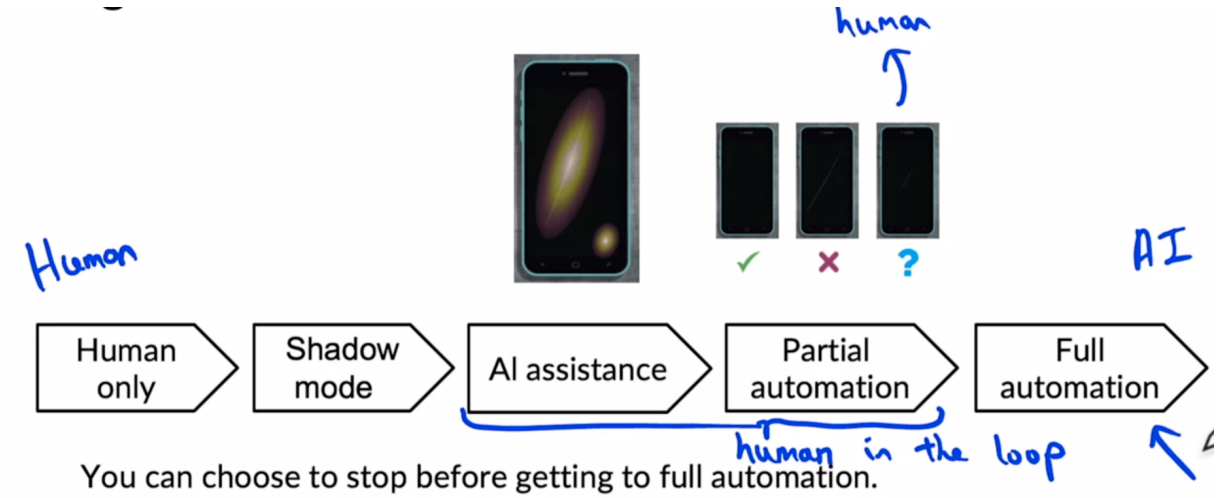
This deployment allows you to spot problems early on before there are consequences.

Blue-green deployment:



Easy way to enable rollback, switch to send to green version some percentage of requests and increase slowly.

Degrees of automation:

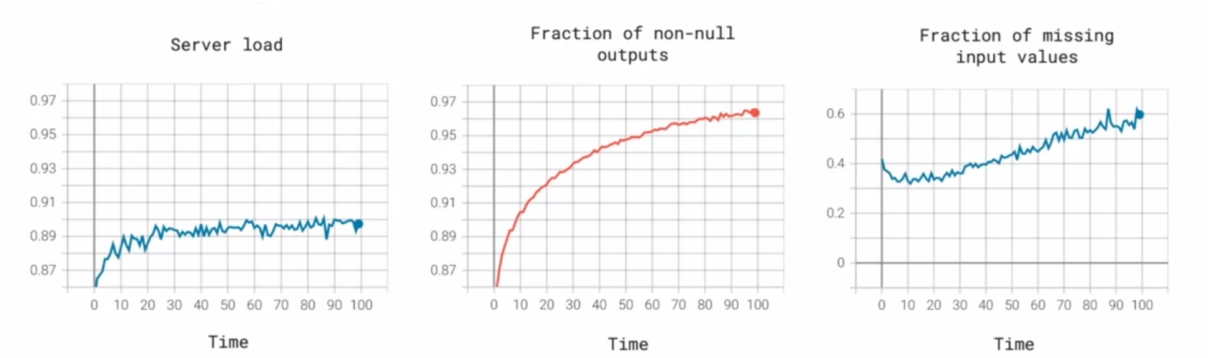


Partial automation has into account the learning algorithm confidence output, if it is not a 0 or 1, send it to a human to review the algorithm decision.

There are some applications that human in the loop is not feasible. For example consumer internet applications such as Web search engine, recommendation system, Online speech recognition system.

Monitoring:

- Monitoring dashboard:

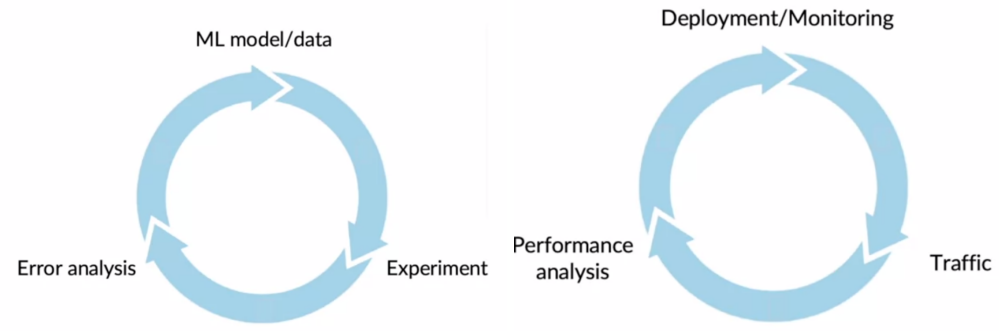


  - Brainstorm the things that could go wrong. With your team.
  - Brainstorm a few statistics/metrics that will detect the problem.
  - It is ok to use many metrics initially and gradually remove the ones you find not useful.

Examples of metrics to monitor:

- Software metrics: Memory, compute, latency, throughput, server load. (To monitor the software implementation is good)
- Input metrics: Avg input length (input for speech recognition system), Avg input volume, Num missing values, Avg image brightness (to know if image condition system changes), These metrics help to monitor data drift issues.
- Output metrics: Times return “” (null), times user redoes search (make the same request with slightly equal data) this is a sign that the model fails to recognize, times user switches to typing (for speech recognition), CTR (for web search) click-through rate. These metrics can help you figure out if Y has changed in a significant way.

JUST AS ML MODELING IS ITERATIVE, SO IS DEPLOYMENT!!!!

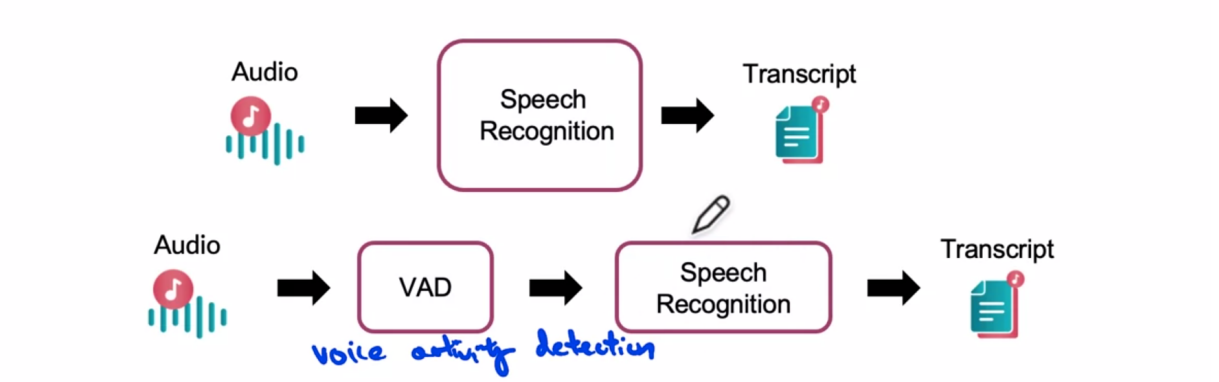


Common practices for dashboards:

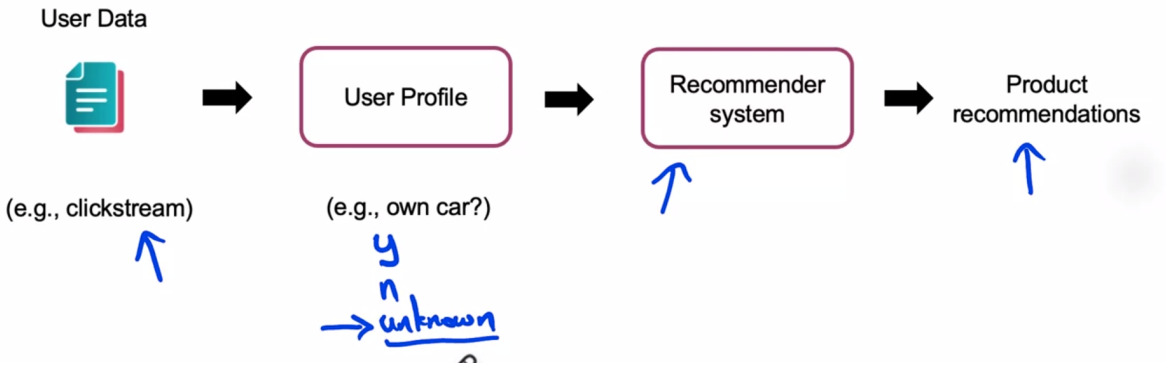
- Set thresholds for alarms!
- Adapt metrics and thresholds overtime to make sure that they flag the respective important alarms to you.
- Many machine learning will need maintenance:
  - Manual retraining. -> this is more common.
  - Automatic retraining. -> this happens in a consumer internet application.

Pipeline monitoring:

Many AI systems are not only one single model but a multi pipeline model.



changes on the first module affect the second model because the second model input depends on the first model output.



- Brainstorm metrics to monitor:
    - Software metrics
    - Input metrics
    - Output metrics

For individual models in the pipeline will help you to spot new metrics for model-dependent.

How quickly do the data change?

- **very problem dependant.**
- Enterprise data (B2B applications) can shift fast.

Some extra information for this first week:

Week 1: Overview of the ML Lifecycle and Deployment

Concept and Data Drift

Monitoring ML Models

A Chat with Andrew on MLOps: From Model-centric to Data-centric

Papers

Konstantinos, Katsiapis, Karmarkar, A., Altay, A., Zaks, A., Polyzotis, N., … Li, Z. (2020). Towards ML Engineering: A brief history of TensorFlow Extended (TFX). http://arxiv.org/abs/2010.02013

Paleyes, A., Urma, R.-G., & Lawrence, N. D. (2020). Challenges in deploying machine learning: A survey of case studies. http://arxiv.org/abs/2011.09926

Sculley, D., Holt, G., Golovin, D., Davydov, E., & Phillips, T. (n.d.). Hidden technical debt in machine learning systems. Retrieved April 28, 2021, from Nips. c https://papers.nips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf