Machine Learning Systems Design

Introduction to ML System Design

Lecture 3: ML System Development Life Cycle

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io

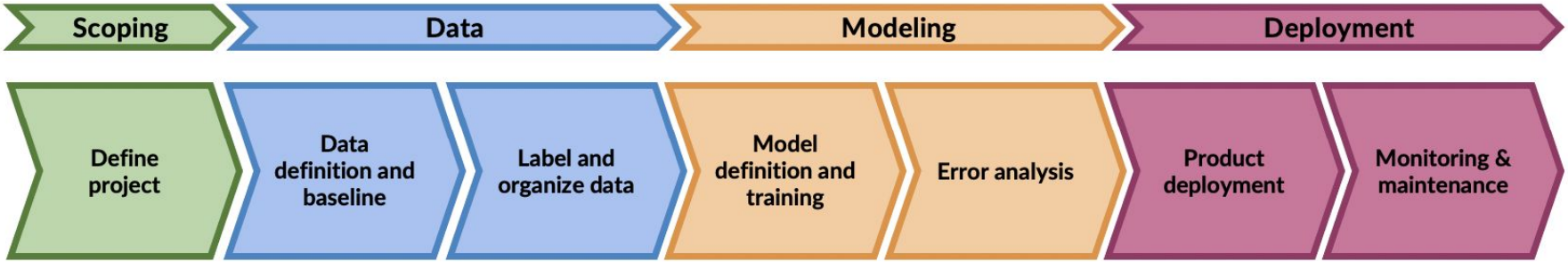
Agenda

1. Iterative Process of ML System Design
2. Data-centric AI
3. MLOps

1. Iterative Process of ML System Design

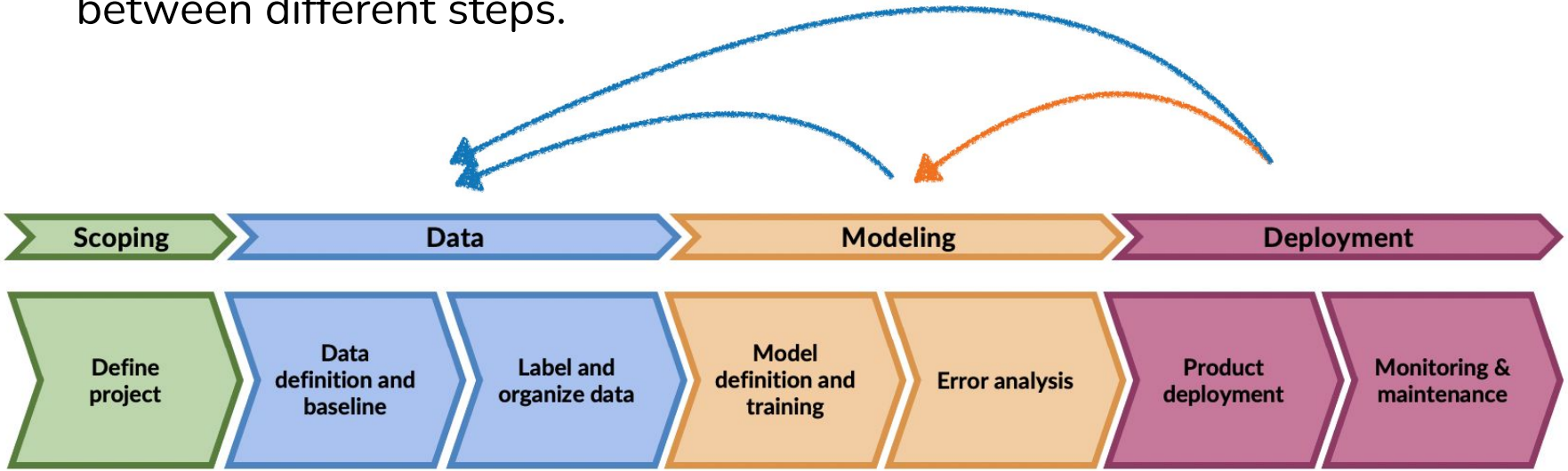
ML system development life cycle

- Developing an ML system is an iterative and, in most cases, never-ending process.

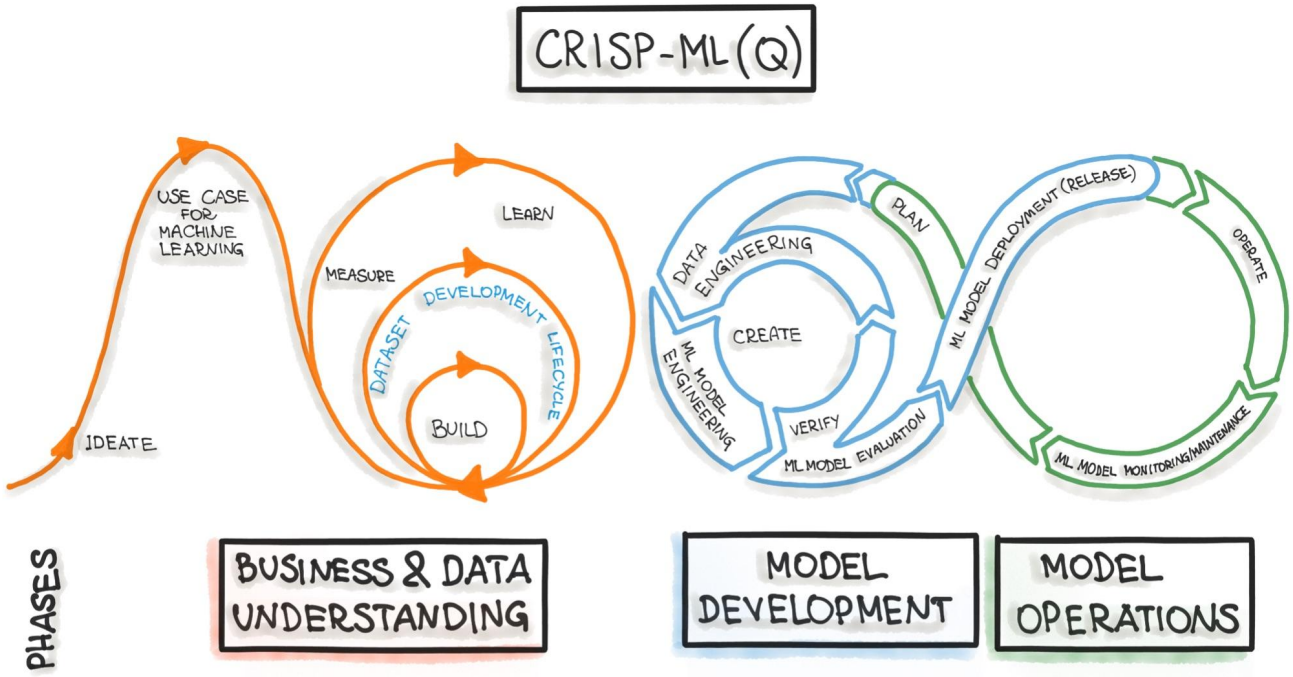


ML system development life cycle

- But, It is not a straightforward process, there are lot of back and forth between different steps.

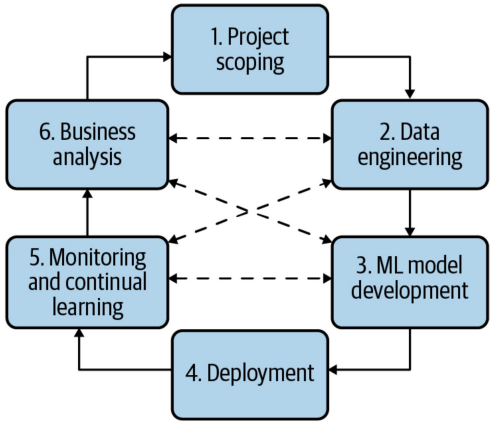


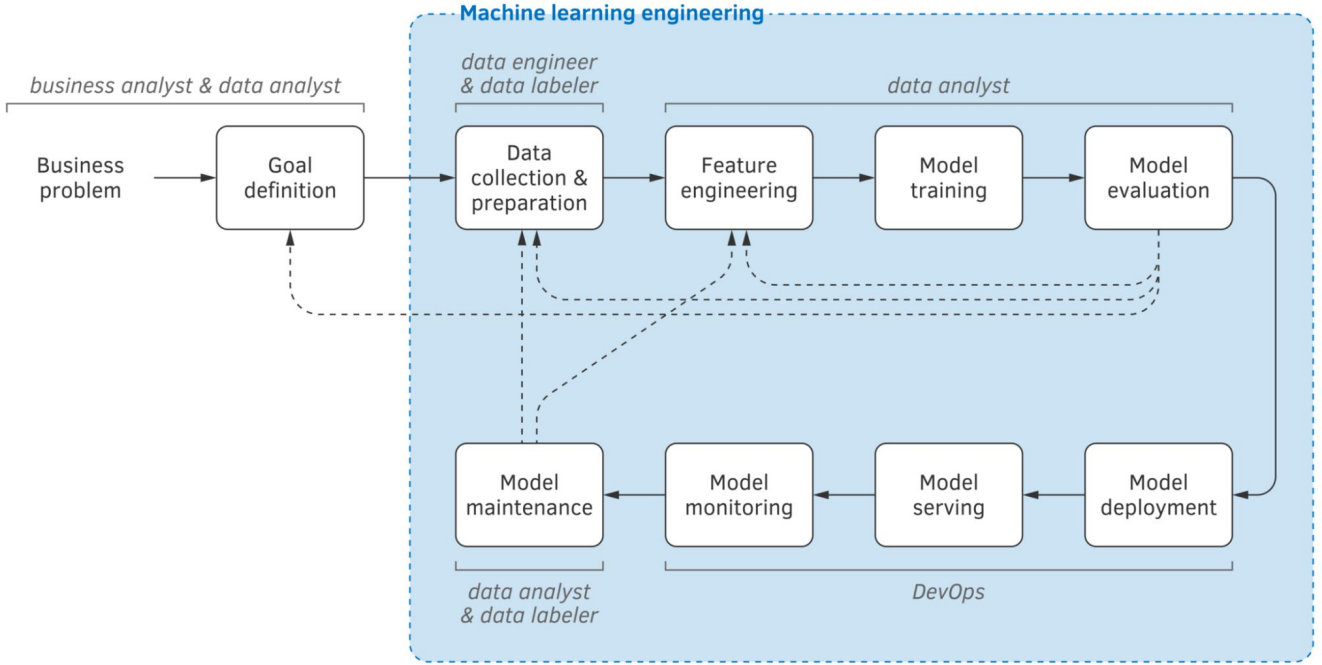
ML system development life cycle



CRISP-ML(Q): *Cross-Industry Standard Process for the development of Machine Learning applications with Quality assurance methodology*

ML system development life cycle





There are always four main steps: ***scopeing, data, modeling, deployment***

Example: Ad click prediction

1. Choose a metric to optimize. For example, you might want to optimize for impressions—the number of times an ad is shown.
2. Collect data and obtain labels.
3. Engineer features.
4. Train models.
5. During error analysis, you realize that errors are caused by the wrong labels, so you relabel the data.
6. Train the model again.

Example: Ad click prediction

7. During error analysis, you realize that your model always predicts that an ad shouldn't be shown, and the reason is because 99.99% of the data you have have NEGATIVE labels (ads that shouldn't be shown). So you have to collect more data of ads that should be shown.
8. Train the model again.
9. The model performs well on your existing test data, which is by now two months old. However, it performs poorly on the data from yesterday. Your model is now stale, so you need to update it on more recent data.
10. Train the model again.
11. Deploy the model.

Example: Ad click prediction

12. The model seems to be performing well, but then the business people come knocking on your door asking why the revenue is decreasing. It turns out the ads are being shown, but few people click on them. So you want to change your model to optimize for ad click-through rate instead.
13. Go to step 1

2. Data-centric AI

Data-centric AI Movement

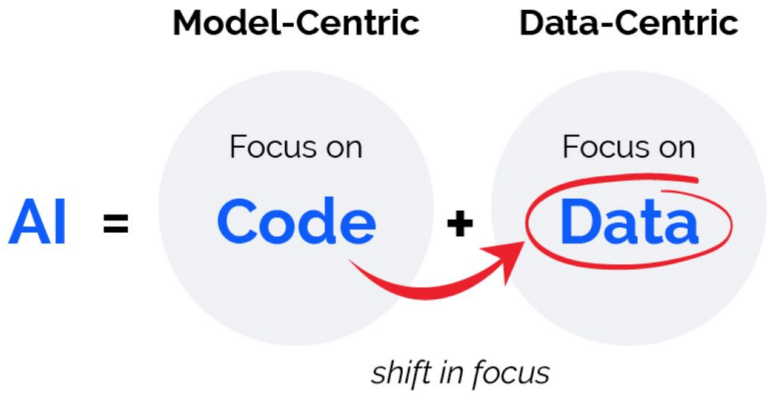
"Instead of focusing on the code, companies should focus on developing systematic engineering practices for improving data in ways that are reliable, efficient, and systematic. In other words, ***companies need to move from a model-centric approach to a data-centric approach****."*

— Andrew Ng, CEO and Founder of LandingAI

Watch this!

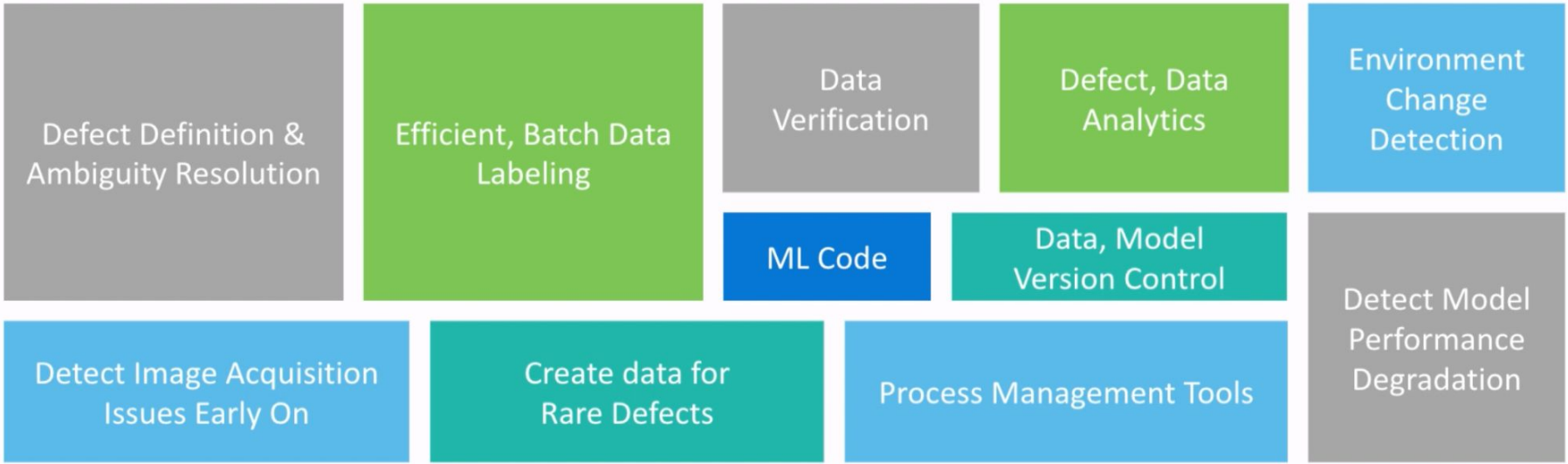
What Is Data-Centric AI?

- Think of a Data-Centric AI system as programming with focus on data instead of code



Check landing AI and DCAI

What Is Data-Centric AI?

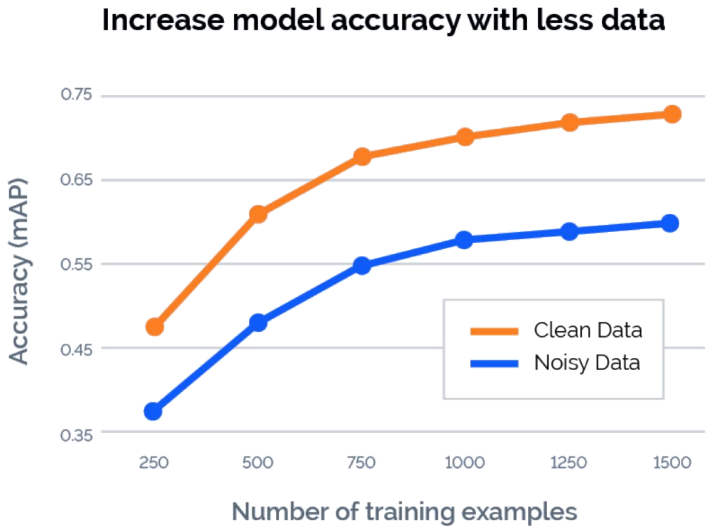


Only **5% - 10%** of the workload in AI projects is building ML code

Why Does Data-Centric AI Matter?

- With low-quality data you may not have any improvement over the baselines

Computer vision task (steel sheet inspection)	Accuracy
Baseline	**76.2%**
Model-Centric	**+0%**
Data-Centric	**+16.9% (93.1%)**



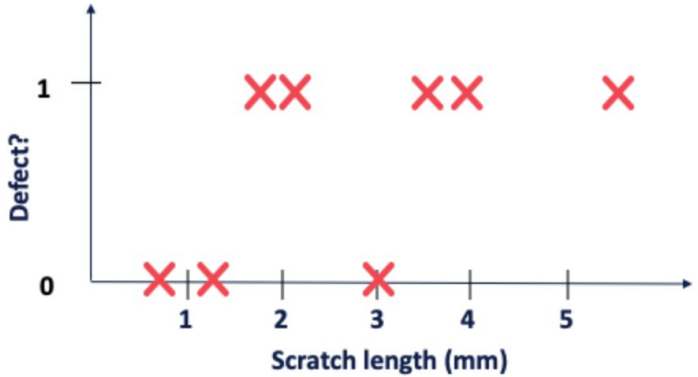
Some Tips for a Data-Centric AI Approach

- You’re trying to help a pharmaceutical company inspect pills to find defects, like scratches.
- So, you start creating a dataset and label it

See tips for dcai blog post

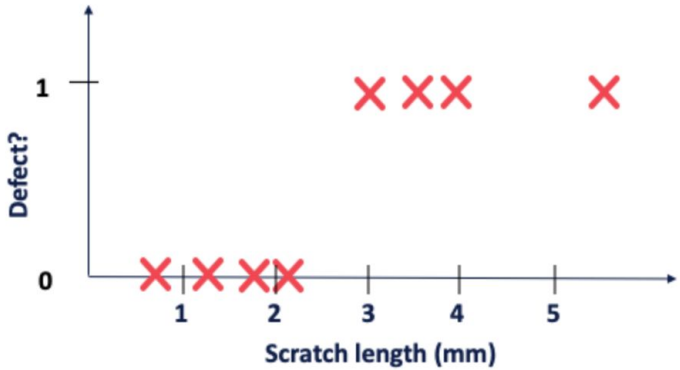
Tip 1: Make the labels y consistent

- Even expert inspectors, will disagree with each other on labels!



Tip 1: Make the labels y consistent

- Select a threshold (clear rule) for labelers to have more consistent labels.



Tip 2: Use multiple labelers to spot inconsistencies

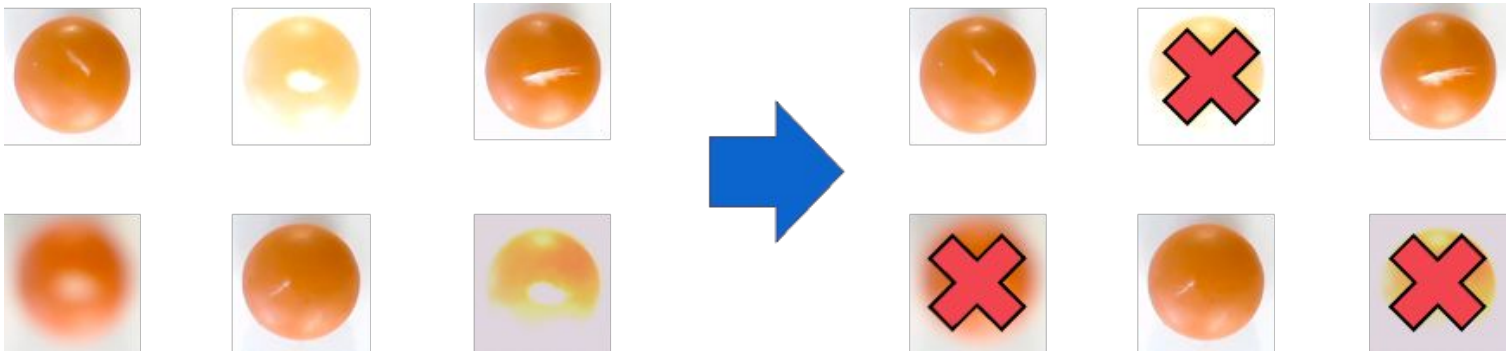
- Ask more than one labelers to label the same example

Tip 3: Repeatedly clarify labeling instructions by tracking down ambiguous examples

- Identifying ambiguous examples can help you clarify instructions for how to label edge cases.
- Then, provide a definitive labeling decision in your documentation

Tip 4: Toss out bad examples. More data is not always better!

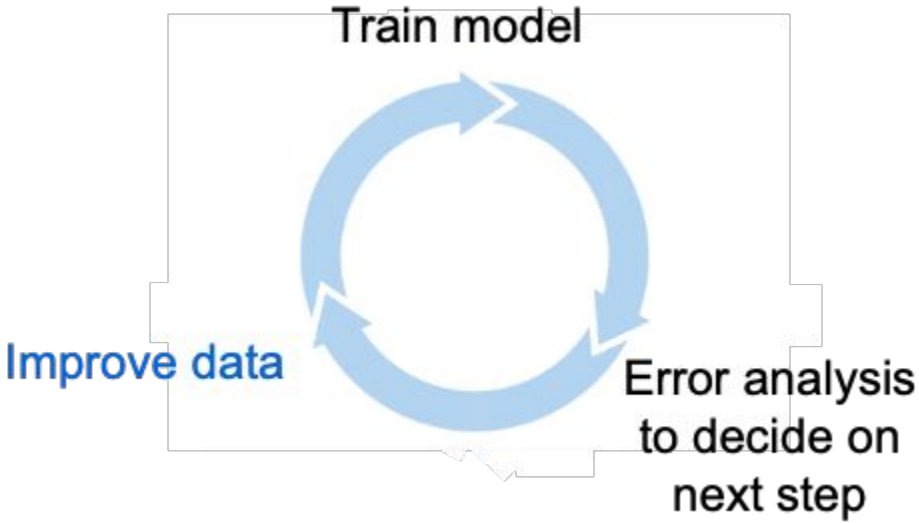
- It can be much more efficient to fix the image acquisition to get better images and make the problem easier for the algorithm.



Tip 5: Use error analysis to focus on subset of data to improve

- Trying to improve all these different aspects of the data all at once can be too broad and unfocused an effort.

Summary


Train model
Improve data
Error analysis to decide on next step

Mind vs Data debate

- **Mind** might be disguised as inductive biases or intelligent architectural designs.
- **Data** might be grouped together with computation.

Mind vs Data debate

"*ML will not be the same in 3–5 years, and ML folks who continue to follow the current data-centric paradigm will find themselves outdated, if not jobless. Take note.*"

— **Judea Pearl,** Turing Award winner computer scientist

Mind vs Data debate

"*Huge computation and a massive amount of data with a simple learning algorithm create incredibly bad learners. The* ***structure*** *allows us to design systems that can learn more from less data.*"

— **Christopher Manning,** director of the Stanford Artificial Intelligence Laboratory

Mind vs Data debate

*“The biggest lesson that can be read from 70 years of AI research is that g**eneral methods that leverage computation** are ultimately the most effective, and by a large margin.... Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation”*

— **Richard Sutton**, professor of computing science at the University of Alberta and a distinguished research scientist at DeepMind

Mind vs Data debate

“*We don’t have better algorithms. We just have more data*”

— **Peter Norvig,** Google’s director of search quality, Distinguished Education Fellow at the Stanford

Mind vs Data debate

- Regardless of which camp will prove to be right eventually, no one can deny that data is essential, for now.

3. MLOps

Myth #1: Deploying is hard

Myth #1: Deploying is hard

Deploying is easy. Deploying reliably is hard

Myth #2: You only deploy one or two ML models at a time

Myth #2: You only deploy one or two ML models at a time

Booking.com: 150+ models, Uber: thousands

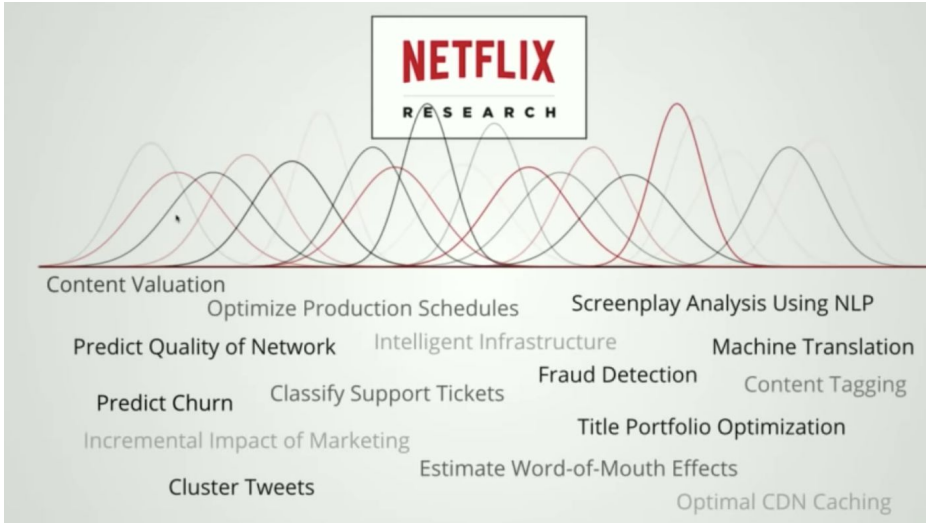


Image from Ville Tuulos (Netflix, Outerbounds)

Myth #3: You won’t need to update your models as much

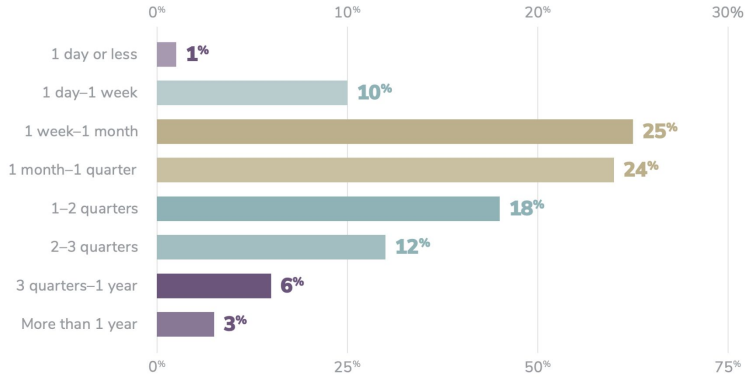
DevOps: Pace of software delivery is accelerating

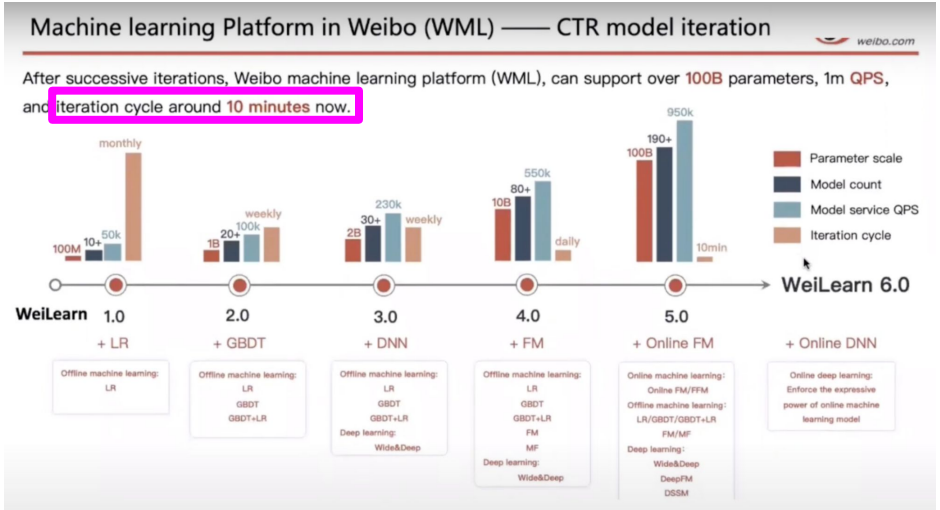
- Elite performers deploy **973x** more frequently with **6570x** faster lead time to deploy (Google DevOps Report, 2021)
- DevOps standard (2015)
  - Etsy deployed 50 times/day
  - Netflix 1000s times/day
  - AWS every 11.7 seconds

DevOps to MLOps: Slow vs. Fast

We'll learn how to do minute-iteration cycle!

Only 11% of organizations can put a model into production within a week, and 64% take a month or longer





Left image from Algorithmia | Right image: Machine learning with Flink in Weibo (Qian Yu, QCon 2019)

Accelerating ML Delivery

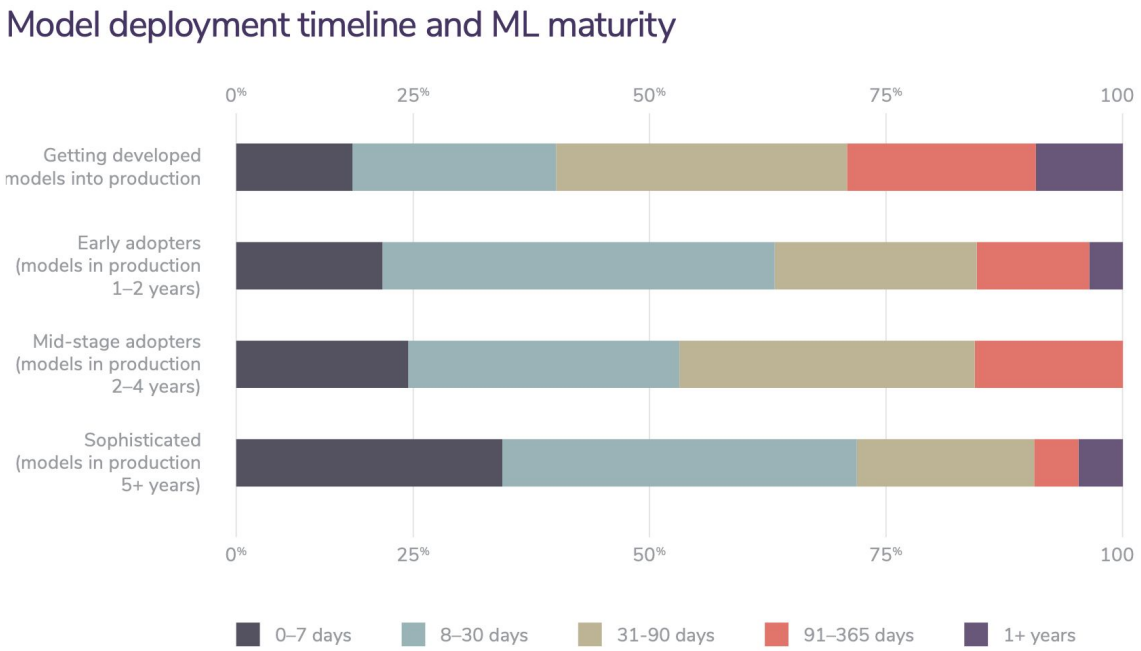
ML + DevOps = 

Myth #4: ML can magically transform your business overnight

Myth #4: ML can magically transform your business overnight

Magically: possible
Overnight: no

Efficiency improves with maturity



2020 state of enterprise machine learning (Algorithmia, 2020)

ML engineering is more engineering than ML

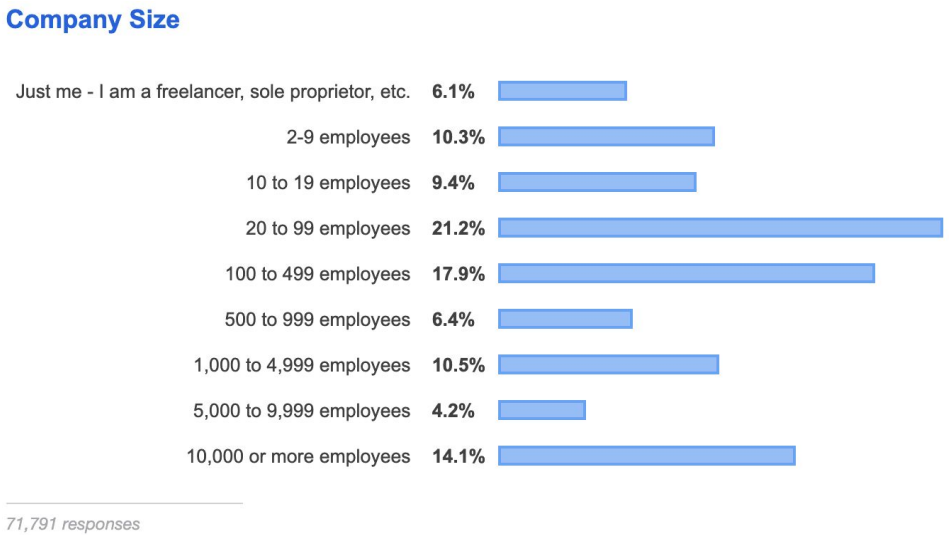
MLEs might spend most of their time:

- wrangling data
- understanding data
- setting up infrastructure
- deploying models

instead of training ML models

Myth #5: Most ML engineers don’t need to worry about scale

Myth #5: Most ML engineers don’t need to worry about scale



StackOverflow Developer Survey 2019

Challenges of ML lifecycle management

- **Manual labor** from data to deployment.
- **Disconnection between teams** makes the deployment process more complex.
- **Scalability** issue sets a limit for an organization to scale up its machine learning applications while relying on manual processes.

Best practices for ML lifecycle management

- **Automation of the lifecycle** decreases the time allocated to resource-consuming steps such as feature engineering, model training, monitoring, and retraining. It frees up time to rapidly experiment with new models.
- **Standardization of the process** enables efficient communication between diverse teams.
- **Continuous training** of deployed ML model is required to maintain model performance.

Key Features of MLOps Platforms and Tools

Data Acquisition and Preparation					
Data Acquisition	Data Visualization and Exploration	Data Preparation and Transformation	Data Versioning	Data Pipelines	Data Labeling
Model Development and Training					
AutoML	Feature Extraction and Engineering	Feature Store	ML Pipelines or Workflows	Development Environment Support	Model Repository
Model Marketplace	Model Training	Distributed Model Training	Model Debugging	Experiment Management	Deep Learning Support
Reinforcement Learning Support	Bias Detection and Mitigation	Model Explainability	Tuning and Optimization		
Model Deployment and Operations					
Model Portability and Compression	Model Deployment	Edge ML Support	Model Monitoring	Cost Management	
System-wide features					
ML Infrastructure Orchestration	Accelerator Support	Kubernetes Support	Teamwork and Collaboration	Enterprise Security	Governance

MLOps tools

MLOps tools can be divided into three major areas dealing with:

- Data management
- Modeling
- Operationalization

See Made With ML, and Awesome MLOps for a complete list of tools!

MLOps platforms

MLOps platforms providing end-to-end machine learning lifecycle management:

- Google Cloud AI Platform
- Amazon SageMaker
- Azure Machine Learning
- Databricks

Data management tools

- **Data Labeling** tools are used to label large volumes of data such as texts, images, or audio, e.g., Label Studio.
- **Data Versioning** tools enable managing different versions of datasets and storing them in an accessible and well-organized way, e.g., DVC.
- **Feature Engineering** tools automate the process of extracting useful features from raw datasets to create better training data for machine learning models, e.g. Feast.

Modeling tools

- **Experiment Tracking** tools save all the necessary information about different experiments, e.g., MLFlow.
- **Hyperparameter Optimization** tools automate the process of searching and selecting hyperparameters that give optimal performance for machine learning models, e.g., hyperopt

Operationalization tools

- **Model Deployment/Serving** tools facilitate integrating ML models into a production environment to make predictions, e.g., Kubeflow
- **Model Monitoring** tools detect data drifts and anomalies over time and allow setting up alerts in case of performance issues, e.g., Evidently AI.

Hands-on MLOps

In the following chapters we will have:

- tutorial on MLFlow and DVC,
- assignments and final project to practice with MLOps tools.

Machine Learning Systems Design

Introduction to ML System Design

Next Lecture: Data Engineering Fundamentals

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io