Software Engineering for AI Applications

17-313 Fall 2025

Foundations of Software Engineering

https://cmu-17313q.github.io

Eduardo Feo Flushing

Outline

- Why ML/AI projects fail?
  - Data quality
  - Fairness issues
- What's wrong with the model-centric pipeline?
- Are there any new challenges?
- What is ML Ops?

Why ML/AI projects fail?

"***Data***" science

"Data cleaning and repairing account for about 60% of the work of data scientists."

What makes good quality data?

- Accuracy
  - The data was recorded correctly.
- Completeness
  - All relevant data was recorded.
- Uniqueness
  - The entries are recorded once.
- Consistency
  - Format, units, data agrees with itself
- Timeliness
  - The data is kept up to date.

Data is noisy

- Multiple sources
- Unreliable sensors or data entry
- Wrong results and computations, crashes
- Duplicate data, near-duplicate data
- Out of order data
- Data format invalid

Data quality and ML

- More data -> better models (up to a point)
- Noisy data (imprecise) -> less confident models
  - Some ML techniques are more or less robust to noise
- Inaccurate data: misleading models, biased models
- Need the "right" data
- Invest in data quality, not just quantity

Validating the model

- Validation data should reflect usage data
- Be aware of data/concept drift? (face recognition during pandemic, new patterns in credit card fraud detection)

Training Data

Real Data

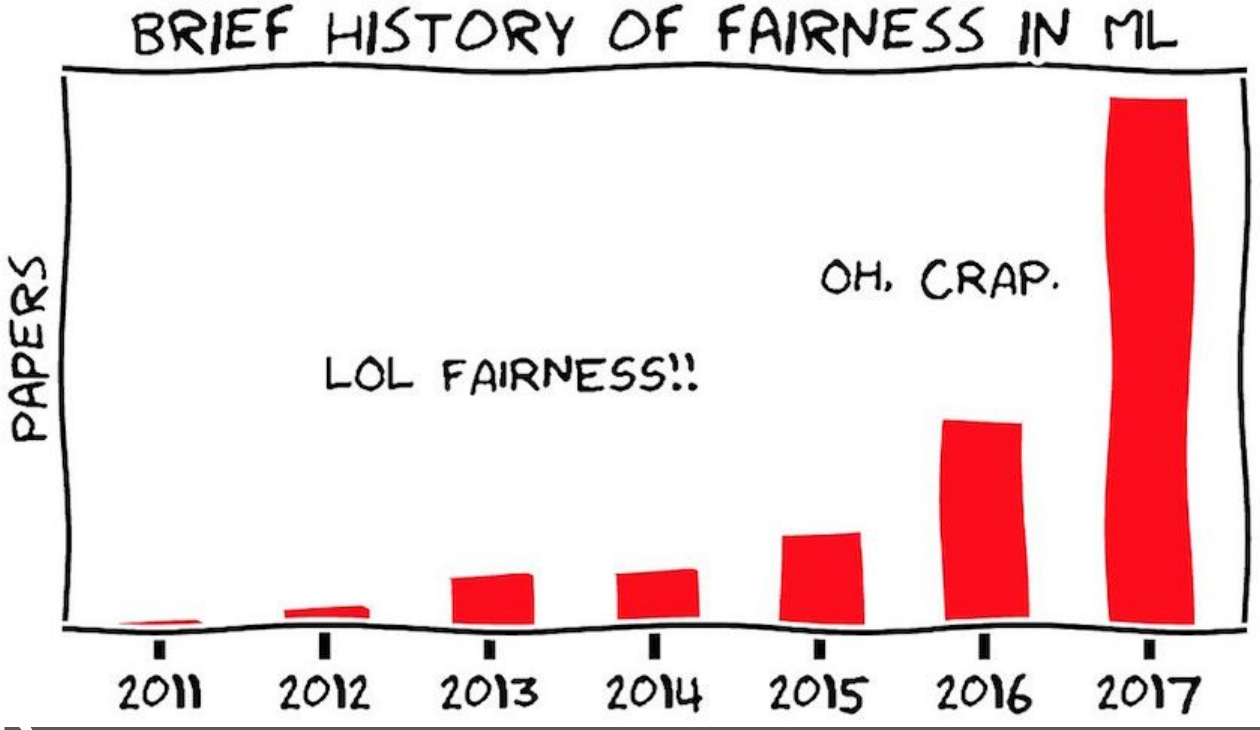
Fairness: What is fair?

Fairness discourse asks questions about how to treat people and whether treating different groups of people differently is ethical. If two groups of people are systematically treated differently, this is often considered unfair.

Philosophy: "what is fair is also what is morally right."

Law: "protect individuals and groups from discrimination or mistreatment with a focus on prohibiting behaviors, biases and basing decisions on certain protected factors or social group categories"

Fairness is still an actively studied & disputed concept!



Example - Mortgage Applications

- Home ownership is key path to build generational wealth
- Past decisions often discriminatory (redlining)
- Replace biased human decisions by objective and more accurate ML model
  - income, other debt, home value
  - past debt and payment behavior (credit score)
- Reduce operational costs and turn times within the mortgage process.

FORBES > MONEY

The Future Of Mortgage Lending: How AI And Humans Can Coexist

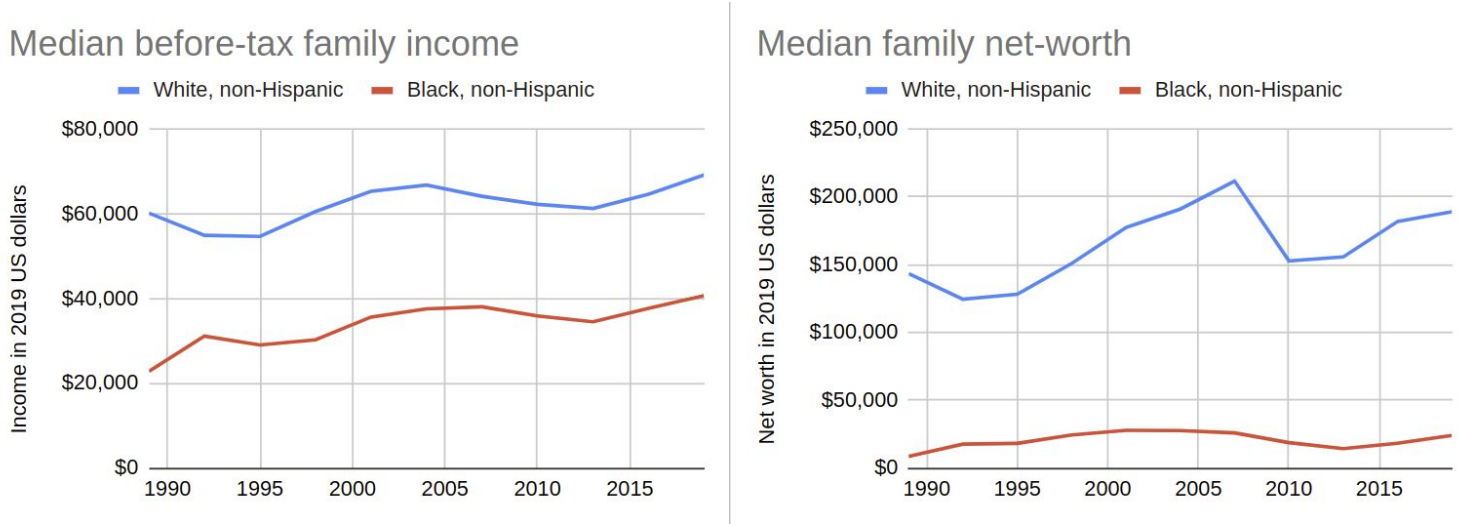
Forbes

Alec Hanson Forbes Councils Member

Forbes Finance Council COUNCIL POST | Membership (Fee-Based)

Mar 9, 2023, 07:30am EST

Past bias, different starting positions



Source: Federal Reserve's Survey of Consumer Finances

Varieties of fairness

- Group unaware
  - *Don't use sensitive attributes (e.g., gender, race) in the model*
- Demographic parity
  - *Outcomes (e.g., acceptance rate) should be similar across groups*
- Equalized odds
  - *Accuracy of predictions should be equal across groups*

Outline

- Why ML/AI projects fail?
  - Data quality
  - Fairness issues
- **What's wrong with the model-centric pipeline?**
- Are there any new challenges?
- What is ML Ops?

Why ML/AI projects fail? What's wrong?

NATIONAL HARBOR Md., June 7, 2022

Gartner Predicts Half of Finance AI Projects Will Be Delayed or Cancelled By 2024

FORBES > INNOVATION

Why Most Machine Learning Applications Fail To Deploy

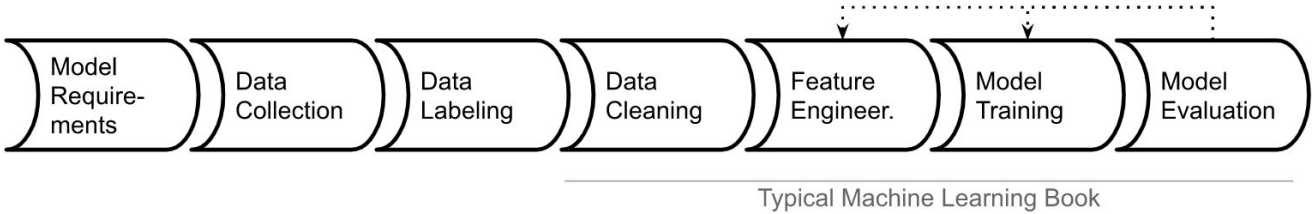
Usama Fayyad Forbes Councils Member

Forbes Technology Council COUNCIL POST | Membership (Fee-Based)

Apr 10, 2023, 08:45am EDT

Model-centric vs system-wide focus

- Traditional Model Focus (data science)

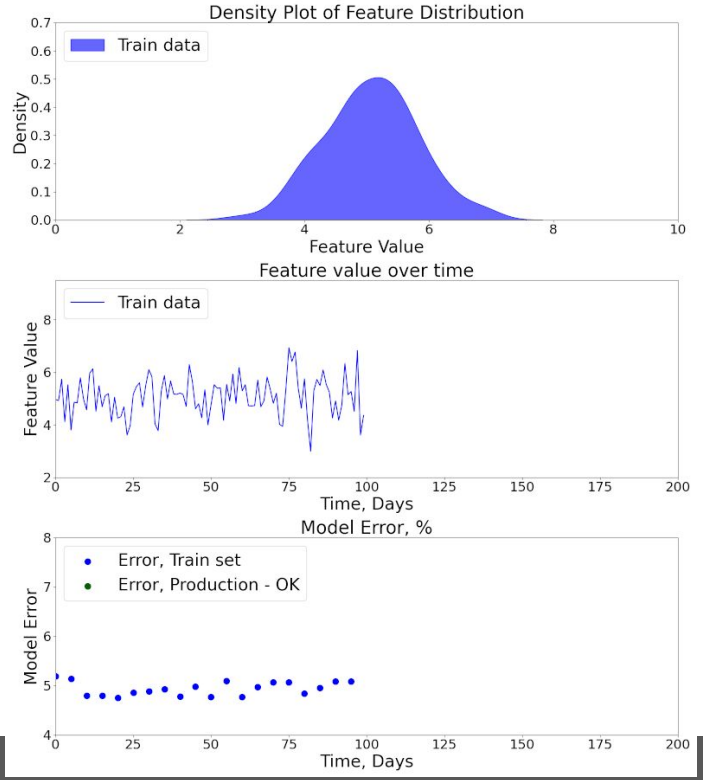


What's wrong with the model-centric pipeline?

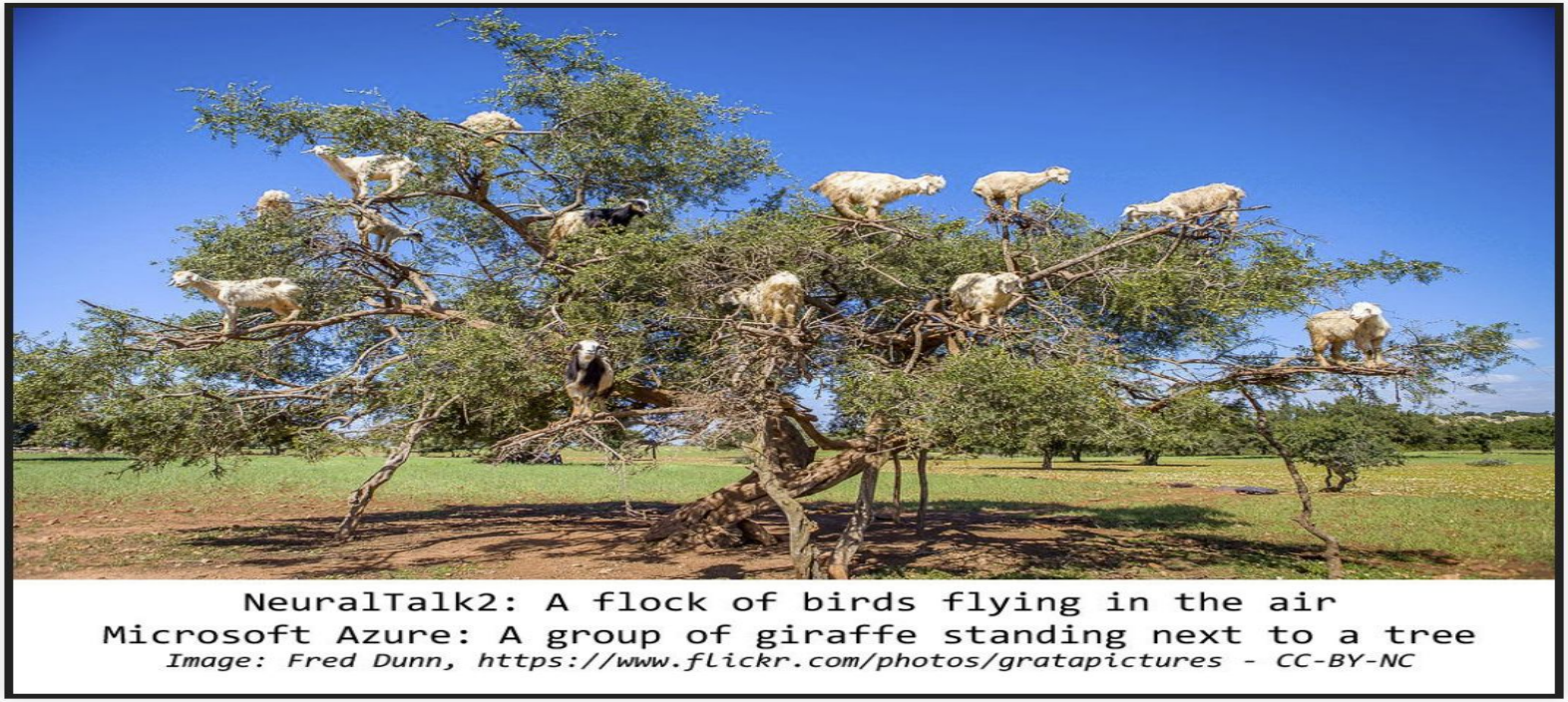
World is not static

- Concepts drift
  - ML estimates `f(x) = y`
  - What if the relationship between x & y changes over time?
- Data drift
  - Statistical properties of the input data change over time
  - Causes:
    - External factors (e.g., market trends, user behavior shifts).
    - Sensor recalibrations or environmental changes.
    - Changes in data collection methods or quality.
  - Impact:
    - Model performance degrades as the training data no longer accurately represents the real-world scenario.

Data drift - Monitoring



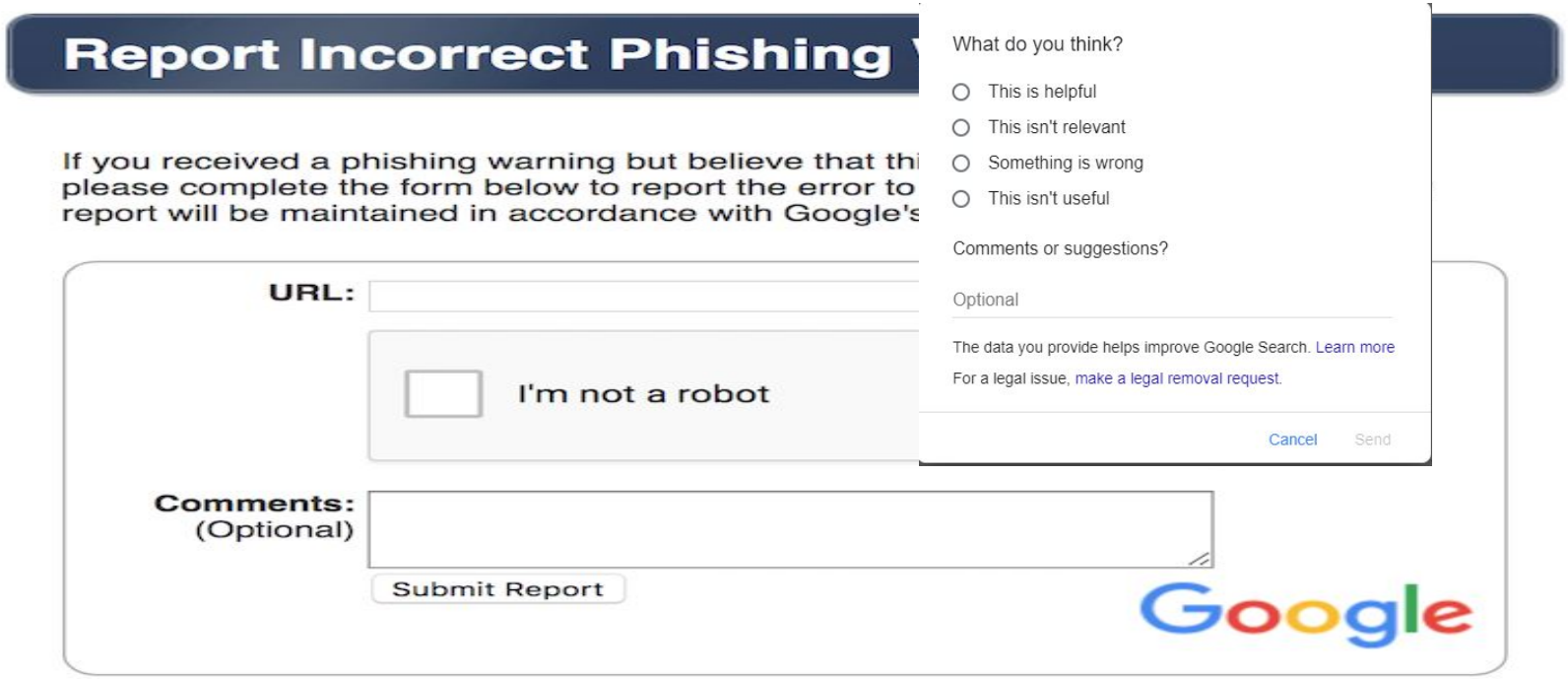
ML makes mistakes



NeuralTalk2: A flock of birds flying in the air
Microsoft Azure: A group of giraffe standing next to a tree
Image: Fred Dunn, https://www.flickr.com/photos/gratapictures - CC-BY-NC

Mitigation strategies?

Collecting feedback



Updating Models

- Models are rarely static outside the lab
- Data drift, feedback loops, new features, new requirements
- We should consider when and how to update models

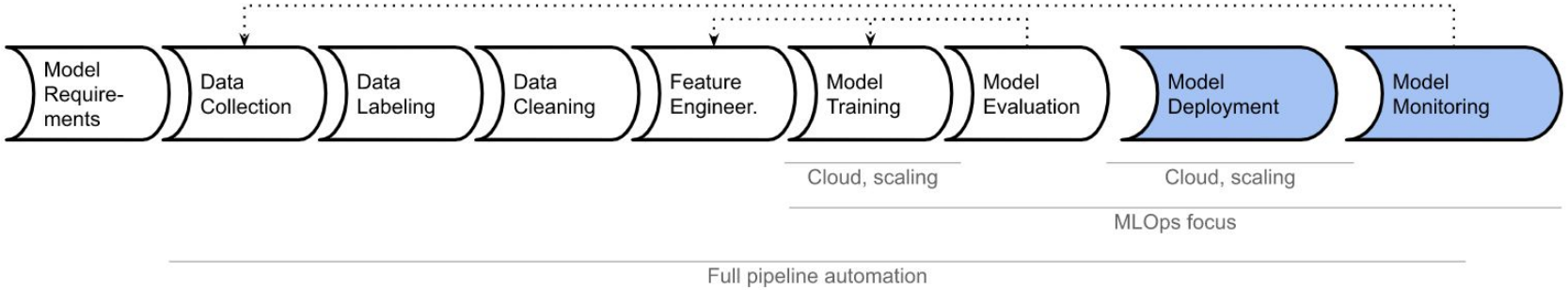
Human in the loop

Design for failures/mistakes

- Human-AI interaction design (human in the loop)
- Guardrails
- Mistakes detection and correction
- Undoable actions

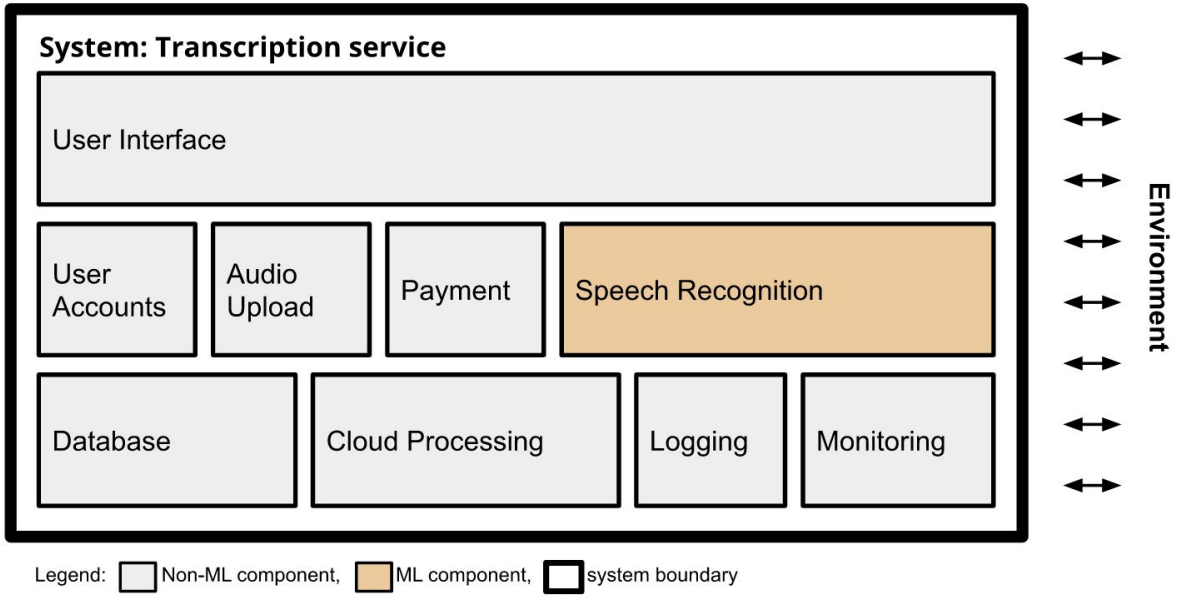
https://ckaestne.medium.com/safety-in-ml-enabled-systems-b5a5901933ac

System-wide pipeline



Focus: experimenting, deploying, scaling training and serving, model monitoring and updating

ML models as part of a system



Outline

- Why ML/AI projects fail?
  - Data quality
  - Fairness issues
- What's wrong with the model-centric pipeline?
- **Are there any new challenges?**
- What is ML Ops?

What (real) challenges are there in building and deploying systems with ML?

What makes software with ML challenging?

- Lack of specification (unreliability)?
- Complexity?
- Big Data?
- Interaction with the environment?

Which of these are truly new challenges, and which are just old software-engineering problems in disguise?

Interaction with the environment: safety

https://www.alphr.com › review › smart-toaster

The Highest-Rated Smart Toasters in 2022 - Alphr Reviews

Aug 19, 2022 — Works on **artificial intelligence** (**AI**). A **smart toaster** operates on **artificial intelligence** to detect and control the whole toast-making process, ...

American Medical Association

AI scribe saves doctors an hour at the keyboard every day

The Permanente Medical Group's rollout of ambient AI scribes to reduce documentation burdens has been deemed a success, saving most of the physicians using it...

Mar 18, 2024

Safety risks?

How can you mitigate these risks?

What makes software with ML challenging?

- Lack of specification (unreliability)
- Complexity
- Big Data
- Interaction with the environment

What makes software (systems) with ML challenging?

It’s not all new… ML intensifies our challenges

System Architecture Tradeoffs

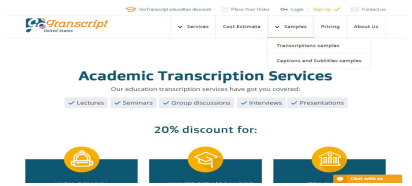
Activity

Pick one scenario based on where you are seating

- Transcription Services (front rows)
- Parking Sensor (middle rows)
- Surge Prediction (back rows)

Discuss in groups these questions:

- Where should the model be deployed? e.g., in the cloud, on-premises, on directly on the devices?
- What are the key factors influencing this choice (e.g., latency, computational power, data privacy)?



Where should the model live?

Laptop

Local Server

Academic Transcriber

Cloud

Where should the model live?

Car

Phone

Cloud

Surge Prediction

Where should the model live?

Pod

Gateway

Cloud

Car Detector

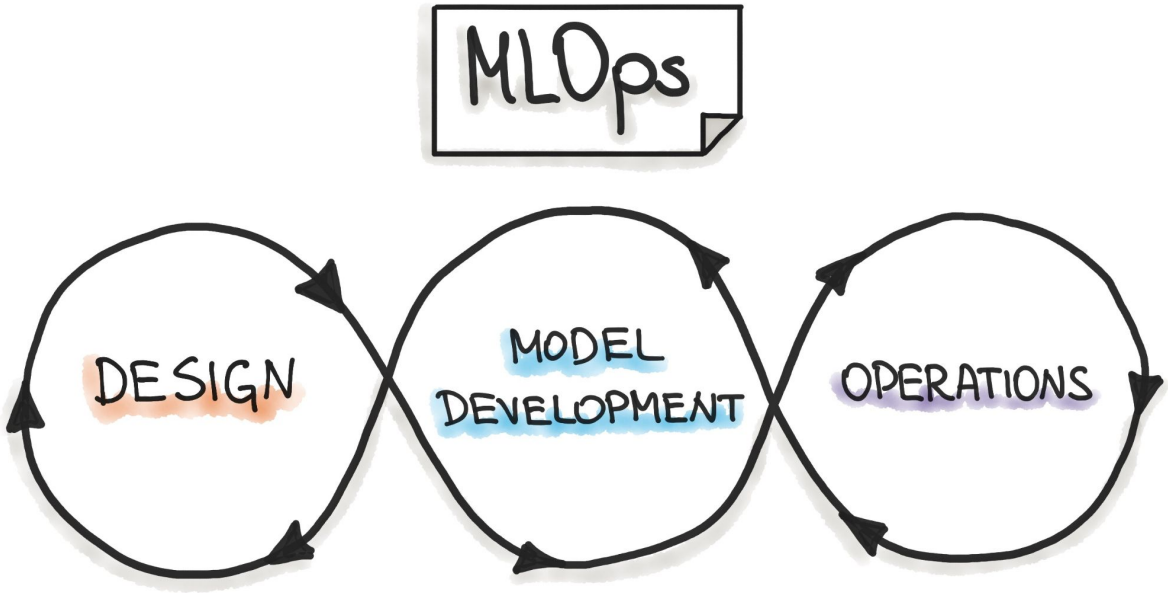
Typical Designs

- Static intelligence in the product
  - difficult to update
  - good execution latency
  - cheap operation
  - offline operation
  - no telemetry to evaluate and improve
- Client-side intelligence
  - updates costly/slow, out of sync problems
  - complexity in clients
  - offline operation, low execution latency

Considerations for deployment

- How much data is needed as input for the model?
- How much output data is produced by the model?
- How fast/energy consuming is model execution?
- What latency is needed for the application?
- How big is the model? How often does it need to be updated?
- Cost of operating the model? (distribution + execution)
- Opportunities for telemetry?
- What happens if users are offline?

MLOps



MLOps

- Many vague buzzwords, often not clearly defined
- MLOps: Collaboration and communication between data scientists and operators, e.g.,
  - Automate model deployment
  - Model training and versioning infrastructure
  - Model deployment and monitoring

MLOps Overview

- Integrate ML artifacts into software release process, unify process (i.e., DevOps extension)
- Automated data and model validation (continuous deployment)
- Continuous deployment for ML models: from experimenting in notebooks to quick feedback in production
- Versioning of models and datasets
- Monitoring in production

MLOps Tools (examples)

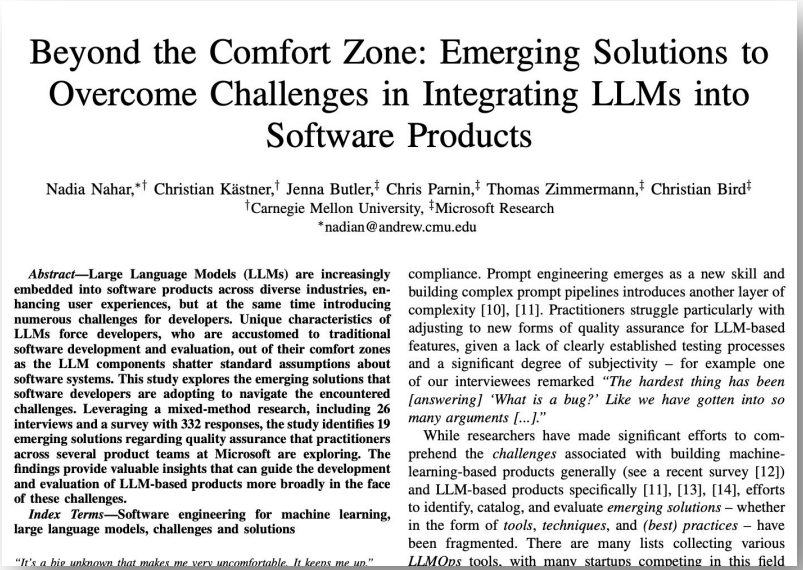
- Model versioning and metadata: MLFlow, Neptune, ModelDB, WandB, …
- Model monitoring: Fiddler, Hydrosphere
- Data pipeline automation and workflows: DVC, Kubeflow, Airflow
- Model packaging and deployment: BentoML, Cortex
- Distributed learning and deployment: Dask, Ray, …
- Feature store: Feast, Tecton
- Integrated platforms: Sagemaker, Valohai, …
- Data validation: Cerberus, Great Expectations, …

Long list: https://github.com/kelvins/awesome-mlops

Process for AI-Enabled Systems

Change of process/ metrics/ mindsets needed…

> “We often run into engineers thinking about these as unit tests. […] It is OK that there is 63 failures. Engineers tend to think about it as ohh […] I need […]. **100% pass rate**



Beyond the Comfort Zone: Emerging Solutions to Overcome Challenges in Integrating LLMs into Software Products

Nadia Nahar,*† Christian Kästner,† Jenna Butler,‡ Chris Parnin,‡ Thomas Zimmermann,‡ Christian Bird‡
†Carnegie Mellon University, ‡Microsoft Research
*nadian@andrew.cmu.edu

***Abstract*—Large Language Models (LLMs) are increasingly embedded into software products across diverse industries, enhancing user experiences, but at the same time introducing numerous challenges for developers. Unique characteristics of LLMs force developers, who are accustomed to traditional software development and evaluation, out of their comfort zones as the LLM components shatter standard assumptions about software systems. This study explores the emerging solutions that software developers are adopting to navigate the encountered challenges. Leveraging a mixed-method research, including 26 interviews and a survey with 332 responses, the study identifies 19 emerging solutions regarding quality assurance that practitioners across several product teams at Microsoft are exploring. The findings provide valuable insights that can guide the development and evaluation of LLM-based products more broadly in the face of these challenges.**

***Index Terms*—Software engineering for machine learning, large language models, challenges and solutions**

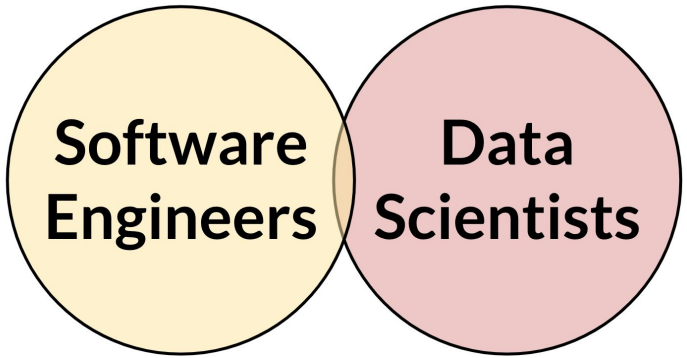
"It's a big unknown that makes me very uncomfortable. It keeps me up."

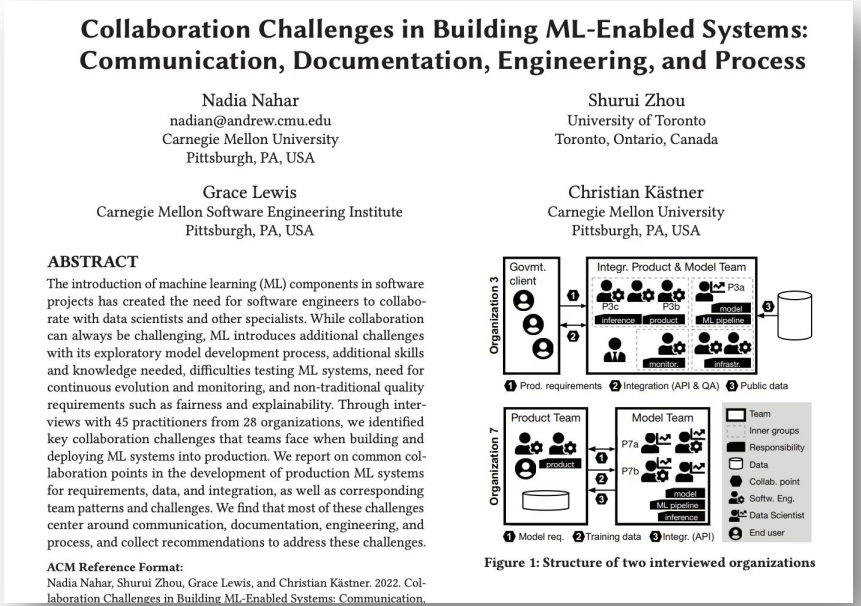
compliance. Prompt engineering emerges as a new skill and building complex prompt pipelines introduces another layer of complexity [10], [11]. Practitioners struggle particularly with adjusting to new forms of quality assurance for LLM-based features, given a lack of clearly established testing processes and a significant degree of subjectivity – for example one of our interviewees remarked *"The hardest thing has been [answering] 'What is a bug?' Like we have gotten into so many arguments […]."*

While researchers have made significant efforts to comprehend the *challenges* associated with building machine-learning-based products generally (see a recent survey [12]) and LLM-based products specifically [11], [13], [14], efforts to identify, catalog, and evaluate *emerging solutions* – whether in the form of *tools*, *techniques*, and *(best) practices* – have been fragmented. There are many lists collecting various *LLMOps* tools, with many startups competing in this field

Nahar, Nadia, et al. "Beyond the Comfort Zone: Emerging Solutions to Overcome Challenges in Integrating LLMs into Software Products." ICSE SEIP 2024.

Change of process/ metrics/ mindsets needed…





Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process

Nadia Nahar
nadian@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Shurui Zhou
University of Toronto
Toronto, Ontario, Canada

Grace Lewis
Carnegie Mellon Software Engineering Institute
Pittsburgh, PA, USA

Christian Kästner
Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

The introduction of machine learning (ML) components in software projects has created the need for software engineers to collaborate with data scientists and other specialists. While collaboration can always be challenging, ML introduces additional challenges with its exploratory model development process, additional skills and knowledge needed, difficulties testing ML systems, need for continuous evolution and monitoring, and non-traditional quality requirements such as fairness and explainability. Through interviews with 45 practitioners from 28 organizations, we identified key collaboration challenges that teams face when building and deploying ML systems into production. We report on common collaboration points in the development of production ML systems for requirements, data, and integration, as well as corresponding team patterns and challenges. We find that most of these challenges center around communication, documentation, engineering, and process, and collect recommendations to address these challenges.

ACM Reference Format:
Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. 2022. Collaboration Challenges in Building ML-Enabled Systems: Communication,

Figure 1: Structure of two interviewed organizations

Nahar, Nadia, et al. "Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process." *Proceedings of the 44th international conference on software engineering*. 2022.

Data Science is Iterative and Exploratory

- Science mindset: start with rough goal, no clear specification, unclear whether possible
- Heuristics and experience to guide the process
- Try and error, refine iteratively, hypothesis testing
- Go back to data collection and cleaning if needed, revise goals

Trajectories

- Not every project follows the same development process, e.g.
  - Small ML addition: Product first, add ML feature later
  - Research only: Explore feasibility before thinking about a product
  - AI first: Model as central component of potential product, build system around it
- Different focus on system requirements, qualities, and upfront planning
- Manage interdisciplinary teams and different expectations

Computational Notebooks

- Origins in "*literate programming*", interleaving text and code, treating programs as literature (Knuth 84)
- Document with text and code cells, showing execution results under cells
- Code of cells is executed, per cell, in a kernel
- Many notebook implementations and supported languages, Python + Jupyter currently most popular

```
# load data collected from team1
import pandas as pd


url = 'http://128.2.25.78:8080/private/log1.clean'
df = pd.read_csv(url)
df.head()
```

	dayIdx	user	userAvgTime	location	dow	isWeekend	time
0	0	Pittsburgh66Correy	7.045001	Pittsburgh	6	True	0.000000
1	1	Pittsburgh66Correy	7.045001	Pittsburgh	7	True	6.883333
2	2	Pittsburgh66Correy	7.045001	Pittsburgh	1	False	6.816667
3	3	Pittsburgh66Correy	7.045001	Pittsburgh	2	False	7.383333
4	4	Pittsburgh66Correy	7.045001	Pittsburgh	3	False	0.000000

Data was preprocessed externally, identifying the time at a given day when the light was first turned on
12pm). Weather and sunrise information is not included here, though that'd be important. If the light w
this morning (quite common), 0 is recorded.

```
[ ] # just data encoding and splitting X and Y

    X = df.drop(['time'], axis=1)
    YnonZero = df['time'] > 0
    Y = df['time']

    from sklearn import preprocessing
    # leDate = preprocessing.LabelEncoder()
    # leDate.fit(X['date'])
    # leDate.transform(X['date'])

    X=X.apply(preprocessing.LabelEncoder().fit_transform)
    X
```

Notebooks Support Iteration and Exploration

- Quick feedback, similar to REPL
- Visual feedback including figures and tables
- Incremental computation: running individual cells
- Quick and easy: copy paste, no abstraction needed
- Easy to share: document includes text, code, and results

Brief Discussion: Notebook Limitations and Drawbacks?

Summary

- Production AI-enabled systems require a *whole system perspective* beyond just the model or the pipeline
- Machine learning brings new challenges and intensifies old ones
- Building ML systems need team efforts
  - Collaborative culture among Software Engineers, Data Scientists, Stakeholders is necessary