From Model to System

Jin Guo
SOCS McGill University

What should we test for ML models (other than Model Performance)

- Algorithmic Correctness (Design and Implementation)

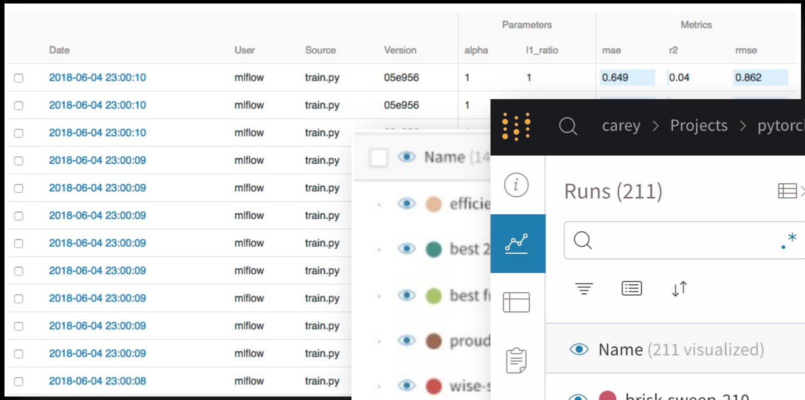
  - Verify if the loss decreases after training for a few iterations
  - Overfit small dataset
  - Test specific sub-computations

What should we test for ML models (other than Model Performance)

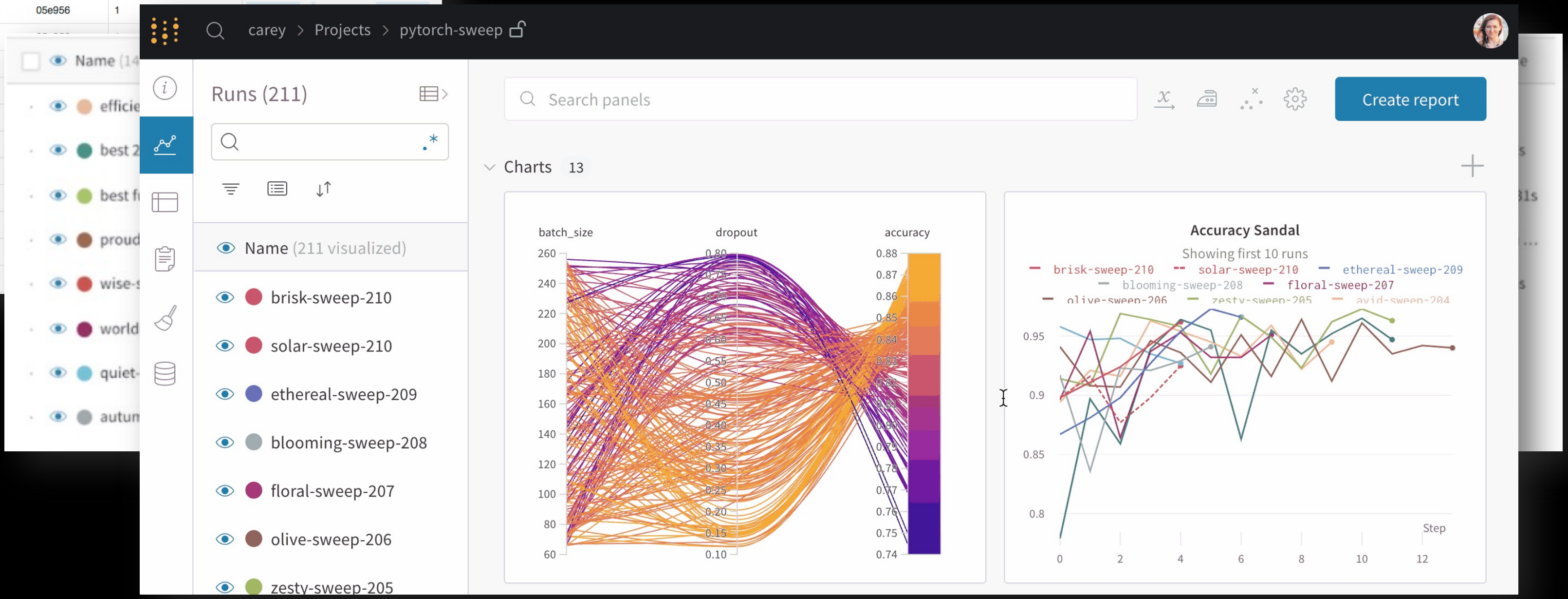
- Algorithmic Correctness (Design and Implementation)
- Reproducible Training
    - Deterministically seed the random number generator
    - Initialize model components in a fixed order
    - Average several runs of the model.
    - Use version control (ML experiment management tools)

mlflow

W&B



				Parameters		Metrics		
Date	User	Source	Version	alpha	l1_ratio	mae	r2	rmse
2018-06-04 23:00:10	mlflow	train.py	05e956	1	1	0.649	0.04	0.862
2018-06-04 23:00:10	mlflow	train.py	05e956	1				
2018-06-04 23:00:10	mlflow	train.py						
2018-06-04 23:00:09	mlflow	train.py						
2018-06-04 23:00:09	mlflow	train.py						
2018-06-04 23:00:09	mlflow	train.py						
2018-06-04 23:00:09	mlflow	train.py						
2018-06-04 23:00:09	mlflow	train.py						
2018-06-04 23:00:09	mlflow	train.py						
2018-06-04 23:00:09	mlflow	train.py						
2018-06-04 23:00:09	mlflow	train.py						
2018-06-04 23:00:08	mlflow	train.py						



Runs (211)

Name (211 visualized)

- brisk-sweep-210
- solar-sweep-210
- ethereal-sweep-209
- blooming-sweep-208
- floral-sweep-207
- olive-sweep-206
- zesty-sweep-205

Images from https://mlflow.org/docs/, https://docs.wandb.ai/

What should we test for ML models (other than Model Performance)

- Algorithmic Correctness (Design and Implementation)
- Reproducible Training
- Model Quality Degradation
    - Sudden or Slow degradation

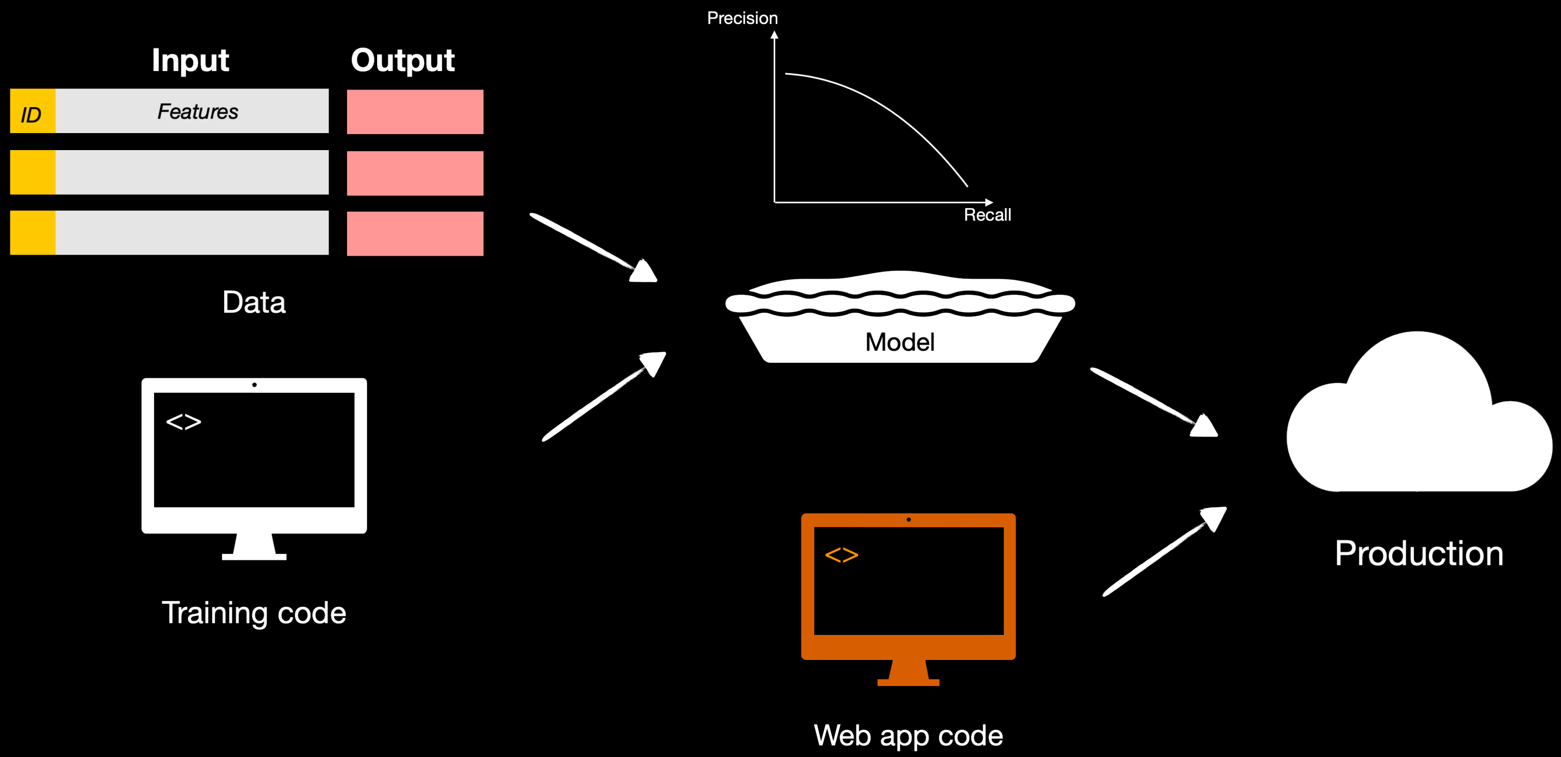
What should we test for ML models (other than Model Performance)

- Algorithmic Correctness (Design and Implementation)

- Reproducible Training

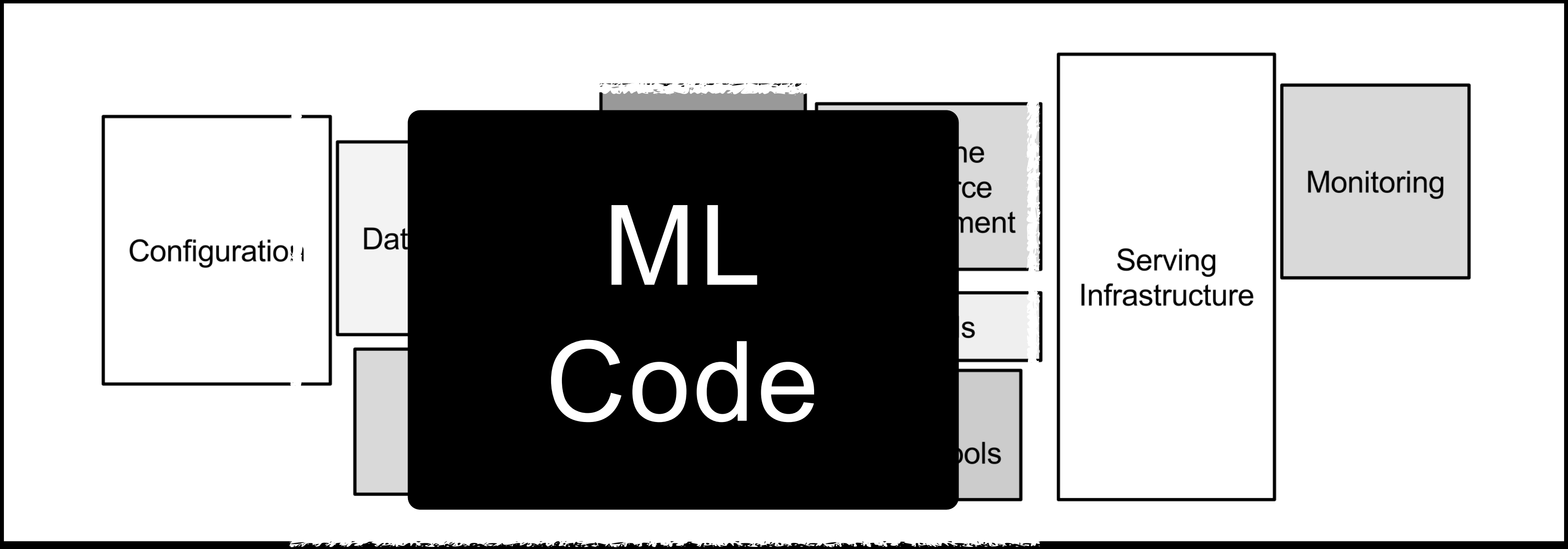
- Model Quality Degradation

Static Model vs Dynamic Model?



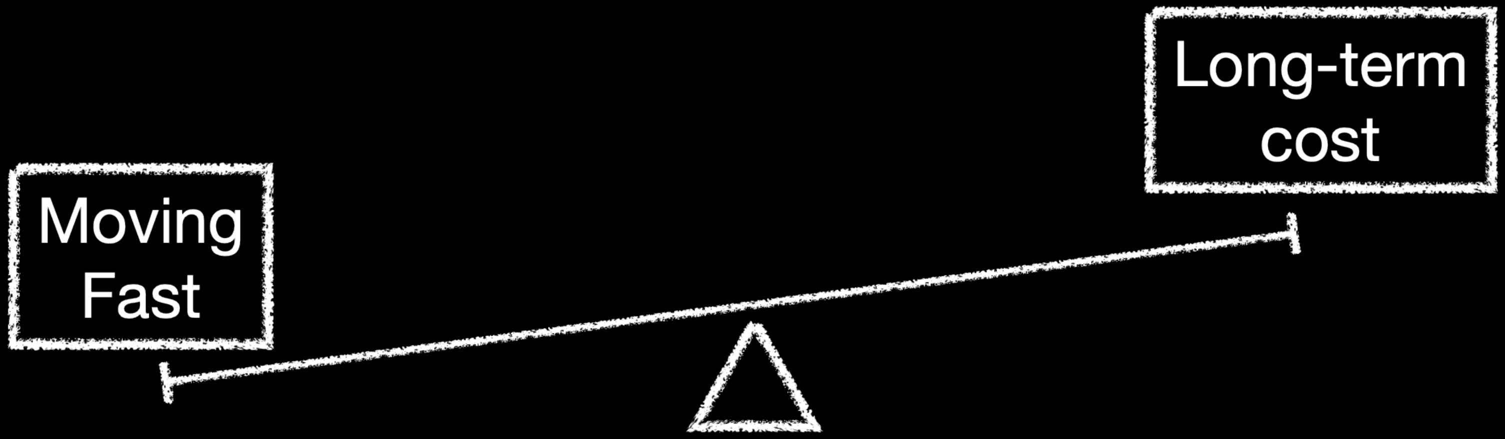
What's missing?

The 2-5%



Sculley, David, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. "Hidden technical debt in machine learning systems." In Advances in neural information processing systems, pp. 2503-2511. 2015.

Technical Debt

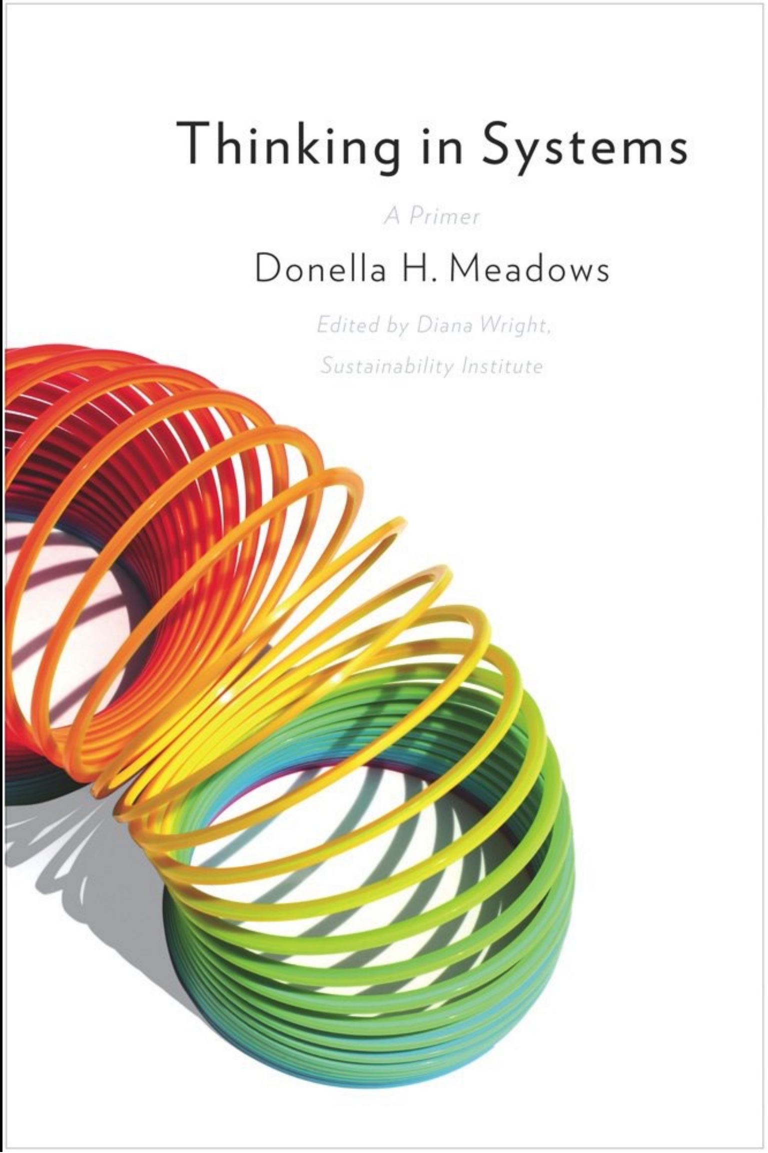


"not quite right code which we postpone making it right"

Ward Cunningham, "The WyCash Portfolio Management System,"
Proc. OOPSLA, ACM, 1992; http://c2.com/doc/oopsla92.html.

"A system is an interconnected set of elements that is coherently organized in a way that achieves something."

- Meadows


Thinking in Systems

A Primer

Donella H. Meadows

Edited by Diana Wright, Sustainability Institute

Why systems surprises us

- Beguiling events

A system is a big black box
Of which we can't unlock the locks,
And all we can find out about
Is what goes in and what comes out.
Perceiving input-output pairs,
Related by parameters,
Permits us, sometimes, to relate
An input, output and a state.
If this relation's good and stable
Then to predict we may be able,
But if this fails us—heaven forbid!
We'll be compelled to force the lid!

—Kenneth Boulding

Why systems surprises us

- Beguiling events
  - System structure is the source of system behavior. System behavior reveals itself as a series of events over time.
  - We are insufficiently skilled at seeing in their history clues to the structures from which behavior and events flow.

Why systems surprises us

- Linear Minds in a Nonlinear World
  - Linear systems can be taken apart and put them together again— the pieces add up.
  - A nonlinear relationship is one in which the cause does not produce a proportional effect.

Why systems surprises us

- Nonexistent Boundaries
  - Boundaries are of our own making, and that they can and should be reconsidered for each new discussion, problem, or purpose.

Why systems surprises us

- Layers of Limits

  - There are layers of limits around every growing plant, child, epidemic, new product, technological advance, company, city, economy, and population.

  - There always will be limits to growth. They can be self-imposed. If they aren’t, they will be system imposed.

Why systems surprises us

- Ubiquitous Delay
  - Most flows in systems have delays—shipping delays, perception delays, processing delays, maturation delays.
  - When there are long delays in feedback loops, some sort of foresight is essential.

Why systems surprises us

- Bounded Rationality (Herbert A. Simon)
  - People make quite reasonable decisions based on the information they have. But they don’t have perfect information, especially about more distant parts of the systems.
  - From a winder perspective, information flows, goals, incentives, and disincentives can be restructured so that separate, bounded, rational actions do add up to results that everyone desires — the right feedbacks gets to the right place at the right time.

Activity

- Choose at least two types of the hidden technical debts from the assigned reading for the group.
- Each member pick two concrete debts, read them and explain them to your group.
- Identify if those technical debts are due to any of the following reasons:
  - Beguiling Events
  - Linear Minds in a Nonlinear World
  - Nonexistent Boundaries
  - Layers of Limits
  - Ubiquitous Delays
  - Bounded Rationality

A process view

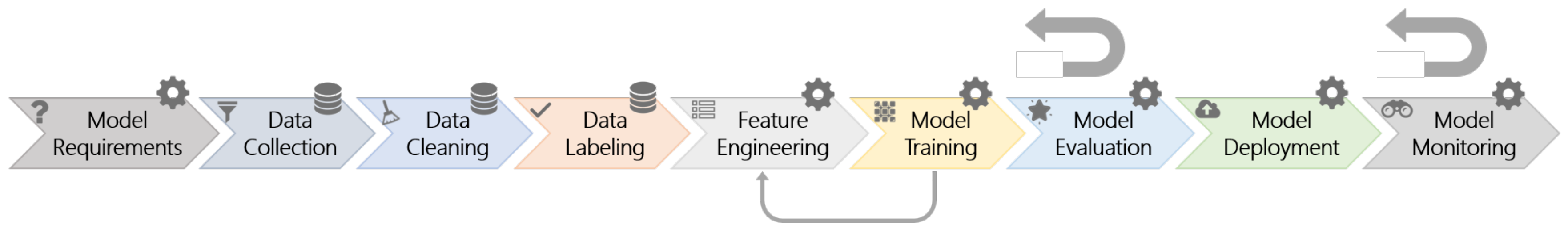


Fig. 1. The nine stages of the machine learning workflow. Some stages are data-oriented (e.g., collection, cleaning, and labeling) and others are model-oriented (e.g., model requirements, feature engineering, training, evaluation, deployment, and monitoring). There are many feedback loops in the workflow. The larger feedback arrows denote that model evaluation and monitoring may loop back to any of the previous stages. The smaller feedback arrow illustrates that model training may loop back to feature engineering (e.g., in representation learning).

Amershi, Saleema, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. "Software engineering for machine learning: A case study." In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 291-300. IEEE, 2019.

Challenges

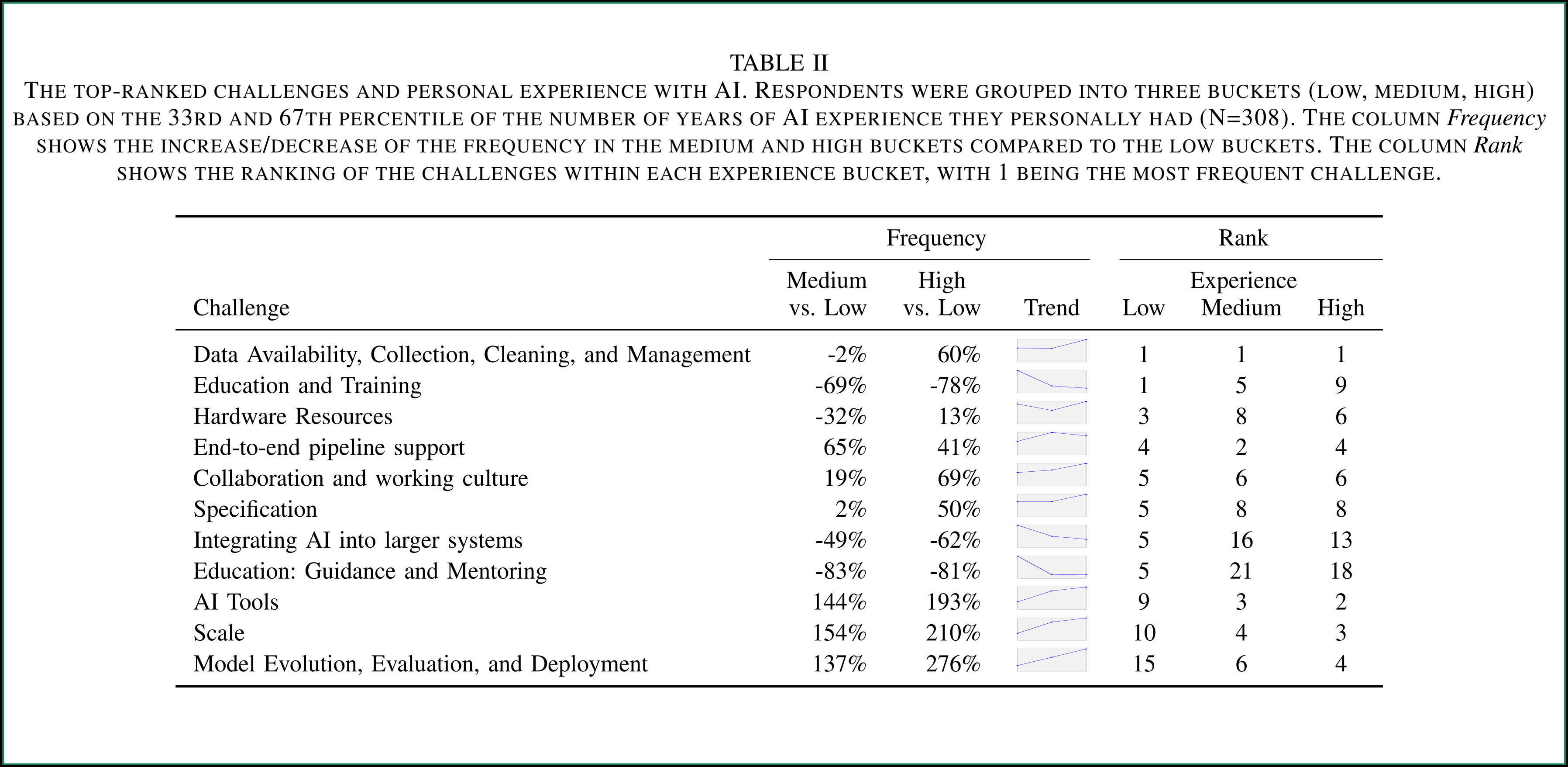


TABLE II
The top-ranked challenges and personal experience with AI. Respondents were grouped into three buckets (low, medium, high) based on the 33rd and 67th percentile of the number of years of AI experience they personally had (N=308). The column *Frequency* shows the increase/decrease of the frequency in the medium and high buckets compared to the low buckets. The column *Rank* shows the ranking of the challenges within each experience bucket, with 1 being the most frequent challenge.

	Frequency			Rank		
	Medium	High			Experience	
Challenge	vs. Low	vs. Low	Trend	Low	Medium	High
Data Availability, Collection, Cleaning, and Management	-2%	60%		1	1	1
Education and Training	-69%	-78%		1	5	9
Hardware Resources	-32%	13%		3	8	6
End-to-end pipeline support	65%	41%		4	2	4
Collaboration and working culture	19%	69%		5	6	6
Specification	2%	50%		5	8	8
Integrating AI into larger systems	-49%	-62%		5	16	13
Education: Guidance and Mentoring	-83%	-81%		5	21	18
AI Tools	144%	193%		9	3	2
Scale	154%	210%		10	4	3
Model Evolution, Evaluation, and Deployment	137%	276%		15	6	4

On Next Tuesday:

Data Acquisition and Management