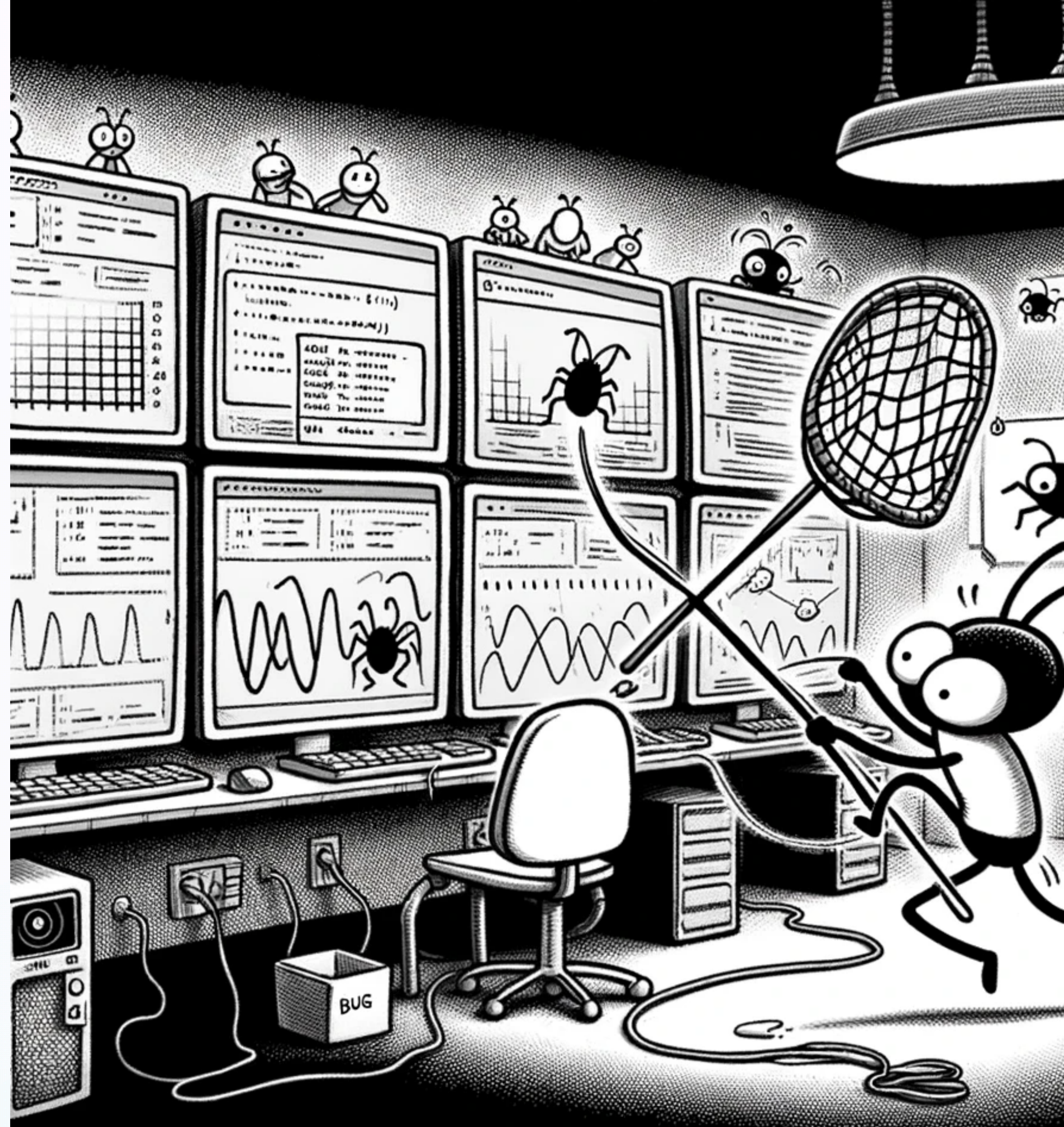




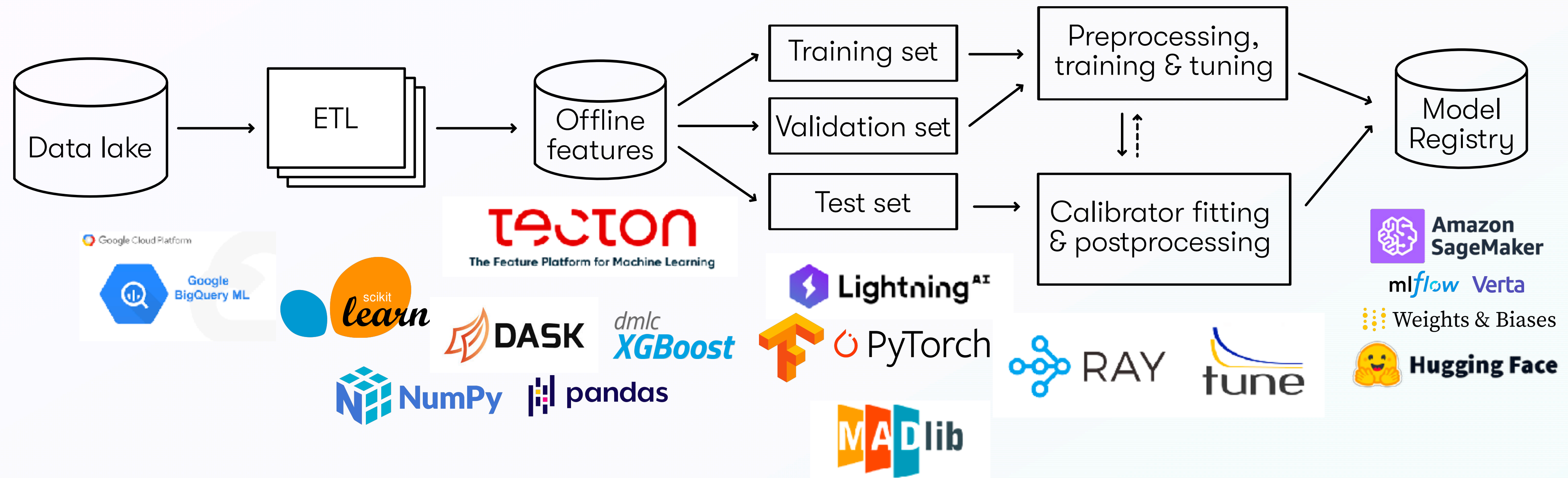
Towards Better Tooling for AI and ML Engineers

Shreya Shankar
November 2023

EPIC
DATA lab
UC Berkeley



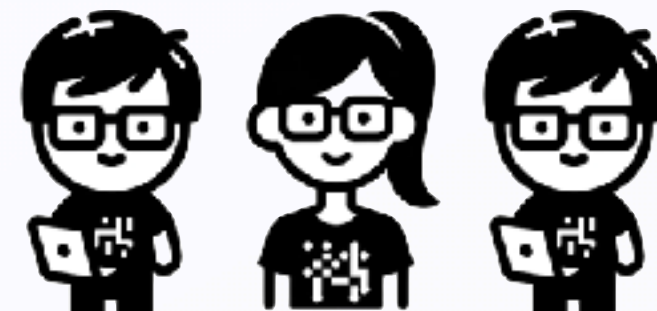
ML Pipelines are Increasing in Complexity



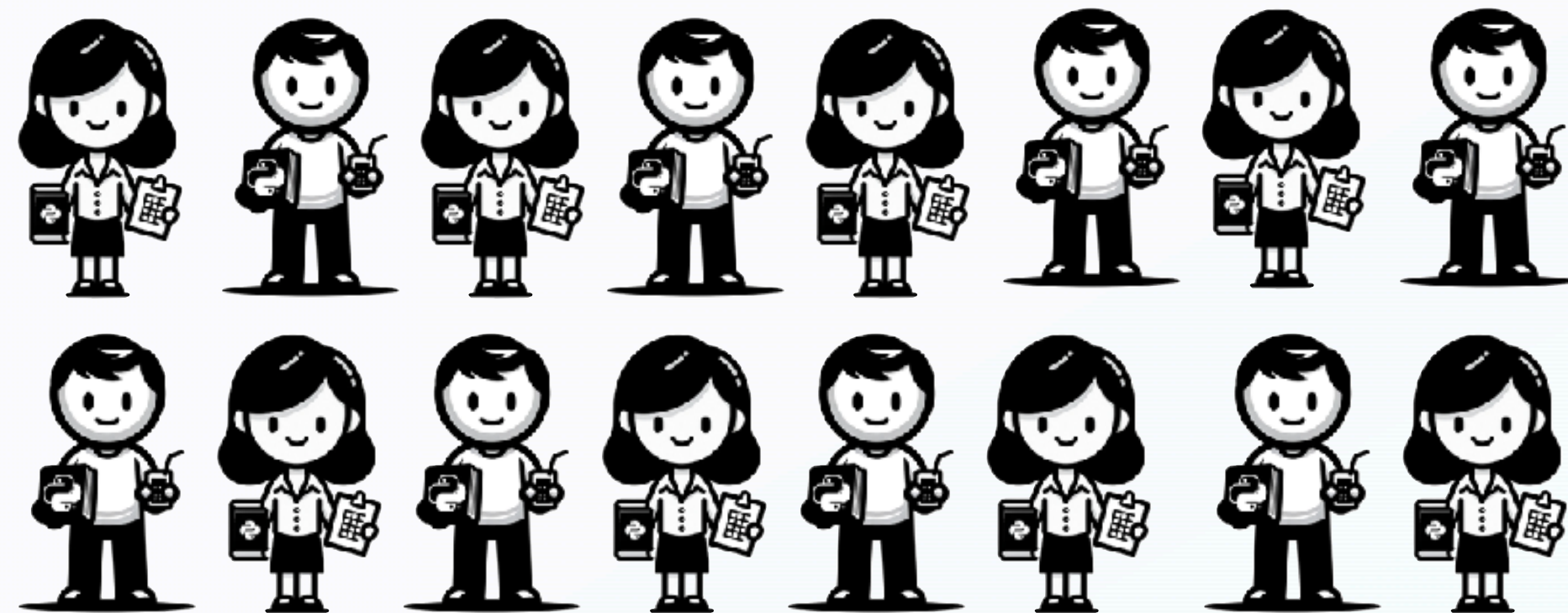
This is limited to tools from a few years ago, but there are now 100s of tools for every ML pipeline component

Zooming in on the ML Engineer Persona

ML engineers must know data engineering, analysis, statistics, ML, software engineering, and more. Even then, their job is difficult!



ML engineers

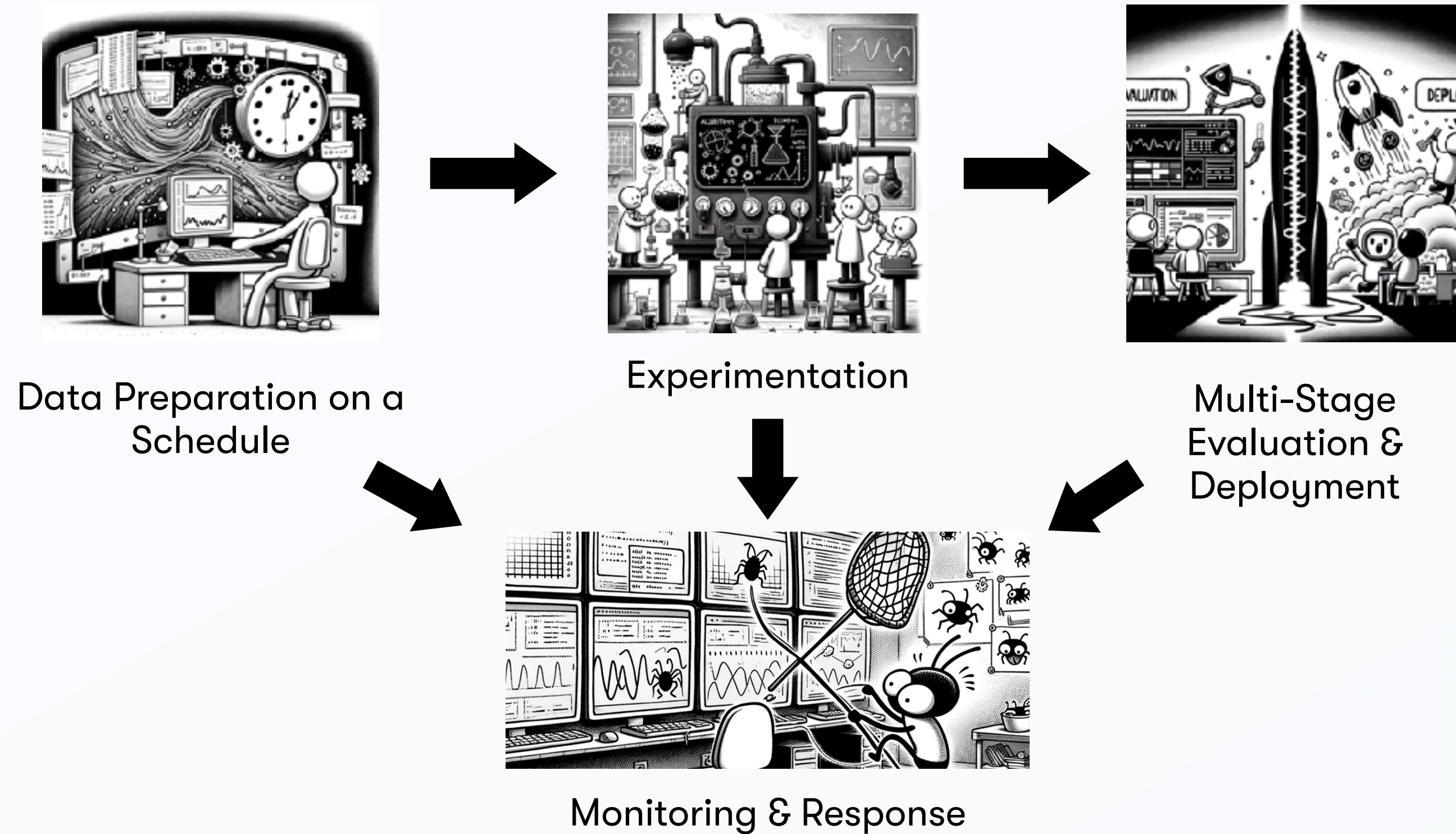


people who know Python

Can we make life *easier* for  & ML engineering more *accessible* for  ?

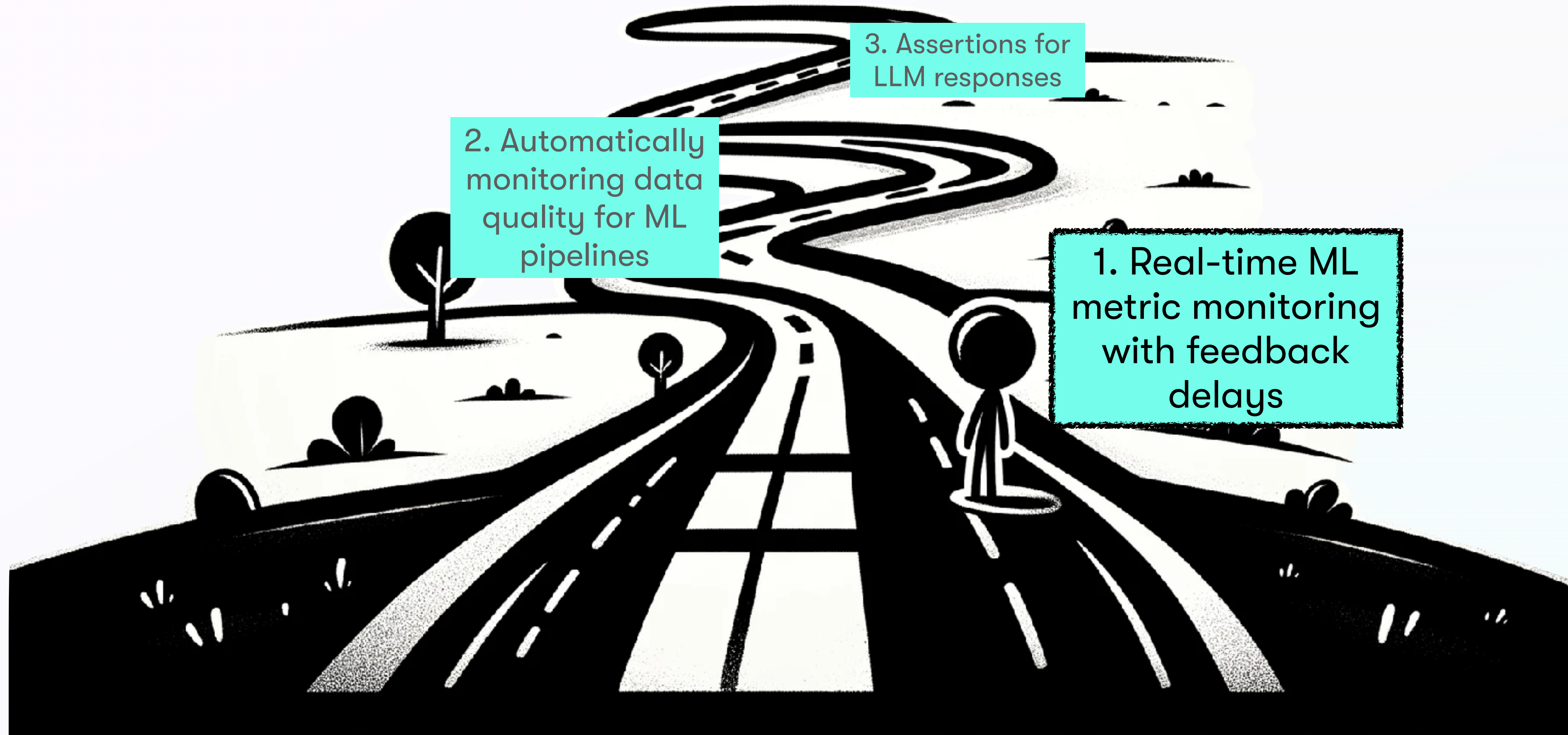
Understanding the Human-Centered MLOps Workflow

Operationalizing Machine Learning: An Interview Study (2022)



- We thought the ML lifecycle is fully amenable to automation, but it's not
- Manual work includes:
 - Handling feedback or ground-truth delays
 - Monitoring pipeline inputs & outputs
 - And many more tasks!

Today's Roadmap

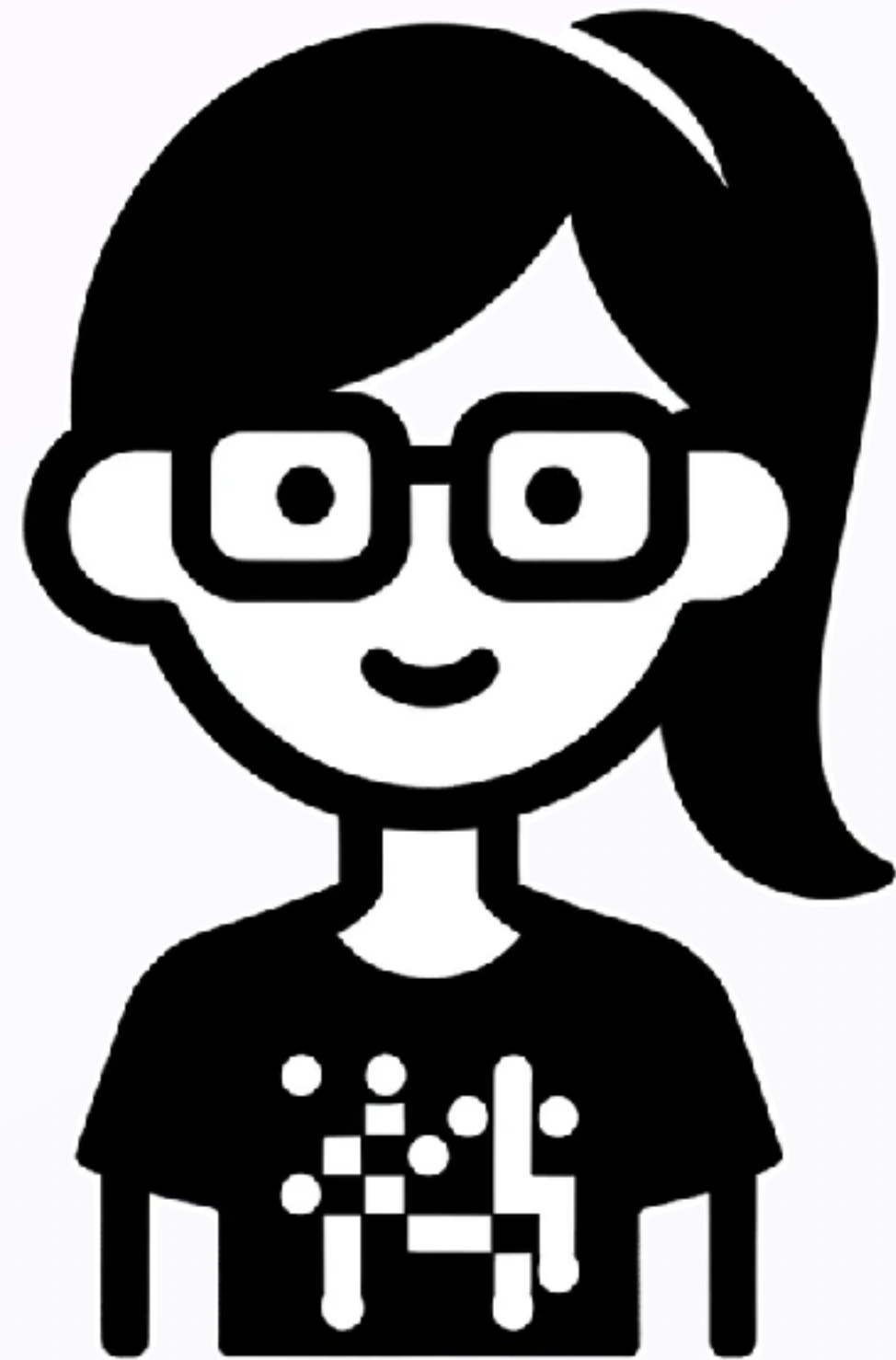


2. Automatically monitoring data quality for ML pipelines

3. Assertions for LLM responses

1. Real-time ML metric monitoring with feedback delays

Handling Feedback Delays

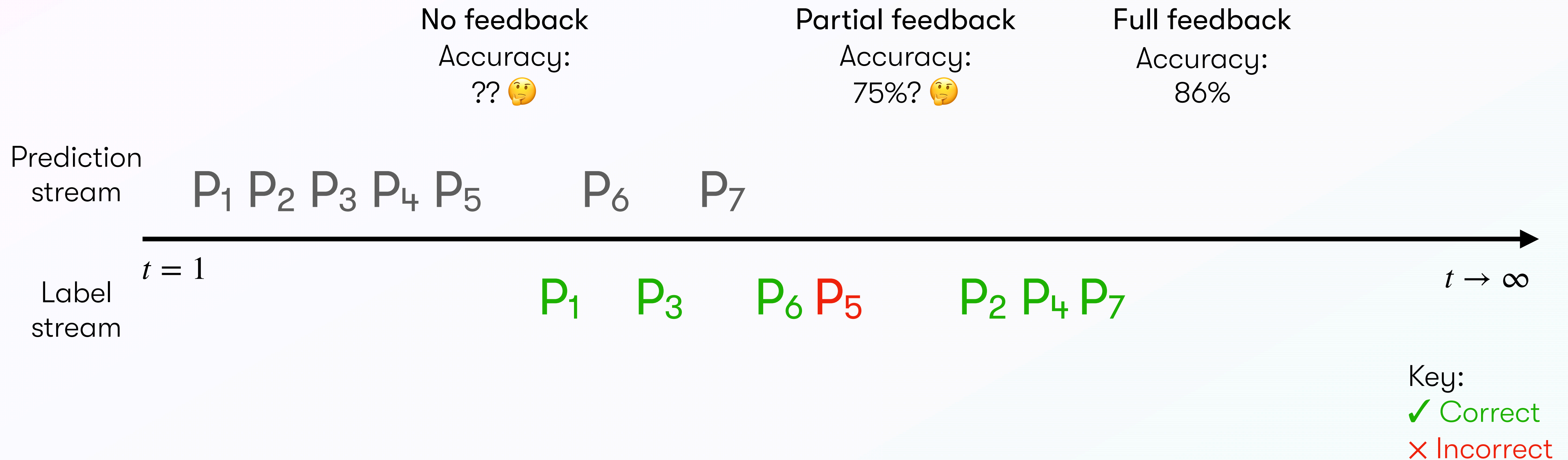


“I have no idea how well [models] actually perform on live data. Feedback is always delayed by at least 2 weeks.”

- In the absence of labels, to learn of prediction errors, engineers often do ad-hoc manual inspection of data or rely on loosely-correlated product metrics
- We started building MLTRACE, a bolt-on system for observability in end-to-end pipelines. We integrated provenance and runtime assertion testing & monitoring capabilities
- Can we try to estimate metrics that require labels (e.g., accuracy)?

Impact of Feedback Delays

Towards Observability for Production ML Pipelines (2023)



How to get the accuracy of Ps that don't have labels yet? 🤔

Estimating Performance with Feedback Delays

💡 Estimate unknown performance with known, labeled data

id	feature 1	feature 2	label
A	TRUE
B	FALSE
C	FALSE
D	TRUE
E	TRUE
F	TRUE
G	FALSE
H	TRUE
...

Estimating Performance with Feedback Delays

💡 Estimate unknown performance with known, labeled data

id	feature 1	feature 2	label
A	TRUE
B	FALSE
C	FALSE
D	TRUE
E	TRUE
F	TRUE
G	FALSE
H	TRUE
...

Training set

Validation set

🪣 “Bucket” validation set to characterize data distribution

🧮 Compute per-bucket accuracies

🕵️ Use buckets to maintain histogram of unlabeled data

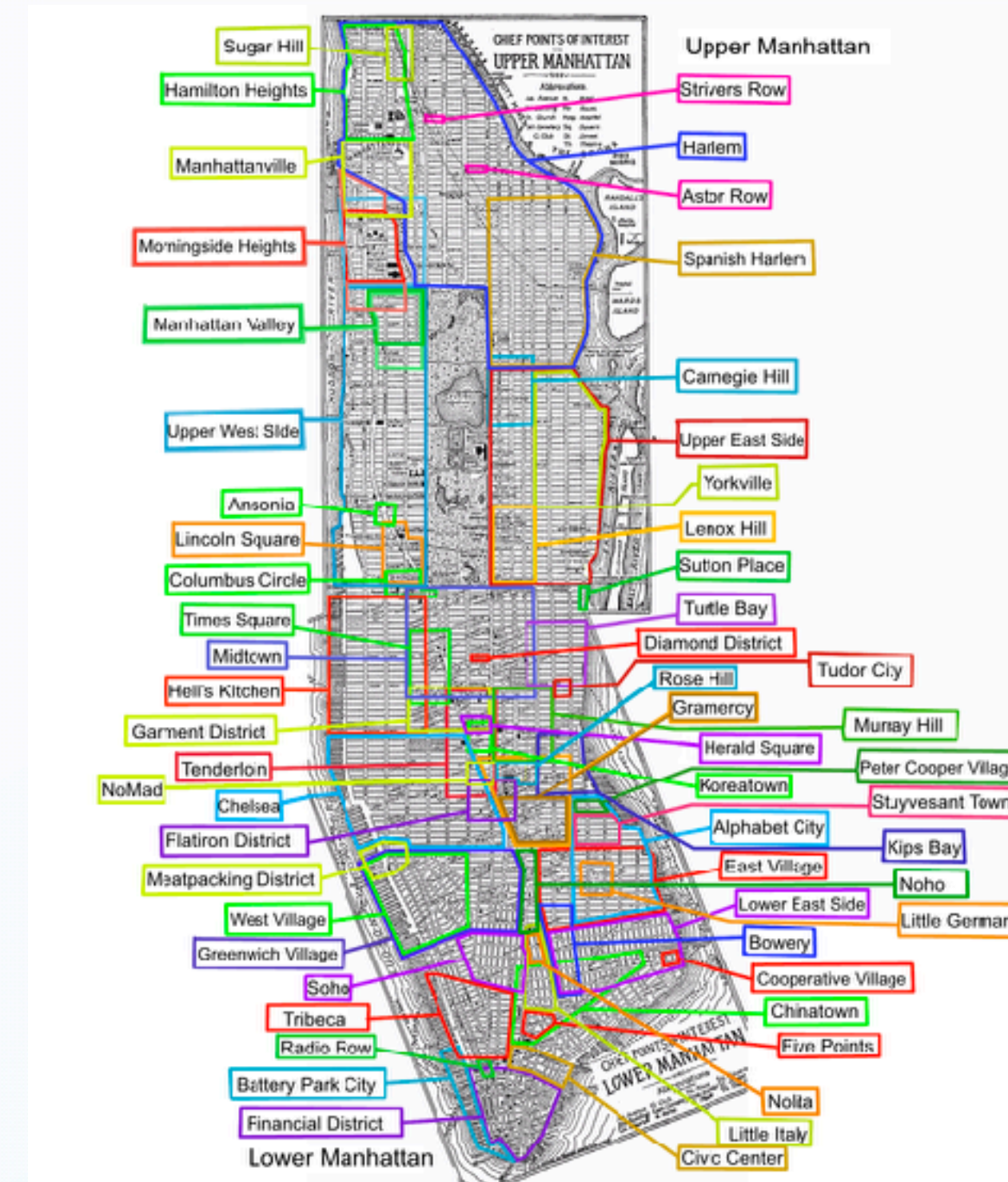
Estimating Performance with Feedback Delays

Illustrative example with NYC Taxicab Dataset to predict tip > 10%

Training set

id	# passengers	pickup loc	dropoff loc	...	label
A	TRUE
B	FALSE
C	FALSE
D	TRUE
E	TRUE
F	TRUE
G	FALSE
H	TRUE
...

Validation set



 on the neighborhoods?

Wikipedia "List of Manhattan Neighborhoods"

Estimating Performance with Feedback Delays

Illustrative example with NYC Taxicab Dataset to predict tip > 10%

Training set

id	# passengers	pickup loc	dropoff loc	...	label
A	TRUE
B	FALSE
C	FALSE
D	TRUE
E	TRUE
F	TRUE
G	FALSE
H	TRUE
...

 on the neighborhoods?

- FiDi Chelsea
- FiDi Chinatown
- Midtown Upper West Side
- East Village
- Harlem
- East Village SoHo
- Chelsea Upper West Side
- Upper East Side
- Tribeca SoHo


Validation set

Estimating Performance with Feedback Delays

Illustrative example with NYC Taxicab Dataset to predict tip > 10%

Training set

id	# passengers	pickup loc	dropoff loc	...	label
A	TRUE
B	FALSE
C	FALSE
D	TRUE
E	TRUE
F	TRUE
G	FALSE
H	TRUE
...

 Chelsea = 50%, SoHo = 80%

If we have 100 unlabeled SoHo rides & 200 Chelsea rides....

$$\approx 0.8 \times 100 + 0.5 \times 200 = \frac{80 + 100}{300} = 60\%$$



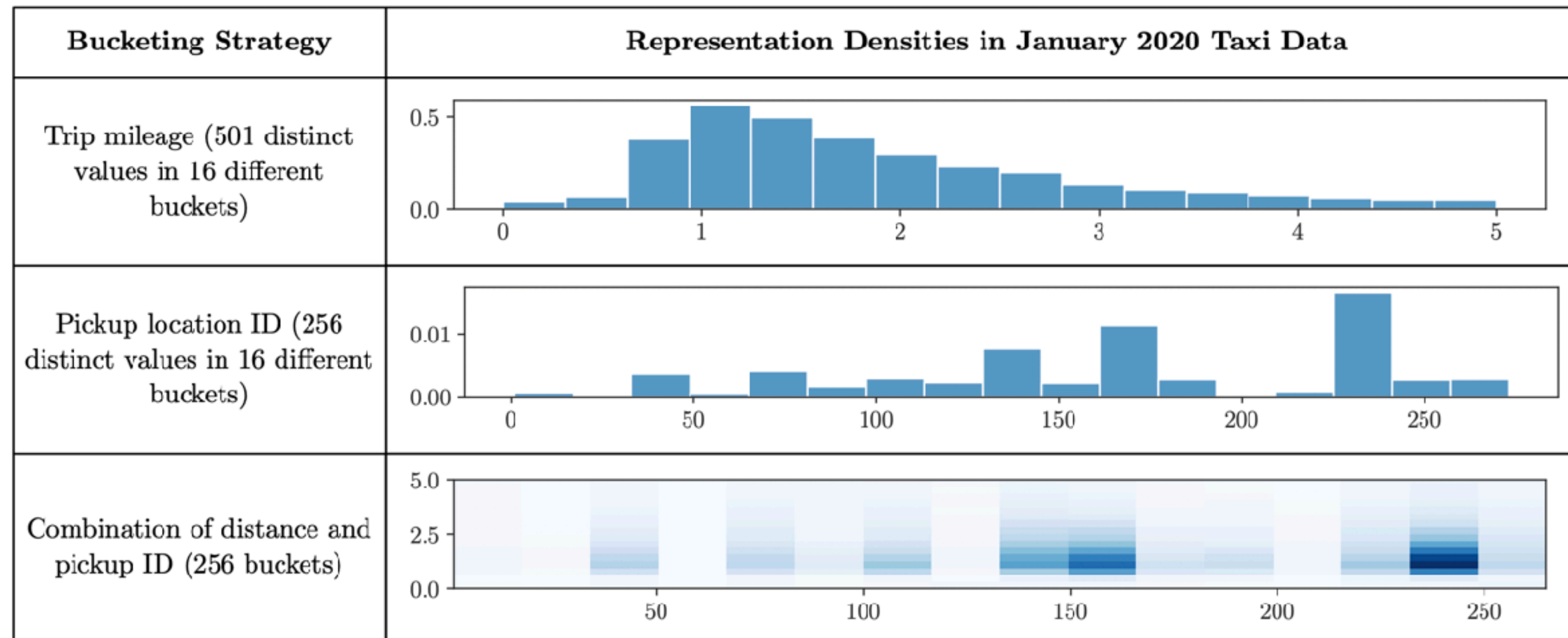
Validation set

Estimating Performance with Feedback Delays

- 😊 Small space
- 😞 Not predictive

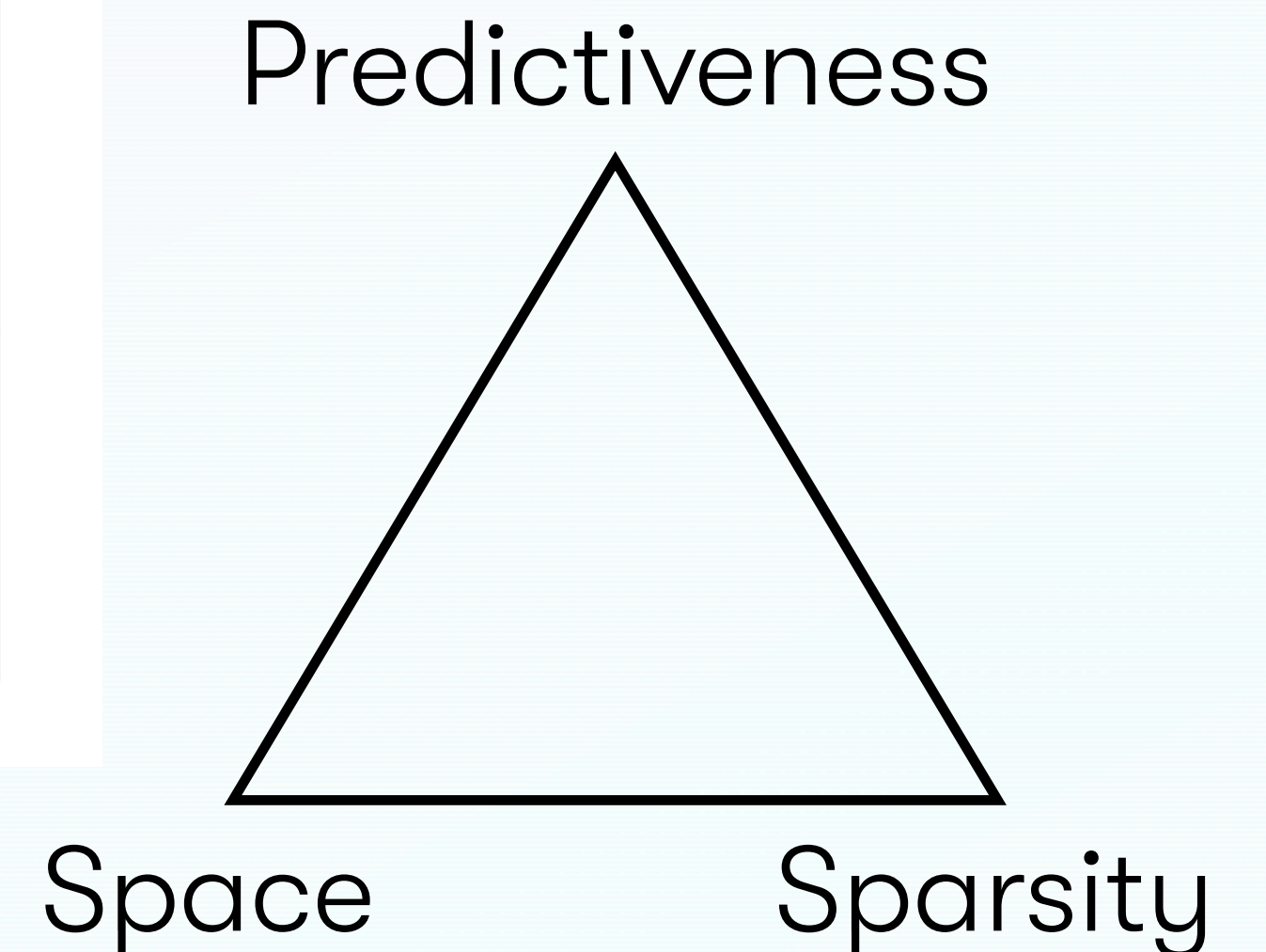
Bucketing strategy matters a *lot!*

Increasing
granularity
and
sparsity



- 😞 Large
- 😊 More predictive?

Empty buckets have no accuracy estimates



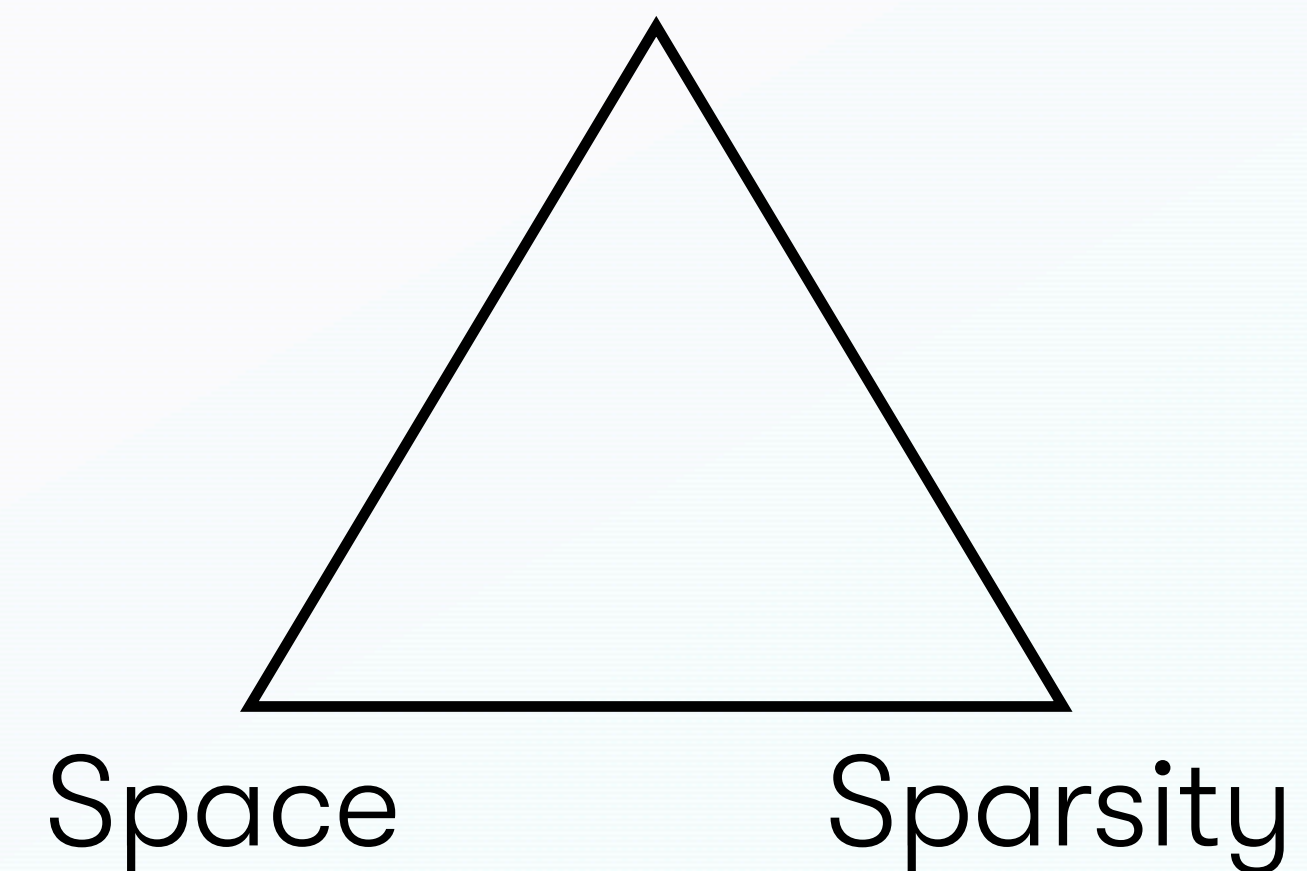
But Data Drifts!



Importance-weighted estimated vs real accuracy on a weekly basis.

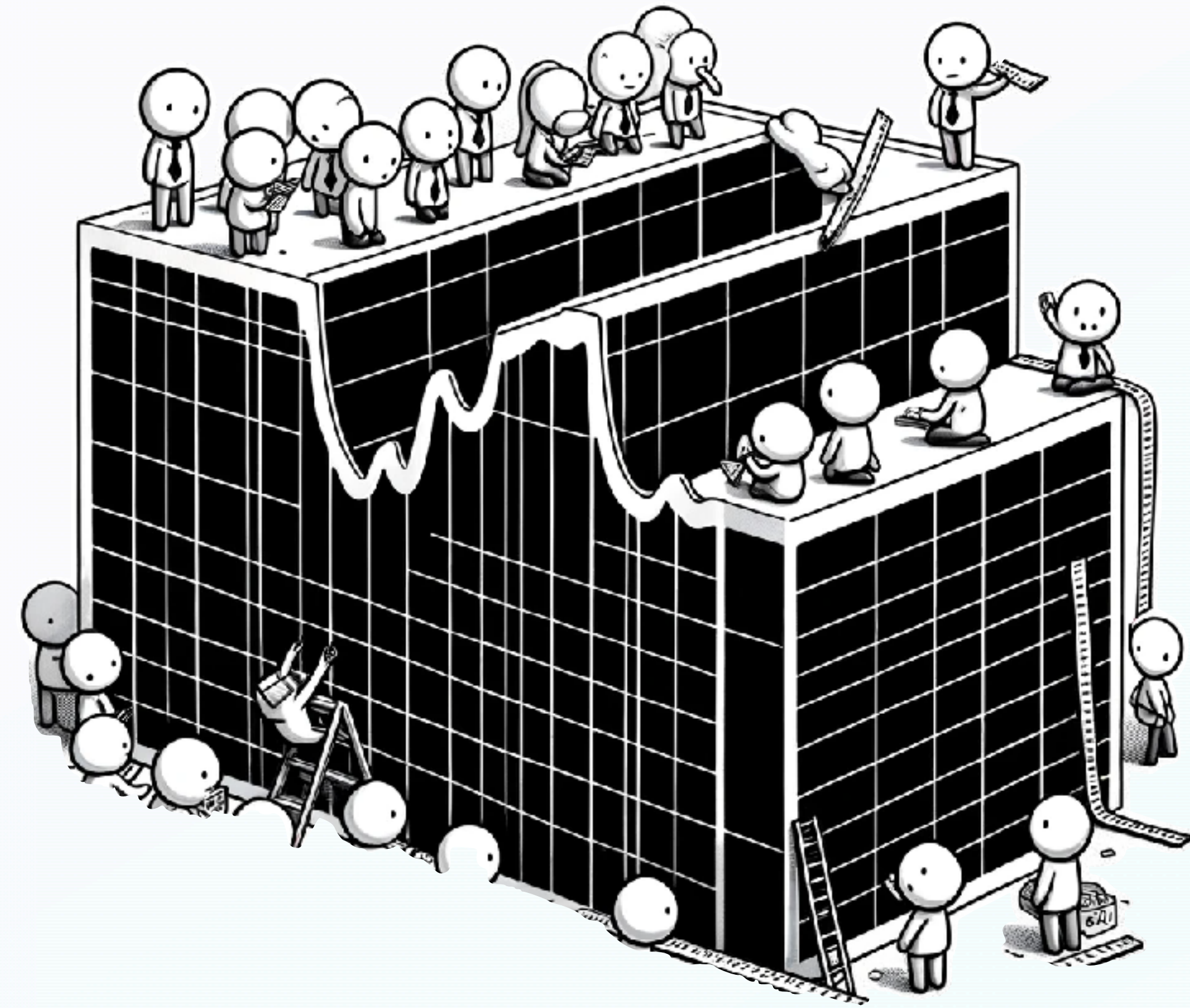
Bucketing strategy should change when data drifts!

Predictiveness over time



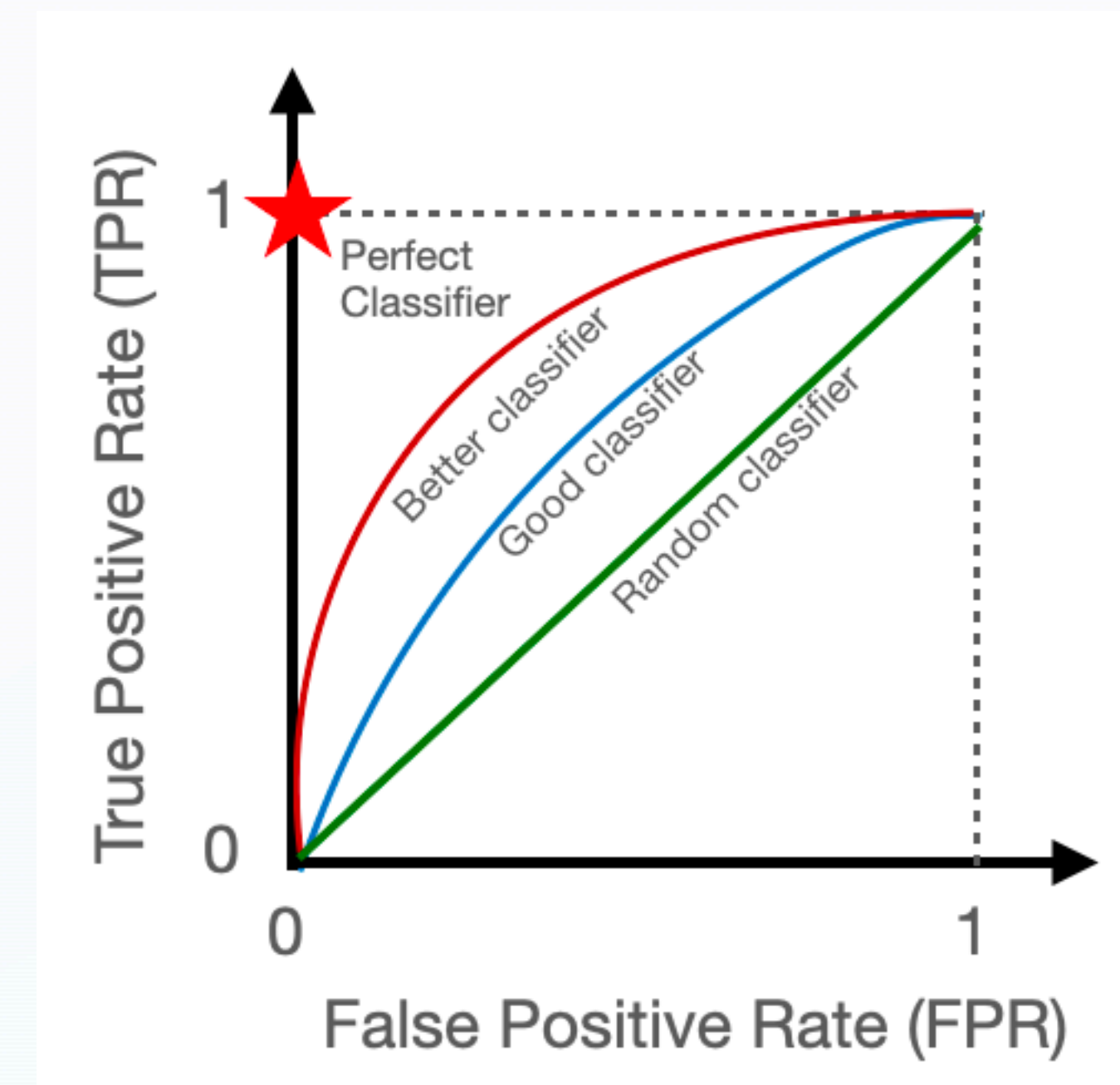
But Data Drifts!

- Importance weighting tricks don't work when serving/live data D' has *drifted* from validation set D enough to result in accuracy drop
- Prior work: track divergence metrics between D and D'
 - **×** Large memory footprint
 - **×** Inaccurate at scale



Drift Detection in Pipeline Components

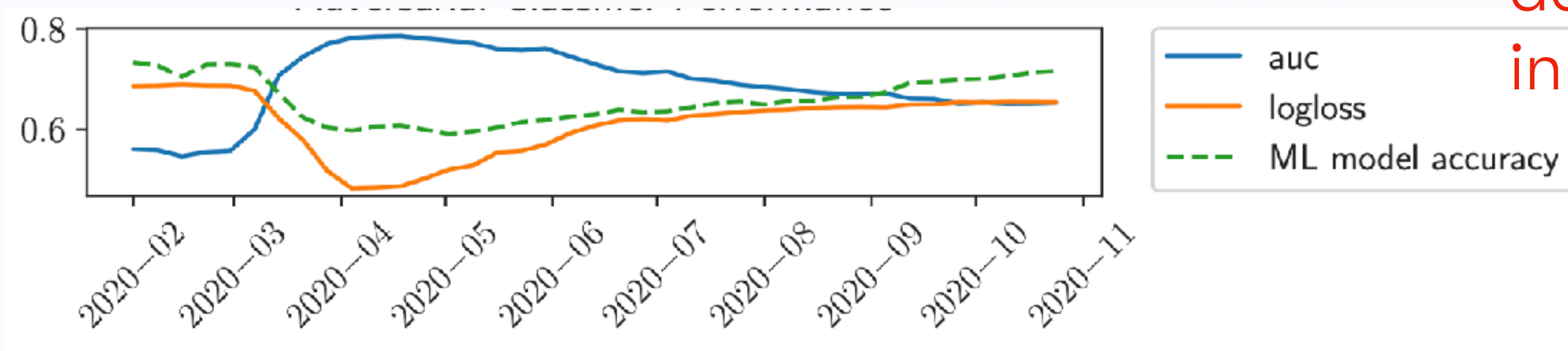
- Can't we train a model to detect drift??
- Naive solution: proxy model F trained to predict whether d comes from D or D'
 - If $F(d)$ converges to 50% AUC, $D \approx D'$
 - Prototyped at Uber ([Pan et al. 2020](#))
 - *Adversarial validation* (think of F as a discriminator)
- Too expensive for real-time monitoring



[Di Sipio et al. 2021](#)

Detecting Drift in Real-Time

- Our insight: incrementally fit the discriminator $F(d)$ for real-time use
 - On new prediction, sample some tuple from D or D' with $p = 0.5$
 - Do forward & backward pass on the tuple
 - Log F 's loss over time

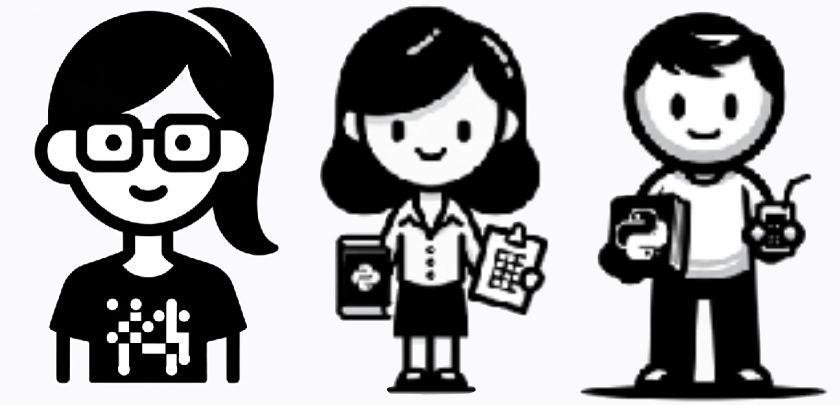


↑
Can't keep
whole
datasets
in memory

↑
Breaks for many real-
world tasks & metrics

- Tasks where subgroups are not balanced
- Weighted metrics (e.g., F-1)

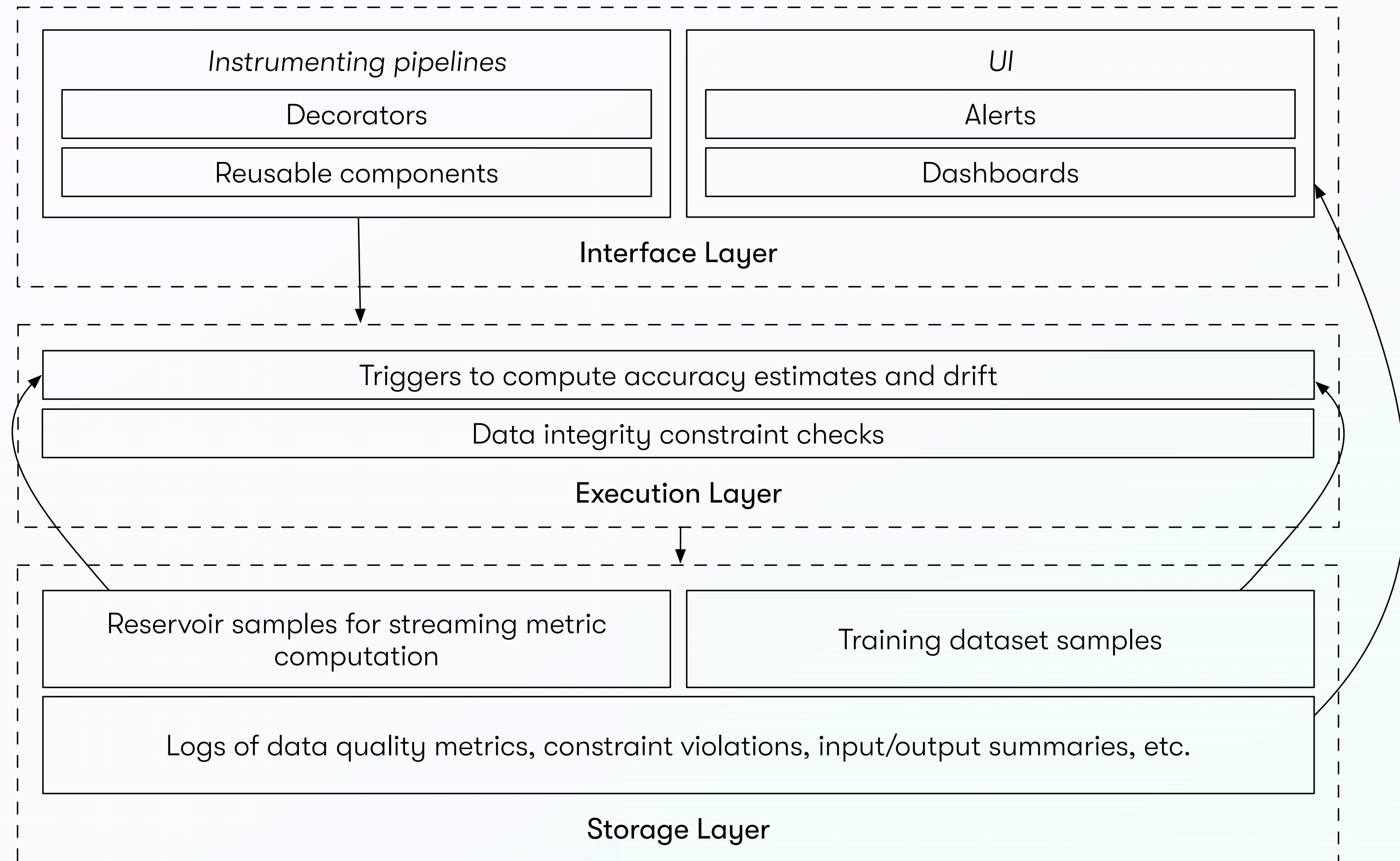
Proposed MLTRACE v2 Architecture



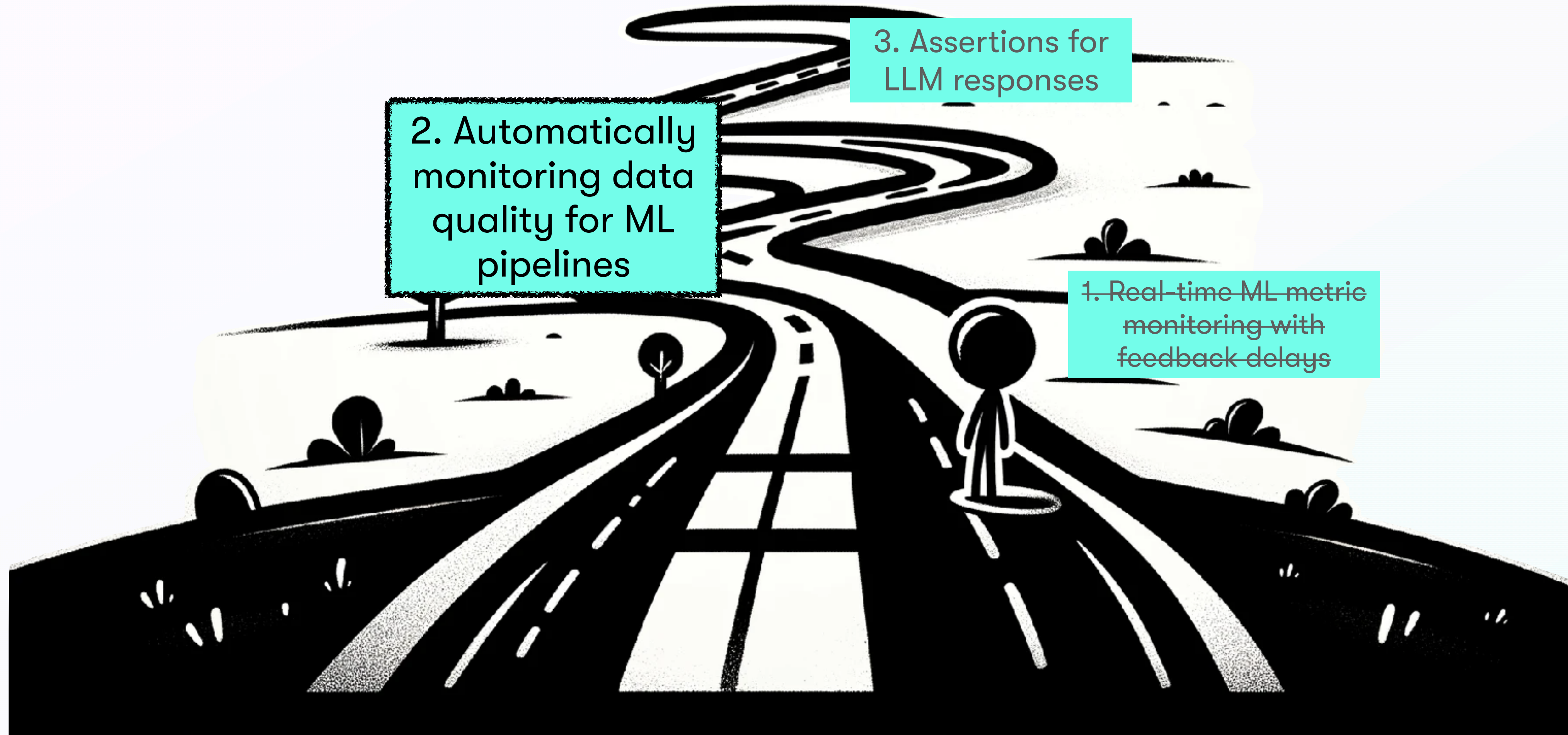
✦ Flexible Python instrumentation

✦ Auto-tuning, low-overhead computation

✦ Minimal information to cover all debugging queries



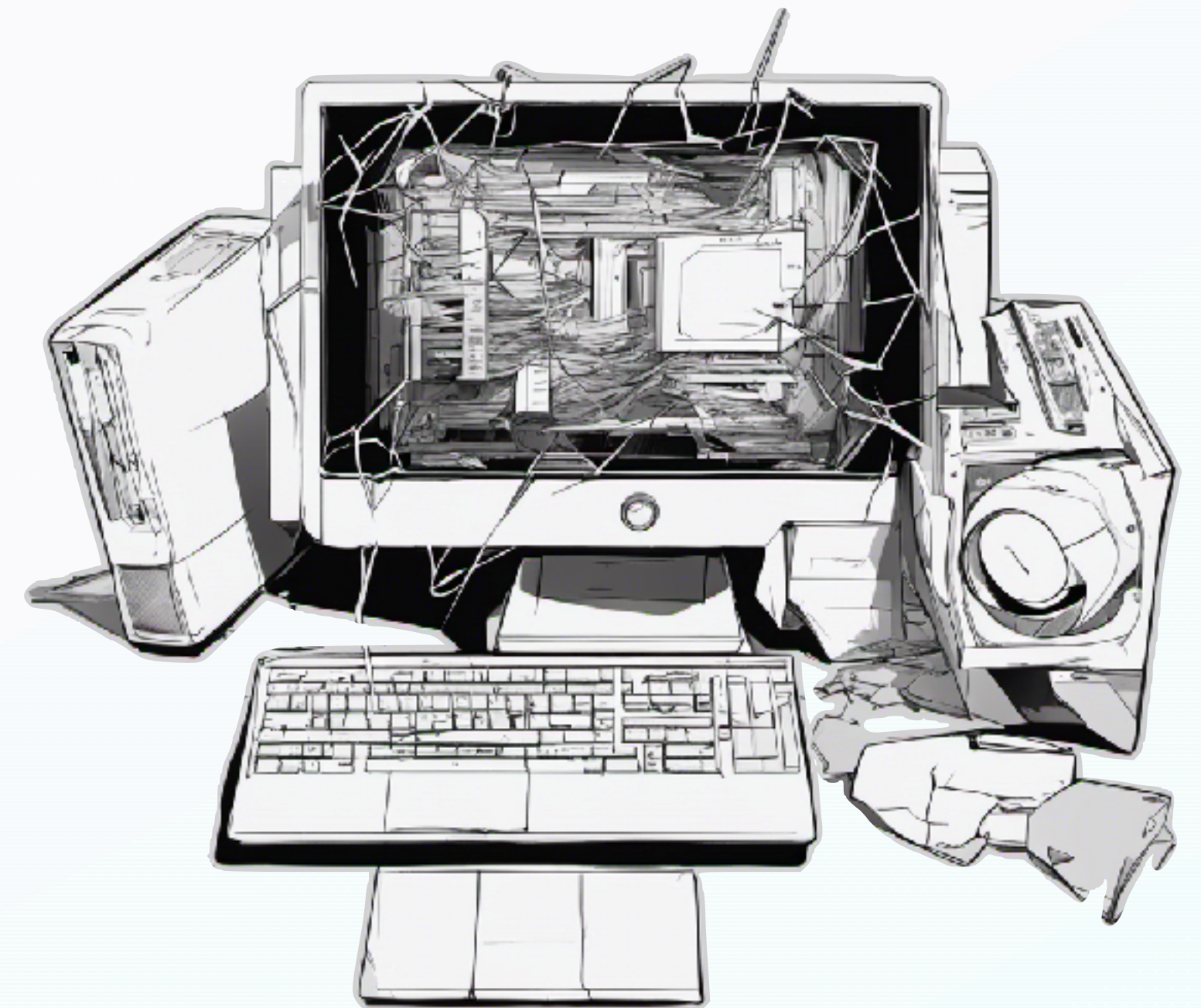
Today's Roadmap



Data errors → Bad ML Performance

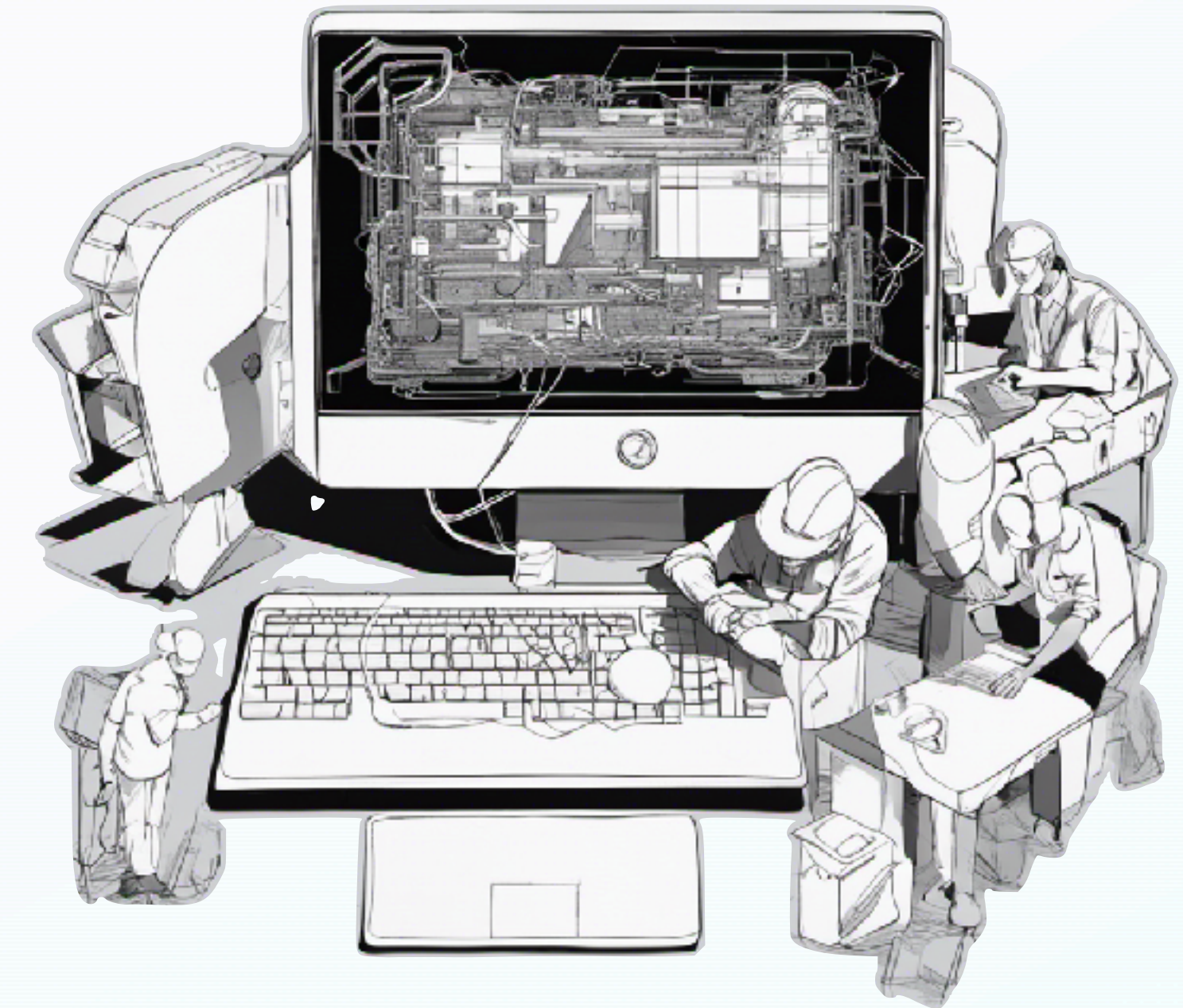
Automatic and Precise Data Validation for Machine Learning (2023)

- Data errors are bad
 - Models are typically trained on clean data, so behavior is unknown on bad data
- Data errors are *especially* bad for production
 - A corrupted partition of data leads to:
 - Bad ML predictions for that partition
 - Bad ML predictions from any models retrained on the corrupted data




Infinitely Many Data Errors

- We get data errors we can't anticipate
- Ex: feature derived from an API
 - `num_followers = api.get_num_followers(user_id)`
 - What if this function is broken?



We need to *automatically* detect when pipeline input data is corrupted!

Data Validation for Machine Learning

- A data validation method takes some data quality statistic(s) and triggers an alert  if some condition is satisfied
- What makes a good data validation system?
 - Triggers alerts for corrupted data that leads to meaningful performance drop (recall)
 - Doesn't cause alert fatigue, or trigger too many false positive alerts (precision)
 - Scalable
 - Should handle thousands of features, many correlated
 - Shouldn't require manual tuning

Prior ML Data Validation Techniques

- Schema Validation (Breck et al., *Data Validation for Machine Learning*)
 - type-check features
 - assert completeness
 - ensure values lie within a predefined vocabulary or bounds

feature	checks
num_followers	≥ 0 , int-valued
viewer_id	foreign key
session_time	≥ 0 , float-valued
...	...

X {num_followers: -19,
viewer_id: 1021344,
session_time: ...}

? {num_followers: 100000000,
viewer_id: 1021344,
session_time: ...}

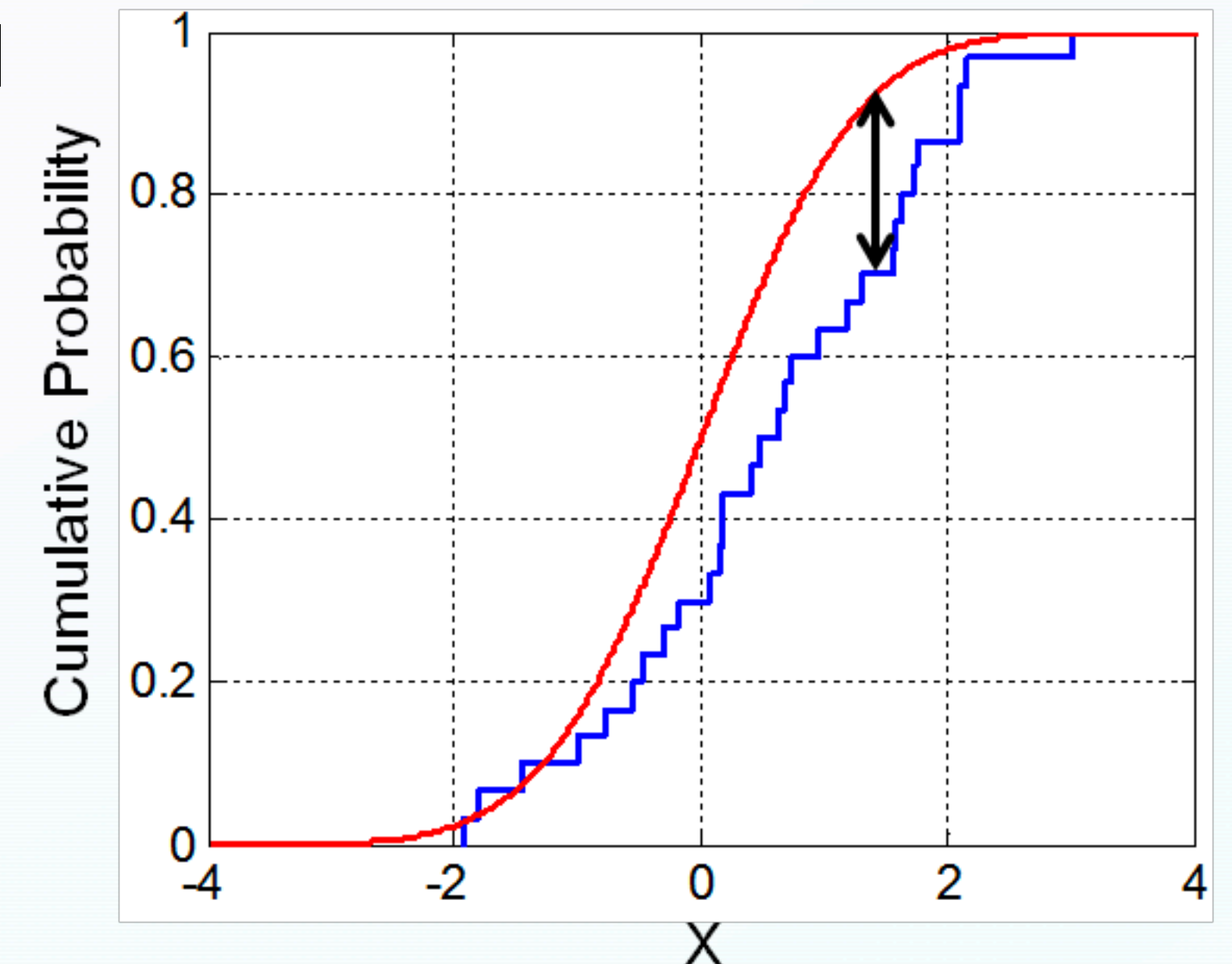
Prior ML Data Validation Techniques

- Column-level constraints (Schelter et al., Automating Large-Scale Data Quality Verification):
 - Monitor aggregate statistics (e.g., mean, completeness)
 - Requires engineer specification and manual fine-tuning

feature	checks
num_followers	completeness \geq 90%, mean \in [5, 100]
viewer_id	N/A
session_time	completeness \geq 90%
...	...

Prior ML Data Validation Techniques

- Distributional tests
 - Verify differences between current partition and historical partitions
 - E.g., KL divergence, Earth-Movers Distance
- Pitfalls
 - Always gives small p-values with a reasonable number of tuples
 - Univariate
 - Can neglect temporal patterns



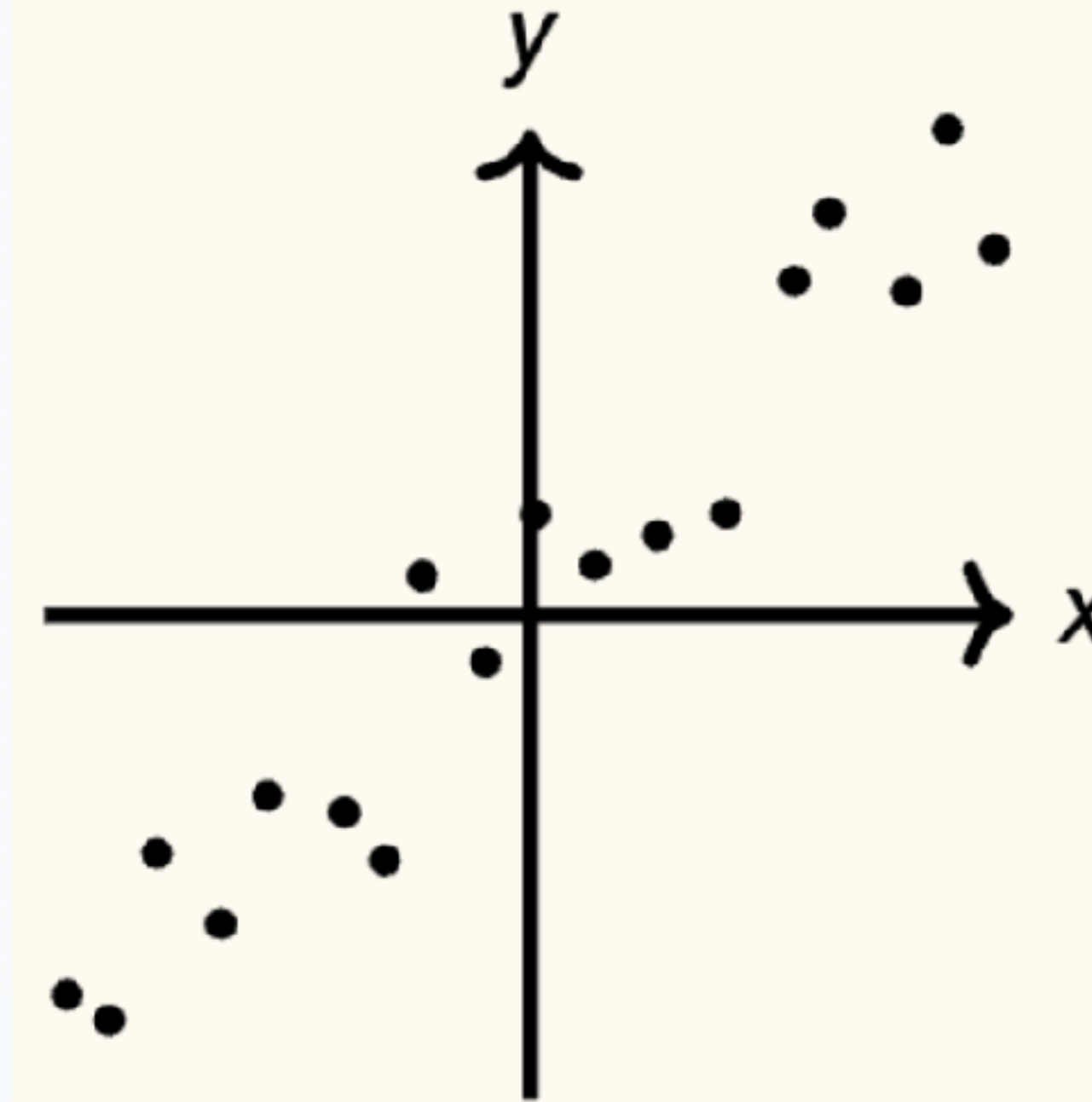
Prior ML Data Validation Techniques

Method	No Manual Tuning Needed	Handles Correlated Features	High Precision and Recall
Schema validation	✗	✗	✗
Monitoring aggregate statistics of features	✗	✗	✗
Statistical tests	✓	✗	✗
GATE (our method)	✓	✓	✓

Why is precise data validation so hard?



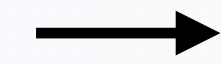
Temporal Patterns



Highly-Correlated Features in Training Datasets

Partition Summarization

num_followers	viewer_id	session_time	...	date
19	1021344	3.1	...	10/21
20	1021344	3.4
21	1021344	1.9	...	10/22
21	1021344	0	...	10/22
21	1021344	0.1	...	10/22
21	1021344	1
21	1021344	3.7	...	10/23
21	1021344	3.9	...	10/23
21	1021344	4.1	...	10/23
21	1021344	21
54	1021344	1	...	10/24
54	1021344	2	...	10/24
54	1021344	10
55	1021344	1	...	10/25
...



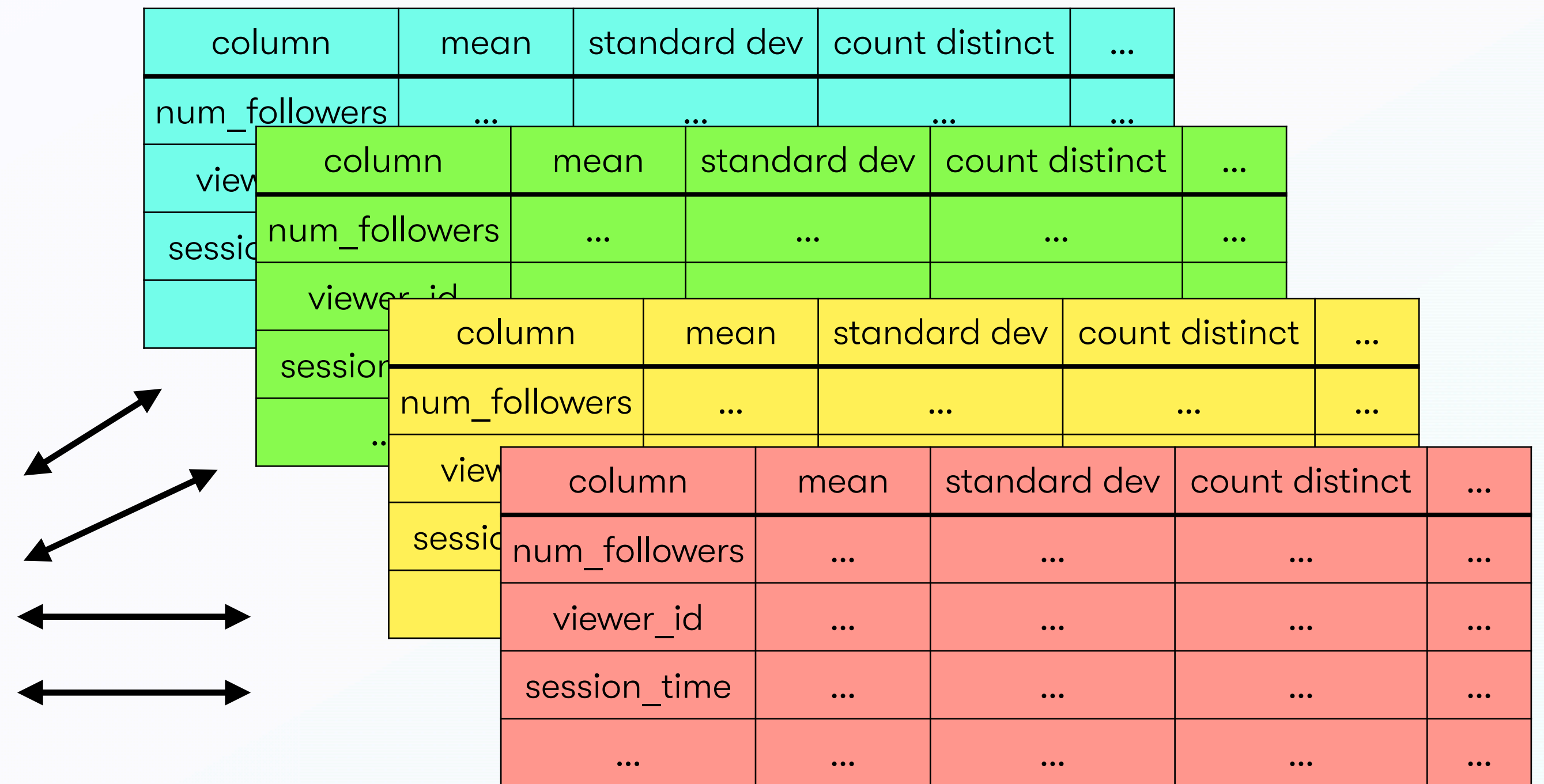
column	mean	standard dev	count distinct	...
num_followers
column	mean	standard dev	count distinct	...
num_followers
viewer_id
session_time
column	mean	standard dev	count distinct	...
num_followers
viewer_id
session time
column	mean	standard dev	count distinct	...
num_followers
viewer_id
column	mean	standard dev	count distinct	...
num_followers
viewer_id
session_time
...

Partition Summarization

Summary statistics:

- Completeness
- Mean
- Standard Deviation
- Num Unique Values
- Top frequency
- Earth-mover's distance

column	mean	standard dev	count distinct	...
num_followers
viewer_id
session_time
...



Compare partition summaries to each historical partition summaries & run anomaly detection

Adapting Existing Methods to the PS Setting

- Existing methods center around some statistic(s)
- Our adaptation
 - Step 1: compute this statistic Q at the partition granularity for each feature
 - Step 2: normalize Q across the features (via z-score)
 - Step 3: alert if average Q is an outlier (e.g., 95th percentile, exceeds threshold, etc)
- Example: completeness (fraction of non-nulls)

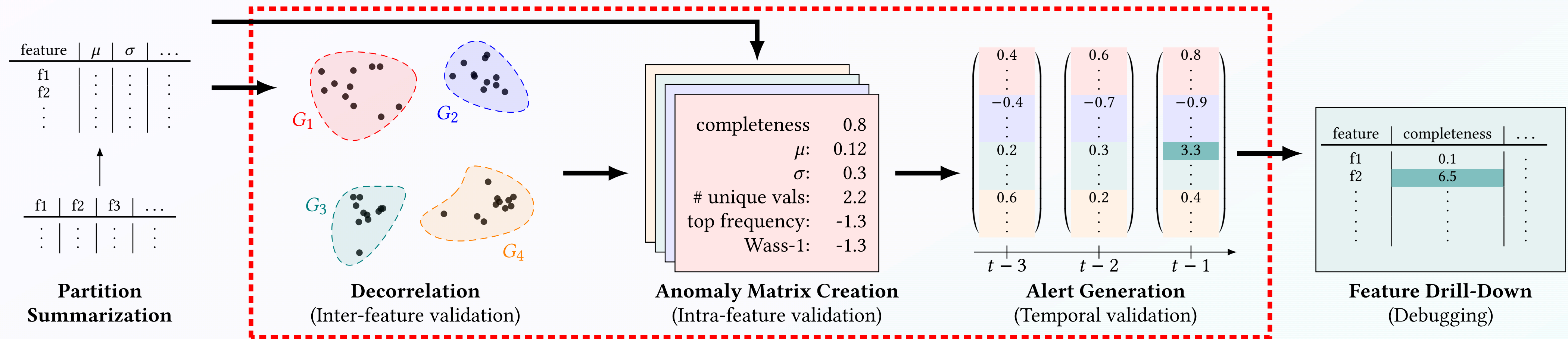
Adapting Existing Methods to the PS Setting

- Example: completeness (fraction of non-nulls)
- Step 1: compute completeness per column per partition
- Step 2: normalize across features
- Step 3: average normalized completeness score across all columns per partition (1 score per partition)
- Alerting: if a partition's average score \geq threshold

num_followers	viewer_id	session_time	...	date
19	1021344	3.1	...	10/21
20	1021344	3.4
21	1021344	1.9	...	10/22
20	1021344	Null	...	10/22
Null	Null	0.1	...	10/22
21	1021344	Null
21	Null	3.7	...	10/23
Null	avg(z)
21	1021344	4.1	...	10/23
21	1021344	21

GATE: Partition Summarization + Decorrelation

- Adapted several data validation methods, but they still had false positives because of correlated feature columns



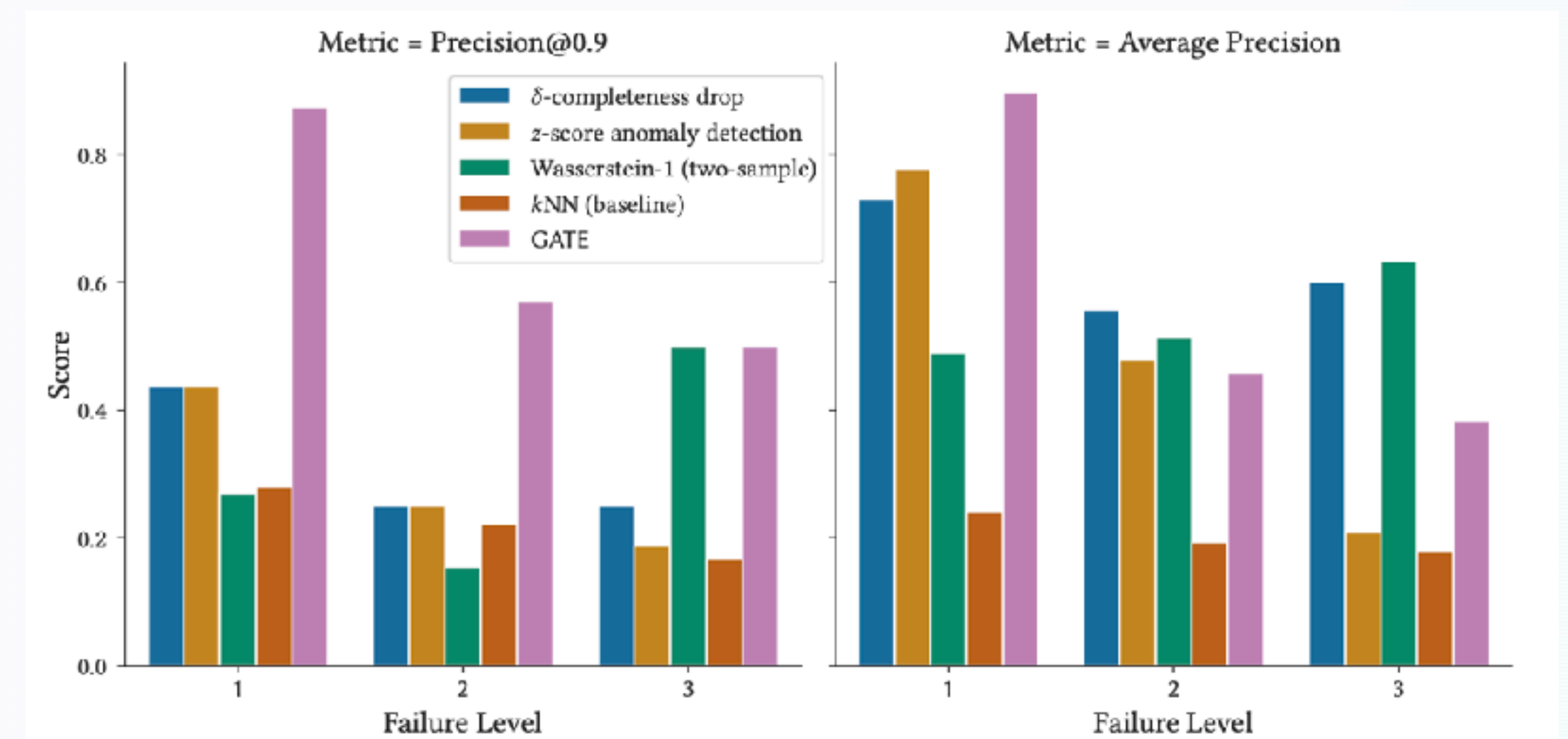
Reduce false positive alerts on data errors by clustering correlated features

Empirical Study

- Datasets: 2 months of Instagram ML pipeline input datasets
 - Tens of thousands of feature columns
 - Different levels of data quality corruptions (labeled by on-call ML engineers)
- Methods: adapted baselines to the PS setting & GATE
- Evaluation: measuring precision @ 90% recall

Empirical Study

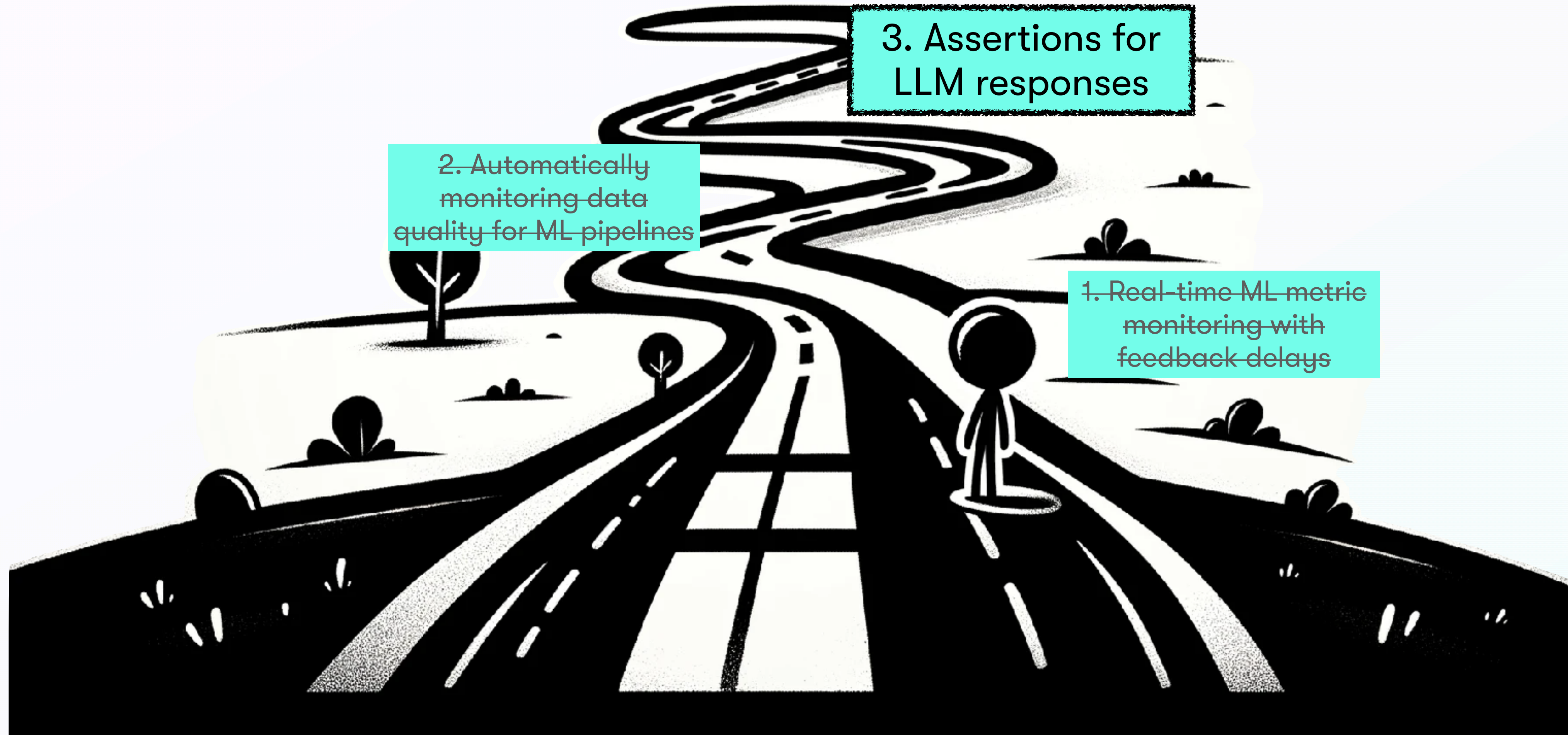
- 2.1x average improvement in precision@0.9
- ~2 orders of magnitude faster than 2-sample statistical tests
- < 1 second for a partition of 10k features on average
- Works well when there are low data corruption rates (< 30%)



Takeaway: GATE precisely and automatically detects ML pipeline input data errors

<https://github.com/dm4ml/gate>

Today's Roadmap



2. Automatically monitoring data quality for ML pipelines

3. Assertions for LLM responses

1. Real-time ML metric monitoring with feedback delays

Monitoring LLM Response Quality is Hard

Work in Progress

- Most people muddle their way to a deployed LLM pipeline without a clear sense of progress and how well the pipeline might do in production
- Many of our tricks for monitoring tabular data don't easily apply here
- Accuracy and “good” are poorly defined for free-form responses



“We have ground truth guidance, not labels. It takes a human to see if a response is good.”

“In traditional ML, you have statistics to optimize for. But now I don't know how to optimize for vibes; I don't know how to optimize for vibes”

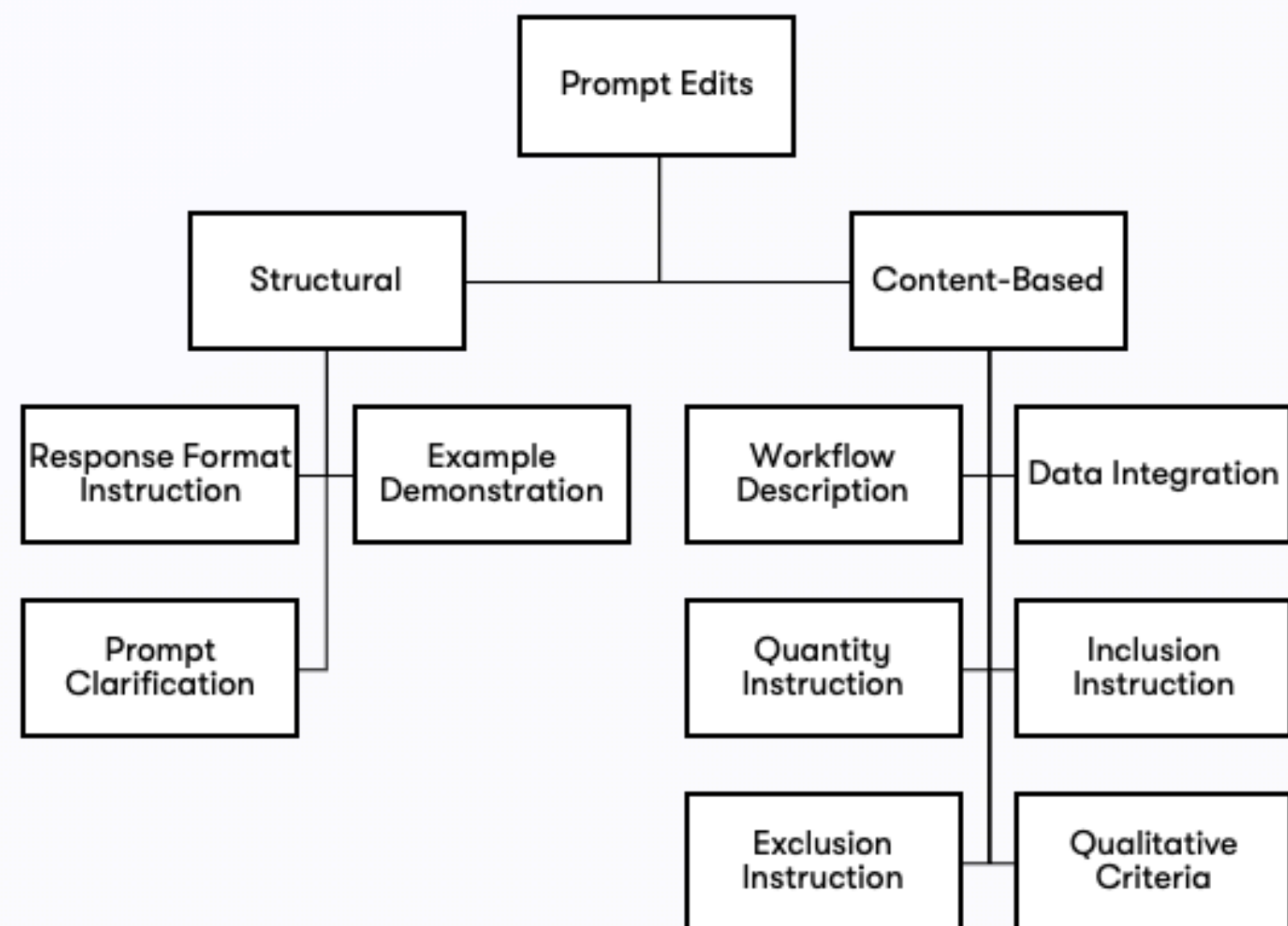
SPADE ♠: System for Prompt Analysis and Delta-Based Evaluation

- How can we *automatically* suggest assertions for developers to run on all their prompts & responses?
- We learn from *prompt version history* to identify what's important to the engineer and what LLMs are uniquely bad at

Version	Prompt Template
1	Suggest 5 apparel items to wear to {event}. Return your answer as a Python list of strings.
2	A client ({client_genders}) wants to be styled for {event}. Suggest 5 apparel items for {client_pronoun} to wear. Return your answer as a Python list of strings.
3	A client ({client_genders}) wants to be styled for {event}. Suggest 5 apparel items for {client_pronoun} to wear. For wedding-related events, don't suggest any white items unless the client explicitly states that they want to be styled for their wedding. Return your answer as a python list of strings

SPADE ♠ Taxonomy

- SPADE first finds the diffs between consecutive prompt versions, i.e., any new instructions that didn't exist in the earlier version
- For each statement in the diff, SPADE categorizes this delta according to our taxonomy



Category	Example Addition or Edit to a Prompt	Evaluation Idea
Response Format Instruction	"Return your answer as a Python dictionary"	Verify LLM response can be parsed correctly
Example Demonstration	"Here is an example question and response: Question: What should I wear to a workout class? Answer: {'tops': 'black moisture-wicking tank top', 'shoes': 'black Nike Pegasus...'"	Infer detailed structure from example (e.g., specific keys, headers) and verify this in responses
Prompt Clarification	" Return Give me a descriptive list"	N/A (as long as the meaning of the prompt is unchanged)
Workflow Description	"First, identify the dress code of the event. Then..."	Check that the LLM response matches a dress code
Data Integration	"The user does not like {dislikes_placeholder}"	N/A
Quantity Instruction	"The outfit must have at least 3 items"	Assert that the response satisfies the count
Inclusion Instruction	"Make sure your outfit is complete, i.e., it includes a top, shoe, and lower-body garment"	Assert specific phrases or keywords are included from responses
Exclusion Instruction	"Do not suggest sneakers for wedding-related events"	Assert specific phrases or keywords are excluded from responses
Qualitative Criteria	"Include a statement piece in your suggestion"	Create a "scorecard" to ask an LLM or expert to evaluate

SPADE ♠ Function Generation

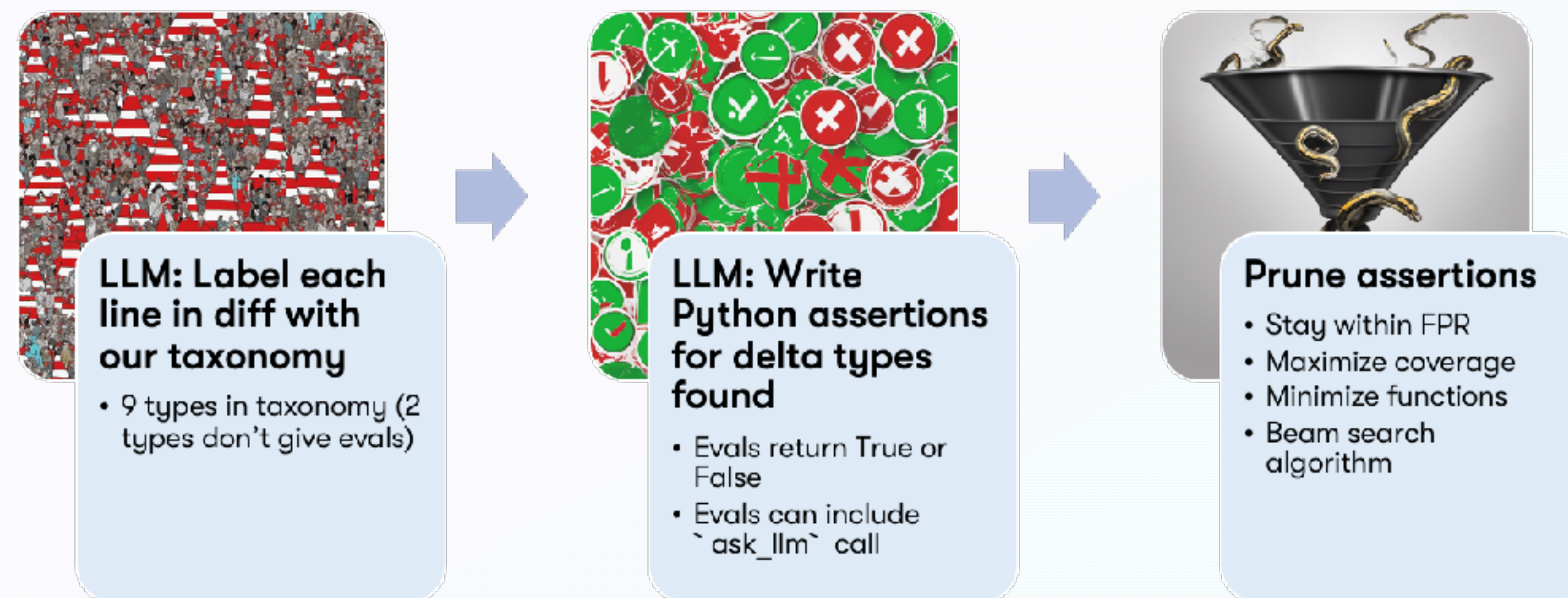
- Based on the categories tagged, SPADE uses GPT-4 to generate predicates to monitor

```
# Needs LLM: False
def check_excludes_white_wedding(prompt: str, response: str) -> bool:
    """
    This function checks if the response does not include white items for wedding-related events,
    unless explicitly stated by the client.
    """
    # Check if event is wedding-related
    if "wedding" in prompt.lower() and "my wedding" not in prompt.lower():
        # Check if the response includes the word "white"
        return "white" not in response.lower()
    else:
        return True
```

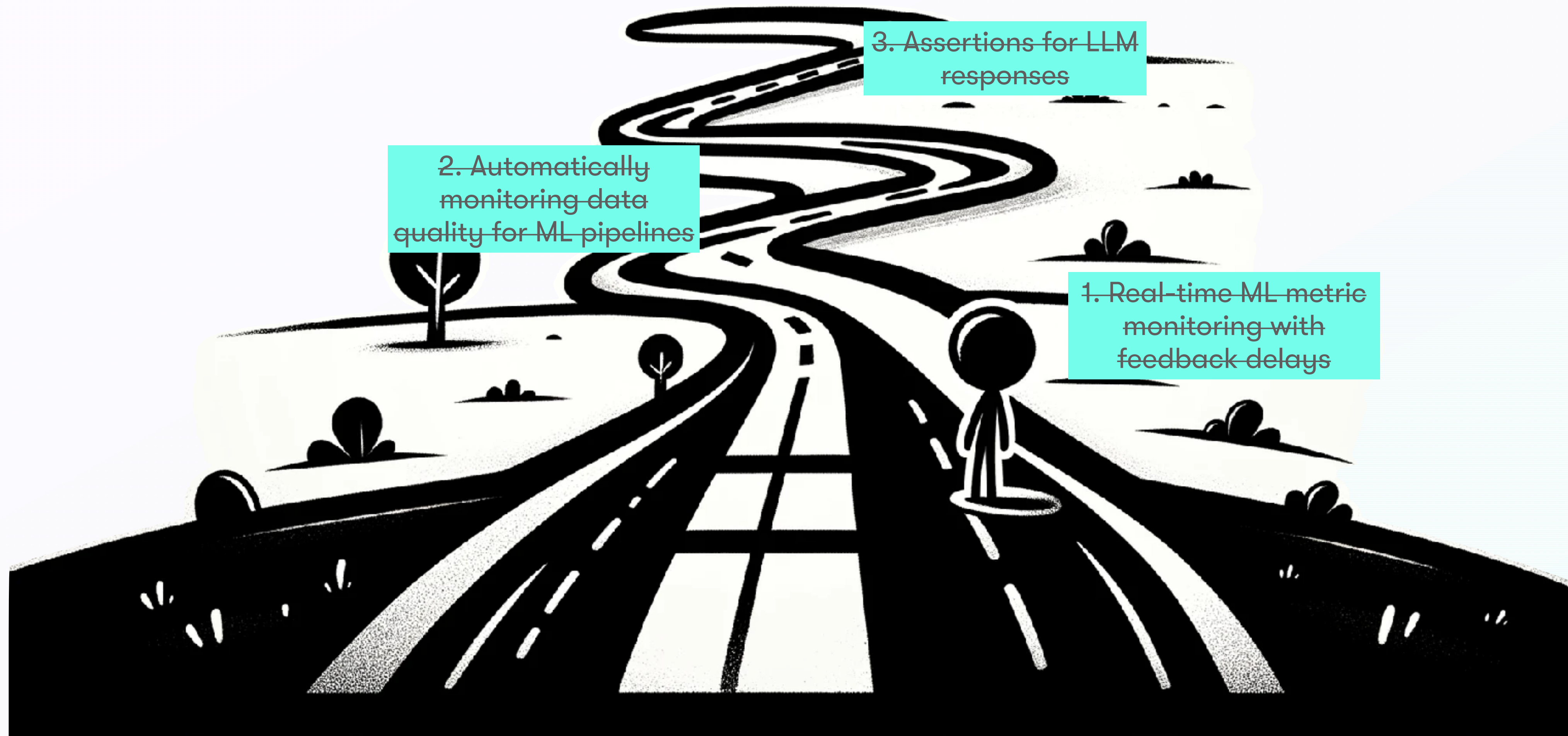
- Depending on how many prompt versions exist, 10s or even 100s of predicates get generated! 🤯

SPADE ♠ Pipeline

- Pruning the assertions generated is hard because there usually isn't a big dataset of examples (i.e., prompt-response pairs)
- It's like trying to find functional dependencies: if assertion A \rightarrow B then drop B



Today's Roadmap



3. Assertions for LLM responses

2. Automatically monitoring data quality for ML pipelines

1. Real-time ML metric monitoring with feedback delays

Summary and Looking Ahead

- It is a great time to be working on data management for ML!
- We only focused on deployment & monitoring, but there are many opportunities to improve ML engineering workflows and lower the barrier to entry for ML
- shreyashankar@berkeley.edu

