FSDL 2022

Continual Learning

Josh Tobin

September 12, 2022

Josh Tobin
@josh_tobin_

In the real world, data distributions are almost never static.

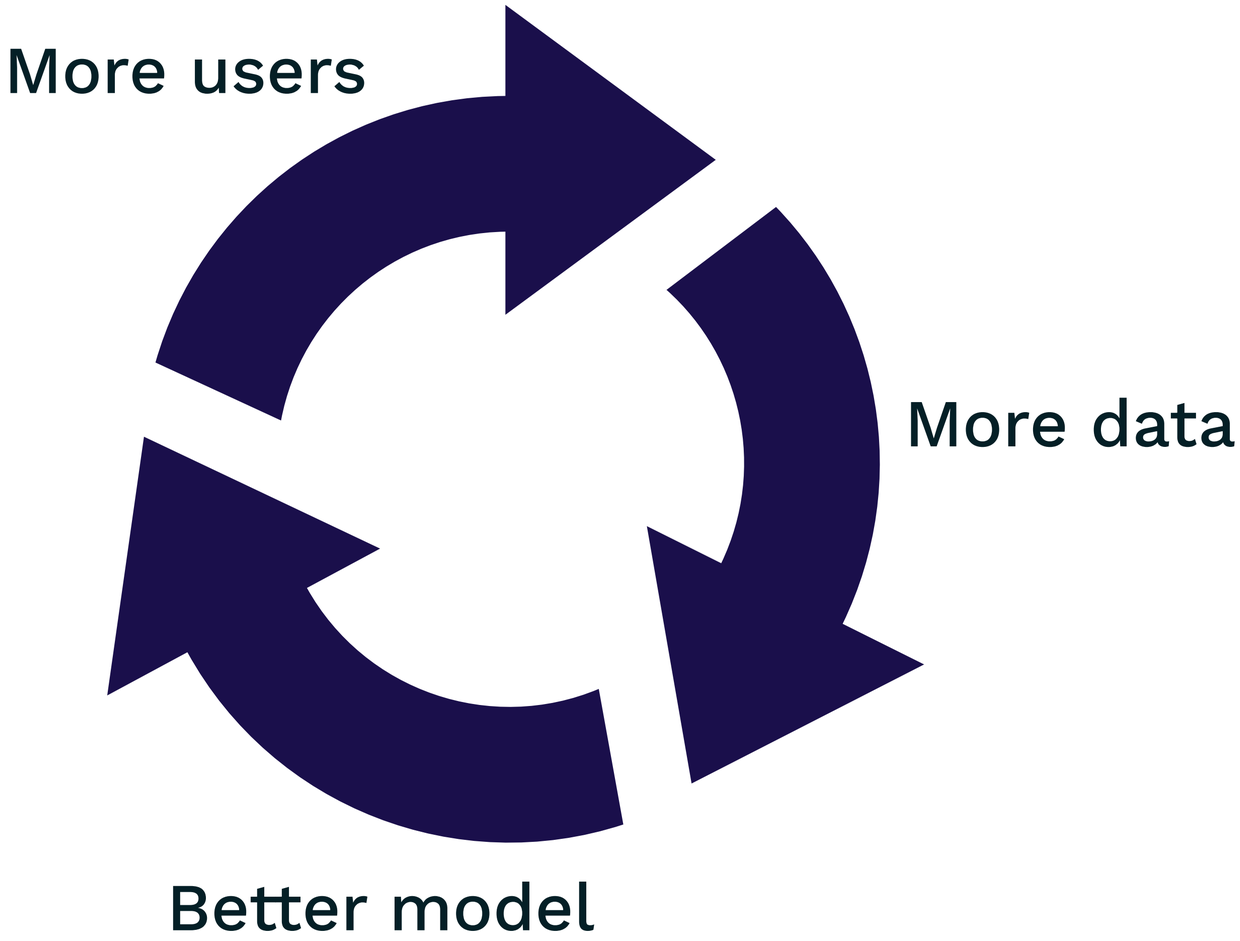
So if you want to use ML in production, your goal should be to build a continual learning machine, not a static model.

2:35 PM · May 8, 2021 · Twitter Web App

View Tweet analytics

81 Retweets **4** Quote Tweets **585** Likes

The dream of how this would work

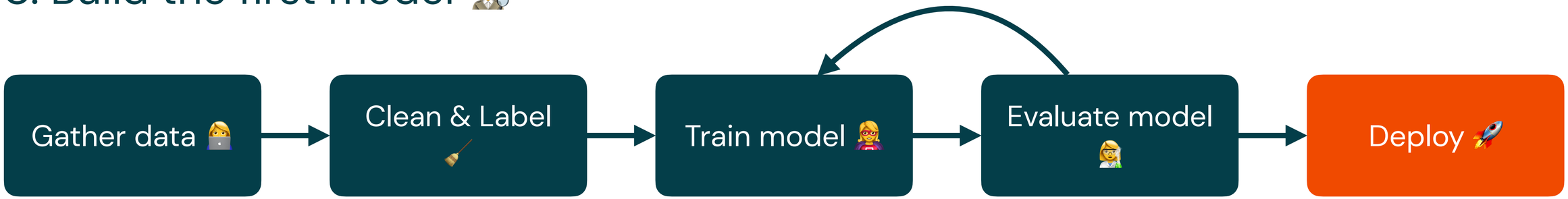


"Operation vacation"

https://www.youtube.com/watch?v=hx7BXih7zx8

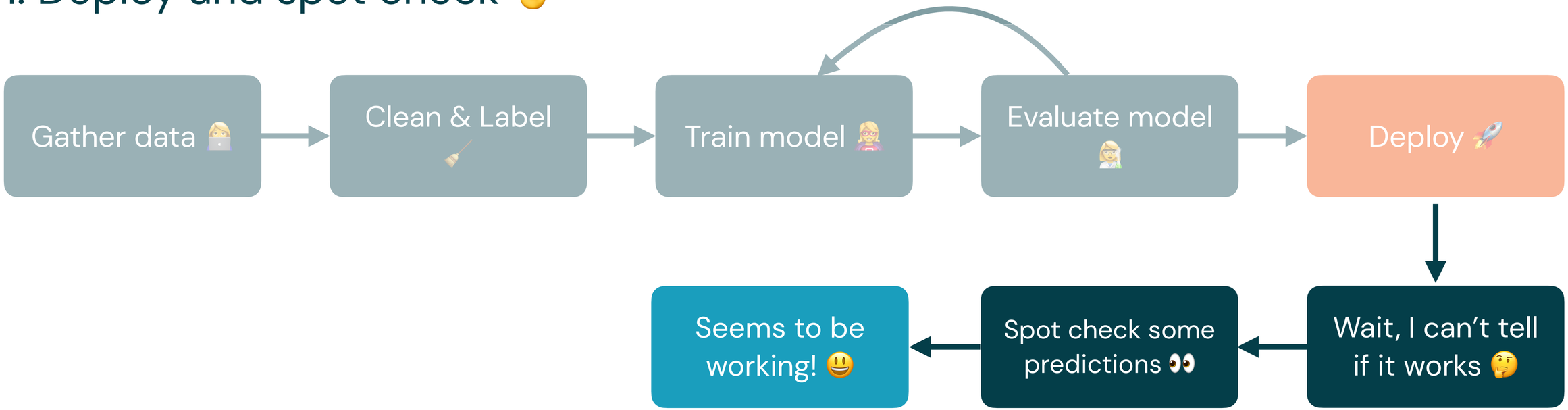
The reality

0. Build the first model 🕵️



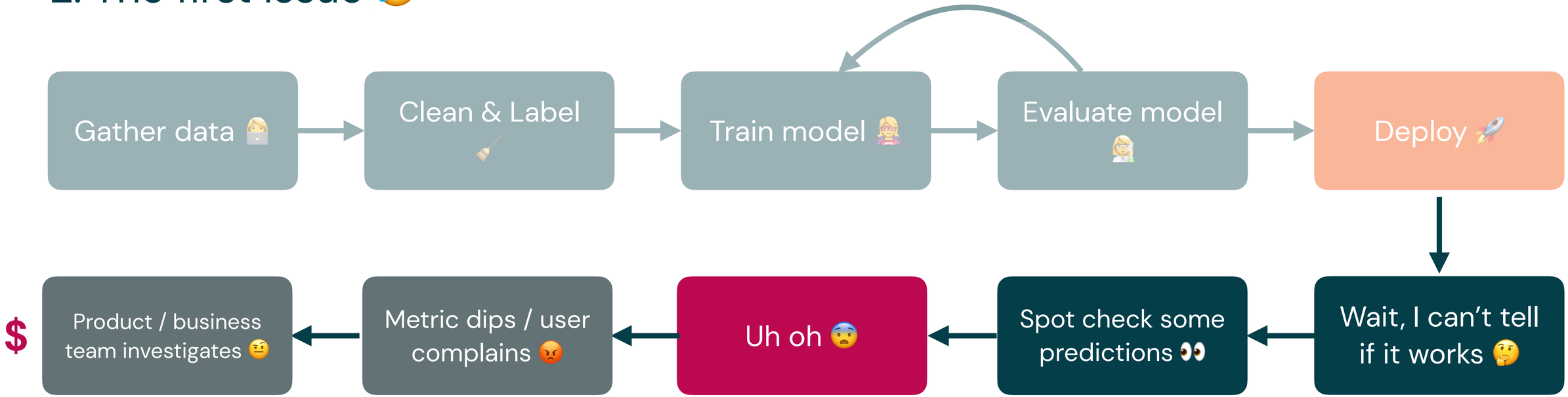
The reality

1. Deploy and spot check 🤞



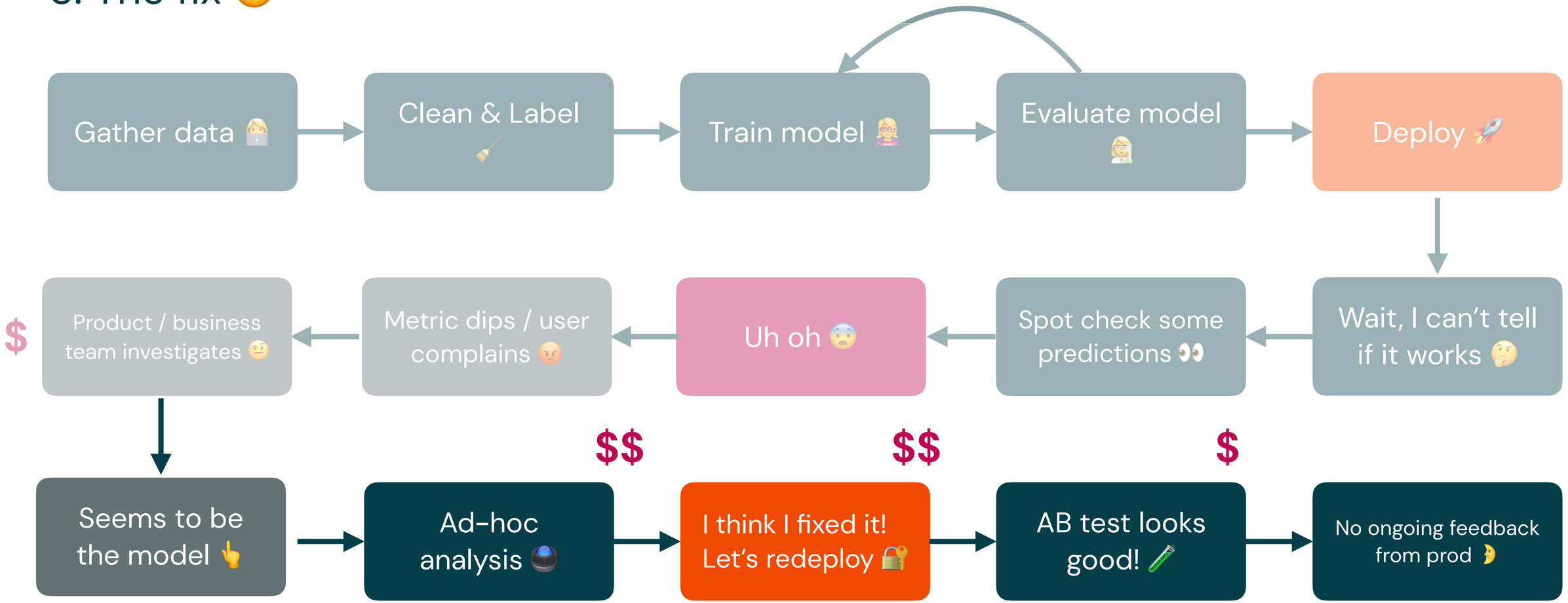
The reality

2. The first issue 😰



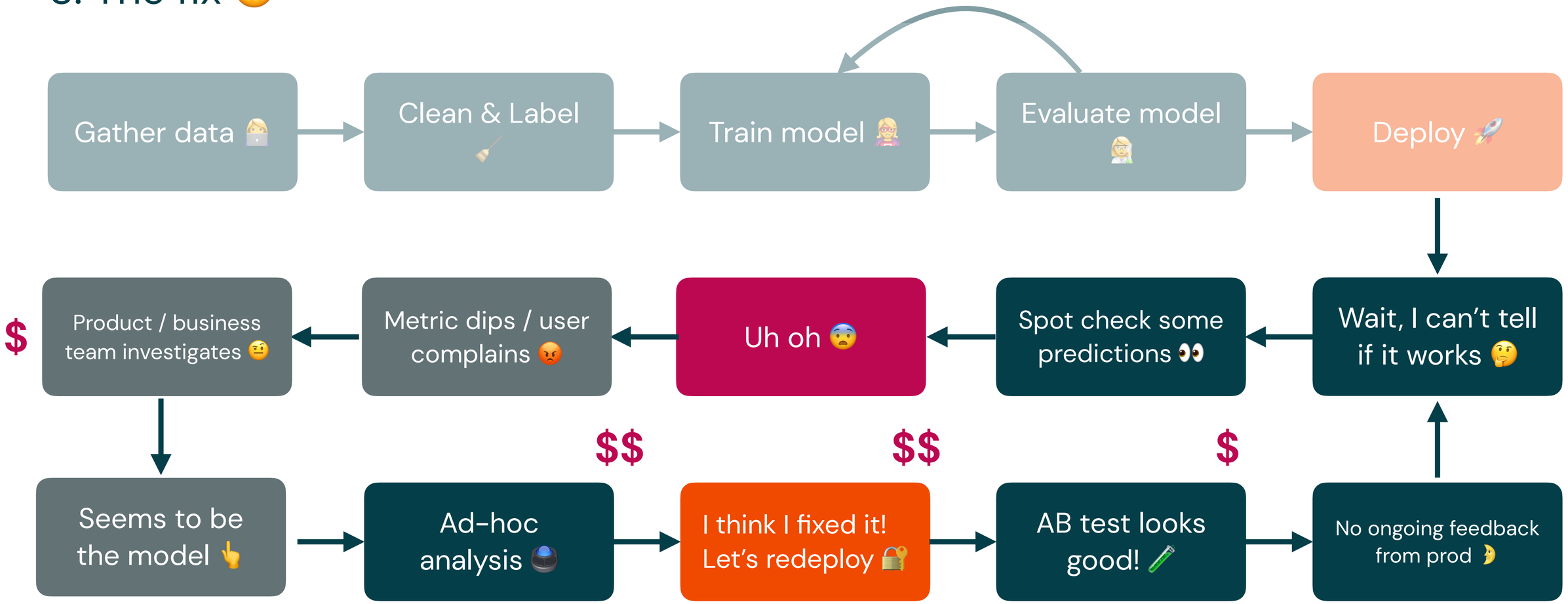
The reality

3. The fix 😵‍💫



The reality

3. The fix 😵‍💫

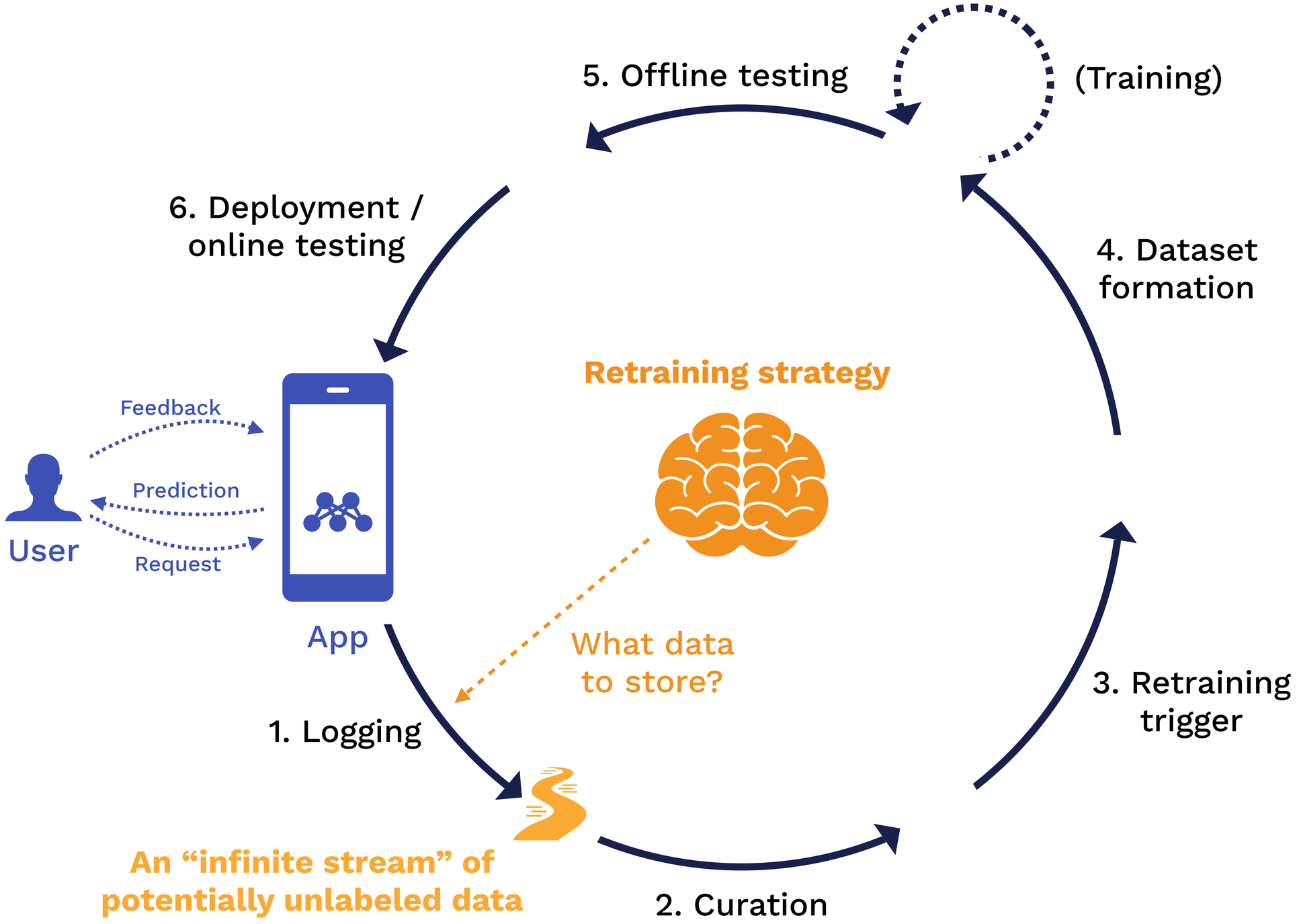


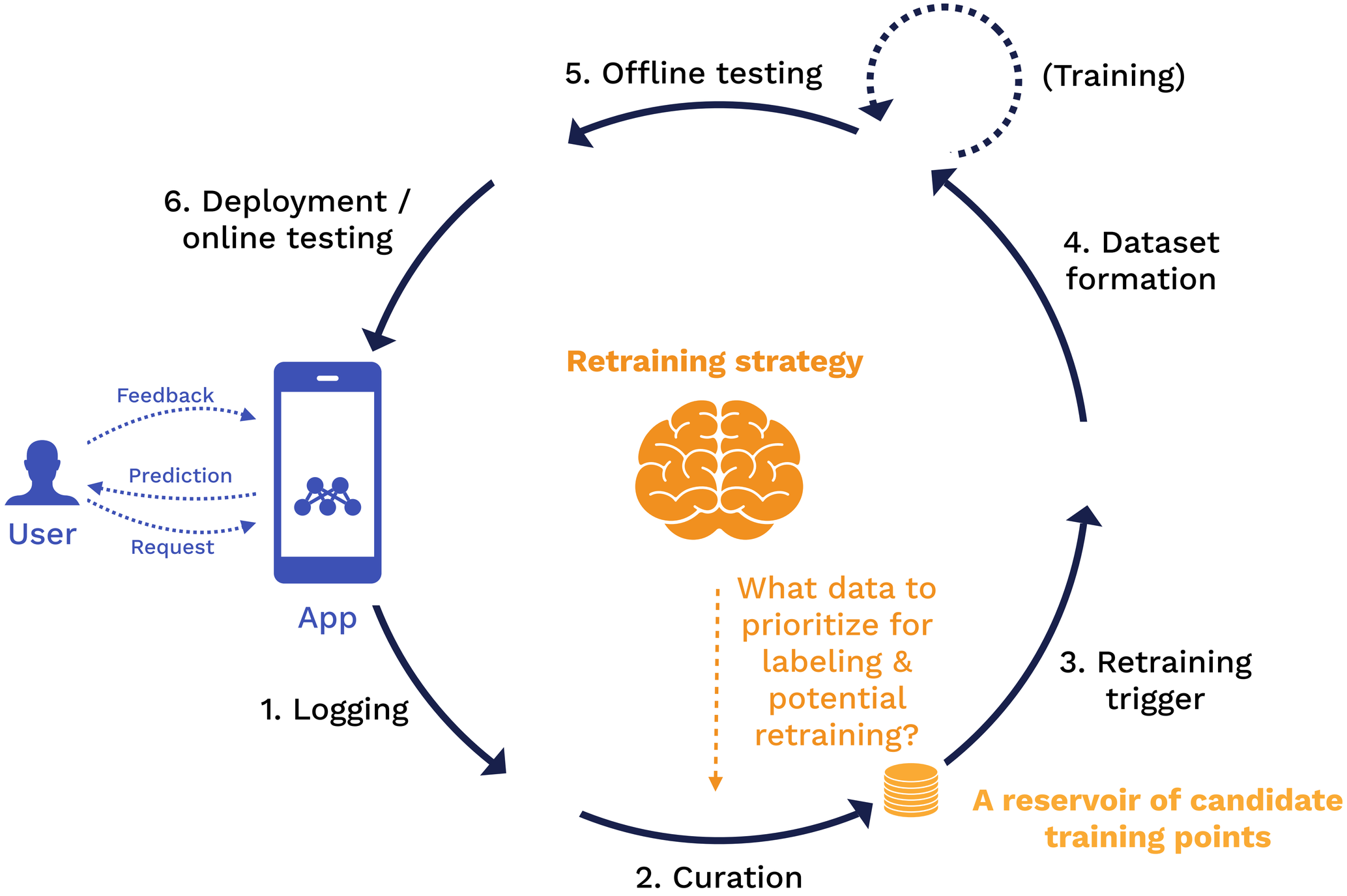
The reality

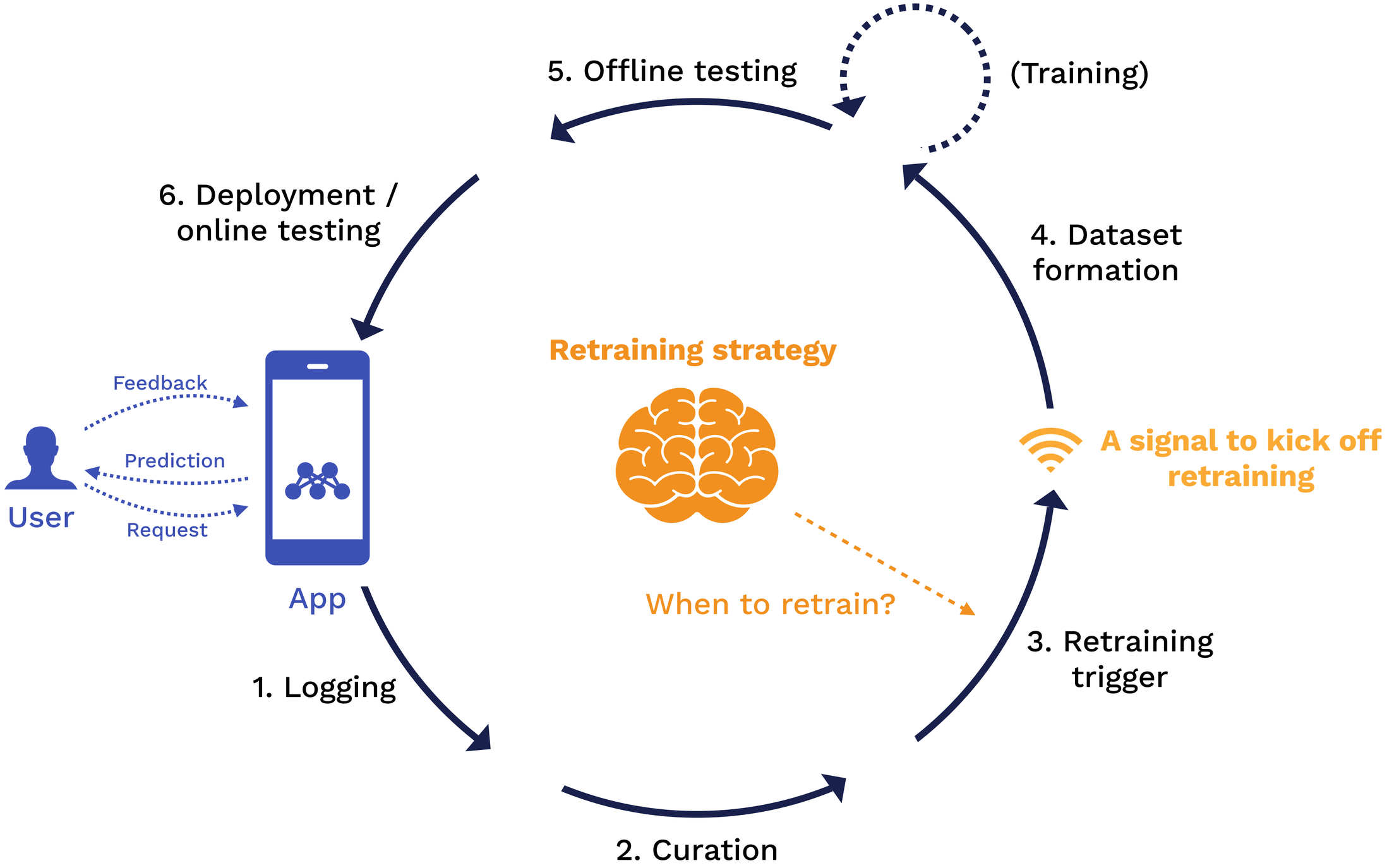
- Continual learning is the least well-understand part of the ML lifecycle
- Few companies do it well today
- Goal of this lecture:
  - Framework for how to think about continual learning
  - Pointers for how to learn about each of the steps
  - Recommendations for how to adopt this gradually

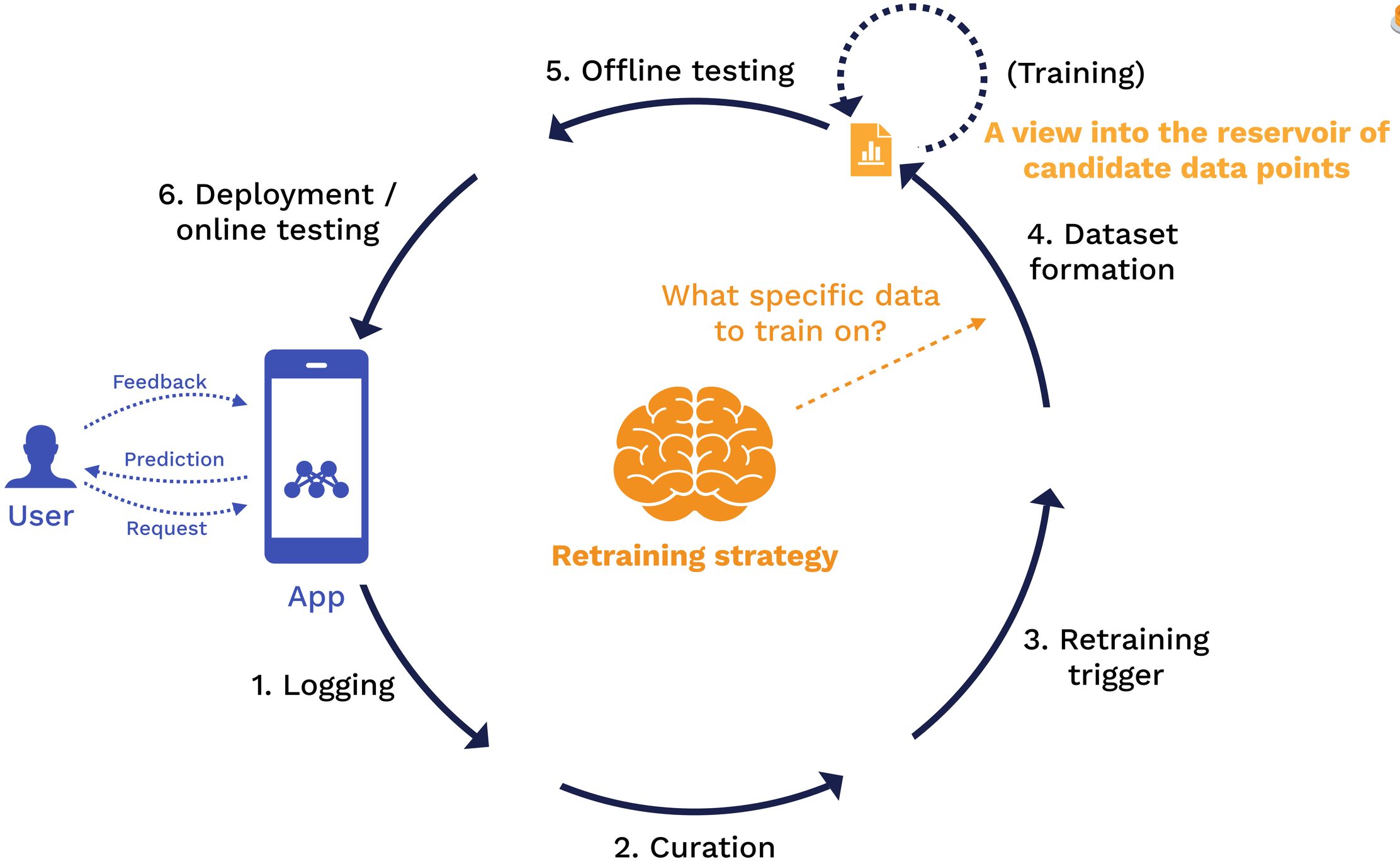
How to think about continual learning

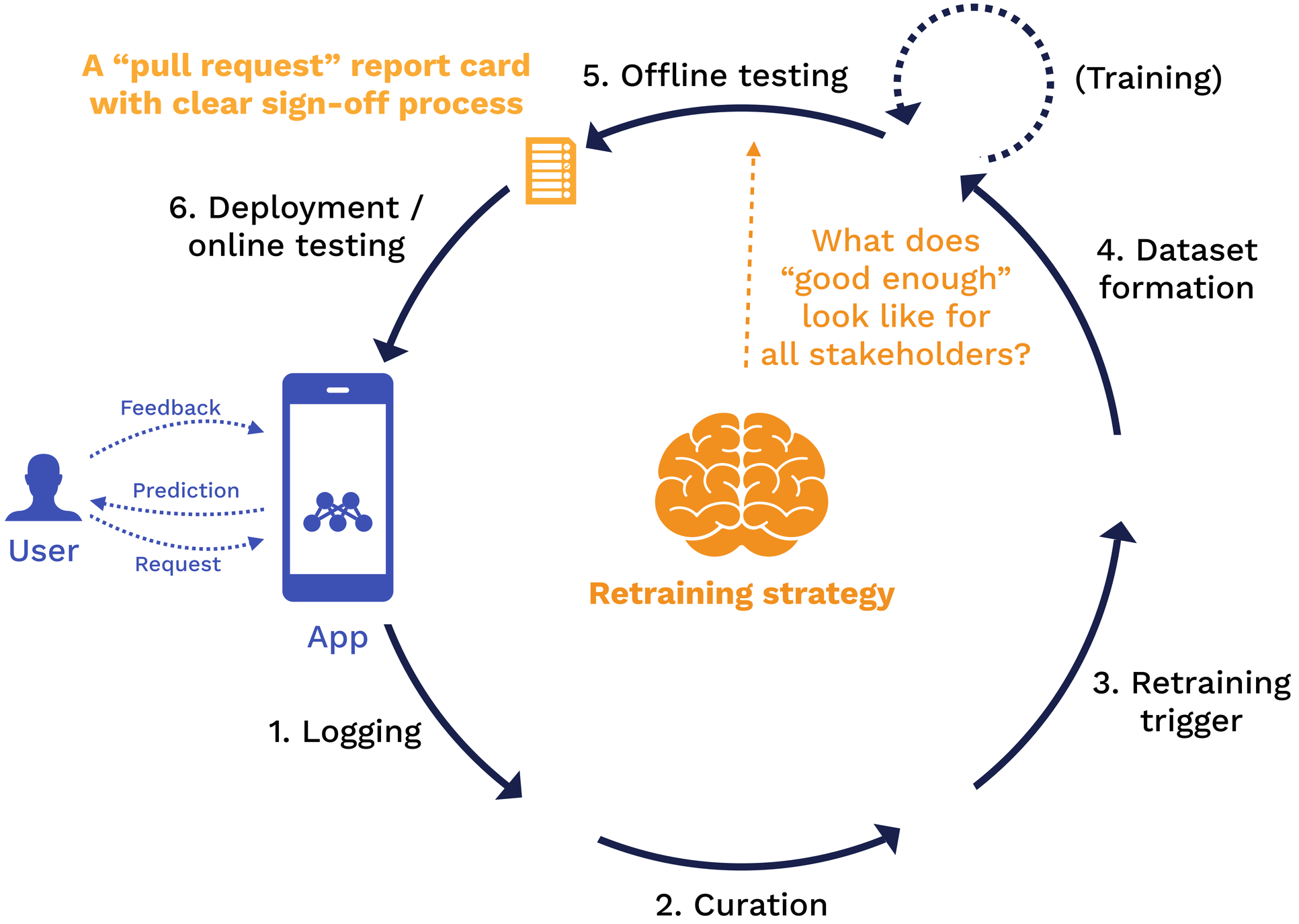
Continual learning: training a sequence of models to adapt to a continuous data stream

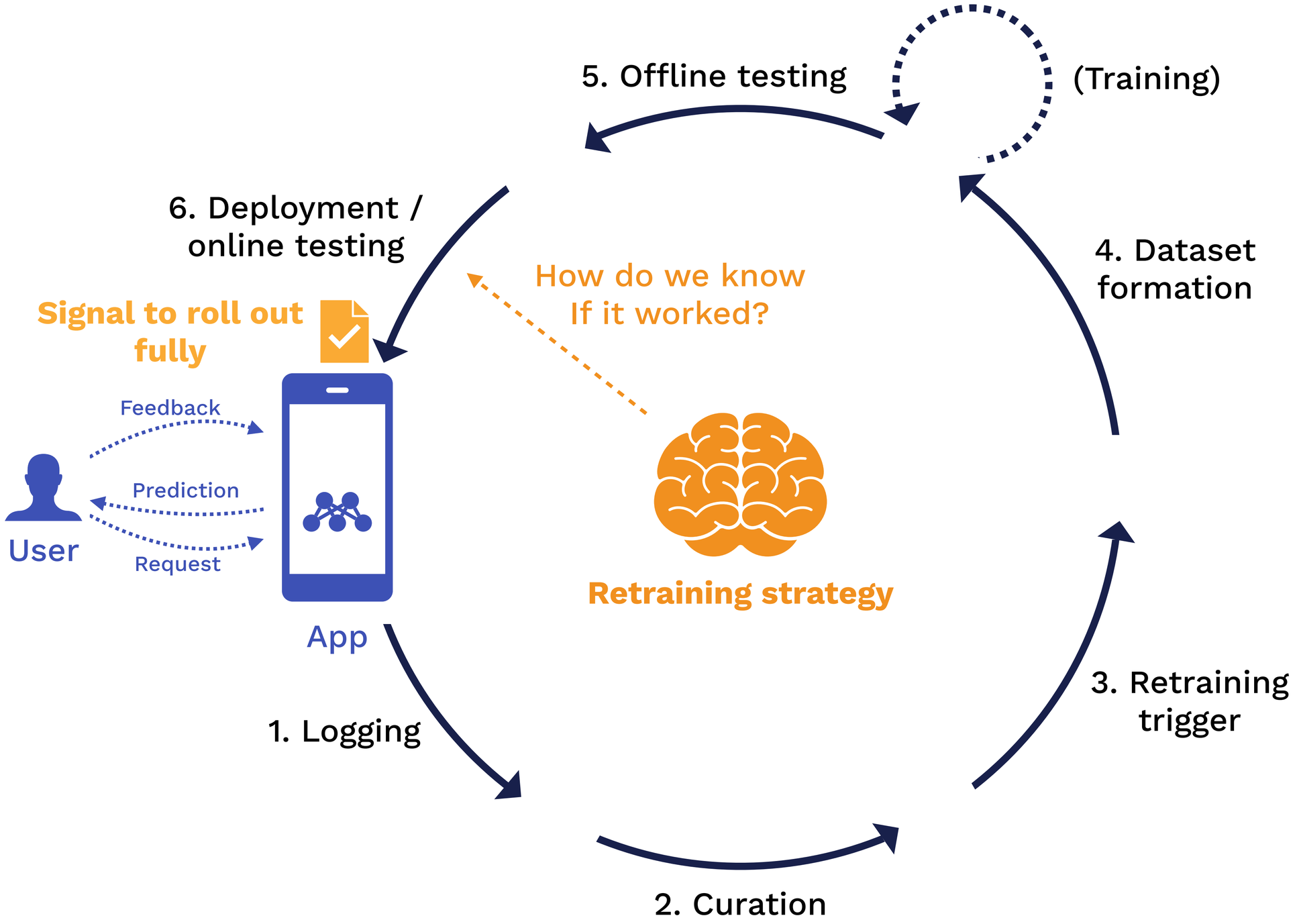

5. Offline testing
(Training)
6. Deployment / online testing
4. Dataset formation
Retraining strategy
Feedback
Prediction
Request
User
App
What data to store?
1. Logging
3. Retraining trigger
An “infinite stream” of potentially unlabeled data
2. Curation

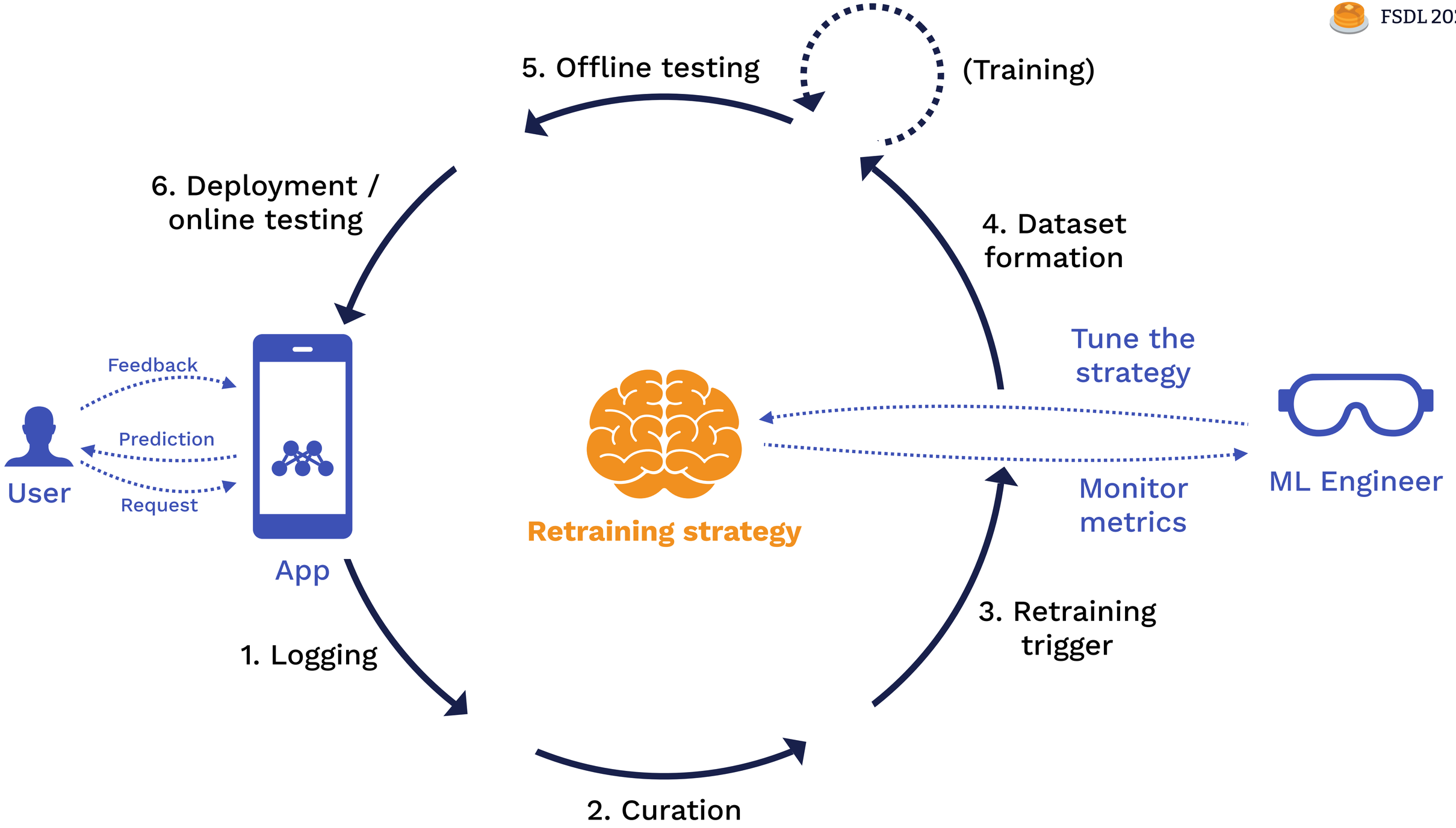

5. Offline testing
(Training)
6. Deployment / online testing
4. Dataset formation
Retraining strategy
Feedback
Prediction
Request
User
App
1. Logging
What data to prioritize for labeling & potential retraining?
3. Retraining trigger
A reservoir of candidate training points
2. Curation


5. Offline testing
(Training)
6. Deployment / online testing
4. Dataset formation
Retraining strategy
Feedback
Prediction
Request
User
App
A signal to kick off retraining
When to retrain?
1. Logging
3. Retraining trigger
2. Curation


5. Offline testing
(Training)
A view into the reservoir of candidate data points
6. Deployment / online testing
4. Dataset formation
What specific data to train on?
Feedback
Prediction
Request
User
App
Retraining strategy
1. Logging
3. Retraining trigger
2. Curation


A “pull request” report card with clear sign-off process
5. Offline testing
(Training)
6. Deployment / online testing
What does “good enough” look like for all stakeholders?
4. Dataset formation
Feedback
Prediction
Request
User
App
Retraining strategy
1. Logging
3. Retraining trigger
2. Curation


5. Offline testing
(Training)
6. Deployment / online testing
How do we know If it worked?
4. Dataset formation
Signal to roll out fully
Feedback
Prediction
Request
User
App
Retraining strategy
1. Logging
3. Retraining trigger
2. Curation


5. Offline testing
(Training)
6. Deployment / online testing
4. Dataset formation
Feedback
Prediction
Request
User
App
Tune the strategy
Retraining strategy
Monitor metrics
ML Engineer
1. Logging
3. Retraining trigger
2. Curation

The simplest retraining strategy is retraining whenever you feel like it

When should you move on from ad-hoc retraining?

- You can consistently achieve similar results when retraining on new data, and
- No one is actively working on the model anymore, or
- Your data changes quickly enough that retraining more than ~1x / month gives a boost

A baseline: periodic retraining

A baseline: periodic retraining

1. Logging	• Log everything
2. Curation	• Sample uniformly at random to give you the max data points you can handle • Label them if applicable using Scale
3. Trigger	• Retrain periodically (e.g., once a week)
4. Dataset formation	• Use a rolling window of data (e.g., all data curated in the past month)
5. Offline testing	• Compute test-set accuracy after each training • Either set a threshold, or manually review and spot check each time
6. Online testing	• Do spot evaluations of the deployed model

Where will this fail?

- **High volume.** You might not be able to log or train on all of your data
- **Long tail.** Sampling uniformly misses “high signal” data like edge cases
- **Human-in-the-loop.**
  - Labeling is too expensive
  - You need custom labeling rules (e.g., medicine)
  - Labeling is part of the “product” (e.g., content moderation)
- **High cost of retraining.** Retraining periodically on a rolling window leaves $$ on the table
- **Rapidly-evolving data.** You can partially solve this by increasing the retraining frequency. Tradeoff: cost and evaluation burden
- **High cost of bad predictions.** Retraining introduces risk. The more you retrain and more sensitive you are to failures, the more thoughtful you need to be on eval.

Iterating on your retraining strategy

TL/DR: monitoring / debugging and retraining are two sides of the same coin

- Monitor metrics that actually matter, use the rest for debugging
- Debugging investigations lead to changes in strategy: retraining triggers, offline tests, sampling strategies, observability, etc
- As you get more confident in your monitoring, you can introduce more automation

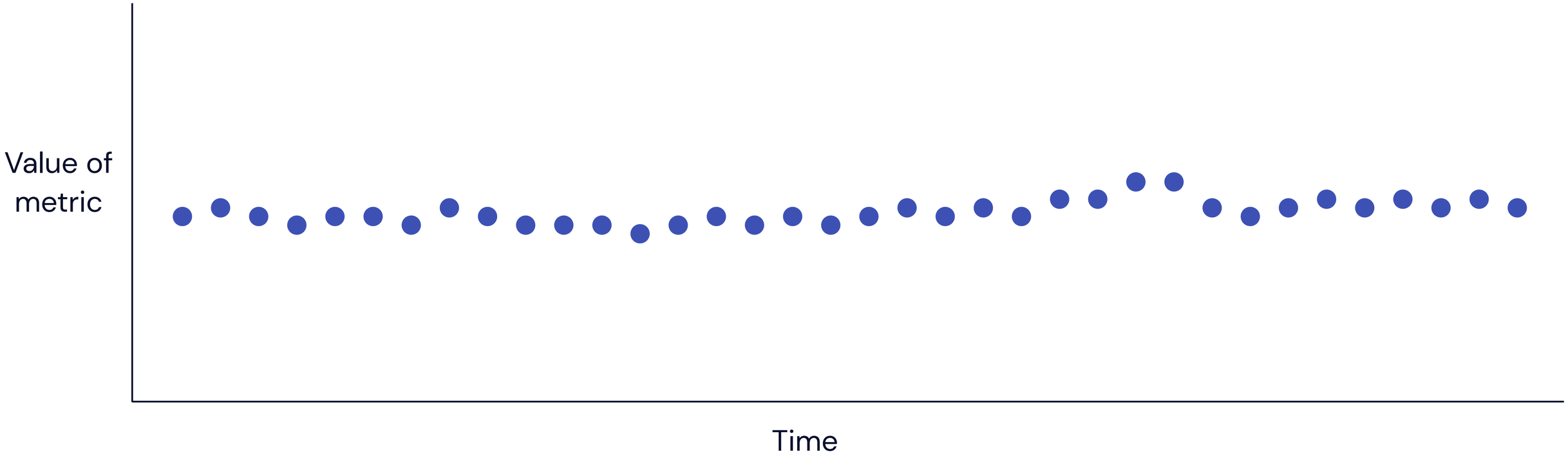
0. Monitoring & debugging machine learning models in production

Monitoring TL/DR

- No real standards / best practices here yet, and a lot of bad advice out there
- Monitor what matters & what breaks empirically
- Compute other signals too, but use them for observability & debugging

https://gantry.io/blog/youre-probably-monitoring-your-models-wrong/

The monitoring problem



Is this bad, or ok?

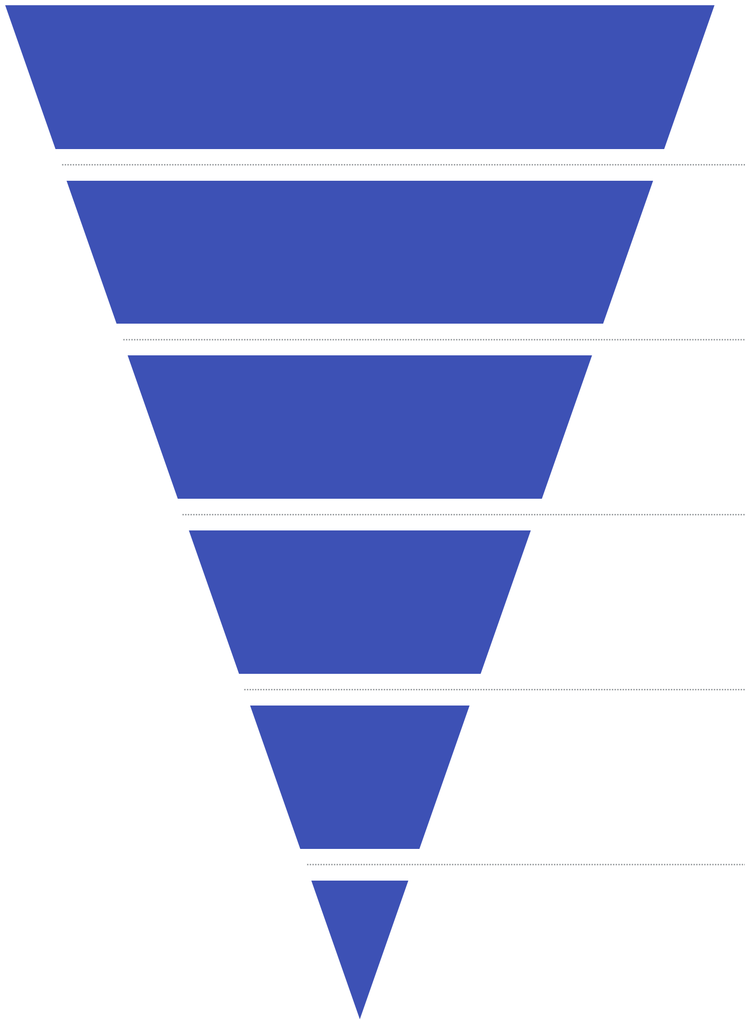
Outline

- What metric(s) to monitor
- How to tell if those metrics are “bad”
- Tools for monitoring

Outline

- **What metric(s) to monitor**
- How to tell if those metrics are “bad”
- Tools for monitoring

Most valuable signals to monitor



Outcomes and feedback from users

Model performance metrics

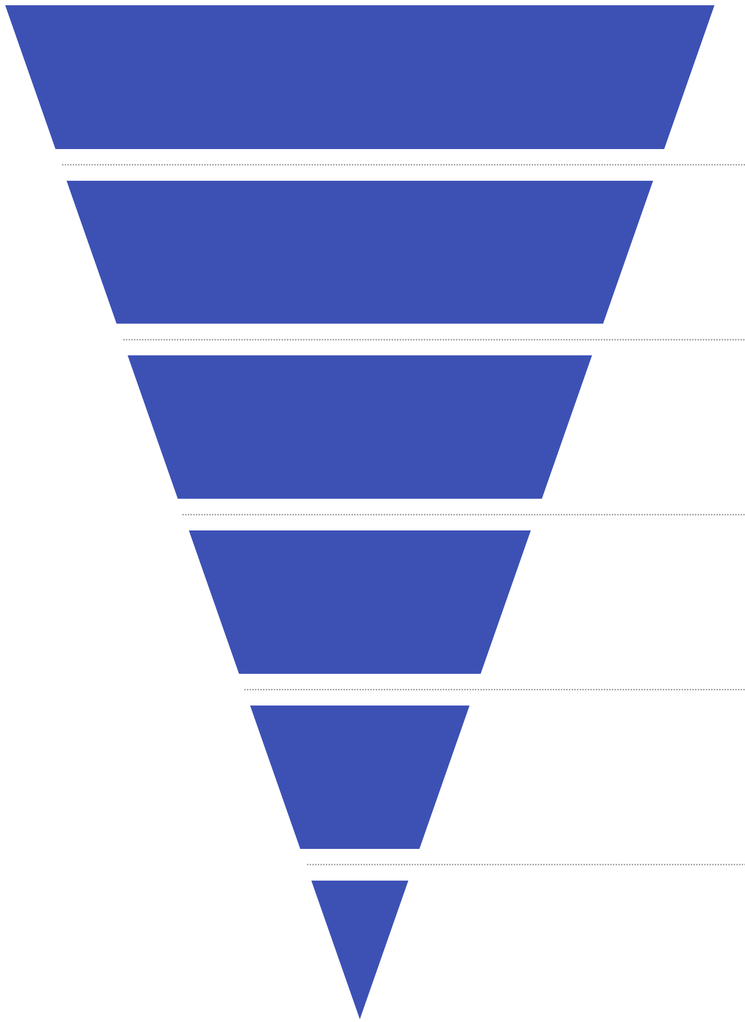
Proxy metrics

Data quality

Distribution drift

System metrics

Most valuable signals to monitor



Outcomes and feedback from users

Model performance metrics

Proxy metrics

Data quality

Distribution drift

System metrics

Outcomes & user feedback are *by far*
most useful signal to look at

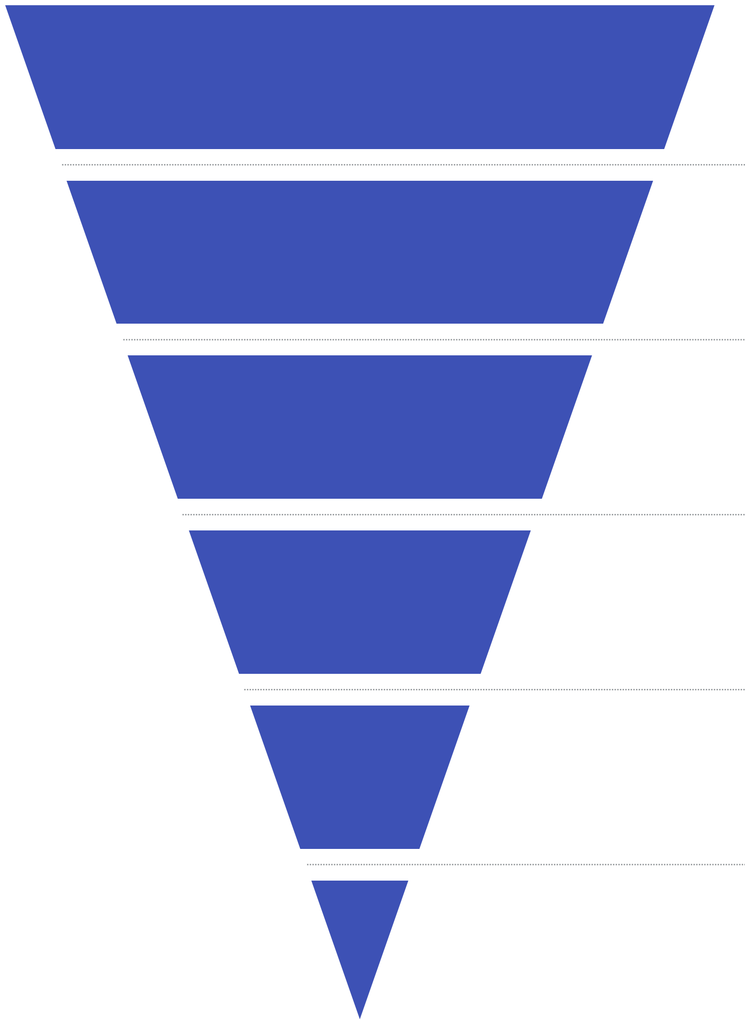
Capturing outcomes & user feedback

- No one-size fits all way to do this: depends on the product you are building
- Examples
  - Clicking on a recommendation
  - Churning from the product
  - Giving a generation a “thumbs up”
  - Flagging a generation as “offensive”
  - Correcting a classification by changing
  - Verifying if two people in a photo are the same
  - Intervening with autopilot

More in ML product management lecture

1 Lina Weichbrodt: What I learned from monitoring more than 30 Machine Learning Use Cases (https://www.youtube.com/watch?v=wWxqnZb-LSk)

Most valuable signals to monitor



Outcomes and feedback from users

Model performance metrics

Proxy metrics

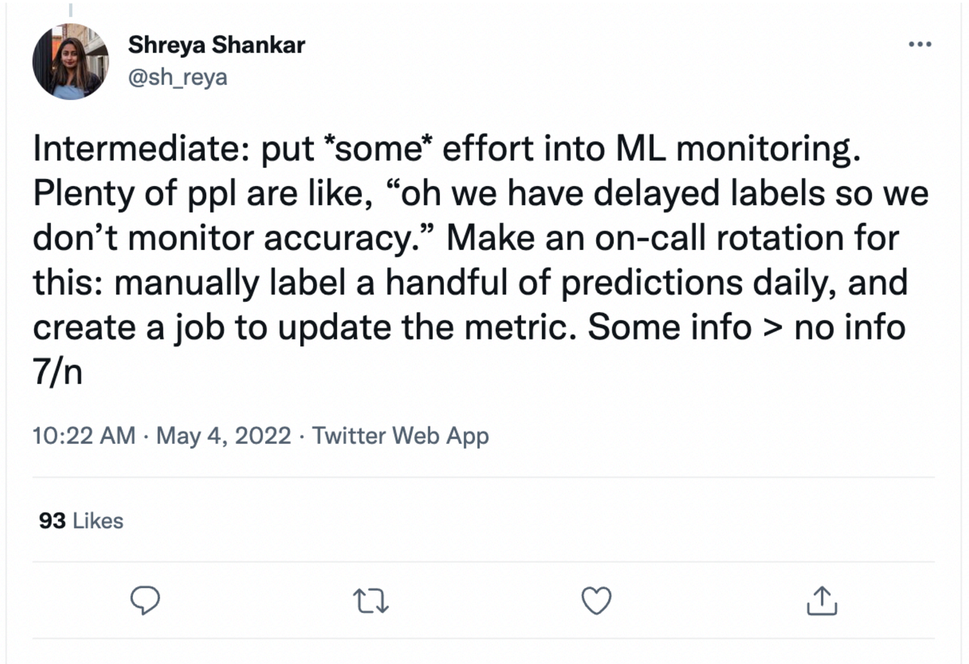
Data quality

Distribution drift

System metrics

Monitoring model performance metrics

- Worse than feedback metrics b/c of ***loss mismatch***
  - Common experience: improving model performance leads to same or worse outcome
- Just label some production data each day
  - Doesn't have to be a ton: have a label party!



Loss mismatch

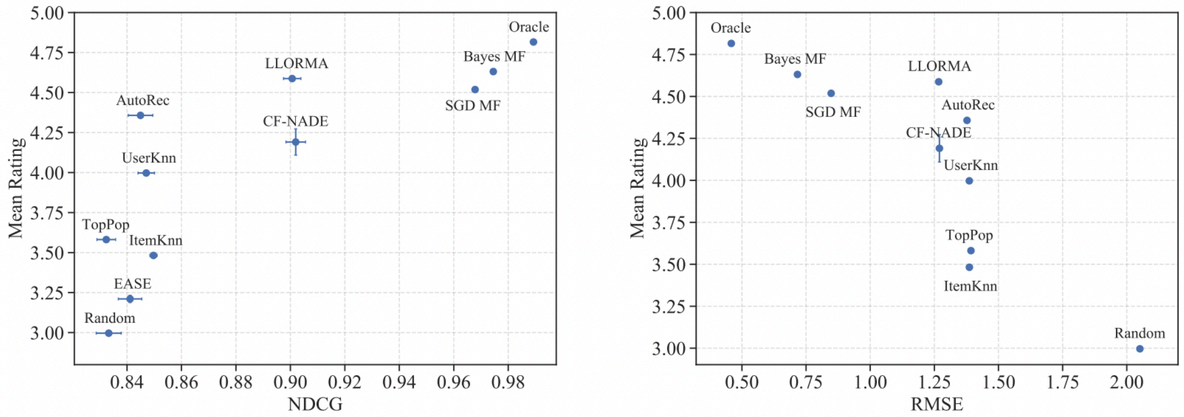
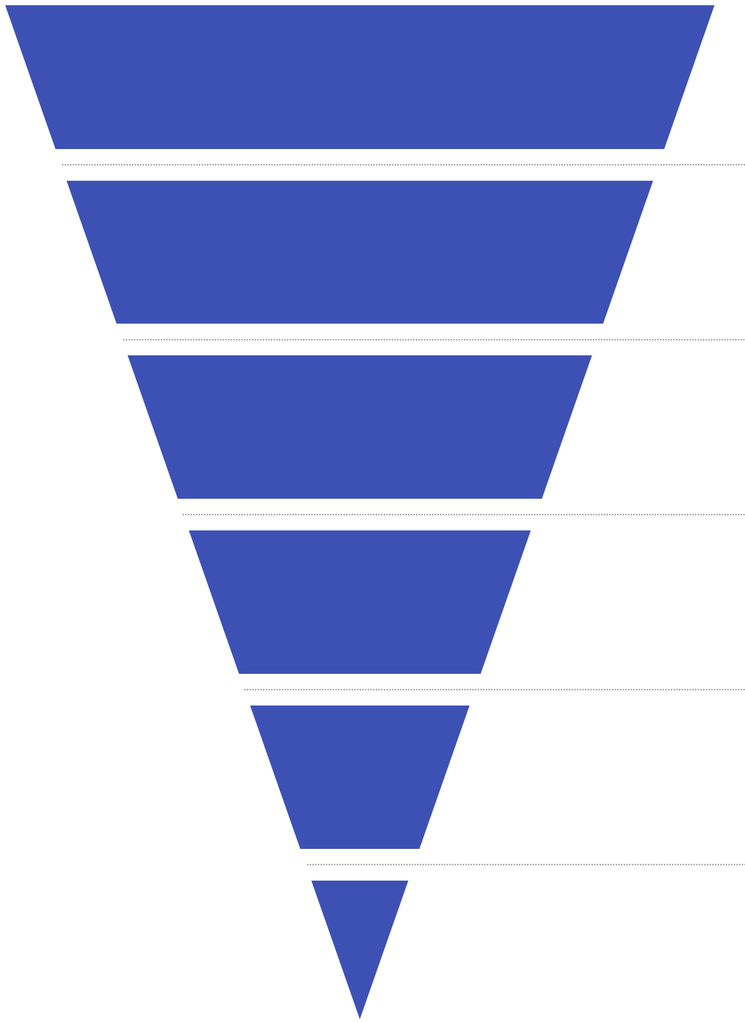


Figure 1: Left: The nDCG@20 plotted against the mean user ratings of all recommended items on the `topics-dynamic` environment. Right: The RMSE plotted against the mean user ratings. nDCG and RMSE are averaged across 5 folds on the offline dataset associated with the environment, user ratings are averaged across 10 trials. Each point represents a single model evaluation with error bars representing 95% confidence intervals.

metrics. Figure 1 shows such a comparison on the `topics-dynamic` environment. While there is a strong correlation between nDCG and mean user rating, we note that improvements in nDCG past a certain point suffer from diminishing improvement in mean user ratings. We examine these effects in more details in Section 5.

Most valuable signals to monitor



Outcomes and feedback from users

Model performance metrics

Proxy metrics

Data quality

Distribution drift

System metrics

Proxy metrics

- Metrics that are **correlated with bad model performance**
- Mostly domain-specific, e.g.,
  - Repetitive outputs (like “hi hi hi hi”) are bad for language models (but common)
  - Toxic outputs are for language models are bad
  - Share of personalized responses for personalization algorithm[1]
- Edge cases can be a good proxy metric[2]

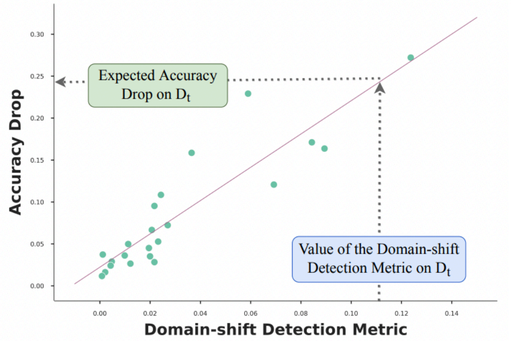
1 Lina Weichbrodt: What I learned from monitoring more than 30 Machine Learning Use Cases (https://www.youtube.com/watch?v=wWxqnZb-LSk)
2 We’ll come back to this!

Some academic content follows...

Toward universal proxy metrics

To Annotate or Not?
Predicting Performance Drop under Domain Shift

Hady Elsahar and **Matthias Gallé**
NAVER LABS Europe
{hady.elsahar,matthias.galle}@naverlabs.com



Learning Prediction Intervals for Model Performance

Benjamin Elder,[1] **Matthew Arnold,** [1] **Anupama Murthi** [1] **Jiri Navratil** [1]
[1] IBM Research
benjamin.elder@ibm.com, marnold@us.ibm.com, anupama.murthi@ibm.com, jiri@us.ibm.com

3 Method

Our approach estimates prediction intervals for the performance of a black-box classification model on a pool of unlabeled data. First, we use meta-modeling to predict the accuracy of the base classification model (*performance prediction*). Second, a pre-trained *uncertainty model* is used to estimate a prediction interval, which describes the probable range for the true value of the accuracy.

LEVERAGING UNLABELED DATA TO PREDICT OUT-OF-DISTRIBUTION PERFORMANCE

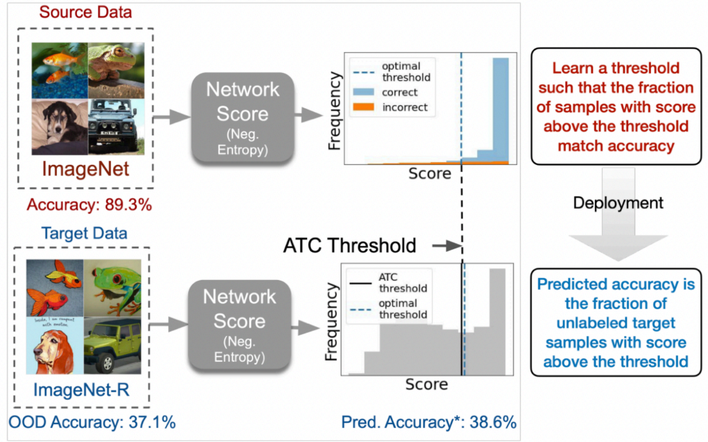
Saurabh Garg*
Carnegie Mellon University
sgarg2@andrew.cmu.edu

Sivaraman Balakrishnan
Carnegie Mellon University
sbalakri@andrew.cmu.edu

Zachary C. Lipton
Carnegie Mellon University
zlipton@andrew.cmu.edu

Behnam Neyshabur
Google Research, Blueshift team
neyshabur@google.com

Hanie Sedghi
Google Research, Brain team
hsedghi@google.com

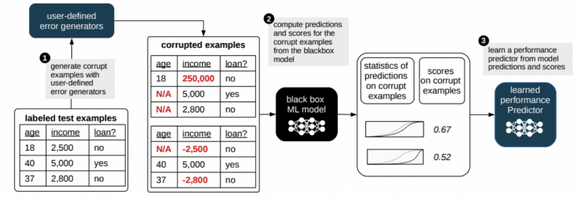


Learning to Validate the Predictions of Black Box Classifiers on Unseen Data

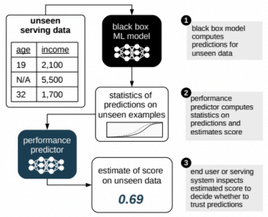
Sebastian Schelter
sebastian.schelter@nyu.edu
New York University

Tammo Rukat
tammruka@amazon.com
Amazon Research

Felix Biessmann
fbiessmann@beuth-hochschule.de
Beuth University Berlin



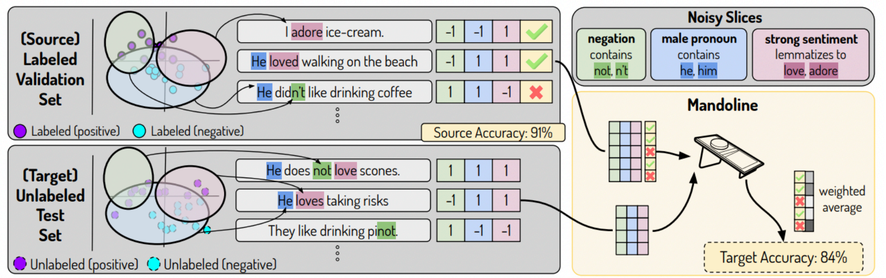
(a) Learning a performance predictor for a pretrained black box classifier, using synthetically corrupted data to explore how data errors impact the resulting prediction quality.



(b) Using the performance predictor to estimate the accuracy on unseen serving data.

MANDOLINE: Model Evaluation under Distribution Shift

Mayee Chen [*1] **Karan Goel** [*1] **Nimit Sohoni** [*2] **Fait Poms** [1] **Kayvon Fatahalian** [1] **Christopher Ré** [1]



Is it always possible to approximate performance accurately?

LEVERAGING UNLABELED DATA TO PREDICT OUT-OF-DISTRIBUTION PERFORMANCE

Saurabh Garg*
Carnegie Mellon University
sgarg2@andrew.cmu.edu

Sivaraman Balakrishnan
Carnegie Mellon University
sbalakri@andrew.cmu.edu

Zachary C. Lipton
Carnegie Mellon University
zlipton@andrew.cmu.edu

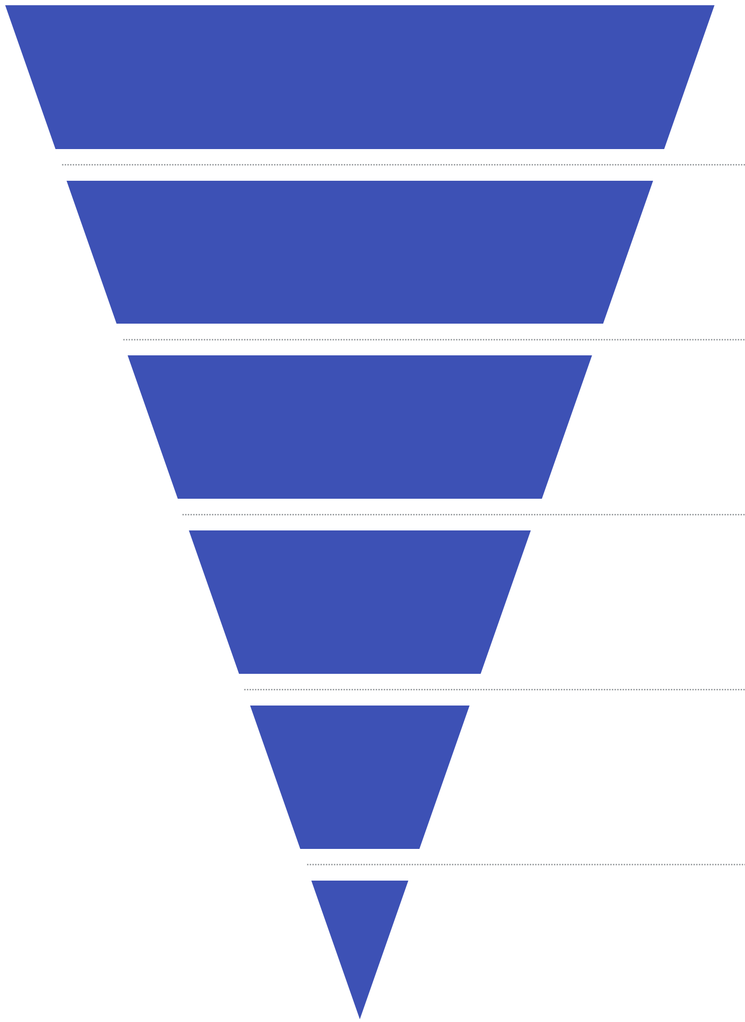
Behnam Neyshabur
Google Research, Blueshift team
neyshabur@google.com

Hanie Sedghi
Google Research, Brain team
hsedghi@google.com

Corollary 1. *Absent assumptions on the classifier f, no method of estimating accuracy will work in all scenarios, i.e., for different nature of distribution shifts.*

...back to scheduled programming

Most valuable signals to monitor



Outcomes and feedback from users

Model performance metrics

Proxy metrics

Data quality

Distribution drift

System metrics

Data quality testing

- A set of rules that measure the quality of your data:
  - **Accuracy**: How well does a piece of information reflect reality?
  - **Completeness**: Does it fulfill your expectations of what's comprehensive?
  - **Consistency**: Does information stored in one place match relevant data stored elsewhere?
  - **Timeliness**: Is your information available when you need it?
  - **Validity (aka Conformity)**: Is the information in a specific format, type, or size? Does it follow business rules/best practices?
  - **Integrity**: Can different data sets be joined correctly to reflect a larger picture? Are relations well defined and implemented?

https://lakefs.io/data-quality-testing/

Examples

- Metadata testing
  - Does it have the right schema?
- Accuracy testing
  - **Fact-checking**, e.g., are these names present in the dictionary?
  - **Set-level sanity**, e.g., is most of the data in the expected range?
  - **History-based set-level sanity**, e.g., are the number of records anomalous today?

https://lakefs.io/data-quality-testing/

Data problems are the most common issue in practice

How ML Breaks

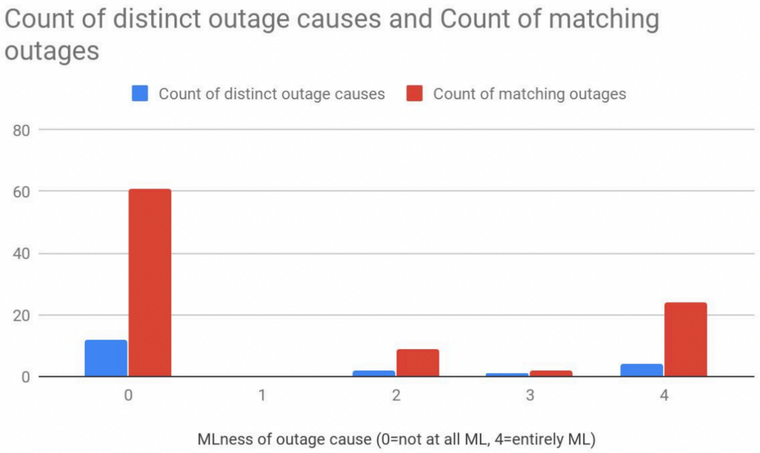
Fifteen years of ML production pipeline outages and insight

dannyp@google.com, tmu@google.com

Finding #1: Most outages and their causes aren't ML

Failures are not characteristic of ML

- The plurality of our distinct causes, accounting for the majority of our outages, were rated as not at all characteristic of ML.
- A subset of our outages were rated as being purely characteristic of ML. The middle ground is rare.

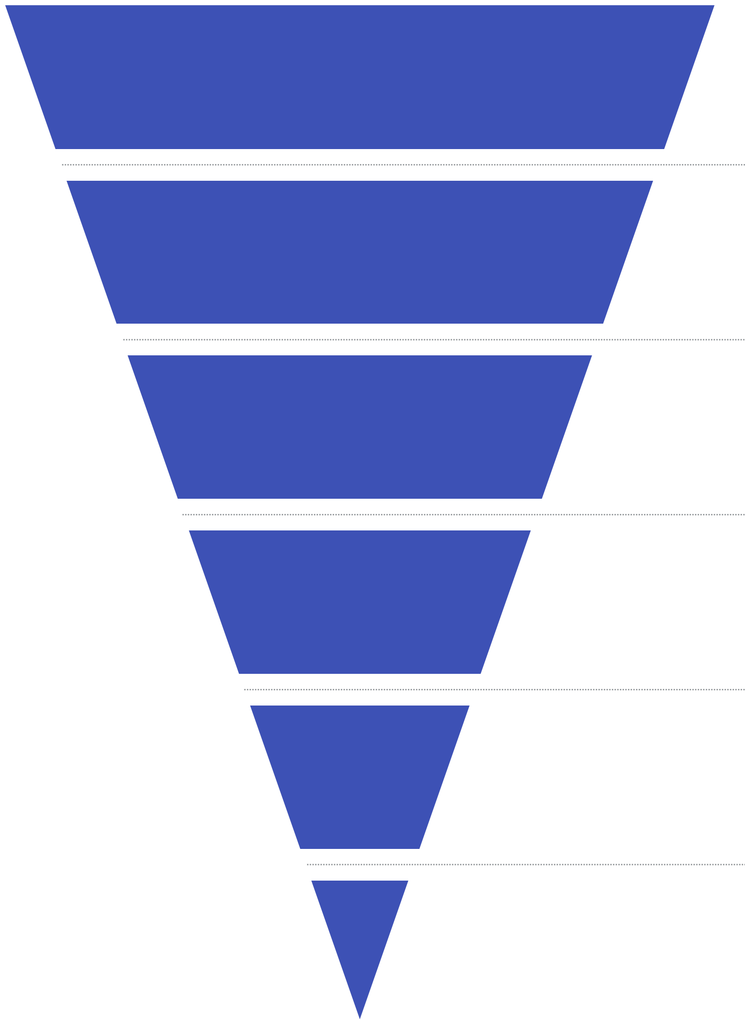


Another Example:

A boring but totally common failure:

- Data input processing pipeline joins data from a structured data source.
- In order to save resources in the original location we copied the structured data source to a new location
- The pipeline lacked permissions to read the data source in the new location, failing to join the contents of that data source.
- The entire pipeline lost the ability to process new data.

Most valuable signals to monitor



Outcomes and feedback from users

Model performance metrics

Proxy metrics

Data quality

Distribution drift

System metrics

Distribution drift

- Why monitor distribution drift?
- Types of distribution drift
- How to measure it?
- What metrics to use?
- Dealing with high-dimensional data
- Cons of looking at distribution drift

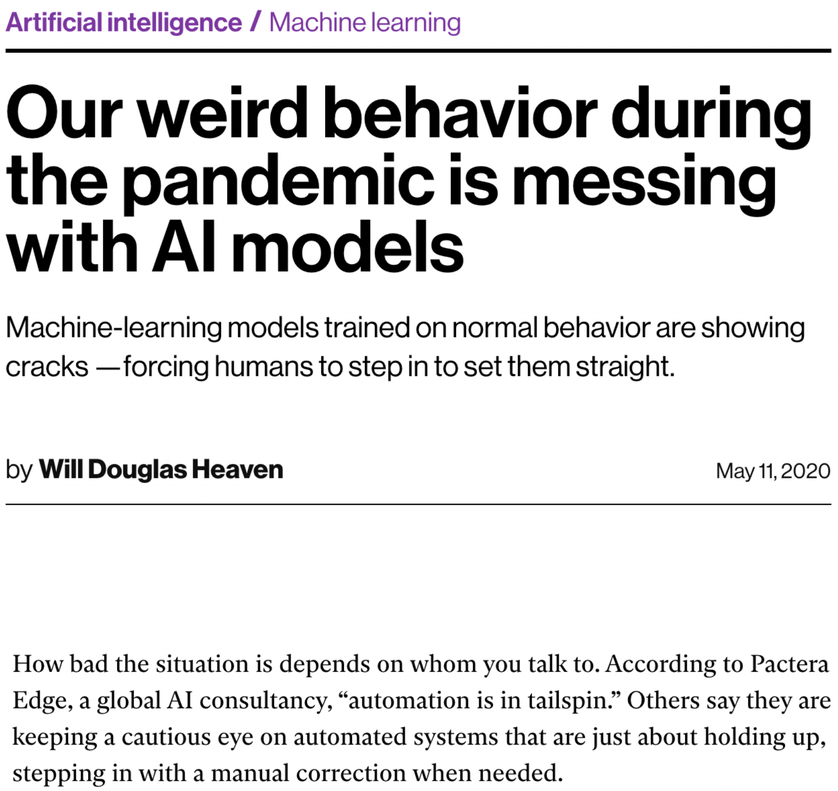
Distribution drift

- **Why measure distribution drift?**
- Types of distribution drift
- How to measure it?
- What metrics to use?
- Dealing with high-dimensional data
- Cons of looking at distribution drift

Why measure distribution drift?

Your model's performance is only guaranteed on **data sampled from the *same distribution*** as it was trained on

This can have a huge impact in practice



Artificial intelligence / Machine learning

Our weird behavior during the pandemic is messing with AI models

Machine-learning models trained on normal behavior are showing cracks — forcing humans to step in to set them straight.

by **Will Douglas Heaven** May 11, 2020

How bad the situation is depends on whom you talk to. According to Pactera Edge, a global AI consultancy, "automation is in tailspin." Others say they are keeping a cautious eye on automated systems that are just about holding up, stepping in with a manual correction when needed.

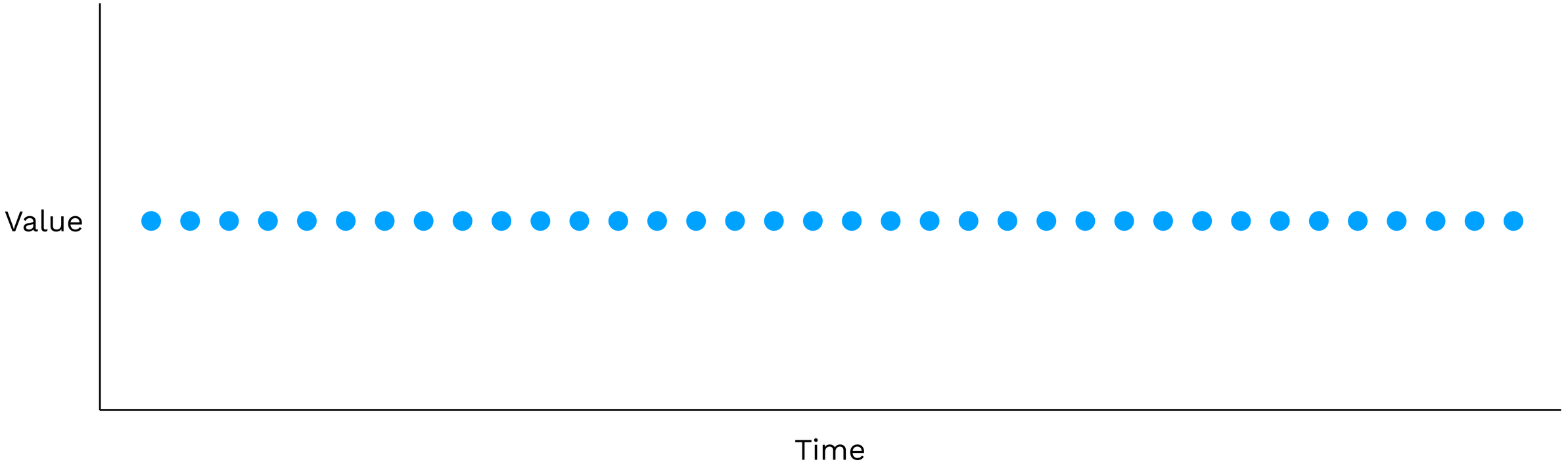
Story from global e-commerce company:

- Bug in retraining pipeline caused recommendations not to be updated for new users
- New users got the same stale recommendations for weeks before the bug was caught
- Estimated $Ms in revenue lost

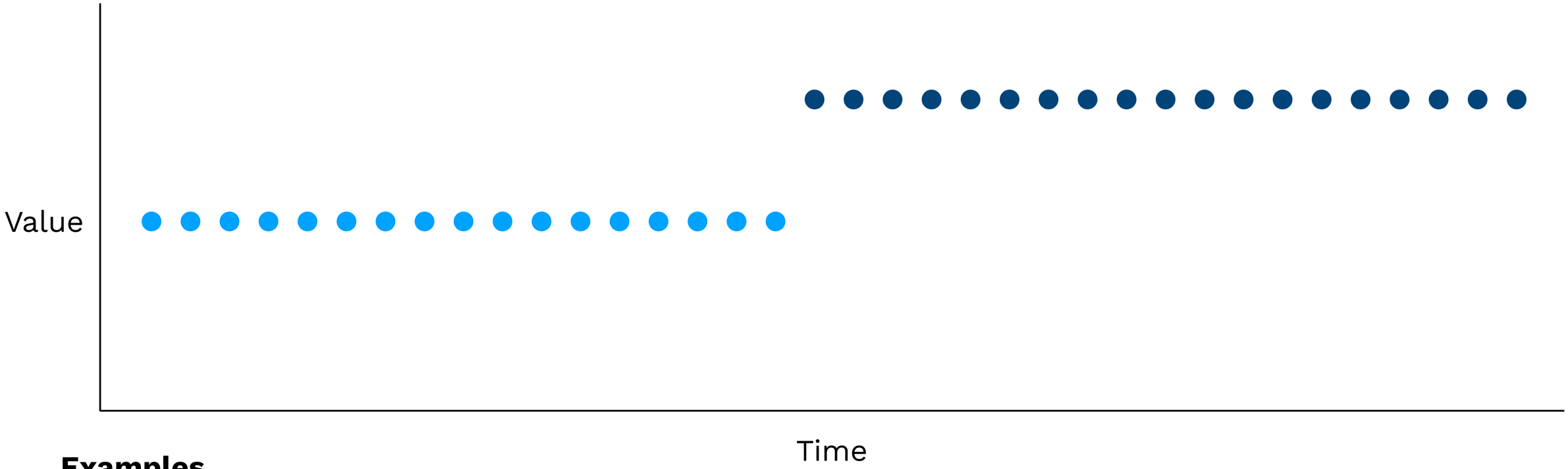
Distribution drift

- Why measure distribution drift?
- **Types of distribution drift**
- How to measure it?
- What metrics to use?
- Dealing with high-dimensional data
- Cons of looking at distribution drift

Types of data drift



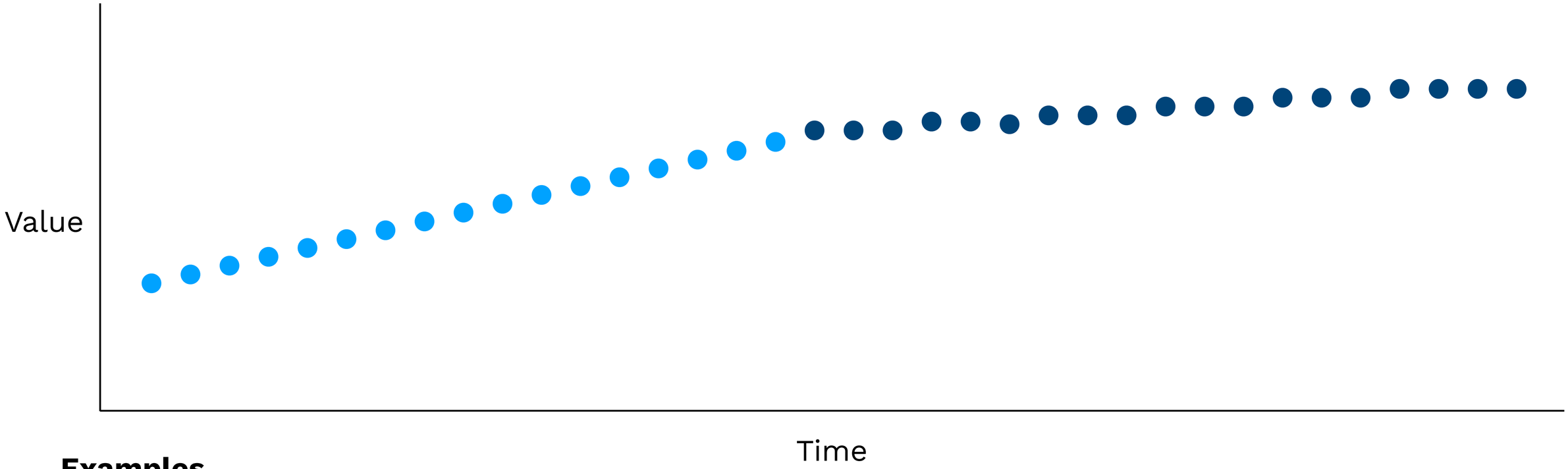
Types of data drift



Examples

- Model deployed in a new domain
- A bug is introduced in the preprocessing pipeline
- Big external shift, like Covid

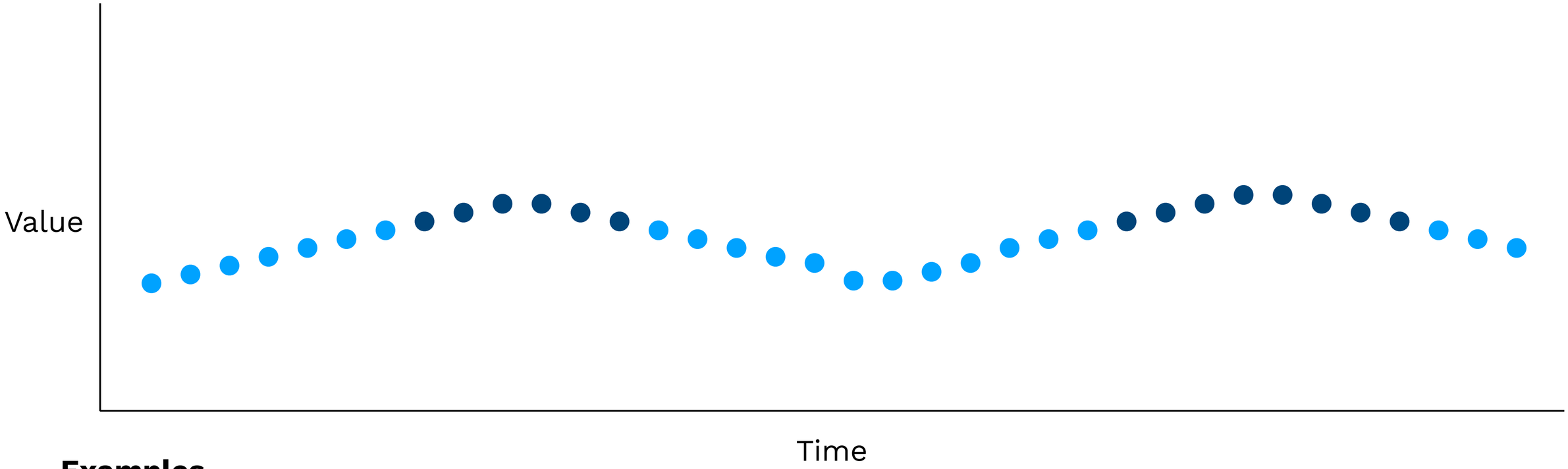
Types of data drift



Examples

- Users preferences change over time
- New concepts get introduced to the corpus over time

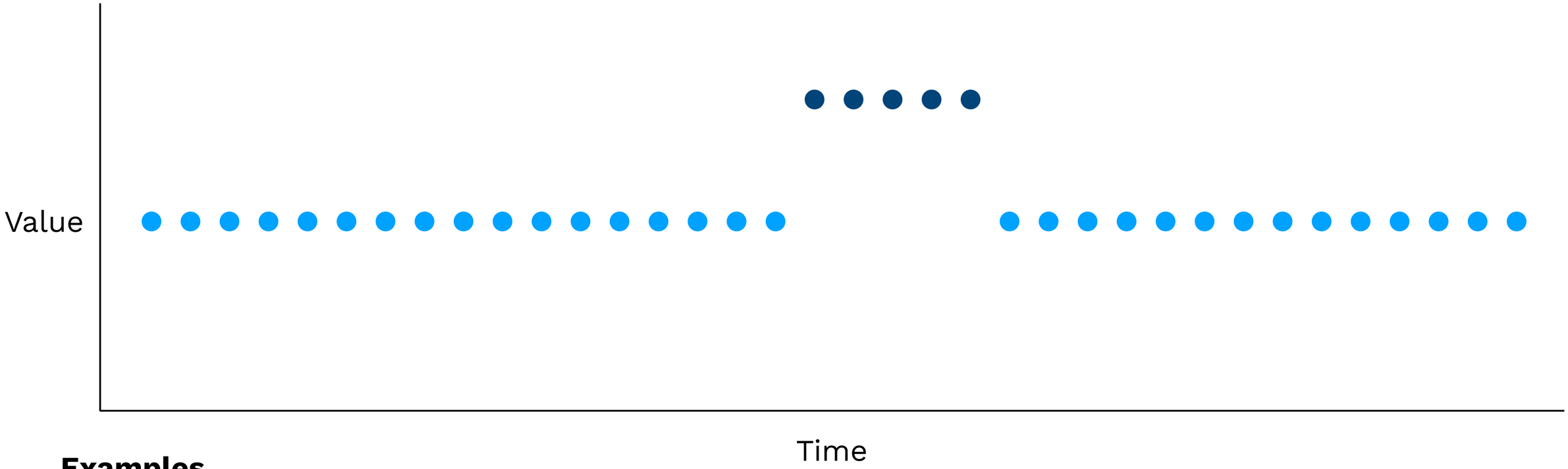
Types of data drift



Examples

- Users preferences are seasonal
- People in different timezones use your model differently

Types of data drift



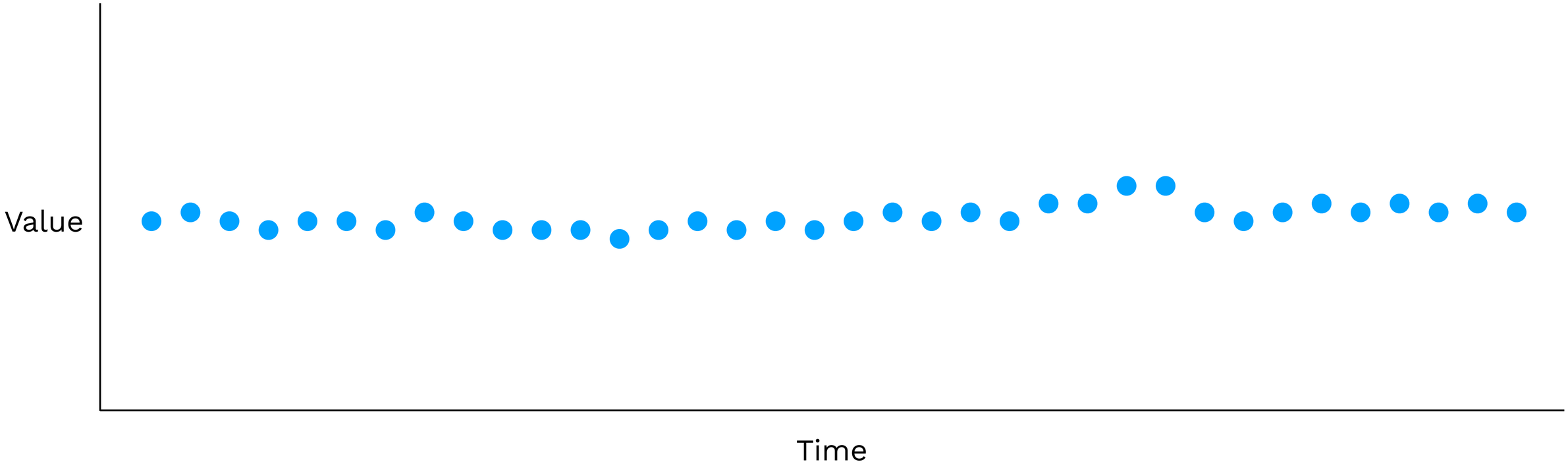
Examples

- Malicious user attacks your model
- A new user with different demographics tries your product and then churns
- Someone uses your product in a way it was not intended to be used

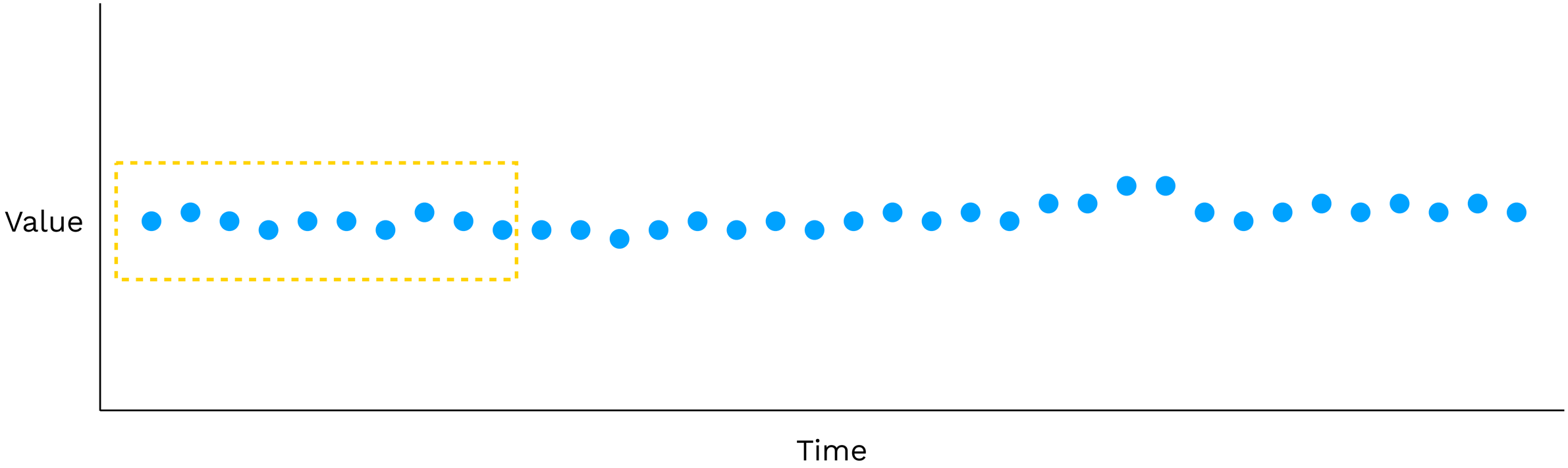
Distribution drift

- Why measure distribution drift?
- Types of distribution drift
- **How to measure it?**
- What metrics to use?
- Dealing with high-dimensional data
- Cons of looking at distribution drift

Measuring distribution change



Measuring distribution change



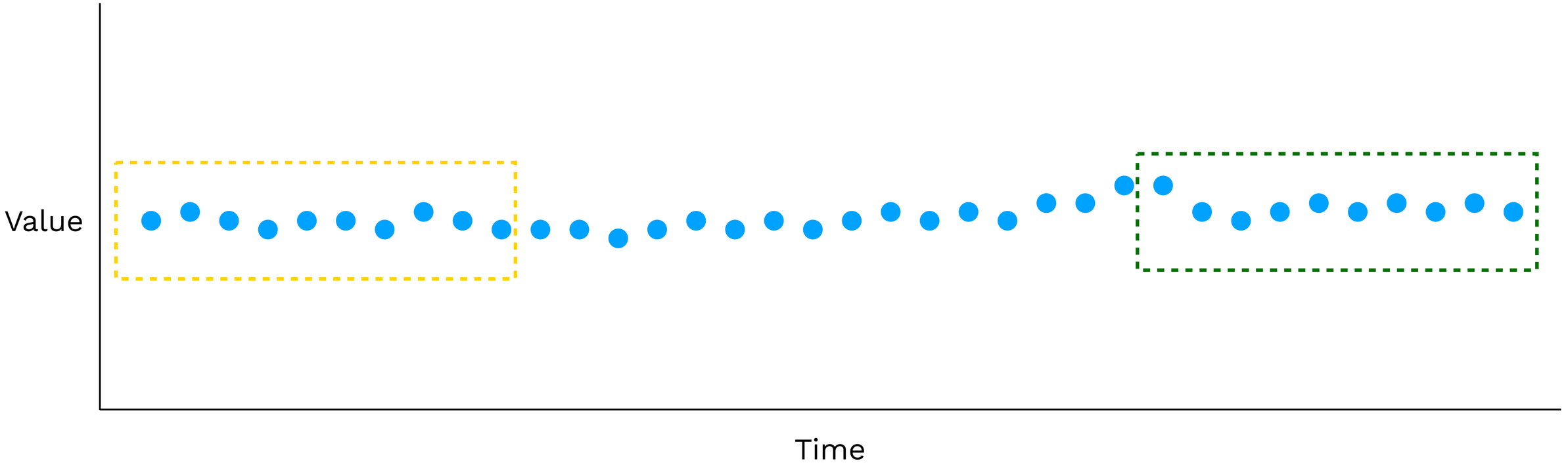
1. Select a window of “good” data to serve as a *reference*

Selecting a reference window

- You can use a fixed window of production data you believe to be healthy
- Some papers [1] advocate for using a sliding window of production data
- In practice, most of the time you probably should use your validation data as the reference

[1] Automatic Model Monitoring for Data Streams, Pinto et al. (https://arxiv.org/abs/1908.04240)

Measuring distribution change



2. Select a new window of data to measure distance on

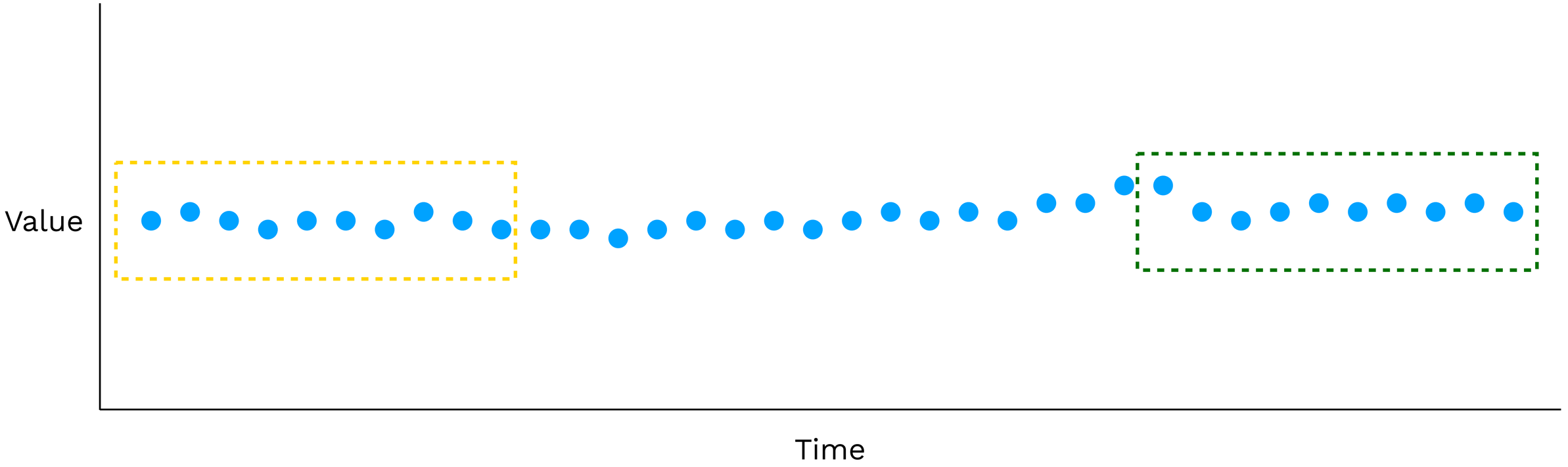
Selecting the measurement window

- Problem-dependent — not aware of a principled approach
- Pragmatic solution: pick one or several window sizes with a reasonable amount of data and slide them [1]
- Special case: window size = 1, i.e., outlier detection [2]

[1] To avoid recomputing metrics over and over again when you slide the window, it's worth looking into the literature on mergeable (quantile) sketching algorithms (https://www.sketchingbigdata.org/)

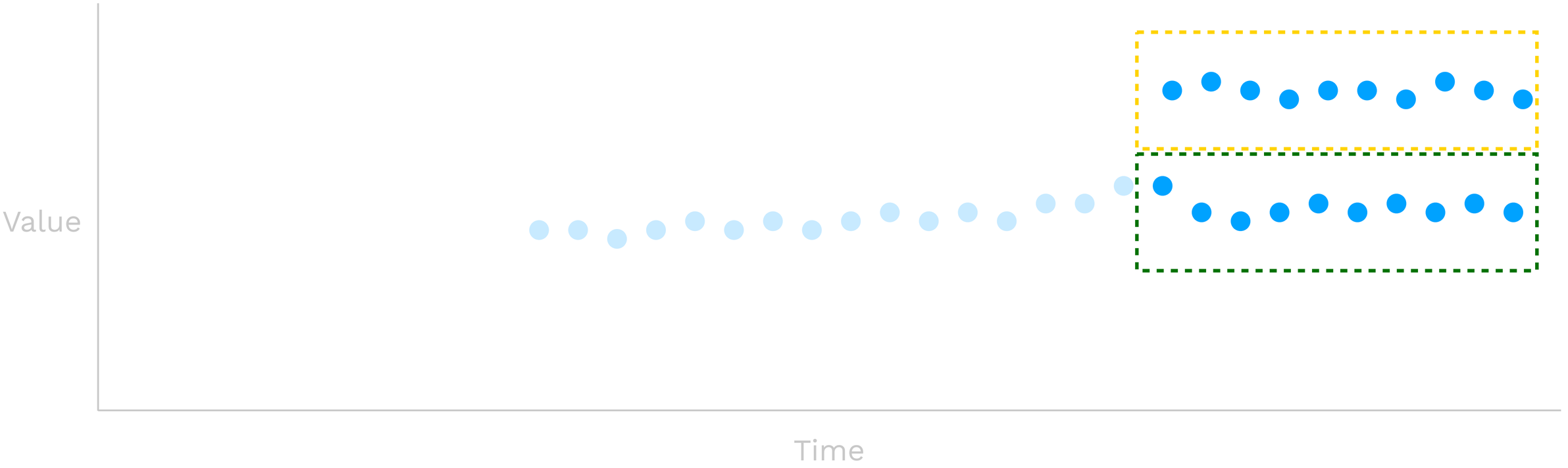
[2] E.g., check out https://pyod.readthedocs.io/en/latest/

Measuring distribution change



3. Compare the windows using a distance metric

Measuring distribution change

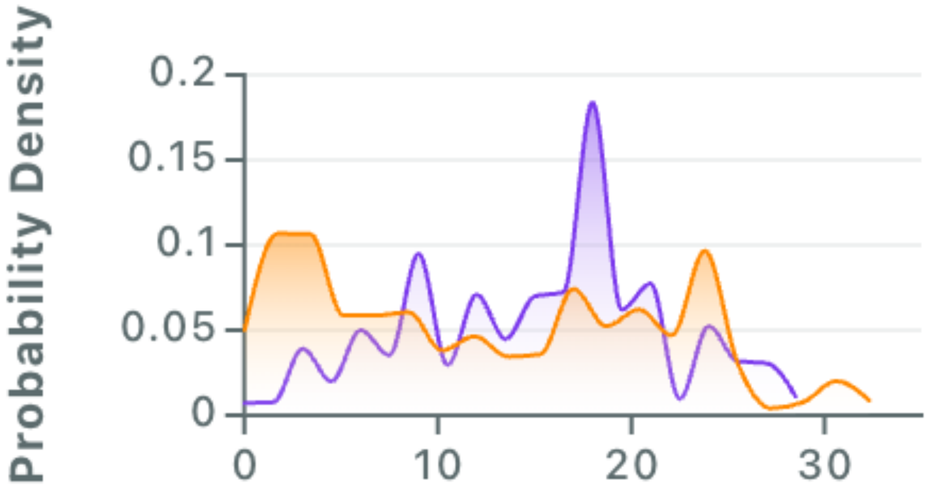


3. Compare the windows using a distance metric

Distribution drift

- Why measure distribution drift?
- Types of distribution drift
- How to measure it?
- **What metrics to use?**
- Dealing with high-dimensional data
- Cons of looking at distribution drift

Measuring distribution drift: 1D case



What's the distance between these distributions?

Options for distribution “distance” metrics

- Not-so-good options
  - Kullback-Leibler (KL) divergence
  - Kolmogorov-Smirnov (KS) test
- Sometimes-better options
  - Infinity norm or 1-norm of the diff between probabilities for each category
  - Earth-mover’s distance

https://gantry.io/blog/youre-probably-monitoring-your-models-wrong/

Distribution drift

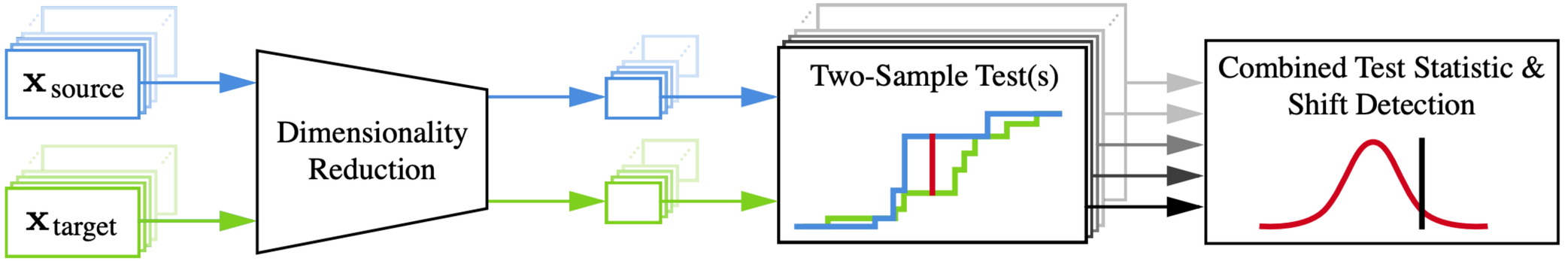
- Why measure distribution drift?
- Types of distribution drift
- How to measure it?
- What metrics to use?
- **Dealing with high-dimensional data**
- Cons of looking at distribution drift

What to do if you have many features?

- Measure drift on all of the features independently?
  - Be careful — *multiple hypothesis testing*[1]
  - Doesn't capture cross-correlation
- Measure drift on only the *important* features
  - Outputs are more useful in general than inputs
  - Can use *feature importance*[2]
- Multi-dimensional drift measurement with maximum mean discrepancy[3] or approximate earth-mover's distance[4]
  - Hard to interpret

1 https://multithreaded.stitchfix.com/blog/2015/10/15/multiple-hypothesis-testing/
2 https://christophm.github.io/interpretable-ml-book/feature-importance.html
3 https://jmlr.csail.mit.edu/papers/v13/gretton12a.html
4 https://arxiv.org/abs/1904.05877

Even higher dimensions: *projections*

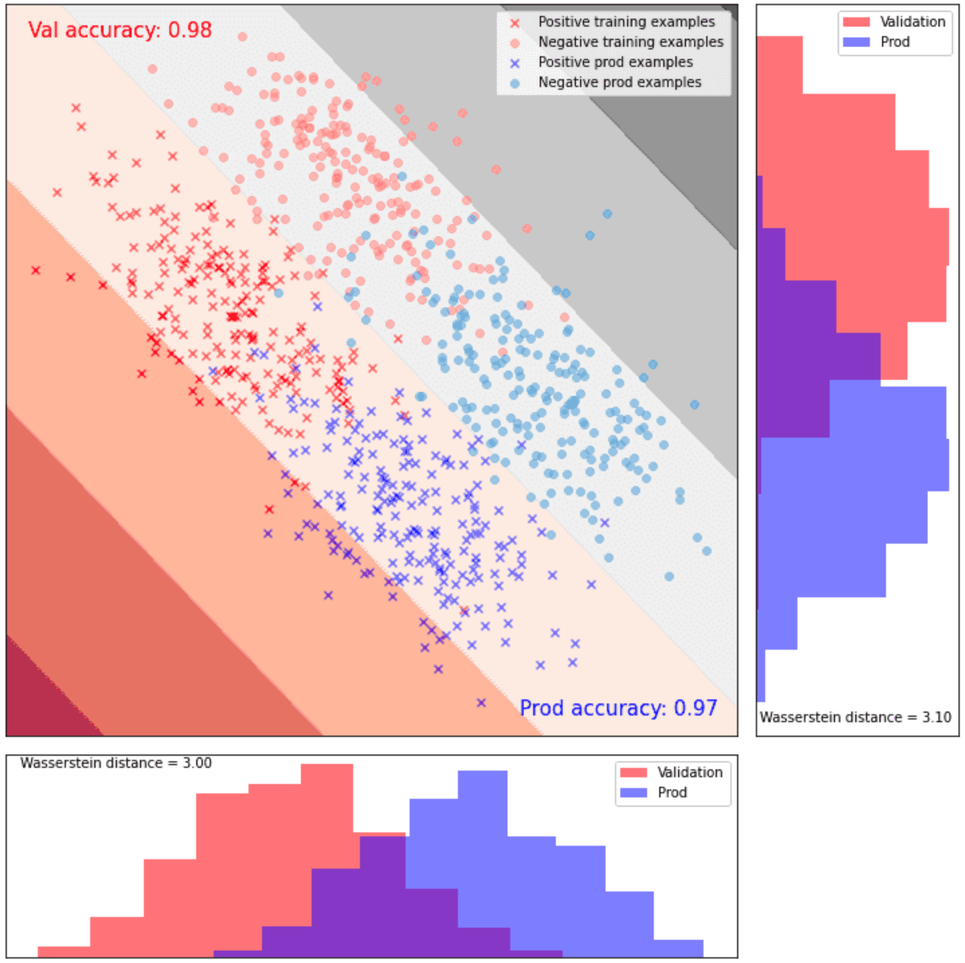


- Works for any data: images, text, etc
- Different kinds:
  - Analytical projections (e.g., mean pixel value, length of sentence, or any other function)
  - Random projections (e.g., linear)
  - Statistical projections (e.g., Autoencoder or other density model, T-SNE)

Distribution drift

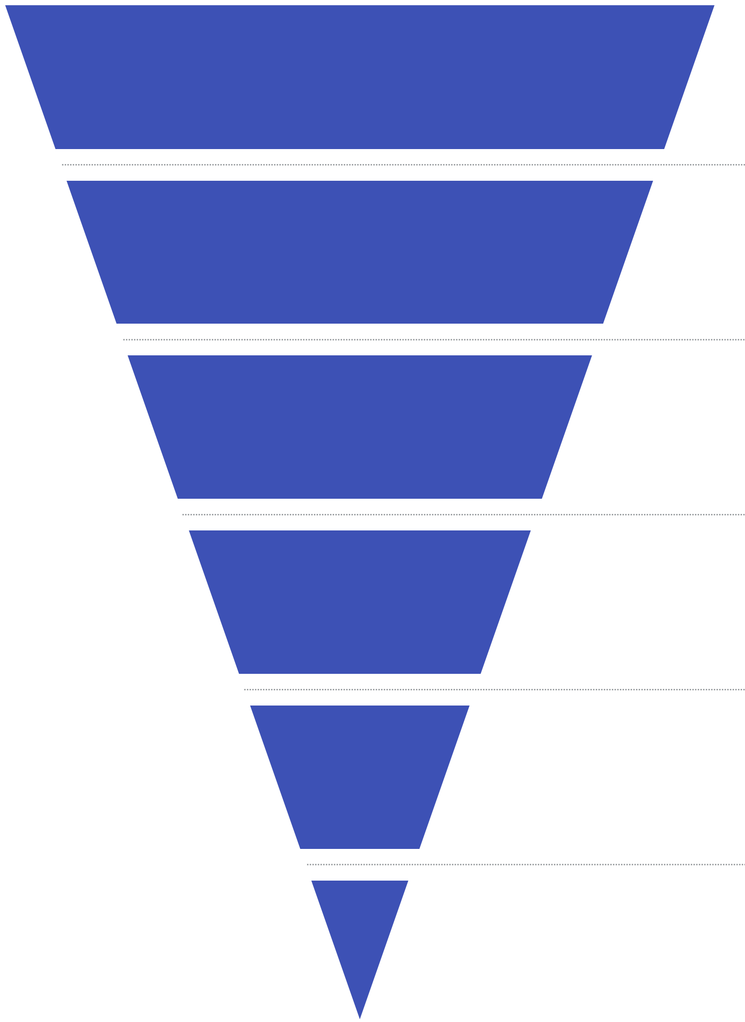
- Why measure distribution drift?
- Types of distribution drift
- How to measure it?
- What metrics to use?
- Dealing with high-dimensional data
- **Cons of looking at distribution drift**

Models are robust to some distribution shift



https://gantry.io/blog/youre-probably-monitoring-your-models-wrong/

Most valuable signals to monitor



Outcomes and feedback from users

Model performance metrics

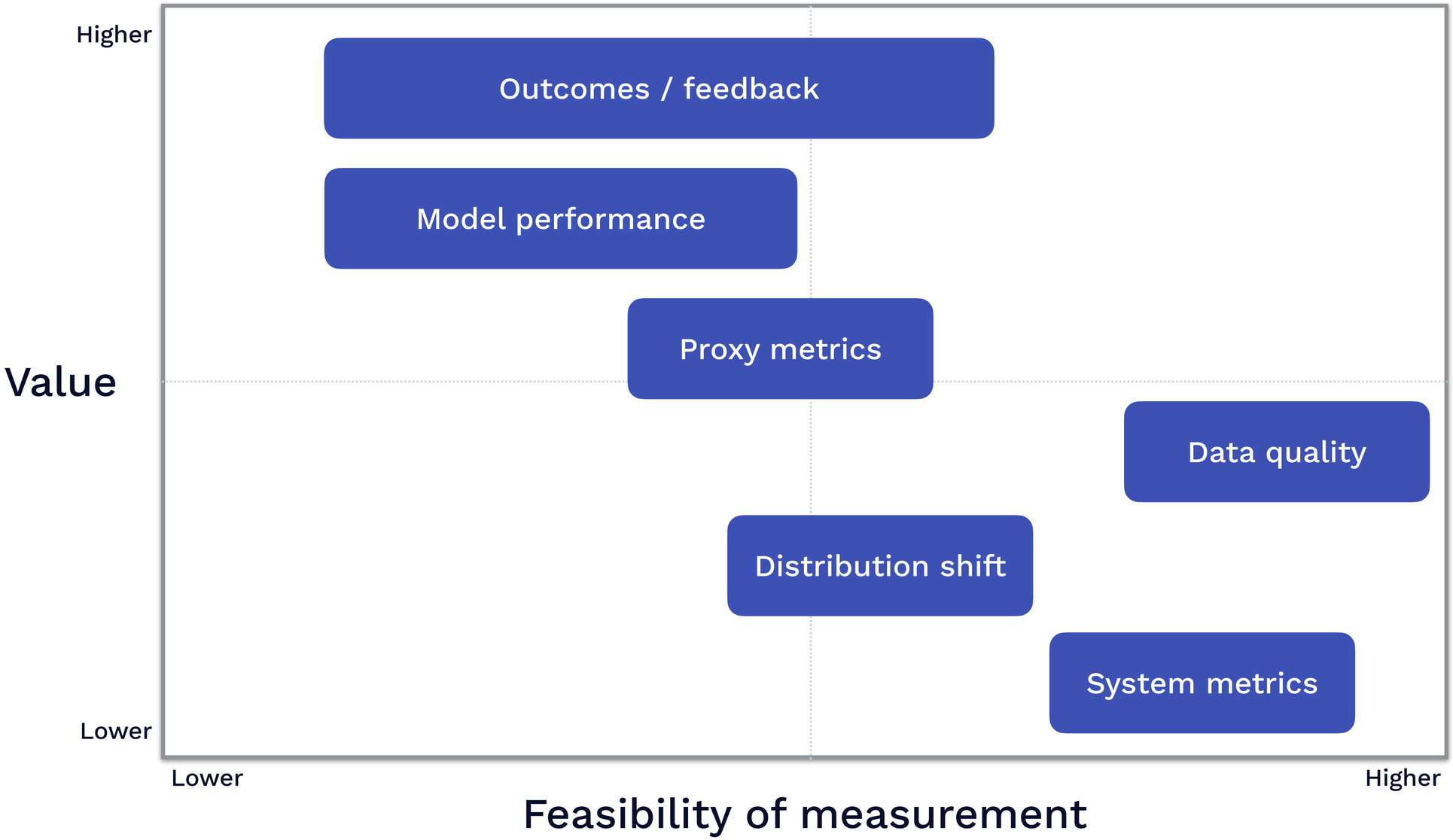
Proxy metrics

Data quality

Distribution drift

System metrics

What to monitor? Practical recommendation



What to monitor? Practical recommendation

- Basic data quality checks are zero-regret, especially if you are retraining your model
- Get *some way* to measure feedback, model performance, or proxy metrics, even if it's hacky or not scalable
- If your model produces low-dimensional outputs, monitoring those for distribution shift is also a good idea
- As you evolve your system, practice the *observability mindset*

Observability mindset

- Monitoring: known-unknowns
  - E.g., set alerts on a few key metrics
- Observability: unknown-unknowns
  - E.g., have the power to ask arbitrary questions about your system when it breaks (like, how does my accuracy differ by region?)
- Observability mindset means:
  - Keep around the context / raw data
  - Measure everything you can think of, but don't necessarily alert on it
  - Drift is a great example: very useful for debugging, less so for monitoring!

https://www.honeycomb.io/blog/observability-a-manifesto/

Go beyond aggregate metrics

- If your model is 99% accurate in aggregate, but only 50% accurate for your most important user, is it still “good”?
- Flag important subgroups / cohorts of data and alert on important metrics across them
  - E.g., categories you don’t want to be biased against
  - E.g., “important” categories of users
  - E.g., categories you might expect to perform differently on, like languages, regions, etc

Outline

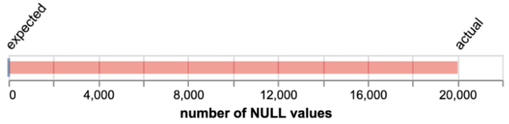
- What metric(s) to monitor
- **How to tell if those metrics are “bad”**
- Tools for monitoring

How to tell if your metrics are “bad”?

- Statistical tests (e.g., KS-**Test**) **<- Not recommended!**
  - Typically, return a p-value for likelihood that the data distributions are **not the same**
  - When you have a lot of data, you **will** get very small p-values for small shifts.

1. Fixed rule

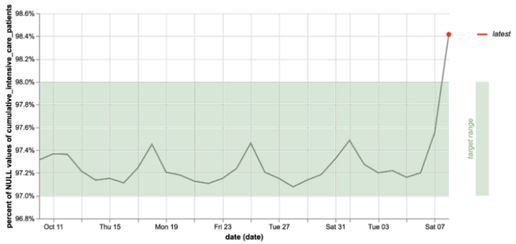
There are never any NULL values



Alert whenever a condition is not perfectly satisfied

2. Specified range

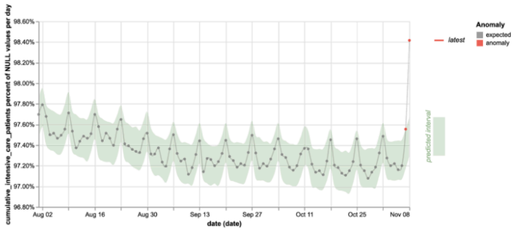
The NULL value % is within 97% and 98%



Alert whenever a value falls outside of a predetermined range specified by the user

3. Predicted range

The NULL value % is not anomalous



Alert whenever a value falls outside of the predicted range from an automated model

4. Unsupervised detection

This pattern is new and concerning

Alert whenever important data changes in an unusual way using unsupervised machine learning algorithms

Used most in practice

https://blog.anomalo.com/dynamic-data-testing-f831435dba90

Outline

- What metric(s) to monitor
- How to tell if those metrics are “bad”
- **Tools for monitoring**

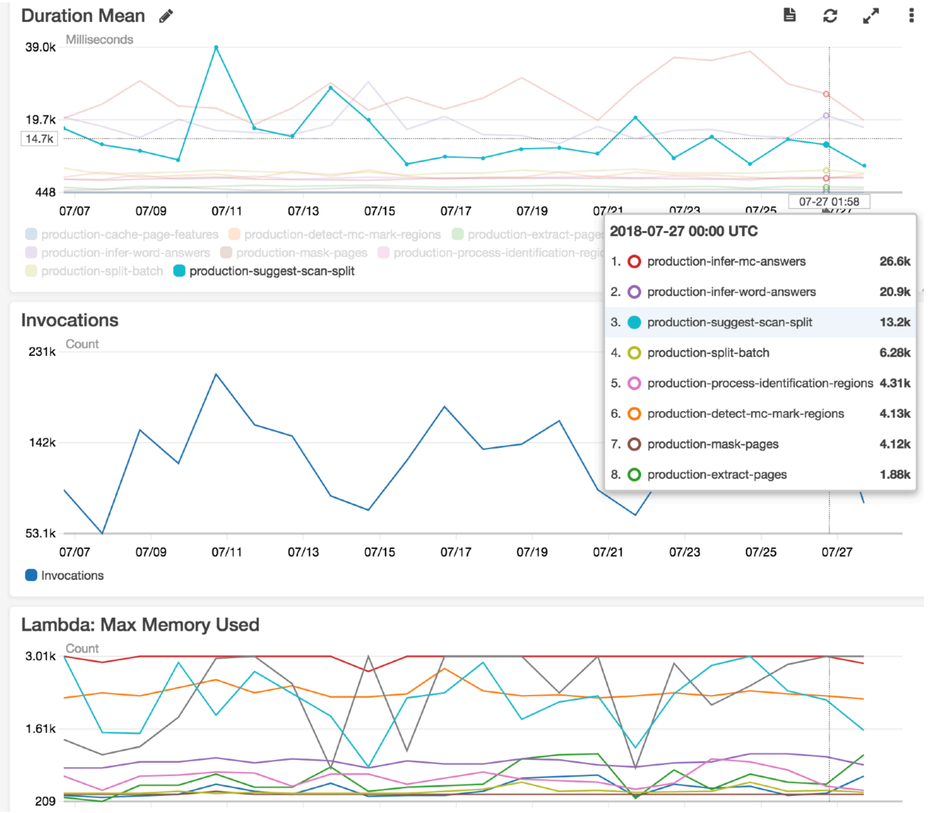
System monitoring tools

DATADOG

honeycomb.io

New Relic.

Amazon CloudWatch



- Alarms for when things go wrong, and records for tuning things.
- Cloud providers have decent monitoring solutions.
- Anything that can be logged can be monitored

Can we use system monitoring tools for ML metrics?

- It’s feasible but highly painful
- Lots of technical reasons why — lack of sliding windows, difficulty handling cross-component work (like combining predictions & feedback), etc

https://www.shreya-shankar.com/rethinking-ml-monitoring-3/

OSS ML monitoring

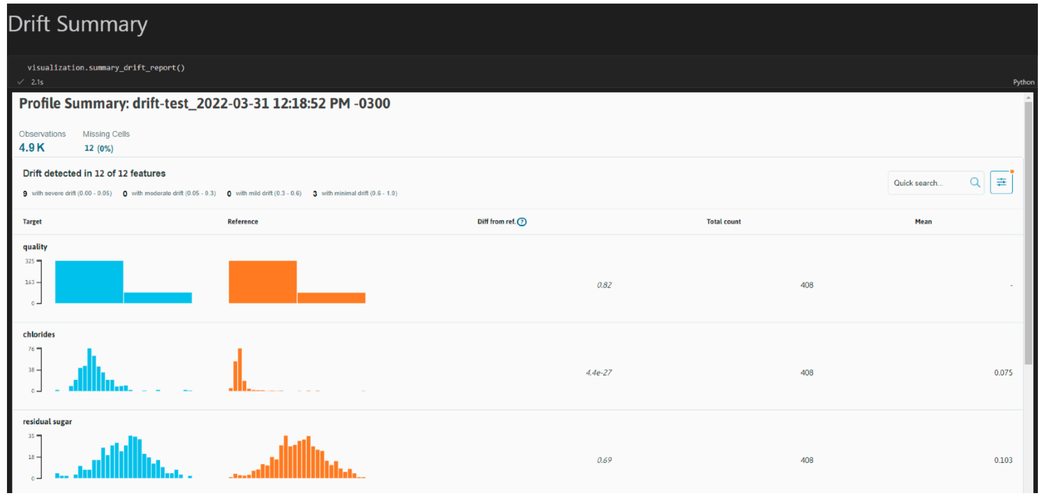
Evidently AI

Search

	Feature	Type	Reference Distribution	Current Distribution	Data drift	P-Value for Similarity Test
>	worst concave points	num			Detected	0.000002
>	worst perimeter	num			Detected	0.000005
>	mean concave points	num			Detected	0.000009

- Nice reports for drift, data quality, etc
- Doesn't solve the data infra / scale problem: you still need to provide the datasets
- Less useful for unstructured data and observability / debugging

Why logs



- Similar to Evidently but more focused on helping with logging data from production
- Some support for unstructured data
- Same limitations for debugging

ML monitoring / observability vendors

1. Logging

Logging: get data from your model to a place where you can analyze it

What data to log?

What data to log?

- For most of us, just log all of the data
  - Storage is cheap
  - Better to have it than not
- When can’t you log all data?
  - Massive traffic volume
  - Data privacy concerns
  - Expensive data transfer (e.g., edge)

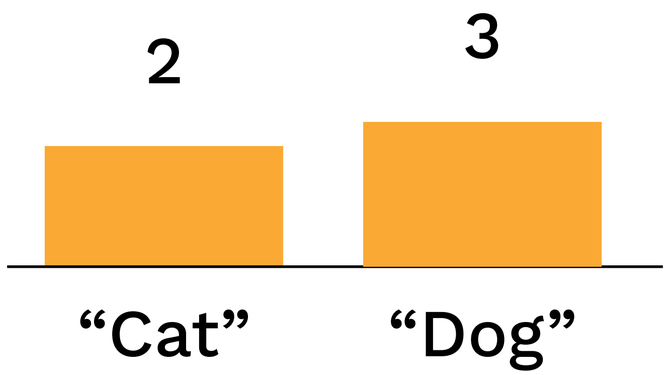
Two strategies when you can't log all of the data

	What is it?	Advantages	Use it for...
Profiling	Compute statistical profiles of data on the edge that describe the distribution	• Data security • Minimal storage cost • You don't "miss" what happens in the tails	Security-critical applications
Sampling	Sample certain data points to send back "home"	• Minimal impact on inference resources • You get the raw data for debugging / retraining	Most applications

Statistical profiles

“Cat” “Cat”

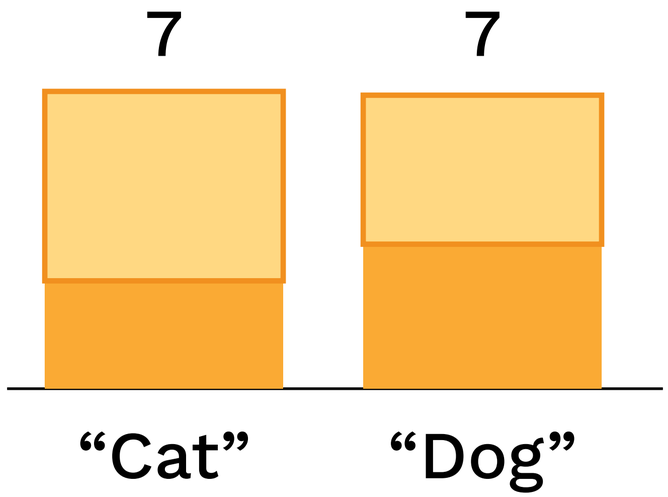
“Dog” “Dog” “Dog”



Statistical profiles

“Cat” “Cat” “Cat” “Cat” “Cat” “Cat” “Cat”

“Dog” “Dog” “Dog” “Dog” “Dog” “Dog” “Dog”



2. Curation

Curation: turn your infinite “stream” of data from production into a finite “reservoir” of enriched data for potential training

What data to select for enrichment?

Strategies for picking data to enrich

- **L1: just sample randomly**
  - Most data might not be helpful
  - Miss “rare” classes or events
- **L2: Stratified sampling**
  - Sample specific proportions of individuals from various subpopulations (e.g., balance among classes, genders, etc)
- **L3: curate “interesting” data**
  - User-driven curation (aka feedback loops)
  - Manual curation (aka error analysis)
  - Automatic curation (aka active learning)

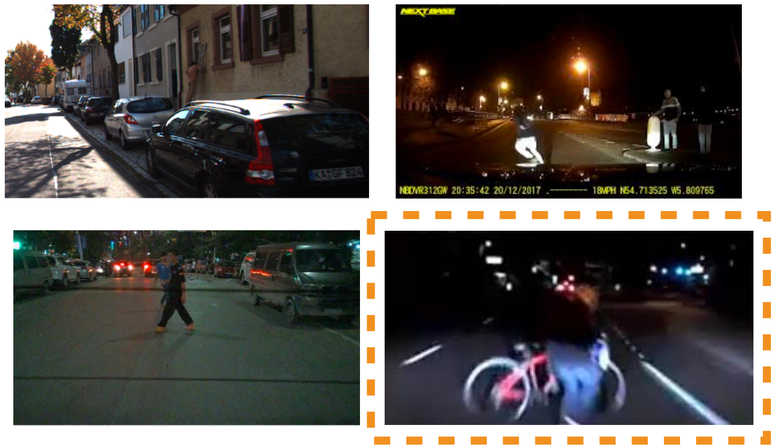
Letting users curate data with feedback loops

- Pick data based on signals that your user didn’t “like” the predictions
- Examples
  - User churned
  - User filed a support ticket
  - User clicked “thumbs down”
  - User changed the label
  - User “intervened” with automatic system, etc

Manually curating data via error analysis

- Look at our model's errors
- Reason about failure modes
- Write functions to curate data representing the failure modes

Similarity-based curation



Nearest neighbors of this

- Easy, fast, beginning to be widely used in practice
- Good for rare but easy to detect events

Projection-based curation

Write a function (or train a model) to detect this (but not these)

- Requires domain knowledge
- Good for more subtle error modes

Automatically curating data using active learning

- Given a large amount of unlabeled data, determine which data points would improve model performance the most if you label them next
- Define a *sampling strategy* or *query strategy* by ranking examples using a *scoring function* U(x) and taking those with the highest scores

https://lilianweng.github.io/posts/2022-02-20-active-learning/

A quick tour of scoring functions

- **Most uncertain:** sample low-confidence / high entropy predictions, or predictions that an ensemble disagrees on
- **Highest predicted loss**: train a separate model that predicts loss on unlabeled points. Sample the highest predicted loss
- **Most different from labels:** train a model (GAN, etc) to distinguish labeled and unlabeled data. Sample the easiest to distinguish
- **Most representative:** choose points such that no data is too far away from anything we sampled
- **Big impact on training:** choose points such that the expected gradient is large, or points where the model changes its mind the most about its prediction during training

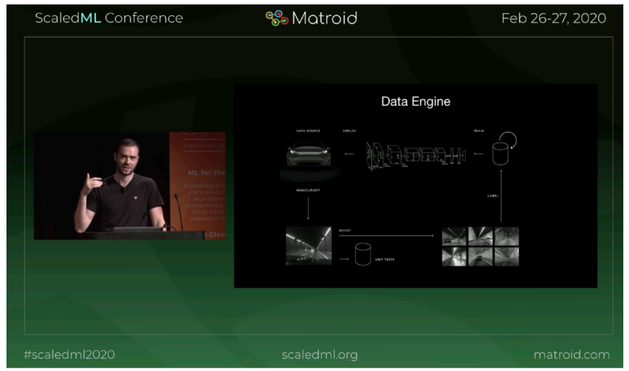
https://lilianweng.github.io/posts/2022-02-20-active-learning/

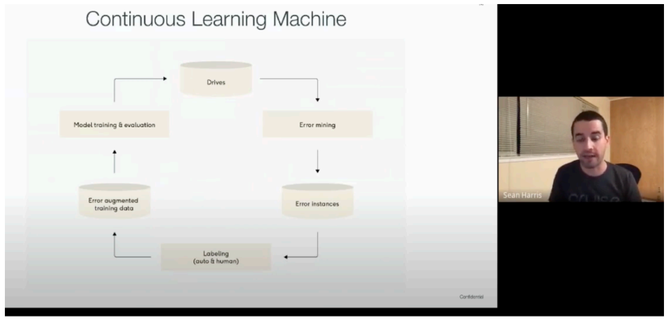
Observation: data curation ~= monitoring

Data curation technique	Equivalent monitoring technique	Example
User-driven curation	Monitoring user feedback metrics	Monitor & curate based on driver interventions
Stratified sampling	Subgroup / cohort analysis	Monitor performance by age cohort, and curate to balance across age
Projection-based curation	Projection-based monitoring	Write a projection to detect language. Alert and curate on underperforming ones
Predicted loss-based active learning	Predicted OOD performance	Train a loss predictor and use it to alert and curate
Uncertainty-based active learning	Uncertainty as a proxy metric	Capture the uncertainty of your image classifier

Data curation in practice

DALL·E 2 Pre-Training Mitigations





OpenAI

- Active learning (uncertainty sampling) to reduce false-positives
- Manual curation (similarity search) to reduce false-negatives

Tesla

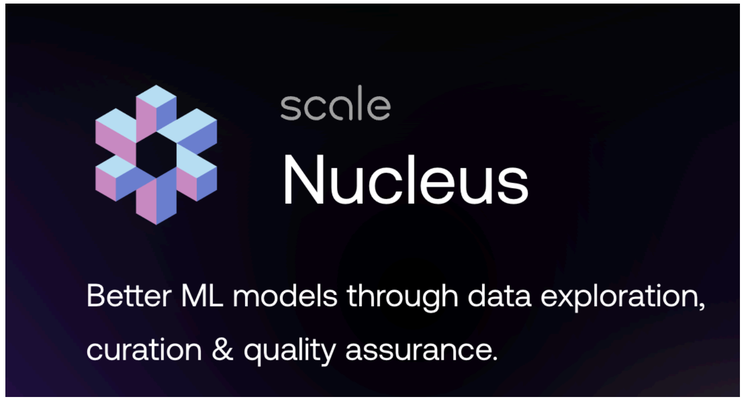
- Feedback loops
- Manual curation via projections for edge case detection (deployed on the fleet!)

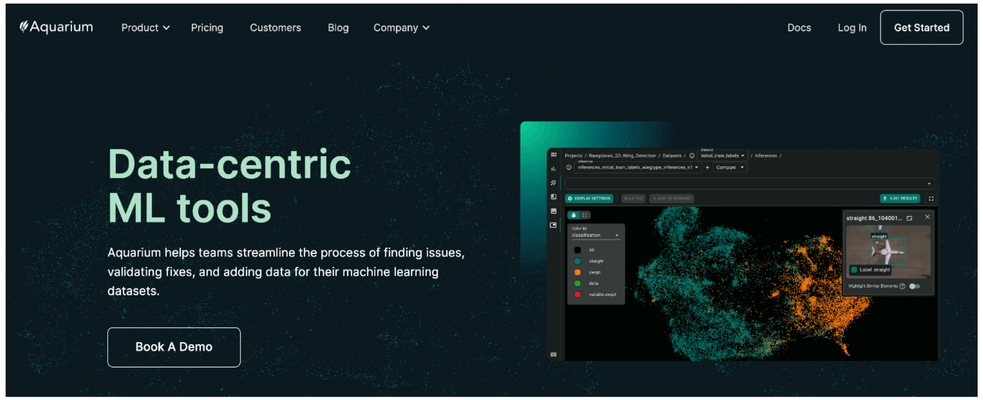
Cruise

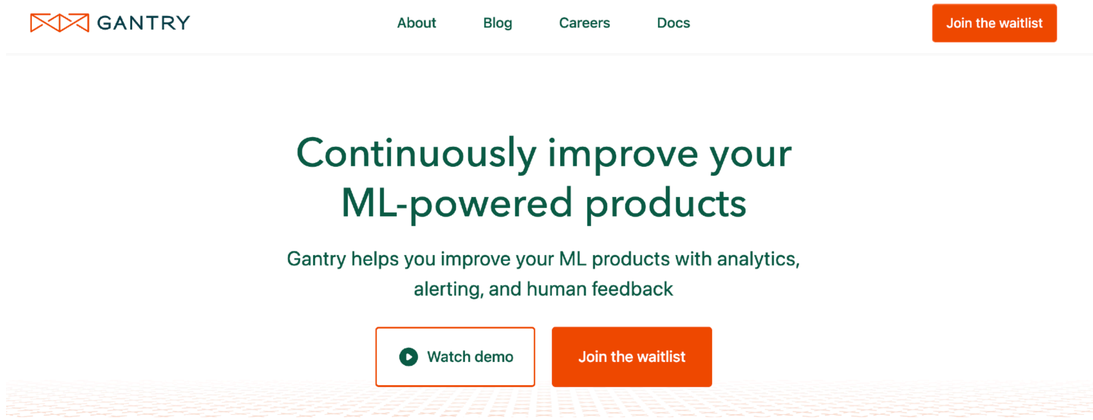
- Feedback loops

https://openai.com/blog/dall-e-2-pre-training-mitigations/
https://www.youtube.com/watch?v=hx7BXih7zx8

Tools for data curation







Data curation: recommendations

- Random sampling is a fine starting point
  - If you have want to avoid bias or have rare classes, do stratified sampling instead
- If you have a feedback loop, then user-driven curation is a no-brainer
  - If not, confidence-based active learning is easy to implement
- As your model performance increases, you'll have to look harder for challenging training points
  - Manual techniques are unavoidable and should be embraced. Know your data!

3. Retraining triggers

Retraining triggers: signal when to retrain your model

Moving from manual to automated retraining is not always necessary, but it can:

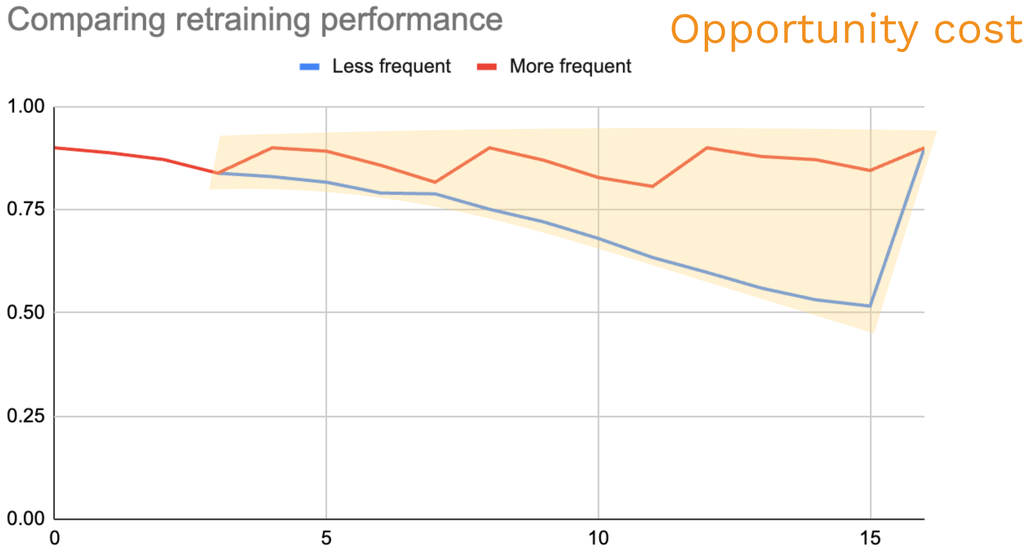
- Save you time
- Lead to better model performance

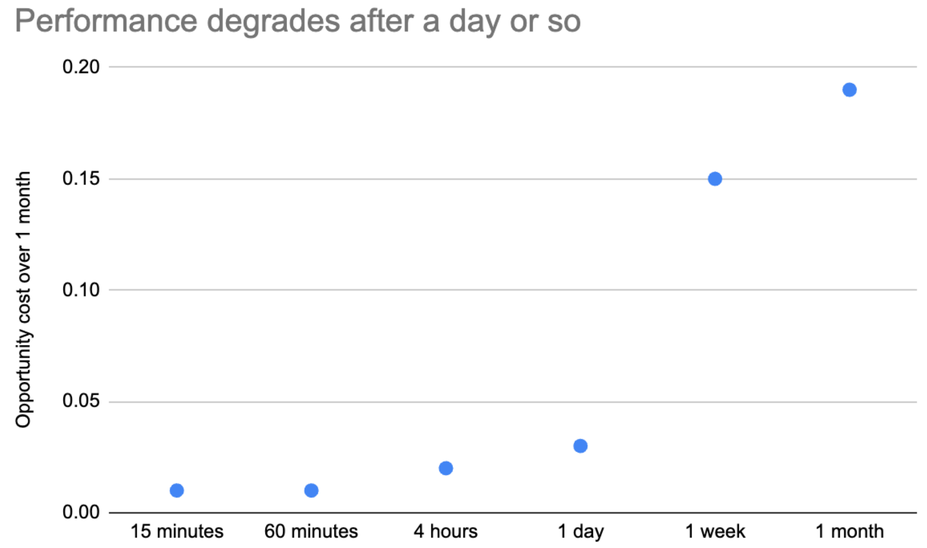
When to move to automated retraining

- You can reproduce model performance when retraining, and
- It’s time to move on to other projects
- Or, you find yourself retraining more than ~1x / month

Recommendation: just retrain periodically

- Set a retraining frequency like 1x / week, etc
- Re-run training on that schedule
- How to pick the frequency?
  - Measure how performance degrades over time
  - Compare different retraining schedules, assuming you get back to same performance
  - Area between the curves = opportunity cost
  - Balance performance degradation with retraining and operational costs





Request for research

- Automate determining the optimal retraining strategy based on:
  - Performance decay
  - Sensitivity to performance
  - Operational costs
  - Retraining costs

More advanced: retrain based on performance triggers

- Set triggers like “retraining when accuracy is below 90%”
- Pros
  - React more quickly to unexpected changes
  - More cost-optimal
- Cons
  - Need good instrumentation / measurement to compute these signals
  - Not very well understood theoretically
  - Adds operational complexity

Online learning: train on every data point as it comes in

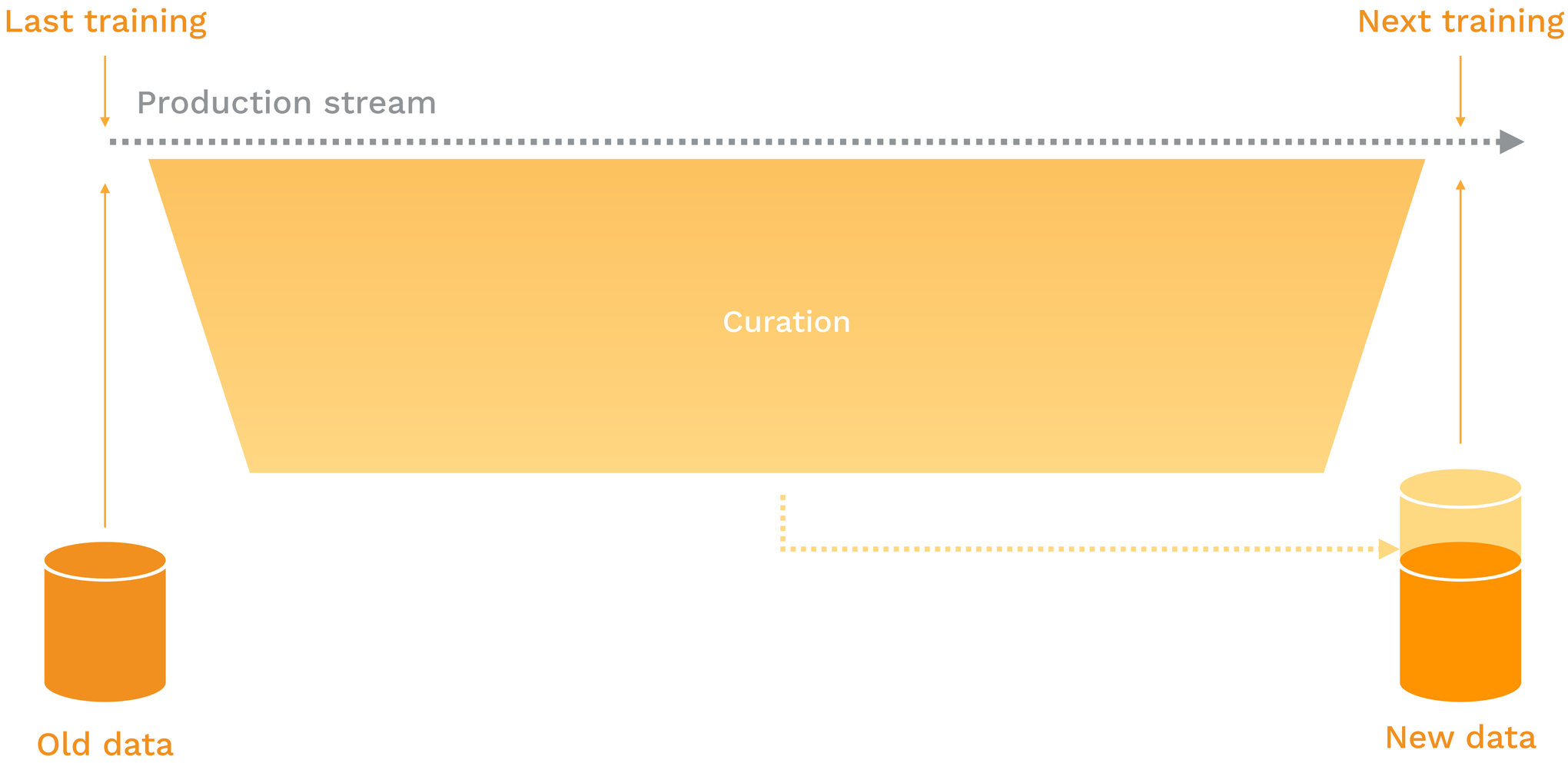
- Not used much in practice: complex and error-prone
- More common: online *adaptation*
- *Policies*: the rule that turns the raw prediction into what the user sees
  - E.g., classification threshold, ensemble weights
- Online *adaptation* tunes the policy using multi-arm bandits

4. Dataset formation

Dataset formation: pick points from the reservoir to train on

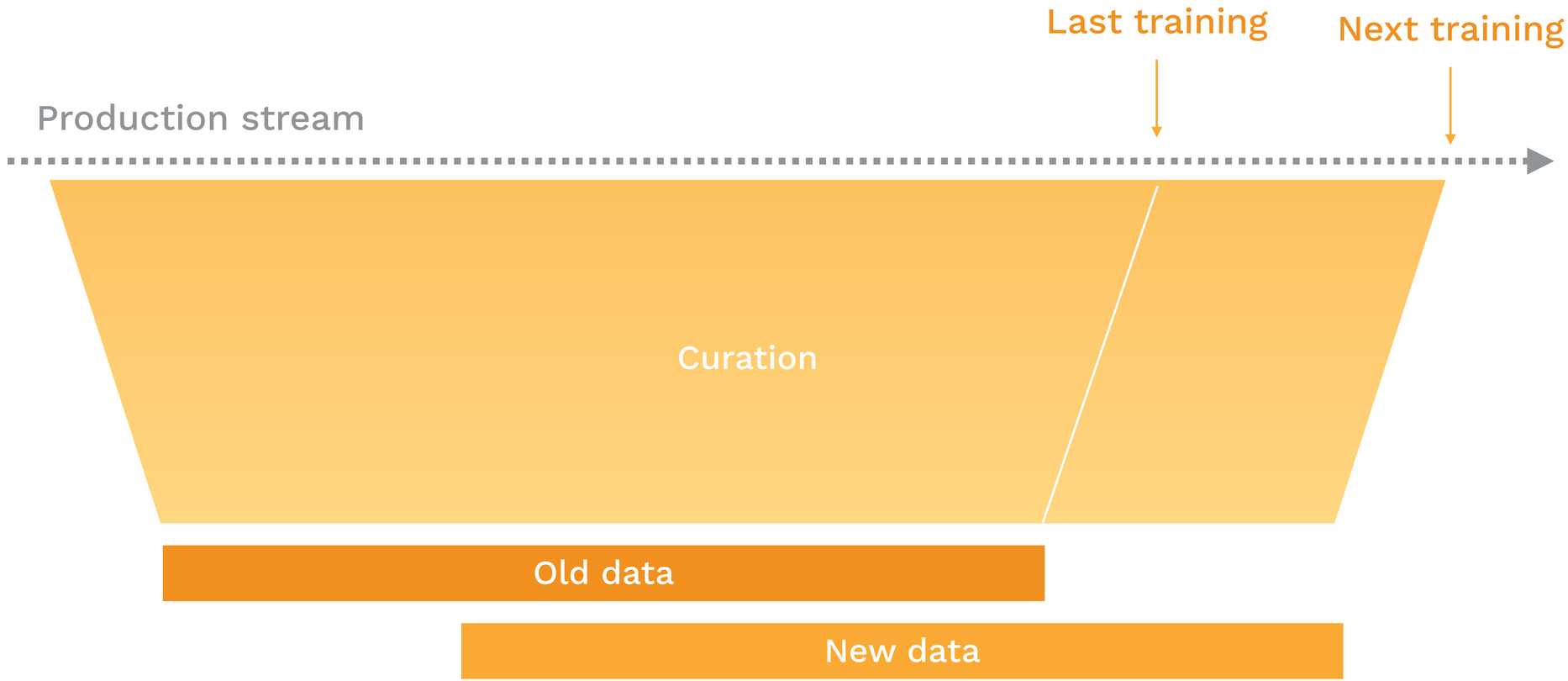
- Most of the time in deep learning, train on all available data
- If you have too much data:
  - Use a sliding window if data changes quickly
  - Sampling or online batch selection if not
- Advanced / not recommended (yet) — continual fine-tuning

Option 1: train on all available data



- Keep it version controlled (W&B, DVC are options)
- Keep track of the curation “rules” used to add the new data

Option 2: sliding window



- Look at the different statistics between old & new to catch bugs
- Comparison can be a challenge: compare in aggregate and on the intersection

Option 3: online batch selection

- Normally in SGD: sample mini batches over and over until we run out of compute budget
- Instead, *online batch selection*: Before each training step:
  - Sample a larger batch
  - Rank those items by a "label-aware selection function" (LASF)
  - Pick the top n items and train on them

Prioritized Training on Points that are learnable, Worth Learning, and Not Yet Learnt

Sören Mindermann [*1] Jan Brauner [*1] Muhammed Razzak [*1] Mrinank Sharma [*2] Andreas Kirsch [1]
Winnie Xu [3 4] Benedikt Höltgen [1] Aidan N. Gomez [3 1] Adrien Morisot [3] Sebastian Farquhar [1] Yarin Gal [1]

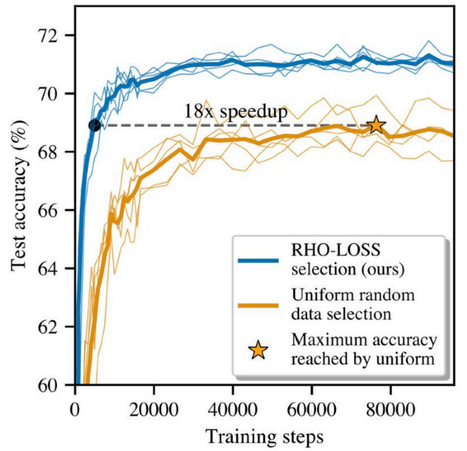


Figure 1: **Speedup on large-scale classification of web-scraped data (Clothing-1M).** RHO-LOSS trains all architectures with fewer gradient steps than standard uniform data selection (i.e. shuffling), helping reduce training time. Thin lines: ResNet-50, MobileNet v2, DenseNet121, Inception v3, GoogleNet, mean across seeds. Bold lines: mean across all architectures.

Reducible holdout loss selection (RHO-LOSS)

$$\underset{(x,y)\in B_t}{\arg\max} \overbrace{\underbrace{L[y \mid x; \mathcal{D}_t]}_{\text{training loss}} - \underbrace{L[y \mid x; \mathcal{D}_{ho}]}_{\text{irreducible holdout loss (IL)}}}^{\text{reducible holdout loss}} \quad (3)$$

Option 4: continual fine-tuning

- Rather than retrain from scratch each time, fine-tune on newly sampled data
- More cost-effective: 45x more for Grubhub
- Challenges: model may “forget” what it learned before
- Need to invest heavily in evaluation

Online Learning for Recommendations at Grubhub

ALEX EGG

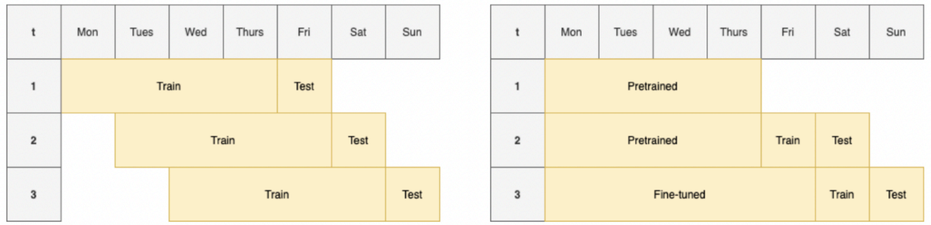


Fig. 1. 1a Example of daily Stateless updates with a 4-day sliding-window and next day hold-out for Cross Validation. 1b Example of daily incremental updates with state & Bootstrapping. Notice the model is only updating on *new* data, not the sliding-window in Fig 1a. The cadence can be 1-day, as in this example, or as small as a 10 minutes as in [2]

5. Offline testing

Offline testing: produce a “report” that answers whether the new model is better

What should go into the report?

Evaluating a model before deployment

- Compare current to previous model on the following
  - All metrics
  - On all important “slices” of data
  - On all “edge cases”
  - Adjusted to account for sampling bias

A comparison report

	Metrics		
Datasets	**Accuracy**	**Precision**	**Recall**
Main validation set (adjusted for sampling)	0.94 (+0.01)	0.91 (+ 0.00)	0.92 (+0.01)
...User age			
0-18	0.90 (-0.05)	0.89 (+ 0.00)	0.89 (-0.06)
18-35	0.93 (+0.01)	0.90 (+ 0.00)	0.92 (+0.01)
35-60	0.94 (+0.01)	0.91 (+ 0.00)	0.93 (+0.02)
60+	0.94 (+0.01)	0.91 (+ 0.00)	0.91 (+0.01)
...Account age			
<1 week	0.94 (+0.01)	0.91 (+ 0.00)	0.92 (+0.01)
1-4 weeks	0.90 (-0.01)	0.88 (+ 0.00)	0.90 (-0.01)
4-12 weeks	0.91 (+0.01)	0.90(+ 0.00)	0.91 (+0.01)
12+ weeks	0.93 (+0.01)	0.93 (+ 0.00)	0.90 (+0.00)
Error cases			
Copypasta	0.80 (+0.01)	0.91 (+ 0.00)	0.85 (+0.00)
Poor grammar	0.83 (+0.01)	0.91 (+ 0.02)	0.82 (+0.01)
Gen Z slang	0.72 (-0.09)	0.74 (- 0.08)	0.71 (-0.10)

Evaluation sets are dynamic

- As you curate new data, add some of it to your evaluation sets
  - E.g., change how you do sampling —> add that newly sampled data to eval
  - E.g., new edge case —> create a test case for it
- Corollary 1: you should version control your evaluation sets as well
- Corollary 2: if your data changes quickly, always hold out the most recent data for evaluation

Advanced: consider using *expectation tests*

- Pairs of examples with a fixed relationship
  - E.g., ex2 should be *more positive* than ex1
- Tests the ability of the model to generalize in predictable ways

Beyond Accuracy: Behavioral Testing of NLP Models with CheckList

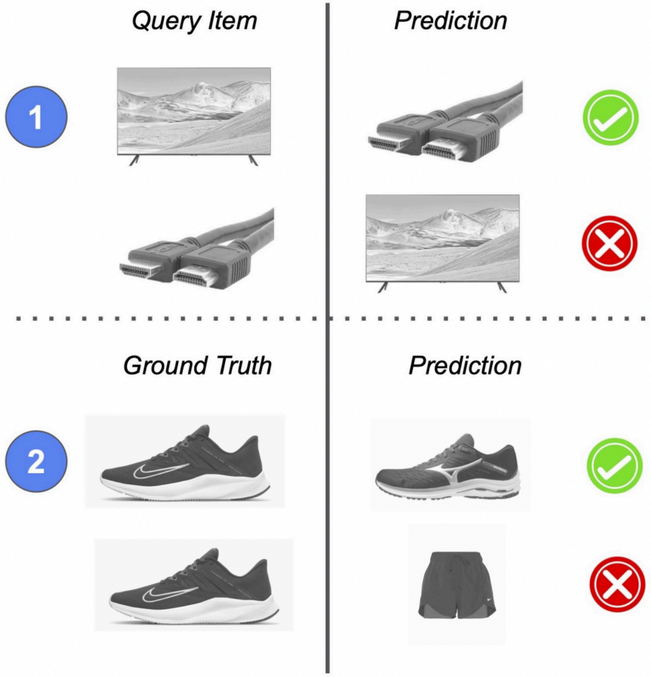
Marco Tulio Ribeiro Microsoft Research marcotcr@microsoft.com
Tongshuang Wu Univ. of Washington wtshuang@cs.uw.edu
Carlos Guestrin Univ. of Washington guestrin@cs.uw.edu
Sameer Singh Univ. of California, Irvine sameer@uci.edu

Capability	Min Func Test	INVariance	DIRectional
Vocabulary	Fail. rate=15.0%	16.2%	C 34.6%
NER	0.0%	B 20.8%	N/A
Negation	A 76.4%	N/A	N/A
...			

Test case	Expected	Predicted	Pass?
A Testing **Negation** with ***MFT***	Labels: negative, positive, neutral		
`Template: I {NEGATION} {POS_VERB} the {THING}.`			
I can't say I recommend the food.	neg	pos	x
I didn't love the flight.	neg	neutral	x
...			
Failure rate = 76.4%			
B Testing **NER** with ***INV***	Same pred. (`inv`) after removals / additions		
@AmericanAir thank you we got on a different flight to [Chicago → Dallas].	inv	pos / neutral	x
@VirginAmerica I can't lose my luggage, moving to [Brazil → Turkey] soon, ugh.	inv	neutral / neg	x
...			
Failure rate = 20.8%			
C Testing **Vocabulary** with ***DIR***	Sentiment monotonic decreasing (↓)		
@AmericanAir service wasn't great. You are lame.	↓	neg / neutral	x
@JetBlue why won't YOU help them?! Ugh. I dread you.	↓	neg / neutral	x
...			
Failure rate = 34.6%			

Beyond NDCG: behavioral testing of recommender systems with RecList

Patrick John Chia* Coveo Canada pchia@coveo.com
Jacopo Tagliabue Coveo Labs United States jtagliabue@coveo.com
Federico Bianchi Bocconi University Italy f.bianchi@unibocconi.it
Chloe He Stanford University United States chloehe@stanford.edu
Brian Ko KOSA AI United States sangwoo@kosa.ai



Observation: testing ~= monitoring

- Just like in monitoring, we want to observe metrics across subsets of data and edge cases
- Different metrics are available though
  - More likely to have labels offline
  - More likely to have feedback online
- Request for research: predicting online metrics from offline ones

6. Online testing

Covered last time!

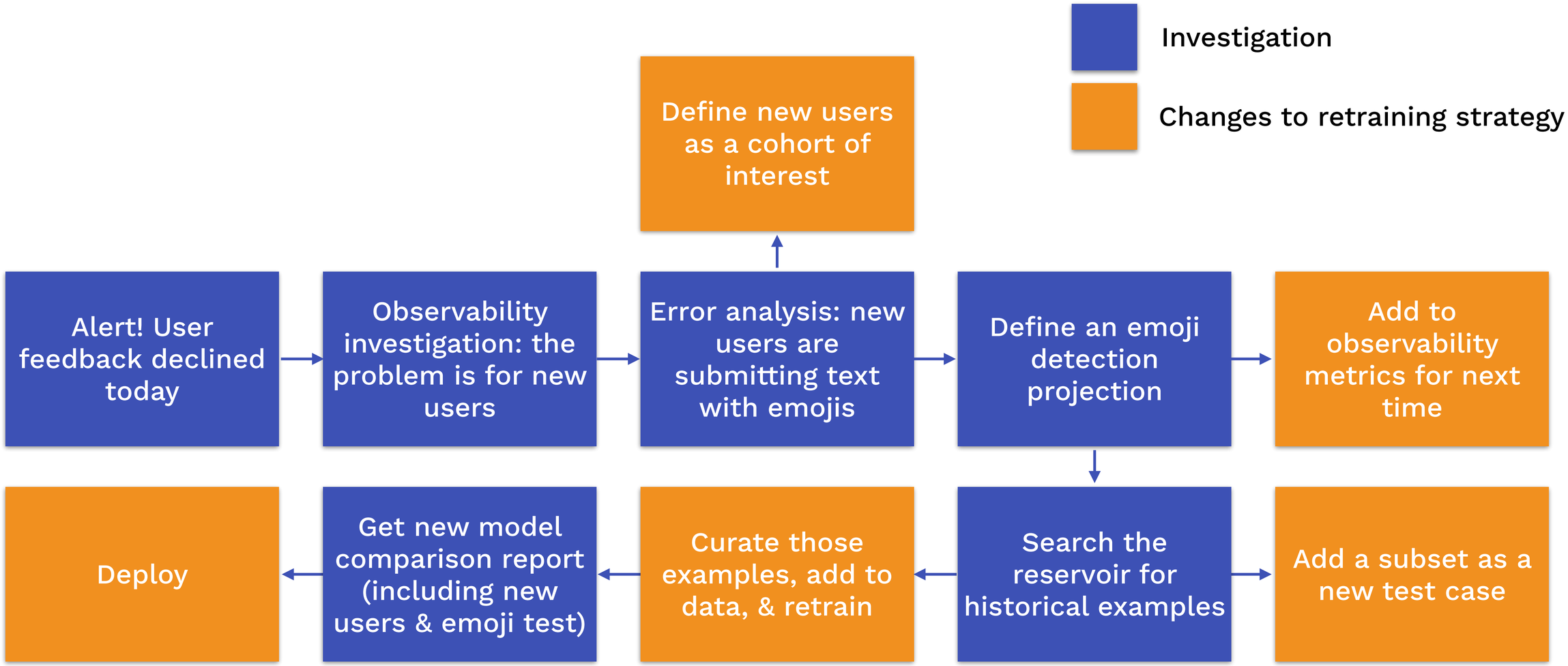
- If you have the capability to do so:
  - Run in shadow mode
  - Run an AB test
  - Roll out gradually
  - Roll back if you see issues during rollout

Tying it all together: the continual improvement workflow

Prerequisite: a place to store (and version) your strategy

- **Metric definitions** for monitoring, observability, and offline testing
- **Performance thresholds** for monitoring and offline testing
- **Projection definitions** for monitoring and manual data curation
- **Subgroups / cohorts** of interest for monitoring and offline testing
- **Data curation logic**
- **Datasets** for training and evaluation

A continual improvement loop



Continual learning

- A complicated, evolving, poorly understood topic — watch this space!
- We broke down the concept of a ***retraining strategy***, which consists of:
  - Metric definitions, subgroups of interest, projection definitions, performance thresholds, data curation logic, and datasets for training / evaluation
- As MLE, your goal after you deploy the minimum viable model is to use monitoring & observability to iterate on the strategy
- Like all aspects of the ML lifecycle — start simple, and add complexity as you go