Data Engineering

DMLS Ch. 8: Data Distribution Shift and Monitoring

Sundong Kim

https://sundong.kim/courses/dataeng24sp

Contents from Shreya Shankar's Slide

Agenda

- Dealing with machine learning (ML) pipelines sucks
- Shift recap & existing methods
- Toy ML task introduction
- Monitoring challenges & solution ideas

Dealing with ML Pipelines Sucks 🤮

Production ML

An on-call engineer's biggest nightmare 😱

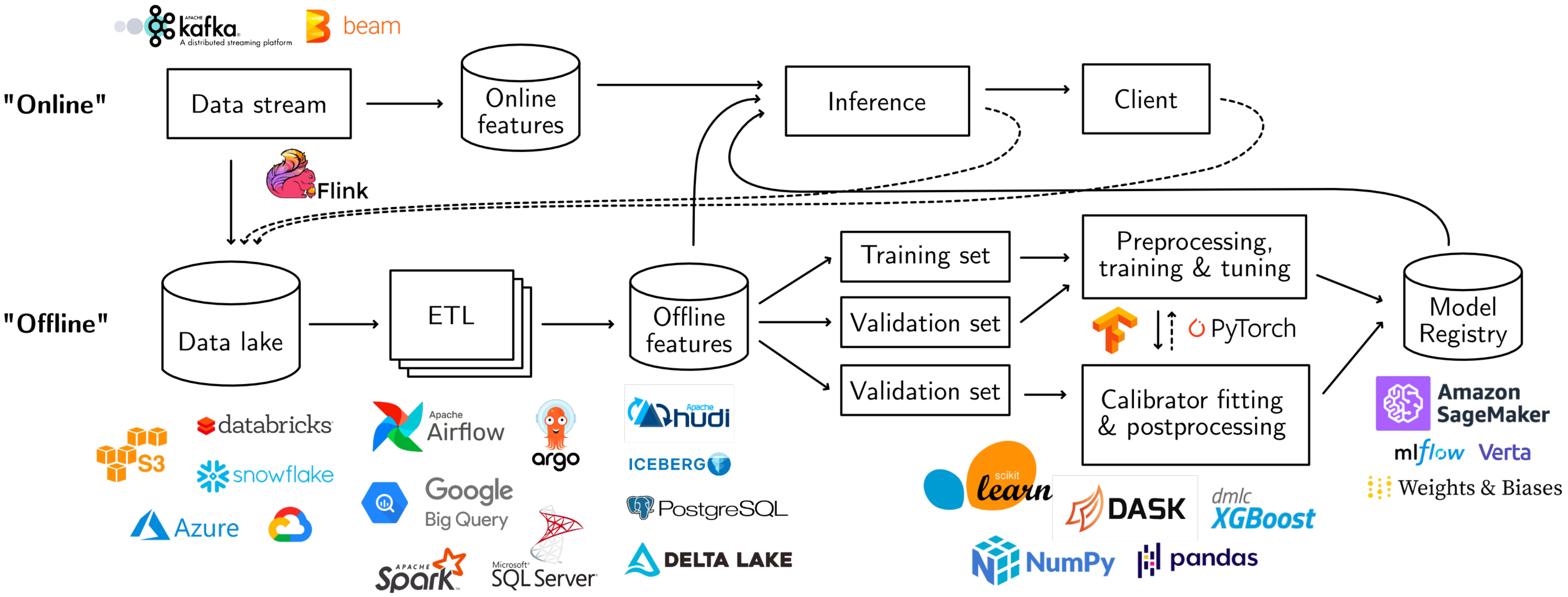


Figure 1: High-level architecture of a generic end-to-end machine learning pipeline. Logos represent a sample of tools used to construct components of the pipeline, illustrating heterogeneity in the tool stack. *Shankar et al. 2021*

Production ML

An on-call engineer's biggest nightmare 😱

- Many problems arise post-deployment
  - Corrupted upstream data
  - Model developer is on leave
  - Training assumptions don't hold in practice
  - Data "drifts" over time
  - And more...

Why Observability?

- Can't catch all bugs before they happen, but we want to *minimize downtime*
- We should:
    - Help engineers *detect* bugs
    - Help engineers *diagnose* bugs
- Need to support a wide variety of skill sets
    - Engineers, data scientists, etc.

Types of ML Data Management Solutions

Pre-training

- What do I need to start training a model?
- Feature stores, ETL pipelining, etc

Experiment Tracking

- What's the best model for a pipeline?
- `mlflow`, `wandb`, etc

Observability

- There's a bug in my pipeline; where is it?
- Real-time ML performance monitoring

Real-Time ML Performance Monitoring: Background

Why is this Hard?

Data "shifts"... 🤔

- Determining real-time performance requires *labels*
  - ...which are not always available post-deployment
- Is performance drop temporary (e.g., seasonal) or forever?
- Degenerate feedback loops
  - I.e., when predictions influence feedback (which labels are extracted from)

Shift Recap

Notation 🎶

- X is feature (covariate) space, Y is label space
- P(X): distribution of features
- P(Y): distribution of labels
- P(X | Y): distribution of features given specific labels
- P(Y | X): distribution of labels given specific features
  - *This is what ML models are trying to learn!*

Shift Recap

Terminology 🧑‍🏫

- Covariate shift
  - P(Y | X) is the same but P(X) changes
- Label shift
  - P(Y) changes but P(X | Y) is the same
- Concept shift
  - P(Y | X) changes but P(X) is the same

Existing Methods for Tackling Shift

Levels of sophistication 👸

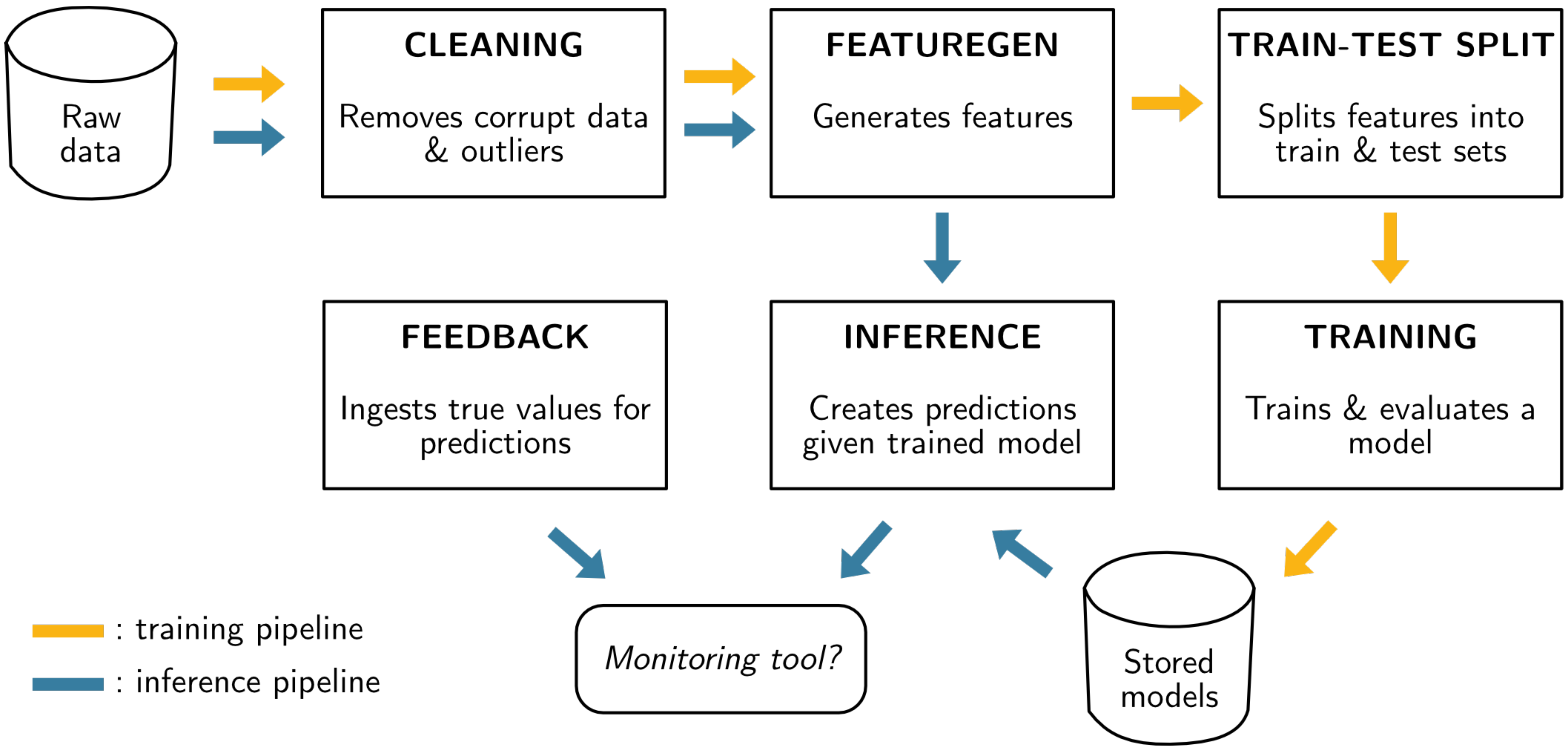
- Straw-man approach 🥤
  - Tracking means & quantiles of features and outputs
- "I took a stats class" approach 🧑‍🎓
  - Tracking MMD, KS & Chi-Square test statistics, etc
  - `alibi-detect`
- **Both approaches are label-unaware and don't use all the information we have. Can we do better?**

Toy ML Task: Running Example 🪀

Task familiarization 🚕

- Binary classification task: predict whether a passenger in a NYC taxi ride will give the driver a "reasonable" tip (>10% of fare)
- Using subsampled data from NYC Taxi & Limousine Commission public dataset
- Using `pd.DataFrame` and `sklearn` Random Forest Classifier
- Evaluating **accuracy**

Pipeline familiarization 👷



Shift Recap

Examples 🧑‍🏫

- X = features (e.g., location), Y = labels (high tip indicator)
- Covariate shift
  - More taxi rides in Midtown area around NYE 🎆
- Label shift
  - Stimulus check causes people to tip more 💸
- Concept shift
  - Heavy construction in certain areas causes people to tip less 🚧

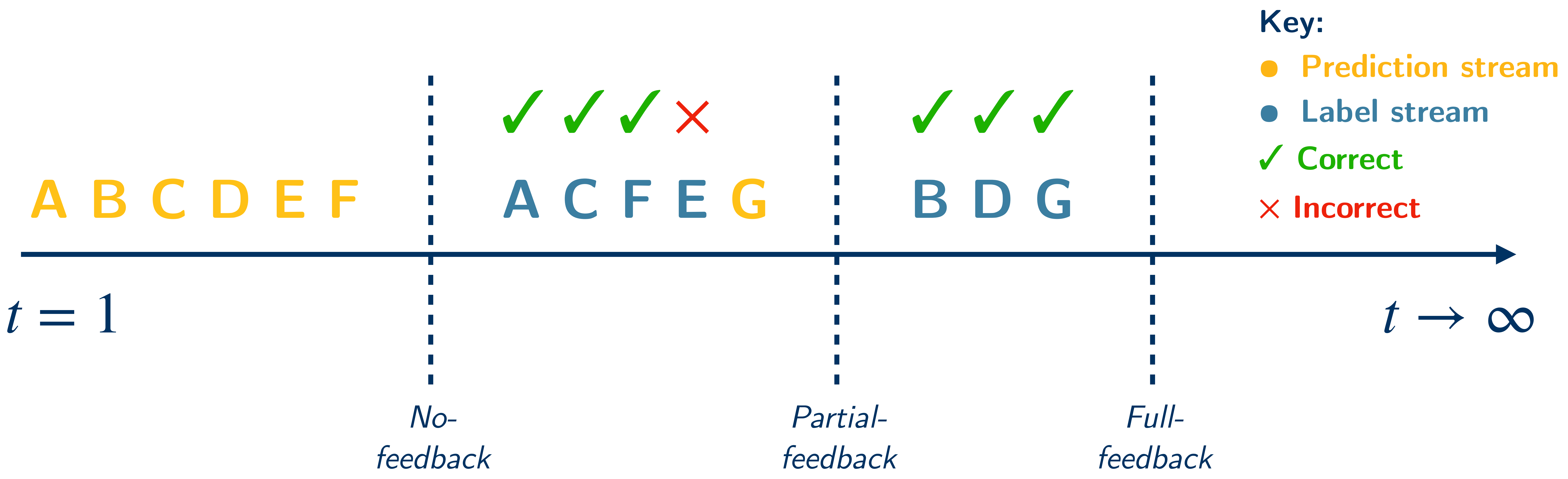
Real-Time ML Performance Monitoring: Challenges 👷

Challenge Tree 🌳

- "Coarse-grained" monitoring: *detecting* performance issues with label delays
  - Full-feedback, no-feedback, and partial-feedback cases
- "Fine-grained" monitoring: *diagnosing* performance issues
  - Teasing out engineering issues from data shift

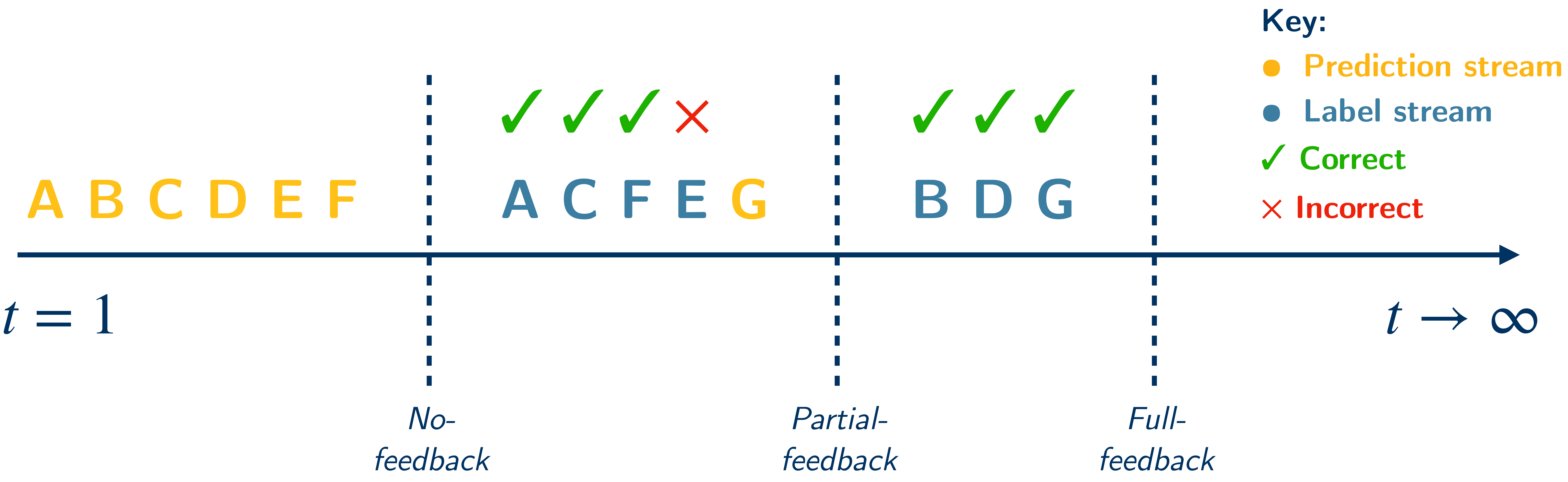
Coarse-grained Monitoring 🔎

Detecting performance issues: Feedback Delays



Coarse-grained Monitoring 🔎

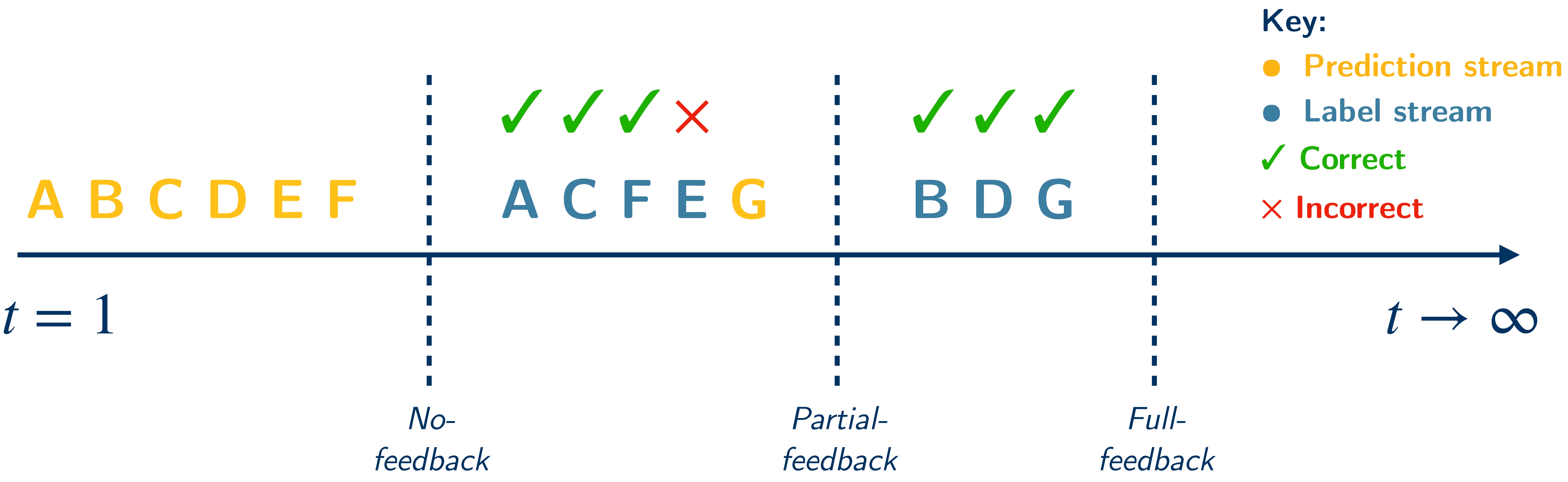
Detecting performance issues: Feedback Delays



Accuracy: ?? 🤔

Coarse-grained Monitoring 🔎

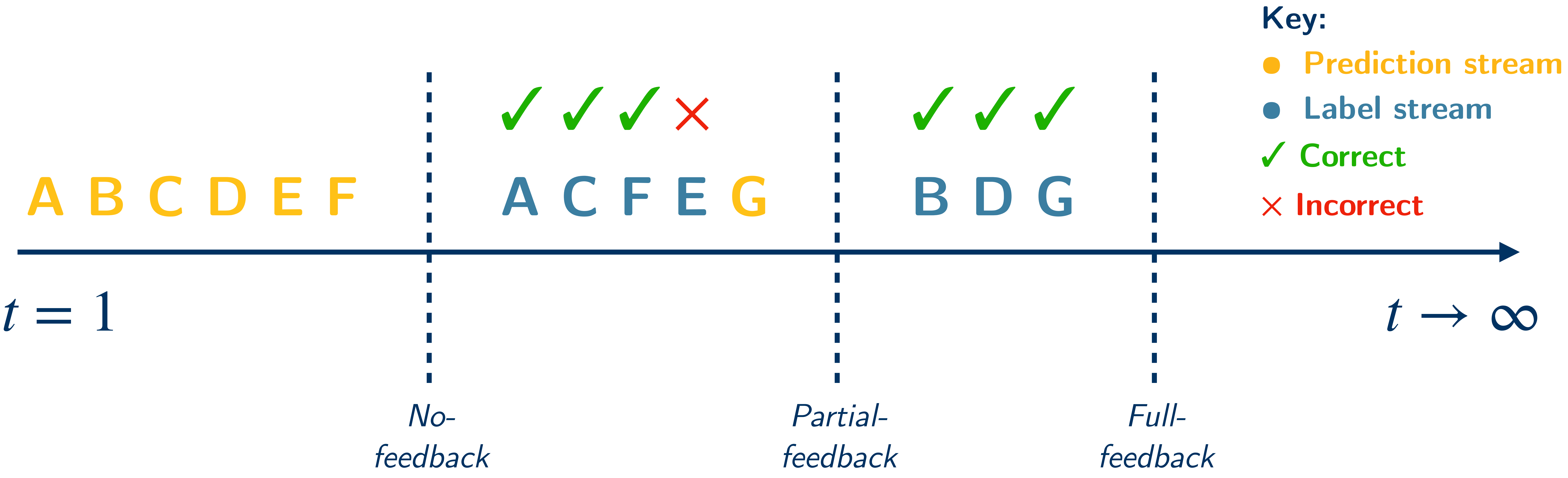
***Detecting* performance issues: Feedback Delays**



Accuracy: 75% ?? 🤔

Coarse-grained Monitoring 🔎

Detecting **performance issues: Feedback Delays**



Accuracy: 86% 😀

Coarse-grained Monitoring 🔎

***Detecting* performance issues: full-feedback** 📫

- # predictions made = # labels received
- Simplest case
  - 1) Do streaming join on predictions & feedback
  - 2) Compute accuracy on result
- What if...data is too large to fit in memory?

Coarse-grained Monitoring 🔎

Detecting performance issues: full-feedback 📫

- What if...data is too large to fit in memory?
  - Approximate streaming joins
  - Uniformly subsampling streams before joins yields *quadratically fewer* resulting tuples
- Idea: stratified subsampling
  - How to construct strata?

Coarse-grained Monitoring 🔎

Detecting performance issues: full-feedback 📫

- **Problem**: randomly subsampling predictions and labels before the join yields quadratically fewer samples to compute accuracy on
- **Solution**: stratified sampling
- How to construct strata/subgroups?
    - **Want**: most accurate overall approximate accuracy
    - **Need**: subgroups with similar prediction errors/losses

Coarse-grained Monitoring 🔎

Detecting performance issues: no-feedback 📭

- Occurs immediately after deployment
- **Problem**: no labels
- **Solution**: importance-weight training subgroup accuracy
  - Split train set into subgroups with similar prediction errors
  - Create criteria for subgroups
  - Determine training accuracy for each subgroup

Coarse-grained Monitoring 🔎

Detecting performance issues: no-feedback 📭

- At inference, classify data point (feature vector) into subgroup
- Importance-weight subgroup training accuracies by inference representation
- Example
  - Subgroups FiDi and Midtown have accuracies of 80% and 50%
  - After deployment, we see 100 FiDi rides and 500 Midtown rides
  - Estimated accuracy $= 0.8 \times 100 + 0.5 \times 500 = \dfrac{80 + 250}{500} =$ **55%**

Coarse-grained Monitoring 🔎

Detecting performance issues: no-feedback 📭

- **Solution**: importance-weight training subgroup accuracy
- How to construct subgroups?
  - **Want**: most accurate overall approximate accuracy
  - **Need**: subgroups with similar prediction errors/losses

Coarse-grained Monitoring 🔎

Detecting performance issues: partial-feedback 📬

- Hybrid of full-feedback & no-feedback?
- Some data points have longer feedback delays than others
  - Delays aren't necessarily uniformly distributed
  - Why?
- **Additional problem**: identify groups of data points with similar feedback delays

Coarse-grained Monitoring 🔎

All feedback schemes boil down to the same research question...

- **How to create dynamically evolving subgroups with similar prediction errors/losses?**
- Solution ideas
  - Train decision tree to predict loss & use leaves as clusters
  - Frequent item-set or predicate search in loss "clusters"
- Lots of hyperparameters to decide 😔
- Need to constantly retrain subgroup models?

Fine-grained Monitoring 🔎

Diagnosing performance issues: data quality issues 🔧

- Instrument pipelines with data quality checks
  - Alert on missing data
  - Set upper and lower bounds for feature values
  - Set constraints for column statistics (e.g., expected mean, median)
- Tedious to scale to 1000s of features 🙀
- Practitioners push DQ verification onto “shift” detection...

Fine-grained Monitoring 🔎

***Diagnosing* performance issues: towards retraining models** 🔁

- Using existing methods to compute shift doesn't work in practice
  - E.g., KS test has low p-values for O(1000) data points
  - Alert fatigue when monitoring every feature and output column
  - Seasonal & expected shifts
- **Idea: look into these statistics when coarse-grained approximated metrics are low**

Fine-grained Monitoring 🔎

***Diagnosing* performance issues: towards retraining models** 🔁

- Different shifts imply different retraining strategies, e.g.,
  - Covariate shift: augment some subgroups in training
  - Concept shift: retrain on recent window
- **Research question: how to create self-tuning training sets?**

`mltrace`: Ongoing Work ⚙️

Ongoing Research Projects

- mltrace: lightweight, "bolt-on" ML observability tool in the making with projects in several research areas

Data Systems	Machine Learning	HCI
• Mitigating effects of feedback delays on real-time ML performance • Differential dataflow to compute streaming ML metrics quickly and efficiently at scale	• Creating streaming ML benchmarks • Building repository of tasks with "temporally evolving tabular data" (e.g. Ethereum gas price prediction)	• Interview study on best practices in CI / CD for ML • Visualizing large-scale data drift

Readings and Resources 📕

- Towards Observability for Machine Learning Pipelines
- The Modern ML Monitoring Mess
  - Rethinking Streaming ML Evaluation
  - Categorizing Post-Deployment ML Issues
  - Failure Modes in Existing Observability Tools
  - Research Challenges

- Contact: shreyashankar@berkeley.edu 📧