Symptom-based alerting for machine learning

Dublin, 10.10.2023

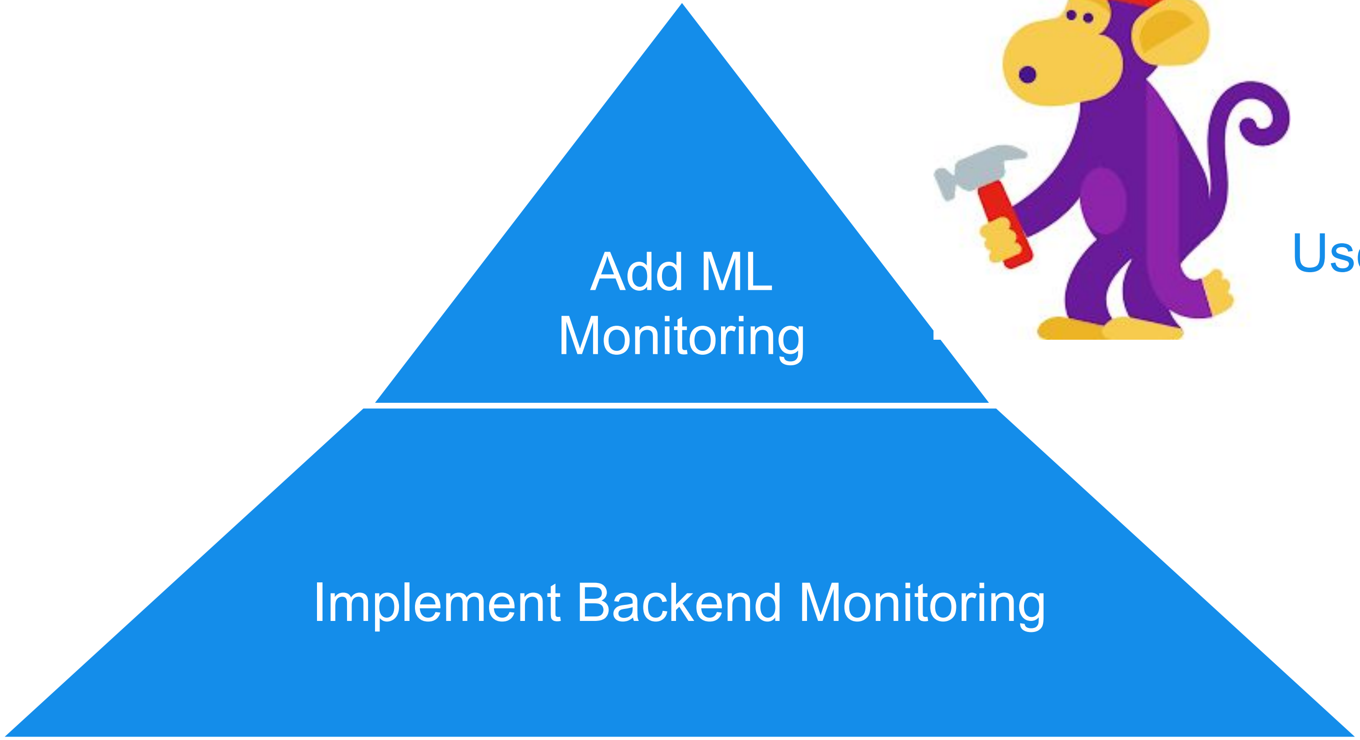
Why sit through this talk

- You are a backend engineer or SRE
- want to monitor a machine learning service
- with easy, existing tooling
- detect quality issues not covered by backend monitoring

HEY, I AM LINA WEICHBRODT

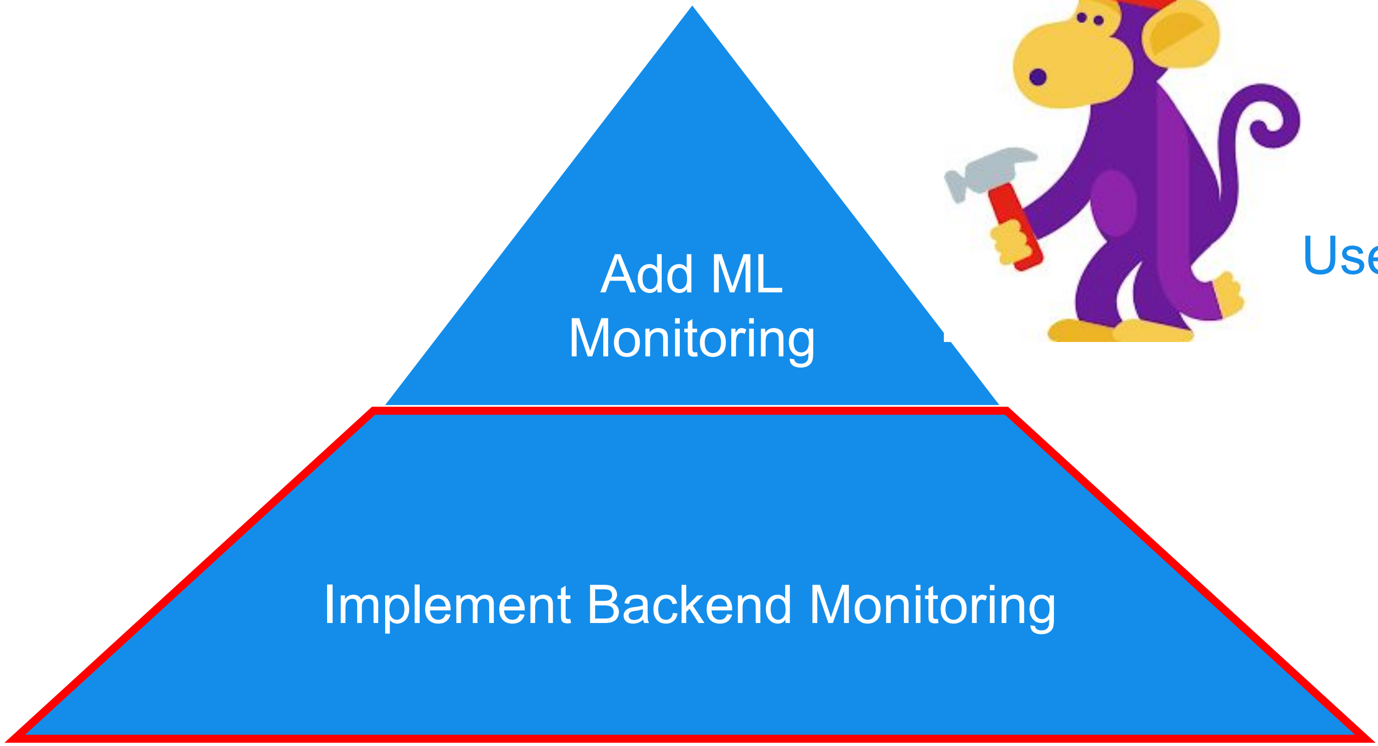
- Machine Learning Consultant and Freelancer with clients working mostly with startups
- Ex Lead Machine Learning Engineer @DKB and Senior Research Engineer @Zalando
- Ran >30 machine learning models in production: Recommender Systems, Personalization, NLP in Customer Service, Finance, Travel

Agenda



Use simple tooling

Agenda



Use simple tooling

Use the four golden signals

Latency: the time it takes to serve a request

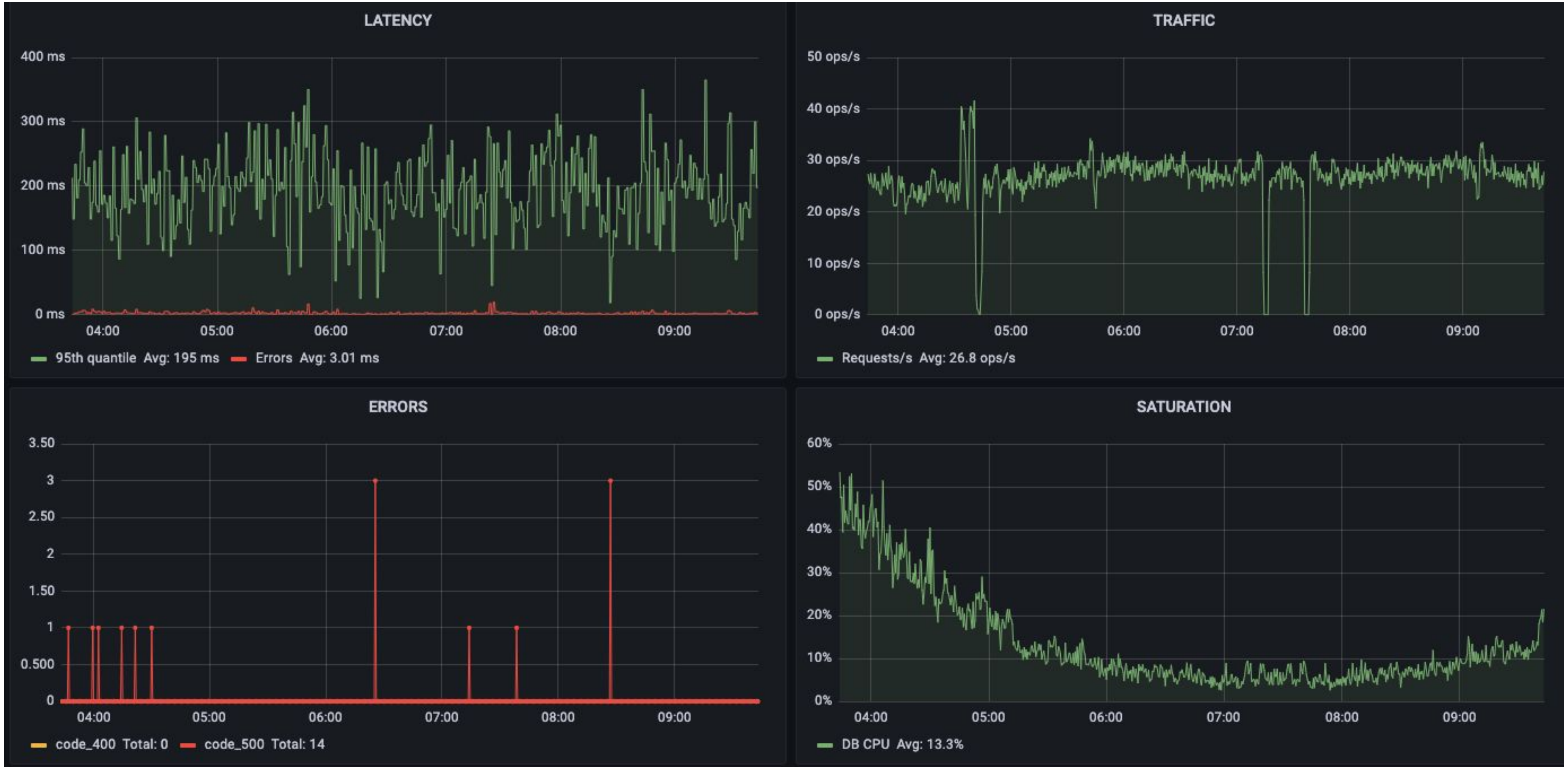
Traffic: the total number of requests

Errors: the number of requests that fail

Saturation: the load on your network and servers

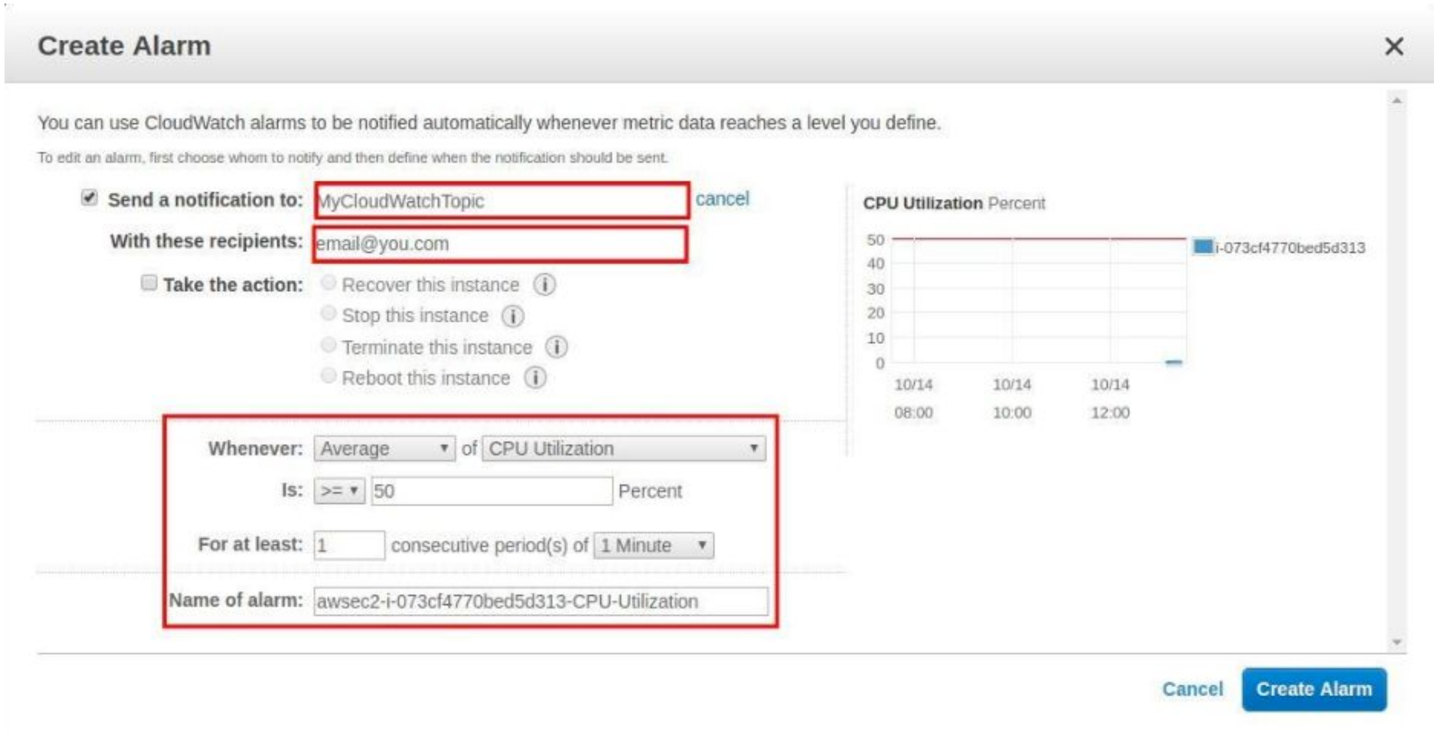
→ we focus on **symptoms**, meaning **end-user pain**, not causes

Monitoring in practice: Live Dashboards



Monitoring in practice: Get notified if a metric is too low or high

Example with AWS Cloudwatch (many vendors offer this needed functionality)



Create Alarm

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.

To edit an alarm, first choose whom to notify and then define when the notification should be sent.

[✓] **Send a notification to:** MyCloudWatchTopic cancel

With these recipients: email@you.com

☐ **Take the action:**
- ○ Recover this instance
- ○ Stop this instance
- ○ Terminate this instance
- ○ Reboot this instance

Whenever: Average of CPU Utilization

Is: >= 50 Percent

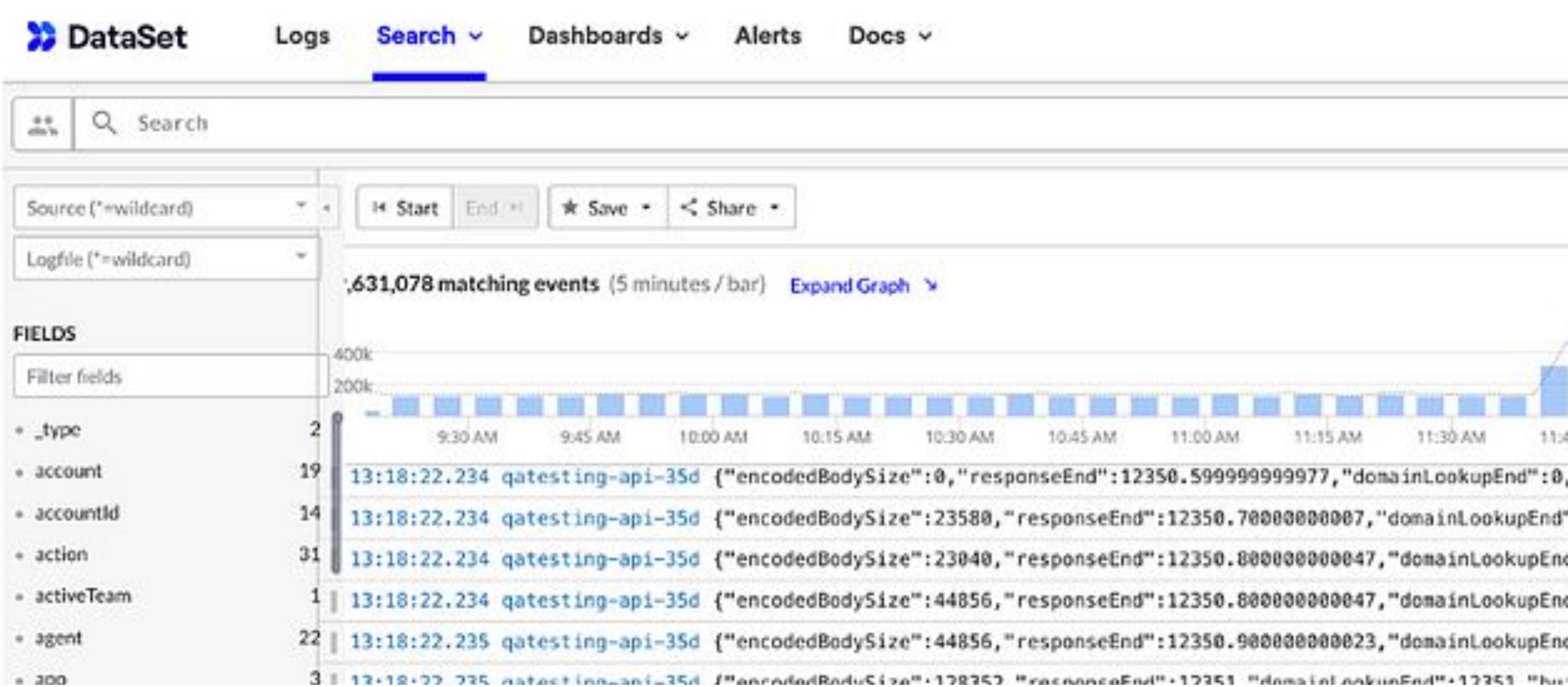
For at least: 1 consecutive period(s) of 1 Minute

Name of alarm: awsec2-i-073cf4770bed5d313-CPU-Utilization

Logging: Write out warnings and caught errors

Filter and analyze errors and warnings

[illegible]

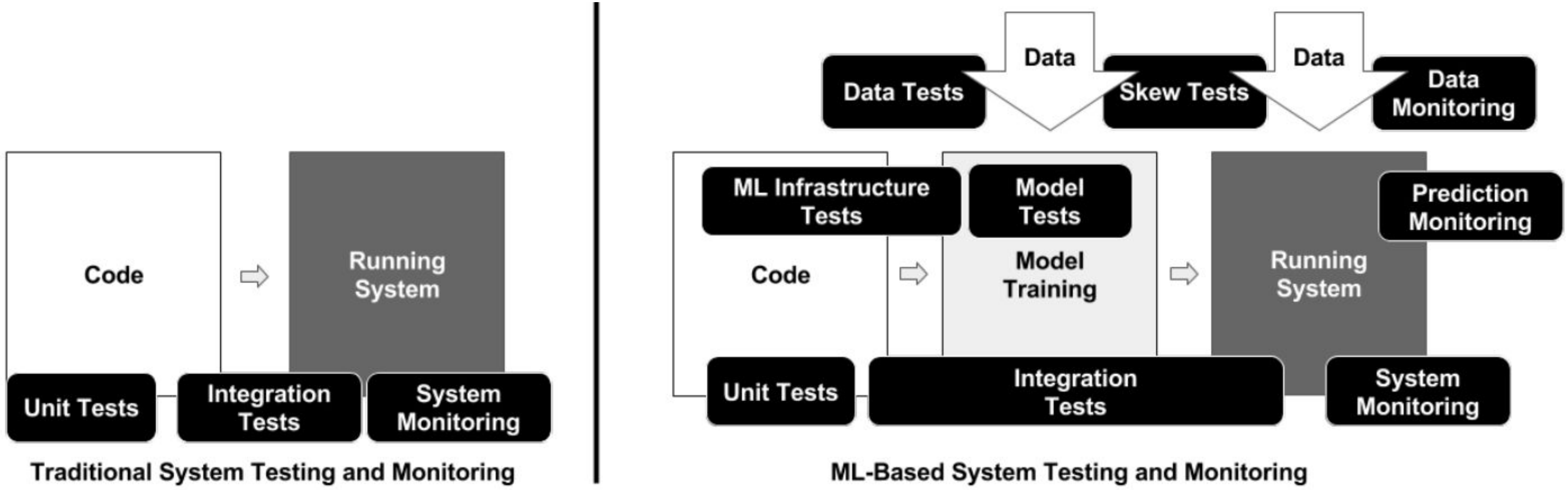


,631,078 matching events (5 minutes / bar)

_type	2
account	19
accountId	14
action	31
activeTeam	1
agent	22
app	3

```
13:18:22.234 qatesting-api-35d {"encodedBodySize":0,"responseEnd":12350.599999999977,"domainLookupEnd":0,
13:18:22.234 qatesting-api-35d {"encodedBodySize":23580,"responseEnd":12350.70000000007,"domainLookupEnd"
13:18:22.234 qatesting-api-35d {"encodedBodySize":23040,"responseEnd":12350.800000000047,"domainLookupEnd
13:18:22.234 qatesting-api-35d {"encodedBodySize":44856,"responseEnd":12350.800000000047,"domainLookupEnd
13:18:22.235 qatesting-api-35d {"encodedBodySize":44856,"responseEnd":12350.900000000023,"domainLookupEnd
13:18:22.235 qatesting-api-35d {"encodedBodySize":128352,"responseEnd":12351,"domainLookupEnd":12351,"bu
```

Is traditional software monitoring enough for machine learning services?



Google paper „ML Test Score“ shows the higher complexity

Silent failures causes huge commercial impact

Examples of silent failure I personally experienced:

- Input data changes
  - Input for fraud model changed from sec to msec
  - External service we used for data enrichment migrated to other technology, data loss
- Aggressive post-processing filters applied
  - Field used for filtering was filled 80%, drifted to 20%
  - Filter “on sale” articles -> fewer articles during non-sale season
- Bugs in our own code e.g. get last 10 orders vs last 10 bought articles
- Model is automatically trained and released, but model is worse
- Tensorflow version not pinned, we got a faulty version
- Client changes the way the product works without telling us, e.g. product is now used by not logged in users

Silent failures causes huge commercial impact

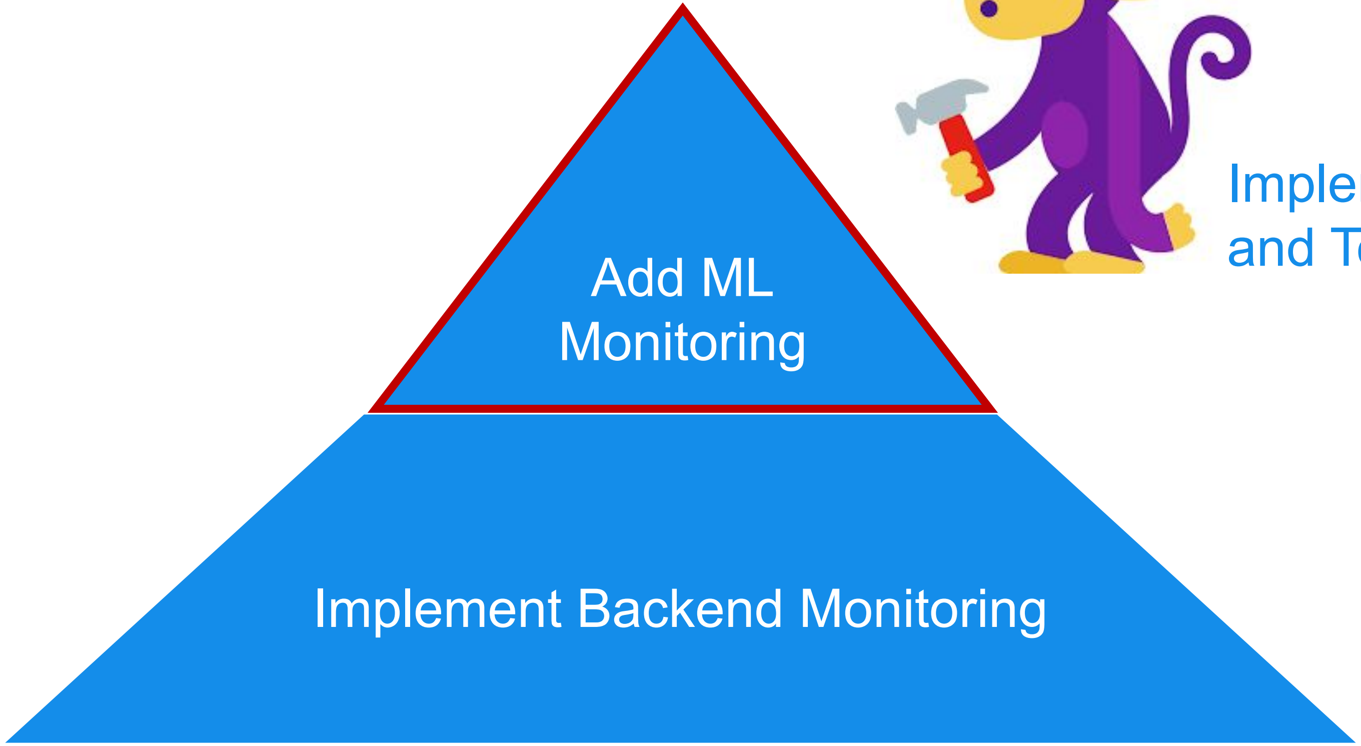
Example

- Input
  - Inp
  - Ext gy, data loss
- Aggres
  - Fiel
  - Filt
- Bugs i
- Model
- Tensor
- Client changes the way the product works without telling us, e.g. product is now used by not logged in users

Permanent, silent loss.

Bigger $$ impact than most model improvements

Agenda



Implementation and Tooling

What to alert on

Alerting should be both **hard failure–centric** and **human-centric**.

Distributed Systems Observability e-Book, Chapter 2: Monitoring and Observability

Keep alerting simple, **alert on symptoms**. Aim to have as few alerts as possible, by alerting on symptoms that are associated with **end-user pain** rather than trying to catch every possible way that pain could be caused.

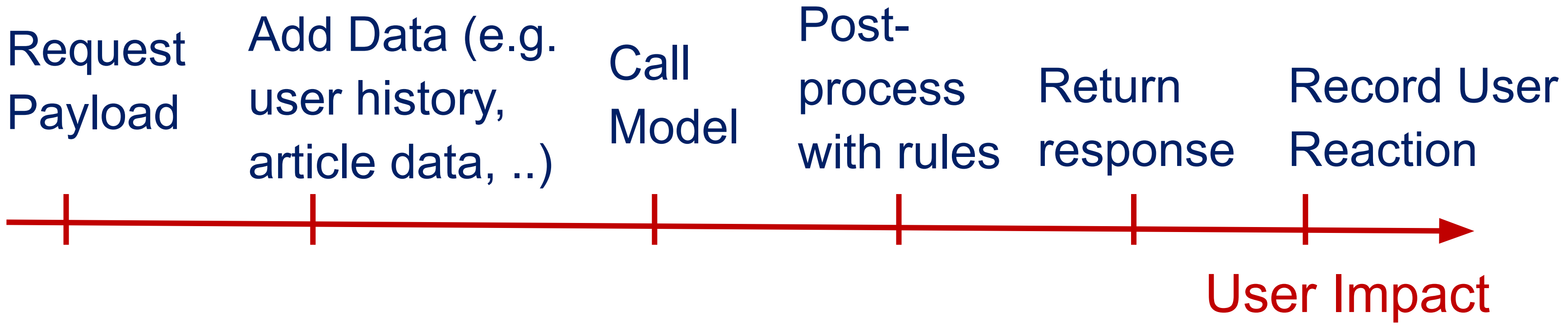
Prometheus Best Practices, https://prometheus.io/docs/practices/alerting/

Transfer Symptom-based Alerting to Machine Learning

Idea from SRE: we focus on **symptoms**, meaning **end-user pain**, not causes

→ Try to translate this to a machine learning service

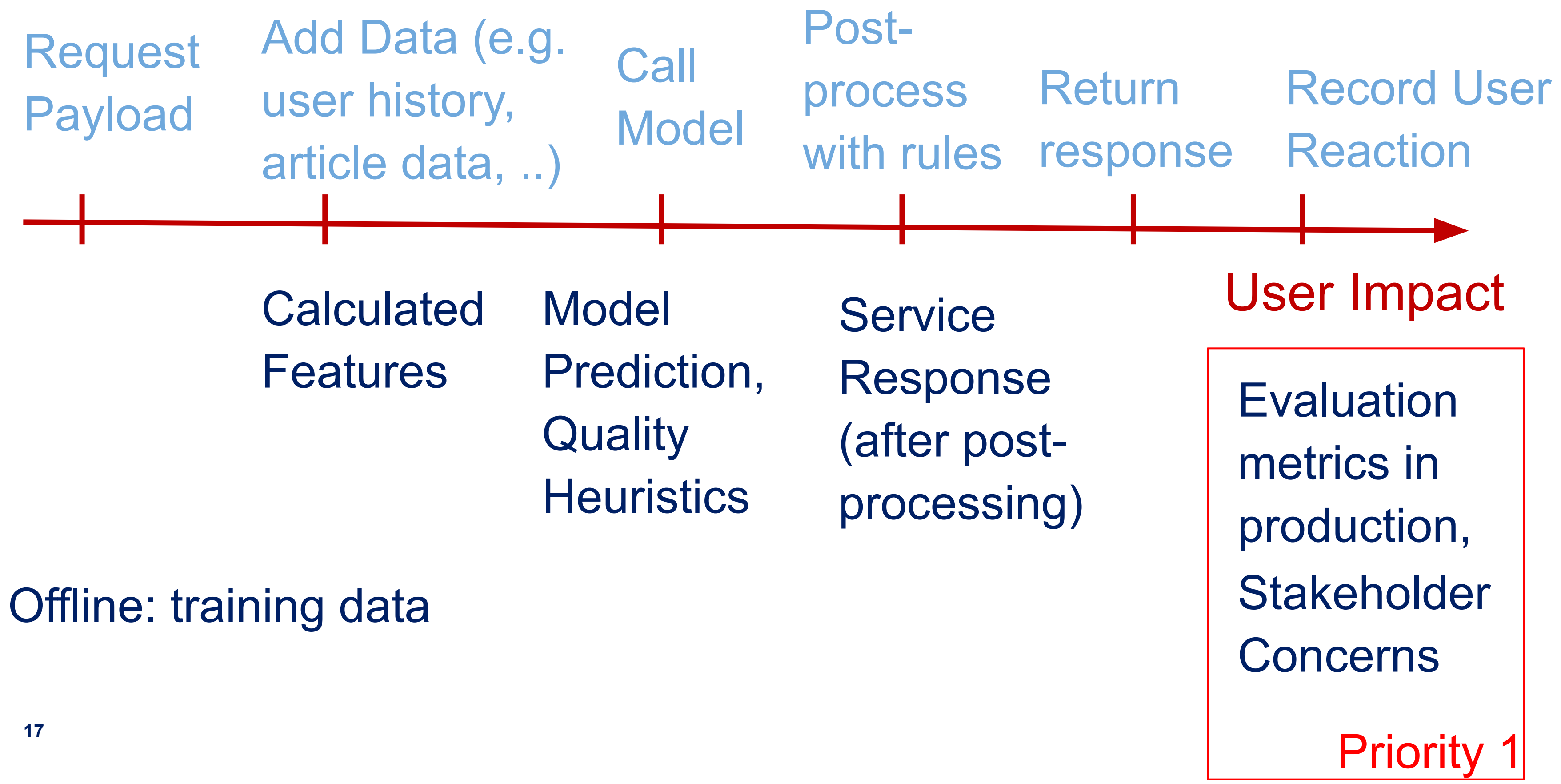
A typical Sequence of a Machine Learning Request



→ End-User-Pain is best detected in later steps, prioritize monitoring efforts on **outputs**

→ Problem: A lot of machine learning monitoring currently focused on inputs and data, not outputs 😱😱😱

Symptom based Monitoring: Prioritize backwards from Output


Request Payload
Add Data (e.g. user history, article data, ..)
Call Model
Post-process with rules
Return response
Record User Reaction
Calculated Features
Model Prediction, Quality Heuristics
Service Response (after post-processing)
User Impact
Evaluation metrics in production, Stakeholder Concerns
Priority 1
Offline: training data

Monitoring Priority 1: Evaluation Metrics in Production

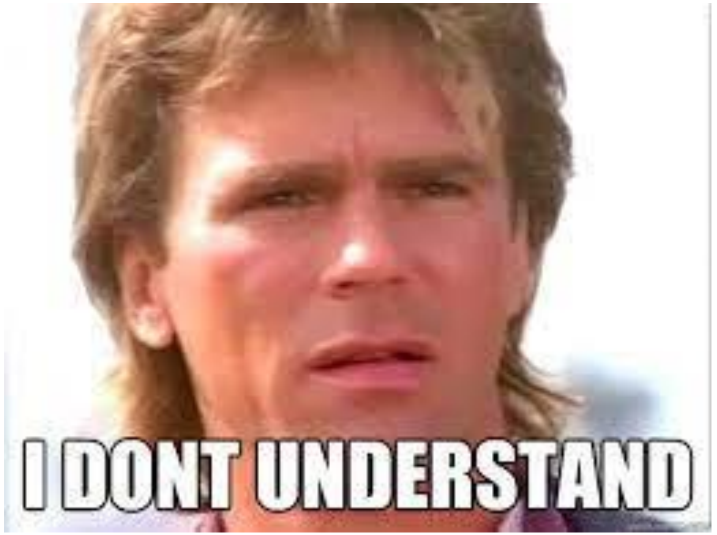
Backend Developers:

“ What are evaluation metrics?”

→ quality metrics used in training, e.g. Accuracy, precision/recall etc

Data Scientist:

“I can monitor my evaluation metrics in production??”



Monitoring Priority 1: Evaluation Metrics in Production

Answer: You can if you get ground truth result close in time, e.g. Food delivery knows the true delivery time after ~1h → calculate evaluation metrics like mean error

Common problems:

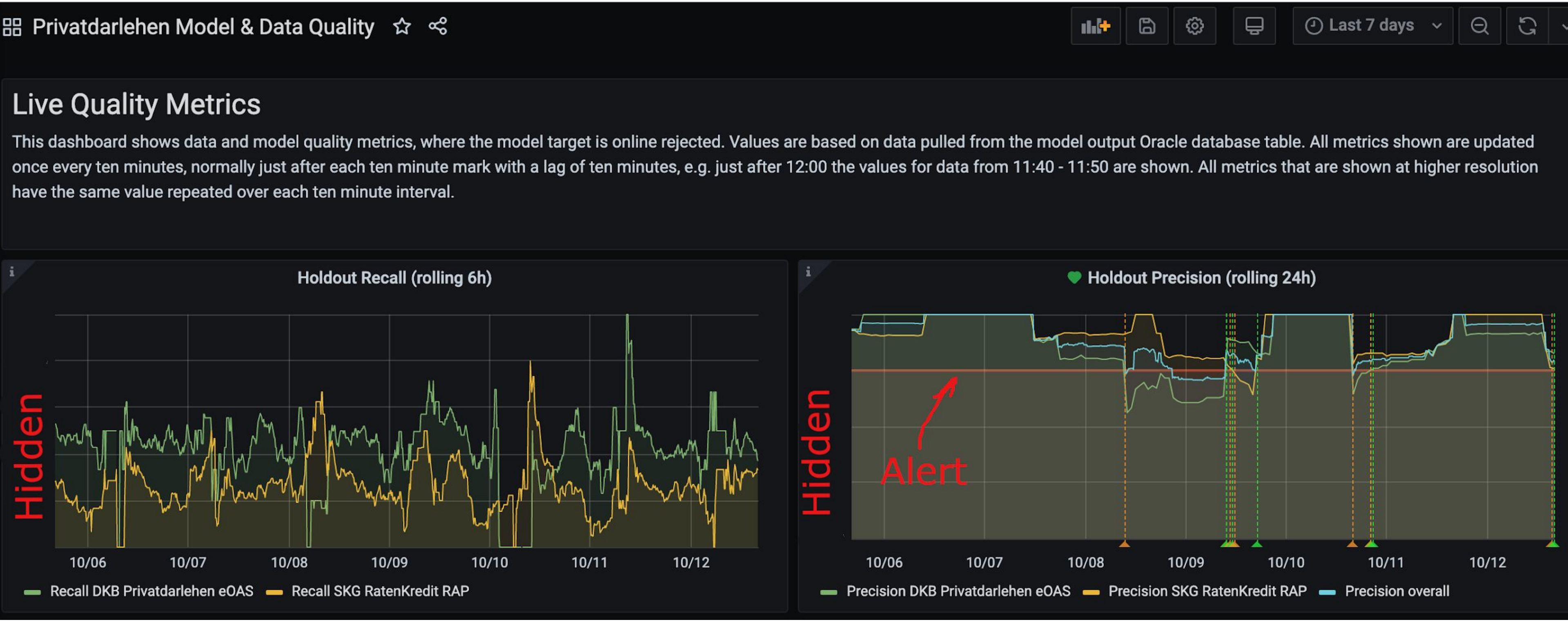
- Unknown result e.g. if a user is rejected because of a high fraud probability, we don‘t know if we made an error
- Delayed result, e.g. if we predict the delivery time for a package we know the true delivery time days later
- Filter bubble effect, e.g. algorithm decides what to show the customer. Unseen options cannot be evaluated

Monitoring Priority 1: Evaluation Metrics in Production

Implementation:

- Store prediction and true value
- Calculate the metrics used during model training, e.g. batch job and or create an endpoint to receive a feedback call
- Add metrics to dashboard and create an alert

Real-Time Dashboard: Evaluation Metrics in Production



Privatdarlehen Model & Data Quality

Last 7 days

Live Quality Metrics

This dashboard shows data and model quality metrics, where the model target is online rejected. Values are based on data pulled from the model output Oracle database table. All metrics shown are updated once every ten minutes, normally just after each ten minute mark with a lag of ten minutes, e.g. just after 12:00 the values for data from 11:40 - 11:50 are shown. All metrics that are shown at higher resolution have the same value repeated over each ten minute interval.

Holdout Recall (rolling 6h)
Hidden
10/06
10/07
10/08
10/09
10/10
10/11
10/12
Recall DKB Privatdarlehen eOAS
Recall SKG RatenKredit RAP
Holdout Precision (rolling 24h)
Hidden
Alert
10/06
10/07
10/08
10/09
10/10
10/11
10/12
Precision DKB Privatdarlehen eOAS
Precision SKG RatenKredit RAP
Precision overall

Monitoring Priority 1: Stakeholder Fear Signals

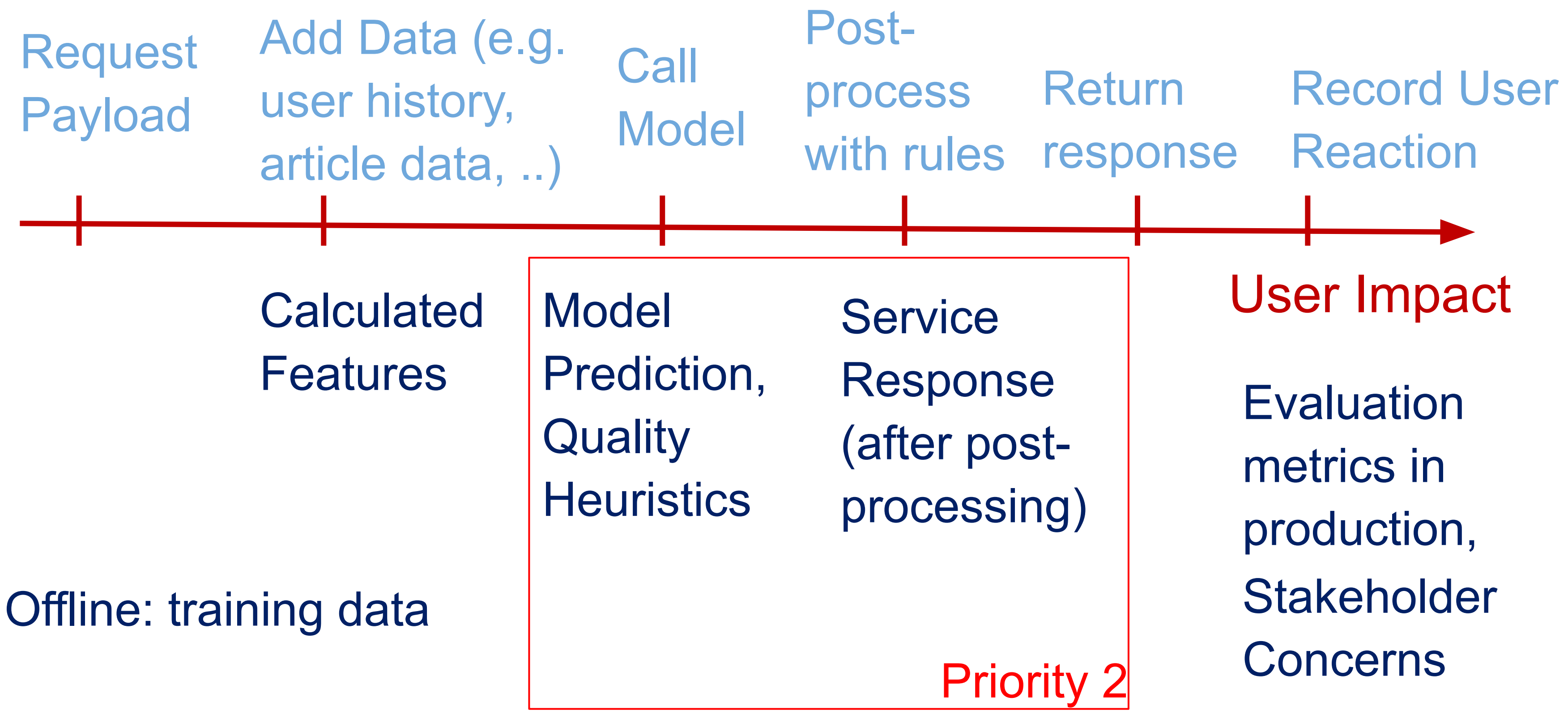
Monitor **what the stakeholders want to avoid**

- Machine Learning Applications need trust → ask stakeholders for their worst-case scenarios, e.g. service makes wrong decision, is uncertain, doesn't answer
- Put these fears into metrics to make sure you would detect these scenarios
- Add metrics to dashboard and alert

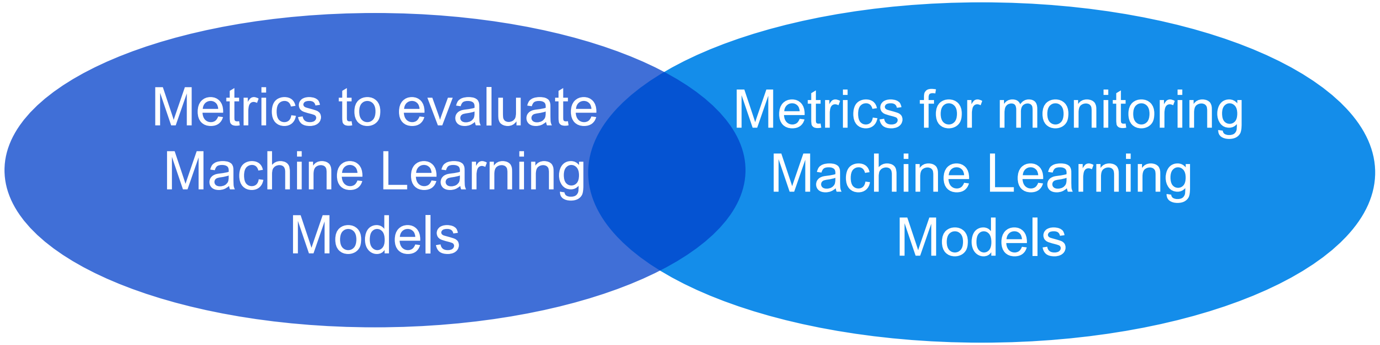
Example from a Loan Rejection Project:

- Fear: unfairly reject applications → alert on precision <95%
- Fear: Make application slow → alert on p95 speed <x msec

Symptom based monitoring: prioritize backwards from output


Request Payload
Add Data (e.g. user history, article data, ..)
Call Model
Post-process with rules
Return response
Record User Reaction
Calculated Features
Model Prediction, Quality Heuristics
Service Response (after post-processing)
User Impact
Evaluation metrics in production, Stakeholder Concerns
Offline: training data
Priority 2

Insight: A lot of Machine Learning Monitoring is done without the evaluation metrics


Metrics to evaluate Machine Learning Models
Metrics for monitoring Machine Learning Models

- Measured to evaluate **model quality**, e.g. precision, recall, NDCG, …
- To calculate evaluation metrics we compare the prediction against outcome in production
- Often not available or not available close in time

- Measured in order **detect** a problem, not to capture model quality
- Detection Metrics are easier to implement

Monitoring Priority 2: Service response distribution

Monitor the service response (after postprocessing rules)

Example outputs:

- classification: fraud/not fraud, things in an image (often with a prediction score)
- regression: forecasting (number)
- recommendations items (often with score)
- LLM summarization: text

…

→ monitor those outputs or metrics based on these outputs

Monitoring Priority 2: Response distribution

Monitor the response distribution

- good „catch all“ technique, needed if you cannot calculate evaluation metrics in production or if there is a delay between prediction and outcome
- Able to detect smaller changes compared to monitoring user reactions (more data, less noisy)
- Detect slow or sudden shifts of response distribution
- Often easy to do (just one or few outputs), real-time
- The importance of a change is more clear compared to input monitoring (an input field change might not be relevant to the output)

→ Rule Based Distance Metrics: Median, Quantiles, Share of empty/insufficient outputs

→ Statistical distance metrics: Kolmogorov-Smirnov Statistic, D1 Distance, Population Stability Index

Monitoring Priority 2: Response distribution

Example distribution distance metric: D1

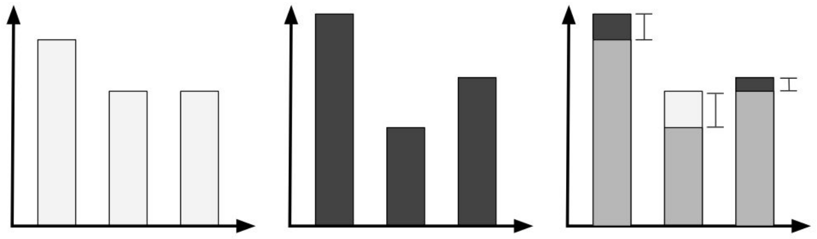


Figure 8: Two distributions, white and black are compared. When overlaying them, the difference can be "seen". The sum of the magnitude of these visible differences is the d_1 distance.

$$d_1(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

→ Sum of Distances of Probability Density Functions

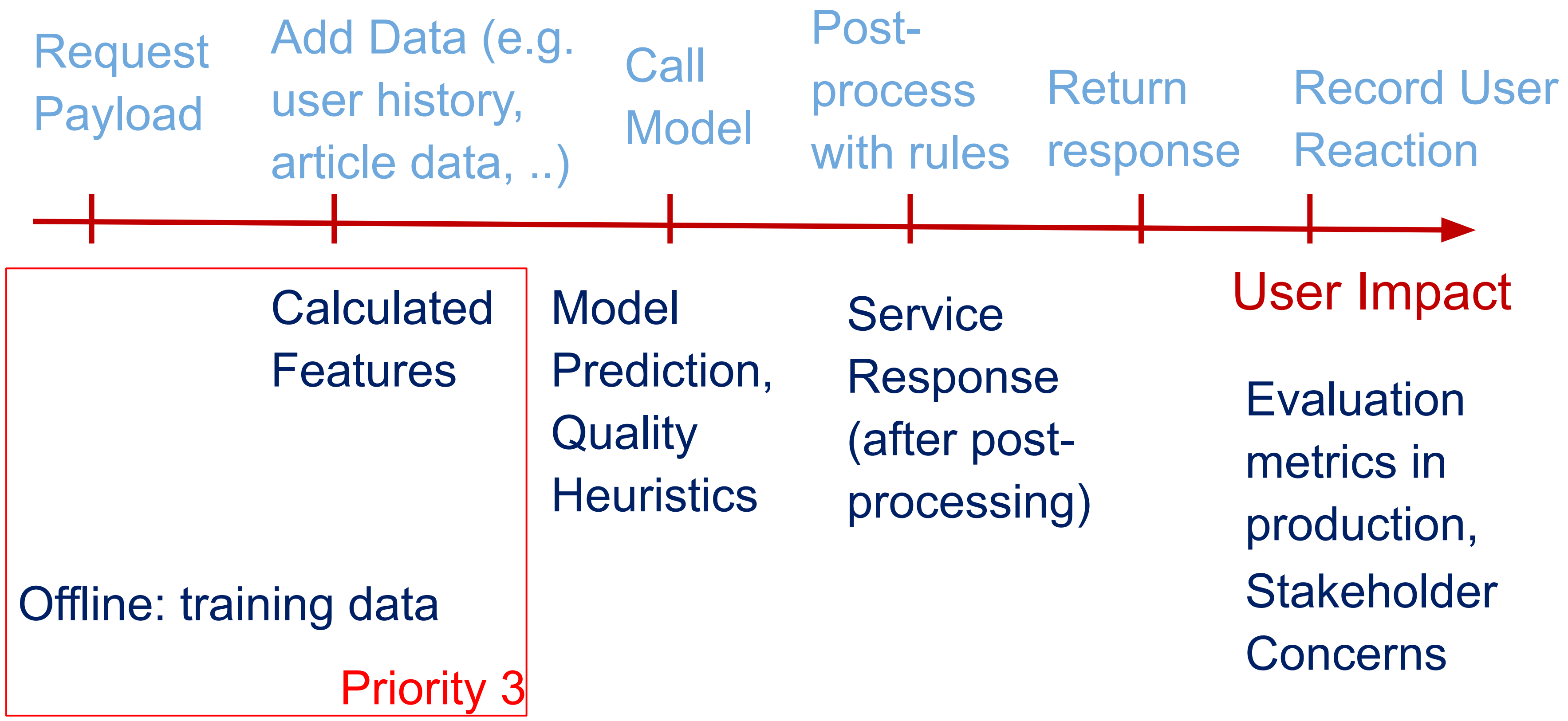
Source: Google Paper: Data Validation for Machine Learning 2019

Monitoring Priority 2: Response Heuristic Quality Metrics

Heuristic Quality Metrics based on the service response:

- Create use-case-specific, human understandable quality indicators, e.g. heuristic for a „really good“ or „bad“ response and common sense heuristics
- The metric doesn’t have to be a great quality indicator, just go down if quality goes down (do not aim to measure objective quality!)
- E.g.
  - Common-Sense-Metric for a personalized algorithm: Share of personalized responses
  - Metric for bad responses: Share of empty responses/fallback responses
  - Common-Sense-Metric for a personalized home page ranking: What is the rank of a user‘s most used carousel?

Symptom based monitoring: prioritize backwards from output



Request Payload

Add Data (e.g. user history, article data, ..)

Call Model

Post-process with rules

Return response

Record User Reaction

Calculated Features

Offline: training data

Priority 3

Model Prediction, Quality Heuristics

Service Response (after post-processing)

User Impact

Evaluation metrics in production, Stakeholder Concerns

Monitoring Priority 3: Input and Feature Data distribution

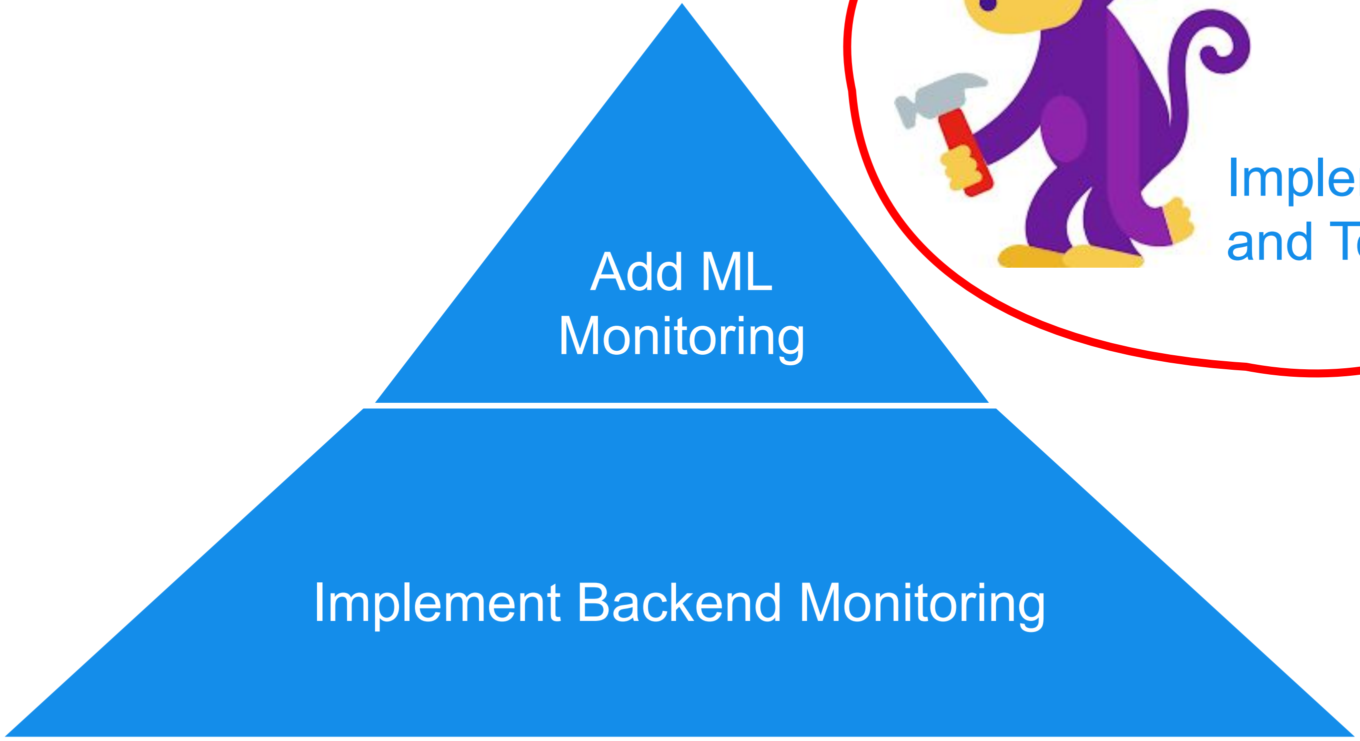
Monitoring Input and feature distribution

- Compare the serving distribution over time: a sudden shift indicates a problem
- Good for root cause analysis of an output oriented alert

→ same metrics as in priority 2

Exception, Prio 1: Compare difference between training and serving or train on features you logged (Google Rules of Machine Learning, Rule #29), there are always differences!

Agenda



Implementation and Tooling

Do I need an ML Ops Monitoring Tool?

"We introduced a machine learning observability tool and now we get several alerts per week that an input field's distribution changed. The reasons are mostly upstream business changes or unexplained changes in the input data. We did not take any action on the alerts."
Source: Data scientist at leading lending company

ML observability tools do not prioritize and solve ML monitoring:

- What are your requirements
- Who is addressing and owning alerts, can they debug inputs, model, outputs? (SRE, data scientists?)
- Clear data ownership and how to prioritize data issues?

Start simple, re-evaluate later

Pure Data Science Product, starting from scratch or very advanced product

→ evaluate full featured machine learning platforms

Your company offers services for monitoring and alerting, only some of your services are machine learning services

→ start with existing tools for metric collection and alerting, e.g. Prometheus, Grafana, a job scheduler

Grafana

Prometheus

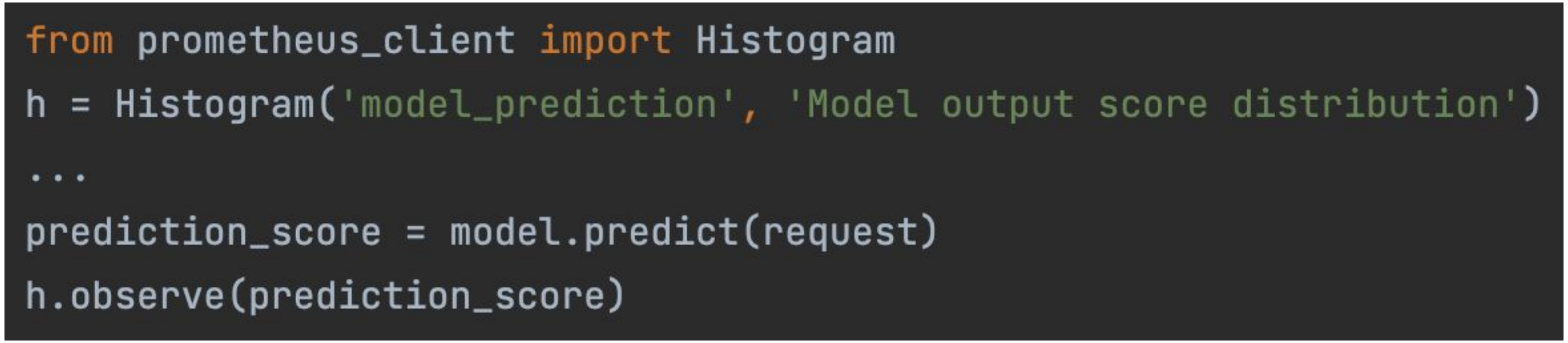
Machine Learning Monitoring: Use your existing stack

Advantages of using existing monitoring and alerting stack:

- No new tool(s) needed
- Immediate start
- Usually sufficient (unless you do a lot of ML debugging)
- Integration with other metrics on same dashboard
- Find out which metrics matter and if you need more visibility
- Time for team and org to grow into data and ml ops responsibility

Monitoring: Example Implementation

- Add metrics to your inference code:



```
from prometheus_client import Histogram
h = Histogram('model_prediction', 'Model output score distribution')
...
prediction_score = model.predict(request)
h.observe(prediction_score)
```

- For complicated calculations: log response to storage e.g. s3, run a script every 10 mins to calculate (raw) metric components

- Create a dashboard and create alerts

Takeaways

- Monitor golden signals + add machine learning monitoring
- Prioritize monitoring output metrics (user impact!) like response monitoring and if available evaluation metrics in production
- You often don’t need a new tool, use the tools you already have and add a few metrics

Talk to me

I am available for Consulting and Networking.

Join the ML Ops Slack Channel to talk to others working on Machine Learning in production.