Machine Learning Systems Design

Deployment and Monitoring

Lecture 21: Model Monitoring

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io

Agenda

3. Data Distribution Shifts

Data distribution shift

● Source distribution:	data the model is trained on
● Target distribution:	data the model runs inference on

Supervised learning: $P(X, Y)$

1. $P(X, Y) = P(Y|X)P(X)$
2. $P(X, Y) = P(X|Y)P(Y)$

Types of data distribution shifts

Type	**Meaning**	**Decomposition**
Covariate shift	● $P(X)$ changes ● $P(Y\|X)$ remains the same	$P(X, Y) = P(Y\|X)P(X)$
Label shift	● $P(Y)$ changes ● $P(X\|Y)$ remains the same	$P(X, Y) = P(X\|Y)P(Y)$
Concept drift	● $P(X)$ remains the same ● $P(Y\|X)$ changes	$P(X, Y) = P(Y\|X)P(X)$

Covariate shift

- P(X) changes
- P(Y|X) remains the same

- Statistics: a covariate is an independent variable that can influence the outcome of a given statistical trial.
- Supervised ML: input features are covariates

Covariate shift

- $P(X)$ changes
- $P(Y|X)$ remains the same

- Statistics: a covariate is an independent variable that can influence the outcome of a given statistical trial.
- Supervised ML: input features are covariates
- Input distribution changes, but for *a given input*, output is the same

Covariate shift: example

- P(X) changes
- P(Y|X) remains the same

- Predicts P(cancer | patient)
- P(age > 40): training > production
- P(cancer | age > 40): training = production

Covariate shift: causes (training)

- P(X) changes
- P(Y|X) remains the same

- Predicts P(cancer | patient)
- P(age > 40):
  - training > production
- P(cancer | age > 40):
  - training = production

- Data collection
  - E.g. people >40 are encouraged by doctors to get checkups
  - Closely related to sampling biases
- Training techniques
  - E.g. oversampling of rare classes
- Learning process
  - E.g. active learning

Covariate shift: causes (prod)

- P(X) changes
- P(Y|X) remains the same

Changes in environments

- Ex 1: P(convert to paid user | free user)
  - New marketing campaign attracting users from with higher income
    - P(high income) increases
    - P(convert to paid user | high level) remains the same

Covariate shift: causes (prod)

- P(X) changes
- P(Y|X) remains the same

Changes in environments

- Ex 2: P(Covid | coughing sound)
  - Training data from clinics, production data from phone recordings
    - P(coughing sound) changes
    - P(Covid | coughing sound) remains the same

Covariate shift

- Research: if knowing in advance how the production data will differ from training data, use importance weighting
- Production: unlikely to know how a distribution will change in advance

Label shift

- P(Y) changes
- P(X|Y) remains the same

- Output distribution changes but for *a given output*, input distribution stays the same.

Label shift & covariate shift

Predicts P(cancer | patient)	
P(age > 40):	training > production
P(cancer | age > 40):	training = production
P(cancer):	training > production
P(age > 40 | cancer):	training = prediction

- P(X) changes
- P(Y|X) remains the same

- P(Y) changes
- P(X|Y) remains the same

P(X) change often leads to P(Y) change, so covariate shift often means label shift

Label shift & covariate shift

- Predicts P(cancer | patient)
- New preventive drug: reducing P(cancer | patient) for all patients
- P(age > 40): training > production
- P(cancer | age > 40): training > production
- P(cancer): training > production
- P(age > 40 | cancer): training = prediction

- P(X) changes
- ~~P(Y|X) remains the same~~

- P(Y) changes
- P(X|Y) remains the same

Not all label shifts are covariate shifts!

Concept Drift

- P(X) remains the same
- P(Y|X) changes

- Same input, expecting different output
- P(houses in SF) remains the same
- Covid causes people to leave SF, housing prices drop
  - P($5M | houses in SF)
    - Pre-covid: high
    - During-covid: low

- P(X) remains the same
- P(Y|X) changes

Concept Drift

- Concept drifts can be cyclic & seasonal
  - Ride sharing demands high during rush hours, low otherwise
  - Flight ticket prices high during holidays, low otherwise

General data changes

- Feature change
    - A feature is added/removed/updated

General data changes

- Feature change
  - A feature is added/removed/updated
- Label schema change
  - Original: *{“POSITIVE”: 0, “NEGATIVE”: 1}*
  - New: *{“POSITIVE”: 0, “NEGATIVE”: 1, “NEUTRAL”: 2}*
  - both P(Y) and P(X|Y) change

Detecting data distribution shifts

How to determine that two distributions are different?

Detecting data distribution shifts

- The first idea might be to monitor your model's accuracy-related metrics
- In production, you don't always have access to labels, and even if you do, labels will be delayed.
- In research, there have been efforts to understand and detect label shifts without labels from the target distribution (Black Box Shift Estimation)
- The idea is that if the model is well-calibrated, meaning that its predicted probabilities match the true probabilities, then we can use its predictions as proxies for the labels
- However, in the industry, most drift detection methods focus on detecting changes in the input distribution.

Detecting data distribution shifts

How to determine that two distributions are different?

1. Compare statistics: mean, median, variance, quantiles, skewness, kurtosis, …
    - Compute these stats during training and compare these stats in production

Detecting data distribution shifts

How to determine that two distributions are different?

1. Compare statistics: mean, median, variance, quantiles, skewness, kurtosis, …
   - Not universal: only useful for distributions where these statistics are meaningful

Detecting data distribution shifts

How to determine that two distributions are different?

1. Compare statistics: mean, median, variance, quantiles, skewness, kurtosis, …
    - Not universal: only useful for distributions where these statistics are meaningful
    - Inconclusive: if statistics differ, distributions differ. If statistics are the same, distributions can still differ.

Detecting data distribution shifts

How to determine that two distributions are different?

1. Compare statistics: mean, median, variance, quantiles, skewness, kurtosis, ...
2. Two-sample hypothesis test
   - Determine whether the difference between two populations is statistically significant
   - If yes, likely from two distinct distributions

E.g.

1. Data from yesterday
2. Data from today

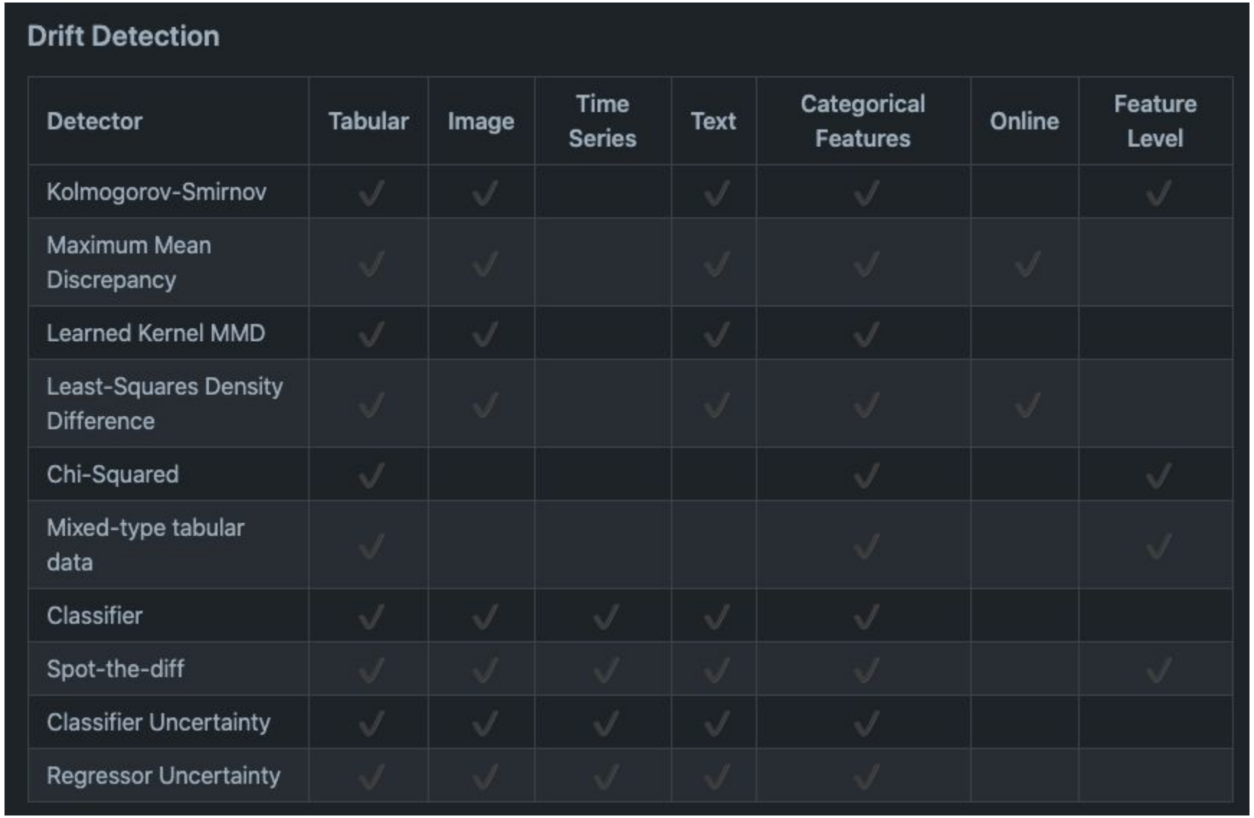
Two-sample test: KS test (Kolmogorov–Smirnov)

- Pros
  - Doesn't require any parameters of the underlying distribution
  - Doesn't make assumptions about distribution
- Cons
  - Only works with one-dimensional data

$\searrow$

- Useful for prediction & label distributions
- Not so useful for features

Two-sample test



Drift Detection

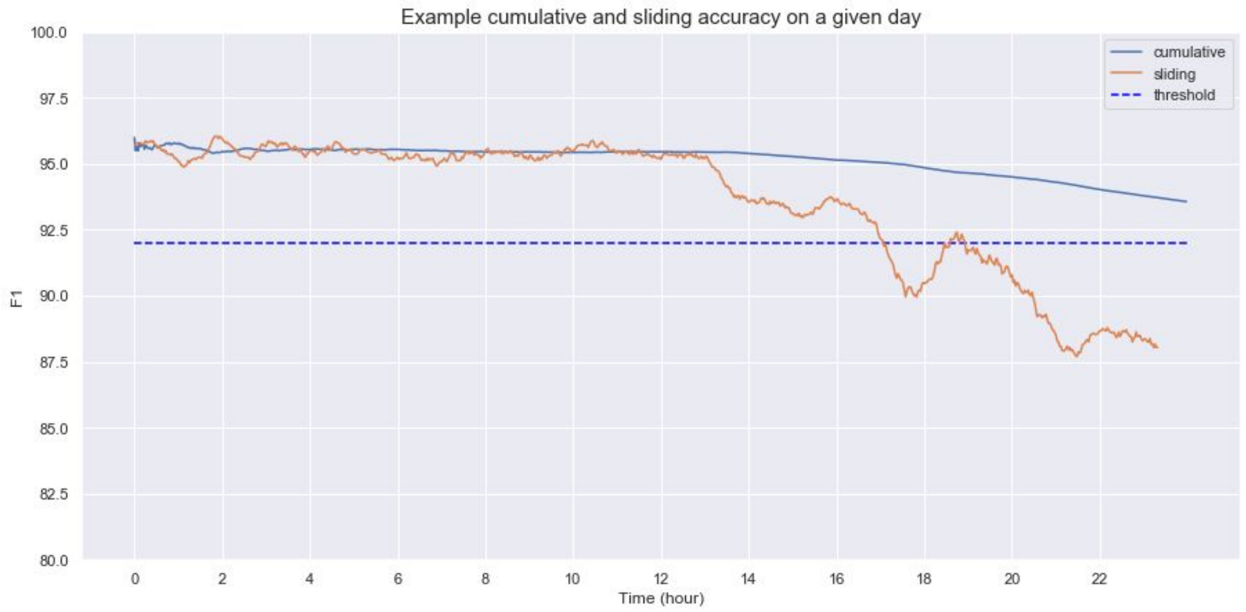
Detector	Tabular	Image	Time Series	Text	Categorical Features	Online	Feature Level
Kolmogorov-Smirnov	✓	✓		✓	✓		✓
Maximum Mean Discrepancy	✓	✓		✓	✓	✓	
Learned Kernel MMD	✓	✓		✓	✓		
Least-Squares Density Difference	✓	✓		✓	✓	✓	
Chi-Squared	✓				✓		✓
Mixed-type tabular data	✓				✓		✓
Classifier	✓	✓	✓	✓	✓		
Spot-the-diff	✓	✓	✓	✓	✓		✓
Classifier Uncertainty	✓	✓	✓	✓	✓		
Regressor Uncertainty	✓	✓	✓	✓	✓		

alibi-detect (OS)

Most tests work better on low-dim data, so dim reduction is recommended beforehand!

Cumulative vs. sliding metrics

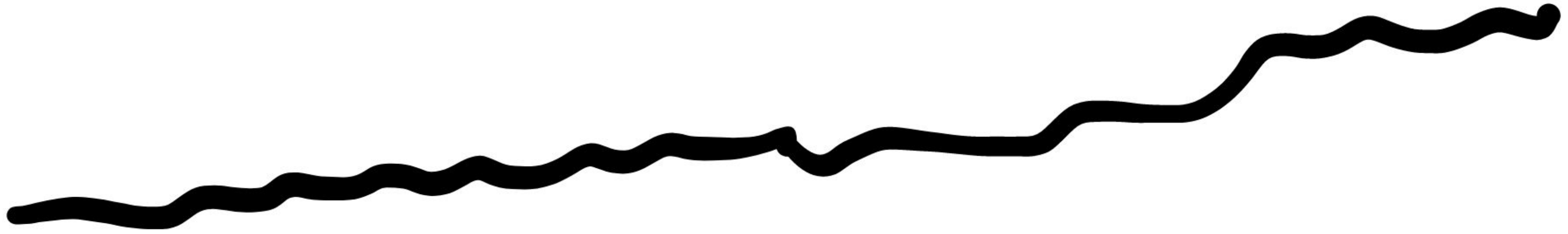
- Sliding: reset at each new time window



This image is based on an example from MadeWithML (Goku Mohandas).

Not all shifts are equal

- Sudden shifts vs. gradual shifts
    - Sudden shifts are easier to detect than gradual shifts



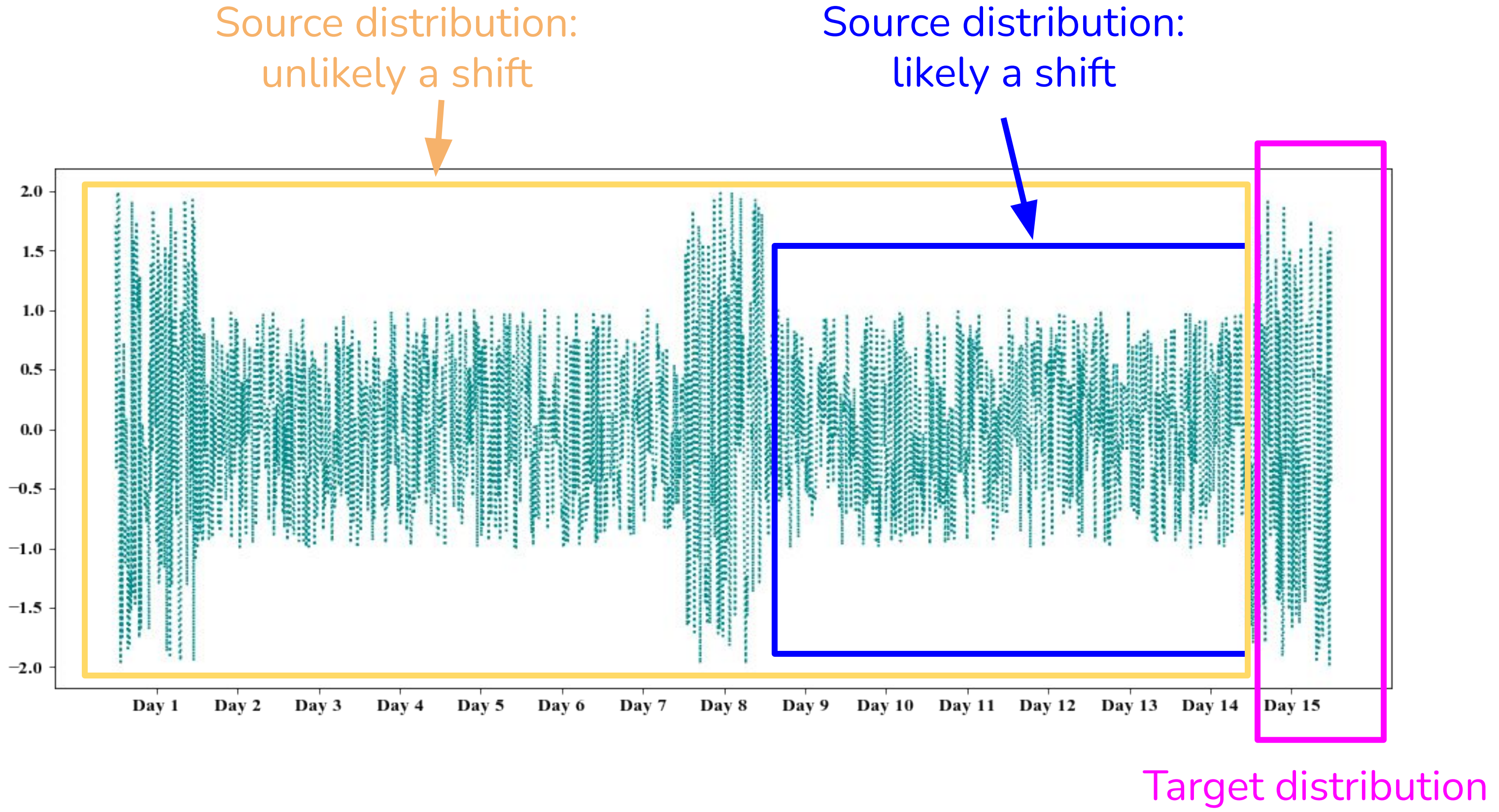
Not all shifts are equal

- Sudden shifts vs. gradual shifts
- Spatial shifts vs. temporal shifts

- New device (e.g. mobile vs. desktop)
- New users (e.g. new country)

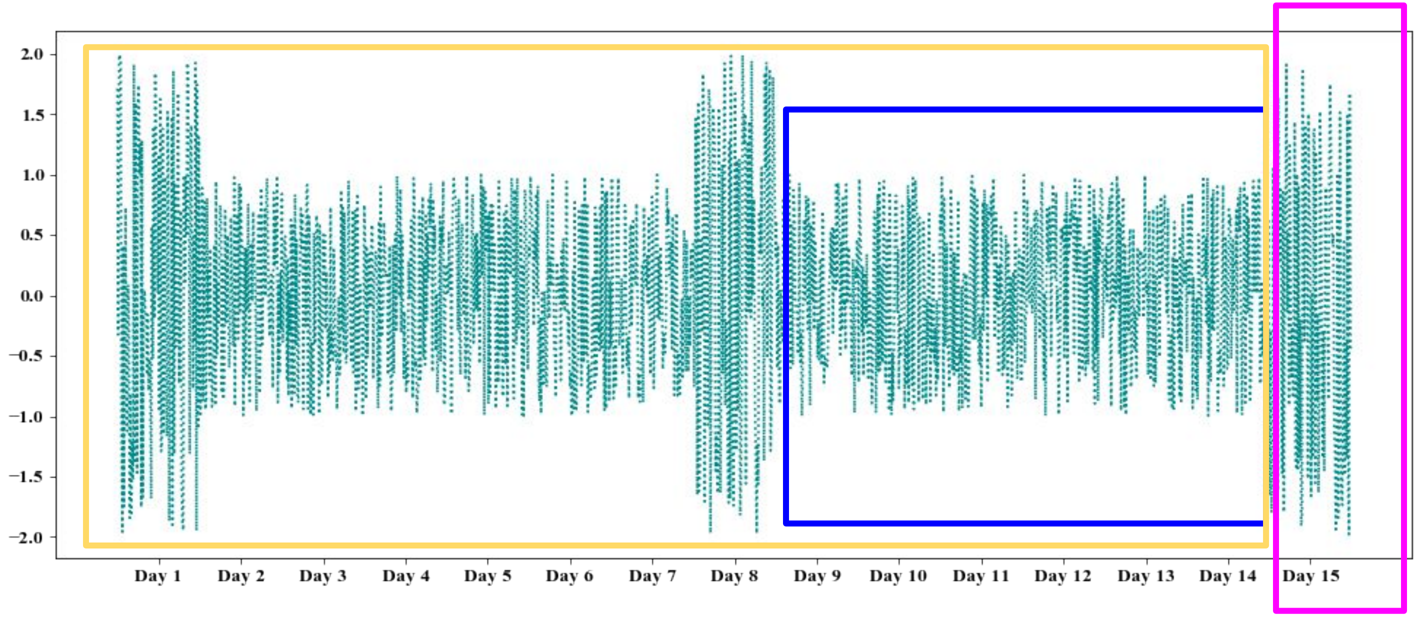
E.g. same users, same device, but behaviors change over time

Temporal shifts: time window scale matters



Temporal shifts: time window scale matters

Difficulty is compounded
by seasonal variation



[how to deal with the seasonality of a market]

Temporal shifts: time window scale matters

- Too short window: false alarms of shifts
- Too long window: takes long to detect shifts

  - Granularity level: hourly, daily

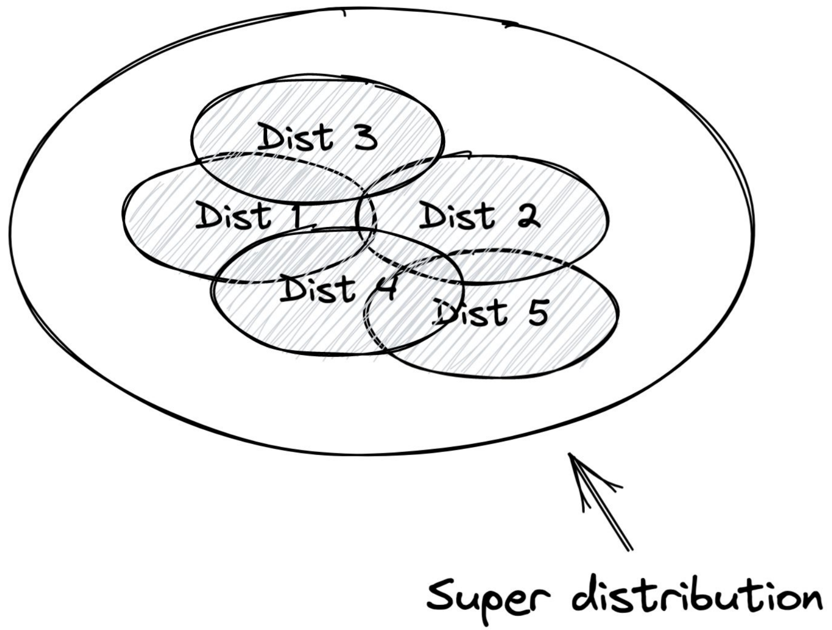
Temporal shifts: time window scale matters

- Too short window: false alarms of shifts
- Too long window: takes long to detect shifts

- Granularity level: hourly, daily
- Merge shorter time scale windows -> larger time scale window
- RCA: automatically analyze various window sizes

Addressing data distribution shifts

1. Train model using a massive dataset



Addressing data distribution shifts

1. Train model using a massive dataset
2. Retrain model with new data from new distribution
    - Mode
        - Train from scratch
        - Fine-tune

Addressing data distribution shifts

1. Train model using a massive dataset
2. Retrain model with new data from new distribution
    - Mode
    - Data
        - Use data from when data started to shift
        - Use data from the last X days/weeks/months
        - Use data form the last fine-tuning point

Need to figure out not just when to retrain models, but also how and what data

4. Monitoring and Observability

Monitoring vs. observability

- Monitoring: tracking, measuring, and logging different metrics that can help us determine when something goes wrong
- Observability: setting up our system in a way that gives us visibility into our system to investigate what went wrong

Monitoring vs. observability

- Monitoring: tracking, measuring, and logging different metrics that can help us determine when something goes wrong
- Observability: setting up our system in a way that gives us visibility into our system to investigate what went wrong

Instrumentation

- adding timers to your functions
- counting NaNs in your features
- logging unusual events e.g. very long inputs
- ...

Monitoring vs. observability

- Monitoring: tracking, measuring, and logging different metrics that can help us determine when something goes wrong
- Observability: setting up our system in a way that gives us visibility into our system to investigate what went wrong

Observability is part of monitoring

Monitoring is all about metrics

- Operational metrics
- ML-specific metrics

Operational metrics

- Latency
- Throughput
- Requests / minute/hour/day
- % requests that return with a 2XX code
- CPU/GPU utilization
- Memory utilization
- **Availability**
- etc.

Operational metrics

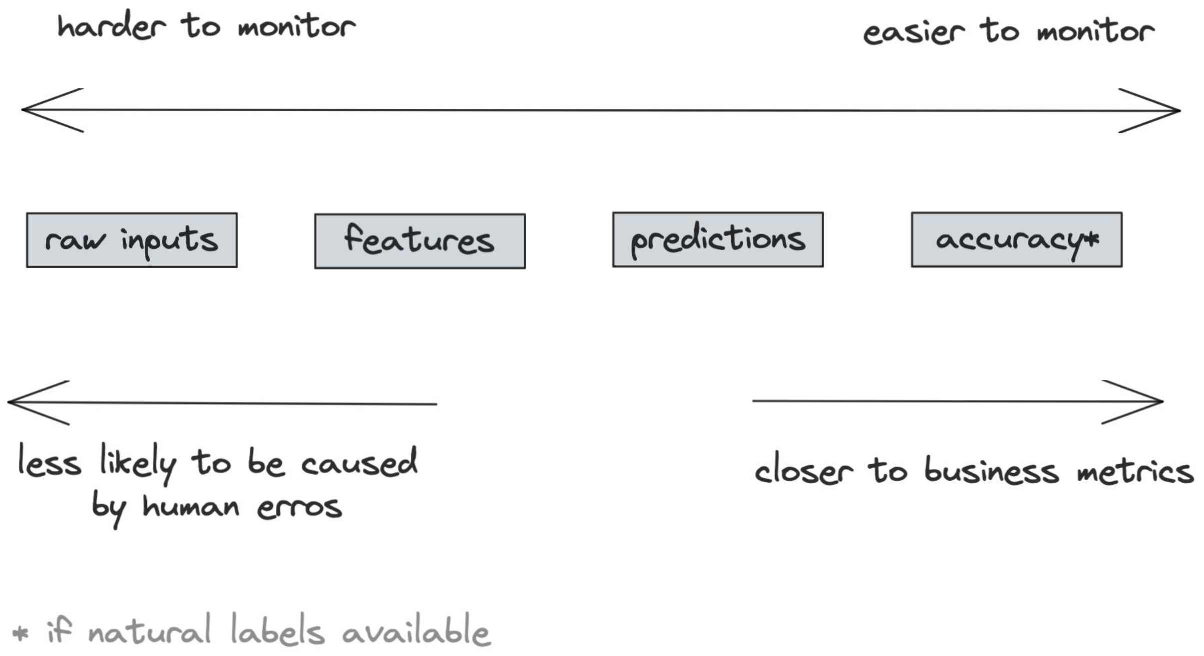
- Latency
- Throughput
- Requests / minute/hour/day
- % requests that return with a 2XX code
- CPU/GPU utilization
- Memory utilization
- **Availability**
- etc.

SLA example

- Up means:
  - median latency <200ms
  - 99th percentile <2s
- 99.99% uptime (four-nines)

SLA for ML?

ML metrics: what to monitor


harder to monitor
easier to monitor
raw inputs
features
predictions
accuracy*
less likely to be caused by human erros
closer to business metrics
* if natural labels available

Monitoring #1: accuracy-related metrics

- Most direct way to monitor a model’s performance
    - Can only do as fast as when feedback is available

Monitoring #1: accuracy-related metrics

- Most direct way to monitor a model’s performance
- Collect as much feedback as possible
- Example: YouTube video recommendations
    - Click through rate
    - Duration watched
    - Completion rate
    - Take rate

Monitoring #2: predictions

- Predictions are low-dim: easy to visualize, compute stats, and do two-sample tests
- Changes in prediction dist. generally mean changes in input dist.

Monitoring #2: predictions

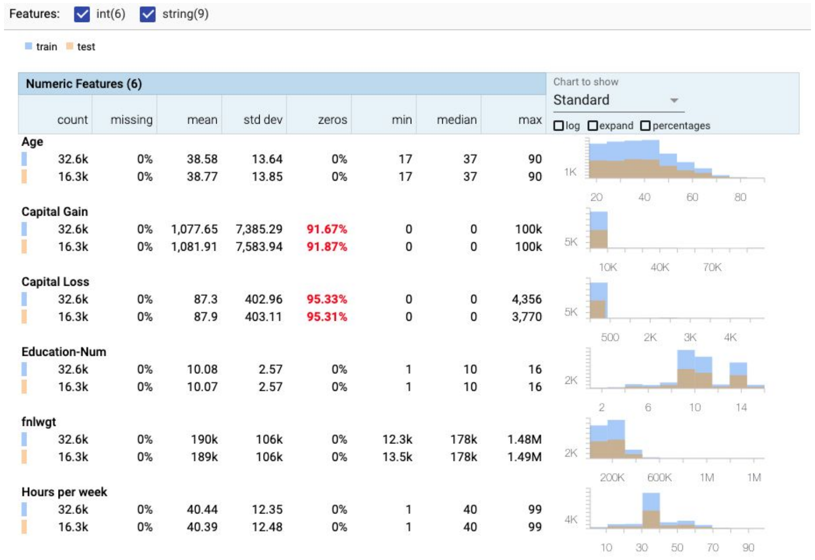
- Predictions are low-dim: easy to visualize, compute stats, and do two-sample tests
- Changes in prediction dist. generally mean changes in input dist.
- Monitor odd things in predictions
    - E.g. if predictions are all False in the last 10 mins

Monitoring #3: features

- Most monitoring tools focus on monitoring features
- Feature schema expectations
    - Generated from the source distribution
    - If violated in production, possibly something is wrong
- Example expectations
    - Common sense: e.g. “the” is most common word in English
    - min, max, or median values of a feature are in [a, b]
    - All values of a feature satisfy a regex
    - Categorical data belongs to a predefined set
    - FEATURE_1 > FEATURE_B

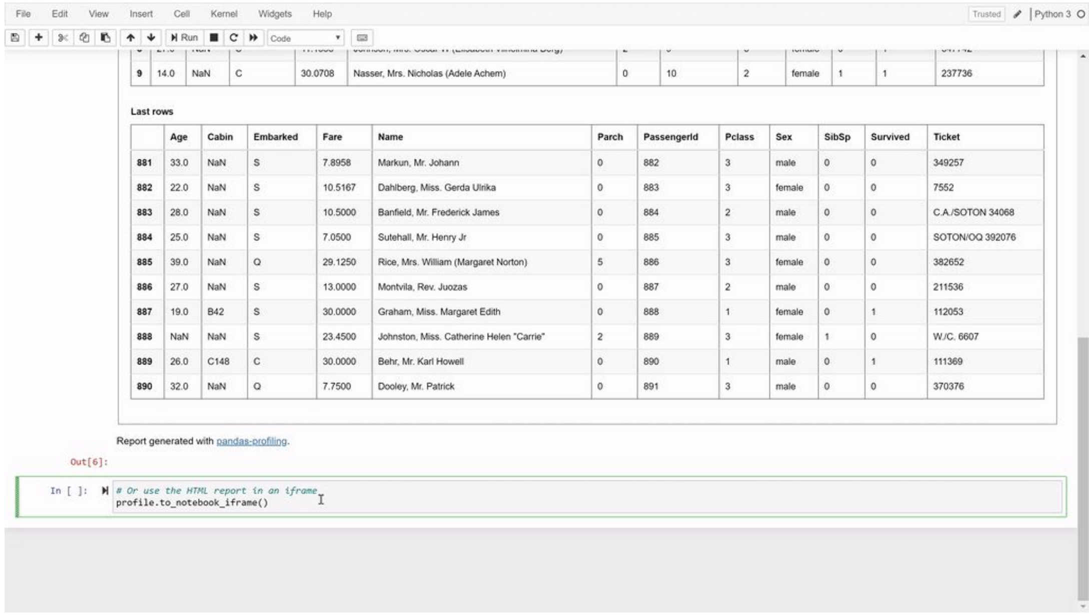
Generate expectations with profiling & visualization

- Examining data & collecting:
  - statistics
  - informative summaries
- [pandas-profiling]
- [facets]



Numeric Features (6)

	count	missing	mean	std dev	zeros	min	median	max
Age								
	32.6k	0%	38.58	13.64	0%	17	37	90
	16.3k	0%	38.77	13.85	0%	17	37	90
Capital Gain								
	32.6k	0%	1,077.65	7,385.29	91.67%	0	0	100k
	16.3k	0%	1,081.91	7,583.94	91.87%	0	0	100k
Capital Loss								
	32.6k	0%	87.3	402.96	95.33%	0	0	4,356
	16.3k	0%	87.9	403.11	95.31%	0	0	3,770
Education-Num								
	32.6k	0%	10.08	2.57	0%	1	10	16
	16.3k	0%	10.07	2.57	0%	1	10	16
fnlwgt								
	32.6k	0%	190k	106k	0%	12.3k	178k	1.48M
	16.3k	0%	189k	106k	0%	13.5k	178k	1.49M
Hours per week								
	32.6k	0%	40.44	12.35	0%	1	40	99
	16.3k	0%	40.39	12.48	0%	1	40	99



9	14.0	NaN	C	30.0708	Nasser, Mrs. Nicholas (Adele Achem)	0	10	2	female	1	1	237736

Last rows

	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket
881	33.0	NaN	S	7.8958	Markun, Mr. Johann	0	882	3	male	0	0	349257
882	22.0	NaN	S	10.5167	Dahlberg, Miss. Gerda Ulrika	0	883	3	female	0	0	7552
883	28.0	NaN	S	10.5000	Banfield, Mr. Frederick James	0	884	2	male	0	0	C.A./SOTON 34068
884	25.0	NaN	S	7.0500	Sutehall, Mr. Henry Jr	0	885	3	male	0	0	SOTON/OQ 392076
885	39.0	NaN	Q	29.1250	Rice, Mrs. William (Margaret Norton)	5	886	3	female	0	0	382652
886	27.0	NaN	S	13.0000	Montvila, Rev. Juozas	0	887	2	male	0	0	211536
887	19.0	B42	S	30.0000	Graham, Miss. Margaret Edith	0	888	1	female	0	1	112053
888	NaN	NaN	S	23.4500	Johnston, Miss. Catherine Helen "Carrie"	2	889	3	female	1	0	W./C. 6607
889	26.0	C148	C	30.0000	Behr, Mr. Karl Howell	0	890	1	male	0	1	111369
890	32.0	NaN	Q	7.7500	Dooley, Mr. Patrick	0	891	3	male	0	0	370376

Report generated with pandas-profiling.

```
Out[6]:

In [ ]:  # Or use the HTML report in an iframe
         profile.to_notebook_iframe()
```

Monitoring #3: features

- Feature schema expectations

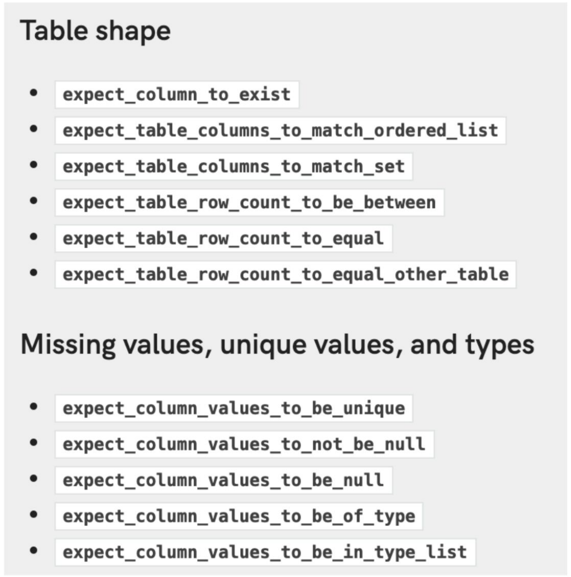


Table shape

- `expect_column_to_exist`
- `expect_table_columns_to_match_ordered_list`
- `expect_table_columns_to_match_set`
- `expect_table_row_count_to_be_between`
- `expect_table_row_count_to_equal`
- `expect_table_row_count_to_equal_other_table`

Missing values, unique values, and types

- `expect_column_values_to_be_unique`
- `expect_column_values_to_not_be_null`
- `expect_column_values_to_be_null`
- `expect_column_values_to_be_of_type`
- `expect_column_values_to_be_in_type_list`

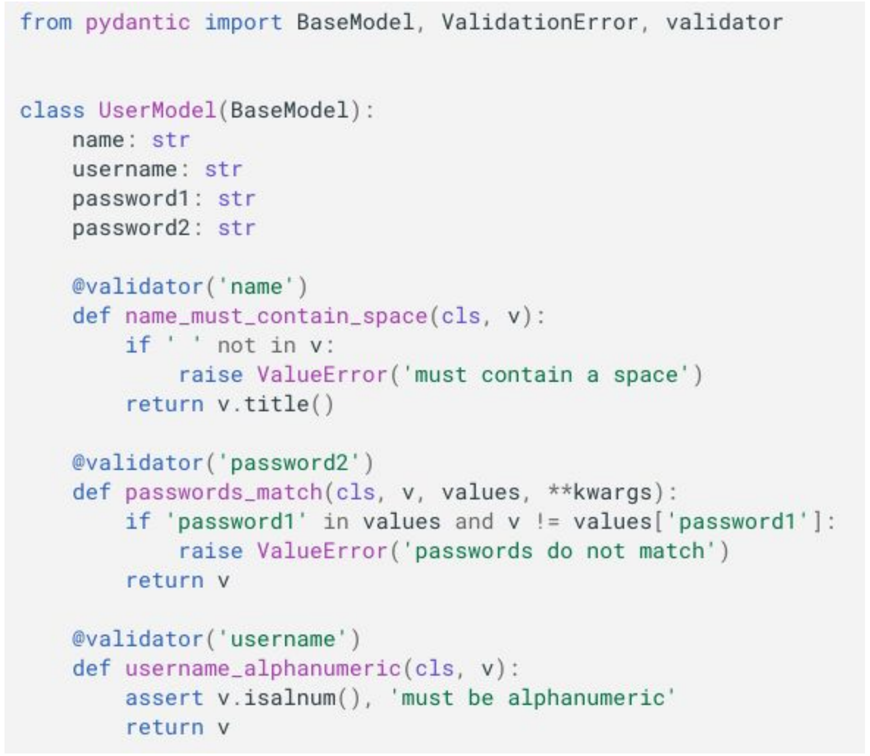
```
expect_column_values_to_be_between(
    column="room_temp",
    min_value=60,
    max_value=75,
    mostly=.95
)
```

“Values in this column should be between 60 and 75, at least 95% of the time.”

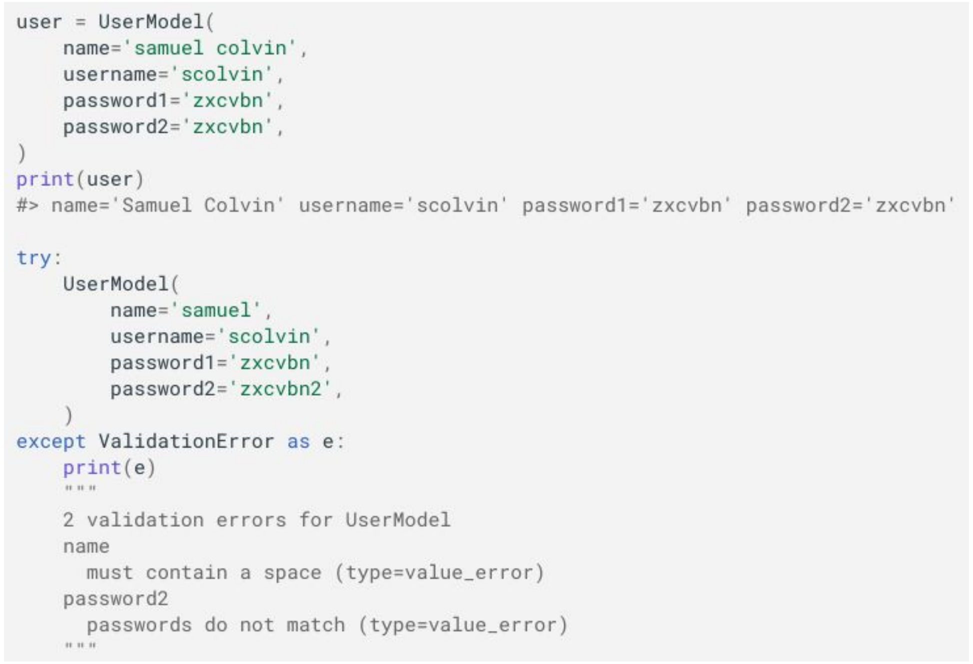
“Warning: more than 5% of values fell outside the specified range of 60 to 75.”

GitHub - great-expectations/great_expectations

Monitoring #3: features schema with pydantic



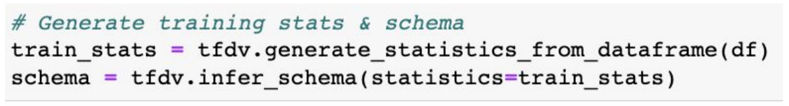
```python
from pydantic import BaseModel, ValidationError, validator


class UserModel(BaseModel):
    name: str
    username: str
    password1: str
    password2: str

    @validator('name')
    def name_must_contain_space(cls, v):
        if ' ' not in v:
            raise ValueError('must contain a space')
        return v.title()

    @validator('password2')
    def passwords_match(cls, v, values, **kwargs):
        if 'password1' in values and v != values['password1']:
            raise ValueError('passwords do not match')
        return v

    @validator('username')
    def username_alphanumeric(cls, v):
        assert v.isalnum(), 'must be alphanumeric'
        return v
```



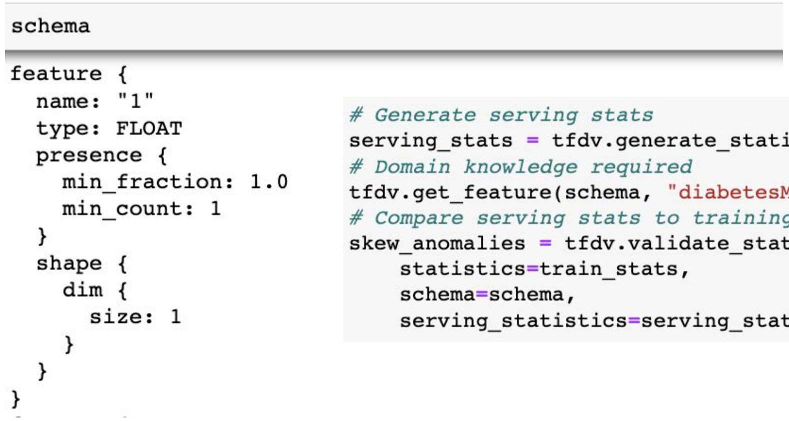
```python
user = UserModel(
    name='samuel colvin',
    username='scolvin',
    password1='zxcvbn',
    password2='zxcvbn',
)
print(user)
#> name='Samuel Colvin' username='scolvin' password1='zxcvbn' password2='zxcvbn'

try:
    UserModel(
        name='samuel',
        username='scolvin',
        password1='zxcvbn',
        password2='zxcvbn2',
    )
except ValidationError as e:
    print(e)
    """
    2 validation errors for UserModel
    name
      must contain a space (type=value_error)
    password2
      passwords do not match (type=value_error)
    """
```

https://pydantic-docs.helpmanual.io/usage/validators/

Monitoring #3: features schema with TFX

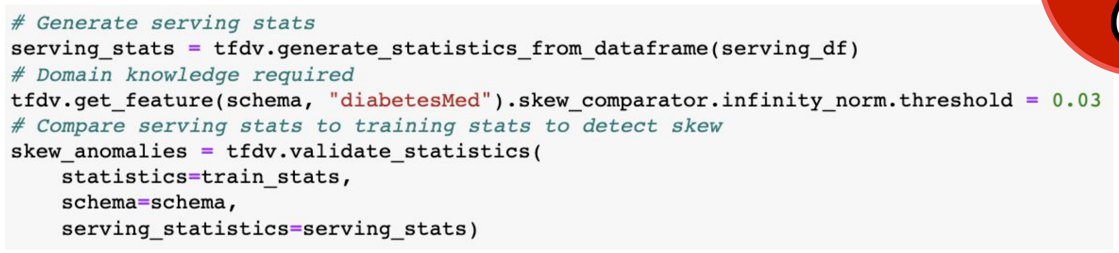


```
# Generate training stats & schema
train_stats = tfdv.generate_statistics_from_dataframe(df)
schema = tfdv.infer_schema(statistics=train_stats)
```

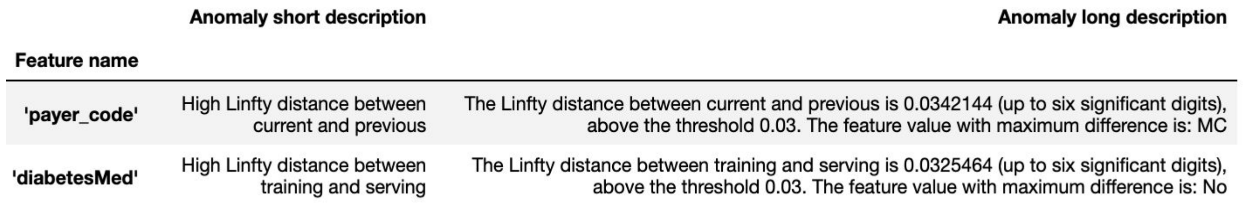


```
schema
```

```
feature {
  name: "1"
  type: FLOAT
  presence {
    min_fraction: 1.0
    min_count: 1
  }
  shape {
    dim {
      size: 1
    }
  }
}
```



```
# Generate serving stats
serving_stats = tfdv.generate_statistics_from_dataframe(serving_df)
# Domain knowledge required
tfdv.get_feature(schema, "diabetesMed").skew_comparator.infinity_norm.threshold = 0.03
# Compare serving stats to training stats to detect skew
skew_anomalies = tfdv.validate_statistics(
    statistics=train_stats,
    schema=schema,
    serving_statistics=serving_stats)
```



	Anomaly short description	Anomaly long description
Feature name		
'payer_code'	High Linfty distance between current and previous	The Linfty distance between current and previous is 0.0342144 (up to six significant digits), above the threshold 0.03. The feature value with maximum difference is: MC
'diabetesMed'	High Linfty distance between training and serving	The Linfty distance between training and serving is 0.0325464 (up to six significant digits), above the threshold 0.03. The feature value with maximum difference is: No

Feature monitoring problems

1. Compute & memory cost
    a. 100s models, each with 100s features
    b. Computing stats for 10000s of features is costly

Feature monitoring problems

1. Compute & memory cost
2. Alert fatigue
    a. Most expectation violations are benign

Feature monitoring problems

1. Compute & memory cost
2. Alert fatigue
3. Schema management
    a. Feature schema changes over time
    b. Need to find a way to map feature to schema version

Monitoring #4: raw inputs

- The way many ML workflows are set up today also makes it impossible for ML engineers to get direct access to raw input data, as the raw input data is often managed by a data platform team
- Monitoring raw inputs is often a responsibility of the data platform team, not the data science or ML team.

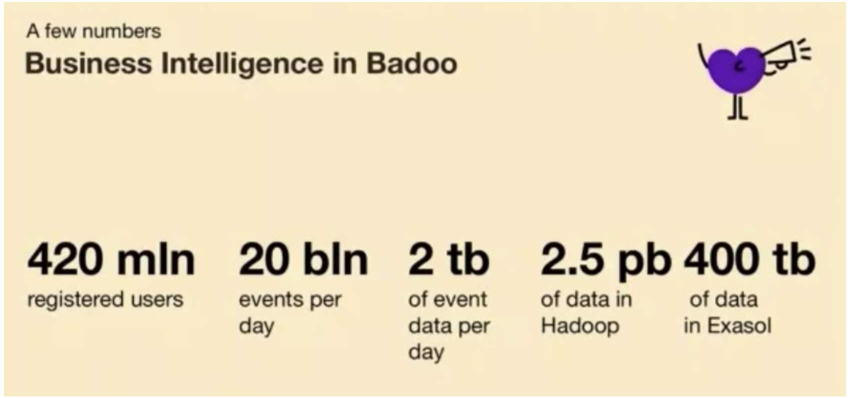
Monitoring toolbox

Measuring, tracking, and interpreting metrics for complex systems is a nontrivial task, and engineers rely on a set of tools to help them do so.

- Logs
- Dashboards
- Alerts

Monitoring toolbox: logs

- Log everything
- A stream processing problem
- Many companies process logs in batch



A few numbers

Business Intelligence in Badoo

420 mln	20 bln	2 tb	2.5 pb	400 tb
registered users	events per day	of event data per day	of data in Hadoop	of data in Exasol

Vladimir Kazanov ([Badoo 2019]())

"If it moves, we track it. Sometimes we'll draw a graph of something that isn't moving yet, just in case it decides to make a run for it."

Ian Malpass ([Etsy 2011]())

Monitoring toolbox: logs (tracing)

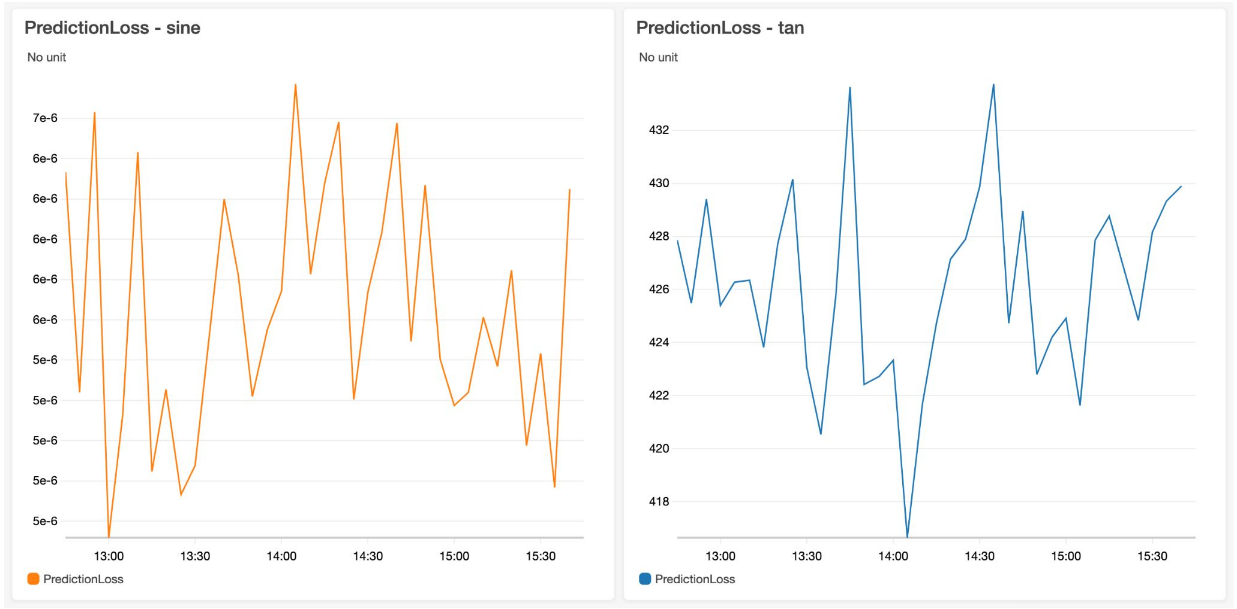
- In the early days of software deployment, an application might be one single service.
- But today, A request may do 20–30 hops from when it’s sent until when a response is received. The hard part might not be in detecting when something happened, but where the problem was
- **Distributed tracing:** in microservice architecture, give each process a unique ID so that, when something goes wrong, the error message contain that ID.

Monitoring toolbox: logs (big data)

- Because logs have grown so large and so difficult to manage, there have been many tools developed to help companies manage and analyze logs.
- Analyzing billions of logged events manually is futile, so many companies use ML to analyze logs
  - anomaly detection
  - when a service fails find the probability of related services being affected

Monitoring toolbox: dashboards

- Make monitoring accessible to non-engineering stakeholders
- Good for visualizations but insufficient for discovering distribution shifts



Graphs are useful for making sense of numbers, but they aren't sufficient

Monitoring toolbox: dashboards

- Excessive metrics on a dashboard can also be counterproductive, a phenomenon known as dashboard rot.
- It’s important to pick the right metrics or abstract out lower-level metrics to compute higher-level signals that make better sense for your specific tasks.

Monitoring toolbox: alerts

- 3 components of a good alerting system
  - Alert policy: condition for alert

be notified when a model's accuracy is under 90%, or that the HTTP response latency is higher than a second for at least 10 minutes.

Monitoring toolbox: alerts

- 3 components of a good alerting system
  - Alert policy: condition for alert
  - Notification channels

you might configure your alerts to be sent to an email address such as mlops-monitoring@*[your company email domain]*, or to post to a Slack channel such as #mlops-monitoring or to PagerDuty.

Monitoring toolbox: alerts

- 3 components of a good alerting system
  - Alert policy: condition for alert
  - Notification channels
  - Description

```
## Recommender model accuracy below 90%

        ${timestamp}: This alert originated from
the service ${service-name}
```

Monitoring toolbox: alerts

- 3 components of a good alerting system
  - Alert policy: condition for alert
  - Notification channels
  - Description
- Alert fatigue
  - How to send only meaningful alerts?

Observability

Better visibility into understanding the complex behavior of software using [outputs] collected from the system at run time.

- Simple old softwares: monitoring
- Complicated today softwares: observability

Observability: how to see inside your ML system

- Imagine you have a black box that does something amazing, like predicting the weather or playing chess. But sometimes it fails or makes mistakes, and you don't know why. How do you fix it?
- You could try to open the box and look inside, but that might break it or take too long. You could also try to change the box's code, but that might make things worse or introduce new bugs.
- A better way is to use observability: a way to see inside the box by looking at what comes out of it. Observability lets you collect and analyze the box's outputs, like its predictions, errors, logs, and metrics.
- **Observability is better than monitoring**, which only looks at some outputs and may not tell you enough about the problem. Observability lets you ask detailed questions about the box's outputs, such as who, what, when, where, and why.

Observability and interpretability in ML

- But observability is not enough. You also need to understand how the box works and why it does what it does. This is called interpretability: a way to explain the box's logic and reasoning.
- Interpretability helps you find out which inputs or features are important for the box and how they affect its outputs. For example, if the box predicts the weather, you might want to know which factors influence its forecasts, like temperature, humidity, wind speed, etc.
- Interpretability helps you fix problems with the box when its performance drops or when it makes mistakes. For example, if the box plays chess, you might want to know why it lost a game or made a bad move.
- Observability and interpretability are part of **continual learning**: a way to update the box to adapt to changes in data or environment. For example, if the box predicts the weather, you might want to update it when the seasons change or when there is a storm.

Monitoring -> Continual Learning

- Monitoring is passive
  - Wait for a shift to happen to detect it
- Continual learning is active
  - Update your models to address shifts

Machine Learning Systems Design

Deployment and Monitoring

Next Lecture: Model Online Evaluation

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io