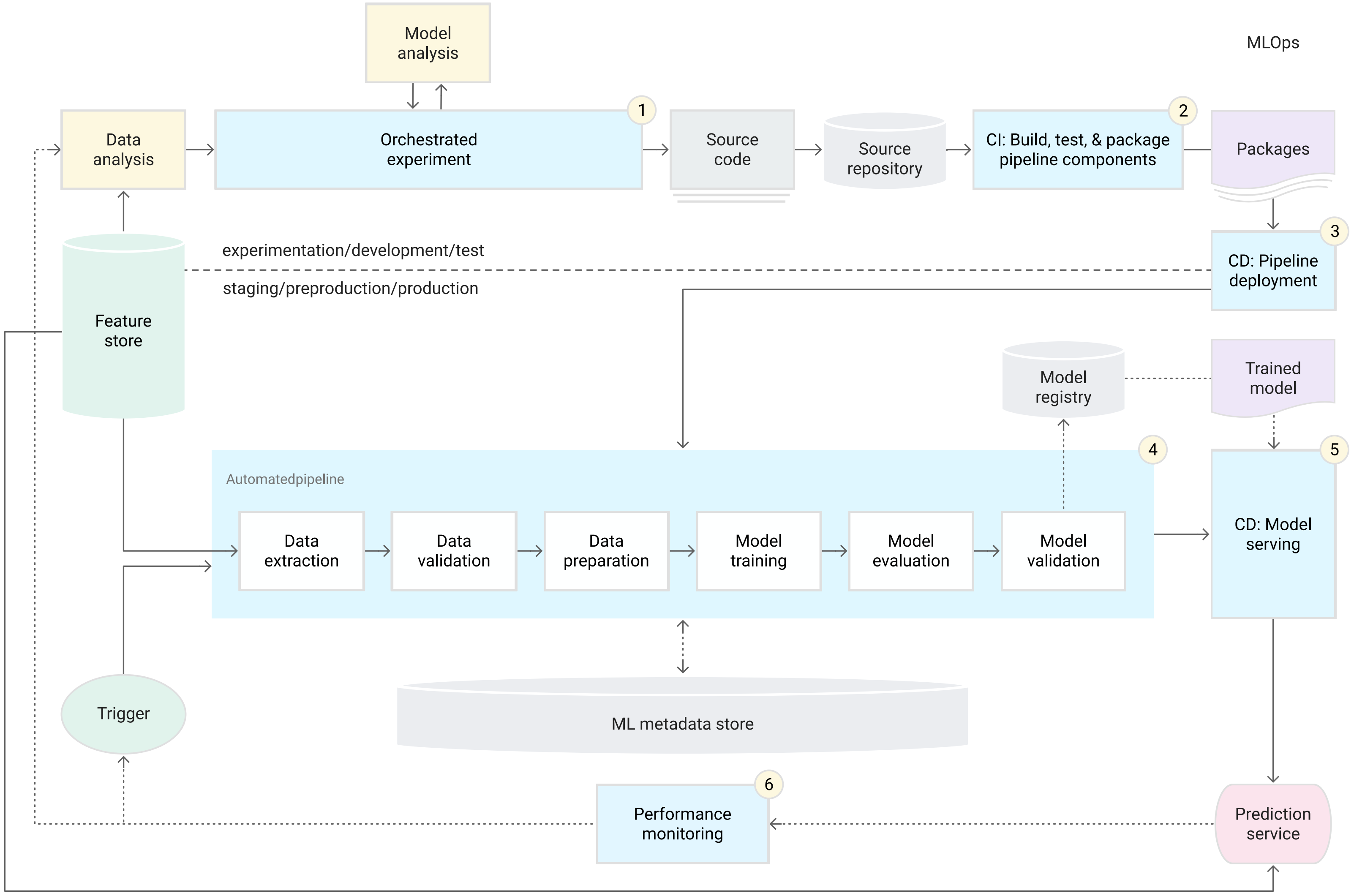
Мониторинг

Семинар 10

Recap MLOps


MLOps
Model analysis
Data analysis
Orchestrated experiment
1
Source code
Source repository
CI: Build, test, & package pipeline components
2
Packages
Feature store
experimentation/development/test
staging/preproduction/production
CD: Pipeline deployment
3
Model registry
Trained model
Automatedpipeline
4
Data extraction
Data validation
Data preparation
Model training
Model evaluation
Model validation
CD: Model serving
5
Trigger
ML metadata store
Performance monitoring
6
Prediction service

Ожидания от мониторинга

- Алертинг о сбоях в системе
- Предупреждения о возможных сбоях и проблемах
- Отражение состояния системы/сервера/сервиса/компонента сервиса
- Сбор статистики и визуализаций
- Отчеты
- Дашборды

4 золотых сигнала

SRE handbook от Google:

1. Latency – время, необходимое на обслуживание запроса
2. Traffic – количество запросов, отправляемых в систему
3. Errors – количество ошибок
4. Saturation – насколько полон ваш сервис

The USE

Resource – все компоненты физического/виртуального сервера (CPU, Disk, RAM, etc.)

Utilization – время, затрачиваемое на выполнение задач

Saturation – показатель, указывающий на количество тасок, которые не могут быть выполнены, попадающие при этом в очередь

Errors – количество ошибок

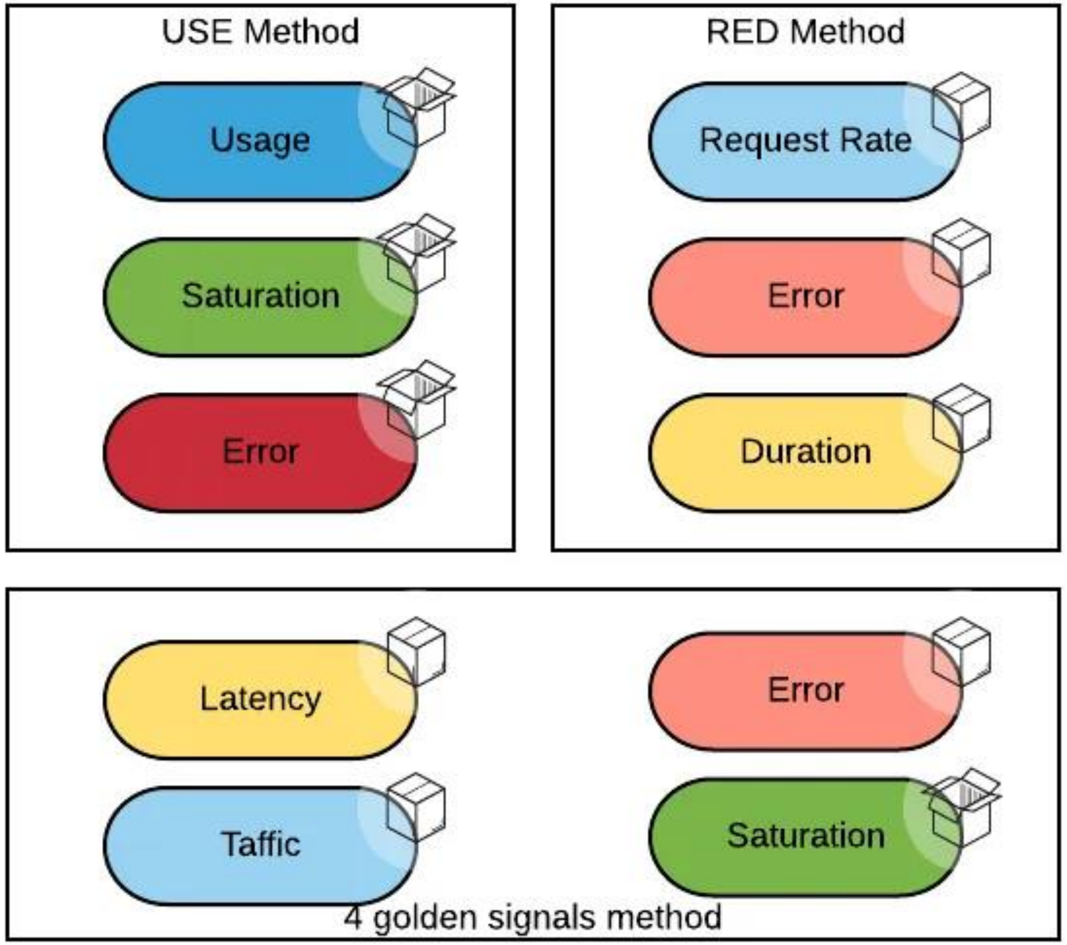
The RED

Rate – количество запросов в секунду

Errors – количество запросов, завершившихся с ошибкой

Duration – количество времени, которое занимает каждый запрос

Все вместе


USE Method
Usage
Saturation
Error
RED Method
Request Rate
Error
Duration
Latency
Error
Taffic
Saturation
4 golden signals method

Метрики

Технические метрики:

Касаются ресурсов (средняя загрузка процессора, занятость диска)

Сервисные метрики:

Касаются запросов (процент успешных запросов, latency)

Бизнес метрики:

MAU, DAU, Clicks, etc

Prometheus

Web-UI

Alertmanager

Prometheus server

Pushgateway

Push & Pull

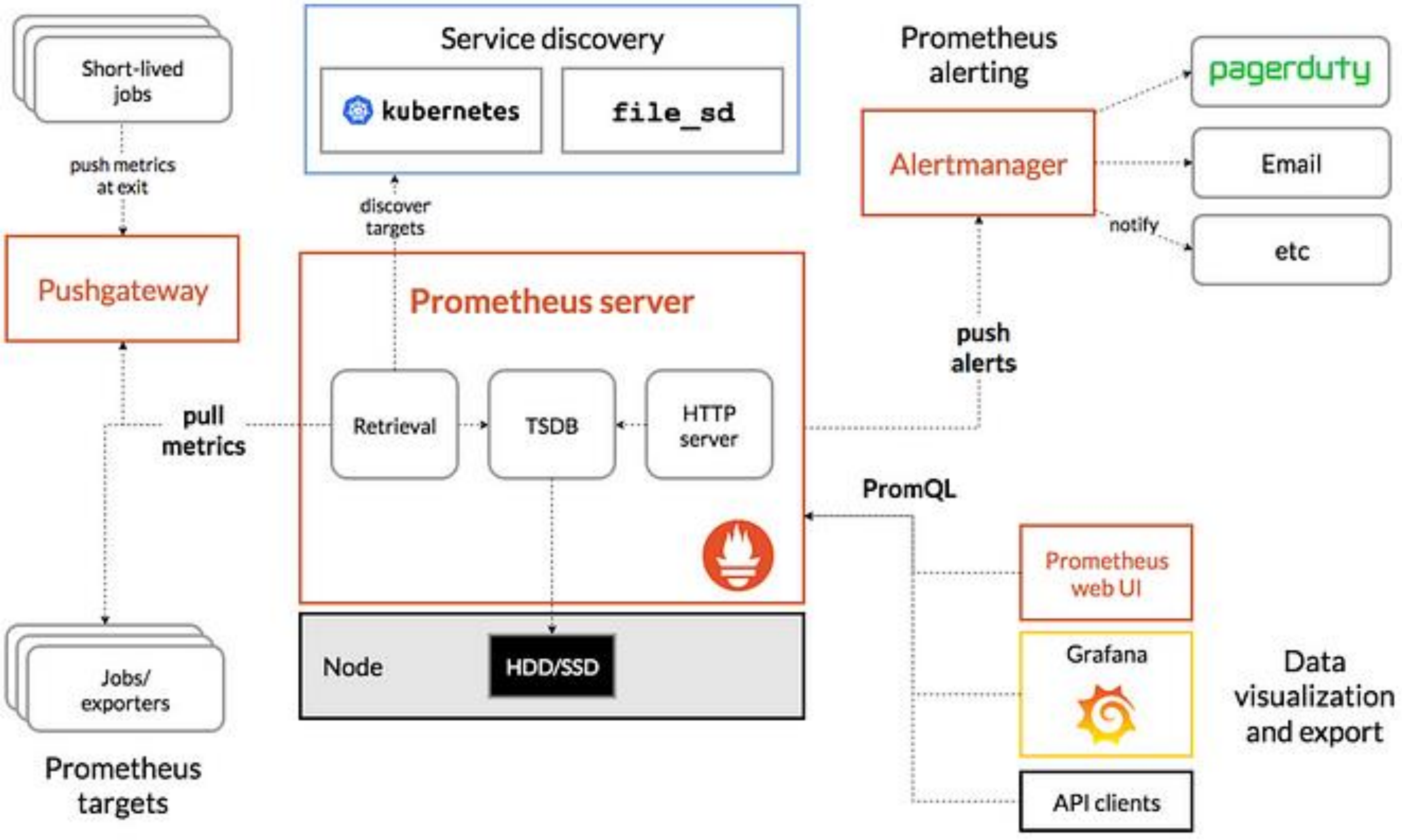
Push

Приложение отправляет куда-то свои метрики

Pull

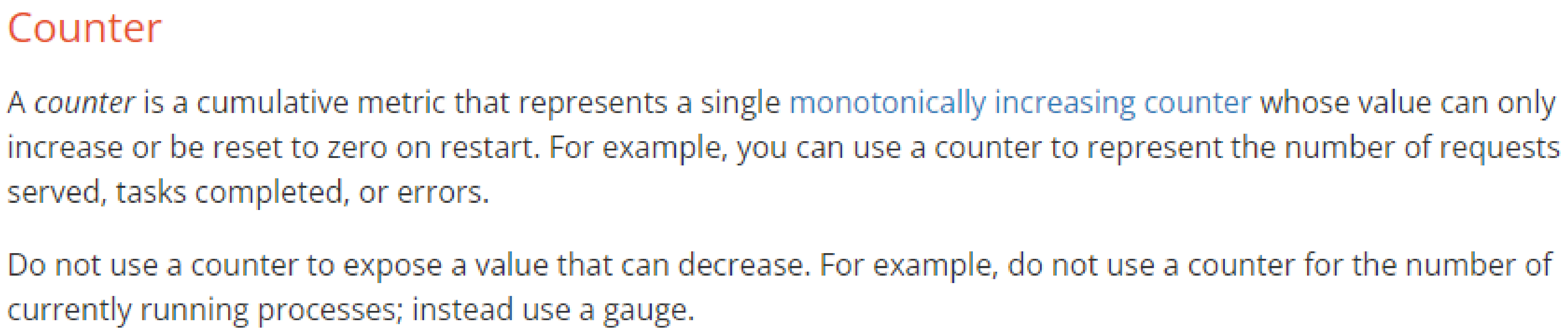
Приложение выставляет endpoint с метриками

Architecture



https://medium.com/@ShadayaniVarshney/a-brief-overview-on-prometheus-monitoring-and-alerting-d5a0080d7e1e

Типы метрик – Counter



Counter

A *counter* is a cumulative metric that represents a single monotonically increasing counter whose value can only increase or be reset to zero on restart. For example, you can use a counter to represent the number of requests served, tasks completed, or errors.

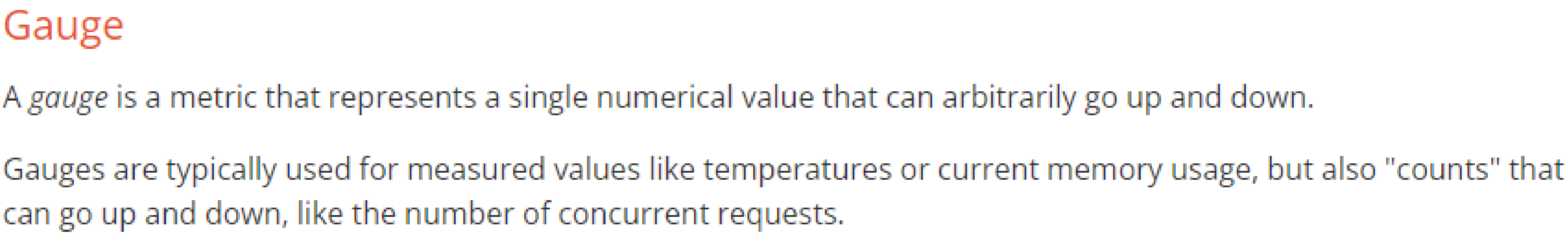
Do not use a counter to expose a value that can decrease. For example, do not use a counter for the number of currently running processes; instead use a gauge.

Примеры:

- Количество успешных запросов
- Количество ошибок

https://prometheus.io/docs/concepts/metric_types/

Типы метрик – Gauge



Gauge

A *gauge* is a metric that represents a single numerical value that can arbitrarily go up and down.

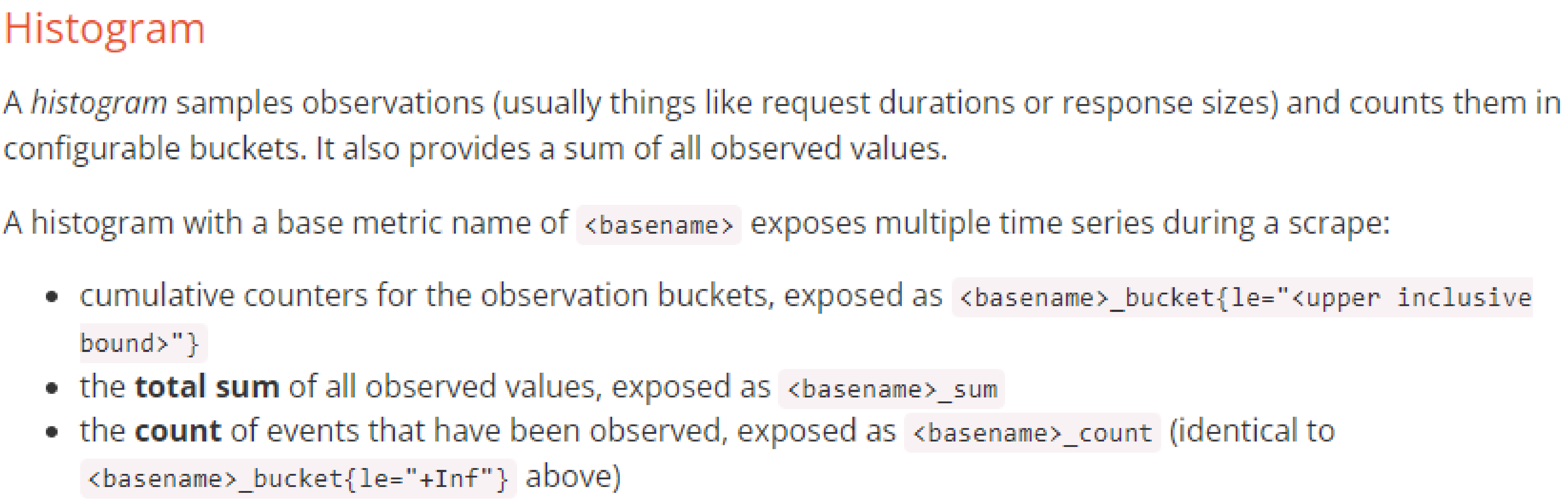
Gauges are typically used for measured values like temperatures or current memory usage, but also "counts" that can go up and down, like the number of concurrent requests.

Примеры:

- Количество машин
- Количество задач в работе

https://prometheus.io/docs/concepts/metric_types/

Типы метрик - Histogram



Histogram

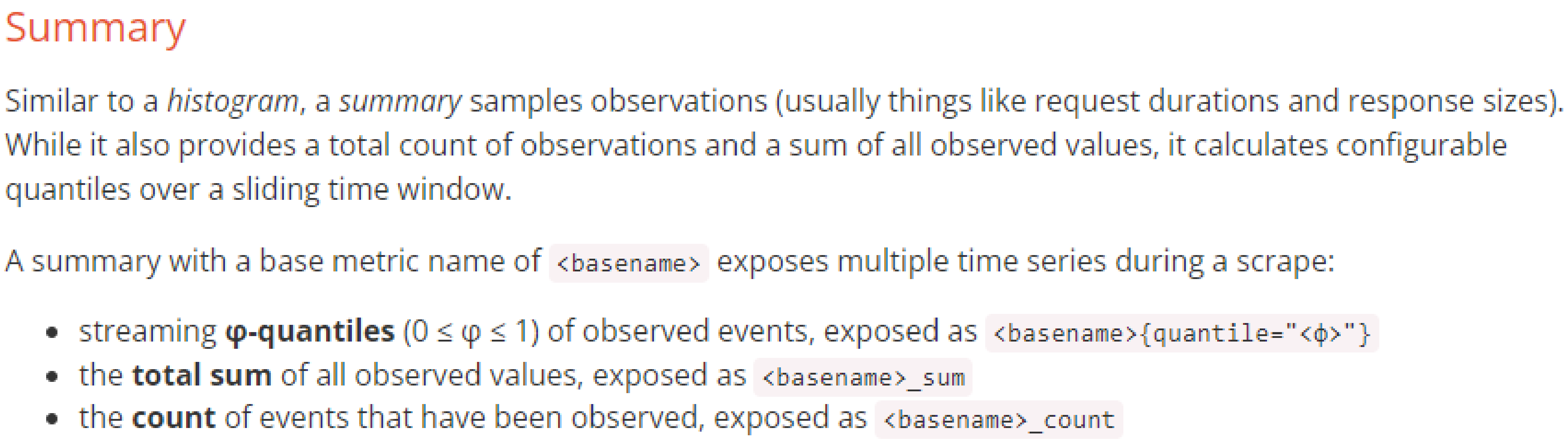
A *histogram* samples observations (usually things like request durations or response sizes) and counts them in configurable buckets. It also provides a sum of all observed values.

A histogram with a base metric name of `<basename>` exposes multiple time series during a scrape:

- cumulative counters for the observation buckets, exposed as `<basename>_bucket{le="<upper inclusive bound>"}`
- the **total sum** of all observed values, exposed as `<basename>_sum`
- the **count** of events that have been observed, exposed as `<basename>_count` (identical to `<basename>_bucket{le="+Inf"}` above)

https://prometheus.io/docs/concepts/metric_types/

Типы метрик – Summary



Summary

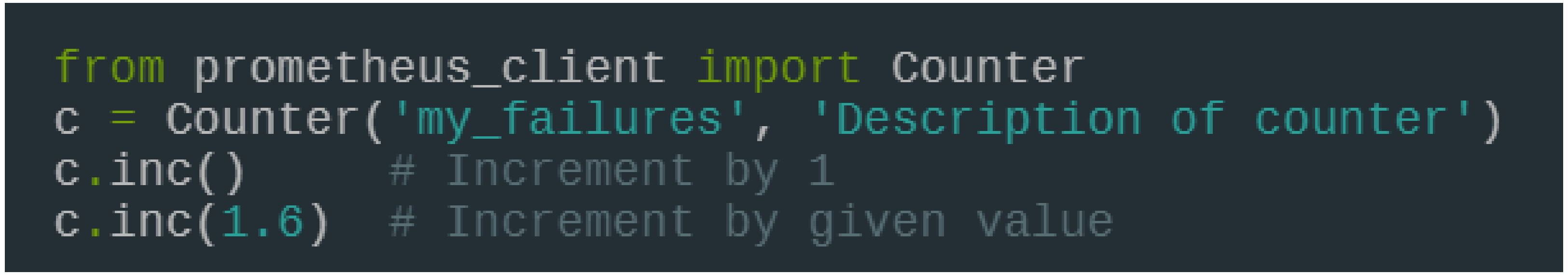
Similar to a *histogram*, a *summary* samples observations (usually things like request durations and response sizes). While it also provides a total count of observations and a sum of all observed values, it calculates configurable quantiles over a sliding time window.

A summary with a base metric name of `<basename>` exposes multiple time series during a scrape:

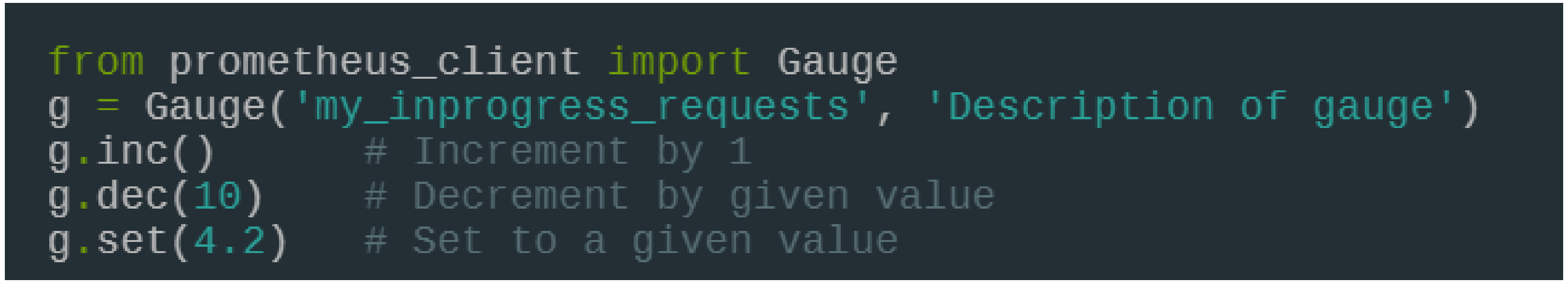
- streaming **φ-quantiles** (0 ≤ φ ≤ 1) of observed events, exposed as `<basename>{quantile="<φ>"}`
- the **total sum** of all observed values, exposed as `<basename>_sum`
- the **count** of events that have been observed, exposed as `<basename>_count`

https://prometheus.io/docs/concepts/metric_types/

Клиенты

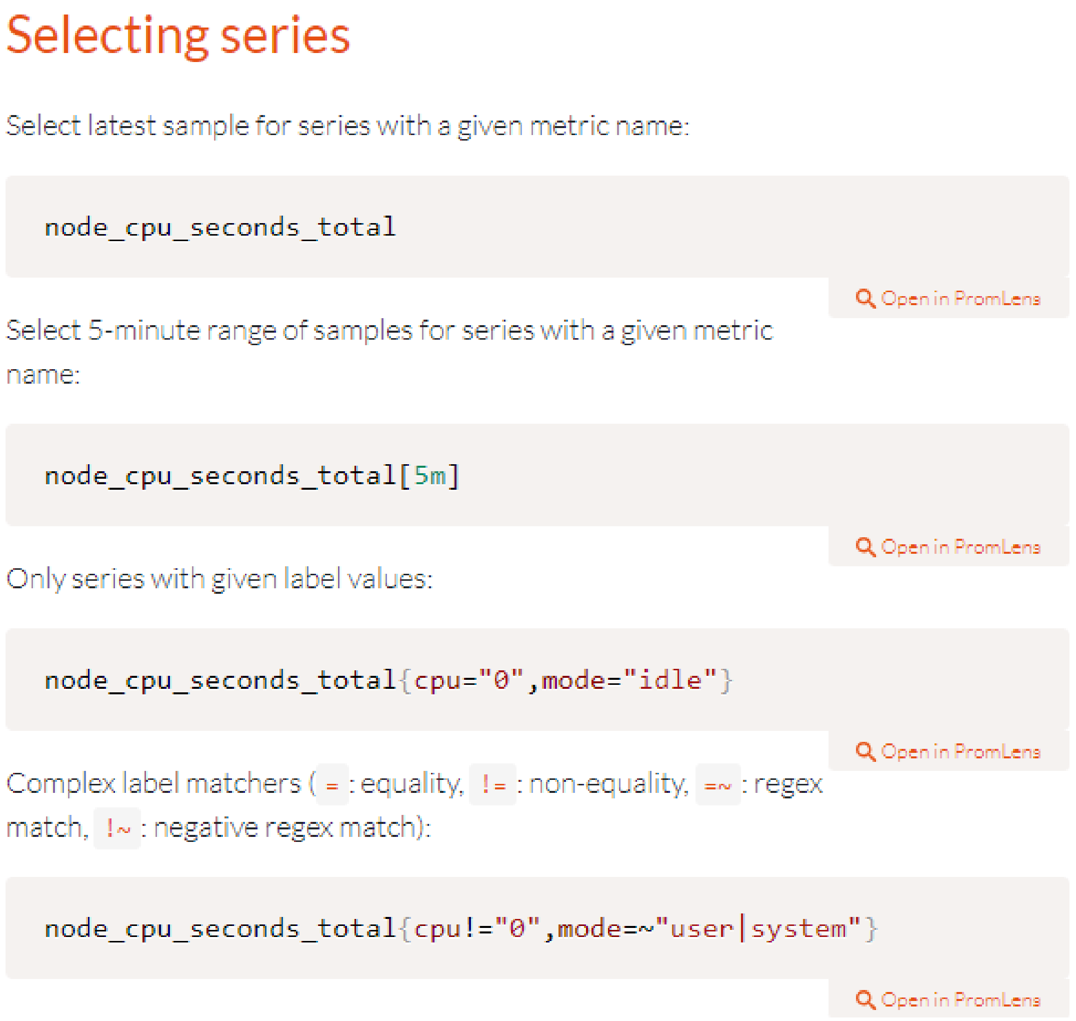


```python
from prometheus_client import Counter
c = Counter('my_failures', 'Description of counter')
c.inc()     # Increment by 1
c.inc(1.6)  # Increment by given value
```



```python
from prometheus_client import Gauge
g = Gauge('my_inprogress_requests', 'Description of gauge')
g.inc()      # Increment by 1
g.dec(10)    # Decrement by given value
g.set(4.2)   # Set to a given value
```

PromQL



Selecting series

Select latest sample for series with a given metric name:

```
node_cpu_seconds_total
```

Select 5-minute range of samples for series with a given metric name:

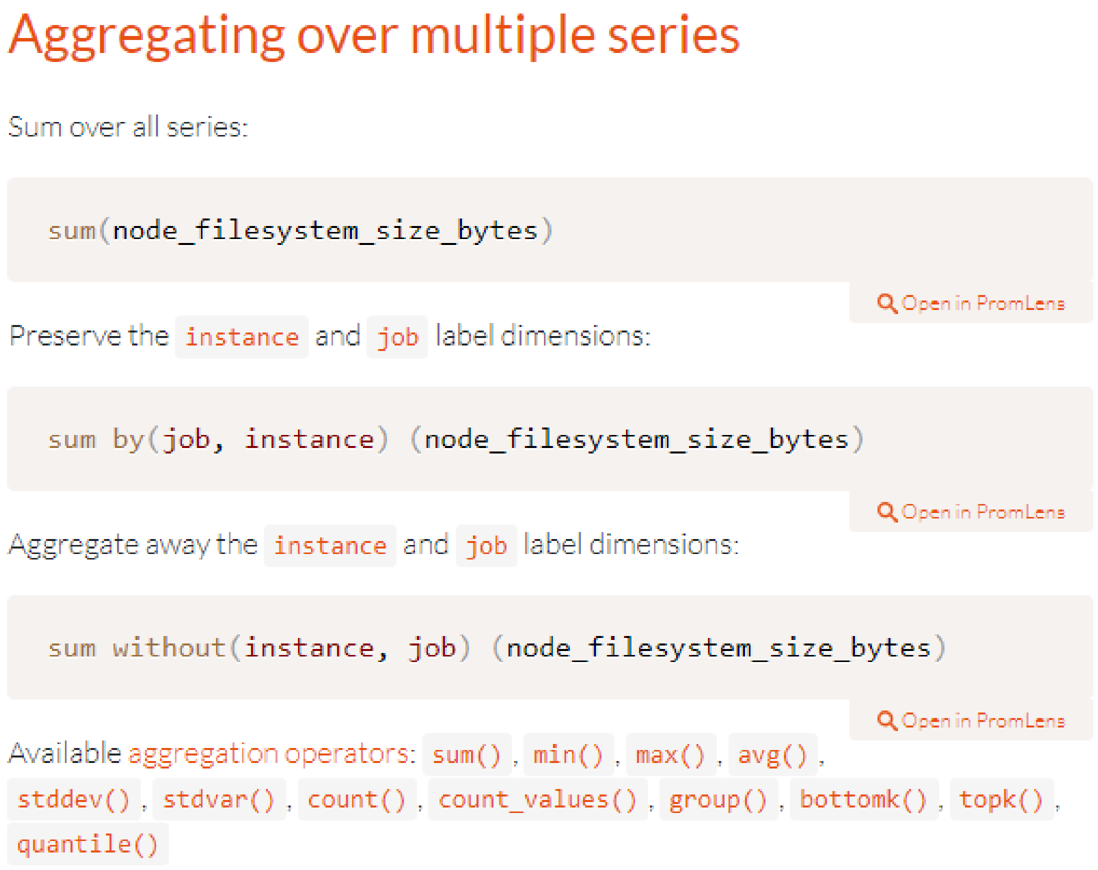
```
node_cpu_seconds_total[5m]
```

Only series with given label values:

```
node_cpu_seconds_total{cpu="0",mode="idle"}
```

Complex label matchers (`=` : equality, `!=` : non-equality, `=~` : regex match, `!~` : negative regex match):

```
node_cpu_seconds_total{cpu!="0",mode=~"user|system"}
```



Aggregating over multiple series

Sum over all series:

```
sum(node_filesystem_size_bytes)
```

Preserve the `instance` and `job` label dimensions:

```
sum by(job, instance) (node_filesystem_size_bytes)
```

Aggregate away the `instance` and `job` label dimensions:

```
sum without(instance, job) (node_filesystem_size_bytes)
```

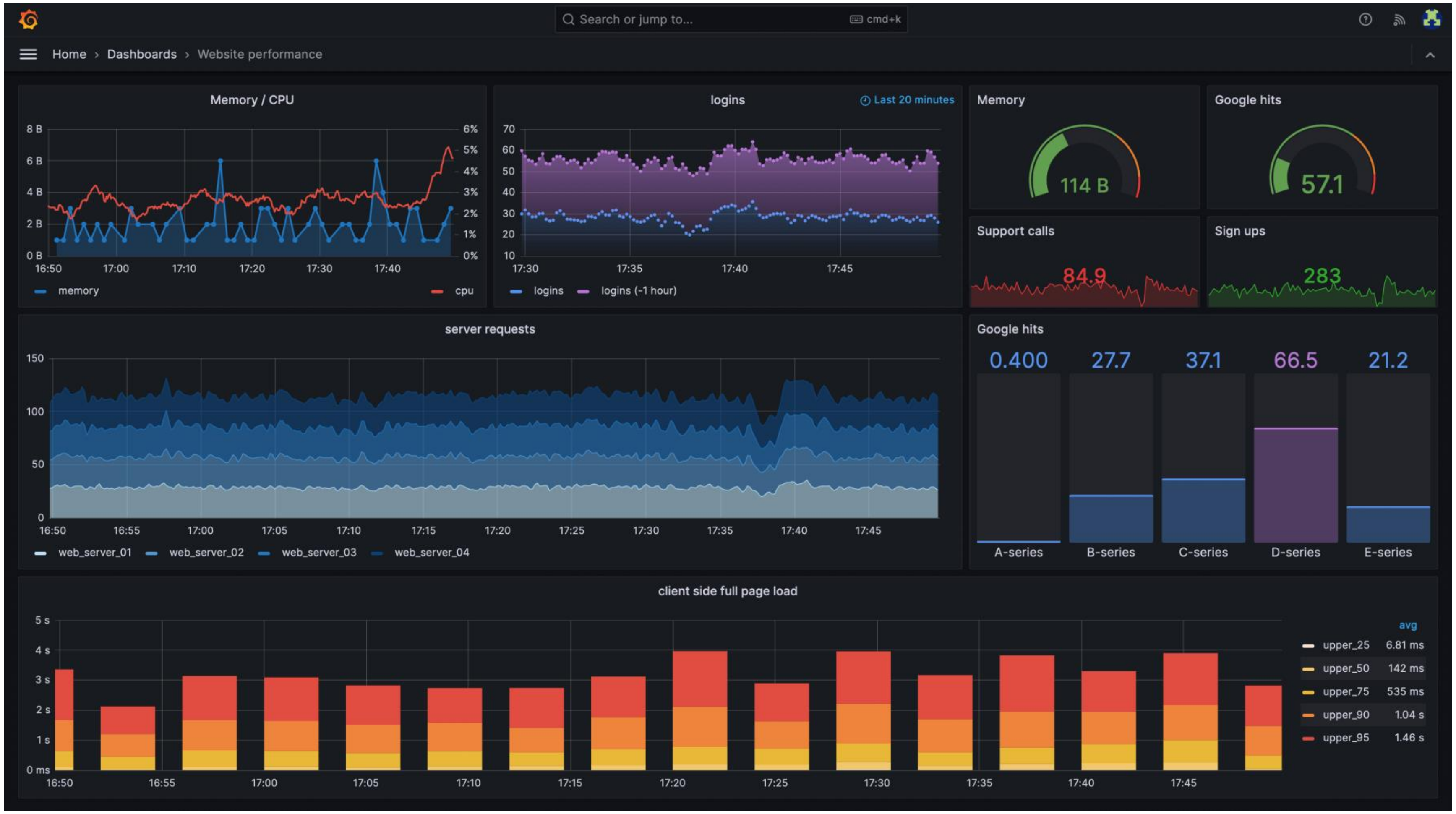
Available aggregation operators: `sum()`, `min()`, `max()`, `avg()`, `stddev()`, `stdvar()`, `count()`, `count_values()`, `group()`, `bottomk()`, `topk()`, `quantile()`

https://promlabs.com/promql-cheat-sheet/

PromQL sandbox

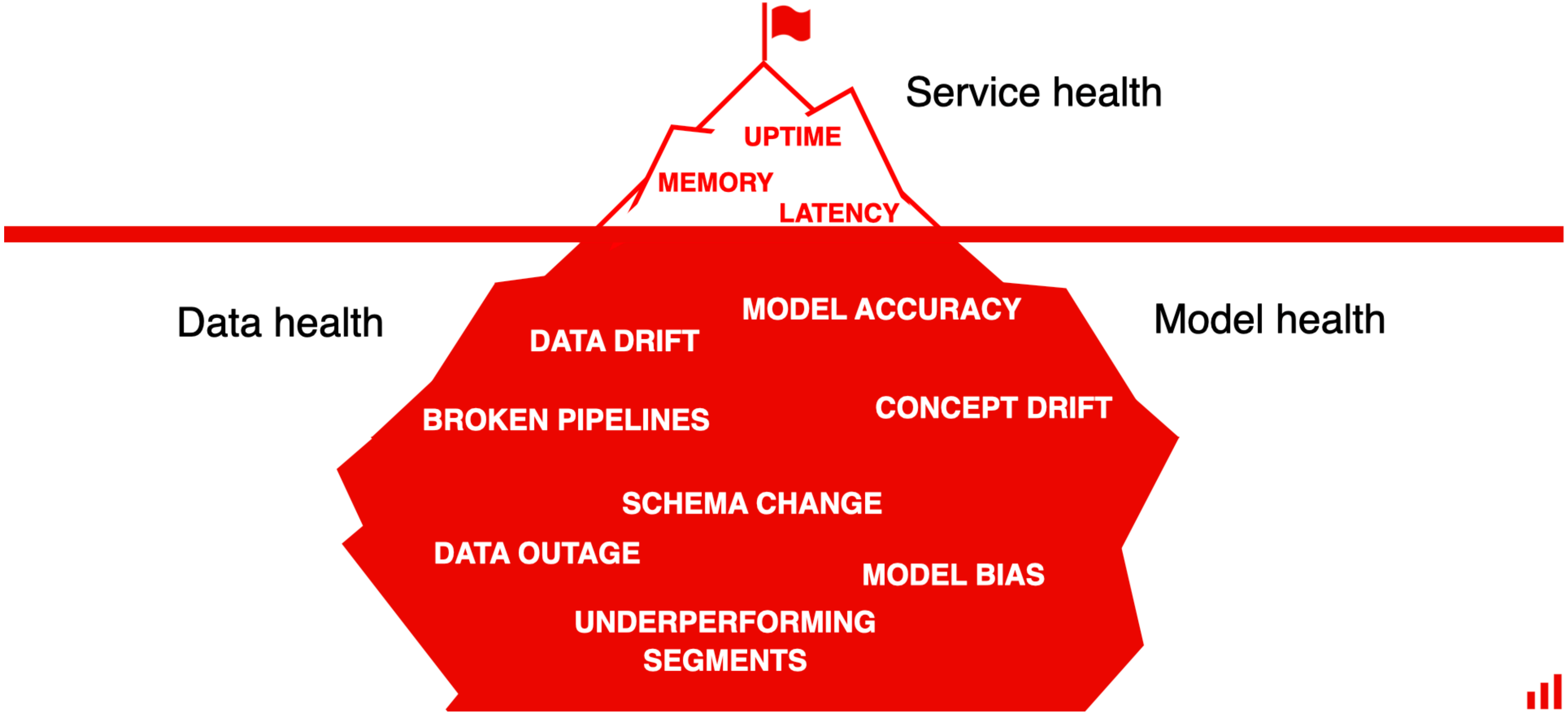
https://demo.promlens.com

Grafana



https://play.grafana.org

Что мониторить еще?



Что мониторить еще?

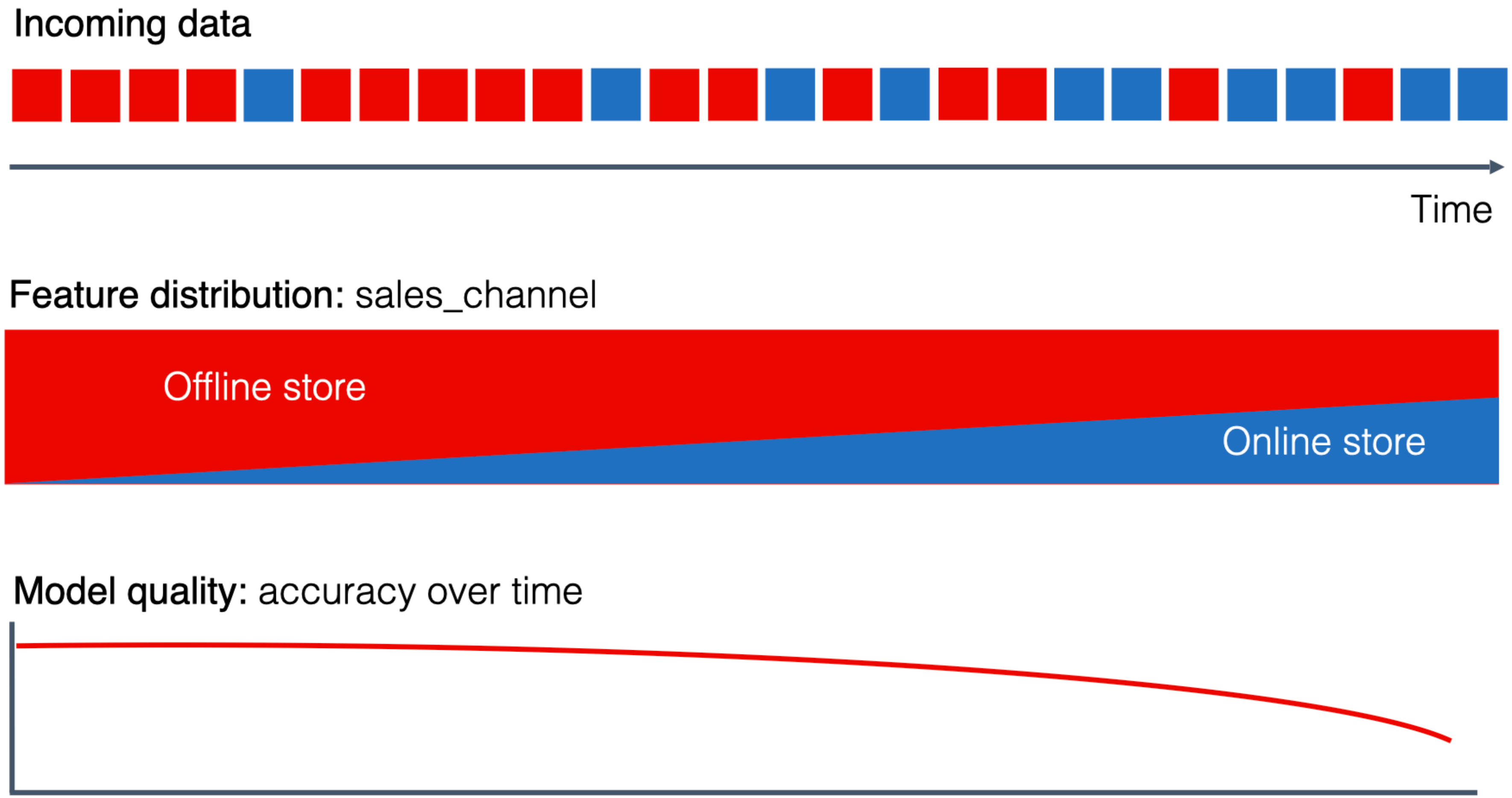
Прекрасные референсы:

https://www.evidentlyai.com/ml-in-production/data-drift

https://www.evidentlyai.com/ml-in-production/model-monitoring

https://www.evidentlyai.com/ml-in-production/concept-drift

Data Drift



Incoming data

Time

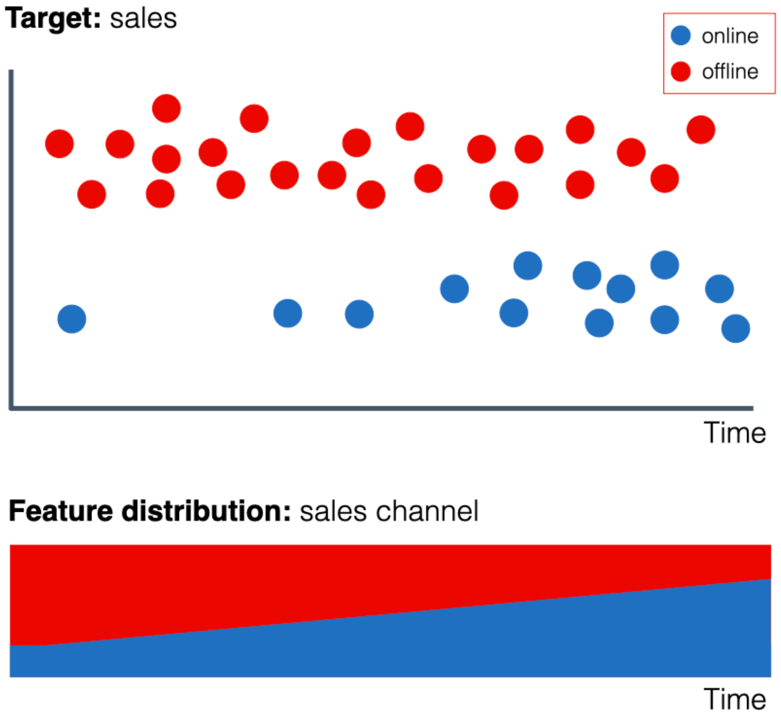
Feature distribution: sales_channel

Model quality: accuracy over time

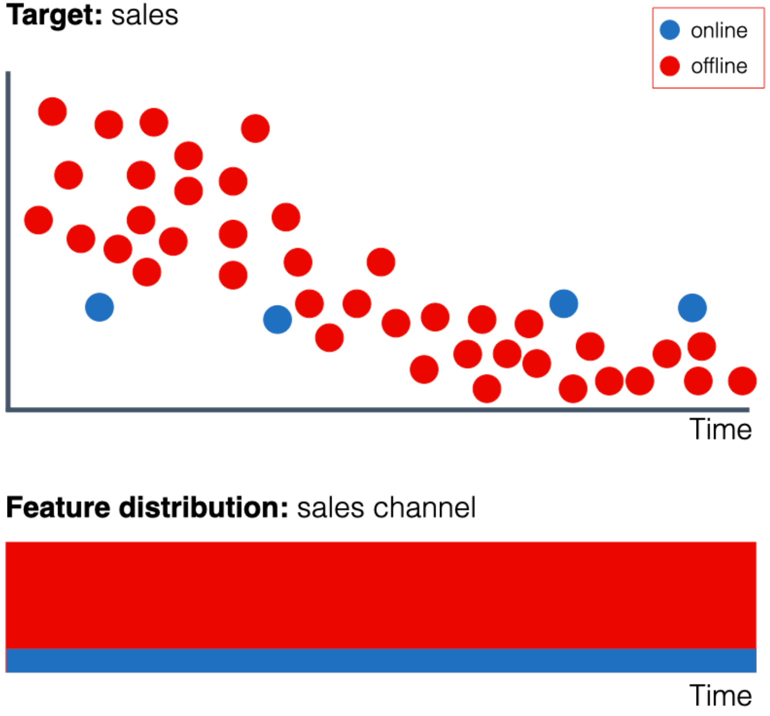
Concept Drift

Data drift	Concept drift
P(X)	P(Y|X)

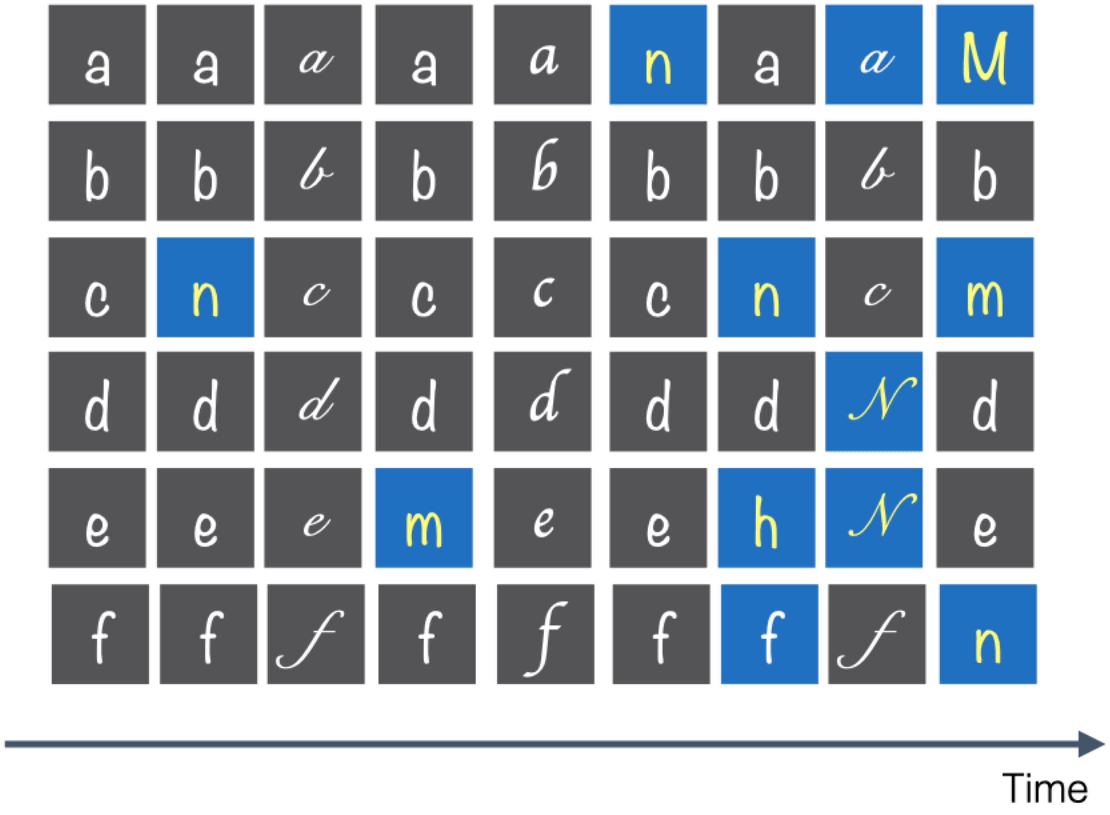
Data drift



Concept drift

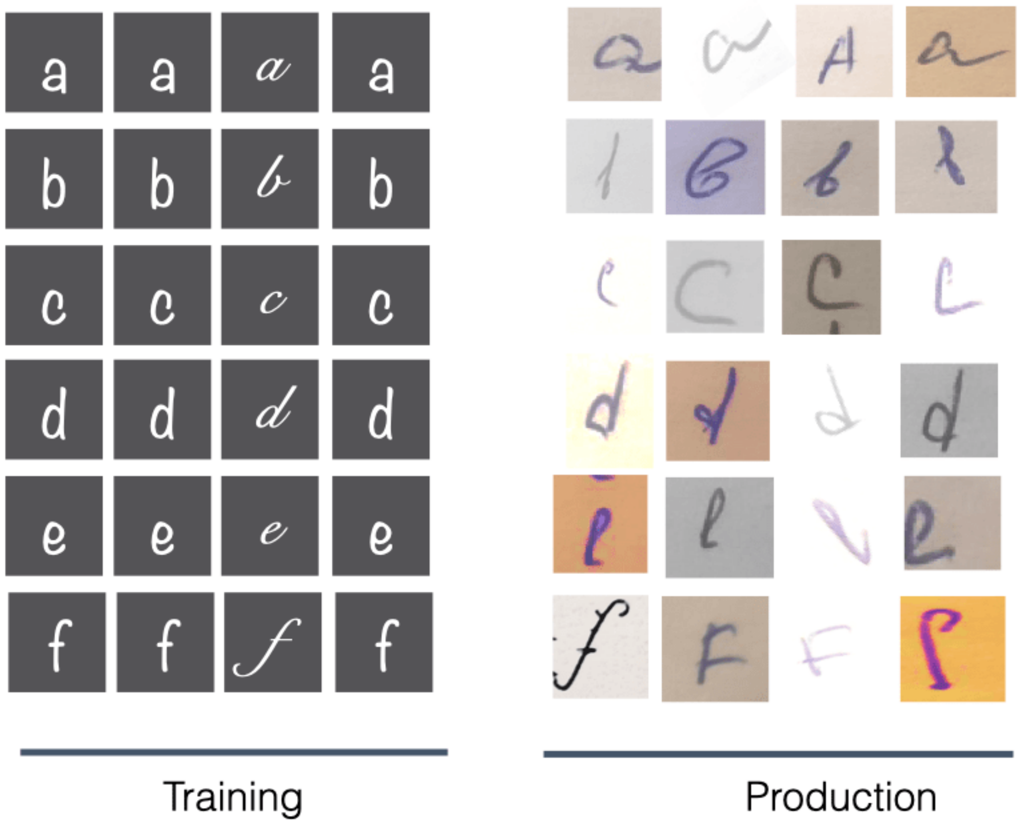


Data drift



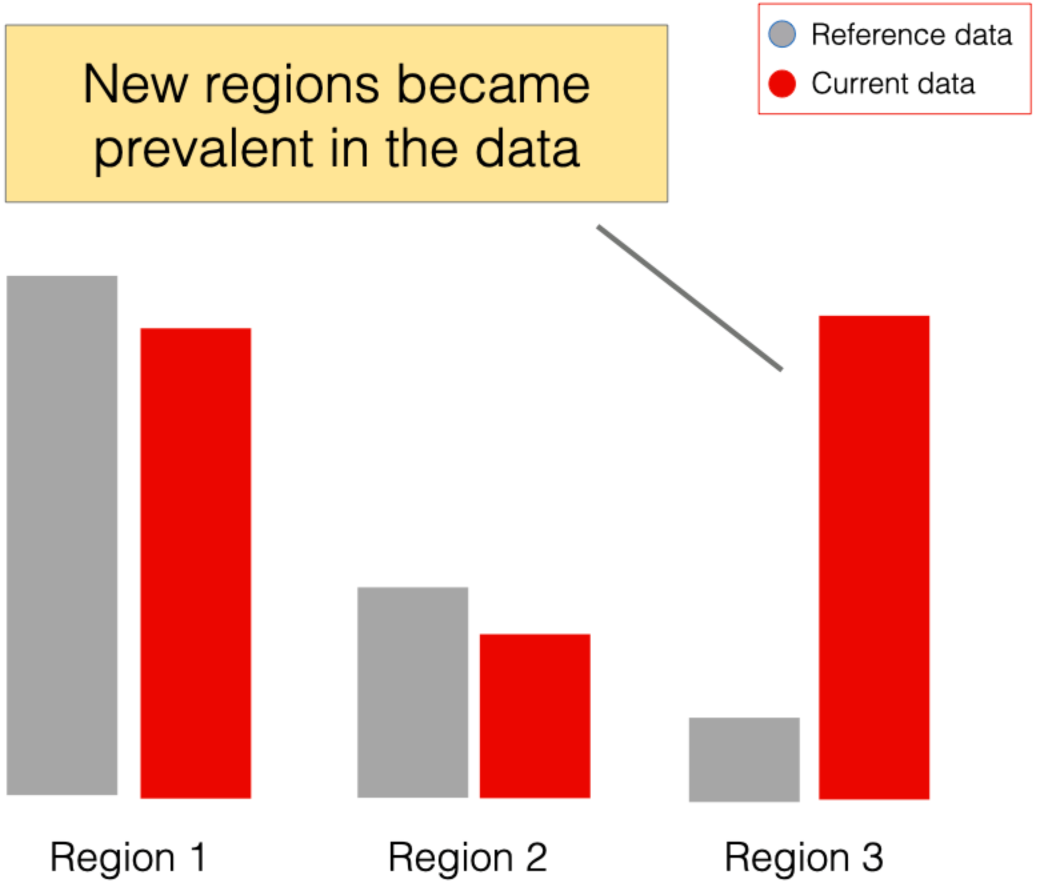
a	a	a	a	a	n	a	a	M
b	b	b	b	b	b	b	b	b
c	n	c	c	c	c	n	c	m
d	d	d	d	d	d	d	N	d
e	e	e	m	e	e	h	N	e
f	f	f	f	f	f	f	f	n

Train-serving skew



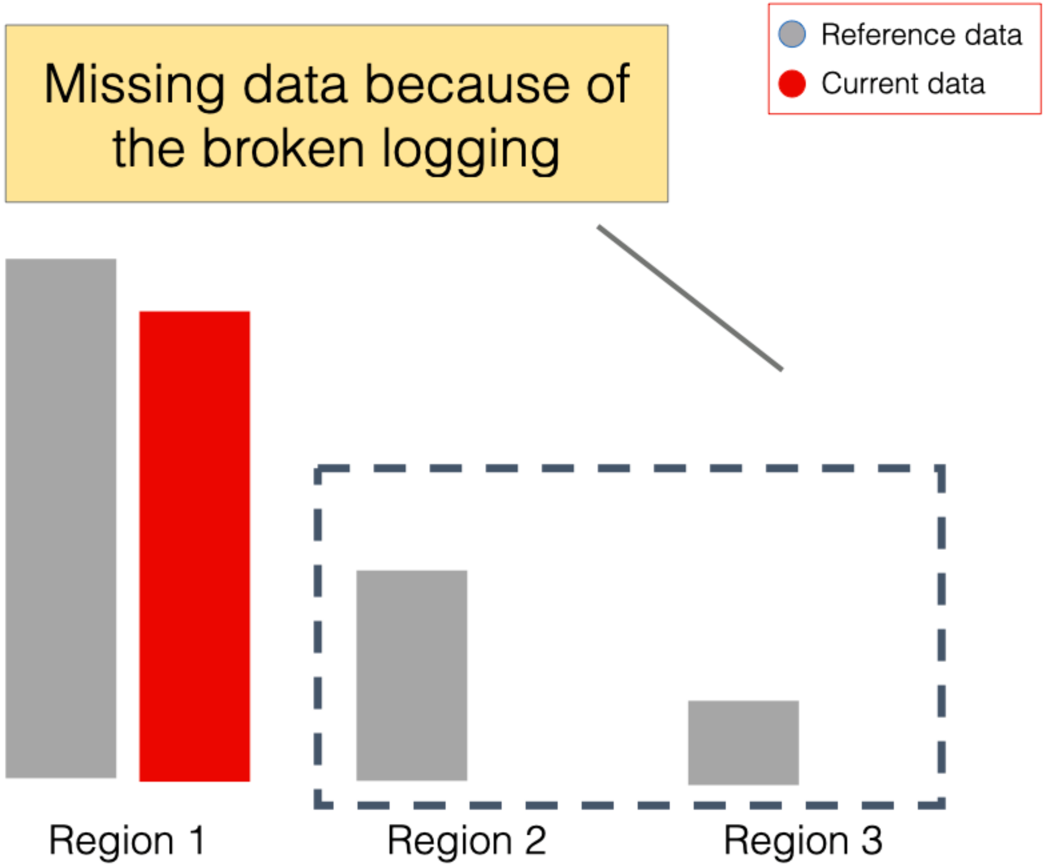
a	a	a	a
b	b	b	b
c	c	c	c
d	d	d	d
e	e	e	e
f	f	f	f

Data drift



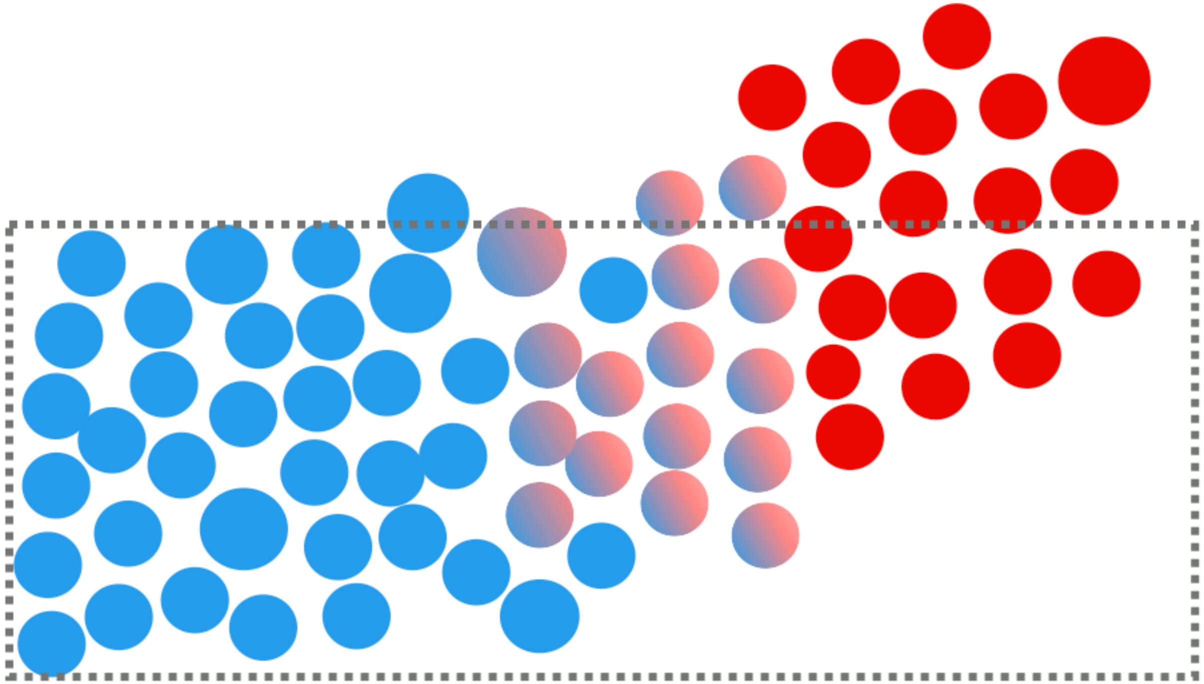
New regions became prevalent in the data

Data quality

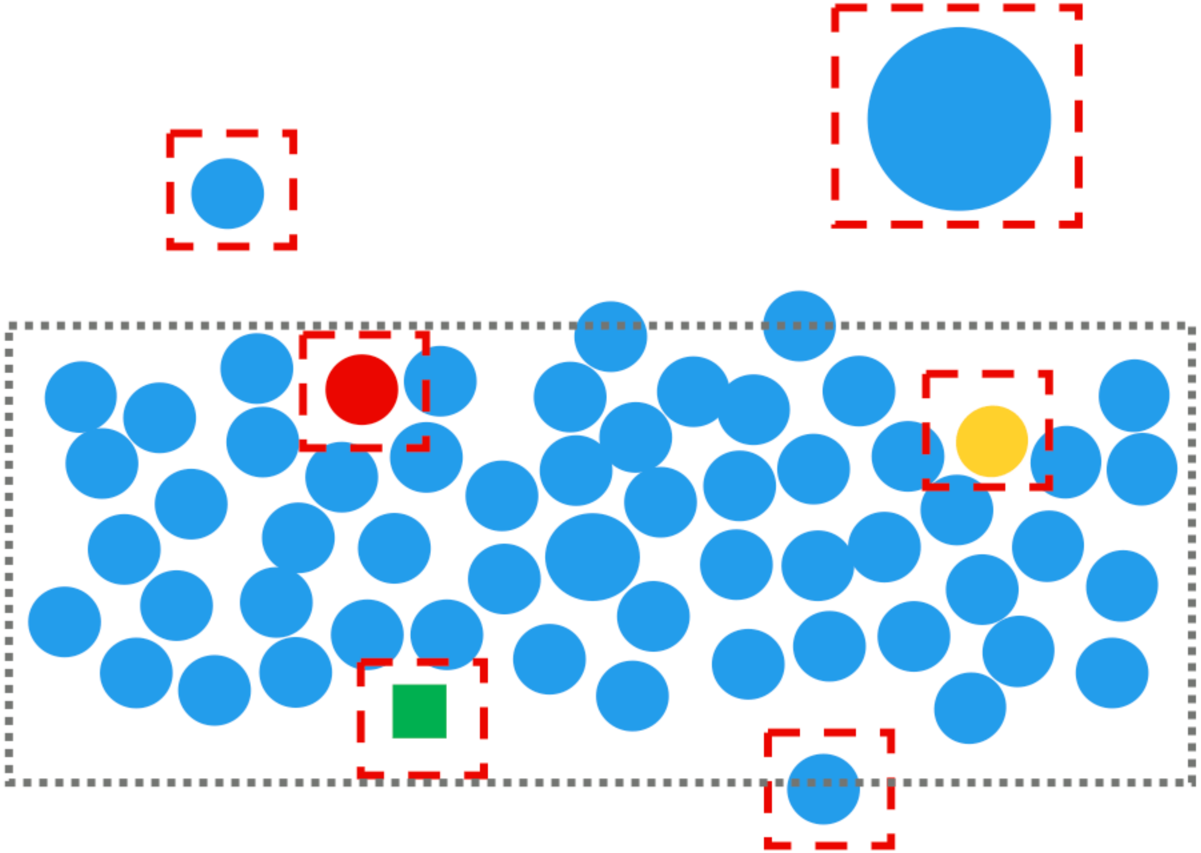


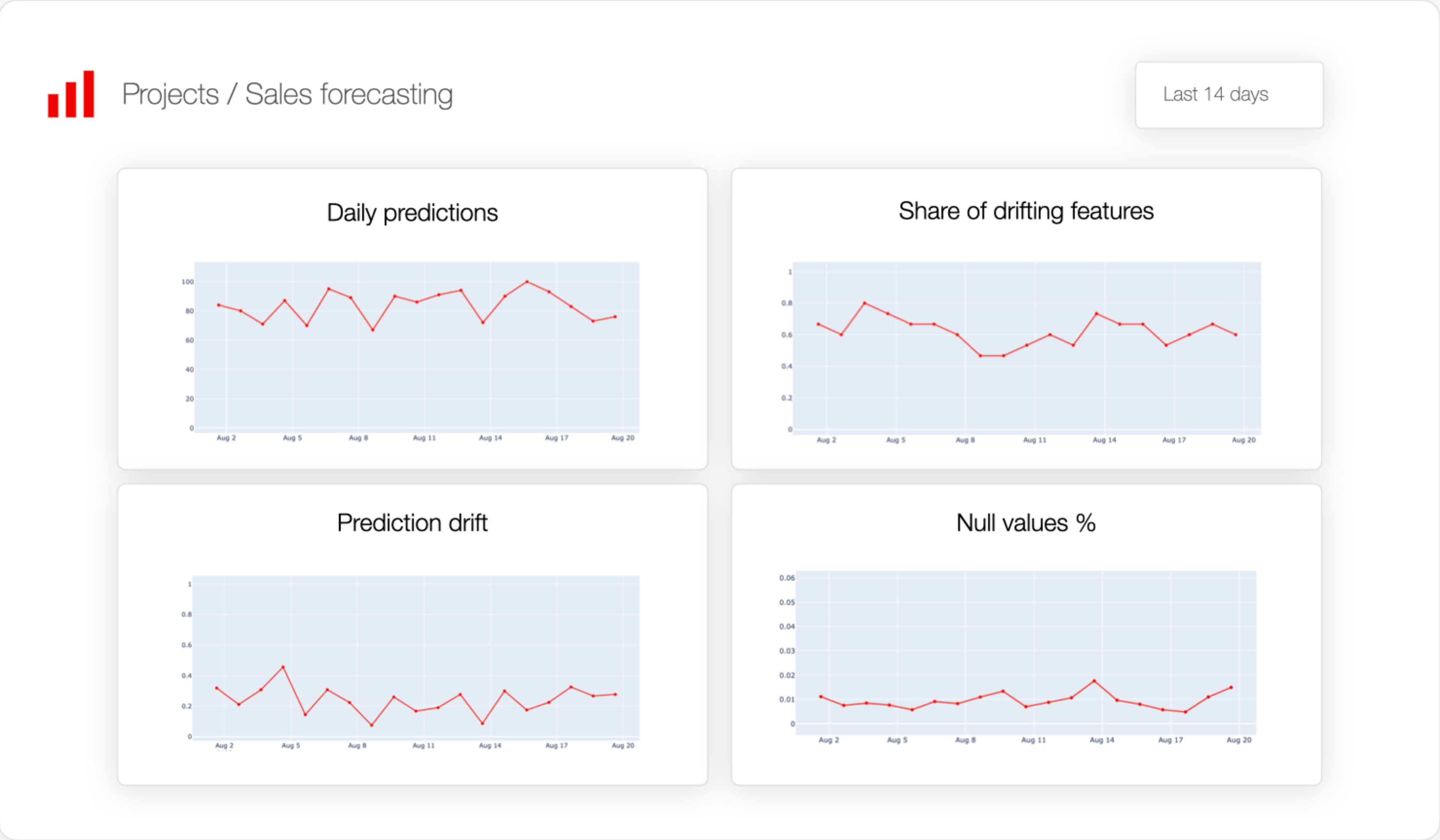
Missing data because of the broken logging

Data drift



Outliers





Projects / Sales forecasting

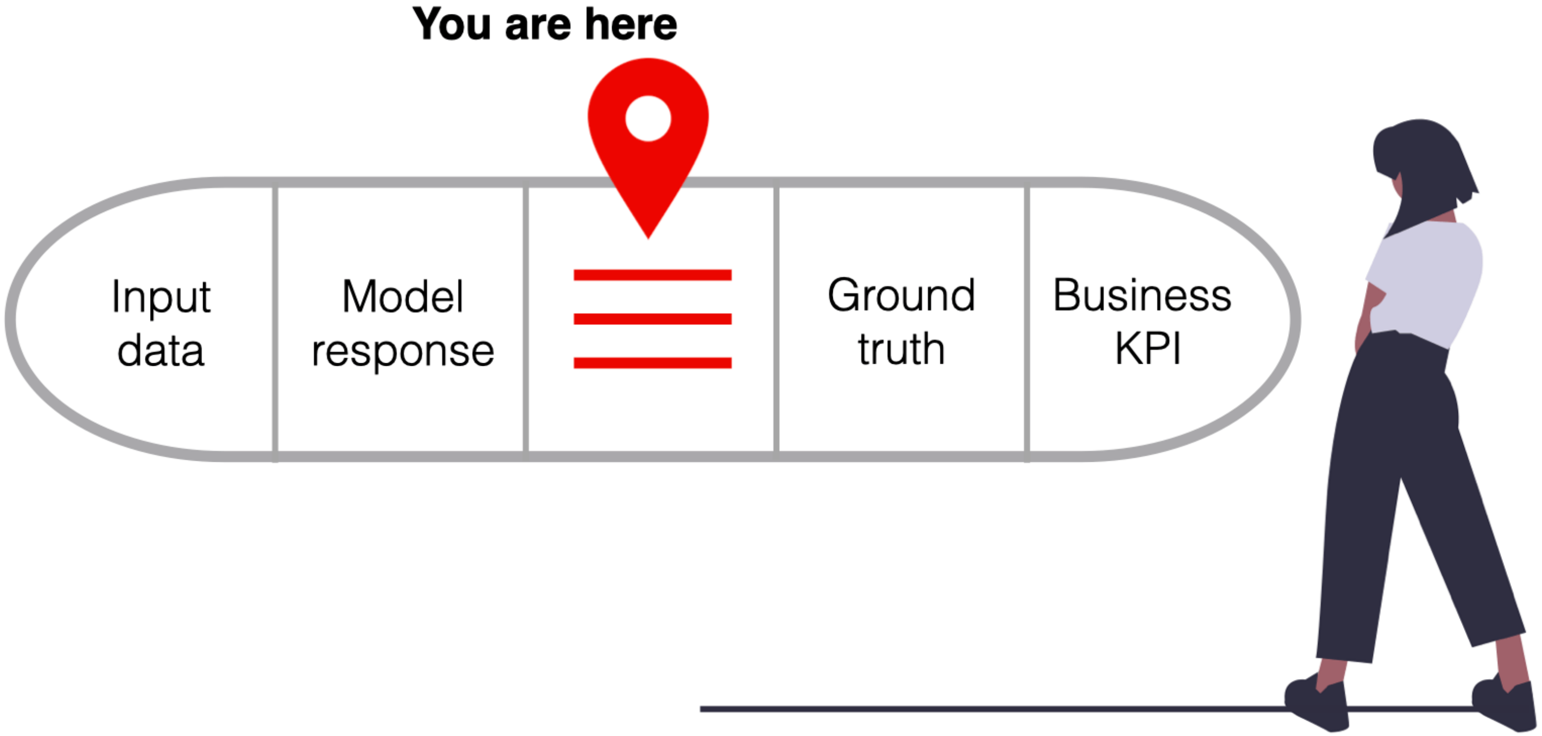
Daily predictions

Share of drifting features

Prediction drift

Null values %

GT delay



You are here

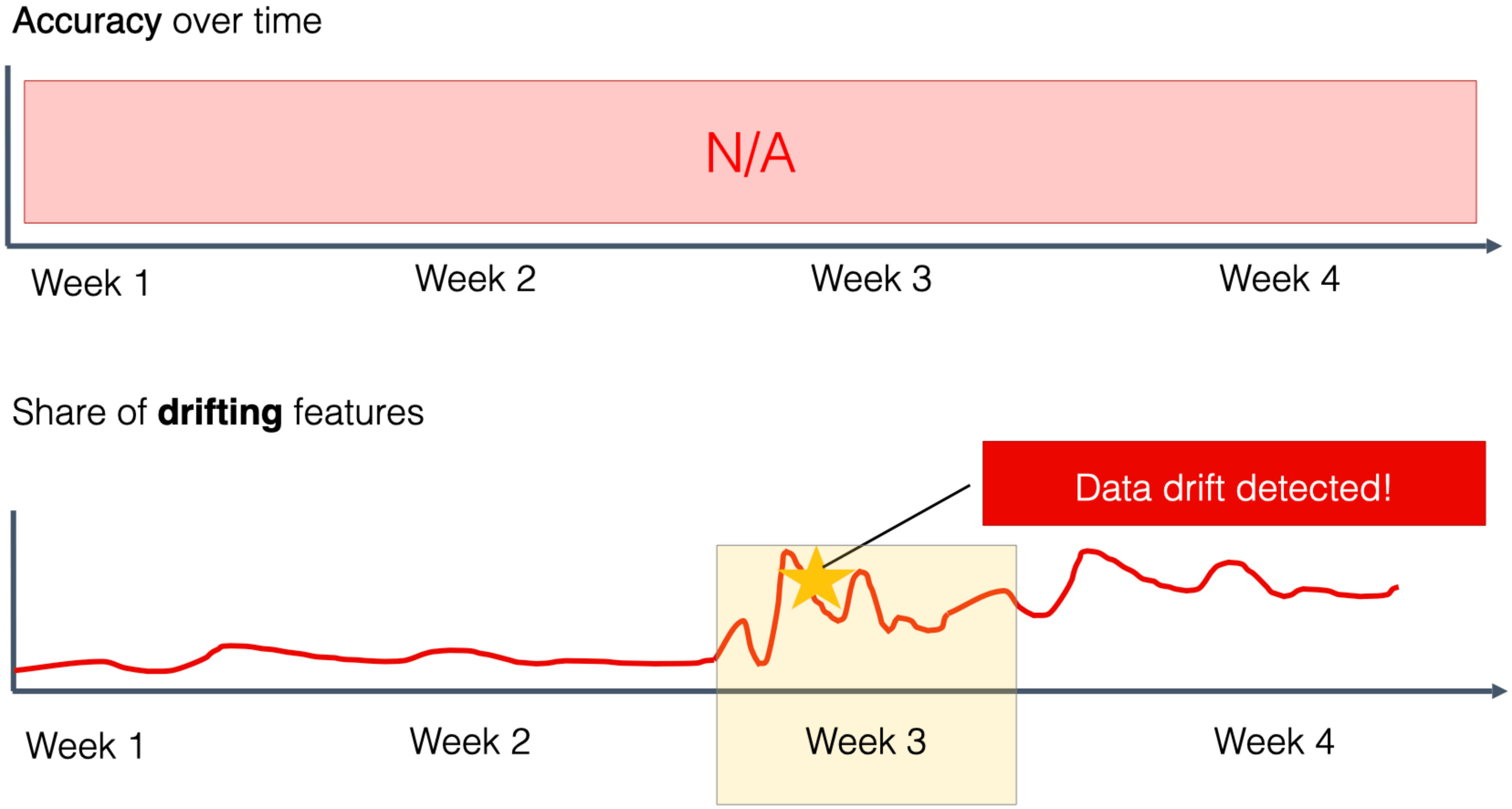
Input data

Model response

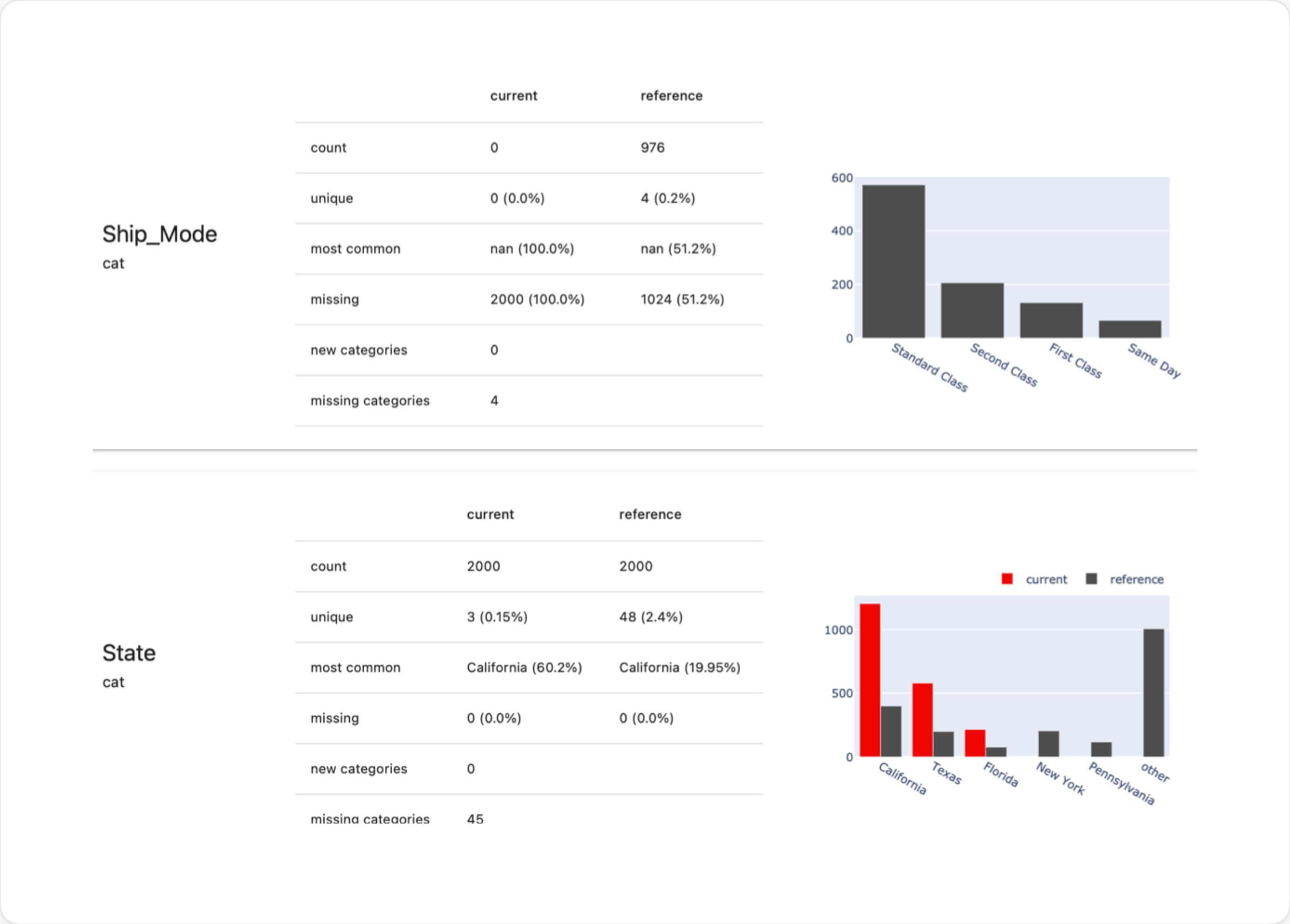
Ground truth

Business KPI

GT delay


Accuracy over time
N/A
Week 1
Week 2
Week 3
Week 4
Share of **drifting** features
Data drift detected!
Week 1
Week 2
Week 3
Week 4

Data Drift Detect. Summary Statistics



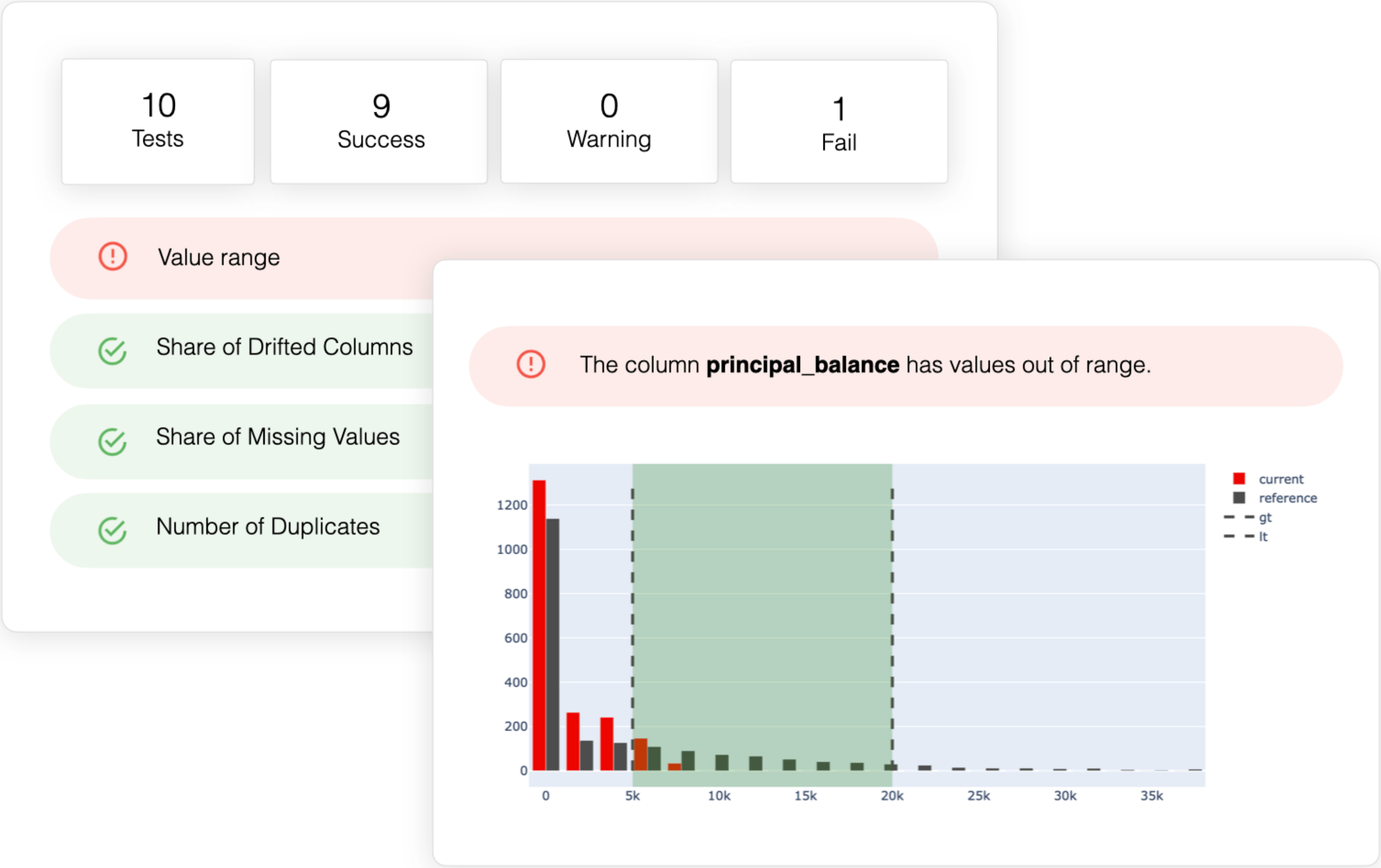
Ship_Mode
cat

	current	reference
count	0	976
unique	0 (0.0%)	4 (0.2%)
most common	nan (100.0%)	nan (51.2%)
missing	2000 (100.0%)	1024 (51.2%)
new categories	0	
missing categories	4	

State
cat

	current	reference
count	2000	2000
unique	3 (0.15%)	48 (2.4%)
most common	California (60.2%)	California (19.95%)
missing	0 (0.0%)	0 (0.0%)
new categories	0	
missing categories	45	

Data Drift Detect. Summary Statistics



10 Tests

9 Success

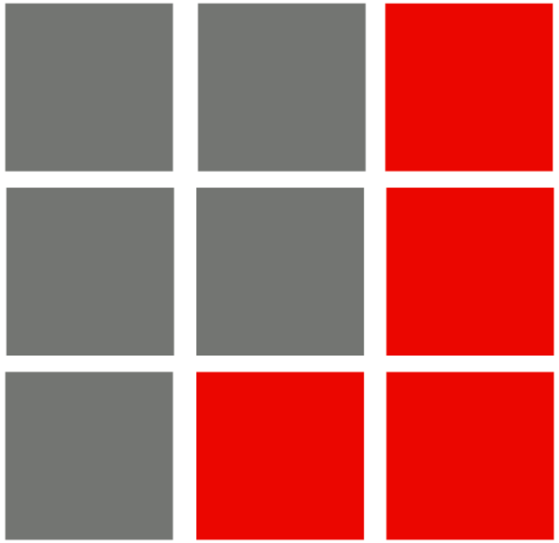
0 Warning

1 Fail

- Value range
- Share of Drifted Columns
- Share of Missing Values
- Number of Duplicates

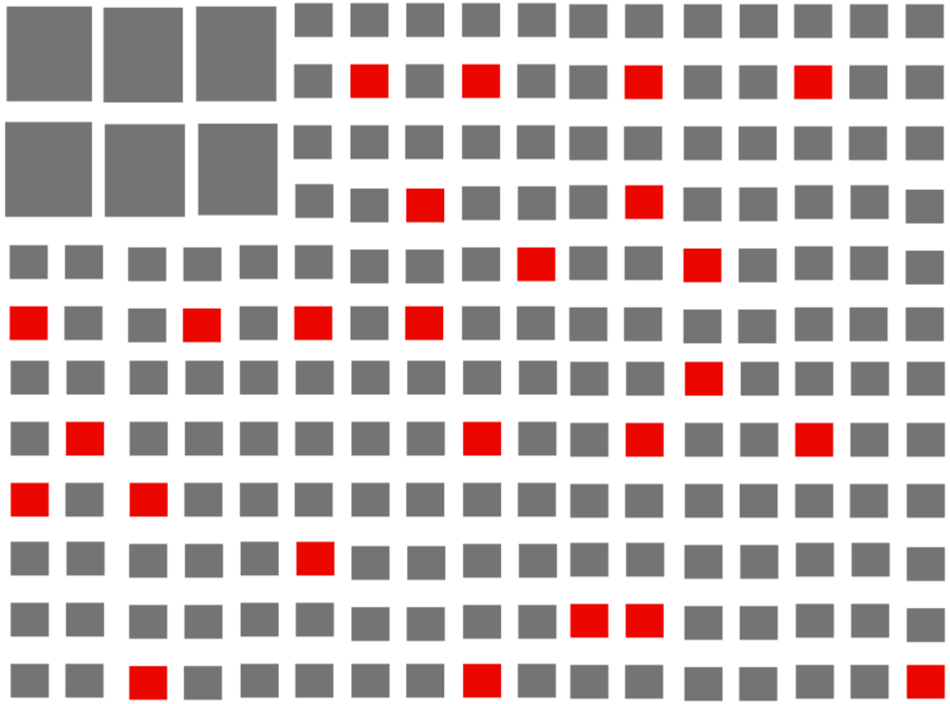
The column **principal_balance** has values out of range.

Data Drift Detect. Статистические тесты



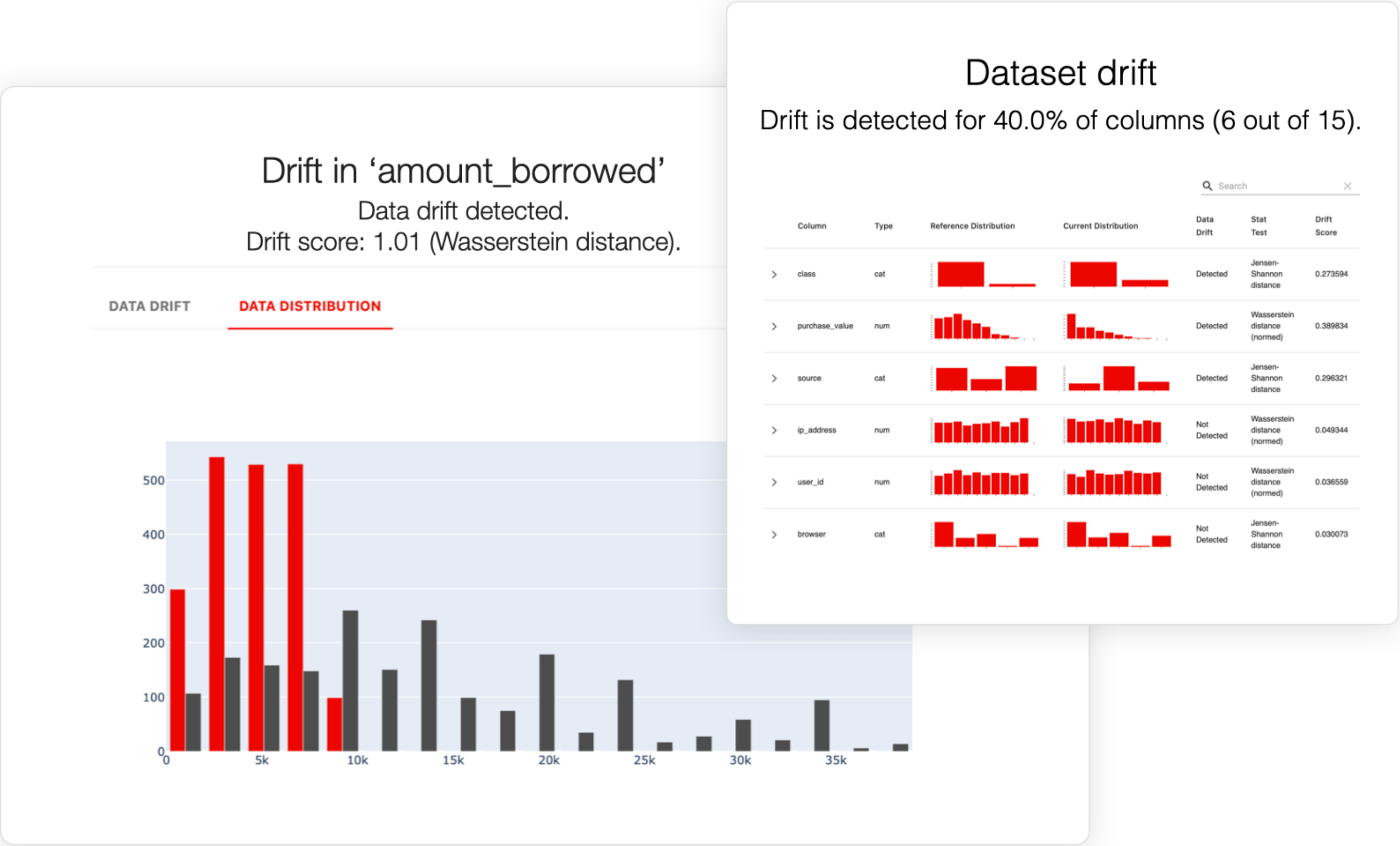
Few important features

VS



A lot of unimportant features

Data Drift Detect. Distance metrics



Drift in ‘amount_borrowed’

Data drift detected.

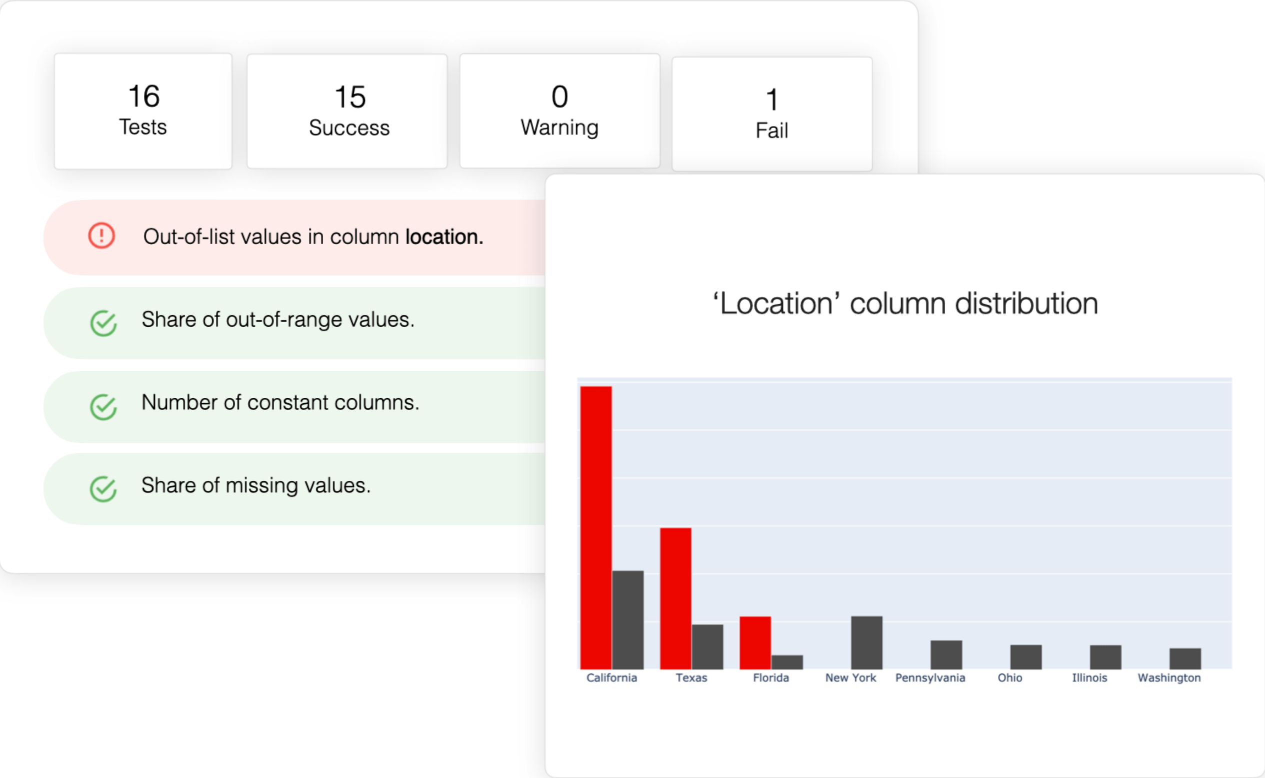
Drift score: 1.01 (Wasserstein distance).

Dataset drift

Drift is detected for 40.0% of columns (6 out of 15).

	Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
›	class	cat			Detected	Jensen-Shannon distance	0.273594
›	purchase_value	num			Detected	Wasserstein distance (normed)	0.389834
›	source	cat			Detected	Jensen-Shannon distance	0.296321
›	ip_address	num			Not Detected	Wasserstein distance (normed)	0.049344
›	user_id	num			Not Detected	Wasserstein distance (normed)	0.036559
›	browser	cat			Not Detected	Jensen-Shannon distance	0.030073

Data Drift Detect. Rule-based checks

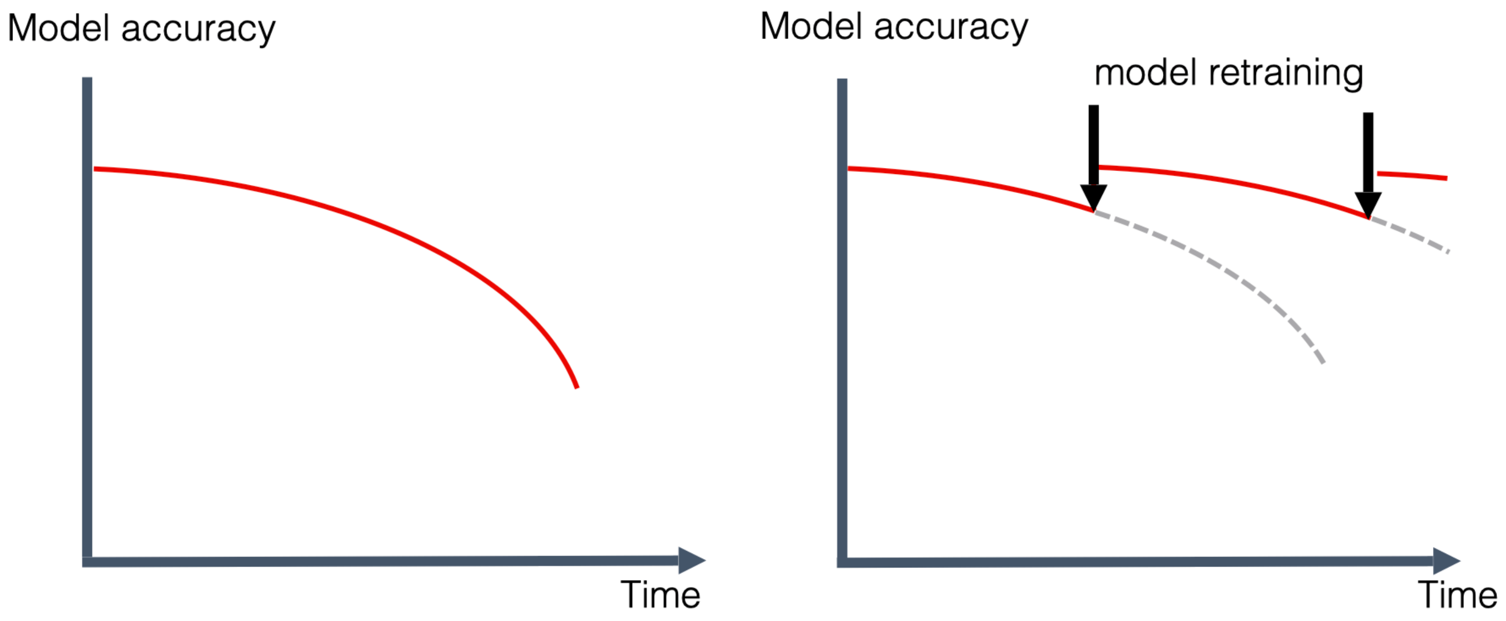


16	15	0	1
Tests	Success	Warning	Fail

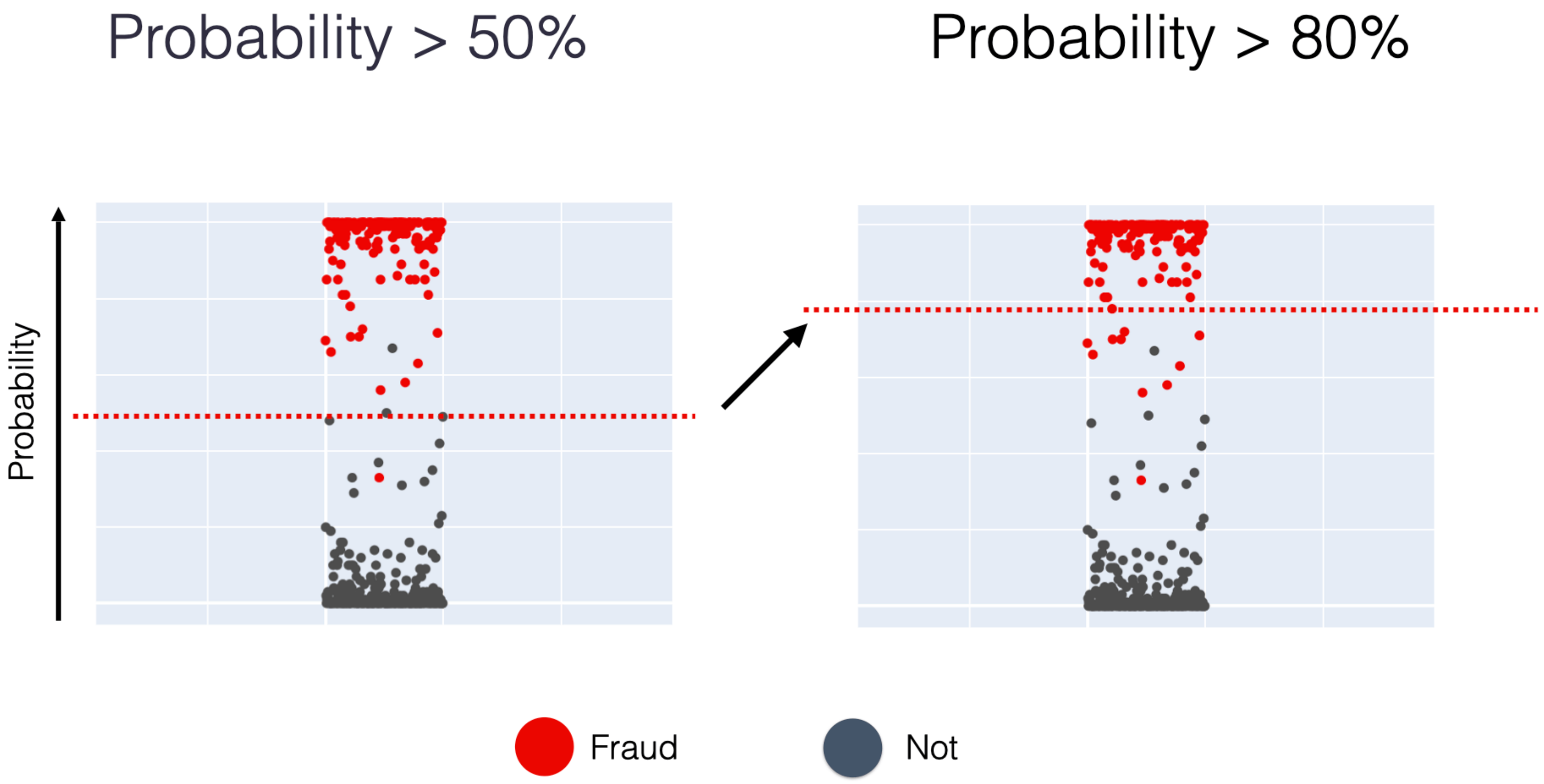
- ! Out-of-list values in column **location.**
- ✓ Share of out-of-range values.
- ✓ Number of constant columns.
- ✓ Share of missing values.

‘Location’ column distribution

Retraining



Process intervention



Probability > 50%

Probability > 80%

Evidently Demo

https://demo.evidentlyai.com