Model Monitoring

Model Monitoring

ML model monitoring involves observing the performance of ML models in a production environment to pinpoint any potential problems that could negatively impact business value. This process helps in proactively managing issues related to prediction quality, data relevance, model accuracy, and bias.

Importance of Monitoring

- Model performance can deteriorate due to data drift, concept drift, or changes in the underlying distribution
- Models may become stale or biased as the real-world data evolves
- Monitoring is crucial for maintaining model performance and reliability
- Enables early detection of issues and timely interventions
- Prevents models from making inaccurate predictions or decisions

ML Monitoring Topics

- Model Inputs (Data Distribution, Data Drift)
- Model Outputs (Performance Metrics, Prediction Drift)
- System Resources (CPU, Memory, Disk Usage)
- Model Artifacts (Model Versions, Configurations)

What needs to be monitored in production?

Two levels of Monitoring:

- Functional: The Input, the Model and the output predictions
- Operational: System Performance, Pipeline and Costs

Functional Level Monitoring - Input Data

- Models rely on the input data they receive, and unexpected inputs can lead to model breakage.
- Monitoring input data is essential for detecting functional performance issues before they affect the machine learning system.
- Key items to monitor from an input data perspective include:
  - **Data quality**:

    Validate production data to ensure data integrity and equivalence of data types, addressing issues like schema changes or data loss.

    Data drift:

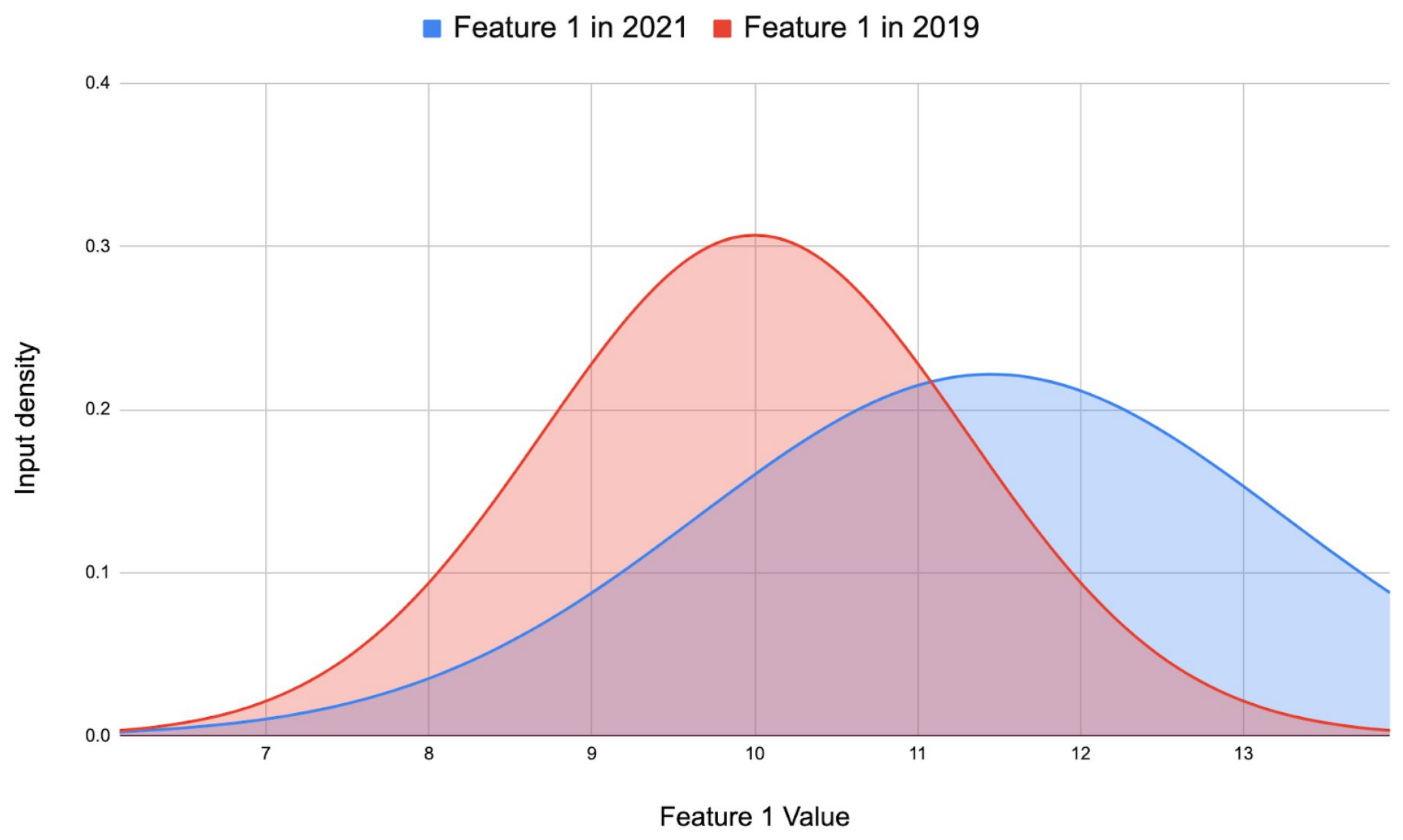
    Monitor changes in data distribution between training data and production data, detecting shifts in statistical properties of feature values over time.
- Real-world data is constantly changing, and as behavior and business context evolve, it may be

Functional Level Monitoring - The Output

- Understanding the performance of a machine learning model in production entails monitoring its output, ensuring it aligns with key performance metrics.
- **Ground truth**: In cases where ground truth labels are available, such as in ad click prediction, model predictions can be compared directly to the actual outcomes to assess performance.
- However, evaluating model predictions against ground truth is challenging in most machine learning scenarios, necessitating alternative methods.
- **Prediction drift**: When ground truth labels are unavailable, monitoring predictions is crucial. A significant shift in prediction distribution can indicate potential issues, like changes in input data structure, system misbehavior, or shifts in the real-world environment, as seen in fraud detection with a sudden increase in flagged transactions

Data Distribution Monitoring

Data drift refers to the change in model input data distribution over time, which was not accounted for during the model training phase. This change can lead to degraded model performance as the model predictions may no longer align with current data realities.



Drift in Machine Learning

Model Drift is a scenario where, ML Models is not performing as per the SLA (Service Level Agreement).
Model Performance degrade after deploying it to production, as the model can receive data that was not introduced during model training.

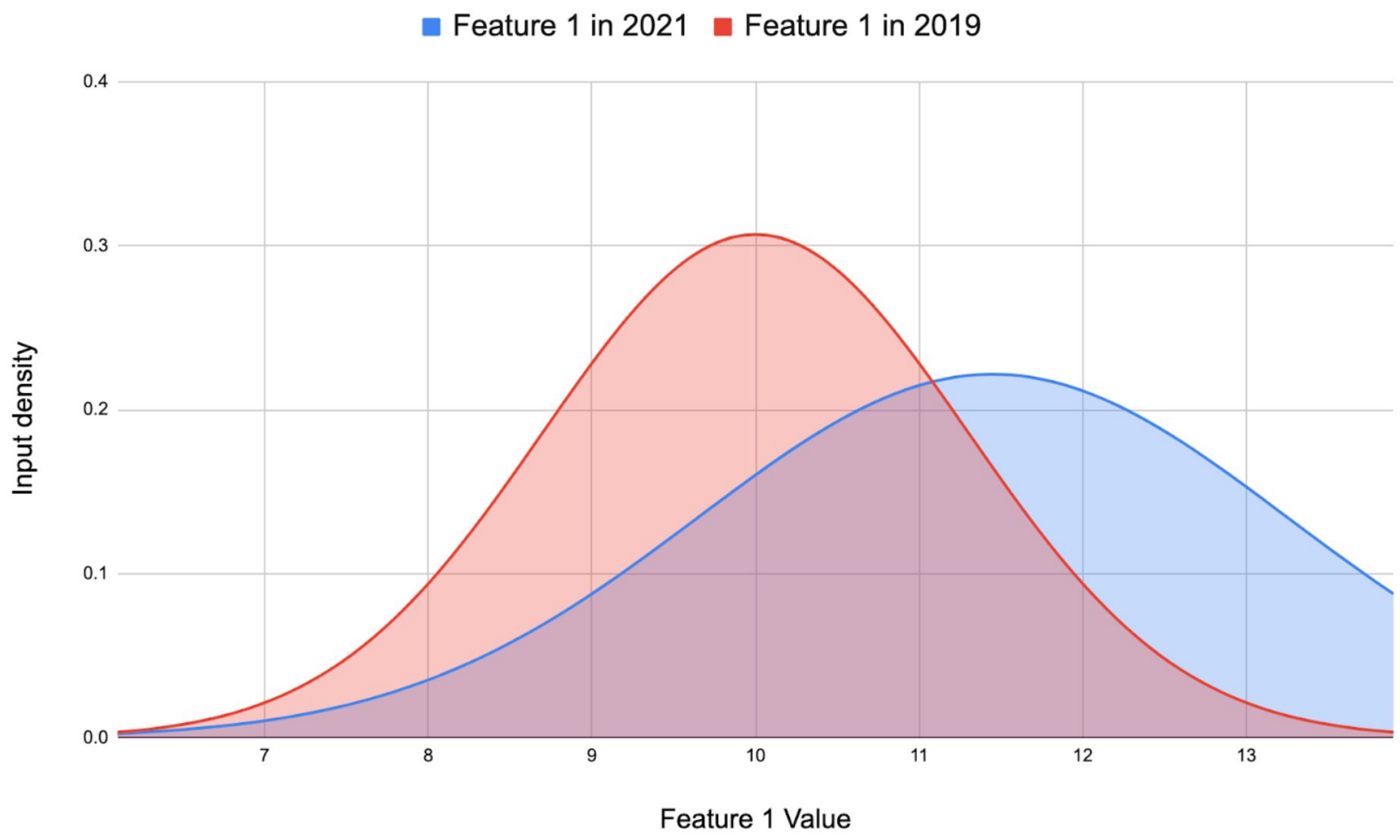
Three Types of Drift in ML		
Data Drift	**Prediction Drift**	**Concept Drift**

Types of Data Drift

- **Covariate Drift**: Occurs when the distribution of input features changes. It does not necessarily affect the output labels but might influence model predictions.
- **Prediction Drift:** Involves changes in the distribution of target variables (output labels). It directly impacts the model's predictive outcomes.
- **Concept Shift**: This type of drift signifies that the relationship between the input data and the output labels has changed. Even if the data distribution remains the same, the model may start performing poorly because the underlying patterns it learned no longer hold.

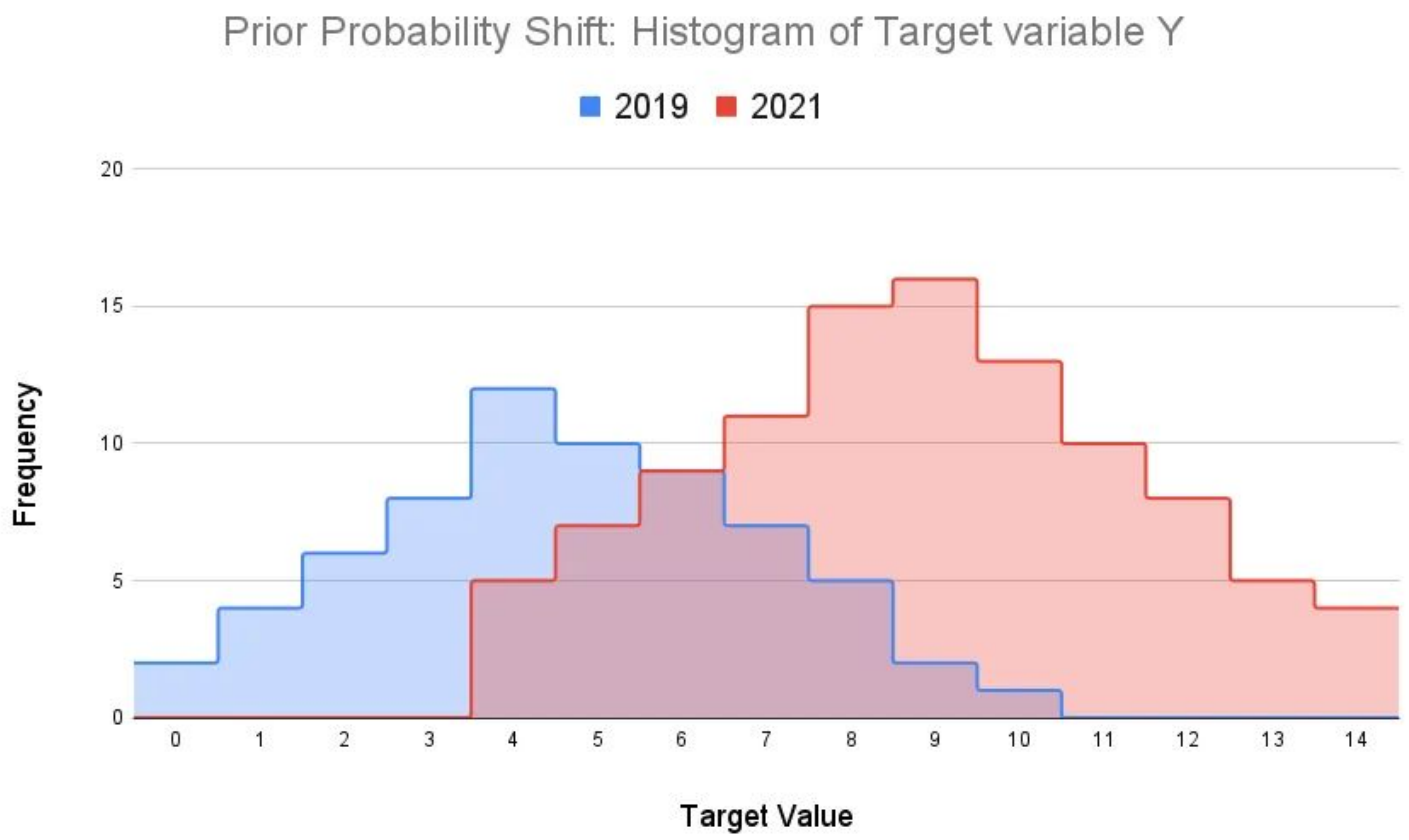
Covariate Drift

compares the distribution of "Feature 1" values between 2019 and 2021. The red curve represents the 2019 data, and the blue curve represents the 2021 data. Noticeable shifts in the distribution indicate how "Feature 1" has changed over time, with the 2019 data peaking around a value of 10, while the 2021 data peaks around a value of 12. This shift could impact the performance of machine learning models trained on earlier data.



Prediction Drift

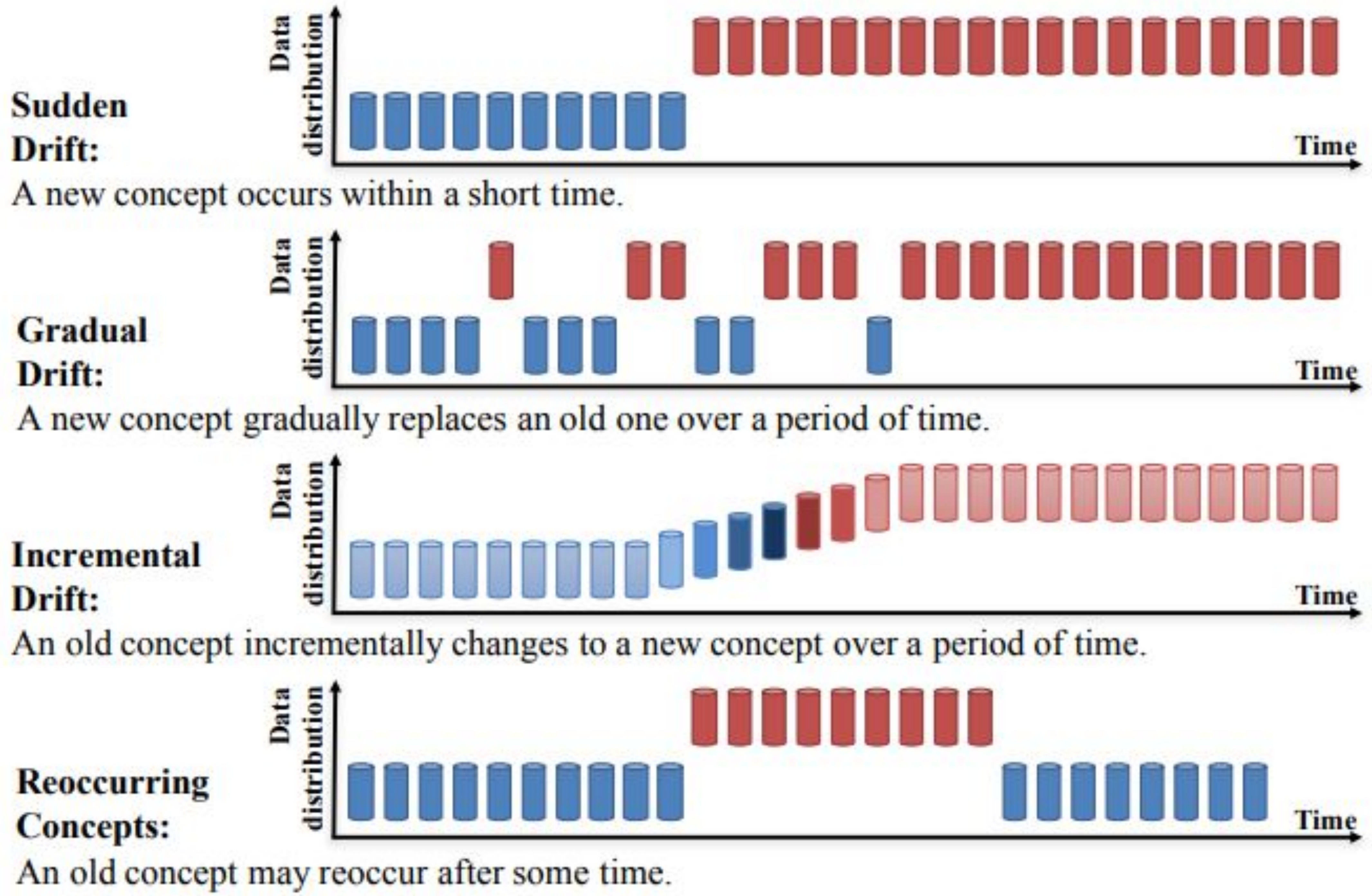
Prediction Drift in the target variable "Y." The histogram displays the distribution of the target variable for 2019 (blue bars) and 2021 (red bars). There is a clear shift in the distribution, with 2021 showing a higher frequency of larger target values compared to 2019



Concept Shift

Concept shift affects the “logic” between features and target — even if input features look the same, the meaning behind them can change!

Concept Shift



Sudden Drift:
A new concept occurs within a short time.

Gradual Drift:
A new concept gradually replaces an old one over a period of time.

Incremental Drift:
An old concept incrementally changes to a new concept over a period of time.

Reoccurring Concepts:
An old concept may reoccur after some time.

Source: *https://arxiv.org/pdf/2004.05785.pdf*

Techniques to Detect Drift in ML

Statistical Distance Metrics:

- **Statistical distance metrics** are valuable for **detecting drift** in machine learning systems.

Dimensionality Reduction:

- For datasets with **numerous independent variables**, **dimensionality reduction techniques** like **PCA (Principal Component Analysis)** can be employed.

Feature Monitoring Challenges:

- Monitoring **many features simultaneously** can **strain the system**, making it challenging to address drift.
- It can be more effective to **focus on specific critical features**.

Basic Statistical Metrics:

- **Mean**, **standard deviation**, **correlation**, and **comparisons of minimum and maximum values** can help **gauge drift** between training and current independent variables.

Advance Techniques to Detect Drift in ML

Distance measures such as:

- Population Stability Index (PSI)
- Characteristic Stability Index (CSI)
- Kullback–Leibler divergence (KL-Divergence)
- Jensen–Shannon divergence (JS-Divergence)
- Kolmogorov–Smirnov statistics (KS)

Addressing the Drift Issues

Data Quality Issues:

- Data quality issues can be resolved if there are problems with input data.

Example: transitioning from high-resolution training images to low-resolution deployment images can be addressed.

When Production Data is Insufficient:

- When production data is insufficient for training, you can combine historical data with recent production data, giving more weight to recent information.

Addressing the Drift Issues

Four Strategies for Retraining:

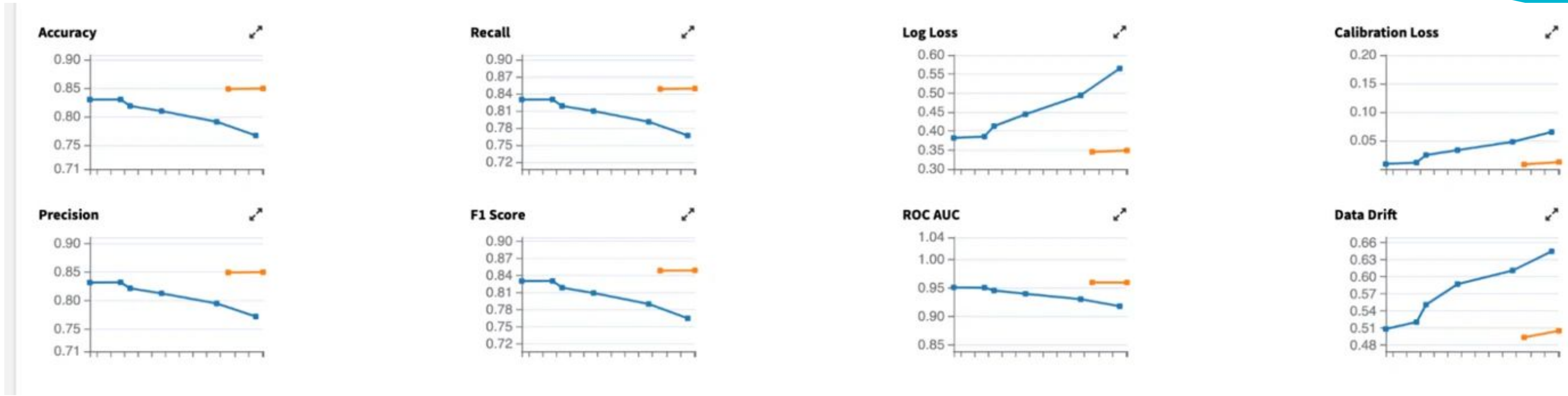
- Periodic retraining at scheduled times.
- Event-driven retraining when new data becomes available.
- Model or metric-driven retraining based on accuracy or SLA thresholds.
- Online learning for continuous real-time or near real-time model improvement.

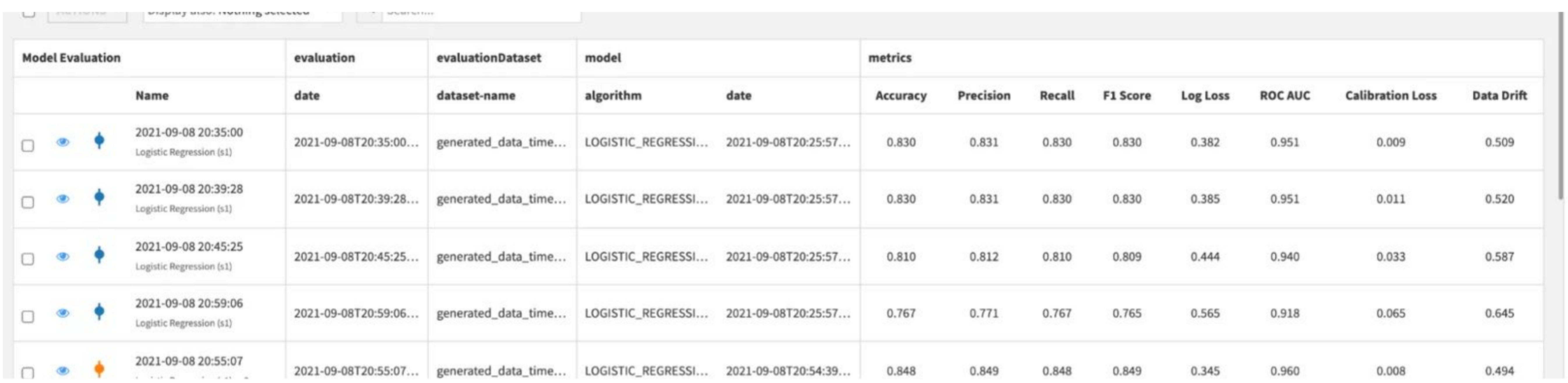
Rebuilding or Tuning the Model:

- If retraining doesn’t yield satisfactory results, rebuilding or tuning the model on recent data may be necessary.
- This process can be automated using a pipeline.

Performance Metrics

Monitoring model performance metrics is a crucial aspect of model monitoring. These metrics provide quantitative measures of how well the model is performing and help identify any degradation in its predictive capabilities.

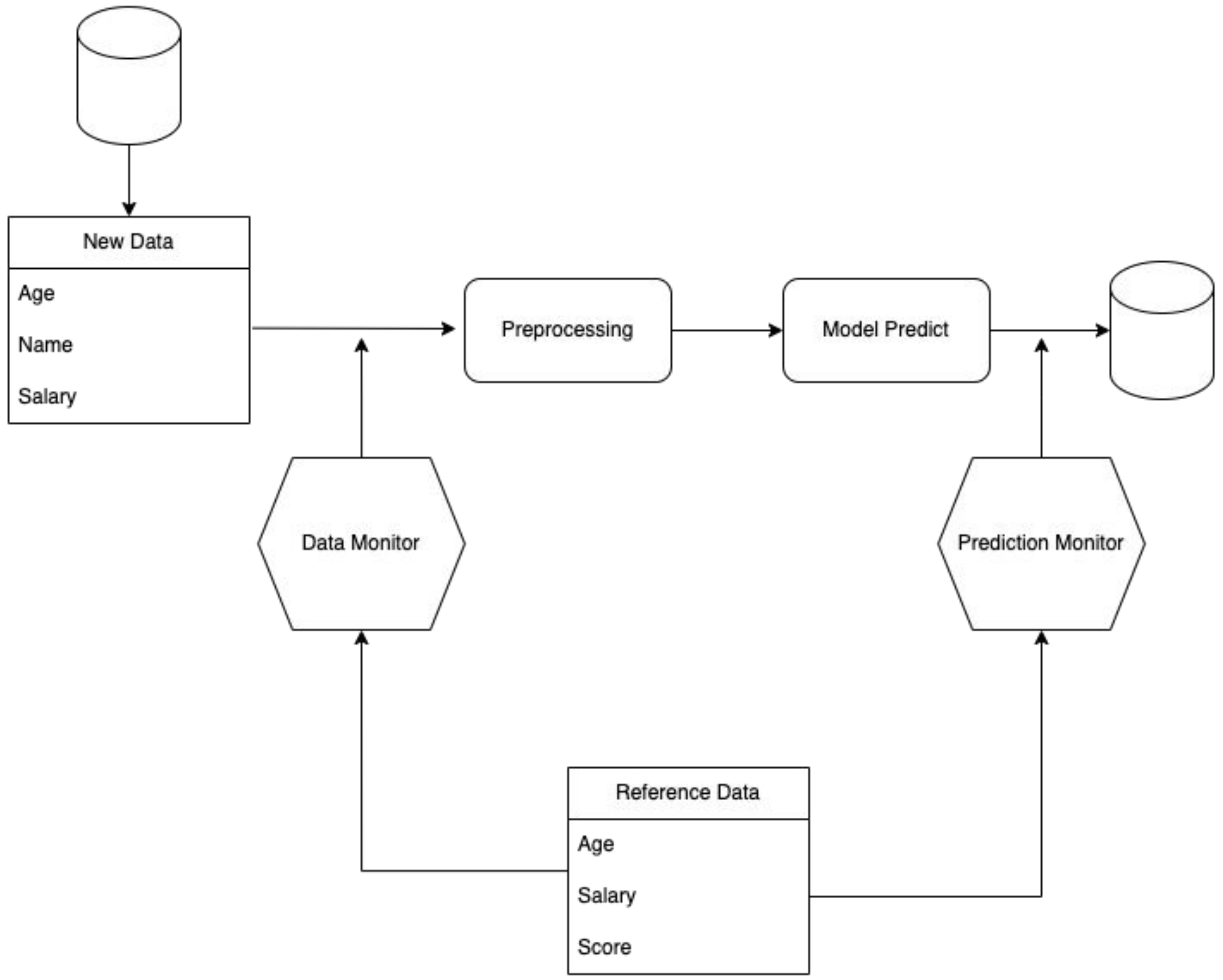




Model Evaluation	evaluation	evaluationDataset	model		metrics							
Name	date	dataset-name	algorithm	date	Accuracy	Precision	Recall	F1 Score	Log Loss	ROC AUC	Calibration Loss	Data Drift
2021-09-08 20:35:00 Logistic Regression (s1)	2021-09-08T20:35:00...	generated_data_time...	LOGISTIC_REGRESSI...	2021-09-08T20:25:57...	0.830	0.831	0.830	0.830	0.382	0.951	0.009	0.509
2021-09-08 20:39:28 Logistic Regression (s1)	2021-09-08T20:39:28...	generated_data_time...	LOGISTIC_REGRESSI...	2021-09-08T20:25:57...	0.830	0.831	0.830	0.830	0.385	0.951	0.011	0.520
2021-09-08 20:45:25 Logistic Regression (s1)	2021-09-08T20:45:25...	generated_data_time...	LOGISTIC_REGRESSI...	2021-09-08T20:25:57...	0.810	0.812	0.810	0.809	0.444	0.940	0.033	0.587
2021-09-08 20:59:06 Logistic Regression (s1)	2021-09-08T20:59:06...	generated_data_time...	LOGISTIC_REGRESSI...	2021-09-08T20:25:57...	0.767	0.771	0.767	0.765	0.565	0.918	0.065	0.645
2021-09-08 20:55:07	2021-09-08T20:55:07...	generated_data_time...	LOGISTIC_REGRESSI...	2021-09-08T20:54:39...	0.848	0.849	0.848	0.849	0.345	0.960	0.008	0.494

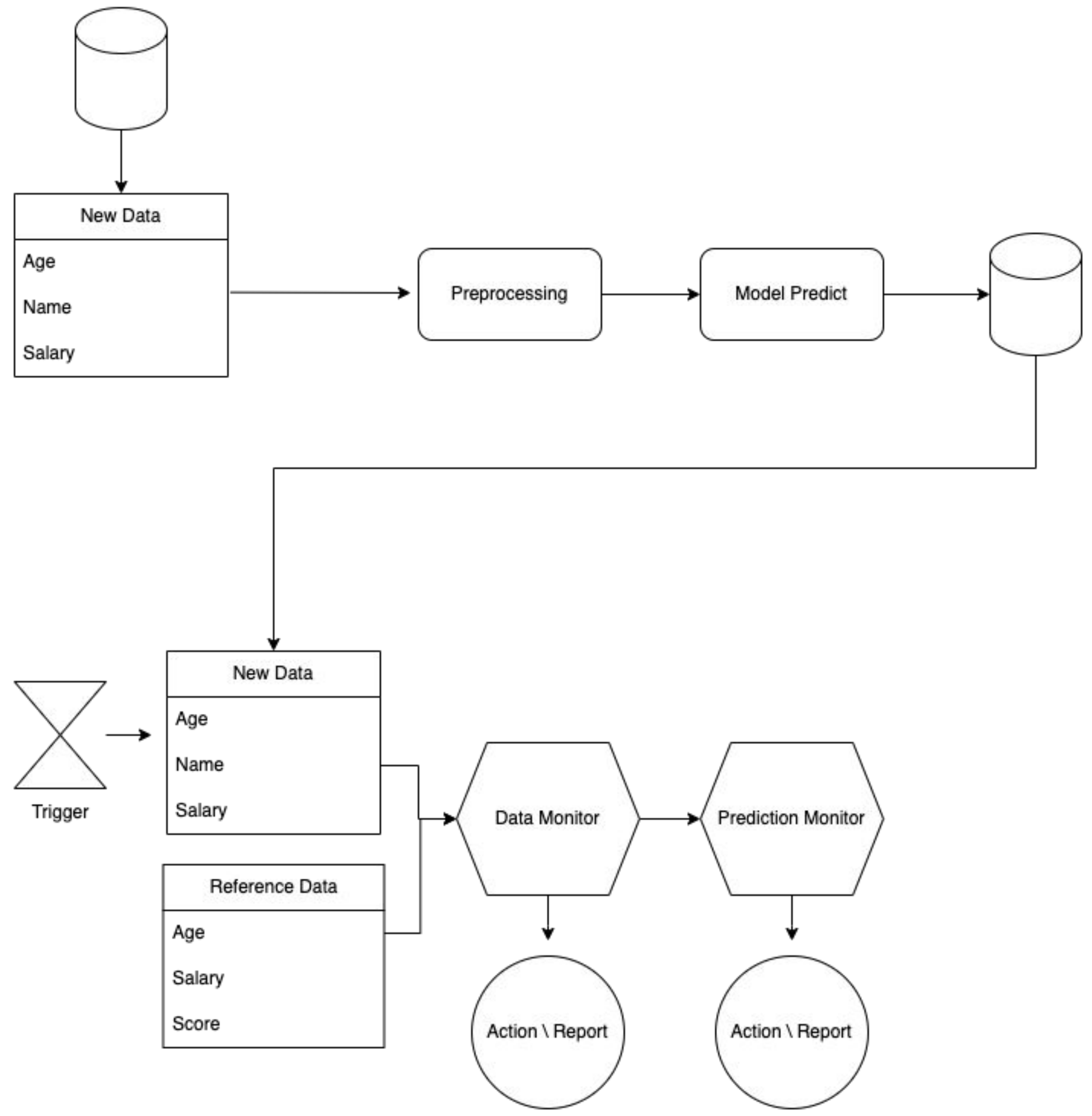
Monitroing Flow

Monitoring through flow



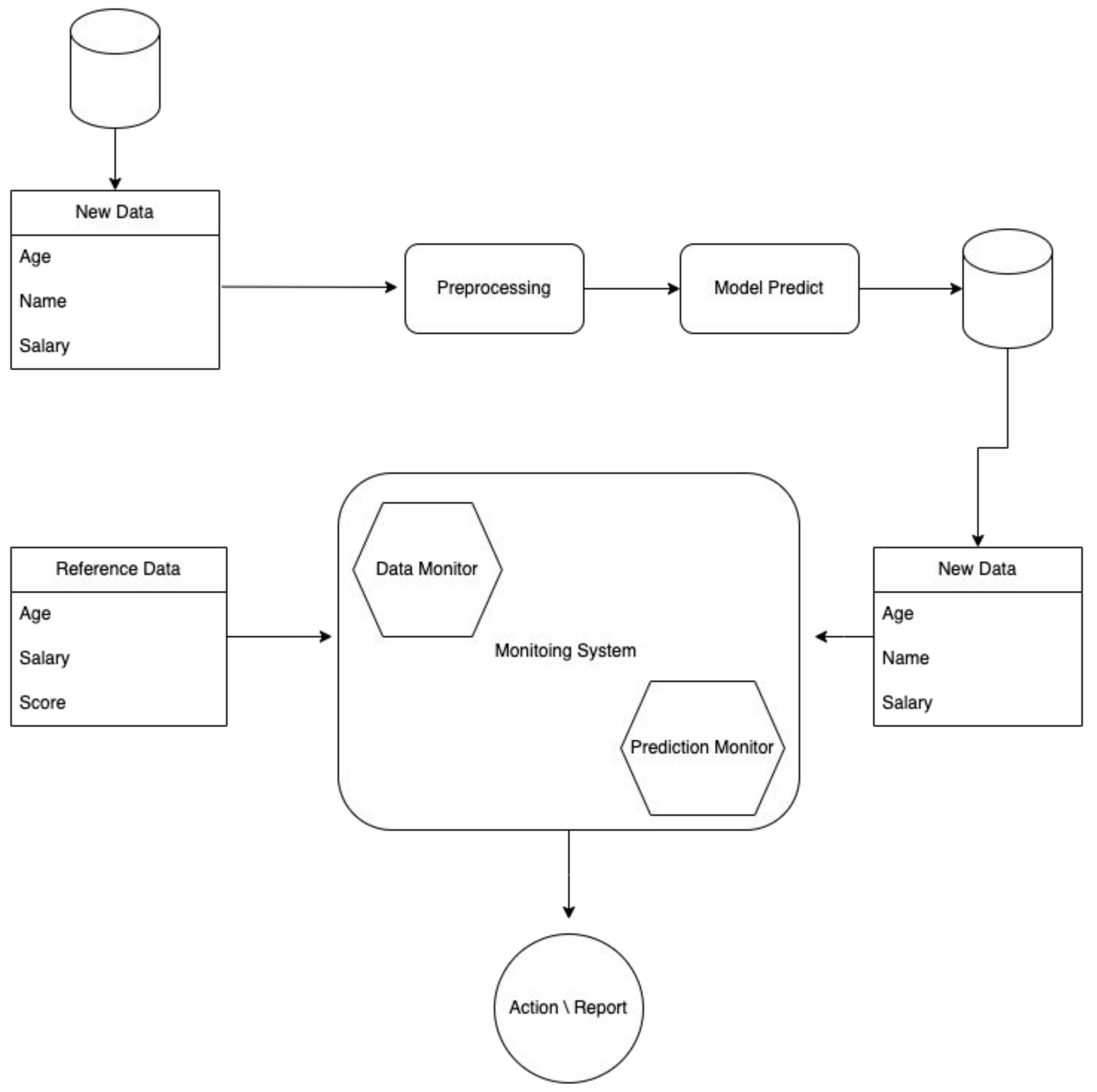
Monitroing Flow

Trigger Monitoring flow



Monitroing Flow

Online Monitoring flow



Monitoring Actions

Once potential issues or performance degradation are detected through model monitoring, appropriate actions need to be taken to mitigate the risks and maintain the reliability of the machine learning system. The following monitoring actions should be considered

- Alerting and Notifications
- Automated Model Retraining and ~~Deployment~~
- Model Rollback or Fallback Strategies
- Human Intervention and Model Governance

Monitoring Actions

Alerting and Notifications:

- Set up alerts and notifications to inform relevant stakeholders when predefined thresholds are breached.
- Integrate with existing monitoring and alerting systems for centralized management.

Automated Model Retraining and Deployment:

- Implement automated pipelines for model retraining and deployment when performance degradation is detected.
- Leverage continuous integration and continuous deployment (CI/CD) practices for seamless updates.
- Ensure proper testing, validation, and approval processes before deploying new model versions.

Monitoring Actions

<u>Model Rollback or Fallback Strategies:</u>

- Have a plan to roll back to a previous stable model version if the new model exhibits issues.
- Implement fallback strategies, such as using a simpler or rule-based model as a backup.
- Ensure smooth transition and minimize disruption to the production environment.

<u>Human Intervention and Model Governance:</u>

- Establish processes for human review and intervention when critical issues are detected.
- Involve domain experts, data scientists, and stakeholders in the decision-making process.
- Implement model governance practices, including model documentation, approval workflows, and auditing.
- Address ethical concerns, such as model bias, fairness, and transparency.

Best Practices

- When it comes to monitoring machine learning (ML) systems, there are two distinct types of monitoring: monitoring for stakeholders and monitoring for data scientists. These two types of monitoring have different objectives, metrics, and values.

- Model retraining is very uncommon in most companies. Instead, the majority of organizations rely on alerts and dashboards as their primary approach to monitoring and managing machine learning models.

- When you start building monitoring check all the next points:
  - Model Inputs (Data Distribution, Data Drift)
  - Model Outputs (Performance Metrics, Prediction Drift)
  - System Resources (CPU, Memory, Disk Usage)

Best Practices

- In many machine learning pipelines, it is a common practice to handle null values in the data. However, it is crucial to establish validation criteria to determine the acceptable level of null values in a row for the model to function effectively. For instance, if a row contains null values for 50% or more of its columns, this could potentially pose a significant issue for the model's performance.

Prometheus

Continuous Monitoring with Prometheus

What is Continuous Monitoring

An automated process by which one can observe and detect compliance issues and security threats during each phase of the DevOps/MLOps pipeline.

Continuous Monitoring Tools in DevOps

- Monitoring Tools
- Alerting Tools
- Metric Storage
- Visualization Tools

Sensu
by sumo logic

Nagios

Monitoring Tools

slack PagerDuty servicenow

Alerting Tools

Metric Storage

Visualization Tools

Grafana

Solution Prometheus and Grafana

- Prometheus provided the company an easy way to collect all the required metrics for day-to-day operations.
- Prometheus also provided the ability to monitor both modern systems and legacy systems.
- The alerting mechanism in the new monitoring system was easy to embed with the dashboards.
- Grafana made the dashboard visualization more accessible for various teams in the organization.

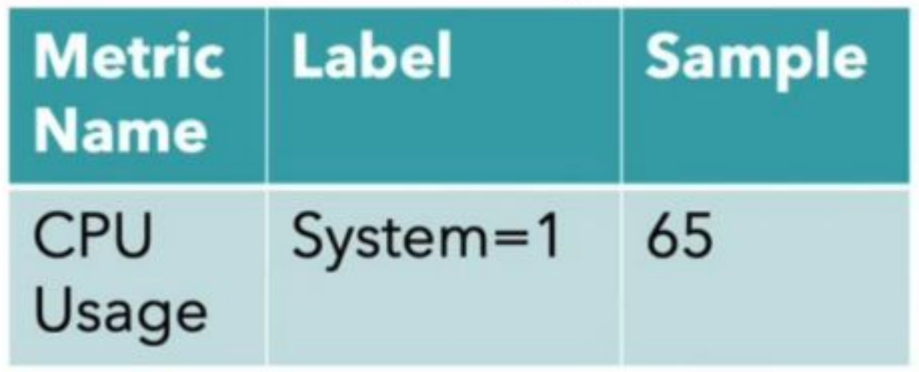
Prometheus

- An open source monitoring solution.
- Started at SoundCloud around 2012–13, and was made public in early 2015.
- Prometheus provides Metrics & Alerting.
- It is inspired by Google's Borgmon, which uses time-series data as a data source to send alerts based on this data.

Prometheus

In Prometheus, we work with Dimensional Data:

Time series are identified by a metric name and a set of key/value pairs (labels).

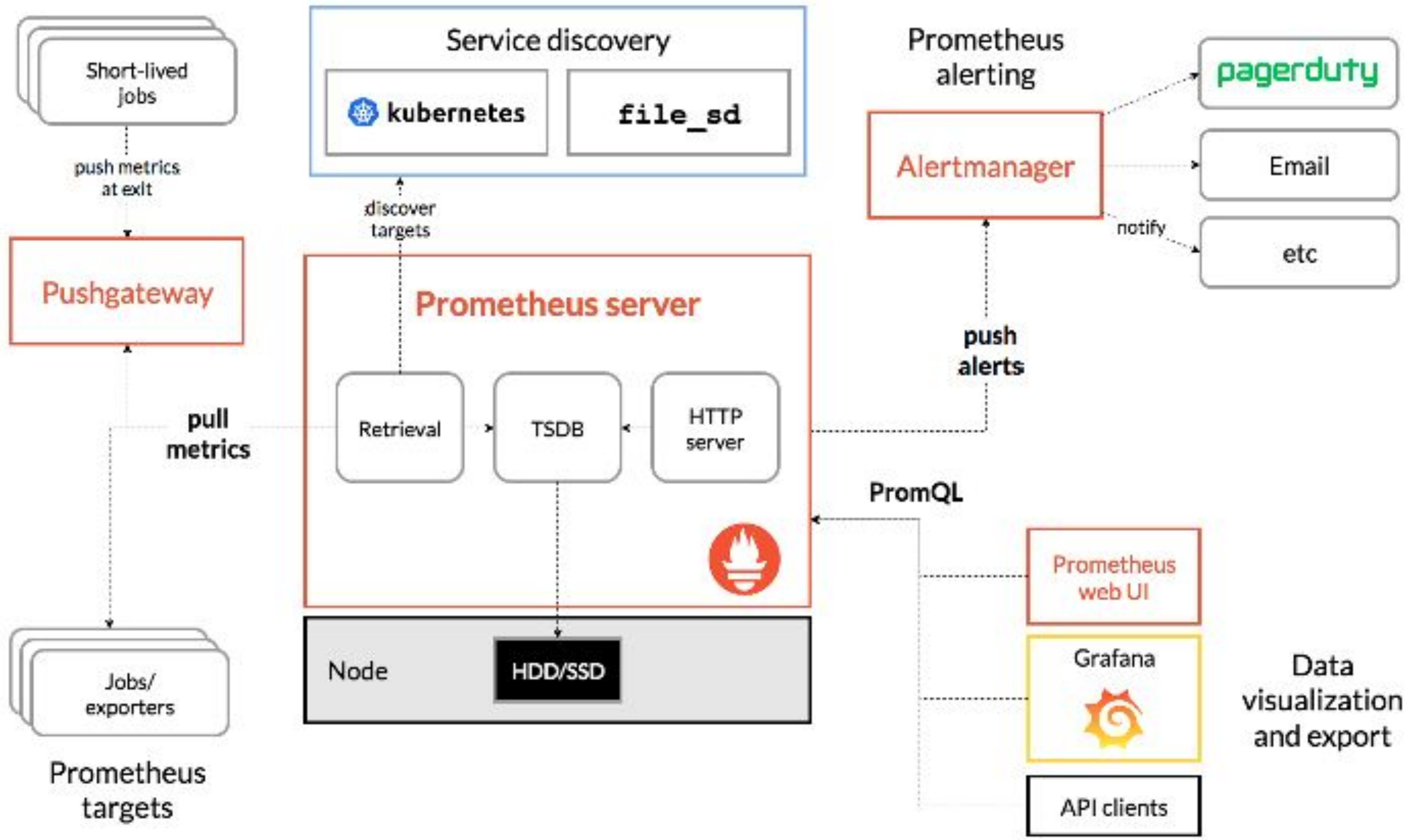


Metric Name	Label	Sample
CPU Usage	System=1	65

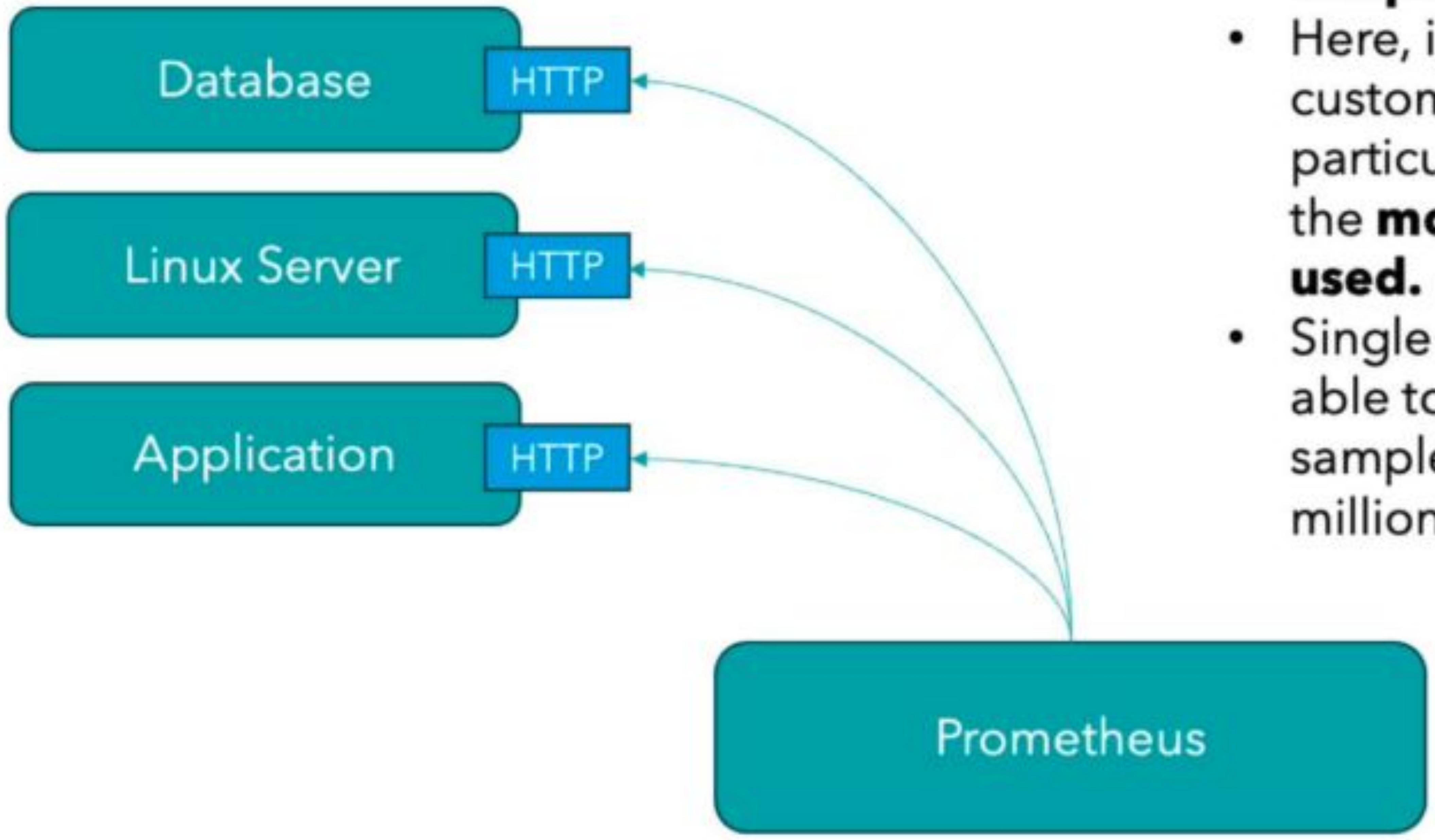
Prometheus

- Prometheus includes a Flexible Query Language — PromQL (read-only).
- Can generate visualizations using a built-in expression browser, or can be integrated with Grafana.
- It stores metrics in memory and local disk in its own custom, efficient format.
- Written in Go.
- Supports multiple client libraries and integrations available.

Prometheus architecture



Prometheus Server



- Prometheus collects the metrics from monitored targets by **scraping the metrics HTTP endpoints**
- Here, instead of running the custom scripts that check on particular services & systems, the **monitoring data itself is used.**
- Single Prometheus server is able to ingest upto one million samples per second as several million time series

Prometheus Push Gateway

- The Prometheus Pushgateway exists to allow ephemeral and batch jobs to expose their metrics to Prometheus.
- Since these kinds of jobs may not exist long enough to be scraped, they can instead push their metrics to a Pushgateway.
- The Pushgateway then exposes these metrics to Prometheus.

Exporters and Integrations

- There are a number of libraries and servers which help in exporting existing metrics from third-party systems as Prometheus metrics.
- This is useful for cases where it is not feasible to instrument a given system with Prometheus metrics directly (for example, HAProxy or Linux system stats).

  https://prometheus.io/docs/instrumenting/exporters/

Prometheus Alertmanager

- The Alertmanager handles alerts sent by client applications, such as the Prometheus server.
  - It takes care of:
    - Deduplicating alerts
    - Grouping alerts
    - Routing them to the correct receiver integrations (such as email, PagerDuty, or OpsGenie).
- It also manages silencing and inhibition of alerts.

PromQL

- PromQL, short for Prometheus Querying Language, is the main way to query metrics within Prometheus.
- You can display an expression’s return either as a graph or export it using the HTTP API.

Metric Types

The Prometheus client libraries offer four core metric types:

- Gauge
- Counter
- Summary
- Histogram

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.

Example: you can use a counter to represent the number of requests served, tasks completed, or errors.

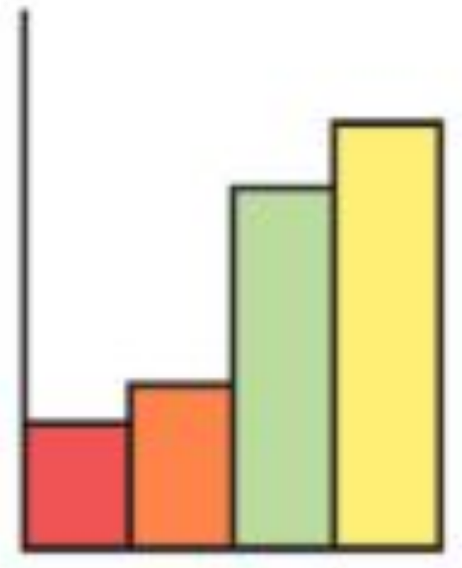
Donot use counter to expose a value that can decrease. (like – Number of currently running process)

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.

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.



Summary

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

THANK YOU