Day11 – Monitoring

02476 Machine Learning Operations

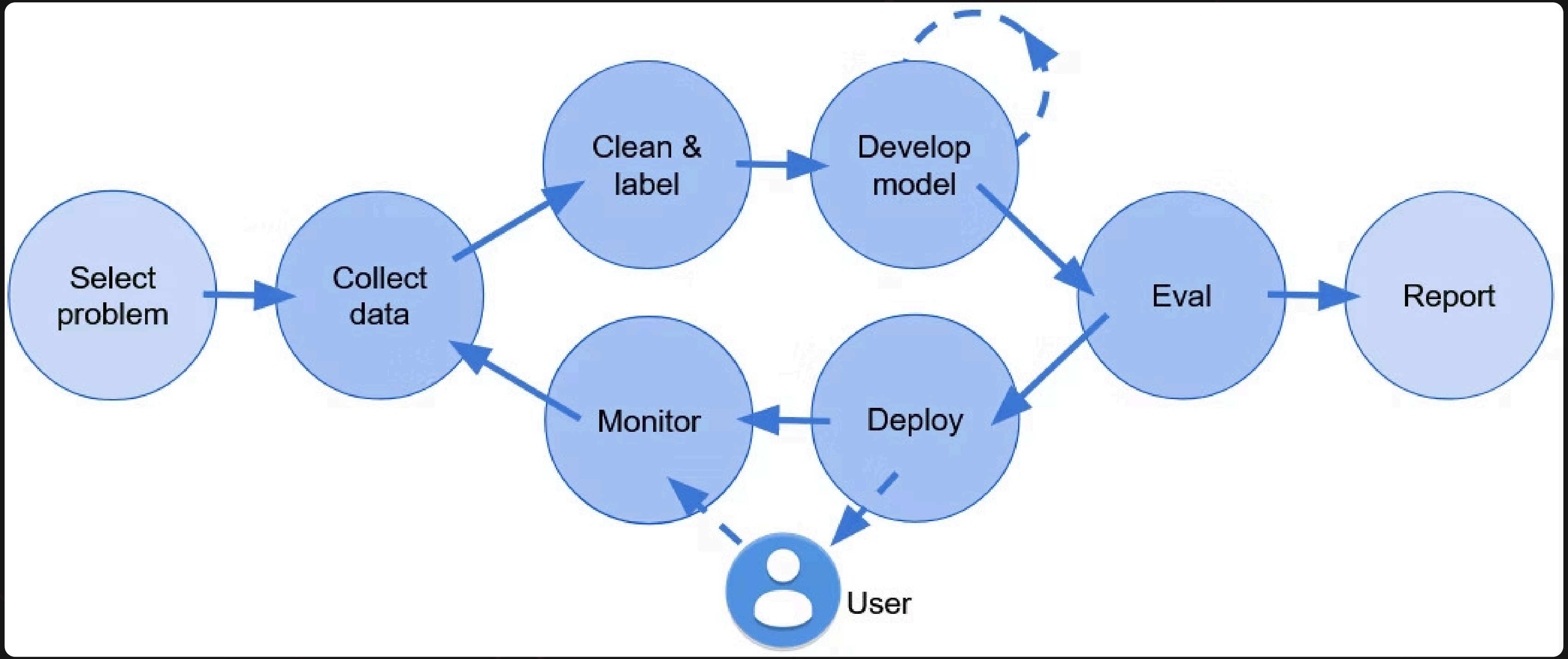
Nicki Skafte Detlefsen, Associate Professor, DTU Compute

January 2026

Monitoring create long term value

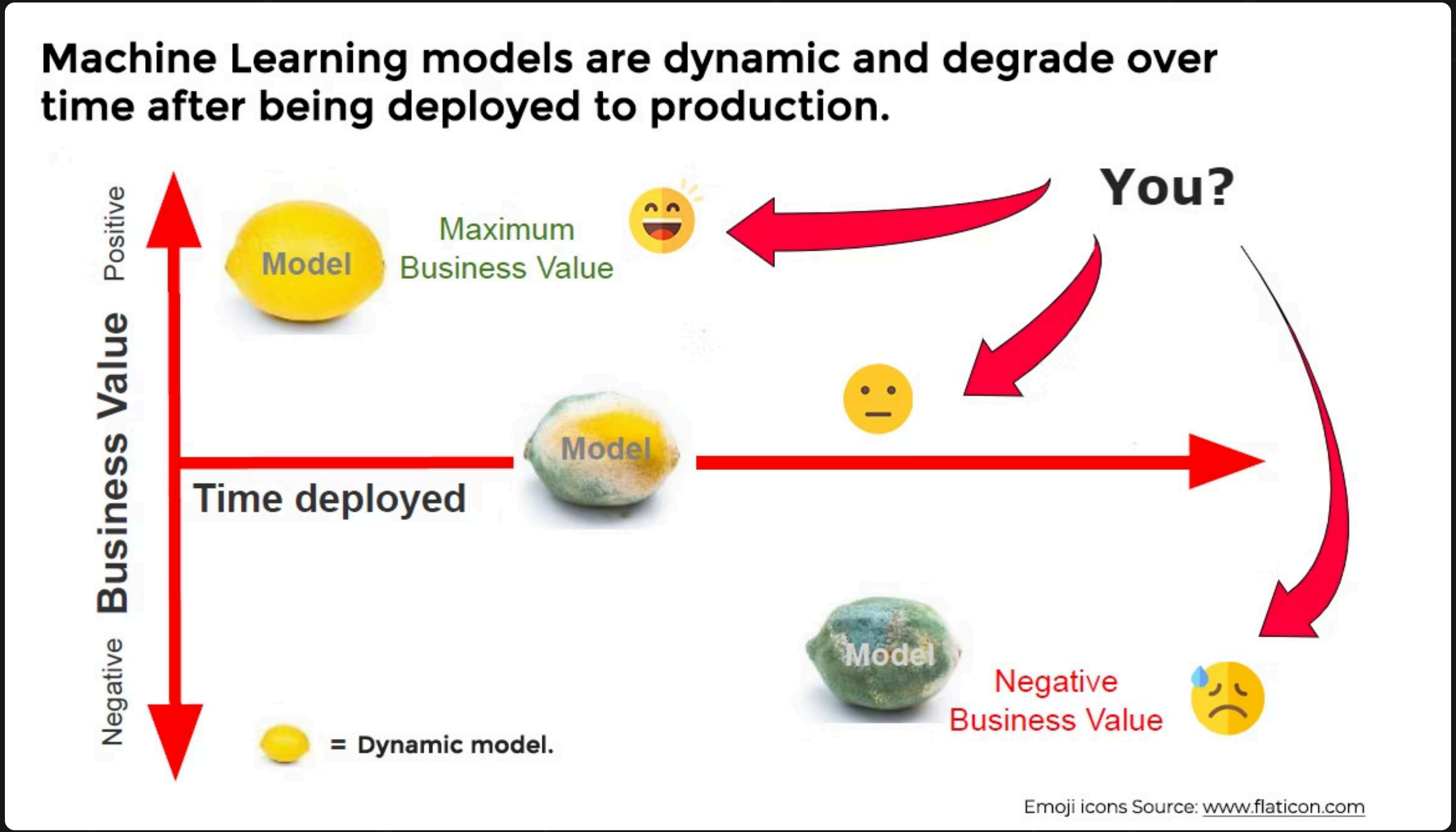
Monitoring ensures that machine learning models consistently generate accurate, reliable predictions, directly driving business outcomes. It's crucial for sustained value from your ML investments.

Without effective monitoring, models can silently degrade, and data drift can go unnoticed, eroding business value. Monitoring transforms ML operations from reactive firefighting to proactive, strategic management.

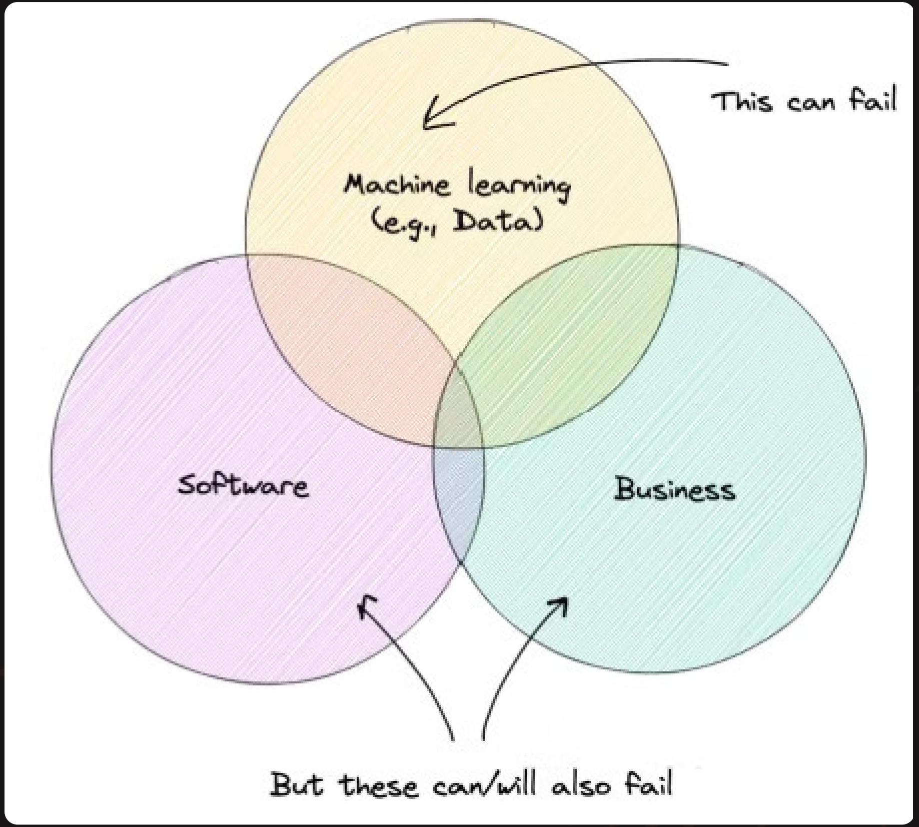


We Are in the Endgame Now

While training a model might take weeks, keeping it healthy in production is an ongoing process that can span years. Success requires vigilance, systematic monitoring, and rapid response to emerging issues.


Machine Learning models are dynamic and degrade over time after being deployed to production.
Positive
Business Value
Negative
Model
Maximum Business Value
You?
Model
Time deployed
Model
Negative Business Value
= Dynamic model.
Emoji icons Source: www.flaticon.com

What Can Fail?



ML Failures

Models degrade, data drifts, edge cases emerge, or predictions become misaligned with reality

Software Failures

Dependencies break, deployments go wrong, hardware fails, or systems crash under load

Business Failures

Models don't move key metrics, predictions aren't actionable, or ML doesn't deliver expected ROI

Software Failures

"Software is never done (only abandoned)", John Saddington

ML applications inherit all the failure modes of traditional software systems—and then add their own unique challenges on top. These failures are well-understood but no less painful when they occur in production.

Dependencies

Library version conflicts, breaking API changes, or deprecated packages can cause silent failures or outright crashes

Deployments

Configuration mistakes, environment mismatches, or rollout errors can take down production systems

Hardware

Server failures, network partitions, disk corruption, or memory exhaustion disrupt service

Downtime/Crashing

Memory leaks, unhandled exceptions, resource exhaustion, or infinite loops cause service interruptions

[1] The Burning of the Packet Ship 'Boston'" af Fitz Henry Lane

ML Failures

Machine learning systems introduce failure modes that don't exist in traditional software. These issues are subtle, difficult to detect, and can degrade system performance gradually over time without triggering obvious errors.

- **Edge Cases**

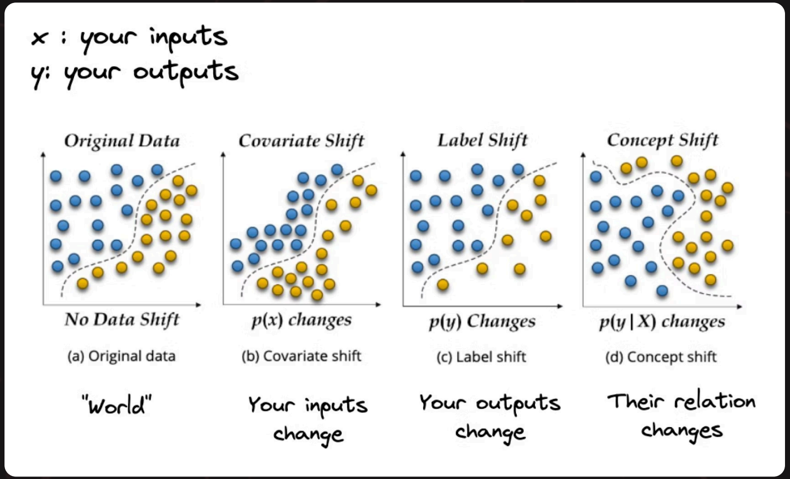
  Rare inputs or unusual combinations that weren't represented in training data produce incorrect or unstable predictions

- **Feedback Loops**

  Model predictions influence future data collection, creating self-reinforcing biases or degenerative behavior

- **Training-Production Skew**

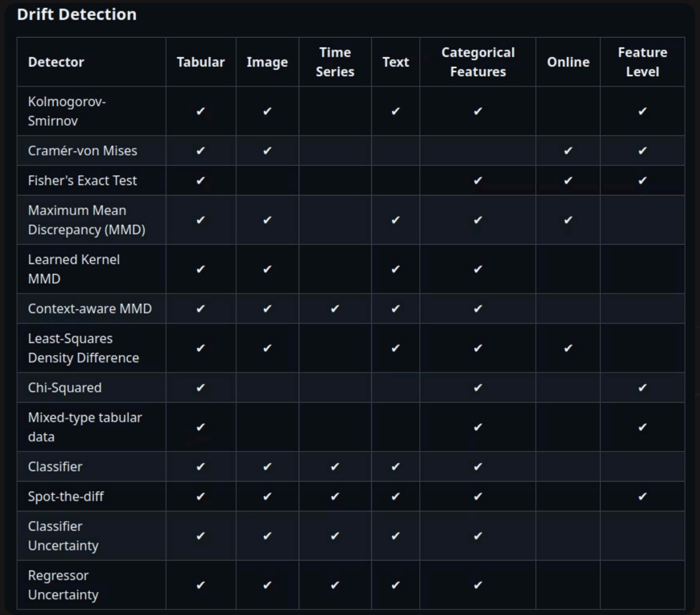
  Discrepancies between training environments and production systems cause unexpected behavior—different preprocessing, feature engineering, or inference frameworks



Drift drift and more drift

Shift Type	What Changes?	What Stays the Same?	🇩🇰 Healthcare Example
Covariate Shift	Input distribution $P(X)$	The rule $P(Y\|X$	**Training** on data from **Copenhagen** ($P_{train}(X)$). **Testing** on data from **rural Jutland** ($P_{test}(X)$). The features (average age, diet, lifestyle) are different, but the medical rule (e.g., high sugar + low exercise still increases risk) is the same.
Label Shift	Label distribution $P(Y)$	Features for each label $P(X\|Y$	**Training** on the **general population** ($P_{train}(Y)$), where diabetes prevalence is ~8%. **Testing** inside a **specialized diabetes clinic** at Rigshospitalet ($P_{test}(Y)$), where prevalence is ~90%. The features of a diabetic patient are the same, but the *frequency* of the "diabetes" label has drastically changed.
Concept Shift	The rule $P(Y\|X$	(Often $P(X)$)	**Training** on data from **1990**, where "high fat intake" (X) was a clear risk factor for diabetes (Y). **Testing** on data from **2025**, where new LCHF (Low Carb High Fat) diets are popular. The *meaning* of the feature "high fat intake" has changed. The old rule $P(Y\|X$ is now wrong or incomplete.

How to detect drift?

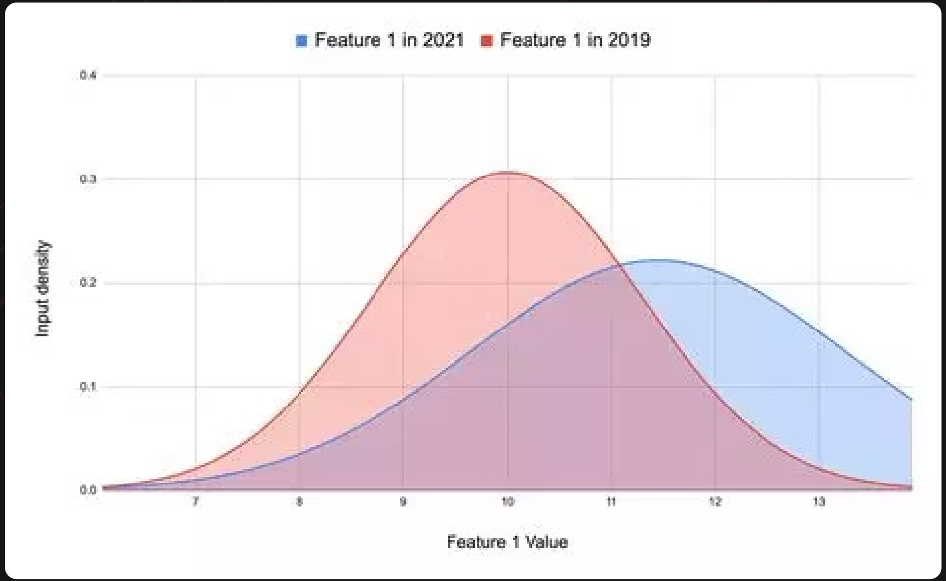


Drift Detection

Detector	Tabular	Image	Time Series	Text	Categorical Features	Online	Feature Level
Kolmogorov-Smirnov	✓	✓		✓	✓		✓
Cramér-von Mises	✓	✓				✓	✓
Fisher's Exact Test	✓				✓	✓	✓
Maximum Mean Discrepancy (MMD)	✓	✓		✓	✓	✓	
Learned Kernel MMD	✓	✓		✓	✓		
Context-aware MMD	✓	✓	✓	✓	✓		
Least-Squares Density Difference	✓	✓		✓	✓	✓	
Chi-Squared	✓				✓		✓
Mixed-type tabular data	✓				✓		✓
Classifier	✓	✓	✓	✓	✓		
Spot-the-diff	✓	✓	✓	✓	✓		✓
Classifier Uncertainty	✓	✓	✓	✓	✓		
Regressor Uncertainty	✓	✓	✓	✓	✓		

Depends on what your data looks like, but in general detect when the two

"data distributions does not overlap enough anymore"



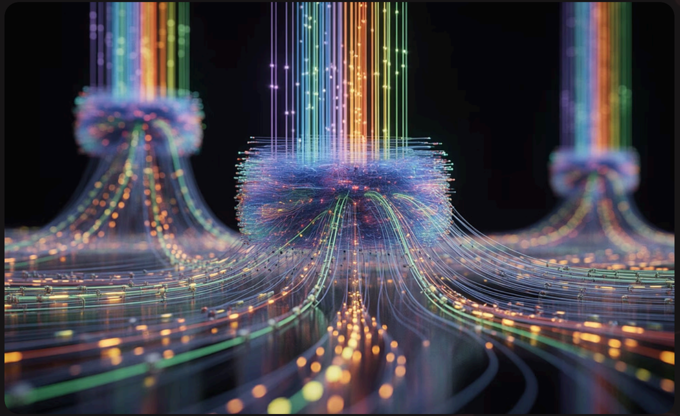
[1] https://github.com/evidentlyai/evidently

[2] https://github.com/great-expectations/great_expectations

[3] https://github.com/SeldonIO/alibi-detect

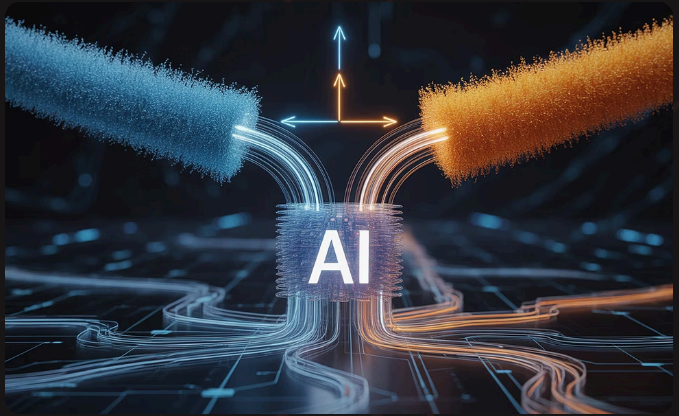
Made with GAMMA

Addressing Drift



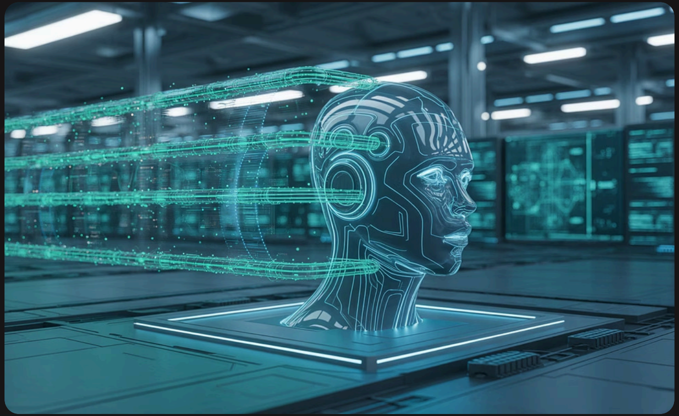
Extensive Initial Training

Train models on large, diverse datasets to capture varied patterns and improve generalization across different conditions.



Domain Adaptation & Transfer Learning

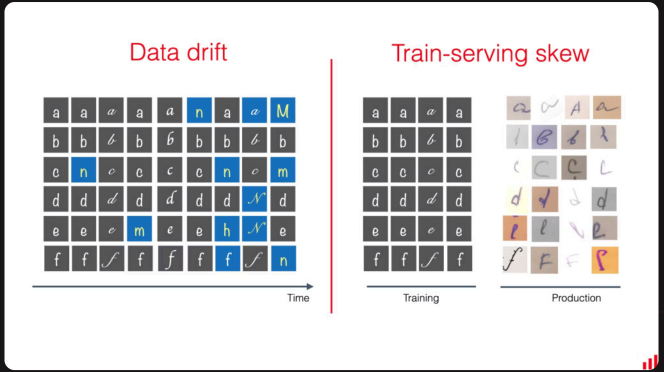
Apply domain adaptation techniques to large pre-trained models, leveraging transfer learning to adjust to new data distributions.

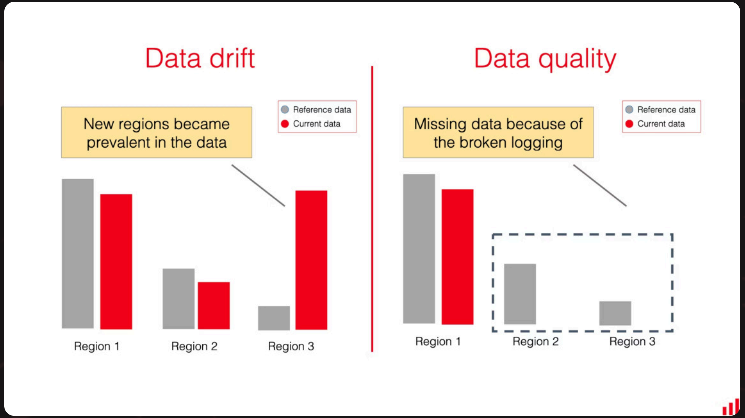


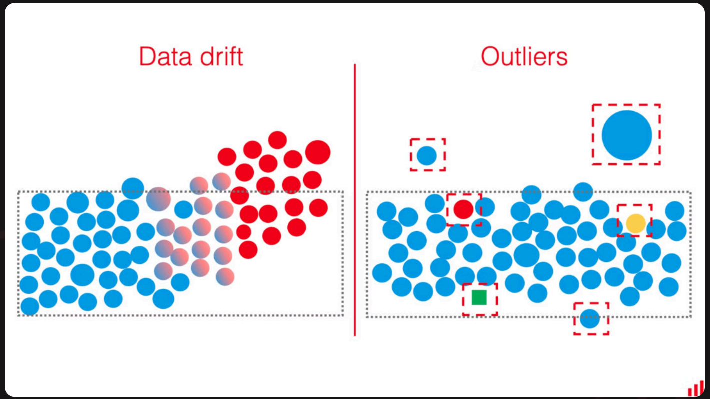
Continuous Retraining & Fine-tuning

Retrain from scratch or fine-tune models regularly using recent data to ensure they remain aligned with current reality and evolving trends.

Other kind of ML related errors







- Train-serving skew
- Data quality issues
- Outliers

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

ML Failures: Getting Labels

To retrain you need labels - how to get them?

Hand Labels

Manual annotation provides high-quality labels but is slow and expensive, best for small-scale or high-value monitoring.

Natural Labels

Leverage labels that emerge naturally over time, such as loan repayment outcomes, to measure historical performance.

Programmatic Labels

Use proxy signals and weak supervision (e.g., user clicks) to approximate true labels and estimate model performance.

An example (what do we think OpenAI can get out of me):

ChatGPT - Python protocol explanation

Exercise 2: The "Natural Label" Lag

> "You have a model predicting if a user will default on a 5-year loan. When do you get the 'natural label'? How do you know if the model is failing before that label arrives?"

Exercise 1: The "Natural Label" Lag

> "You have a model predicting if a user will default on a 5-year loan. When do you get the 'natural label'? How do you know if the model is failing before that label arrives?"

Exercise 1: The Natural Label Lag - Solution

The natural label for a 5-year loan default model is confirmed only at term end or upon actual default, leading to significant feedback delay. To detect failure sooner, monitor:

Leading Indicators (Proxies)

Monitor early signs like missed payments, credit score drops, or employment changes.

Feature Drift (Covariate Shift)

Track changes in input data distribution (e.g., average applicant credit scores). Shifts indicate likely performance degradation.

Prediction Drift

Observe shifts in model output distribution (e.g., unexpected changes in predicted default rates). This flags real-world changes or data pipeline issues.

Macro-Economic Indicators

Monitor external factors like rising interest rates or unemployment, signaling changes in the model's operating environment.

The "Cold Start" Problem in ML Monitoring

When deploying a new machine learning model, initial ground truth labels are often scarce, creating a significant challenge for effective monitoring.

Phase 1: Zero Ground Truth

Focus on data integrity and proxy indicators to ensure model health before true labels become available.

- Input Data Quality
- Input Distribution Shifts
- Output Distribution Anomalies

Phase 2: Labels Arrive

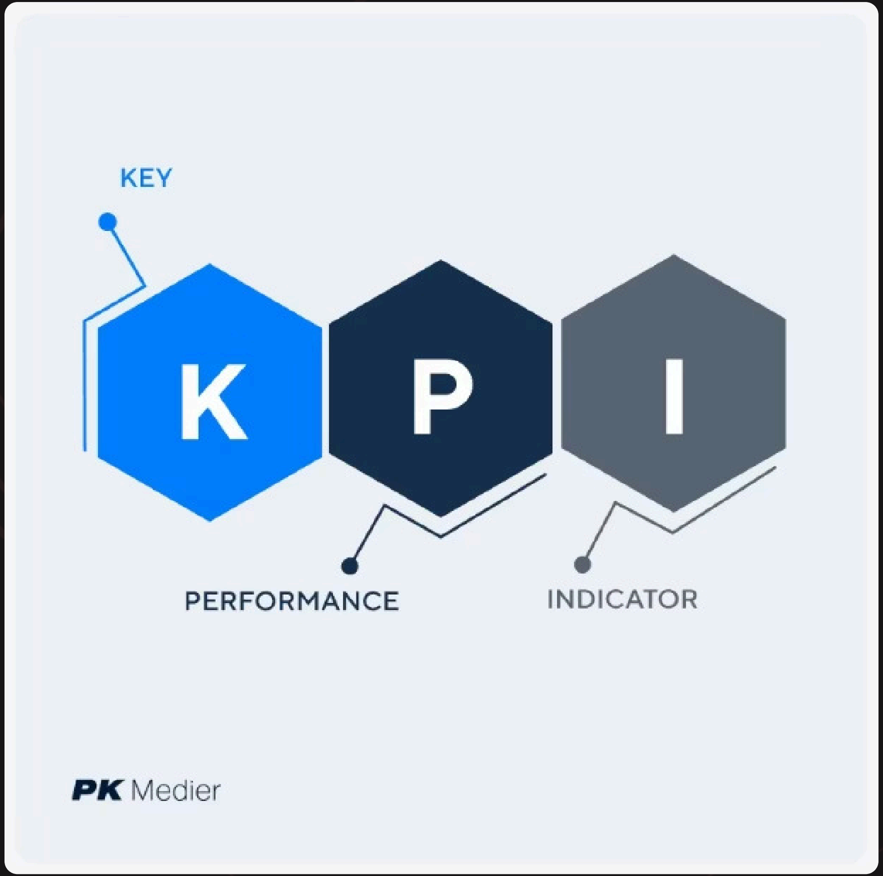
As ground truth data accumulates, evaluate direct model performance and detect nuanced forms of drift.

- Model Performance Metrics
- Concept Drift
- Model Bias & Fairness

Business Failures

- **Track outcome KPIs, not only model metrics.** (e.g., conversions, refund rate, churn, cost per acquisition, time-to-resolution)
- **Map ML signals → business impact.** Link AUC/accuracy → expected revenue or risk changes and validate that link continuously.
- **Detect downstream effects & unintended behavior.** Monitor customer complaints, support load, revenue dips, false positives/negatives' costs.
- **Have governance & playbooks.** Clear owners, escalation paths, rollback criteria and SLAs for business impact incidents.

Solution: Ensure strong business alignment from the start by defining clear metrics, fostering ML/business feedback, and validating real impact.



Exercise 1: Identifying Failure Modes

> Imagine you are running a music recommendation app. Identify one Software failure, one ML failure, and one Business failure that could occur this week.

Exercise 1: Identifying Failure Modes - Solution

Imagine you are running a music recommendation app. Identify one Software failure, one ML failure, and one Business failure that could occur this week.

Software Failure

A critical bug in the API handling song requests leads to intermittent playback errors for 20% of users, causing frustration and app crashes.

ML Failure

A sudden surge in a new music genre, not well-represented in the training data, causes the recommendation algorithm to suggest irrelevant tracks, leading to a drop in user engagement metrics.

Business Failure

Due to poor recommendations and technical issues, user retention drops by 5% this week, directly impacting subscription renewals and new user sign-ups, leading to projected revenue loss.

How do we prevent failures? — with monitoring

Failures

- Software
- ML
- Business

Monitoring

- Observability
- Data & Model
- Business KPIs
- Synthetic tests

Outcomes

- Alerts
- Rollback / Mitigation
- Business decision
- Continuous improvement

Detect the right thing

Instrument errors, data drift, model performance, and business KPIs — not only loss/accuracy.

Prioritize & alert on impact

Map signals to business SLOs so alerts reflect customer or revenue risk.

Make response automatic & owned

Automated mitigation for known failure modes + clear runbooks and a business owner for go/no-go.

Close the loop

Postmortems, retraining triggers, and controlled experiments (canaries / shadow / A/B) to prevent recurrence.

The Three Pillars of Observability

Logs

💡 Logs are textual or structured records generated by applications

💡 They provide a detailed account of events, errors, warnings, and informational messages that occur during the operation of the system

💡 Logs are essential for diagnosing issues, debugging, and auditing.

Traces

💡 Traces are detailed records of specific transactions or events as they move through a system.

💡 A trace typically includes information about the sequence of operations, timing, and dependencies between different components.

💡 Traces help in understanding the flow of a request or a transaction across different components.

Metrics

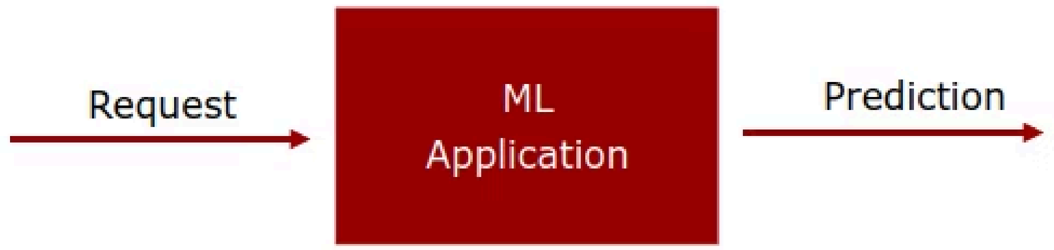
💡 Quantitative measurements of the system.

💡 They are usually numbers that are aggregated over a period of time. E.g. the number of requests per minute.

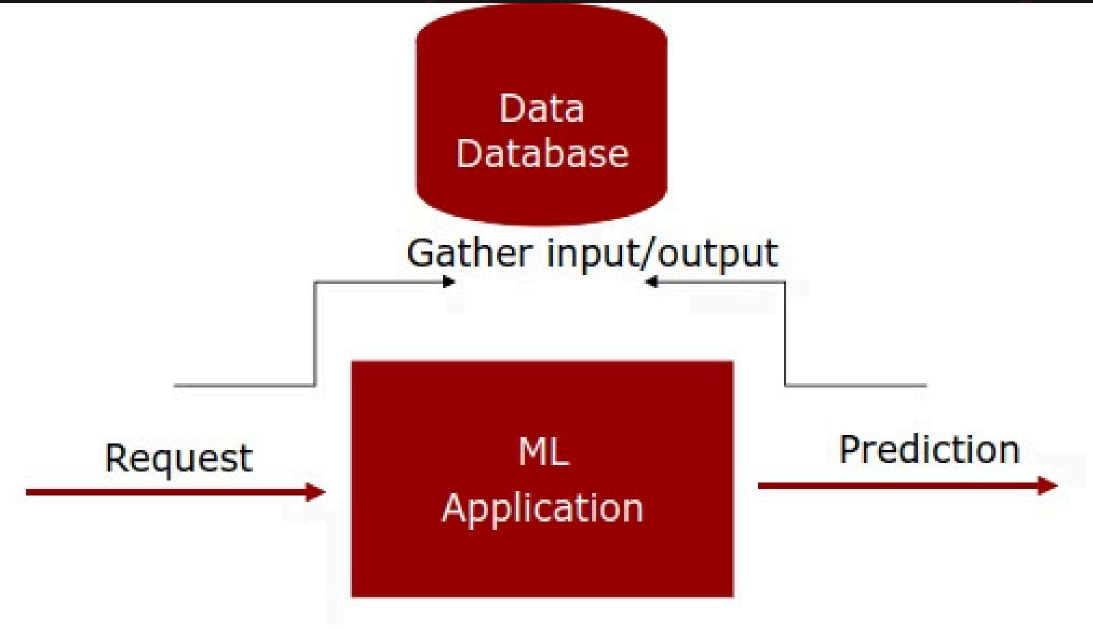
💡 Metrics are used to get an overview of the system

Grafana loki OpenTelemetry Grafana

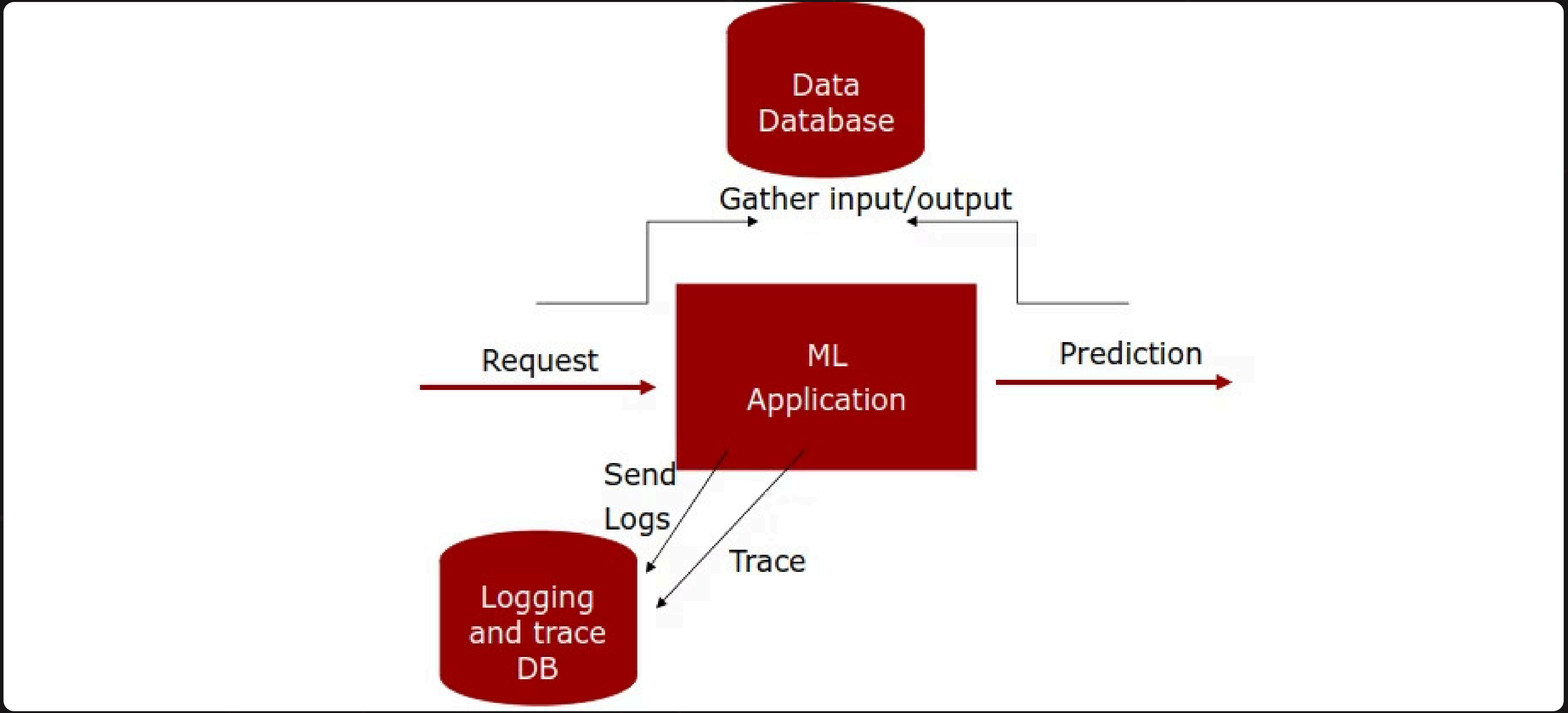
Let setup a monitored system



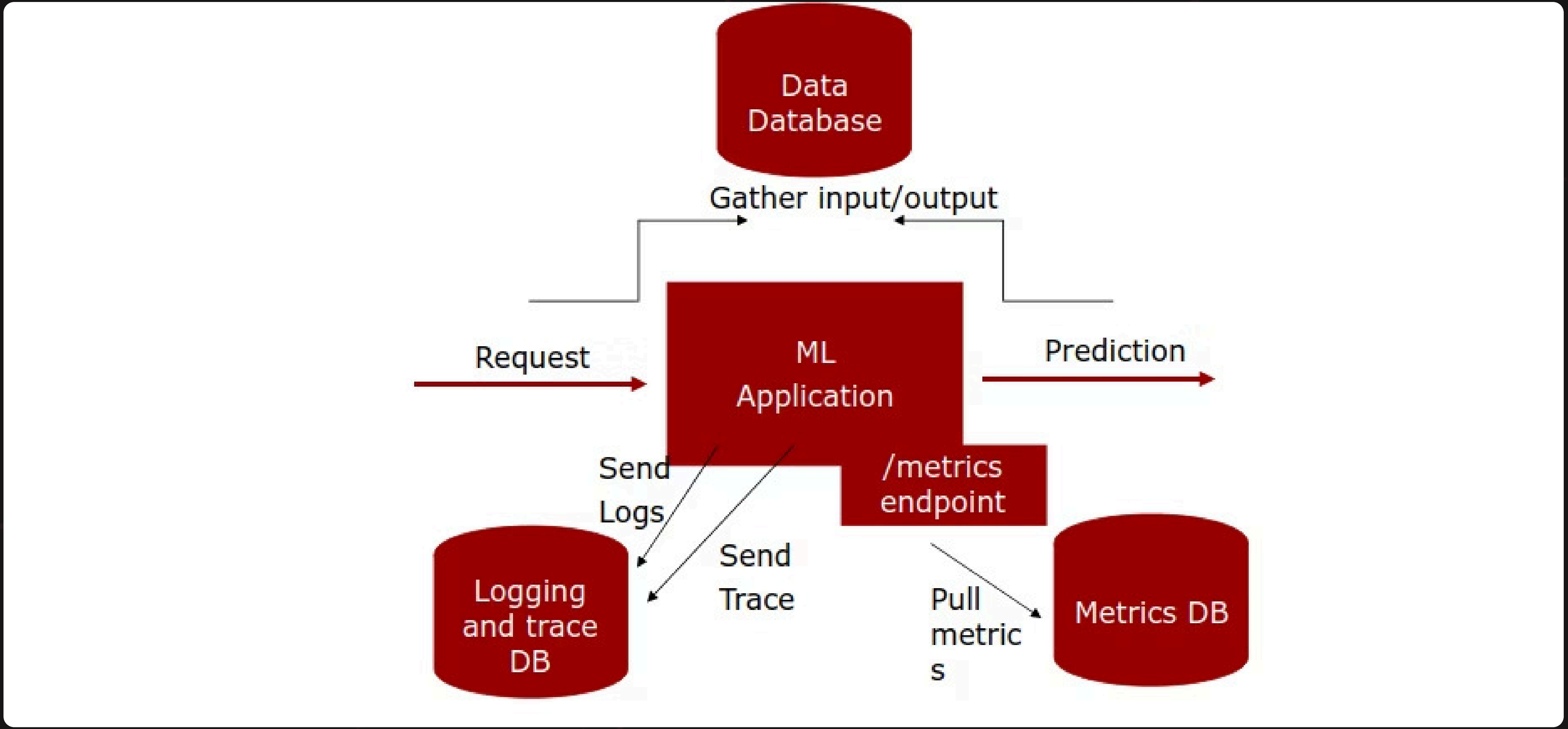
Let setup a monitored system



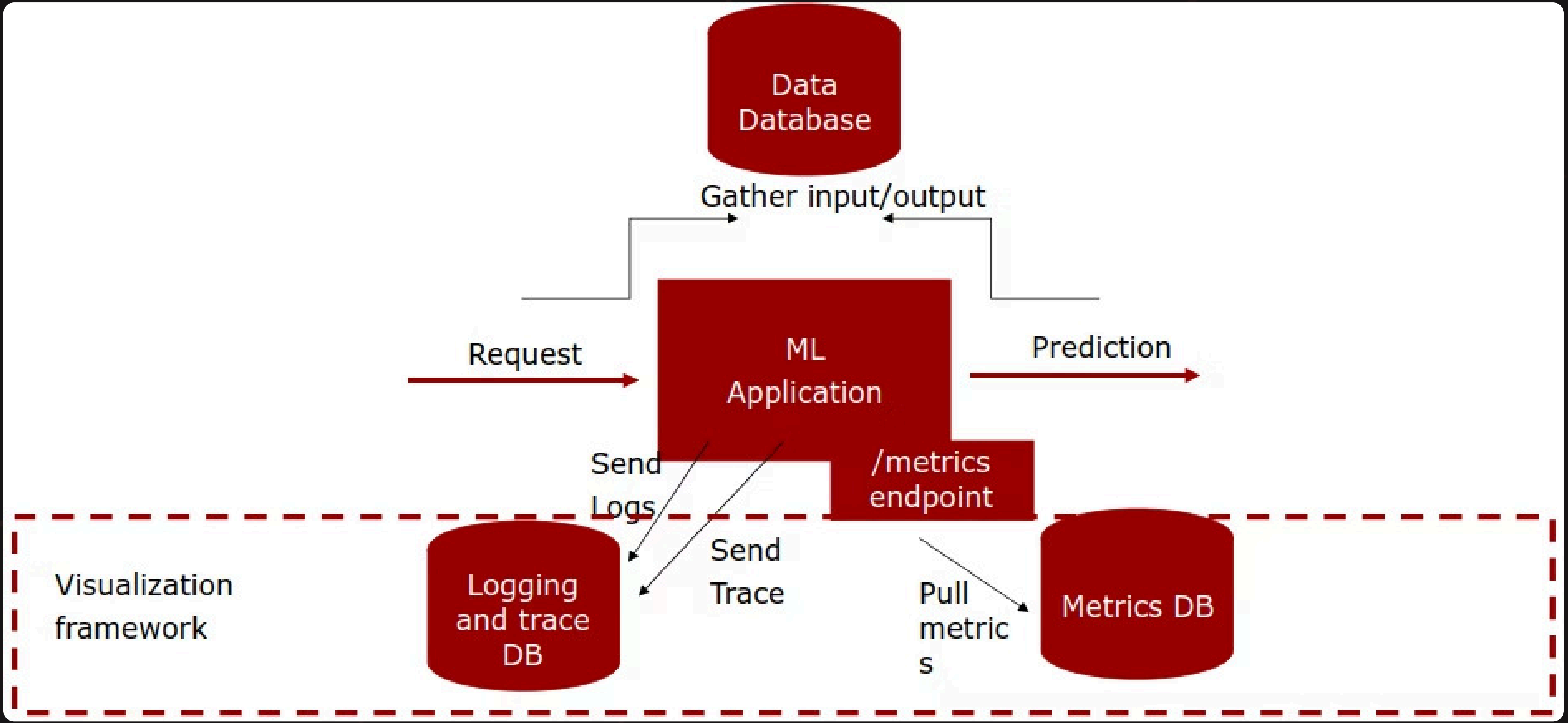
Let setup a monitored system



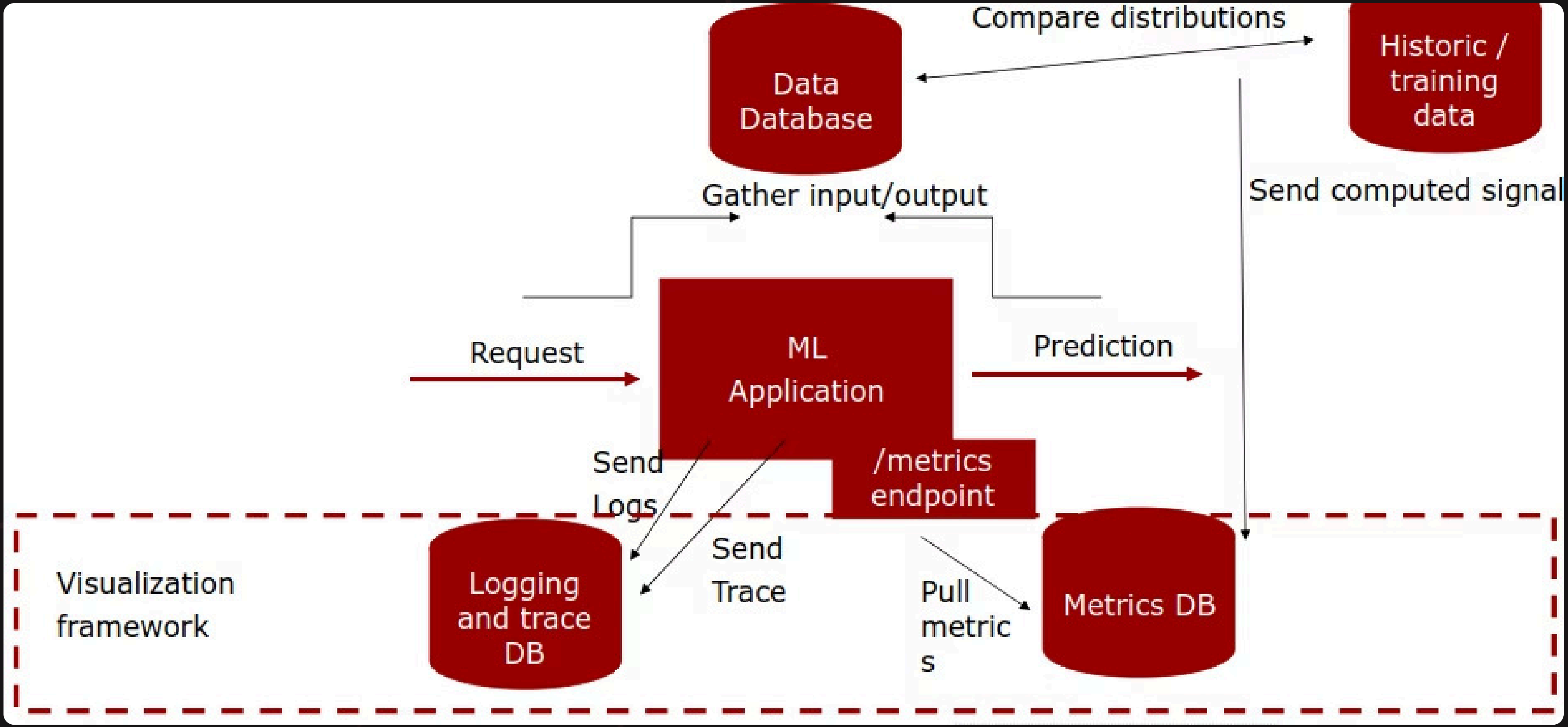
Let setup a monitored system



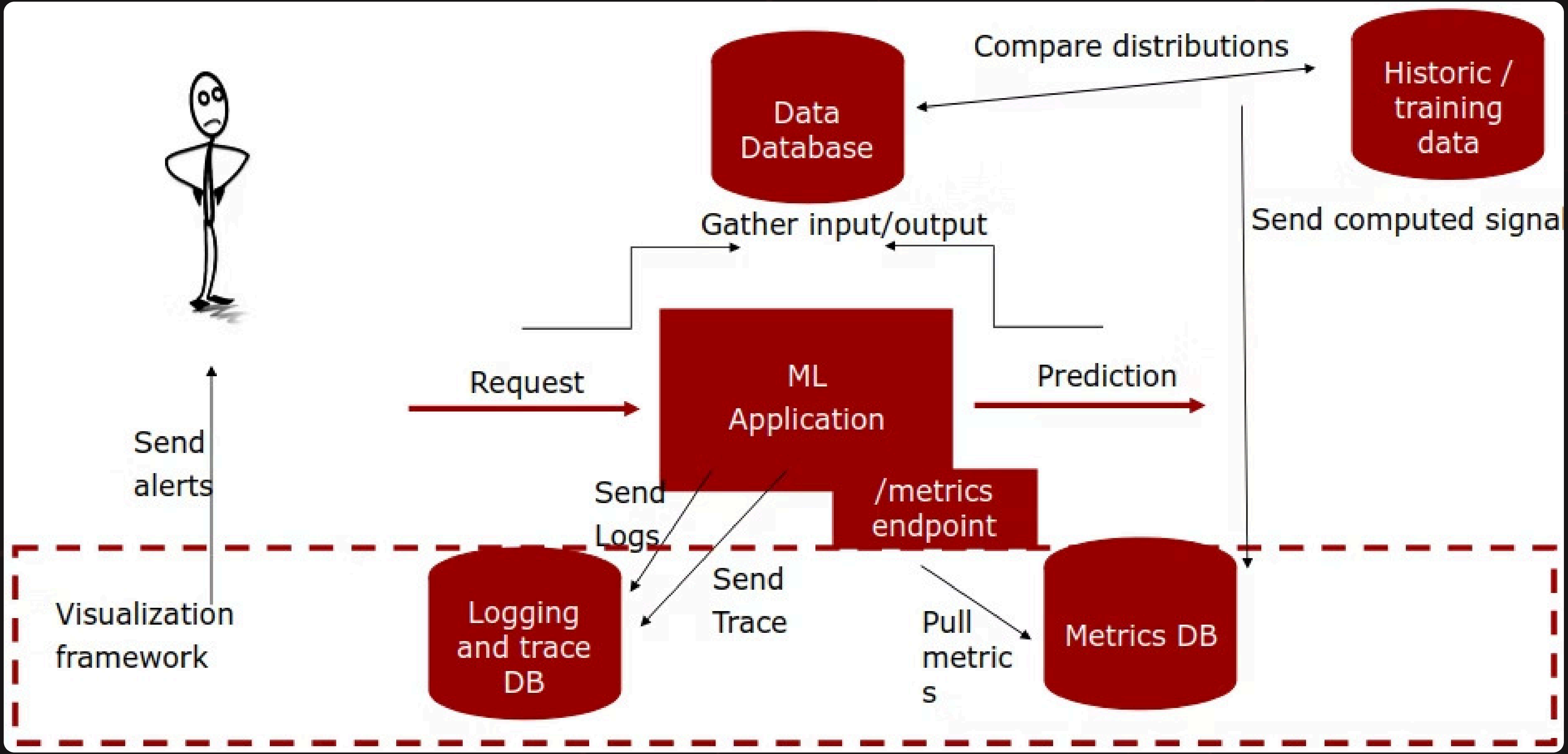
Let setup a monitored system



Let setup a monitored system



Let setup a monitored system



Made with GAMMA

Be Aware of Alert Systems

Alert systems face the "Goldilocks problem"—finding the balance between too many alerts and too few is notoriously difficult.

Too Many Alerts

Alert fatigue sets in when teams are bombarded with notifications. Important alerts get lost in the noise, and engineers learn to ignore or silence warnings. True emergencies blend into background static.

Too Few Alerts

Critical issues go undetected for extended periods, allowing problems to compound. By the time someone notices, customer impact is severe and recovery is costly.

Why This is Hard for ML Systems

- **Feedback is delayed:** Ground truth labels often arrive long after predictions are made, making it difficult to detect model degradation in real-time
- **Data drift is hard to measure:** Statistical tests for distribution shift require careful threshold tuning and may produce false positives from benign variation or false negatives when drift is subtle but meaningful
- **Context matters:** A 5% accuracy drop might be catastrophic for medical diagnosis but acceptable for content recommendations

[1] Shankar, Shreya & Garcia, Rolando & Hellerstein, J. & Parameswaran, Aditya G. (2022). Operationalizing Machine Learning: An Interview Study. ArXiv

Automated Retraining Triggers

Continuous Training (CT) represents the systematic practice of retraining models based on predefined conditions rather than ad-hoc human decisions. Automation ensures consistent response to model degradation while maintaining governance standards.

1. **Performance Deviation Triggers**

   Automated alerts fire when accuracy drops below threshold or significant data drift is detected through statistical tests

2. **New Data Volume Triggers**

   Retraining initiated when operational data volume increases by predefined threshold (typically 5-10% of original training set)

3. **Scheduled Retraining Cycles**

   Fixed periodic retraining (e.g., quarterly or monthly) to systematically incorporate recent environmental changes and business patterns

4. **Compliance-Driven Updates**

   Mandatory retraining required to address newly identified biases, regulatory changes, or policy violations

Critical Requirement: Any retrained model must traverse the same rigorous gated promotion pipeline as the original model. Automation of triggers does not bypass governance controls.

Remember the Human in the Loop

Human oversight is crucial in MLOps, particularly for validating alerts and making model intervention decisions.

Validate Alerts

Verify if detected anomalies or data drift are genuine issues, differentiating them from benign fluctuations or seasonal variations.

Provide Context

Humans interpret model behavior considering real-world events like marketing campaigns or economic shifts that models might misinterpret.

Prevent False Retrains

Human review prevents unnecessary retraining from non-critical data changes, saving resources and maintaining model stability.

Strategic Decision Making

Humans make strategic decisions beyond retraining, such as manual interventions, data cleansing, or adjusting business strategies.

Tool & Framework Suggestions for MLOps

Data & Model Drift Monitoring

- Evidently AI
- Arize
- WhyLabs
- Fiddler

Experiment Tracking & Model Registry

- MLflow
- Weights & Biases
- Neptune

Infrastructure Monitoring

- Prometheus/Grafana
- Datadog

Orchestration & Pipelines

- Apache Airflow
- Prefect
- Kubeflow

End-to-End Platforms

- Domino Data Lab
- SageMaker
- Vertex AI

Meme of the day

https://skaftenicki.github.io/dtu_mlops/s8_monitoring/data_drifting/