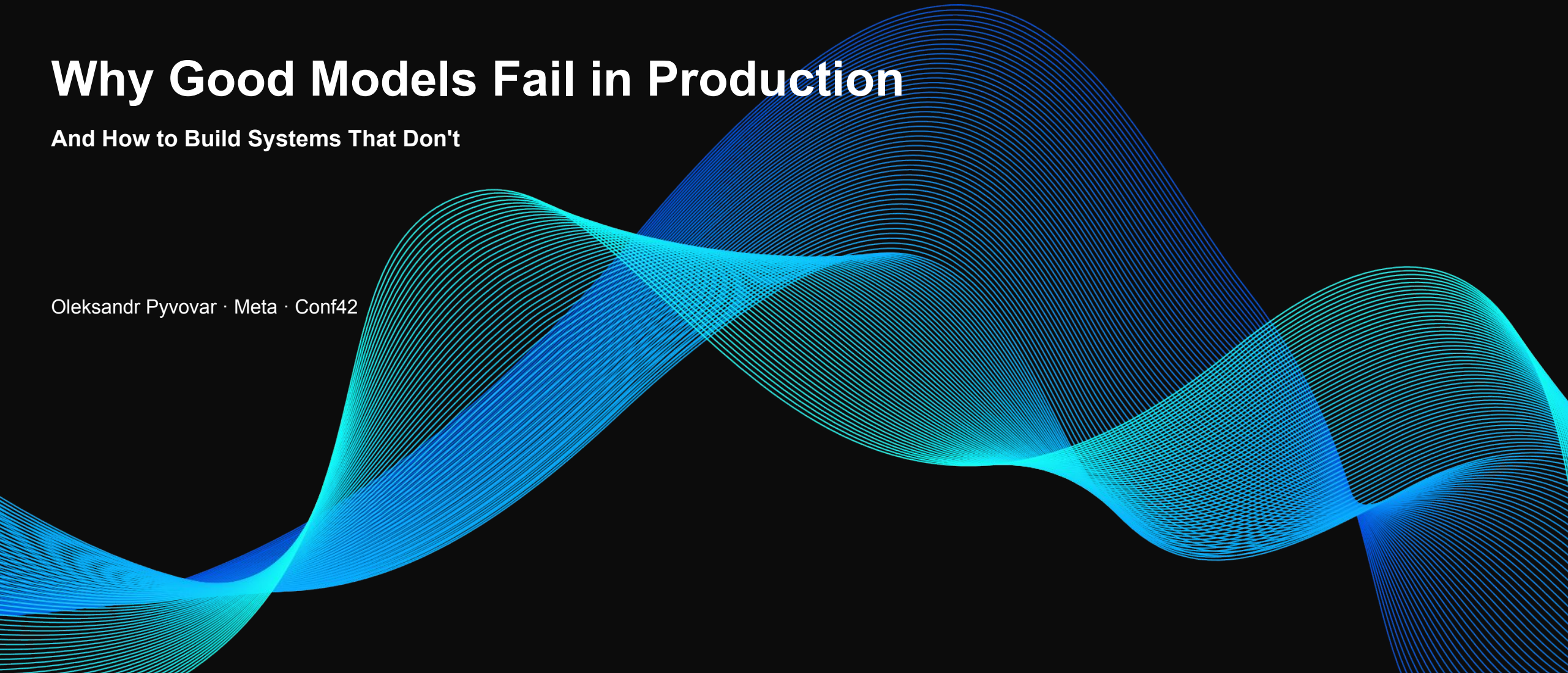
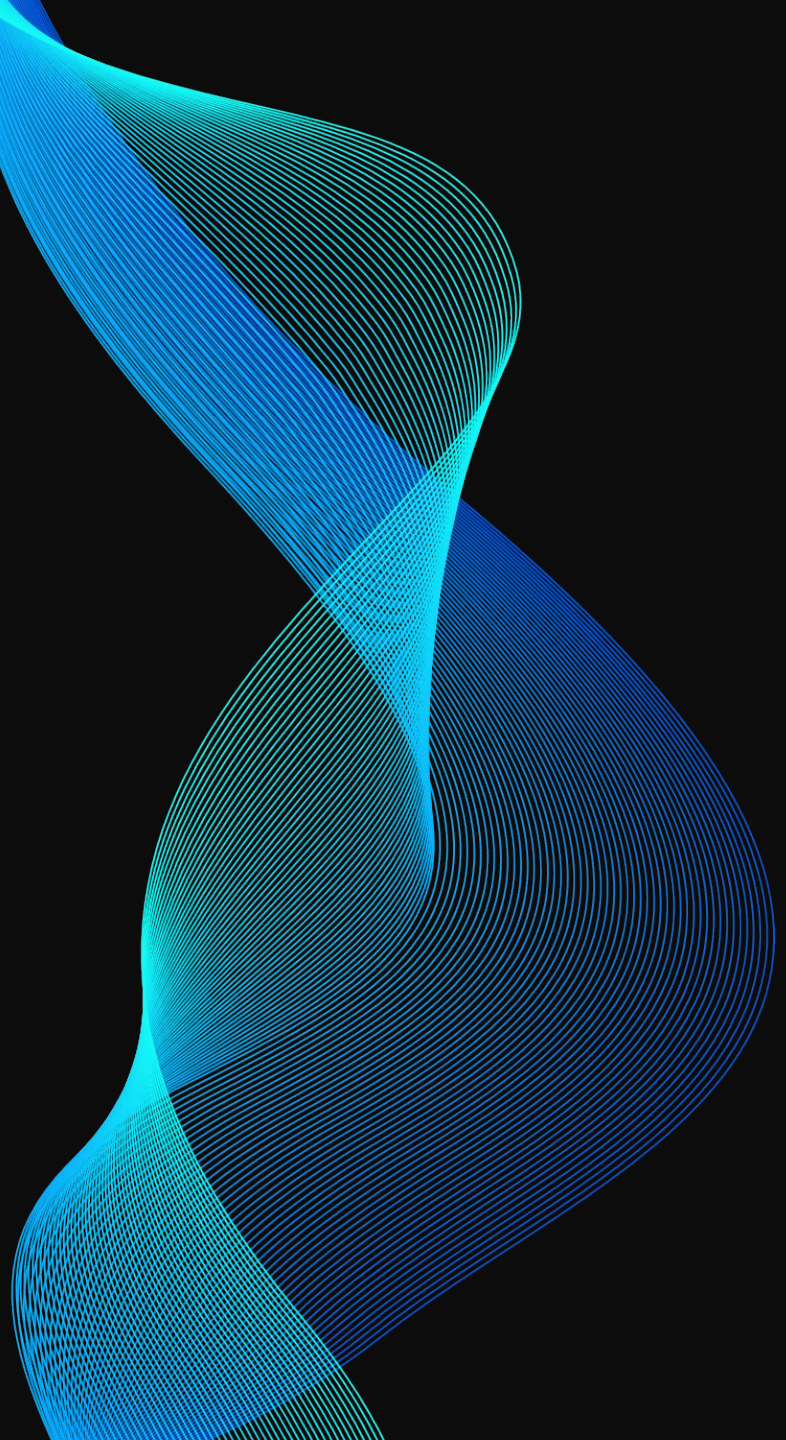
Why Good Models Fail in Production

And How to Build Systems That Don't

Oleksandr Pyvovar · Meta · Conf42





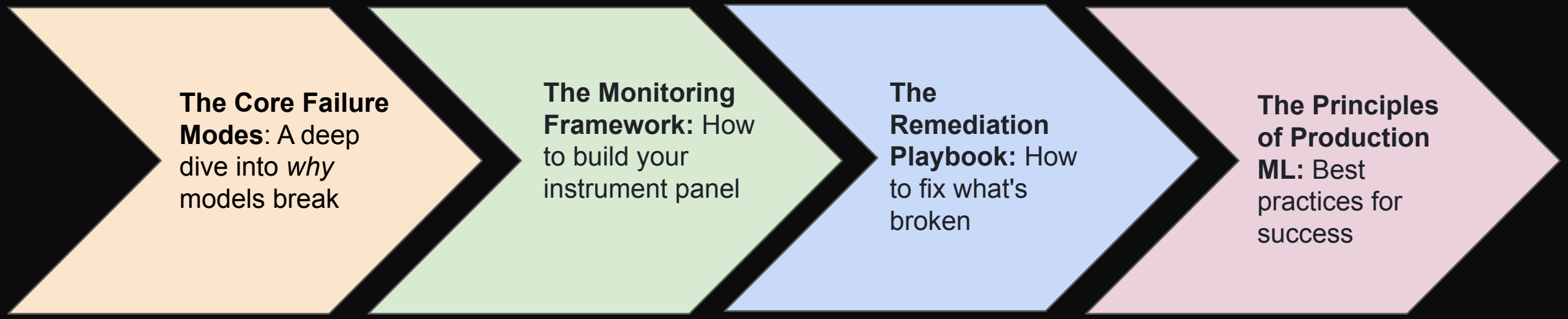
The 95% Accuracy Illusion

Your model achieved 95% accuracy in the lab. You deployed it with confidence. Six months later, it's barely better than random guessing. What happened?

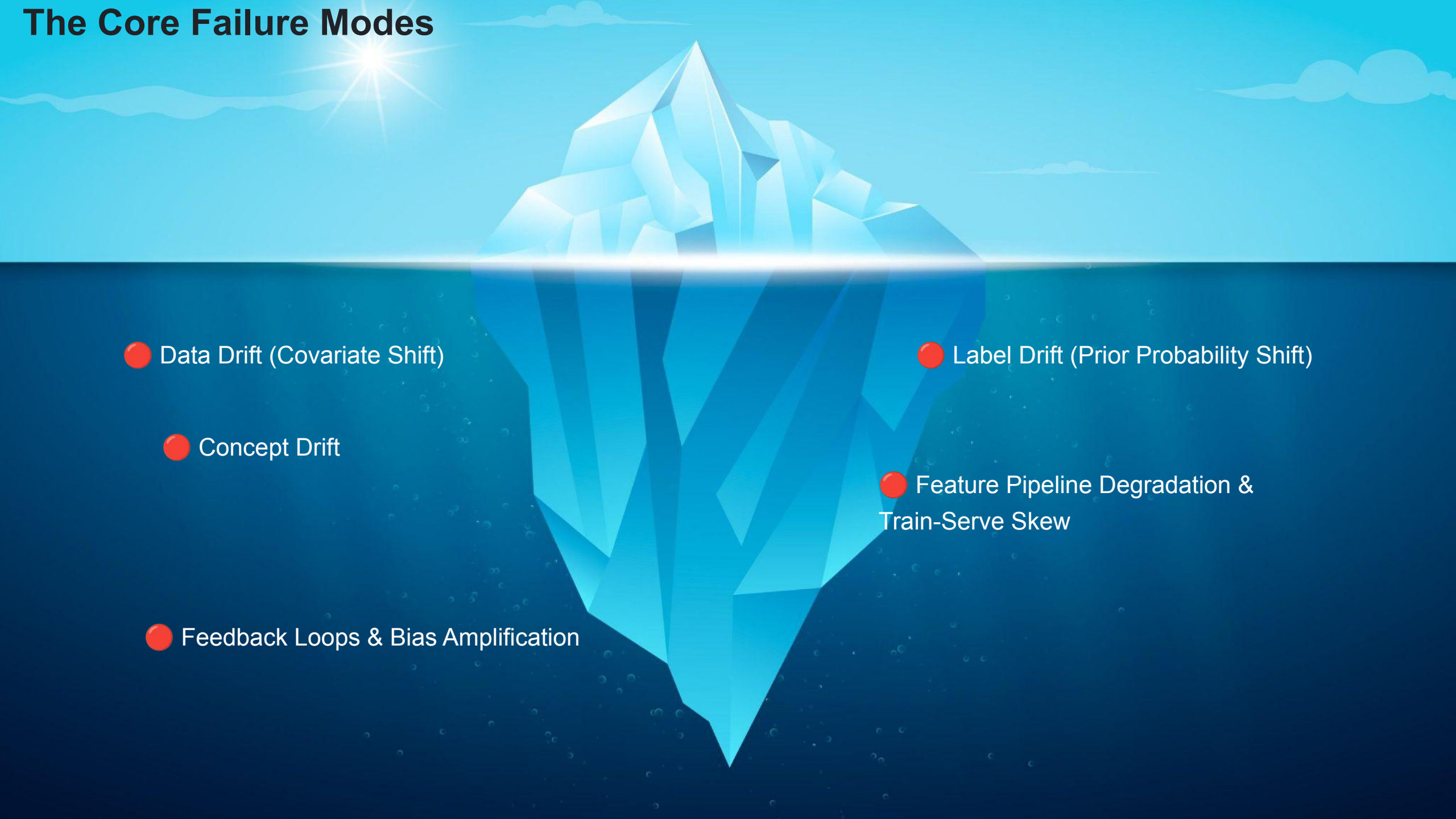
87% of data science projects never make it to production. Of those that do, many fail silently

Our Agenda: A Framework for Resilience

Today's Mission: From Fragile Models to Resilient Systems



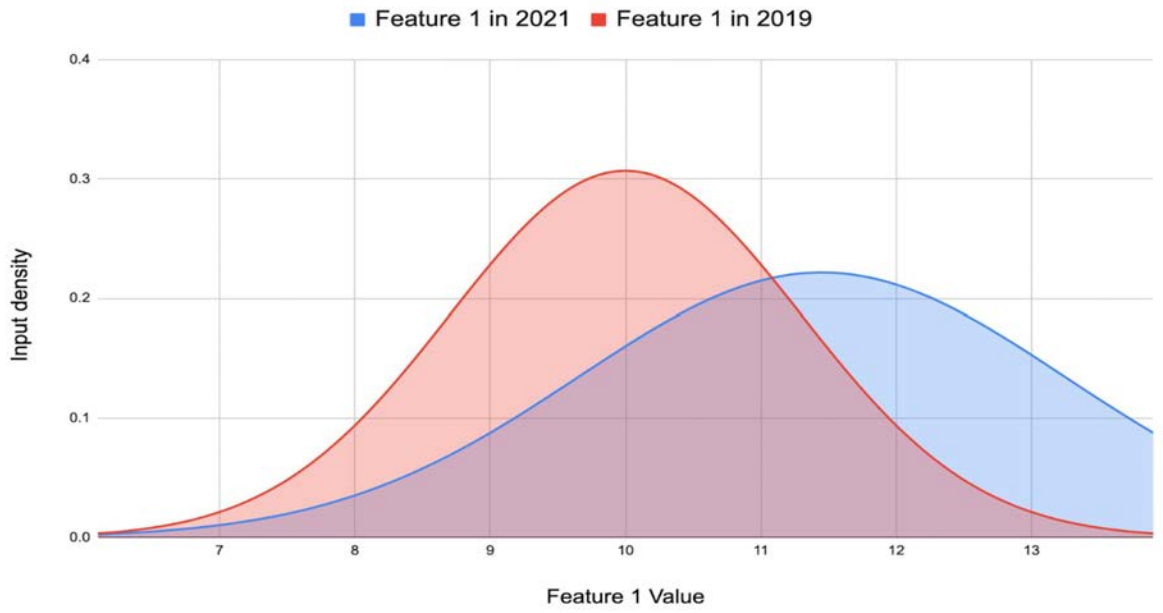
The Core Failure Modes

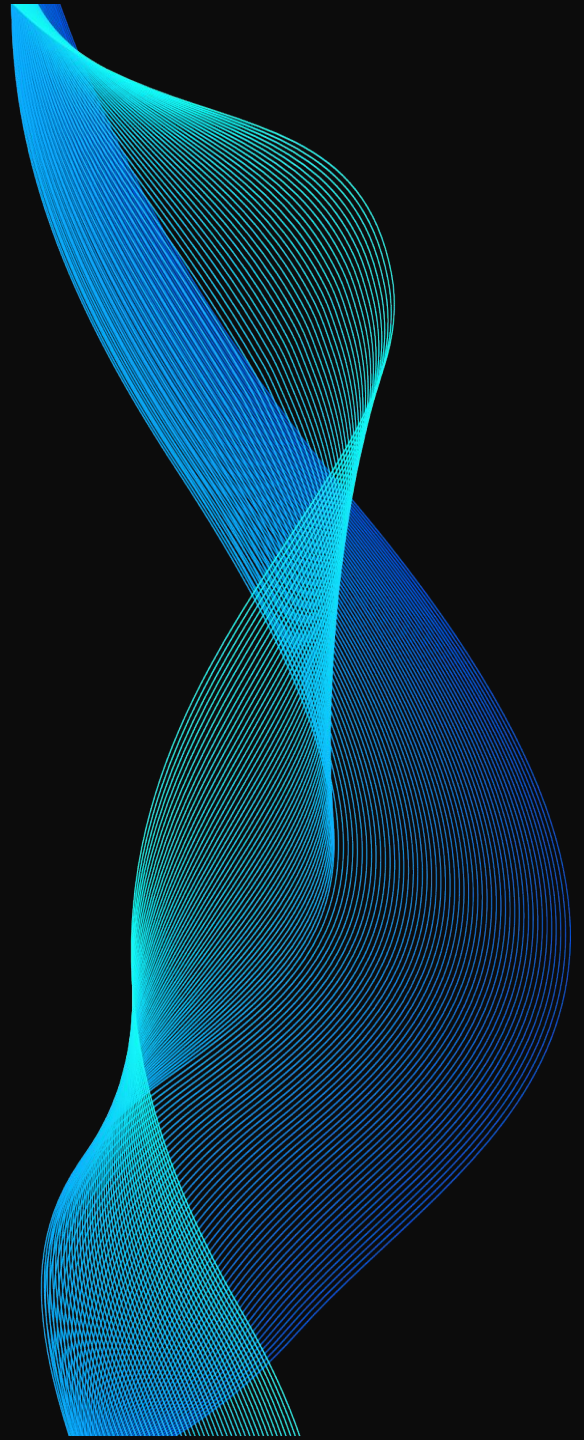


- Data Drift (Covariate Shift)
- Concept Drift
- Feedback Loops & Bias Amplification
- Label Drift (Prior Probability Shift)
- Feature Pipeline Degradation & Train-Serve Skew

Failure Mode 1: Data Drift

The Zillow Buying Failure: In 2021, Zillow's automated home-buying division, Zillow Offers, collapsed, leading to a staggering **$881 million loss** and the **layoff of 25%** of its workforce. While multiple factors were at play, a core issue was the failure of their pricing models to adapt to a rapidly changing housing market. The models, trained on historical data, were unprepared for the unprecedented market volatility and shifting buyer preferences of the post-pandemic era - a classic, and costly, example of data drift.





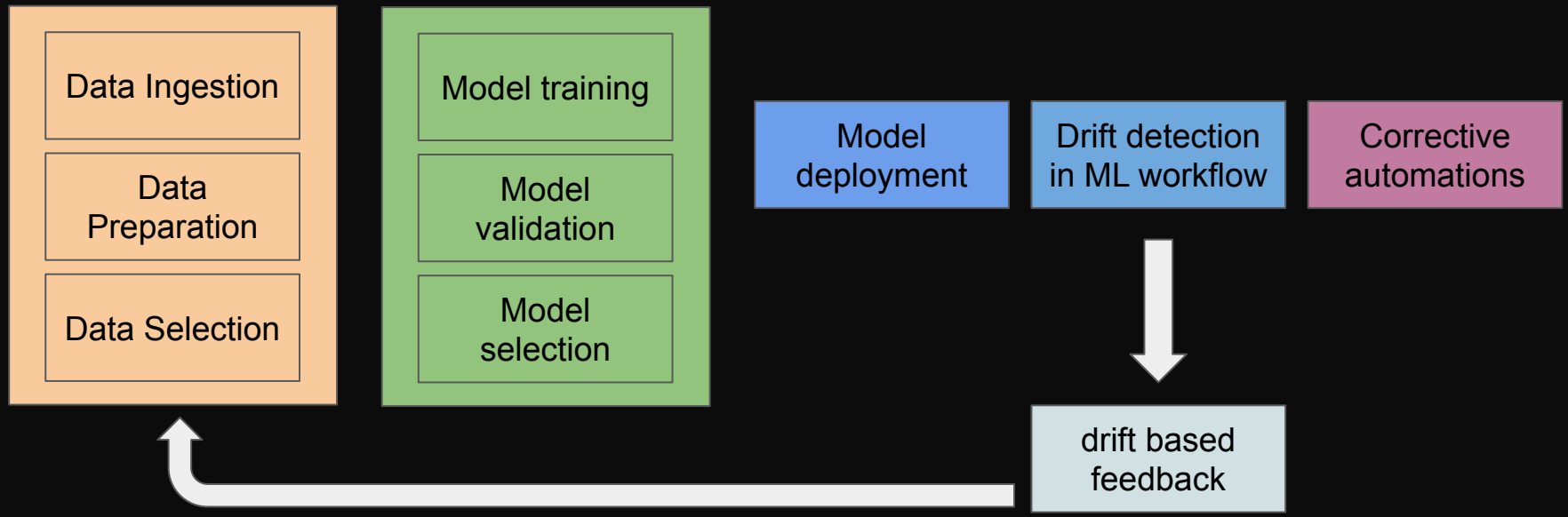
Data Drift: Detection

Data drift occurs when the distribution of the input data in production, *P_prod(X)*, diverges significantly from the distribution of the training data, *P_train(X)*. We can quantify this divergence using statistical tests

Detection Method	Description	Use Case
Population Stability Index (PSI)	Measures the change in the distribution of a single variable. It is calculated by binning the data and comparing the percentage of observations in each bin between the two samples.	Excellent for tracking changes in key features over time. A common rule of thumb is that a PSI > 0.25 indicates significant drift.
Kolmogorov-Smirnov (K-S) Test	A non-parametric test that compares the cumulative distribution functions (CDFs) of two samples. It is sensitive to differences in both location and shape of the distributions.	Best for continuous numerical features. The test returns a p-value; a low p-value (e.g., < 0.05) suggests the distributions are different.
Jensen-Shannon (J-S) Divergence	Measures the similarity between two probability distributions. It is a symmetrized and smoothed version of the Kullback-Leibler (KL) divergence.	Useful for comparing entire probability distributions, especially for categorical features.

Data Drift: Resolution

- Continuous Monitoring & Retraining
- Adaptive Learning
- Domain Adaptation



Failure Mode 2: Concept Drift

The Ever-Evolving World of Fraud: Fraud detection models are in a constant battle against concept drift. A model trained to detect fraudulent credit card transactions might learn that transactions from a certain country late at night are high-risk. Fraudsters adapt. They might start using VPNs to appear as if they are in a low-risk country or switch their attack times. The features of the transaction (amount, location, time) might look the same, but their relationship to the likelihood of fraud has changed because the adversary's strategy has evolved. This is a form of adversarial adaptation, a particularly aggressive type of concept drift.

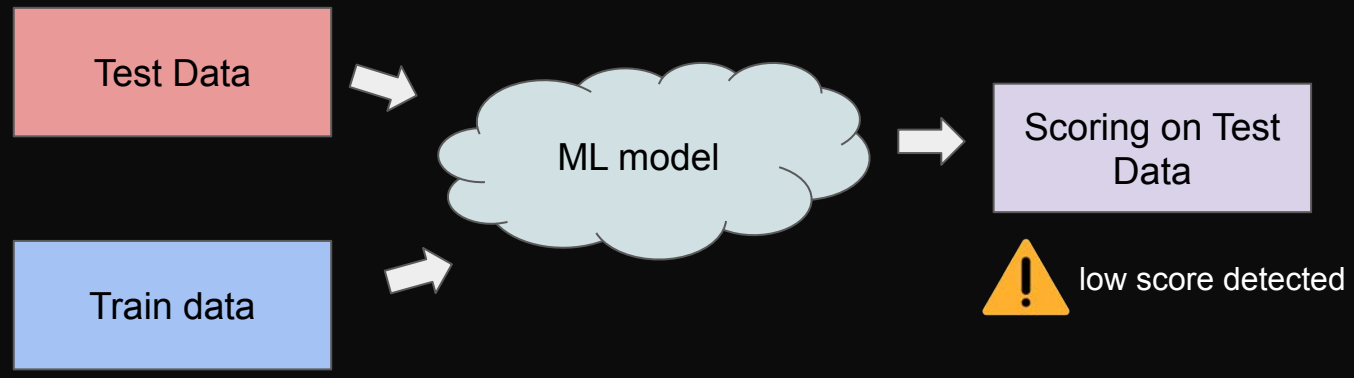
Concept Drift: Detection

Technical Deep Dive & Detection: Concept drift is often more challenging to detect than data drift because it requires ground truth labels (Y) to observe the changing relationship. You can't spot it just by looking at the input data (X). The primary indicator is a degradation in model performance (e.g., accuracy, F1-score) over time, even when the input data distribution remains stable.

Detection Method	Description	Use Case
Performance Monitoring	The most direct method. Continuously track key model metrics (e.g., accuracy, precision, recall, AUC) over time on a labeled validation set. A sustained drop is a strong indicator of concept drift.	Universal. This should be the first line of defense for any production model.
Adaptive Windowing (ADWIN)	An algorithm that maintains a sliding window of recent data. It automatically grows or shrinks the window size when the statistical properties of the data within the window change, signaling a drift.	Effective for detecting both sudden and gradual drifts in streaming data environments.
Drift Detection Method (DDM)	Monitors the model's error rate. It models the error rate as a binomial distribution and triggers a warning when the error rate exceeds a certain threshold, suggesting the underlying concept has changed.	Well-suited for classification tasks where you can track the error rate in near real-time.

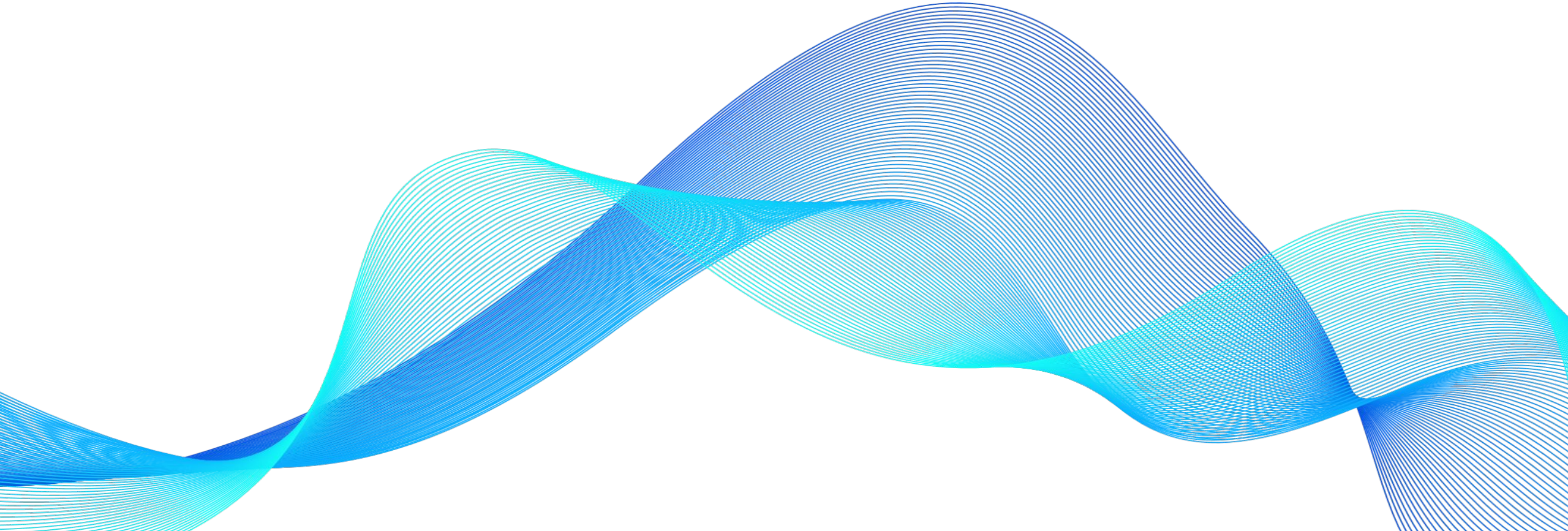
Concept Drift: Resolution

- Online Learning
- Ensemble Methods
- Trigger-based Retraining on Labeled Data



Failure Mode 3: Label Drift

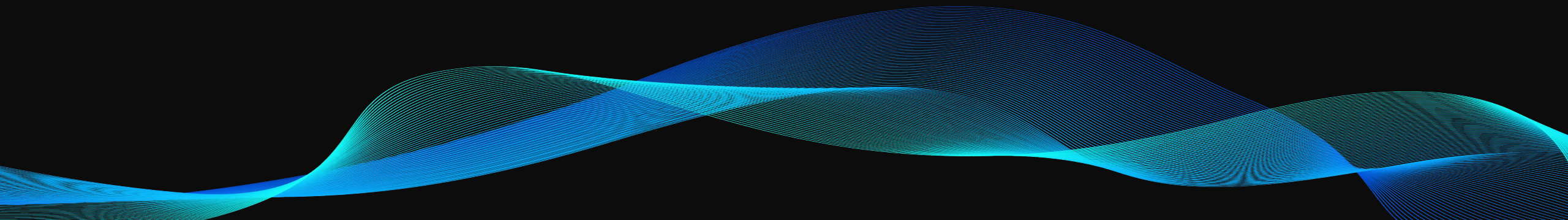
Economic Recessions and Loan Defaults: A model built to predict loan defaults is trained during a period of economic stability, where the default rate is a low and stable 2%. An unexpected recession hits, and unemployment skyrockets. Suddenly, the actual default rate in the population jumps to 10%. The model, calibrated on the 2% rate, will now severely underestimate the risk across the board, potentially leading to massive financial losses for the lending institution. Its internal assumptions about the base rate of default are no longer valid.

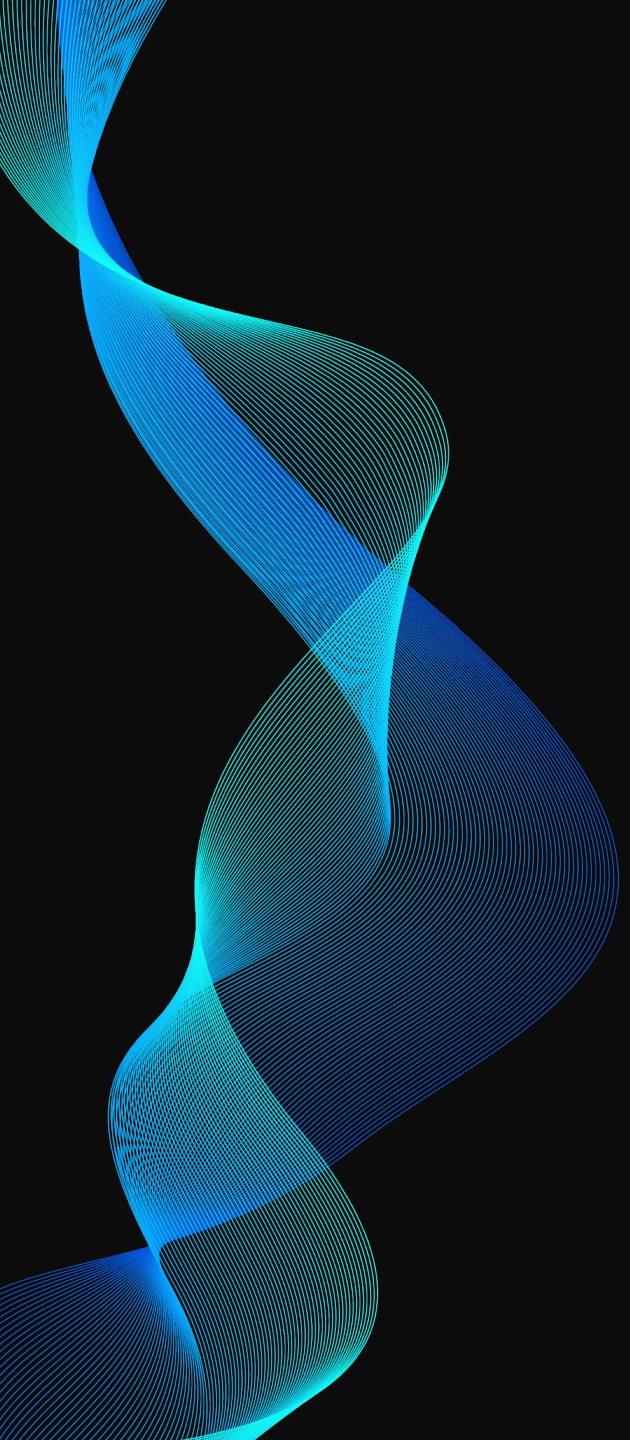


Label Drift: Detection

Technical Deep Dive & Detection: Label drift is detected by monitoring the distribution of the target variable itself. Unlike concept drift, you don't necessarily need to evaluate the entire model's performance to spot it, but you do need access to ground truth labels over time.

Detection Method	Description	Use Case
Monitor Class Distribution	The most straightforward method. Track the proportion of each class in your labeled data over time. A significant change in these proportions is a direct indication of label drift.	Essential for any classification task. Can be visualized with a simple time series plot of class percentages.
Chi-Square Test	A statistical test that can be used to determine if there is a significant association between two categorical variables. In this case, you can compare the distribution of labels in a reference window to the distribution in a current window.	Good for formally testing if the change in class distribution is statistically significant.
Track Prediction Distribution	In the absence of immediate ground truth, you can monitor the distribution of your model's predictions. If a model that used to predict 5% positive cases suddenly starts predicting 15%, it could be a (noisy) indicator of label drift.	Useful as an early warning system when ground truth labels are delayed. However, it can be confounded by data drift, so it should be used with caution.



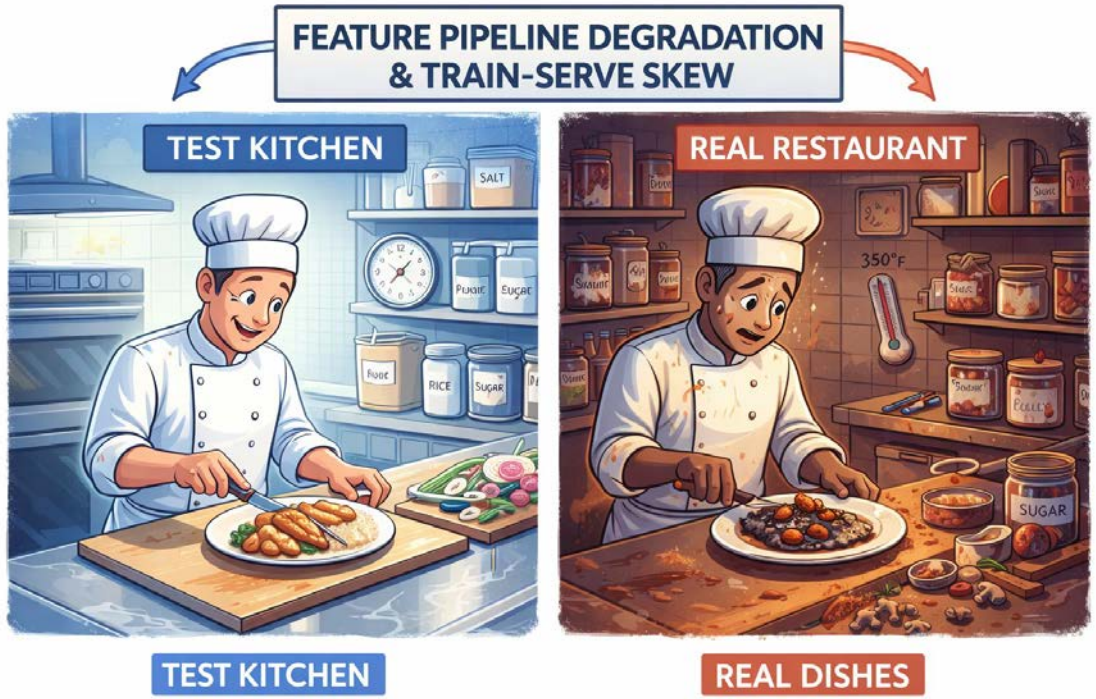


Label Drift: Resolution

- Recalibration: The model's outputs can often be adjusted post-hoc to account for the new class distribution. Techniques like Platt Scaling or Isotonic Regression can be used to recalibrate the model's probability scores to better match the new reality.
- Importance Weighting: During retraining, you can assign weights to the training samples to make the training data distribution resemble the production data distribution. If the proportion of class 'A' has doubled in production, you can give a higher weight to samples of class 'A' during training.
- Cost-Sensitive Learning: If the cost of misclassifying different classes is different (e.g., failing to detect a fraudulent transaction is much worse than flagging a legitimate one), you can adjust the learning algorithm to penalize the more costly errors more heavily. This can make the model more robust to changes in class distribution.

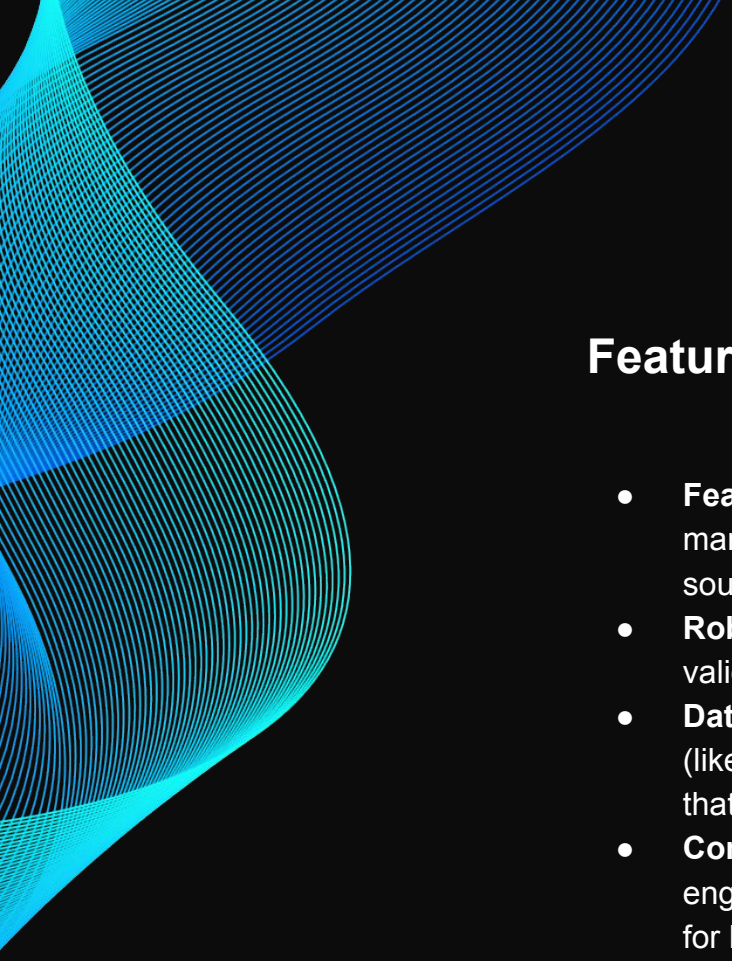
Failure Mode 4: Feature Pipeline Degradation & Train-Serve Skew

Google's Diabetic Retinopathy AI in the Real World: Google developed a state-of-the-art deep learning model that could detect diabetic retinopathy from retinal scans with accuracy on par with human ophthalmologists. In the lab, using high-quality, perfectly captured images, it was a resounding success. However, when deployed in real-world clinics in Thailand, its performance stumbled. The nurses, working in busy, poorly-lit conditions, often captured images that were of lower quality than the pristine training data. The system had a high rejection rate for these real-world images, and intermittent internet connectivity hampered the cloud-based analysis. This is a classic case of train-serve skew: the features (retinal images) available at prediction time were systematically different from the features used to train the model, not because of a change in the patient's eyes, but because of the data capture process itself.



Feature Pipeline Degradation: Detection

Issue Type	Description	Detection Method
Data Quality Issues	The data itself is corrupted. This includes a sudden increase in NULL values, the appearance of outliers, or incorrect data types (e.g., a string appearing in a numeric column).	Data Validation Frameworks like Great Expectations or Deequ. These tools allow you to define a suite of expectations for your data (e.g., **`expect_column_values_to_not_be_null`**, **`expect_column_mean_to_be_between`**). These checks should run every time new data is ingested.
Schema Changes	An upstream data source changes its structure. A column is renamed, removed, or its data type is changed. This will often cause the feature pipeline to crash entirely.	Schema Monitoring. Your pipeline should automatically ingest and validate the schema of incoming data against a known-good reference schema. Alerts should be triggered on any discrepancy.
Train-Serve Skew	The logic used to create features for training is different from the logic used in production. This can be subtle, such as using a library with a different version that has a minor implementation change, or it can be blatant, like using data that would not actually be available at prediction time.	Prediction Validation & Shadow Deployment. Log the inputs and outputs of both the training and production environments for a sample of data. If `f_train(X)` and `f_prod(X)` produce different results for the same input `X`, you have a skew. Shadowing (running the new model alongside the old one without acting on its predictions) is a powerful way to detect this before full deployment.

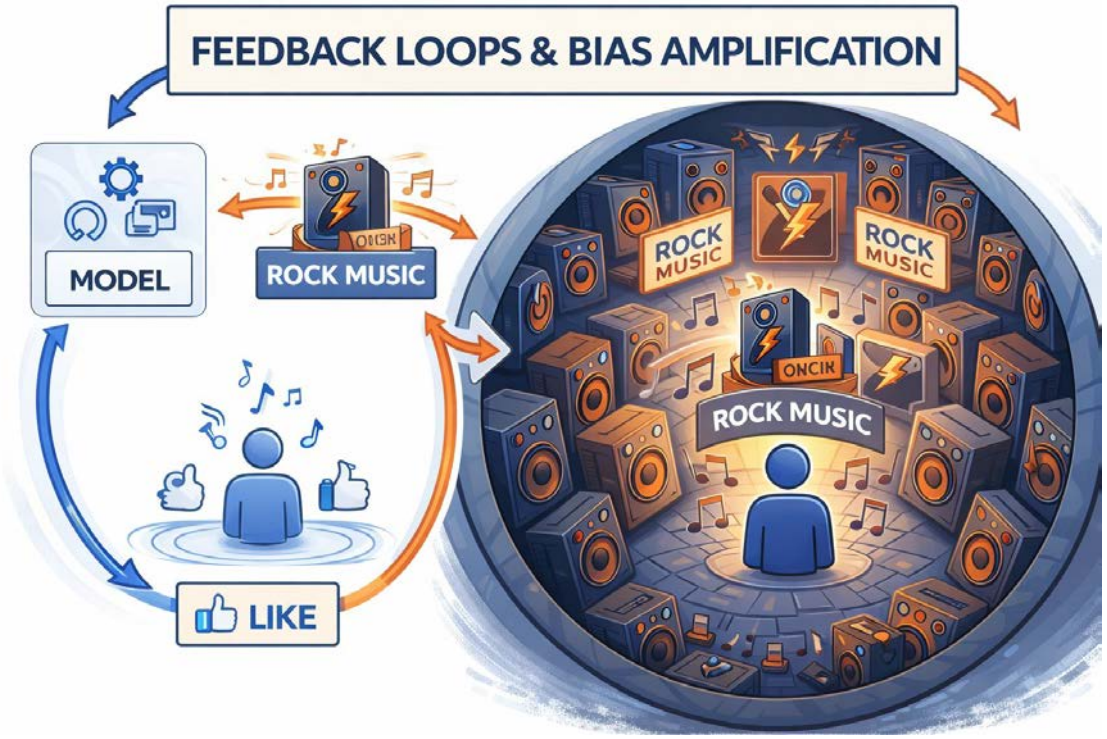


Feature Pipeline Degradation: Resolution

- **Feature Store:** This is the gold standard for solving train-serve skew. A feature store is a centralized system that manages the definition, computation, and serving of features for both training and inference. By using the same source and transformation logic for both environments, it programmatically eliminates this class of problems.
- **Robust Data Validation:** Embed data validation checks directly into your ML pipelines. If incoming data fails validation, the pipeline should halt and send an alert. This prevents corrupted data from ever reaching your model.
- **Data Contracts:** Establish formal agreements between the teams that produce data and the teams that consume it (like the ML team). This contract defines the schema, data types, quality standards, and update cadence, ensuring that any changes are communicated and managed, rather than appearing as a surprise that breaks production.
- **Containerization:** Package your model, its dependencies (including specific library versions), and the feature engineering code into a single container (e.g., using Docker). This ensures that the exact same environment is used for both training and serving, mitigating environment-based discrepancies.

Failure Mode 5: Feedback Loops & Bias Amplification

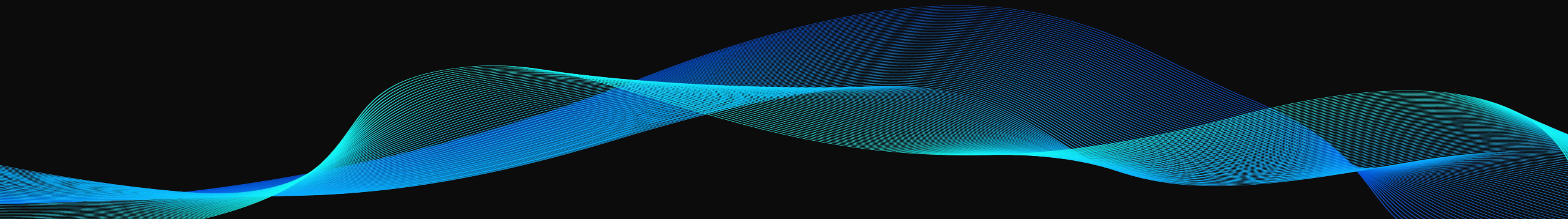
The COMPAS Recidivism Algorithm: The COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) tool has been used across the U.S. justice system to predict the likelihood of a defendant re-offending. A landmark 2016 investigation by ProPublica revealed that the algorithm was significantly biased against Black defendants, flagging them as high-risk for future crimes at nearly twice the rate as white defendants with similar histories. This creates a pernicious feedback loop. A defendant flagged as high-risk is more likely to receive a longer sentence, be denied parole, and face heavier policing, all of which can increase the likelihood of a future arrest, regardless of their individual actions. The model's prediction becomes a self-fulfilling prophecy, and this biased outcome is then recorded as a "correct" data point, further training the model to perpetuate the same bias.



Feedback Loops & Bias Amplification: Detection

Technical Deep Dive & Detection: Feedback loops occur when the model's output, $\hat{y}_t$, influences the state of the world and thus the future data, X_{t+1}, that the model will be trained on. This can lead to a gradual narrowing of the data distribution, $E[X_t]$, towards a biased subset over time, amplifying any initial biases in the model or data.

Detection Method	Description	Use Case
Diversity & Exposure Metrics	Quantify the variety of predictions the model is making. For a recommendation system, this could be the number of unique items recommended over a period. For a loan application model, it could be the demographic distribution of approved applicants. A sharp decrease in diversity is a red flag.	Crucial for recommendation systems, content feeds, and any system that personalizes user experience.
A/B Testing with a Control Group	Run a small percentage of users through a "random" or non-personalized version of the model. Compare the behavior and outcomes of this control group to the main group. If the model-driven group's behavior diverges significantly over time, a strong feedback loop is likely present.	The gold standard for identifying the causal impact of a model on user behavior.
Causal Inference Techniques	Employ more advanced statistical methods to try and disentangle the model's influence from the user's organic preferences. Techniques like uplift modeling can help estimate the true persuasive effect of a prediction.	Useful when a full A/B test is not feasible, but requires more sophisticated data analysis.



Feedback Loops & Bias Amplification: Resolution

- **Exploration Strategies:** Intentionally introduce randomness into the model's predictions to break the feedback loop. In a recommendation system, this could mean occasionally showing items that the model is uncertain about or that are outside the user's typical interest profile (an ε-greedy approach). This allows the model to collect data on a wider range of possibilities.
- **Positional Debiasing:** In ranked lists (like search results or recommendations), users are more likely to click on items at the top. The model can misinterpret this as a stronger preference. Positional debiasing techniques adjust for this by explicitly modeling the probability of a click given its position in the list.
- **Fairness Constraints:** During model training, you can introduce mathematical constraints to ensure that the model's predictions satisfy certain fairness criteria. For example, you could require that the rate of loan approvals be statistically similar across different demographic groups, a concept known as "demographic parity."

A Unified Monitoring Framework

The Three Pillars of Monitoring: A comprehensive framework rests on three pillars, each tracking a different aspect of your model's operational reality.

Pillar	What It Tracks	Why It Matters	Key Metrics
1. Model Performance	The ultimate business value and predictive power of your model.	This is the ground truth of your model's effectiveness. A decline here is a direct threat to the ROI of the project.	Accuracy, Precision, Recall, F1-Score, AUC-ROC, Log-Loss, Business-specific KPIs (e.g., revenue per recommendation, click-through rate).
2. Data Health	The statistical integrity of the data being fed to your model.	This is your early warning system. Data issues are the leading cause of performance degradation, and detecting them here allows you to react *before* business metrics are impacted.	PSI, K-S Statistic, Null Rates, Data Type & Schema Compliance, Outlier Rates, Feature Min/Max/Mean.
3. System Health	The operational performance of the infrastructure serving the model.	A model that gives perfect predictions too slowly is useless. System health ensures your model is meeting its Service Level Agreements (SLAs).	Inference Latency (p50, p95, p99), Throughput (predictions per second), Error Rates (e.g., HTTP 500s), CPU/Memory/GPU Utilization.

Tooling Landscape

The MLOps landscape has matured, and you don't need to build these monitoring systems from scratch. A rich ecosystem of open-source and commercial tools can provide this functionality out-of-the-box.

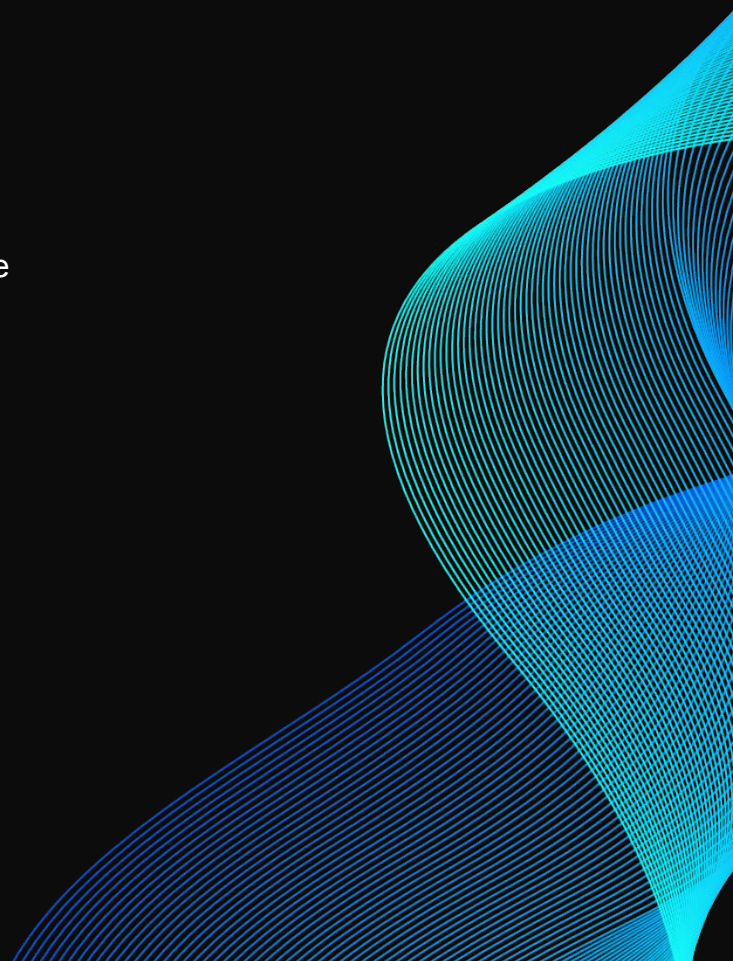
Open Source

Tools like Evidently AI, NannyML, and Alibi Detect provide powerful, flexible frameworks for detecting data drift, concept drift, and performance degradation.

They can generate rich, interactive dashboards and can be integrated into automated workflows.

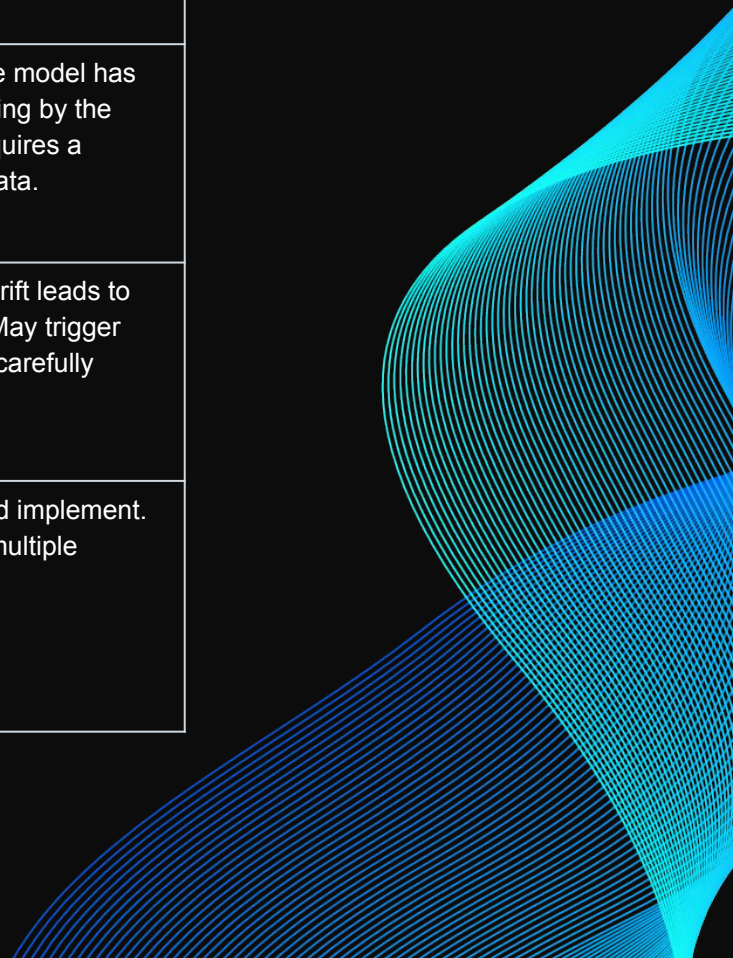
Commercial Platforms

Companies like WhyLabs, Fiddler AI, and Arize AI offer end-to-end ML observability platforms. They often provide more advanced features, enterprise-grade support, and seamless integration with the broader ML ecosystem.



Intelligent Retraining Strategies

Strategy	Description	Pros	Cons
1. Time-based (Scheduled)	Retrain the model on a fixed schedule (e.g., daily, weekly, monthly), regardless of performance.	Simple to implement and predictable. Ensures the model is never too stale.	Highly inefficient. May retrain unnecessarily when no drift has occurred, or fail to react quickly to a sudden drift.
2. Performance-based (Triggered)	Retrain only when a key performance metric (e.g., accuracy, F1-score) drops below a predefined threshold.	Directly tied to business value. Only retrains when there is a demonstrated impact on performance.	Reactive, not proactive. The model has already been underperforming by the time the trigger is fired. Requires a reliable stream of labeled data.
3. Drift-based (Triggered)	Retrain when a data drift metric (e.g., PSI, K-S statistic) for key features or the model's predictions exceeds a threshold.	Proactive. Can trigger a retrain *before* model performance is significantly impacted.	Can be noisy. Not all data drift leads to performance degradation. May trigger unnecessary retrains if not carefully calibrated.
4. Hybrid Approach (Recommended)	Combine multiple triggers into a more sophisticated policy. This is the most robust and practical approach for most production systems.	Balances the pros and cons of the other methods, leading to a more efficient and resilient system.	More complex to design and implement. Requires careful tuning of multiple thresholds.



The Best Practices: Principles of Production ML

Design for Drift from Day 1 - Build monitoring and retraining into your architecture upfront, not as an afterthought.

Implement Gradual Rollouts → Shadow → Canary → Full. Never deploy directly to 100% traffic.

Document Everything - Model Cards, Data Lineage, Decision Logs. Future you will thank you.

Maintain Training-Serving Parity - Use identical code, libraries, and data sources for both training and inference. (Feature stores are the gold standard.)

Instrument for Feedback - Collect ground truth on predictions. This fuels both monitoring and retraining.

FINAL ACTIONABLE ADVICE