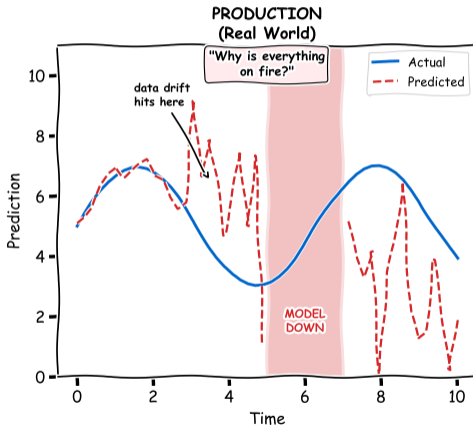
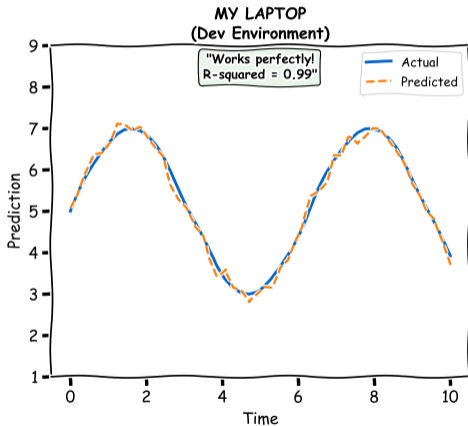


Lesson 5.4: MLOps and Production ML in Finance

Module 5: Automation and Infrastructure

Digital Finance

The MLOps Reality Gap



Building a model is 10% of the work. Running it reliably in production is the other 90%.

By the end of this lesson, you will be able to:

- 1 **Map** the complete ML lifecycle from data ingestion to model retirement
- 2 **Distinguish** concept drift, data drift, and covariate shift with financial examples
- 3 **Design** a model monitoring dashboard with appropriate metrics and alerts
- 4 **Explain** SR 11-7 model risk management requirements for regulated institutions
- 5 **Evaluate** champion-challenger testing frameworks for model replacement decisions

Core theme: Models are not “fire and forget.” They are living systems that require continuous care.

These objectives span technical (drift detection), operational (monitoring), and regulatory (governance) dimensions.

Where we are: We understand what models can and cannot predict.

The new question: How do we run them reliably in production?

What Data Scientists Build

- Jupyter notebook with 95% accuracy
- Feature engineering in pandas
- Model saved as pickle file
- "It works on my machine!"

What Production Requires

- Automated pipeline, 99.9% uptime
- Feature store with point-in-time correctness
- Versioned, auditable model registry
- Drift detection, alerting, governance

The gap between these two worlds is what MLOps solves.

A 2022 Gartner survey found that only 54% of ML models make it from pilot to production.

What is MLOps?

Definition: Machine Learning Operations (MLOps) applies DevOps principles—automation, monitoring, versioning, and governance—to the machine learning lifecycle.

Traditional DevOps

- Code version control (Git)
- Continuous Integration (CI)
- Continuous Deployment (CD)
- Automated testing
- Infrastructure monitoring

MLOps Adds

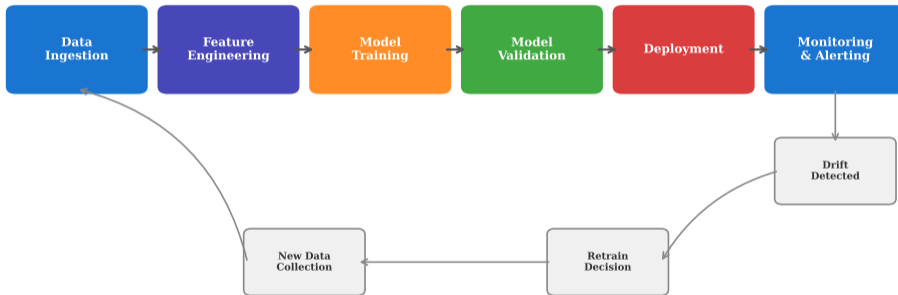
- **Data versioning** (DVC, Delta Lake)
- **Experiment tracking** (MLflow)
- **Feature stores** (Feast, Tecton)
- **Model registry** (staging, production)
- **Drift detection & retraining**

Why it matters in finance: In regulated environments, every model decision must be traceable, reproducible, and auditable. MLOps provides the infrastructure to achieve this.

MLOps bridges the gap between experimental notebooks and enterprise-grade production systems.

End-to-End ML Lifecycle in Finance

Forward Pipeline



Feedback Loop (Continuous Retraining)

- **What you see:** Seven stages in a continuous loop: data ingestion → feature engineering → training → validation → deployment

Data Ingestion

- ETL from databases, APIs, streaming
- Schema validation (types, ranges)
- Data quality checks (missing, outliers)
- Data versioning (snapshots via DVC)

Financial example:

- Transaction data: validate amounts, timestamps
- Market data: check for gaps, split adjustments
- Customer data: PII compliance checks

What is a Feature Store?

A centralized repository that serves features consistently to both training and serving.

Key capabilities:

- Online (low-latency) + offline (batch) storage
- Consistent feature definitions across teams
- Point-in-time correctness for temporal features
- Feature versioning and lineage tracking

Why it matters: Eliminates *training-serving skew*—the silent killer of ML systems.

Training-serving skew occurs when features are computed differently in training vs. production, causing silent accuracy loss.

Reproducibility formula:

$$\text{Model}_v = f(\text{Code}_v, \text{Data}_v, \text{Params}_v, \text{Env}_v)$$

What to log for every experiment:

Inputs

- Git commit SHA
- Data version hash
- Hyperparameters
- Random seeds

Outputs

- Performance metrics
- Confusion matrix
- Feature importance
- Model artifact

Context

- Author / team
- Training duration
- Compute cost
- Environment (Docker image)

Financial regulatory requirement: Every production model must be fully reproducible for audit. SR 11-7 mandates “effective challenge” of models, which requires complete lineage.

Tools: MLflow Tracking, Weights & Biases, Neptune.ai, Comet ML.

What is Model Validation? Independent assessment that a model is fit for its intended purpose.

Validation checks before deployment:

- **Performance:** AUC-ROC, precision, recall, F1 vs. baseline
- **Fairness:** Demographic parity, equalized odds across protected groups
- **Robustness:** Performance under adversarial inputs and edge cases
- **Explainability:** SHAP values, feature importance, model cards
- **Business metrics:** Expected revenue impact, operational cost

Financial-specific validation:

- **Backtesting:** Simulate historical performance (required for VaR models)
- **Stress testing:** Performance during market crises (2008, COVID-19)
- **Sensitivity analysis:** How much do outputs change with small input perturbations?
- **Benchmark comparison:** Must outperform current champion model

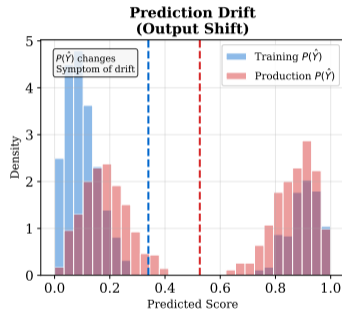
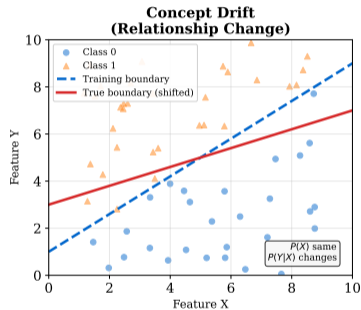
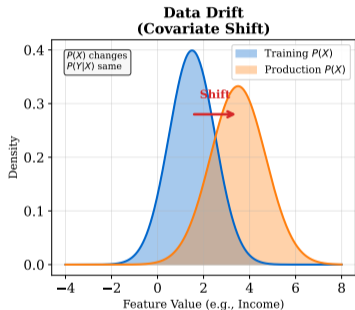
Automated gates: Only promote to production if all thresholds pass.

Validation is a regulatory requirement under SR 11-7, not an optional best practice.

What is Model Drift?

Definition: Progressive degradation of model performance due to changes in data or relationships.

Three Types of Model Drift



- **What you see:** Three drift types—data drift ($P(X)$ changes), concept drift ($P(Y|X)$ changes), prediction drift ($P(\hat{Y})$ changes)
- **Key pattern:** Data drift is common and detectable (PSI); concept drift is rare but catastrophic (model assumptions break)
- **Takeaway:** Monitor all three dimensions—concept drift requires retraining; data drift may only need feature recalibration

All three drift types can occur simultaneously. Concept drift is the most dangerous because the model becomes fundamentally wrong.

What is Data Drift? The distribution of input features $P(X)$ changes, while the relationship $P(Y|X)$ stays constant.

Financial examples:

- Average income rises from \$50k to \$65k after inflation
- Transaction amounts shift toward mobile payments
- Customer age distribution skews younger
- Loan sizes increase due to housing prices

Why models struggle:

- Predictions in unfamiliar input regions
- Feature scaling assumptions break
- Decision boundaries may be wrong for new data ranges

Detection methods:

- **Population Stability Index (PSI)**
- **Kolmogorov-Smirnov (KS) test**
- **Chi-square test** (categorical features)
- **Wasserstein distance**
- **Jensen-Shannon divergence**

Response:

- Monitor performance closely
- Retrain if accuracy degrades
- Update feature preprocessing
- Expand training data range

Data drift is the most common drift type and the easiest to detect automatically.

What is Concept Drift? The relationship between inputs and target $P(Y|X)$ changes, even if $P(X)$ stays constant.

Financial examples:

- COVID-19 changes default patterns (same income, different risk)
- New fraud techniques emerge (same transaction patterns, different labels)
- Regulatory change alters risk definitions
- Market regime shift (same indicators, opposite signal)

Four subtypes:

- **Sudden:** Regulatory shock, market crash
- **Gradual:** Customer preferences evolve
- **Incremental:** Step-by-step changes
- **Recurring:** Seasonal patterns (holidays)

Detection (harder than data drift):

- Track online accuracy / AUC over time
- Sliding window evaluation
- Statistical tests: ADWIN, DDM, EDDM
- Compare error rates to baseline

Response:

- **Retrain** with recent labeled data
- **New features** to capture new patterns
- **Ensemble** old + new models
- **Online learning** for fast adaptation
- In extreme cases: **rebuild** the model

Concept drift is the most dangerous type—the model's fundamental assumptions become wrong.

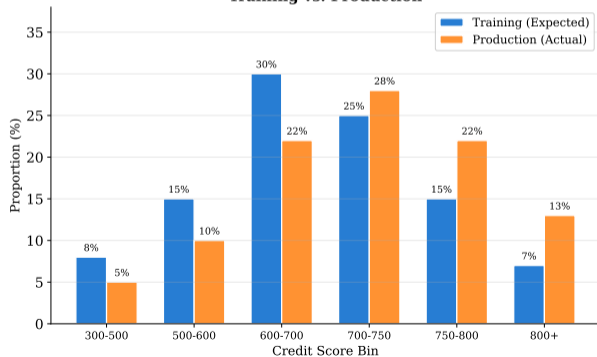
Population Stability Index (PSI)

Formula:

$$PSI = \sum_{i=1}^n (P_{\text{actual},i} - P_{\text{expected},i}) \times \ln\left(\frac{P_{\text{actual},i}}{P_{\text{expected},i}}\right)$$

Population Stability Index (PSI) -- Credit Score Example

Credit Score Distribution:
Training vs. Production

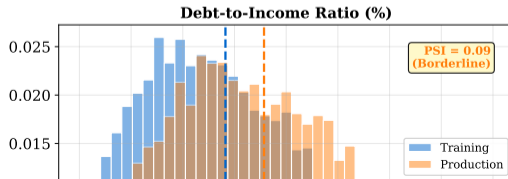
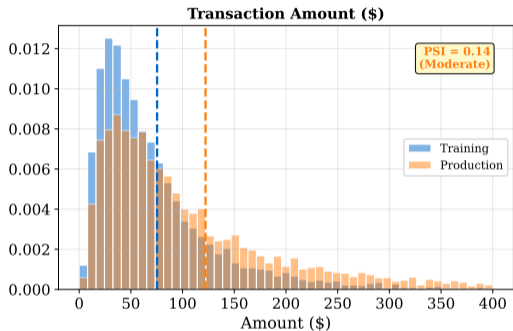


PSI Calculation

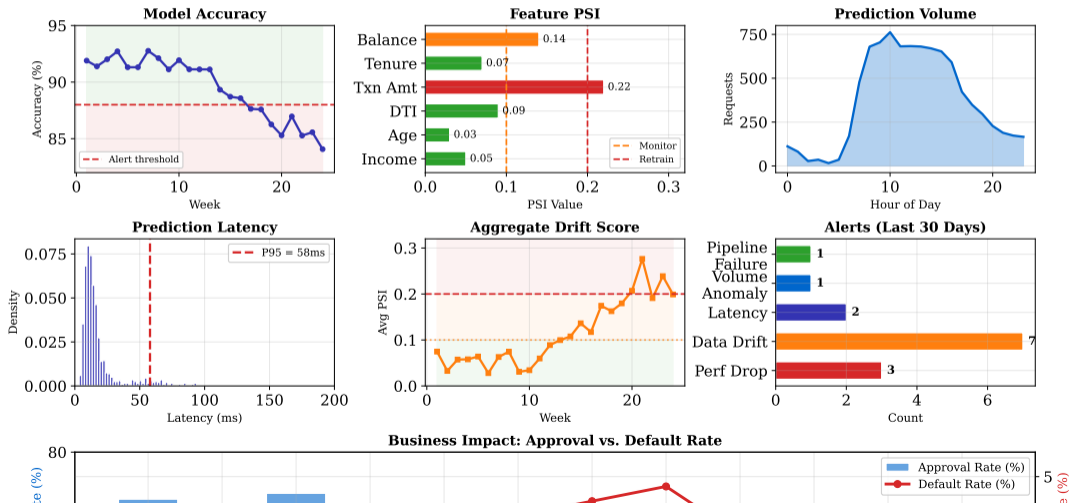
Bin	Exp.	Act.	Diff	ln(A/E)	PSI
300-500	0.08	0.05	-0.03	-0.470	0.0141
500-600	0.15	0.10	-0.05	-0.405	0.0203
600-700	0.30	0.22	-0.08	-0.310	0.0248
700-750	0.25	0.28	+0.03	+0.113	0.0034
750-800	0.15	0.22	+0.07	+0.383	0.0268
800+	0.07	0.13	+0.06	+0.619	0.0371
Total					0.1265

VERDICT: Moderate drift -- monitor closely

Feature Distribution Monitoring: Training vs. Production



Production ML Monitoring Dashboard -- Credit Scoring Model



What to Monitor: Six Metric Categories

1. Model Performance

- Accuracy, AUC-ROC, F1-score
- Compare to baseline SLA

2. Data Drift

- PSI per feature (weekly)
- KS test, Jensen-Shannon divergence

3. Prediction Drift

- Mean, std, percentiles of \hat{Y}
- Prediction rate shift (classifiers)

4. Data Quality

- Missing value rate, schema violations
- Outlier frequency

5. Operational Health

- Latency (P50, P95, P99)
- Error rate, throughput

6. Business Impact

- Approval rate, default rate
- False positive cost
- Revenue per prediction

Alert tiers: CRITICAL (PagerDuty) → WARNING (Slack) → INFO (weekly digest).

Set alert thresholds carefully—alert fatigue is the number-one reason monitoring fails.

What is SR 11-7?

Definition: Federal Reserve guidance (2011) on model risk management for US banking institutions.

Three pillars:

- 1 **Model Development:** Sound theory, appropriate methodology, rigorous testing
- 2 **Model Validation:** Independent review by qualified personnel not involved in development
- 3 **Model Governance:** Board oversight, model inventory, ongoing monitoring, change management

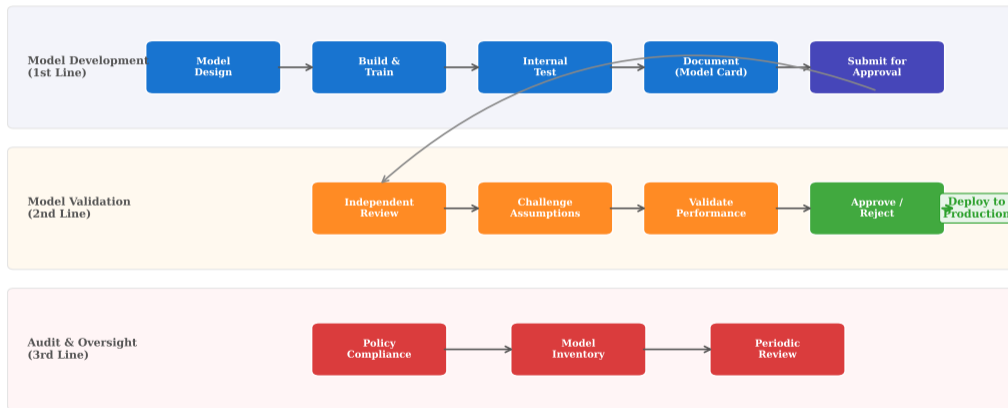
Key requirements for ML models:

- **Model inventory:** Centralized catalog of all models with risk tiers
- **Documentation:** Model cards describing purpose, limitations, performance
- **Independent validation:** Second-line review before production deployment
- **Ongoing monitoring:** Continuous performance and drift tracking
- **Audit trail:** Immutable logs of all model changes and decisions
- **Annual review:** Periodic reassessment of all production models

SR 11-7 applies to any “quantitative method that processes input data into quantitative estimates”—this includes ML.

Model Governance: Three Lines of Defense

Model Governance Framework (Three Lines of Defense)



SR 11-7 / OCC 2011-12: Three Lines of Defense for Model Risk Management

Model Risk Taxonomy for Financial Institutions

Key risk categories from SR 11-7 and industry best practices



What is a Model Inventory?

Definition: A centralized register of every model deployed in the organization.

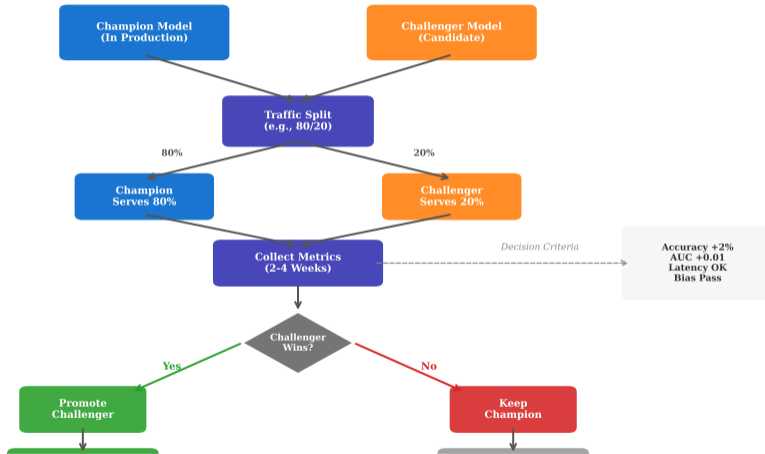
What to track per model:

- **Identity:** Name, version, owner, team
- **Purpose:** Business use case, decision type
- **Risk tier:** High / Medium / Low
- **Status:** Development / Staging / Production / Retired
- **Training data:** Dataset ID, date range, size
- **Performance:** AUC, accuracy, F1 at deployment
- **Validation:** Last review date, reviewer, findings
- **Monitoring:** PSI trend, accuracy trend
- **Dependencies:** Upstream data, downstream consumers
- **Next review:** Scheduled date

Why it matters: Large banks may have 500+ models in production. Without an inventory, “shadow models” proliferate—unmonitored, undocumented, and uncontrolled.

Regulators routinely ask: “How many models do you have in production?” You must be able to answer precisely.

Champion-Challenger Testing Framework



What is A/B Testing for Models?

Definition: Randomly splitting live traffic between the current model (champion) and a candidate (challenger) to compare real-world performance.

Typical setup:

- **Control:** 80% traffic to champion
- **Treatment:** 20% traffic to challenger
- **Duration:** 2–4 weeks for statistical significance
- **Randomization:** User-level (consistent assignment)

Promote challenger if:

- Accuracy improvement $> 2\%$ (statistically significant, $p < 0.05$)
- No increase in latency or error rate
- Passes fairness and bias audit
- Positive business metric lift

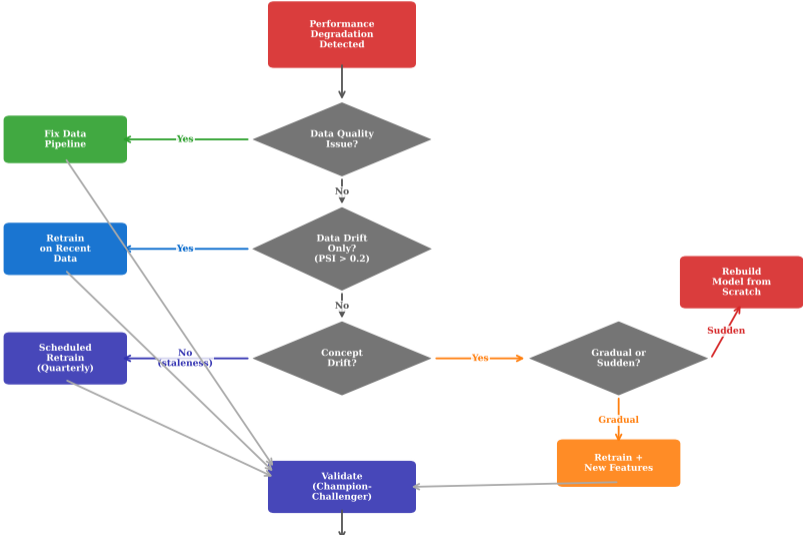
Financial A/B test example (credit scoring):

Metric	Champion	Challenger
AUC-ROC	0.87	0.90
Default rate	3.2%	2.8%
Approval rate	68%	71%
Avg. latency	45ms	52ms

Verdict: Challenger wins on accuracy and business metrics. Latency increase is acceptable. **Promote.**

Offline metrics (cross-validation) and online metrics (A/B test) can disagree—always trust the A/B test.

Retraining Decision Tree: When to Retrain vs. Rebuild



Scheduled Retraining

- Retrain on fixed intervals (weekly, monthly, quarterly)
- Predictable resource planning
- Prevents model staleness
- Aligns with regulatory review cycles

Limitations:

- May retrain unnecessarily (wastes compute)
- May miss urgent drift between cycles

Best for: Stable domains, regulatory compliance

Best practice: Hybrid approach—quarterly scheduled **plus** performance-triggered retraining.

Triggered Retraining

- Retrain when monitoring detects drift or performance drop
- Data-driven, resource-efficient
- Responds quickly to urgent issues

Trigger examples:

- $PSI > 0.20$ on 3+ features
- AUC drops $> 3\%$ from baseline
- Prediction volume anomaly ($>2x$ normal)
- Business KPI breach (default rate spike)

Best for: Dynamic domains (fraud, trading)

Most mature teams use a hybrid strategy: scheduled baselines supplemented by event-driven triggers.

What is CI/CD for ML?

Extension of traditional CI/CD to include data and model artifacts.

CI (Continuous Integration) for ML:

- Automated unit tests for feature engineering code
- Data validation (schema checks, statistical tests)
- Model training smoke tests (does it train without errors?)
- Integration tests (end-to-end pipeline on sample data)

CD (Continuous Delivery) for ML:

- Automated model packaging (Docker containers, ONNX export)
- Shadow deployment for safety validation
- Canary rollout (5% → 20% → 100%)
- Automated rollback if monitoring triggers alert

CT (Continuous Training):

- Automated retraining pipeline triggered by drift detection
- Validation gate: only promote if challenger beats champion
- Full audit trail for regulatory compliance

CI + CD + CT = the full automation stack for production ML.

What is a Feature Store? A data management layer that serves pre-computed features to both training and serving.

Online Store (Low Latency)

- Key-value store (Redis, DynamoDB)
- <10ms lookup per feature vector
- Latest feature values per entity
- Used for real-time inference

Offline Store (High Throughput)

- Data warehouse (BigQuery, S3)
- Historical feature values
- Point-in-time correctness
- Used for model training

Financial feature examples:

- 30-day rolling average transaction amount
- Count of transactions per hour-of-day
- Days since last address change
- Velocity: transactions per minute
- Debt-to-income ratio (updated monthly)

Tools:

- **Feast:** Open-source, cloud-agnostic
- **Tecton:** Enterprise, real-time focus
- **SageMaker Feature Store:** AWS-native
- **Databricks Feature Store:** Spark-native

Feature stores eliminate training-serving skew and enable feature reuse across teams and models.

Challenges Specific to Financial MLOps

Technical Challenges

- **Label delay:** Loan defaults take 6–12 months to materialize
- **Feedback loops:** Model predictions influence future training data
- **Legacy integration:** ML models calling 30-year-old COBOL systems
- **Latency:** Fraud detection needs <50ms response

Organizational Challenges

- Siloed teams: data science vs. engineering vs. risk
- Risk aversion: slow approval processes
- Explainability demands: “Why was this loan denied?”

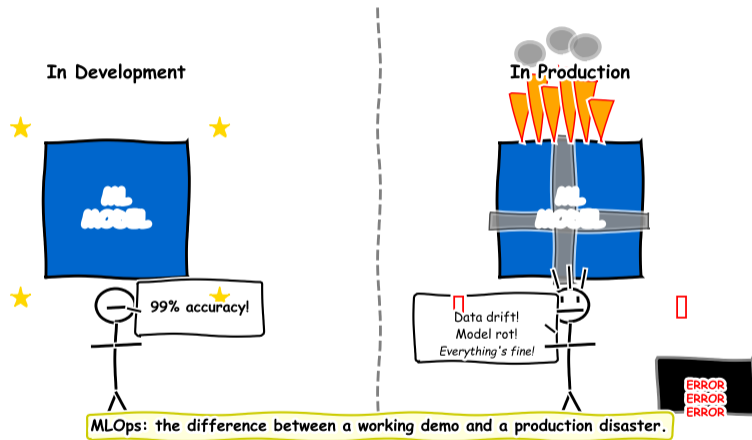
Regulatory Challenges

- **SR 11-7:** Model risk management (US)
- **MiFID II:** Algorithmic trading transparency (EU)
- **GDPR Art. 22:** Right to explanation for automated decisions
- **Fair Lending:** Non-discrimination in credit decisions

Common pitfalls:

- Alert fatigue from noisy monitoring
- Retraining too frequently (model churn)
- Confusing data quality bugs with real drift
- “Shadow models” running without governance

Successful financial MLOps requires solving technical, organizational, and regulatory problems simultaneously.



Sometimes the best way to remember a concept is to laugh about it.

Key Takeaways

- 1 **MLOps = DevOps for ML** – automate the lifecycle from data ingestion to model retirement with versioning, monitoring, and governance.
- 2 **Three drift types** – data drift ($P(X)$ changes), concept drift ($P(Y|X)$ changes), and prediction drift ($P(\hat{Y})$ changes). Each requires different detection and response.
- 3 **PSI is the industry standard** – Population Stability Index measures distribution shift. Thresholds: <0.10 stable, $0.10-0.20$ monitor, >0.20 retrain.
- 4 **Monitor six dimensions** – model performance, data drift, prediction drift, data quality, operational health, and business impact.
- 5 **SR 11-7 governs model risk** – three lines of defense (development, validation, audit) with mandatory model inventories and documentation.
- 6 **Champion-challenger testing** – never deploy without A/B testing against the current production model.
- 7 **Hybrid retraining** – combine scheduled (quarterly) and triggered (drift-based) retraining for the best coverage.

Building a model is 10% of the work. MLOps is the other 90% that makes it production-ready.

What we covered:

- The end-to-end ML lifecycle: ingestion → features → training → validation → deployment → monitoring → retraining
- Three types of model drift and how to detect each
- PSI calculation and interpretation for drift monitoring
- Building a monitoring dashboard with six metric categories
- SR 11-7 model governance and the three lines of defense
- Champion-challenger testing and A/B testing for model replacement
- CI/CD/CT pipelines and feature store architecture

The bottom line:

Without MLOps, models rot in notebooks. With MLOps, models deliver business value continuously—and regulators can sleep at night.

Next lesson: we will explore real-time data pipelines and streaming architectures for financial applications.