

DATA SCIENCE & APPLIED ML PLAYBOOK

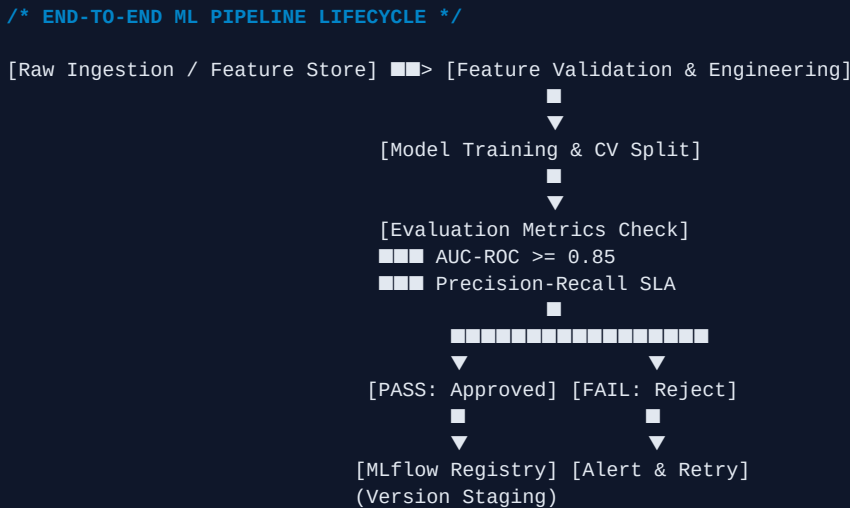
PRODUCTION MLOPS ARCHITECTURE, MODELING PATTERNS & SYSTEM DIAGRAMS

Key Execution Rules: Executive Summary: Applied Data Science requires a rigorous bridge between statistical experimentation and production software engineering. This playbook delivers 8 production-grade MLOps design patterns—from feature engineering and hyperparameter optimization to XAI explainability, drift detection, and real-time model serving.

TOPIC 01 // MLOPS LIFECYCLE

End-to-End Machine Learning Pipeline Architecture

A resilient machine learning pipeline orchestrates four distinct stages: **Data Ingestion & Validation**, **Feature Engineering & Store Ingestion**, **Distributed Model Training & Evaluation**, and **Model Registry (MLflow) Packaging**. Decoupling these stages ensures complete experiment reproducibility, automated retrainability, and seamless rollback capabilities.



Key Execution Rules: 1. Never train models directly on raw, unvalidated feature snapshots; rely on conformed Feature Store features. 2. Version control data splits, code commits, and hyperparameter configs together in the MLflow Model Registry. 3. Automate pipeline triggers based on scheduled cron intervals or data drift alerts.

TOPIC 02 // MODEL TRAINING

Class Imbalance & Cost-Sensitive Learning

In production tasks like fraud detection or medical diagnosis, minority positive classes often represent <1% of records. Standard accuracy metrics yield deceptive 99% baseline scores while missing all positive cases. Production mitigation requires synthetic oversampling (**SMOTE**, **ADASYN**), cost-sensitive loss functions (**Focal Loss**), and

PR-AUC threshold tuning.

Technique	Mechanism	Best Use Case
SMOTE / ADASYN	Interpolates synthetic minority samples in feature space	Moderate Imbalance (1%-10%)
Focal Loss	Down-weights easy negative examples during gradient updates	Extreme Imbalance (<1%)
Threshold Tuning	Optimizes prediction threshold on Precision-Recall Curve	Cost-Asymmetric Decisions

```
/* CLASS IMBALANCE & DECISION THRESHOLD PIPELINE */

[Imbalanced Dataset (99:1 Ratio)]
  ▾
[Apply SMOTE / ADASYN Oversampling] (Synthesize minority vectors)
  ▾
[Cost-Sensitive Gradient Boosted Trees] (Class Weight / Focal Loss)
  ▾
[Evaluate PR-AUC & Cost Matrix Optimization]
  ■■■ False Positive Cost = $10
  ■■■ False Negative Cost = $500
  ▾
[Optimal Decision Boundary Threshold (e.g. p >= 0.28)]
```

Key Execution Rules: 1. Never perform SMOTE or oversampling on the validation/test splits—apply strictly within training folds. 2. Evaluate imbalanced classifiers using PR-AUC (Precision-Recall Area Under Curve) instead of ROC-AUC. 3. Explicitly compute cost matrices (Cost of False Negative vs. False Positive) when selecting decision thresholds.

TOPIC 03 // MODEL TUNING

Hyperparameter Optimization & Cross-Validation

Grid Search and Random Search are inefficient for high-dimensional hyperparameter spaces. Modern HPO frameworks like **Optuna** employ **Tree-structured Parzen Estimators (TPE)** and Hyperband pruning to convergence 10x faster. Cross-validation for time-dependent features requires strict **Time-Series Walk-Forward Splits** to prevent temporal leakage.

```

/* OPTUNA BAYESIAN HPO & TIME-SERIES SPLIT */

[Feature Dataset with Temporal Axis]
    ■
    ▼
[Time-Series Walk-Forward Cross-Validation]
■■■■ Fold 1: [Train 2023] ■■> [Validate Q1 2024]
■■■■ Fold 2: [Train 2023 + Q1 2024] ■■> [Validate Q2 2024]
■■■■ Fold 3: [Train 2023 + Q1-Q2 2024] ■■> [Validate Q3 2024]
    ■
    ▼
[Optuna TPE Bayesian Sampler]
■■■■ Sample Hyperparameter Vector (learning_rate, depth, l2_reg)
■■■■ Evaluate CV Performance
■■■■ Prune Unpromising Trials (MedianPruner)
    ■
    ▼
[Optimal Parameter Configuration]

```

Key Execution Rules: 1. Never use standard K-Fold CV on temporal data; always enforce time-series forward-chaining splits. 2. Use Optuna's MedianPruner to kill low-performing hyperparameter trials early and save compute GPU/CPU hours. 3. Store hyperparameter search trials and convergence plots in MLflow artifact storage.

TOPIC 04 // MODEL GOVERNANCE

Model Explainability & Interpretability (XAI)

Complex ensemble algorithms (XGBoost, LightGBM) and Neural Networks act as black boxes. **SHAP (Shapley Additive exPlanations)** leverages cooperative game theory to assign fair marginal contribution scores to every input feature for both global feature ranking and local individual prediction audits.

```
/* SHAP LOCAL FEATURE ATTRIBUTION FLOW */
```

```
[Individual Model Prediction (e.g. Loan Default Risk = 78%)]  
    ▼  
[SHAP TreeExplainer Calculation]  
(Calculate Shapley values vs base value)  
    ▼  
┌───────────────────────────────────────────────────────────────────────────────────┐  
│                                                                                   │  
▼                                                                                   ▼  
[Positive Contributors (+%)]               [Negative Contributors (-%)]  
■■■ Debt-to-Income (+25%)                 ■■■ High Credit Score (-12%)  
■■■ Recent Delinquencies (+18%)           ■■■ Stable Employment (-8%)  
    ▼                                     ▼  
[Generate Local Waterfall / Force Plot]  
    ▼  
[Auditable Compliance Explanation]
```

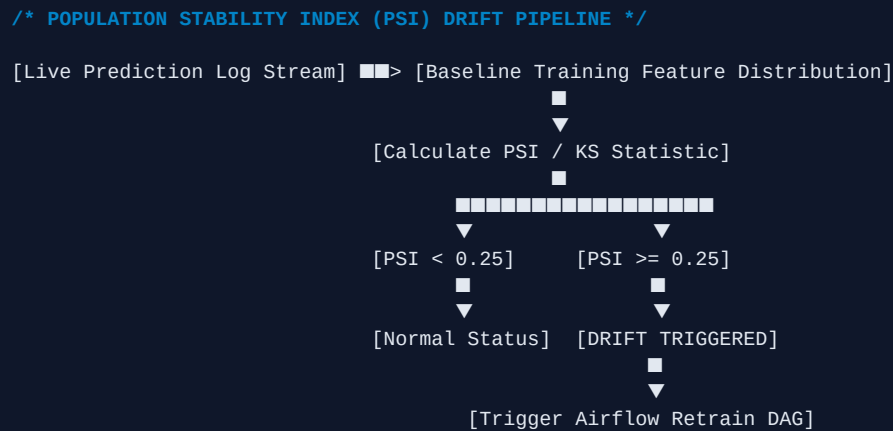
Key Execution Rules: 1. Use TreeSHAP for tree-based ensembles for exact fast computation ($O(TLD^2)$ complexity). 2. Generate global SHAP summary beeswarm plots to detect feature interactions and non-linearities. 3. Export local waterfall plots as JSON payloads alongside production model predictions for regulatory audits.

TOPIC 05 // PRODUCTION OBSERVABILITY

Model & Data Drift Monitoring

Production model performance degrades over time due to **Data Drift** (covariate shift in input features $P(X)$) and **Concept Drift** (relationship shift between features and targets $P(Y|X)$). Statistical tests like the **Population Stability Index (PSI)** and Kolmogorov-Smirnov (KS) test detect drift in real time.

PSI Score Range	Drift Interpretation	Action Required
PSI < 0.10	No significant distribution shift	No action needed (Normal)
0.10 <= PSI < 0.25	Moderate distribution drift	Flag for inspection / Warning
PSI >= 0.25	Severe population shift	Trigger Automated Retraining DAG

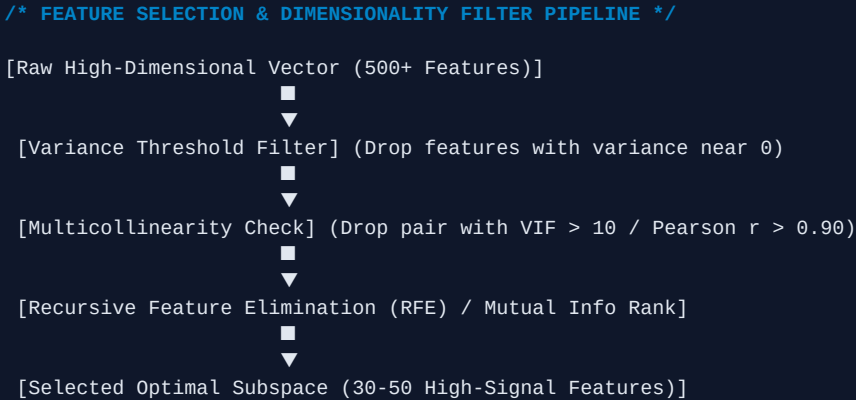


Key Execution Rules: 1. Compute PSI on rolling 7-day windows against baseline training distributions. 2. Distinguish Covariate Drift (input shifts) from Concept Drift (true label shift) before retraining models. 3. Maintain a golden reference dataset from model training for distribution comparisons.

TOPIC 06 // FEATURE SPACE OPTIMIZATION

Feature Selection & Dimensionality Reduction

High-dimensional feature spaces suffer from the curse of dimensionality, leading to model overfitting and high inference latency. Combining filter methods (**Mutual Information**, high collinearity variance thresholding), wrapper methods (**RFE**), and manifold learning (**PCA**, **UMAP**) strips out noise while preserving maximum variance.



Key Execution Rules: 1. Remove collinear features ($r > 0.85$ or $VIF > 10$) to stabilize coefficient estimation. 2. Use Mutual Information for non-linear feature-target relationships that Pearson correlation misses. 3. Fit PCA / UMAP transformers strictly on training sets and transform test sets to avoid data leakage.

TOPIC 07 // MODEL DEPLOYMENT

Real-Time vs. Batch Model Serving Architecture

Machine Learning serving architectures split into two paradigms: **Real-Time REST/gRPC Microservices** (sub-50ms latency) and **High-Throughput Batch Scoring**. Optimized runtime engines (ONNX Runtime, NVIDIA Triton, TorchScript) accelerate model execution speed.

Dimension	Real-Time Inference	Batch Scoring
Latency Target	Sub-50 milliseconds (p99)	Minutes to Hours
Protocol / Format	gRPC / REST (FastAPI, Triton)	Spark / Ray / Parquet Files
Compute Resource	GPU / CPU Pod Auto-scaling	Distributed Transient Clusters

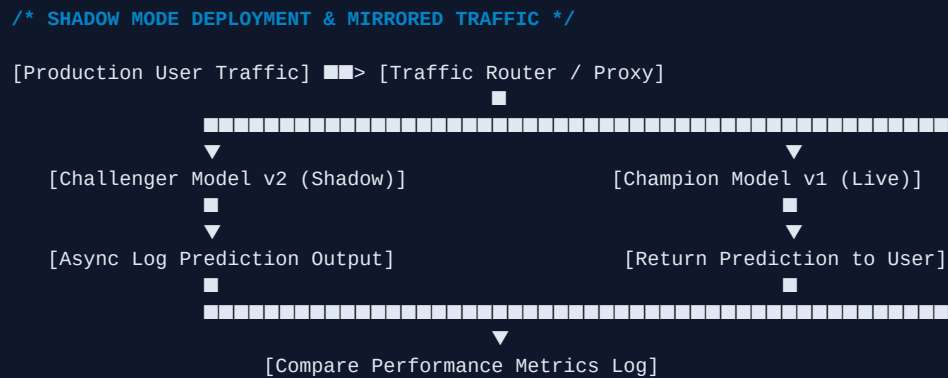


Key Execution Rules: 1. Convert Scikit-Learn and PyTorch models to ONNX format for 3x–5x latency reduction in C++ runtime engines. 2. Implement asynchronous batching in Triton / FastAPI to maximize GPU utilization. 3. Decouple feature fetching (sub-10ms Redis lookup) from model execution.

TOPIC 08 // RELEASE ENGINEERING

Shadow Deployments & Multi-Armed Bandits

Replacing a production model carries operational risk. **Shadow Mode Deployment** mirrors live production traffic to a candidate model without returning its predictions to end users. For dynamic traffic allocation, **Multi-Armed Bandits (Thompson Sampling)** continuously route traffic toward higher-performing model variants.



Key Execution Rules: 1. Always run new model candidates in Shadow Mode for at least 48 hours to evaluate latency and error logs under production traffic. 2. Use Thompson Sampling Multi-Armed Bandits for rapid conversion rate optimization without traditional A/B test sample size delays. 3. Maintain automated instant rollback mechanisms to Champion v1 if Challenger v2 returns HTTP 5xx errors.