

Machine Learning Projects: End-to-End Overview

From Data
to Real-World
Impact



Key Differences

ML projects differ from software: data-dependent, probabilistic, silently degrading, iterative



Lifecycle

Lifecycle: framing → feasibility → data → features → modeling → evaluation → deployment → monitoring → retraining



Effort reality

Effort reality: data prep and features ~60%, modeling ~10%, production engineering 30–40%



Deliverables

Deliverables: working model, reproducible pipeline, versioned data/model record, proven value



Common failures

Common failures: unclear goals, weak data, misaligned stakeholders, unmonitored drift



Roadmap

Roadmap: from framing to a deployment-ready, documented, reproducible project skeleton

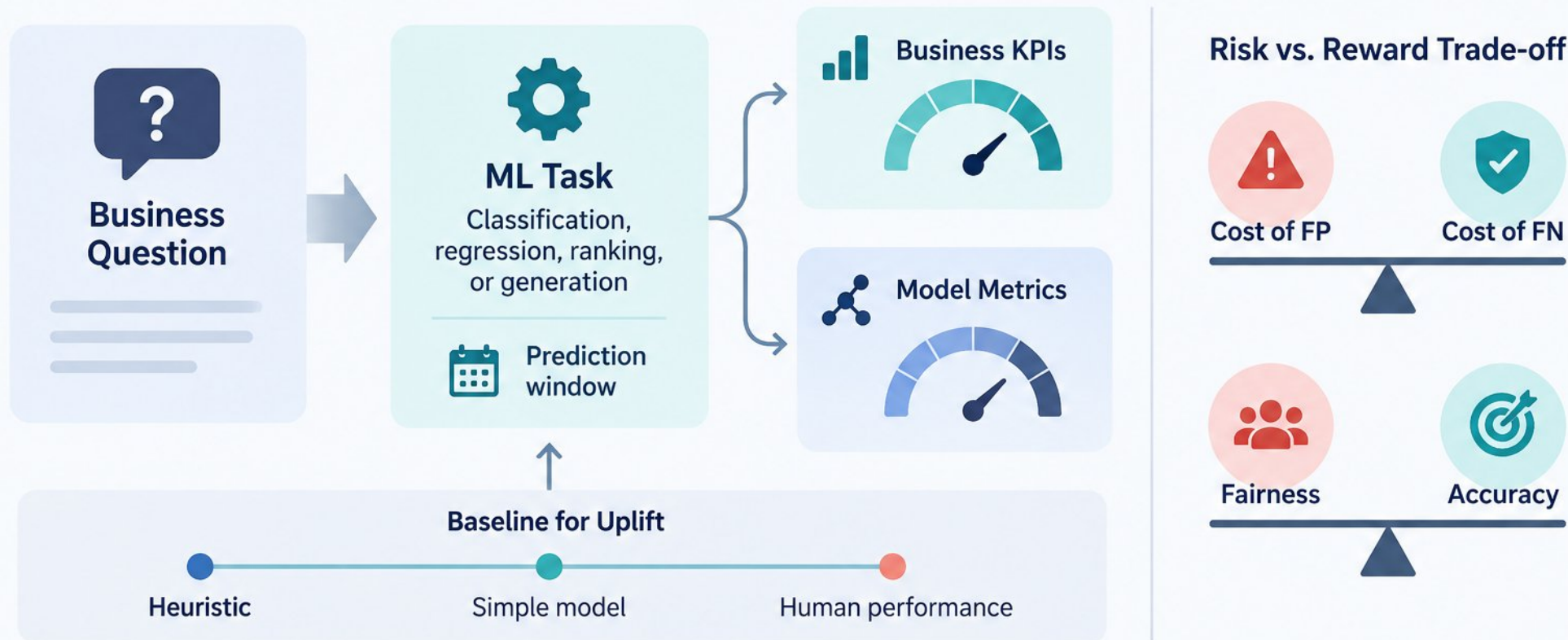


Why ML Projects Fail and How to De-risk Them

- **67–87%** of AI/ML projects fail or never reach production
- Roughly **twice** the failure rate of non-AI IT projects
- **Top causes:** stakeholder expectations (54.8%), data fundamentals (43.5%), unclear goals (38.7%)
- **Technical modes:** data leakage, label noise, training-serving skew
- **Organizational modes:** no owner, scope creep, no rollback authority
- **De-risk:** small reversible steps, one-page brief, evidence gates, kill or pivot fast

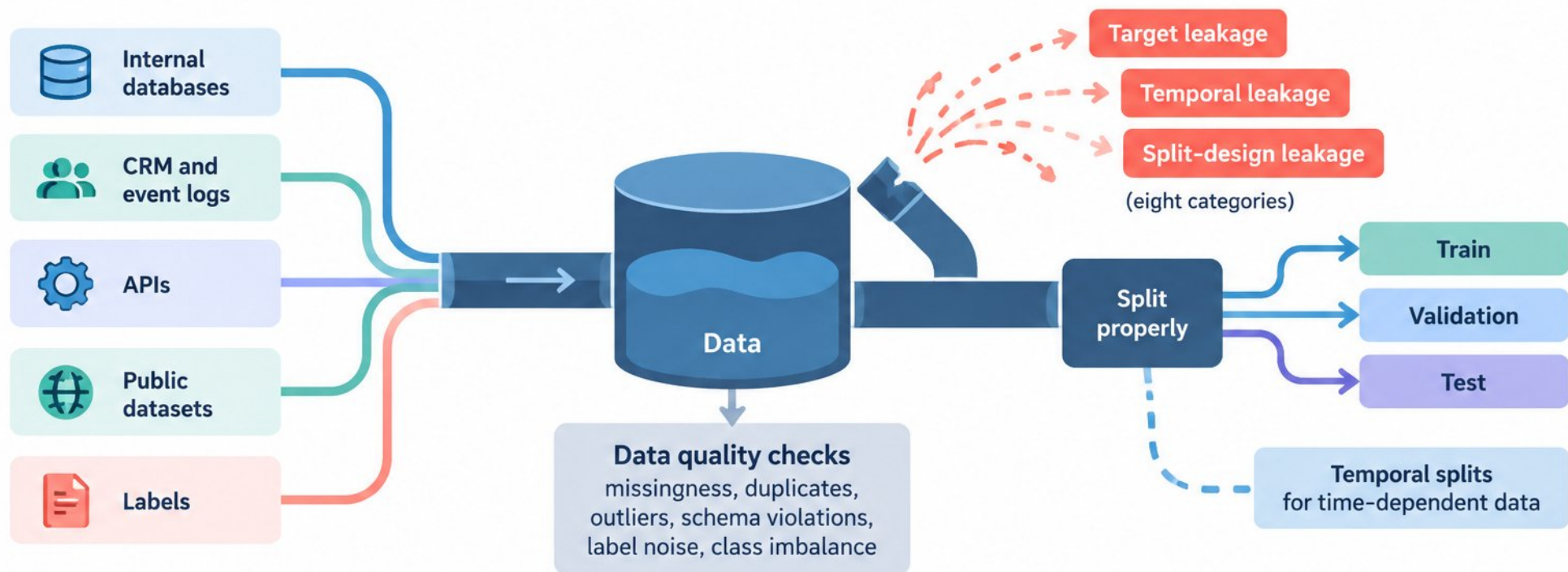


Framing the Problem and Defining Success Metrics



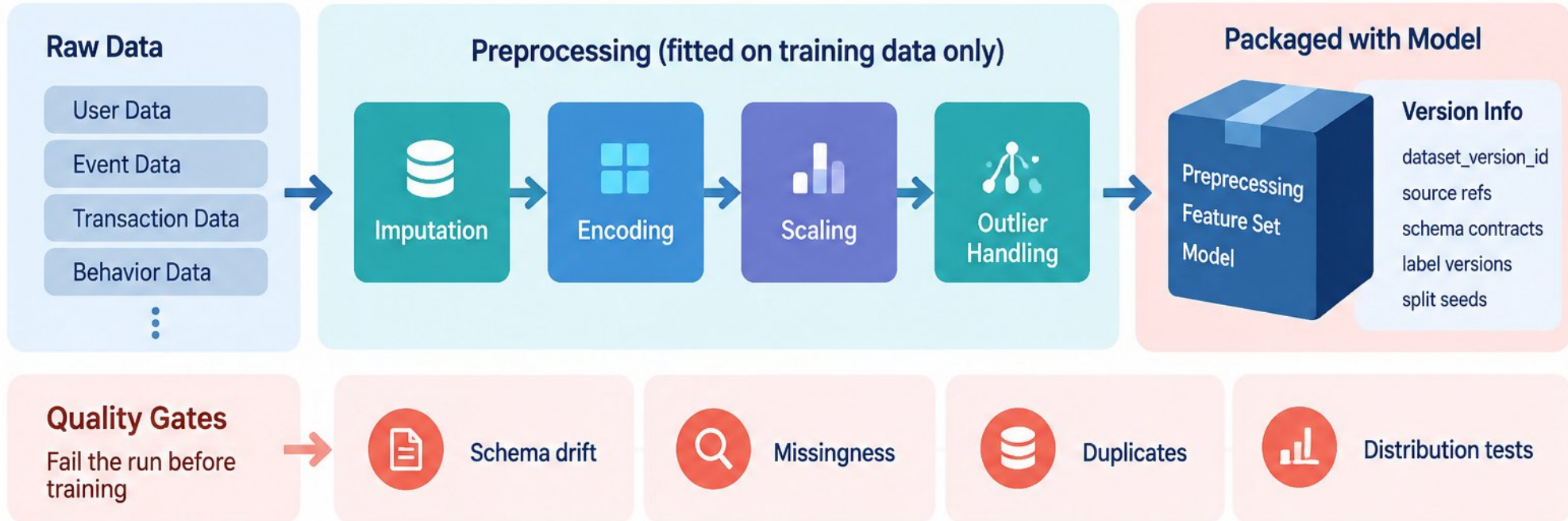
- 1 Translate the business question into a decision, a user, and a failure cost
- 2 Name the ML task: classification, regression, ranking, or generation, plus prediction window
- 3 Track two parallel metric sets: business KPIs and model metrics
- 4 One primary outcome authorizes launch; guardrails can veto it
- 5 Quantify uplift against a baseline: heuristic, simple model, or human performance
- 6 Fix scope early: data, latency, cost per prediction, privacy, regulatory class

Data Acquisition, Quality, and Leakage Prevention



-  Combine internal databases, CRM and event logs, APIs, public datasets, and labels
-  Data quality checks: missingness, duplicates, outliers, schema violations, label noise, class imbalance
-  EDA should inform modeling choices, not produce decorative charts
-  Guard against leakage: target, temporal, and split-design leakage (eight categories)
-  Split properly: train/validation/test, stratified CV, temporal splits for time-dependent data
-  Establish data rights, consent, and retention rules before modeling; log missing evidence

Feature Engineering and Reproducible Data Pipelines



- Domain knowledge beats algorithm switching: one strong feature outperforms a complex model
- Practical patterns: ratios, interactions, and time-based features (recency, frequency, trend)
- Preprocessing fitted on training data only, then packaged with the model
- Data versioning: dataset_version_id, source refs, schema contracts, label versions, split seeds
- Quality gates fail the run before training: schema drift, missingness, duplicates, distribution tests
- Log dataset_version_id + pipeline_version + feature_set on every run for fast debugging

Models, Training Strategies, and Experiment Tracking

From model choice to reproducible results



Model families

- Model families: linear/logistic baselines, tree ensembles for tabular, neural nets when justified



Evaluation protocols

- Evaluation protocols: cross-validation, group splits, temporal splits for time series



Tuning

- Tuning: grid, random, Bayesian, successive halving; add regularization and early stopping



Track everything

- Track everything: parameters, metrics, artifacts, git commit, environment



Tool landscape 2026

- Tool landscape 2026: MLflow (open-source, registry), W&B (visuals, sweeps), Comet, ClearML, DVC, TensorBoard

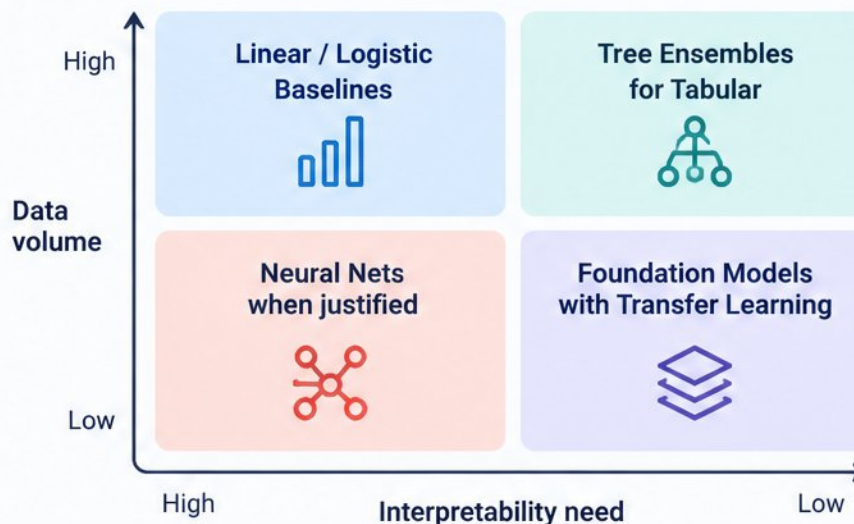


Fair comparison

- Fair comparison: fix splits, never tune on test set, resist overfitting to validation

Model Family Decision Guide

Match your data and goals to the right model family



Data modality



Text



Images



Audio



Tabular

Experiment Tracking Ledger

Automatically record what matters for every run



Parameters



Metrics



Artifacts



Git commit



Environment



Framing



Data



Model



Deploy



Monitor



Retrain

An iterative lifecycle for better models

Evaluation, Validation, and Error Analysis That Build Trust



Choose metrics for the decision:
precision vs. recall is a cost trade-off



ROC-AUC vs. PR-AUC under
imbalance; calibration for probability
reliability



Thresholds are business choices
set by stakeholders, not defaults



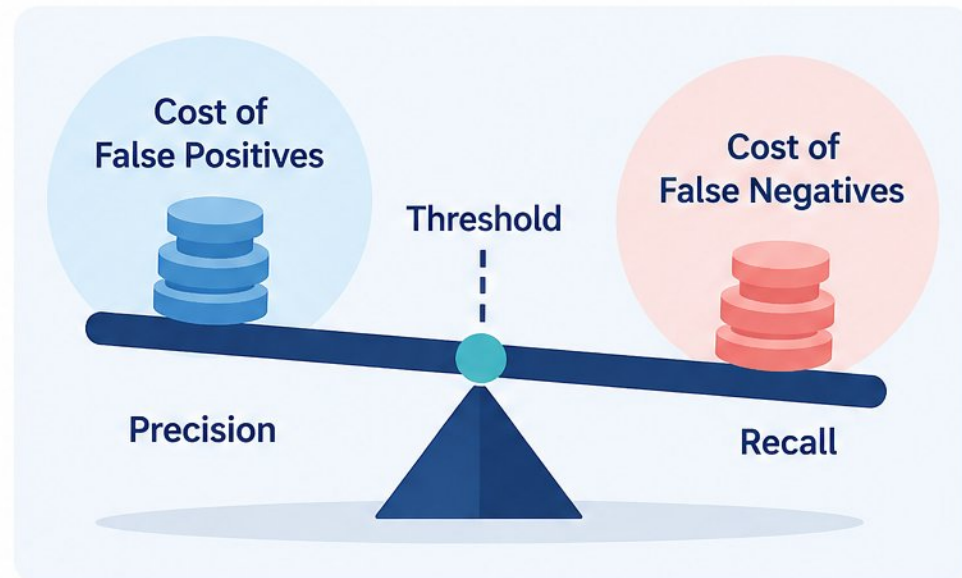
Report confidence intervals and
sample size before declaring a win



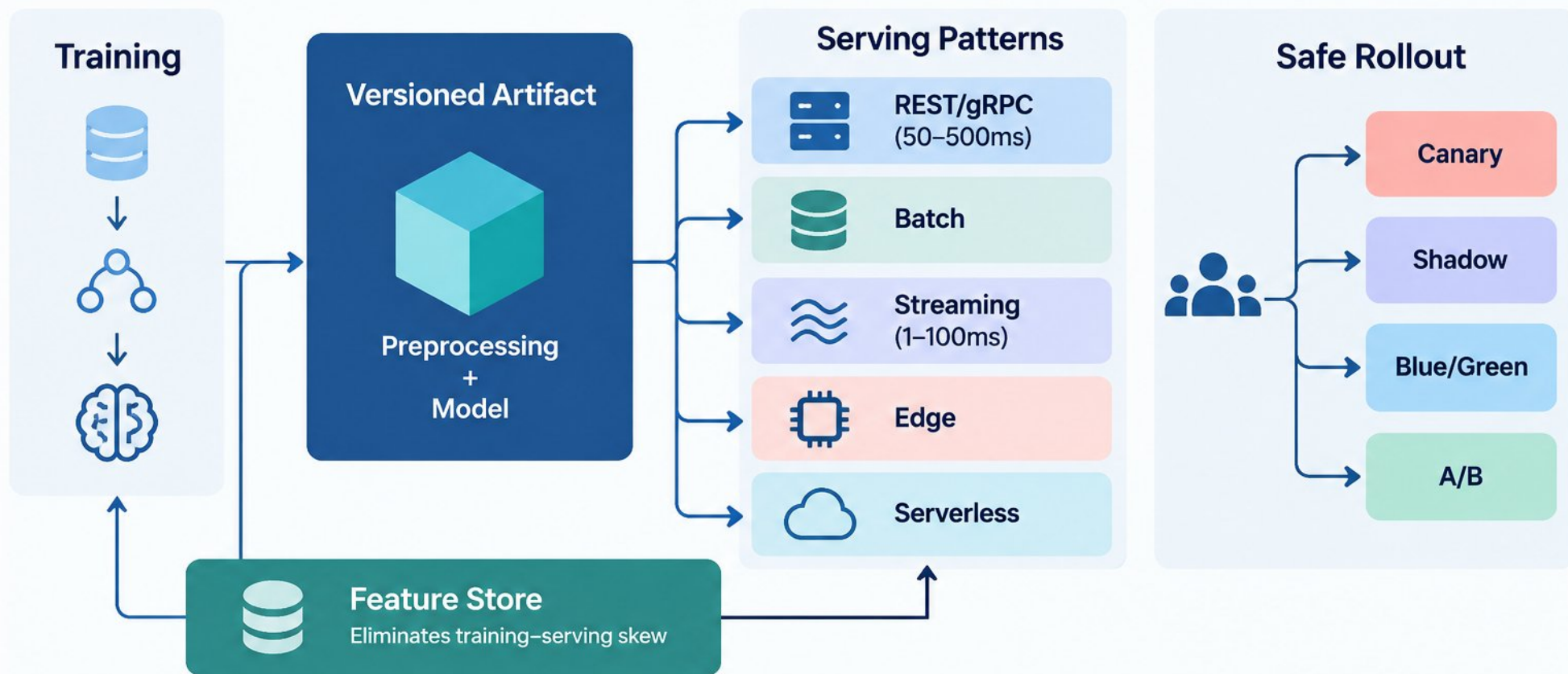
Slice-based error analysis:
worst-case subgroups reveal
hidden failures



Strong offline metrics $\neq$ business
value — validate with a controlled
experiment



From Notebook to Production: Deployment and Serving Patterns



Package preprocessing + model as one versioned artifact; pin environments



Serving patterns: REST/gRPC (50–500ms), batch, streaming (1–100ms), edge



Feature store eliminates training-serving skew, the top production failure



Safe rollout: canary, shadow, blue/green, A/B with predefined thresholds



CI/CD quality gates: schema validation, unit tests, smoke tests, latency checks



Version every endpoint and return model version; add health endpoints

Monitoring, Drift Detection, and Retraining Loops

KEEP MODELS RELIABLE IN A CHANGING WORLD



A degraded model keeps serving: latency, errors, throughput stay normal



Feature drift (PSI/KS): hours • Prediction drift: hours • Concept drift: days to months



PSI thresholds: <0.10 stable, $0.10-0.25$ warning, >0.25 act; recalibrate on your data



Delayed labels: use proxies — prediction shift, confidence drop, segment volume



Retrain on drift first, performance second, plus a maximum-staleness safety net



Promotion gated: champion-challenger, shadow, canary, then promote or auto-rollback



Day-one stack: latency percentiles, error rate, volume, prediction logs, one drift check per feature

Drift Monitoring

Feature drift
(PSI/KS)

<0.10
stable

$0.10-0.25$
warning

>0.25
act

Prediction drift

<0.10
stable

$0.10-0.25$
warning

>0.25
act

Concept drift

<0.10
stable

$0.10-0.25$
warning

>0.25
act



Operational Signals



Latency
percentiles



Error
rate



Volume



Prediction logs



One drift check
per feature

Promotion Gate

Champion-
challenger

Shadow

Canary



Promote



Auto-rollback

Case Studies and Postmortems: What Real Projects Teach

REAL PROJECTS
DEEPER LESSONS
STRONGER OUTCOMES



Demand forecasting at a construction supplier hit ~99.3% WAPE – governance, not the algorithm, sustained it



A skewed fine-tuning quarter caused 10x refunds in four hours, costing \$1.5M



Missed missing-value encoding drift caused 3 weeks of wrong credit decisions



A database permission change produced an oversized feature file – taking down critical traffic



Agents optimize the metric they can see; they introduce leakage and silently drop samples



Technical skill rarely limited outcomes – alignment, data quality, and discipline decided results

REAL INCIDENTS.
LASTING INSIGHTS.

LEARN

IMPROVE

BUILD BETTER

Pitfalls, Responsible Practice, and Governance Essentials

SAFER MODELS. FAIRER OUTCOMES. STRONGER TRUST.



Design against leakage, overfitting to validation, silent failure, unmonitored drift, training–serving skew, no rollback



Fairness metrics must match intended use: performance parity, calibration, equalized odds, statistical parity



Report disaggregated results by demographic group, not one aggregate number



Bias mitigation categories: preprocessing (resampling, reweighting), in-processing, postprocessing (threshold adjustment)



Document with model cards, dataset datasheets, immutable prediction logs, and a named model owner



Governance context: EU AI Act high-risk obligations, NIST AI RMF, ISO/IEC 42001, OECD due diligence steps

From One Number to a Clearer Picture

Report disaggregated results by demographic group, not one aggregate number.



Not one aggregate number



Bias Mitigation Approaches

Preprocessing, in-processing, and postprocessing can help address bias.

Preprocessing
(resampling, reweighting)

In-processing

Postprocessing
(threshold adjustment)

Fairness

Accuracy

Auditability and Accountability

Document key artifacts and assign a model owner with authority to switch the system off.



Model cards



Dataset
datasheets



Immutable
prediction logs



A named
model owner

Governance Context

Align with established frameworks and obligations.

EU AI Act high-risk obligations

NIST AI RMF

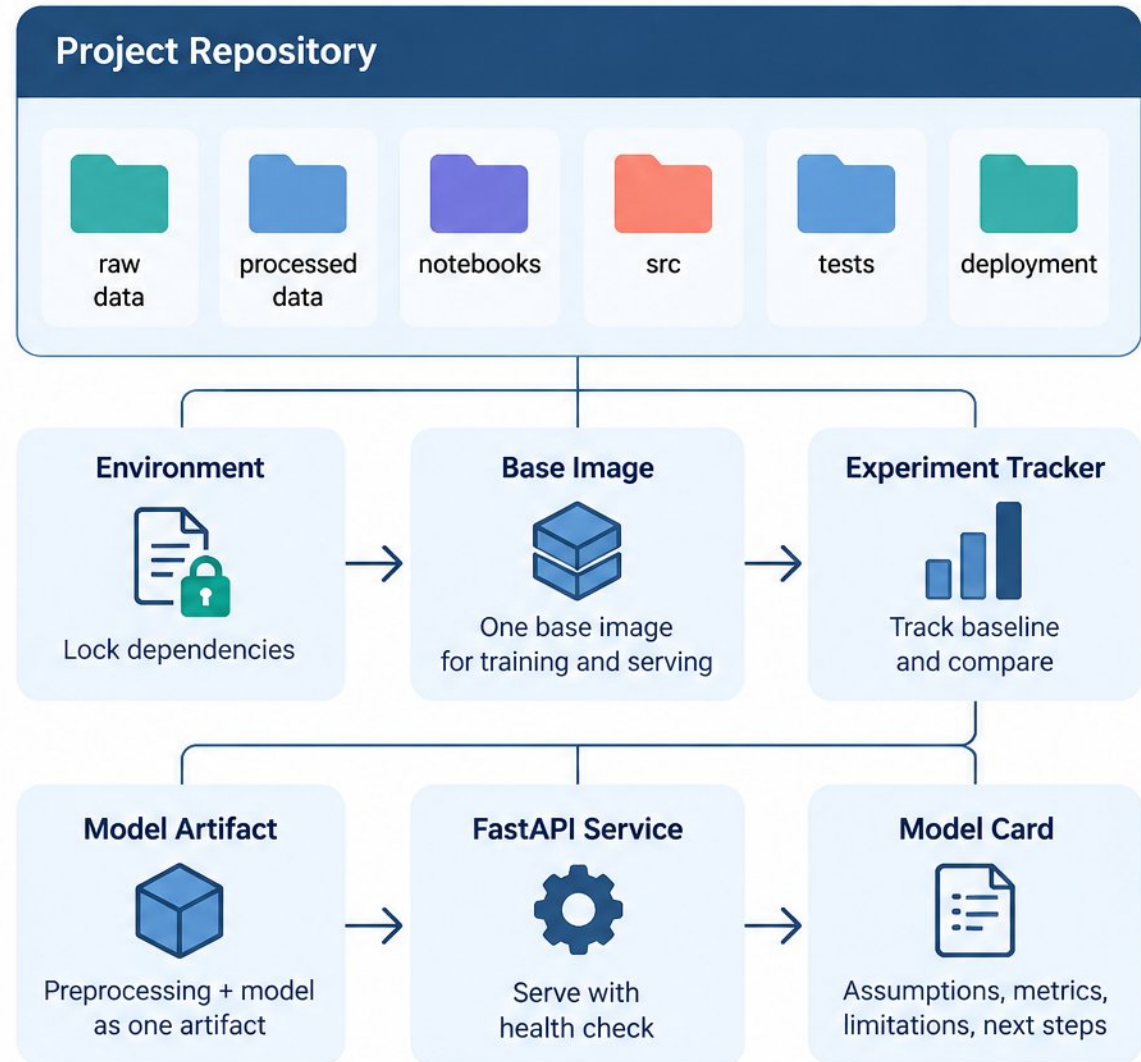
ISO/IEC 42001

OECD due diligence steps

Hands-On Workshop: Building a Reproducible Project Skeleton

FROM STRUCTURE TO A REPEATABLE, DEPLOYABLE PIPELINE

- 1 Repository: raw/processed data, notebooks, installable src, tests, deployment
- 2 Lock dependencies, seed randomness, pin one base image for training and serving
- 3 Version datasets; log dataset_version_id + pipeline_version every run
- 4 Train a baseline first, track it in an experiment tracker, benchmark against it
- 5 Package preprocessing + model as one artifact; serve via FastAPI with health check
- 6 Write a short model card: assumptions, metrics, limitations, next steps
- 7 Extension: add one drift check and one retraining trigger



Next Steps: Portfolio, Specialization, and Career Paths

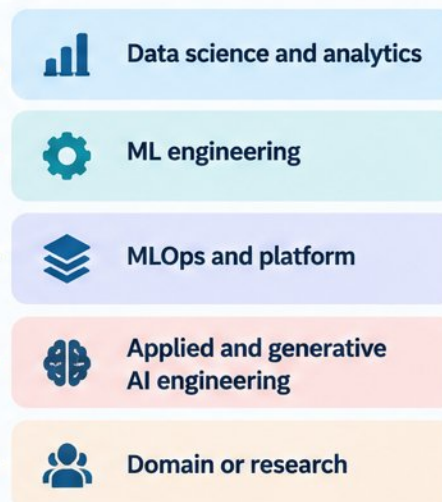
- Two or three deep projects beat ten shallow notebooks
- Hiring signal: deployed endpoint, reproducible code, baseline comparison
- Pick one lane: data science, ML engineering, MLOps, GenAI, domain
- Best projects: churn, fraud, RAG with evals, CV, MLOps, latency
- In interviews, explain every decision, metric trade-off, failure recovery
- Stay current: RAG, LLM evaluation, and production ML now baseline skills



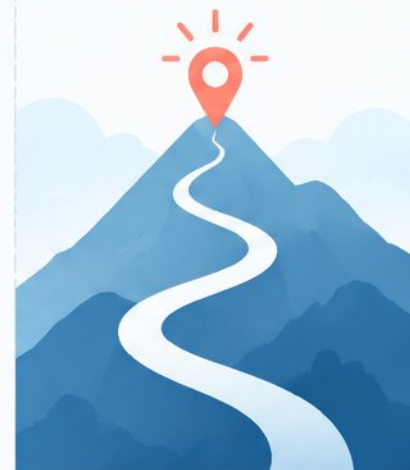
Your Portfolio



Specialization Paths



Next Milestone



PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/14e27bbd/public