



CHECKLIST

Building Secure ML Pipelines with Model Validation Checks: Production Readiness Checklist

A practical checklist for validating machine learning models before promotion to production. Covers data integrity, provenance, performance, security-sensitive behavior, runtime compatibility, and fail-closed approval criteria.

Building Secure ML Pipelines with Model Validation Checks

Production Readiness Checklist

Use this checklist to decide whether a machine learning model is safe enough to move into the next environment. Treat every unchecked item as a release blocker until the issue is explained and resolved.

1) Model artifact and provenance

- ☐ The model artifact has a unique version or immutable identifier.
- ☐ The training code commit hash is recorded.
- ☐ The training dataset version is recorded.
- ☐ The dependency lockfile or package manifest is recorded.



- ☐ The build environment is identified and reproducible.
- ☐ The artifact can be traced back to the source pipeline or training job.
- ☐ The model was produced from an approved source repository or workflow.
- ☐ There is evidence that the artifact was generated by the expected pipeline, not a manual or ad hoc process.

Evidence to collect: model registry entry, build logs, commit hash, dataset version, dependency manifest, pipeline run ID.

2) Data integrity and feature contract

- ☐ The input schema matches the approved feature contract.
- ☐ Required columns are present.
- ☐ No unexpected columns are being consumed silently.
- ☐ Feature names, types, and encodings are validated before promotion.
- ☐ Null, missing, and default value rules are explicit.
- ☐ Categorical values are checked against an allowlist or approved domain.
- ☐ Numeric ranges are checked for impossible or unsafe values.
- ☐ Duplicate records are detected and handled appropriately.
- ☐ Broken joins, label mismatches, or data leakage indicators have been reviewed.
- ☐ Upstream schema changes are detected automatically.

Block promotion if: feature shapes, encodings, or value ranges differ from the approved contract.

3) Training and evaluation data quality

- ☐ The training and validation datasets come from approved sources.
- ☐ Data collection and labeling methods are documented.



- ☐ Poisoning indicators or suspicious anomalies have been reviewed.
- ☐ Train/validation/test splits are stable and reproducible.
- ☐ There is no evidence of target leakage.
- ☐ Class imbalance has been considered in metric selection.
- ☐ Data drift or distribution shift is measured against the expected baseline.
- ☐ Sampling methods are documented and acceptable for the use case.

Evidence to collect: dataset lineage, split logic, quality reports, drift reports, labeling documentation.

4) Baseline performance validation

- ☐ The success metrics were defined before release approval.
- ☐ Accuracy is not the only metric considered.
- ☐ Metrics relevant to the use case are evaluated, such as precision, recall, calibration, false positives, false negatives, or latency.
- ☐ Results meet or exceed the approved baseline.
- ☐ Segment-level performance has been checked where applicable.
- ☐ Worst-case or edge-case behavior is acceptable.
- ☐ Confidence intervals or variability across runs have been reviewed.
- ☐ Regression against prior model versions has been assessed.

Block promotion if: the model improves one metric but meaningfully degrades a critical operational metric.

5) Security-sensitive behavior

- ☐ The model is tested for unusual confidence on out-of-distribution inputs.
- ☐ Predictions are stable for near-identical inputs where stability is expected.



- ☐ Sensitive outputs do not expose training data or private information.
- ☐ Label leakage has been checked in the feature set and evaluation flow.
- ☐ Adversarial or manipulation probes have been run where relevant.
- ☐ Fairness, privacy, and abuse risks have been reviewed for the domain.
- ☐ Thresholds and guardrails are defined for suspicious behavior.
- ☐ Known failure modes are documented and accepted by stakeholders.

Examples of security checks: leakage review, abnormal confidence test, perturbation test, privacy review, adversarial robustness probe.

6) Runtime compatibility and deployability

- ☐ The model loads successfully in the approved inference runtime.
- ☐ Required libraries and system dependencies are available.
- ☐ Serialized format is supported by the target environment.
- ☐ CPU, memory, and storage usage are within approved limits.
- ☐ Inference latency meets service objectives.
- ☐ Hardware assumptions match the deployment target.
- ☐ Base image, OS package, and runtime versions are compatible.
- ☐ Any feature flags or environment settings required for correct behavior are documented.
- ☐ The model can be rolled back safely if needed.

Block promotion if: the model works in the lab but cannot be loaded or executed in the target runtime.

7) Reproducibility and rollback readiness

- ☐ The same inputs can produce the same artifact or an explainable equivalent.



- ☐ Random seeds or training nondeterminism are documented.
- ☐ Build and release steps are automated.
- ☐ Rollback instructions are available and tested.
- ☐ Prior approved versions remain accessible.
- ☐ Version comparison is possible between the current and previous model.
- ☐ The release record includes enough evidence for audit or incident review.

Evidence to collect: release notes, artifact hashes, rollback procedure, reproducibility notes.

8) Operational approval checks

- ☐ The validation results were reviewed by the appropriate owner.
- ☐ Any exceptions are documented and explicitly approved.
- ☐ The model has a named business or system owner.
- ☐ The deployment environment matches the intended risk level.
- ☐ Monitoring thresholds are defined for post-deploy observation.
- ☐ Alerts exist for performance regressions, schema drift, and runtime failures.
- ☐ The release decision is recorded as pass, fail, or approved exception.

Fail closed rule: if evidence is missing, incomplete, or contradictory, stop promotion.

Promotion decision template

Use this short decision record before allowing a model to move forward.

Field Entry
Model name



Model version
Training code commit
Training data version
Validation date
Reviewer
Target environment
Baseline metrics met?
Security checks passed?
Runtime compatibility confirmed?
Exceptions approved?
Final decision Pass / Fail / Hold

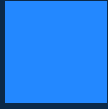
Final release rule

Promote the model only if:

- all required checks are complete,
- all critical thresholds are met,
- all required evidence is present,
- and any exception is explicitly approved.

If you cannot verify the model, the data, or the runtime with confidence, do not deploy it.

Quick one-line test



Ask this before every release:

Can we prove this model is safe enough to trust in the next environment?

If the answer is not clearly yes, the model should remain out of production.