CHECKLIST

Explainable AI for Model Debugging and Risk Control Checklist

A practical checklist for using explainable AI to debug model behavior, validate explanations, and reduce production risk in machine learning systems.

Explainable AI for Model Debugging and Risk Control

Operational Checklist

Use this checklist to decide when explanations are useful, how to interpret them safely, and what actions to take before trusting a model output in production.

---

1) Define the operational question

- [ ] Is the goal to understand why a single prediction happened?
- [ ] Is the goal to check whether the model relies on stable, policy-approved signals?
- [ ] Is the goal to decide whether an output is safe to automate, route, block, or escalate?
- [ ] Have you identified the business or system impact if the prediction is wrong?
- [ ] Have you marked the case as low, medium, or high risk before investigating?

---

2) Confirm the explanation method matches the use case

- [ ] Use a local explanation when triaging one surprising prediction.
- [ ] Use a global explanation when checking model behavior across many cases.
- [ ] Use a counterfactual explanation when you need to understand what change would alter the outcome.
- [ ] Avoid treating any explanation as ground truth.
- [ ] Confirm the explanation method is appropriate for the model type and feature set.
- [ ] Document the explanation method, parameters, and time window used.

---

3) Check whether the explanation makes domain sense

- [ ] Do the top drivers match what subject matter experts would expect?
- [ ] Are any dominant features suspicious because they are unstable, sensitive, or easy to manipulate?
- [ ] Does the explanation rely on proxy variables that may encode protected or policy-restricted information?
- [ ] Are missing values, defaults, or fallback values unexpectedly influential?
- [ ] Are timestamp, identifier, routing, or metadata fields contributing too much weight?
- [ ] Does the output depend on signals that should be weak rather than dominant?

---

4) Compare the case against nearby examples

- [ ] Review a small set of similar records with similar inputs.
- [ ] Check whether the explanation pattern repeats across nearby cases.
- [ ] Determine whether the behavior looks like a one-off anomaly or a systemic issue.

- [ ] Compare affected cases with unaffected cases from the same segment.
- [ ] Look for explanation consistency across the same customer group, traffic source, or workflow branch.

---

5) Validate the model against operational health signals

- [ ] Check input drift for the features driving the explanation.
- [ ] Check concept drift or label shift if outcome data is available.
- [ ] Review calibration to see whether predicted probabilities still match observed outcomes.
- [ ] Inspect feature freshness, null rates, schema changes, and data pipeline failures.
- [ ] Confirm that upstream enrichment, transformation, and feature store jobs are healthy.
- [ ] Correlate the explanation change with any recent release, config change, or data dependency update.

---

6) Distinguish model behavior from pipeline behavior

- [ ] Verify whether the issue started after a code deploy, schema update, or feature change.
- [ ] Check whether the model is reacting to corrupted, missing, delayed, or reordered inputs.
- [ ] Confirm the same behavior appears in a controlled test or replay environment.
- [ ] Separate true model sensitivity from upstream data quality failures.
- [ ] Review whether the issue affects all traffic or only one region, segment, or source.

---

7) Assess whether the output is safe to act on

- [ ] Is the prediction strong enough to automate downstream action?
- [ ] Does the explanation indicate the model is depending on a stable and acceptable signal?
- [ ] Could an attacker manipulate the feature pattern shown in the explanation?
- [ ] Could the output produce unacceptable user impact if wrong?
- [ ] Should the case be routed to manual review instead of automated action?
- [ ] Should the output be blocked, capped, or delayed until the issue is resolved?

---

8) Decide on the operational response

Choose one:

- [ ] Accept ? the explanation is consistent with expectations and health checks are clean.
- [ ] Investigate ? the explanation is plausible but needs deeper review.
- [ ] Block ? the output is too risky to use automatically.
- [ ] Route to manual review ? the model is uncertain or depends on weak signals.
- [ ] Retrain ? the behavior appears stable but undesirable.
- [ ] Rollback or pause ? the issue may be tied to a recent deployment or data change.

Record the decision and rationale.

---

9) Document the investigation

- [ ] Save the prediction ID, timestamp, model version, and feature snapshot.
- [ ] Save the explanation output and any plots or ranked feature lists.

- [ ] Record the expected drivers versus the observed drivers.
- [ ] Note any drift, calibration, or data-quality anomalies.
- [ ] Record the final decision and who approved it.
- [ ] Create a follow-up task if the issue is unresolved.

---

10) Watch for common failure patterns

Use this quick reference during triage.

Signal in the explanation	Possible meaning	Suggested action
Missing-value feature dominates	Upstream data issue or brittle fallback behavior	Check the pipeline and consider manual review
Proxy or sensitive feature becomes important	Policy risk or hidden bias	Investigate immediately and review feature governance
Explanation changes after a release	Release regression or schema mismatch	Compare with previous model version and rollback if ne
Same unusual pattern appears across many cases	Systemic behavior change	Check drift, retraining needs, and segment impact
Explanation looks unstable across similar records	Model fragility or insufficient signal quality	Validate calibration and consider r

---

11) Minimum production checklist before trusting explanations

- [ ] The explanation method is documented and repeatable.
- [ ] The model and feature pipeline versions are known.
- [ ] Input drift and calibration are being monitored.
- [ ] Similar cases show similar explanation patterns.
- [ ] The explanation matches domain expectations.
- [ ] The output is safe to automate or has a fallback path.

- [ ] The team has a clear escalation path for suspicious results.

---

12) Fast decision guide

If the explanation is:

- Consistent and expected ? accept the output.
- Consistent but undesirable ? investigate retraining or policy changes.
- Inconsistent or unstable ? inspect the pipeline and nearby examples.
- Dependent on risky inputs ? block or route to manual review.
- Changed after deployment ? compare release history and consider rollback.

Notes for teams

- Explanations are diagnostic evidence, not proof of correctness.
- The best results come from combining local explanations, global patterns, and operational checks.
- Explainability is most valuable when it helps you decide whether to trust an output, not just when it helps you describe it.

Quick fill-in worksheet

Model name: _____

Model version: _____

Prediction reviewed: _____

Date/time: _____

Why it looked suspicious:

- _____
- _____
- _____

Top explanation drivers:

- _____
- _____
- _____

Expected drivers:

- _____
- _____

Health checks reviewed:

- [ ] Drift
- [ ] Calibration
- [ ] Data quality
- [ ] Pipeline integrity
- [ ] Release history

Decision:

- [ ] Accept
- [ ] Investigate
- [ ] Block
- [ ] Manual review
- [ ] Retrain
- [ ] Rollback or pause

Owner: _____

Follow-up action: _____