

True Model Failure

It's essential to evaluate if the seeded code response contains a true failure. A true failure is defined when the seeded code response fails at least one valid unit test.

The model may or may not pass/fail **FAULTY** unit tests – those are not considered when deciding a yes/no for true model failure.

Unit Test Glossary

| Unit Test Type | Description | Notes |
|--------------------------|---|--|
| Valid/Strong Unit Test | A unit test that tests important/core functionality and asserts a meaningful value. | If the model fails at least one valid/strong unit test, marks YES for model failure. Note that you can ALWAYS mark yes if the model fails one valid/strong unit test. <ul style="list-style-type: none">Explanation: Any faulty unit tests that the model failed/passed must be remove |
| Weak Unit Test | A unit test that tests for invalid input and expects an error to be raised, or e.g, an empty list to be returned. A unit test that has different expectations than the model response due to a difference in implementation. In essence, the test fails not because the solution is incorrect, but because the test expectations are based on a different but also valid approach. | If the model only fails a weak unit test, mark NO for true model failure. |
| Faulty/Invalid Unit Test | A unit test that asserts an incorrect/faulty value | If the model only fails a faulty unit test, mark NO for faulty unit tests. |

Model Failure - Core Rules

| # | Rule Statement | Condition | Notes |
|---|---------------------------------------|--|---|
| 1 | At least one valid unit test | Seeded Code Response fails at least one VALID unit test ALREADY present in the seeded unit tests before any modifications. The first sphere engine/container used to test the seeded model response for failure is read-only. | If the model response contains issues that are not tested in the unit tests it doesn't count towards a model failure. |
| 2 | Criteria for a valid unit test | That single unit test must be relevant for the problem, test a core functionality, and assert the correct value. | |
| 3 | Exclude faulty unit tests | Faulty unit tests do not count as model failures | |

Model Failure - Edge Case

The model may **FAIL** a set of multiple unit tests where some unit tests are actually **FAULTY** (they test a faulty thing or assert the wrong value) and other unit tests are **VALID**.

Clarification:

- Unit tests {A, B, C} are provided, and the model fails all three unit tests.
- You identify that unit tests {B, C} are faulty (they test the wrong thing).
- The model's code also fails the unit test {A}, which is valid.
- Then this IS a **True Model Failure**, as there is at least one valid unit test that the model failed.

You will have the opportunity to remove/fix faulty unit tests in a later stage (in the second workspace container).

When To Mark No For True Model Failure

| # | Scenario | Clarification | Notes |
|---|--|---|--|
| 1 | The Failing Unit Tests Are Invalid Unit Tests | If unit tests {A, B, C} are provided, and the model only fails model failure A, but ultimately A is not a valid unit test, you mark no for model failure. | |
| 2 | The Failing Unit Tests Are Weak Unit Tests | Many unit tests in our pipeline test for the robustness of the program during e.g. invalid inputs/out of range parameters for which the unit tests e.g. expect errors to be thrown or perhaps an empty list, null, etc. | The Aider benchmark doesn't consider those model failures, nor does our dataset. You can only mark yes for |

| | | | |
|---|---|--|------------------|
| | | <p>The model might fail such a unit test. This is considered too weak and doesn't justify a "yes" for model failure validation.</p> <p>Even if the prompt specifies the out-of-bound / invalid ranges, we cannot mark yes for true failure if the program doesn't throw an error for e.g. invalid input.</p> | valid unit tests |
| 3 | The Failing Units Are both Weak and/or Invalid | N/A | |

Detailed Workflow

| # | Step | Description | Notes |
|---|--------------------------------------|---|--|
| 1 | Open The Sphere Engine | Access the first read-only workspace containing the preloaded seeded code response and preloaded unit test suite. Press the blue [Open IDE] button to open the Sphere Engine. | You'll be shown two sphere engines in total. This regards the first sphere engine, which is read-only for model failure validation purposes. |
| 2 | Execute & Analyze Failure | Run the seeded code response against the unit tests to validate if it contains a true model failure. Analyze the failure output to confirm it fails at least one valid unit test. | Any faulty unit tests do not count towards a model failure. |
| 3 | Task Continuation | If no model failure is established, the task is cut short and ends early. If true model failure is confirmed, proceed to the next stage. | N/A |

Model Failure Justification Writing

If there is a true model failure, you briefly write a justification of two sentences explaining on a high level why the model contains a true failure. If there are many failing unit tests, you mention the 2/3 most important failing unit tests.

Conclusion

The essence is that you distinguish between valid and faulty unit tests first. Then you determine if the model indeed fails a faulty unit test while ignoring faulty unit tests.

