Attachment E – Test, implement & iterate

Agencies need to build a system to monitor and log their system’s predictions, actions and the resulting metrics. This monitoring system needs to display the current status of the performance metrics and alerts the AI system administrator if the performance baseline is not met, constraints behaviour is detected or if unfamiliar data is encountered.

Transition from test environment to the real scenarios

Agencies need to test the chosen operating parameters and the performance of the system against the performance metrics.

Early tests will be conducted in an environment that is as close as possible to the real deployment scenario. It may be prudent to start the AI-enabled system interacting with simulated data if available, or with very close human supervision and limitations on the scope of allowable actions.

Agencies can scale up these tests to more realistic scenarios with fewer limits on the system’s actions as the system develops and as it proves efficacy against the performance metrics. While conducting testing and trialling, consider:

- How is the system tracking with respect to its performance baselines (previous systems, minimum satisfactory requirements, etc)?
- Is the data gathered during these tests from the same distribution used to construct the system models (models of state and consequences)? Or are these systems experiencing a distributional (covariate/concept) shift?
- Have there been detected any unintended consequences from the system’s actions?
- Are there any objectives or concerns not been captured?
- Are the metrics failing to capture the intended consideration?
- Have any assumptions been violated regarding the data or models?
- Are engineering or design errors causing unintended behaviours?

Project teams should attempt to diagnose and rectify any of these or other issues that have arisen by repeating the design and test process. Project teams should also aim to test code against sample data frequently to ensure intended outcomes are being achieved at each stage of the testing process.

Iterate the system
Lessons learned from systems testing is important to integrate into every stage of the design process.

Each iteration does not have to occur at the same scale, but it is important to continue to involve decision-makers, experts and data scientists in this process. At each iteration the system must be re-tested; being mindful that changes to fix one issue may create another. It is also important to consider the following:

- Did the team put measures in place to ensure the system is being tested on the most up-to-date data?
- Have the team identified ways in which the system can fail and put in place procedures to flag these failures?
- What type of errors can the team not observe (such as missed opportunities/false negatives)?
- Can the system be audited and tested by an independent third-party? Ask other people to analyse the system, who may see potential problems with a different perspective such as data scientists, engineers or domain experts, both in the organisation or externally.
- What happens if a malicious actor varies the training data or parameters - can the team detect this?
- Testing can still do real harm if the system is making consequential decisions. Consider and account for this possibility even if any human testers are willing volunteers.
- Test the system's infrastructure separately from any inference algorithms. For example, pre-processing stages like standardising continuous data, removing stop words from text etc. These should be unit tested if feasible. This may also be a good stage to develop tests for unexpected inputs during operation
- Look through the system logs for outliers in the data the system was fed, or actions that resulted in unexpected effects (i.e. differing from predictions etc.)
- Break down the errors and outcomes from the system into different demographic groups to see how they are distributed, even if the team has already identified a 'type' of fairness to satisfy.
- In early testing and deployment stages, keep old systems online for comparison purposes if possible. This may help catch errors in the new system. Alternatively, keep the existing system in production and use it to check the new system's actions.
- Be mindful of directly adding data that has been derived from testing into a training data set. The testing may have introduced selection bias that must be accounted for explicitly.