



THE RIGHT TESTING STRATEGY FOR AI SYSTEMS

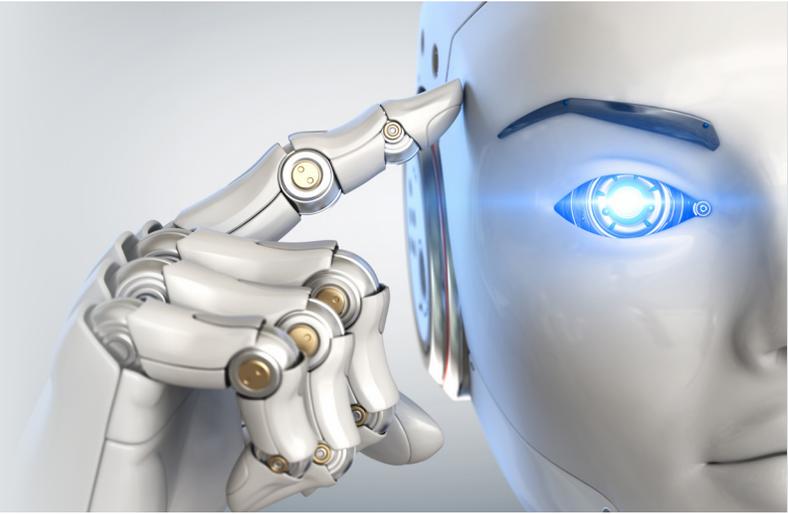
AN INFOSYS VIEWPOINT

VENKATESH IYENGAR,
AVP - Group Practice Engagement Manager, Infosys

SUNDARESA SUBRAMANIAN G,
Practice Engagement Manager, Infosys

Abstract

Over the years, organizations have invested significantly in optimizing their testing processes to ensure continuous releases of high-quality software. When it comes to artificial intelligence, however, testing is more challenging owing to the complexity of AI. Thus, organizations need a different approach to test their AI frameworks and systems to ensure that these meet the desired goals. This paper examines some key failure points in AI frameworks. It also outlines how these failures can be avoided using four main use cases that are critical to ensuring a well-functioning AI system.



Introduction

Experts in nearly every field are in a race to discover how to replicate brain functions – wholly or partially. In fact, by 2025, the value of the artificial intelligence (AI) market will surpass US \$100 billion¹. For corporate organizations, investments in AI are made with the goal of amplifying the human potential, improving efficiency and optimizing processes. However, it is important to be aware that AI too is prone to error owing to its complexity. Let us first understand what makes AI systems different from traditional software systems:

S.No	Software systems	AI systems
1	Features – Software is deterministic, i.e., it is pre-programmed to provide a specific output based on a given set of inputs	Features – Artificial intelligence/machine learning (AI/ML) is non-deterministic, i.e., the algorithm can behave differently for different runs
2	Accuracy – Accurate software depends on the skill of the programmer and is deemed successful if it produces an output in accordance with its design	Accuracy – Accuracy of AI algorithms depends on the training set and data inputs
3	Programming – All software functions are designed based on if-then and for loops to convert input data to output data	Programming – Different input and output combinations are fed to the machine based on which it learns and defines the function
4	Errors – When software encounters an error, remediation depends on human intelligence or a coded exit function	Errors – AI systems have self-healing capabilities whereby they resume operations after handling exceptions/errors

The hierarchy of evolution of AI

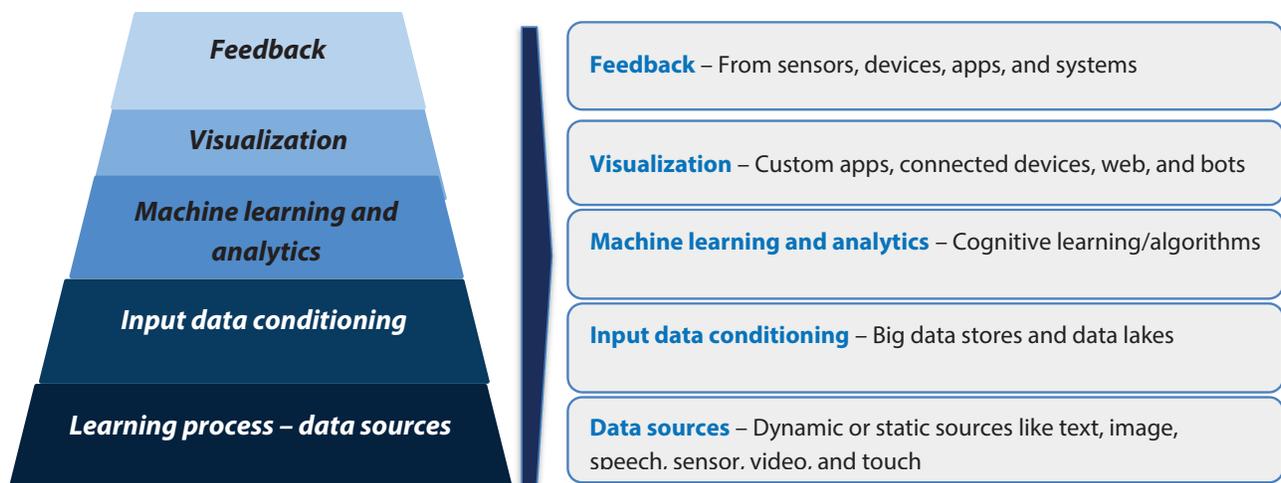


Fig 1: Hierarchy of AI's evolution and techniques

The above figure shows the sequential stages of AI algorithms. While each stage is necessary for successful AI programs, there are some typical failure points that exist within each stage. These must be carefully identified using the right testing technique as shown in the table below:

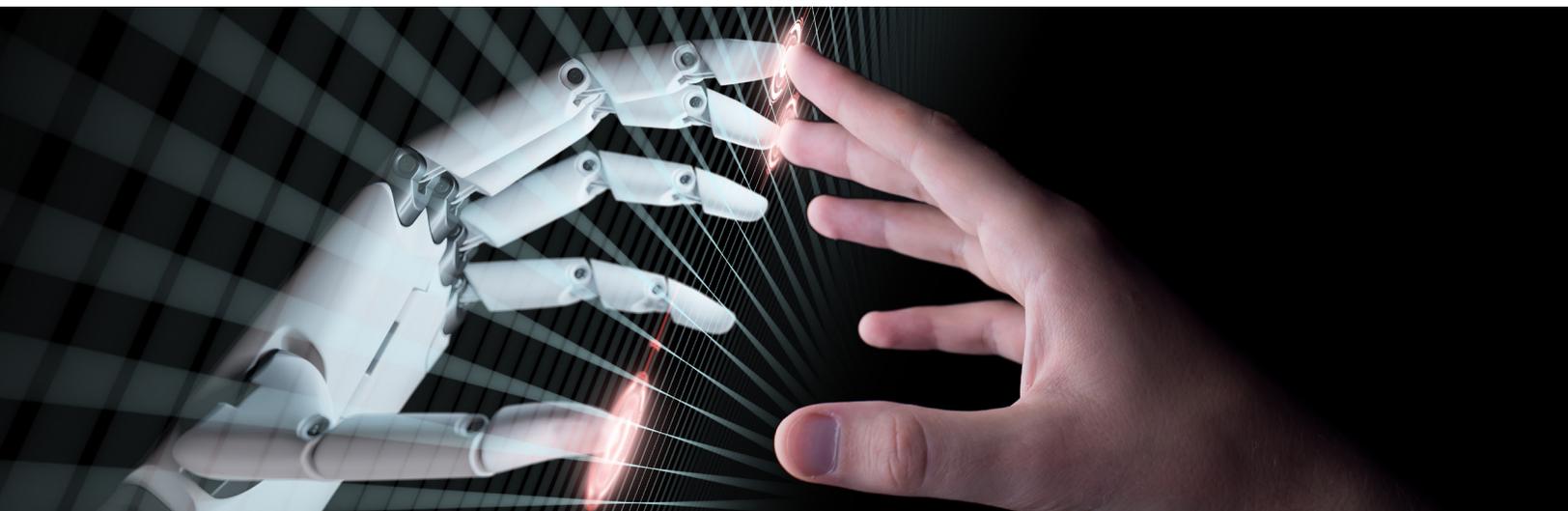
S.No	Evolution stage in AI	Typical failure points	How can they be detected in testing
1	Data sources – Dynamic or static sources	<ul style="list-style-type: none"> • Issues of correctness, completeness and appropriateness of source data quality and formatting • Variety and velocity of dynamic data resulting in errors • Heterogeneous data sources 	<ul style="list-style-type: none"> • Automated data quality checks • Ability to handle heterogeneous data during comparison • Data transformation testing • Sampling and aggregate strategies
2	Input data conditioning – Big data stores and data lakes	<ul style="list-style-type: none"> • Incorrect data load rules and data duplicates • Data nodes partition failure • Truncated data and data drops 	<ul style="list-style-type: none"> • Data ingestion testing • Knowledge of development model and codes • Understanding data needed for testing • Ability to subset and create test data sets
3	ML and analytics – Cognitive learning/ algorithms	<ul style="list-style-type: none"> • Determining how data is split for training and testing • Out-of-sample errors like new behavior in previously unseen data sets • Failure to understand data relationships between entities and tables 	<ul style="list-style-type: none"> • Algorithm testing • System testing • Regression testing
4	Visualization – Custom apps, connected devices, web, and bots	<ul style="list-style-type: none"> • Incorrectly coded rules in custom applications resulting in data issues • Formatting and data reconciliation issues between reports and the back-end • Communication failure in middleware systems/APIs resulting in disconnected data communication and visualization 	<ul style="list-style-type: none"> • API testing • End-to-end functional testing and automation • Testing of analytical models • Reconciliation with development models
5	Feedback – From sensors, devices, apps, and systems	<ul style="list-style-type: none"> • Incorrectly coded rules in custom applications resulting in data issues • Propagation of false positives at the feedback stage resulting in incorrect predictions 	<ul style="list-style-type: none"> • Optical character recognition (OCR) testing • Speech, image and natural language processing (NLP) testing • RPA testing • Chatbot testing frameworks

The right testing strategy for AI systems

Given the fact that there are several failure points, the test strategy for any AI system must be carefully structured to mitigate risk of failure. To begin with, organizations must first understand the various stages in an AI framework as shown in Fig 1. With this understanding, they will be able to

define a comprehensive test strategy with specific testing techniques across the entire framework. Here are four key AI use cases that must be tested to ensure proper AI system functioning:

- Testing standalone cognitive features such as natural language processing (NLP), speech recognition, image recognition, and optical character recognition (OCR)
- Testing AI platforms such as IBM Watson, Infosys NIA, Azure Machine Learning Studio, Microsoft Oxford, and Google DeepMind
- Testing ML-based analytical models
- Testing AI-powered solutions such as virtual assistants and robotic process automation (RPA)



Use case 1: Testing standalone cognitive features

 <p>Natural language processing (NLP)</p>	<ul style="list-style-type: none"> • Test for 'precision' return of the keyword, i.e., a fraction of relevant instances among the total retrieved instances of NLP • Test for 'recall', i.e., a fraction of retrieved instances over total number of retrieved instances available • Test for true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs). Ensure that FPs and FNs are within the defined error/fallout range
 <p>Speech recognition inputs</p>	<ul style="list-style-type: none"> • Conduct basic testing of the speech recognition software to see if the system recognizes speech inputs • Test for pattern recognition to determine if the system can identify when a unique phrase is repeated several times in a known accent and whether it can identify the same phrase when it is repeated in a different accent • Test deep learning, the ability to differentiate between 'New York' and 'Newark' • Test how speech translates to response. For example, a query of "Find me a place I can drink coffee" should not generate a response with coffee shops and driving directions. Instead, it should point to a public place or park where one can enjoy his/her coffee
 <p>Image recognition</p>	<ul style="list-style-type: none"> • Test the image recognition algorithm through basic forms and features • Test supervised learning by distorting or blurring the image to determine the extent of recognition by the algorithm • Test pattern recognition by replacing cartoons with the real image like showing a real dog instead of a cartoon dog • Test deep learning using scenarios to see if the system can find a portion of an object in a larger image canvas and complete a specific action
 <p>Optical character recognition</p>	<ul style="list-style-type: none"> • Test OCR and optical word recognition (OWR) basics by using character or word inputs for the system to recognize • Test supervised learning to see if the system can recognize characters or words from printed, written or cursive scripts • Test deep learning, i.e., whether the system can recognize characters or words from skewed, speckled or binarized (when color is converted to grayscale) documents • Test constrained outputs by introducing a new word in a document that already has a defined lexicon with permitted words

Use case 2: Testing AI platforms

Testing any platform that hosts an AI framework is complex. Typically, it follows many of the steps used during functional testing.

Data source and conditioning testing

- Verify the quality of data from various systems – data correctness, completeness and appropriateness along with format checks, data lineage checks and pattern analysis
- Verify transformation rules and logic applied on raw data to get the desired output format. The testing methodology/automation framework should function irrespective of the nature of data – tables, flat files or big data
- Verify that the output queries or programs provide the intended data output
- Test for positive and negative scenarios

Algorithm testing

- Split input data for learning and for the algorithm
- If the algorithm uses ambiguous datasets, i.e., the output for a single input is not known, the software should be tested by feeding a set of inputs and checking if the output is related. Such relationships must be soundly established to ensure that algorithms do not have defects
- Check the cumulative accuracy of hits (TPs and TNs) over misses (FPs and FNs)

API integration

- Verify input request and response from each application programming interface (API)
- Verify request response pairs
- Test communication between components – input and response returned as well as response format and correctness
- Conduct integration testing of API and algorithms and verify reconciliation/visualization of output

System/regression testing

- Conduct end-to-end implementation testing for specific use cases, i.e., provide an input, verify data ingestion and quality, test the algorithms, verify communication through the API layer, and reconcile the final output on the data visualization platform with expected output
- Check for system security, i.e., static and dynamic security testing
- Conduct user interface and regression testing of the systems

Use case 3: Testing ML-based analytical models

Organizations build analytical models for three main purposes as shown in Fig 2

Fig 2: Types and purposes of analytical models



Fig 2: Types and purposes of analytical models

The validation strategy used while testing the analytical model involves the following three steps:

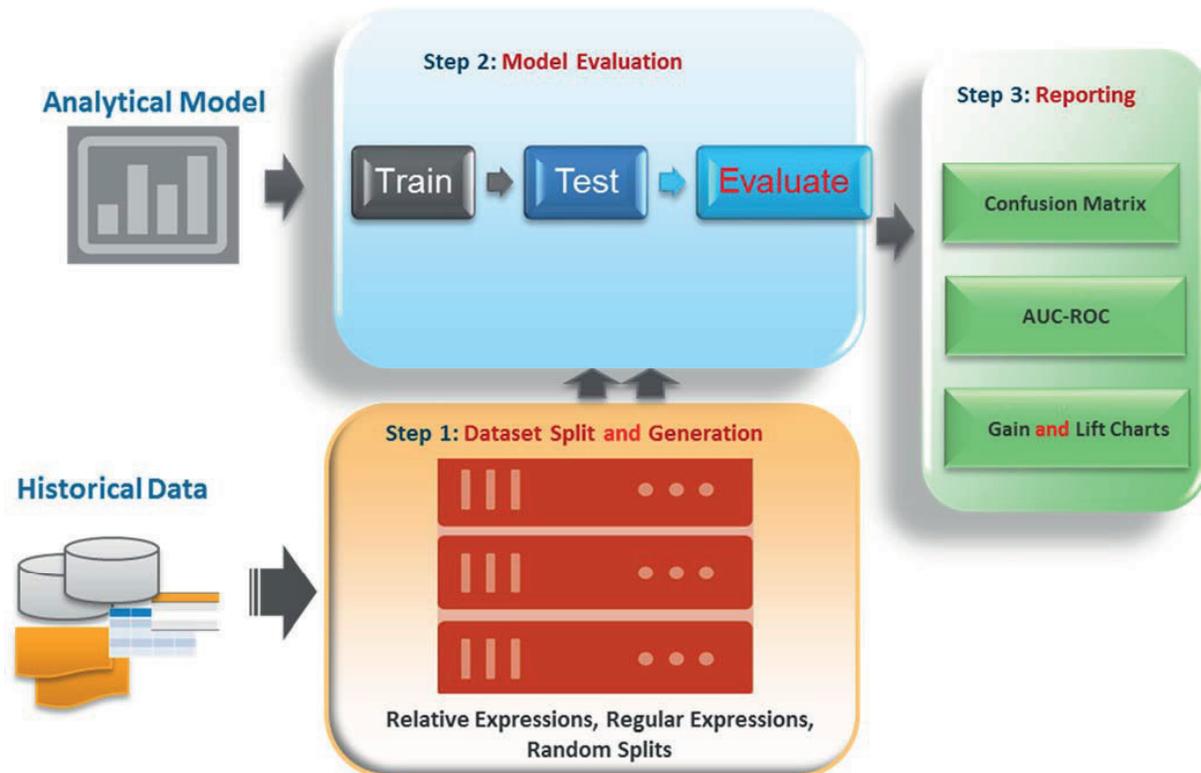
- Split the historical data into 'test' and 'train' datasets
- Train and test the model based on generated datasets
- Report the accuracy of model for the various generated scenarios



Fig 3: Testing analytical models

Organizations build analytical models for three main purposes as shown in Fig 2

Fig 2: Types and purposes of analytical models



While testing a model, it is critical to do the following to ensure success:

- Devise the right strategy to split and subset historical dataset using deep knowledge of development model and code to understand how it works on data
- Model the end-to-end evaluation strategy to train and recreate model in test environments with associated components
- Customize test automation to optimize testing throughput and predictability by leveraging customized solutions to split the dataset, evaluate the model and enable reporting

Use case 4: Testing of AI-powered solutions

 <p>Chatbot testing framework</p>	<ul style="list-style-type: none"> • Test the chatbot framework using semantically equivalent sentences and create an automated library for this purpose • Maintain configurations of basic and advanced semantically equivalent sentences with formal and informal tones and complex words • Automate end-to-end scenario (requesting chatbot, getting a response and validating the response action with accepted output) • Generate automated scripts in Python for execution
 <p>RPA testing framework</p>	<ul style="list-style-type: none"> • Use open source automation or functional testing tools (Selenium, Sikuli, Robot Class, AutoIT) for multiple applications • Use flexible test scripts with the ability to switch between machine language programming (where required as an input to the robot) and high-level language for functional automation • Use a combination of pattern, text, voice, image, and optical character recognition testing techniques with functional automation for true end-to-end testing of applications

Conclusion

AI frameworks typically follow 5 stages – learning from various data sources, input data conditioning, machine learning and analytics, visualization, and feedback. Each stage has specific failure points that can be identified using several techniques. Thus, when testing the AI systems, QA departments must clearly define the test strategy by considering the various challenges and failure points across all stages. Some of the important testing use cases to be considered are testing standalone cognitive features, AI platforms, ML-based analytical models, and AI-powered solutions. Such a comprehensive testing strategy will help organizations streamline their AI frameworks and minimize failures, thereby improving output quality and accuracy.



For more information, contact askus@infosys.com



© 2018 Infosys Limited, Bengaluru, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.