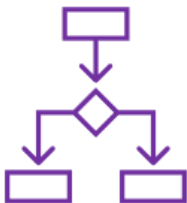


# How to Perform Testing of AI/ML-based Systems with Test Case Designer

Artificial Intelligence is transforming the technology landscape of the digital age. The world is moving towards the adoption of AI-powered smart systems which will increase exponentially over the next few years. While we see the advancements, the key challenge would be the testing of Artificial Intelligence /Machine Learning (AI/ML)-based systems.

There are 3 major challenges in testing AI systems:



Different outputs



Bias



Lack of data

Test Case Designer cannot do much about the first one, so **we will talk primarily about the benefits related to data availability and quality**. After all, [80 % of a scientist's time is spent preparing the training dataset](#).

We will use the phase classification from [Forbes](#):

**TCD applicability to QA in different phases of AI development**

AI algorithm itself	Low
Hyperparameter configuration	Low
Training, validation, and test data	Medium
Integration of the AI system with other workflow elements	High

The rest of the article covers phases 2-4 in more detail. Regarding phase 1, significant customization of the algorithm code is not as prominent and, to borrow the quote from Ron Schmelzer, "There's just one way to do the math!" so the core value proposition of Test Case Designer to explore possible combinations is not as relevant (i.e., low applicability due to the "linear" nature of operations).

- [Phase 2](#)
- [Phase 3](#)
- [Phase 4](#)
- [Conclusion](#)

## Phase 2

The general idea is to include each hyperparameter in the TCD model, breaking down the value lists based on the thresholds derived from theory or practical experience.

AI/ML - Hyperparameters

Learning rate (5)	Preset  1	0.0001 - 0.005	0.01 - 0.04	0.06 - 0.1	0.2 - 1
Regularization (4)	none	L1	L2	dropout	
Batch size (4)	10-100  3	200-500  3	600-1000  3	1100 - 5000  3	
Depth of layers (4)	1	2	3-5	6-10	

[Reference](#)

The specific ranges and value expansions on the screenshot are for example purposes but should sufficiently communicate the “identity” of the approach. Further, constraints and risk-based algorithm settings can be used to control the desired interactions:

Search...

23 scenarios and 108 2-way interactions

#	Learning rate	Regularization	Batch size	Depth of layers
1	0.05 - Preset	none	10 - 10-100	1
2	0.005	L1	200 - 200-500	1
3	0.04	L2	600 - 600-1000	1
4	0.08	dropout	1100 - 1100 - 5000	1
5	0.05 - Preset	L2	500 - 200-500	2
6	0.05 - Preset	L1	1000 - 600-1000	5
7	0.0020	dropout	50 - 10-100	2
8	0.0001	L2	5000 - 1100 - 5000	3
9	0.03	L1	2500 - 1100 - 5000	2
10	0.01	dropout	350 - 200-500	9
11	0.06	L2	100 - 10-100	6

Or you could use the 4-way setting to get the full scope of possible combinations.

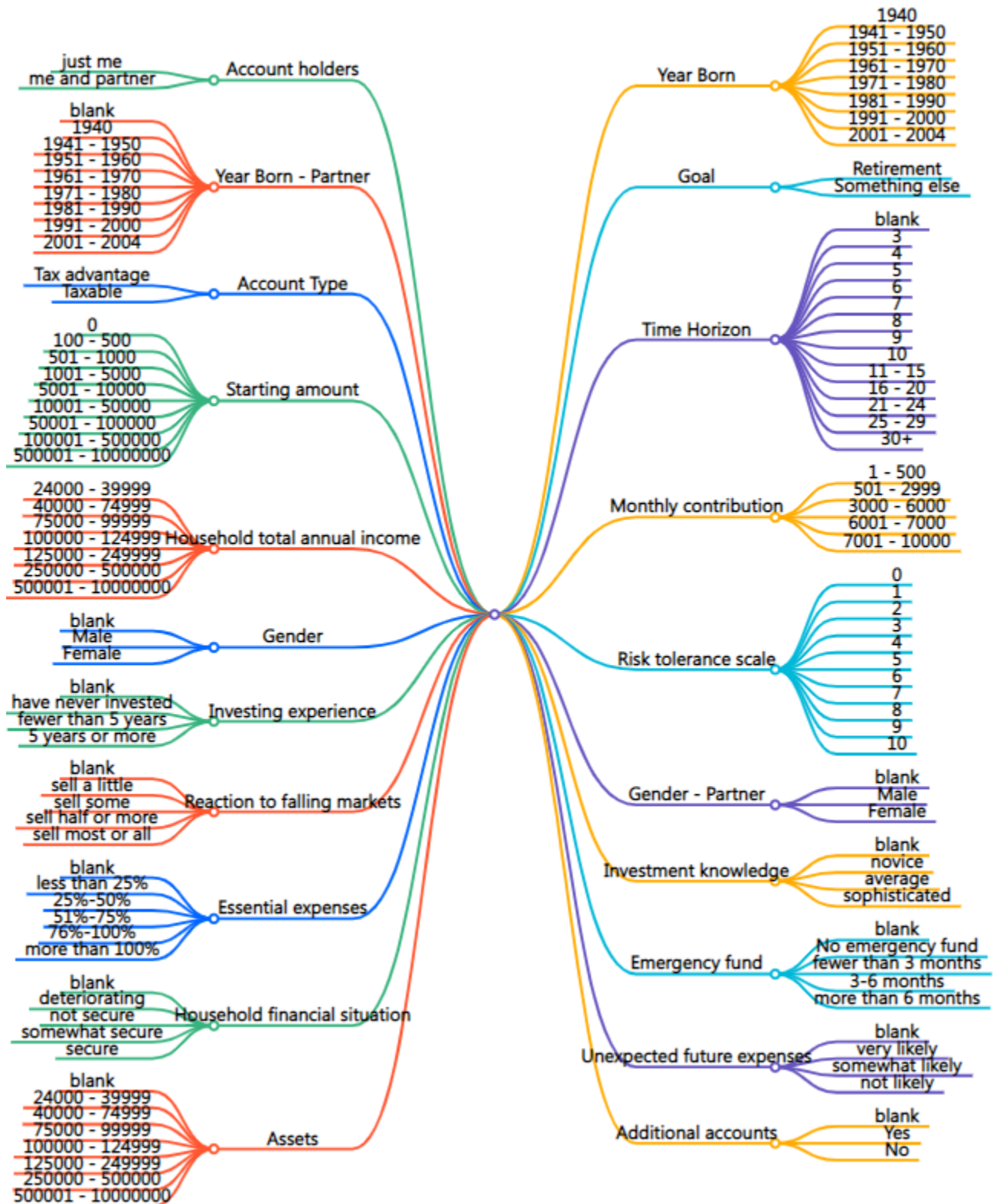
**Strength:** Systematic approach to identifying relevant hyperparameter configuration profiles.

**Weakness:** May explore the profiles with too many changes at a time or require numerous constraints to limit the scope.

### Phase 3

Robo-advisors are a popular application of AI/ML systems in finance. They use online questionnaires that obtain information about the client's degree of risk aversion, financial status, and desired return on investment. For this example, we will use [Fidelity GO](#).

To build the corresponding model in TCD, you will need to forget (temporarily) some of the lessons about parameter & value definitions given different objectives. Instead of optimizing the scenario count, the goal of this data set is to become a representative sample of the real world and eliminate as much human bias as possible. This means not just data quality but also completeness.



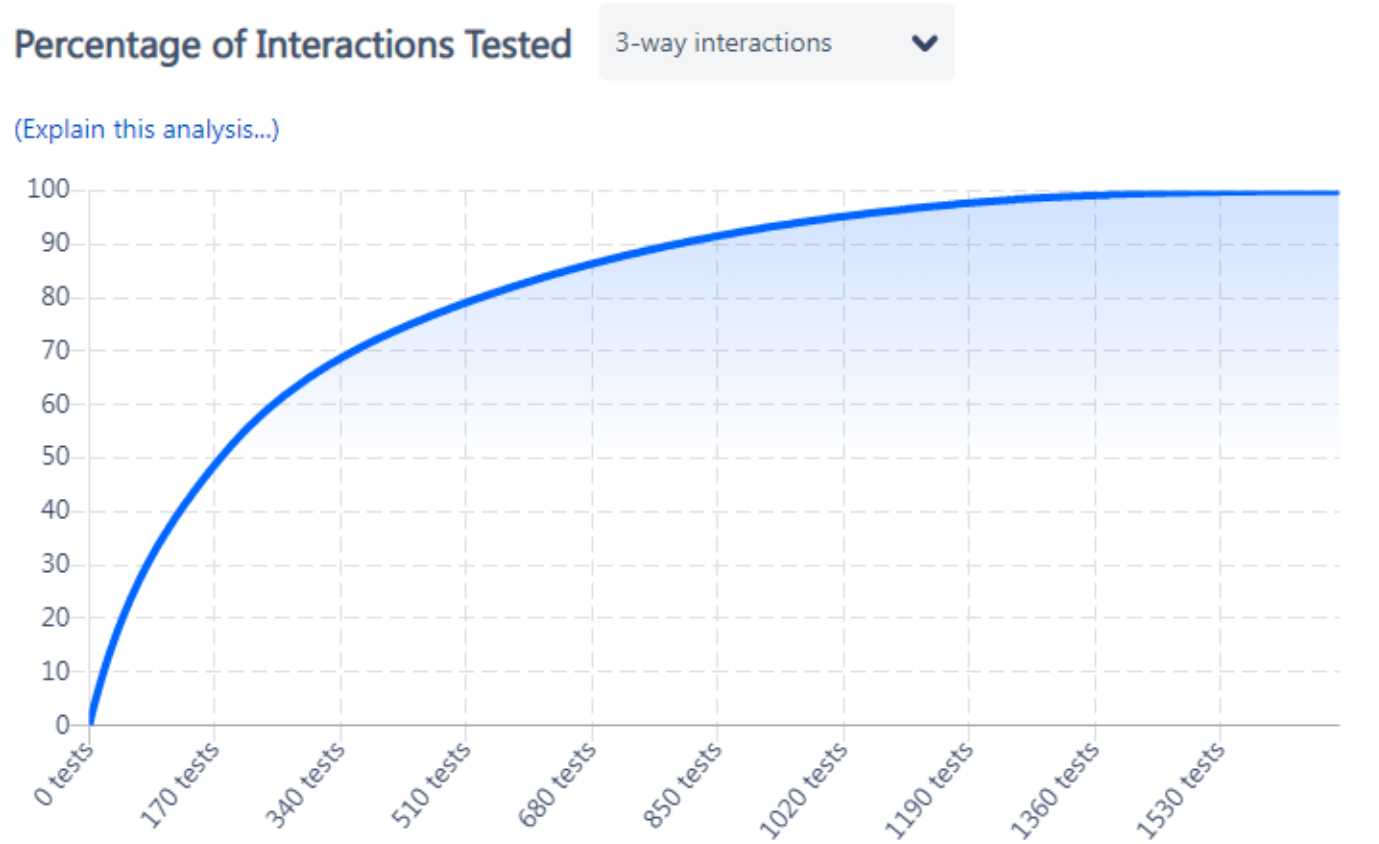
Such a model would include all parameters regardless of the impact on the business outcome and utilize lengthy, highly detailed value lists (often more than 10 per parameter). To distinguish between the review and the "consumption" formats, value names or value expansions can be adjusted accordingly (i.e., the value name can be "sell some" for communication to stakeholders while the expansion can be "3" given the data encoding).

When it comes to the TCD algorithm strength selection, the highest available option is typically the most desired one (see the caveat in the "Weakness" below):

Search... Showing 500 of 1,691 scenarios. Export to see them all. 229,691 3-way interactions ⓘ

#	Account holders	Year Born	Year Born - Partner	Goal	Account Type	Time Horizon	Starting amount
1	just me	1940	blank	Retirement	Tax advantage	blank	0
2	me and partner	1941	1940	Something else	Taxable	3	0
3	me and partner	1951	1941	Something else	Tax advantage	4	0
4	me and partner	1961	1951	Retirement	Taxable	5	0
5	me and partner	1971	1961	Retirement	Taxable	6	0

When this approach is used for generating the validation + test data sets, the TCD Analysis capabilities (in addition to standard statistical methods) can be used to evaluate the diversity of the split:



**Strength:** data sets are intelligently built to test all relevant permutations and combinations to deduce the efficiency of trained models while minimizing bias. Further, the regeneration of such data sets is much faster and easier.

### Weakness:

1. The current scope limitation is 5000 scenarios per Test Case Designer model which may not be sufficient for training or even validation purposes of some AI systems.

As a side note, while “all possible permutations” is a nice goal, it is often not the optimal one – even for representative purposes, having 289,700,167,680,000 scenarios (which is the possible total for the model above) will not be realistic to perform training on. So, the “right” answer still requires balance and prioritization.

2. Despite certain workarounds, programmatic handling of complex expected results would likely require complementary manual effort.

3. The approach depends on the overall ability to leverage synthetic data instead of production copies which may or may not be feasible in your environment.

## Phase 4









This phase is the closest to TCD’s “bread and butter.” The model would serve a dual purpose – 1) smoke testing of the AI; 2) integration testing of how it is operationalized.

```
1 System A - AI [ ***** ]
2 Account holders [ just me , me and partner ]
3 Year Born [ 1940 , 1941 - 1970 , 1971 - 1990 , 1991 - 2004 ]
4 Year Born - Partner [ blank , 1940 , 1941 - 1970 , 1971 - 1990 , 1991 - 2004 ]
5 Goal [ Retirement , Something else ]
6 Account Type [ Tax advantage , Taxable ]
7 Time Horizon [ blank , 3 - 7 , 8 - 10 , 11 - 20 , 21 - 29 , 30+ ]
8 Starting amount [ 0 , 100 - 5000 , 5001 - 50000 , 50001 - 500000 , 500001 - 10000000 ]
9 Monthly contribution [ 1 - 500 , 501 - 6000 , 6001 - 7000 , 7001 - 10000 ]
10 Household total annual income [ 24000 - 74999 , 75000 - 124999 , 125000 - 500000 , 500001 - 10000000 ]
11 Risk tolerance scale [ 0 , 1 - 4 , 5 , 6 - 9 , 10 ]
12 Gender [ blank , Male , Female ]
13 Gender - Partner [ blank , Male , Female ]
14 Investing experience [ blank , not blank ]
15 Investment knowledge [ blank , not blank ]
16 Reaction to falling markets [ blank , not blank ]
17 Emergency fund [ blank , not blank ]
18 Essential expenses [ blank , not blank ]
19 Unexpected future expenses [ blank , very likely , somewhat likely , not likely ]
20 Household financial situation [ blank , deteriorating , not secure , somewhat secure , secure ]
21 Additional accounts [ blank , Yes , No ]
22 Assets [ blank , 24000 - 99999 , 100000 - 500000 , 500001 - 10000000 ]
23 System B [ ***** ]
24 Asset1 Account Type [ Retirement IRA , Roth IRA , Traditional IRA , Taxable , Rollover ]
```

Given the execution setup, you would likely have to keep all the factors consumed by the AI system but, for this phase, reduce the number of values based on the importance (both business- and algorithm-wise).

Scenario volume would still be largely driven by the “standard” integration priorities (i.e., key parameters affecting multiple systems). Still, the number of values and/or the average mixed-strength dropdown selection would be higher than typical.

477 scenarios and 13,812 Mixed-strength interactions 

	3-way 	3-way 	2-way 	3-way 	3-way 	2-way 	3-way 	2-way 
	Account holders	Year Born	Year Born - Partner	Goal	Account Type	Time Horizon	Starting amount	Mont
	just me	1940	blank	Retirement	Tax advantage	blank	0	1
ie	me and partner	1941	1940	Something else	Taxable	3	100	501
ie	just me	1971	1941	Something else	Tax advantage	8	5001	6001

Focusing on the “just right” level of detail for the high-significance factors will guarantee the optimal dataset for sustainable AI testing.

**Strength:**

1. Test Case Designer at its best with the thoroughness, speed, and efficiency benefits.
2. Ability to quickly reuse model elements from Phase 3 and models related to other systems (e.g., the old version of the non-AI advisor for systems B and C).
3. Higher control over the variety of data at the integration points and the workflow as a whole.

**Weakness:** Similar to Phase 3 but usually more manageable given the difference in goals (volume in P3 vs. integration in P4).

## Conclusion

To summarize, the applicability level by phase is repeated below:

AI algorithm itself	Low
Hyperparameter configuration	Low
Training, validation, and test data	Medium
Integration of the AI system with other workflow elements	High

From another perspective, using this stage classification from [Infosys](#), Test Case Designer can deliver the most significant benefits in the highlighted testing areas:

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"> <li>Issues of correctness, completeness and appropriateness of source data quality and formatting</li> <li>Variety and velocity of dynamic data resulting in errors</li> <li>Heterogeneous data sources</li> </ul>	<ul style="list-style-type: none"> <li>Automated data quality checks</li> <li>Ability to handle heterogeneous data during comparison</li> <li>Data transformation testing</li> <li>Sampling and aggregate strategies</li> </ul>
2	Input data conditioning – Big data stores and data lakes	<ul style="list-style-type: none"> <li>Incorrect data load rules and data duplicates</li> <li>Data nodes partition failure</li> <li>Truncated data and data drops</li> </ul>	<ul style="list-style-type: none"> <li>Data ingestion testing</li> <li>Knowledge of development model and codes</li> <li>Understanding data needed for testing</li> <li>Ability to subset and create test data sets</li> </ul>
3	ML and analytics – Cognitive learning/ algorithms	<ul style="list-style-type: none"> <li>Determining how data is split for training and testing</li> <li>Out-of-sample errors like new behavior in previously unseen data sets</li> <li>Failure to understand data relationships between entities and tables</li> </ul>	<ul style="list-style-type: none"> <li>Algorithm testing</li> <li>System testing</li> <li>Regression testing</li> </ul>
4	Visualization – Custom apps, connected devices, web, and bots	<ul style="list-style-type: none"> <li>Incorrectly coded rules in custom applications resulting in data issues</li> <li>Formatting and data reconciliation issues between reports and the back-end</li> <li>Communication failure in middleware systems/APIs resulting in disconnected data communication and visualization</li> </ul>	<ul style="list-style-type: none"> <li>API testing</li> <li>End-to-end functional testing and automation</li> <li>Testing of analytical models</li> <li>Reconciliation with development models</li> </ul>
5	Feedback – From sensors, devices, apps, and systems	<ul style="list-style-type: none"> <li>Incorrectly coded rules in custom applications resulting in data issues</li> <li>Propagation of false positives at the feedback stage resulting in incorrect predictions</li> </ul>	<ul style="list-style-type: none"> <li>Optical character recognition (OCR) testing</li> <li>Speech, image and natural language processing (NLP) testing</li> <li>RPA testing</li> <li>Chatbot testing frameworks</li> </ul>

Given the typical scale of AI projects, possible inputs and outputs combinations will be almost indefinitely high. Moreover, the techniques used to implement self-learning elements are very complex.

Therefore, fully testing these kinds of applications would not be feasible. To overcome this challenge, we need to think more critically about a systematic, risk-based test design approach, such as the one that TCD facilitates.