



AI Introduction for Testers Micro-Credential Syllabus

Copyright Notice

Copyright AT*SQA, All Rights Reserved



Table of Contents

AI Introduction for Testers

3	Overview
7	Introduction
8	Machine Learning Models
10	Neural Networks
13	Machine Learning Workflow
16	Model Training
20	Machine Learning Algorithms
24	Data Considerations
26	Functional Performance and Accuracy
28	Generative AI
30	References
31	Glossary
34	Purpose of this Document

Why AI?

Artificial Intelligence (AI) is here to stay. This is one of the major upheavals in technology, like the PC and the smartphone. It will change what we do, how we do it, and who will do it. AI is a large and emergent field, and any syllabus on the subject will be quickly outdated. This set of micro-credentials is designed to help the tester understand AI, Generative AI, and how this affects testing.

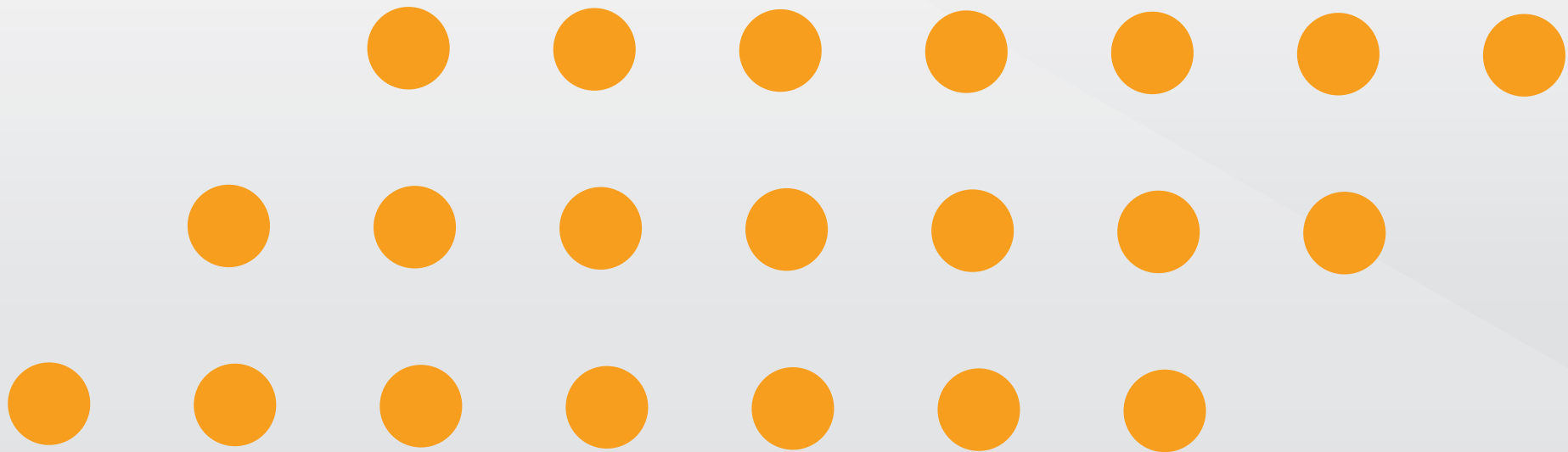
AI, in general, has a plethora of new terms associated with it. One of the purposes of this syllabus is to acquaint the tester with the AI terminology and the basics of AI. It will soon be an expectation that all testers are “familiar” with AI, and that will require having a basic understanding and the ability to learn as AI evolves.

The “Introduction to AI” module covers the basics and touches on a bit of the technology behind AI. This is not intended to be a full and technical explanation of AI; there are many books already devoted to that.

The second module, “What to Test in an AI-based System” discusses how to identify the risk items and conditions that must be tested. It explains data privacy and security concerns with AI-based systems, and it highlights the quality characteristics that are unique to AI-based systems.

The third module, “How to Test an AI-based System” explains how the traditional test levels can be applied to testing AI-based systems. It also discusses testing techniques that are particularly well-suited for AI system testing. The module includes a discussion of the challenges of testing AI systems.

The fourth module, “Using AI to Test”, explores how to use AI to help with testing. This looks at building test cases, building test automation, recording results, and automating many manual aspects of testing. Along the way, caveats are explored including what should be considered when AI is used. As the world has seen, AI is not always accurate.



Keywords

accuracy, activation value, artificial intelligence (AI), association, bias, classification, clustering, confusion matrix, F1-score, foundation LLMs, generative AI (GenAI), instruction-tuned LLMs, intelligent agent, large language models (LLM), machine learning (ML), multimodal models, neural networks, neurons, overfitting, precision, reasoning LLMs, recall, regression, reinforcement learning, retrieval-augmented generation (RAG), small language models (SLM), supervised learning, tokens, training data, transfer learning, transformation model, underfitting, unsupervised learning, vision-language models

LEARNING OBJECTIVES FOR AI INTRODUCTION FOR TESTERS

Machine Learning Models

(K2) Understand the difference between AI and GenAI

Neural Networks

(K2) Summarize the layers of a neural network

Machine Learning

(K2) Understand the three steps in model generation

Model Training

(K2) Understand the uses of a pre-trained model

(K2) Summarize the risks of using pre-trained models

Machine Learning Algorithms

(K2) Understand the differences between supervised and unsupervised learning

Data Considerations

(K2) Summarize the process of data preparation

(K2) Summarize the different datasets and their usage

Functional Performance and Accuracy

(K2) Understand how performance is evaluated for an AI model

Generative AI

(K2) Summarize the differences between foundation, instruction-tuned and reasoning LLMs

AI Introduction for Testers

According to IBM, “Artificial Intelligence (AI) is technology that enables computers and machines to simulate human learning, comprehension, problem solving, decision making, creativity and autonomy” (IBM, 2025). Wikipedia simplifies this as “the capability of computational systems to perform tasks typically associated with human intelligence.” (Wikipedia, 2025).

As testers, why do we care about AI? Is this just another blip in the software world, like Agile and test automation – something we need to learn about but that will not change what we do? No, AI is set to be a major disruptor, more

like the introduction of personal computers and smartphones. It is likely to revolutionize how humans interact with computer systems and, as a result, will impact how we test it and how we test with it.

In order to understand what and how to test AI, we need to have a basic understanding of what it is. One of the challenges to entering the AI world is mastering the terminology. New terms will be explained as they are introduced in this syllabus. There is also a glossary at the end of this syllabus which provides an easy reference for the reader.

Machine Learning Models

AI and Generative AI are not the same thing. AI is the broader field that covers the technologies that enable machines to perform tasks that formerly required a human mind. This includes such things as learning and problem-solving based on the information that has been learned. Generative AI (GenAI) is a subset of AI that focuses on creating or “generating” new text, images, code, etc., based on patterns that have been learned from existing data.

For example, if you want the software to identify a rabbit based on a set of images of rabbits upon which it has been trained, you are using AI. If you want the software to draw a rabbit wearing a top hat, that’s GenAI. The accuracy and capability of AI is heavily dependent on the accuracy and applicability of the data that was used to train it. If the AI module had been shown a set of images of dogs, but told they were rabbits, it won’t be able to accurately identify (or draw) a rabbit.

This brings us to the core concept of AI, which is Machine Learning (ML). Machine Learning refers to the ability of the machine (the computer and computer systems) to “learn” based on the information it is given. This information is used to create algorithms that are then used to analyze the patterns in data (such as words). This information is then used to build predictive models (i.e., what word is likely to follow in a sentence), and make decisions with little or no human intervention. ML systems are expected to learn from experience as they encounter more data, and to become more accurate and efficient.

Large Language Models (LLM) and Small Language Models (SLM) are types of machine learning models. As indicated by the name, LLMs are trained using huge datasets of text. The sheer size of the data used to train an LLM results in significant expense and computational resource requirements. For example, a training dataset for

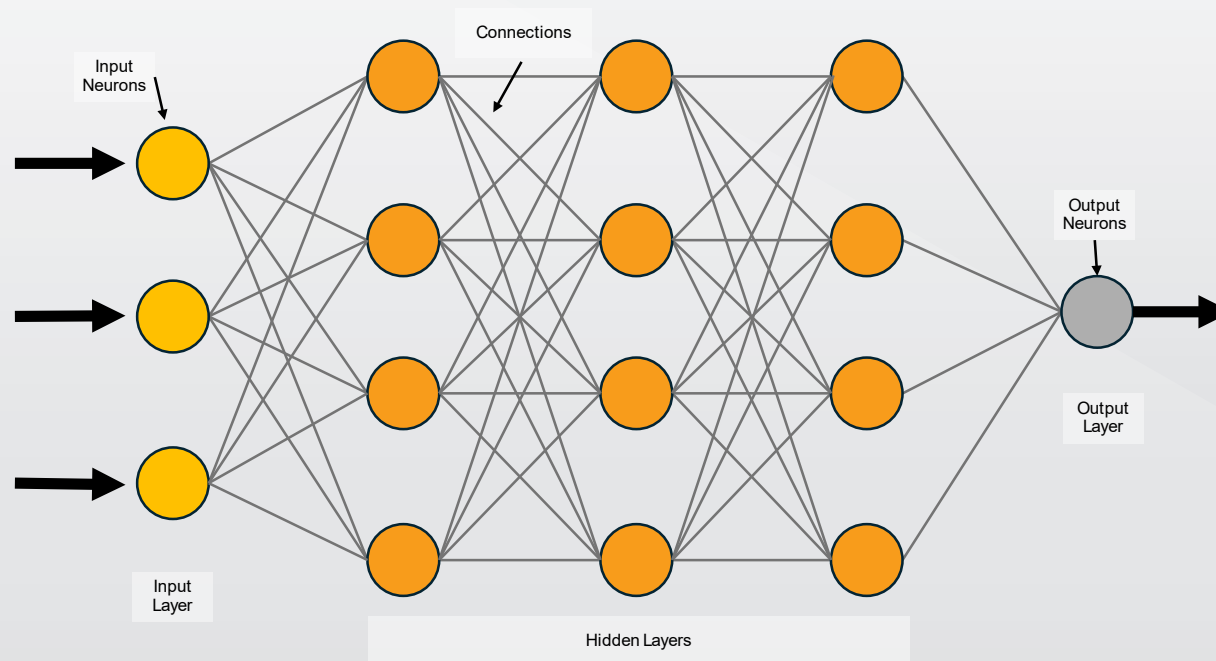
an LLM could be the entire Internet. Because of the size and depth of the training data, LLMs can write new content, answer complex questions, summarize large amounts of information and can even translate languages.

SLMs are smaller and less complex and are generally more targeted in their learning and application. They are generally well-suited for specific tasks or specific domains (such as software testing) based on their focused training. They are less expensive than LLMs and don't have the same resource requirements.

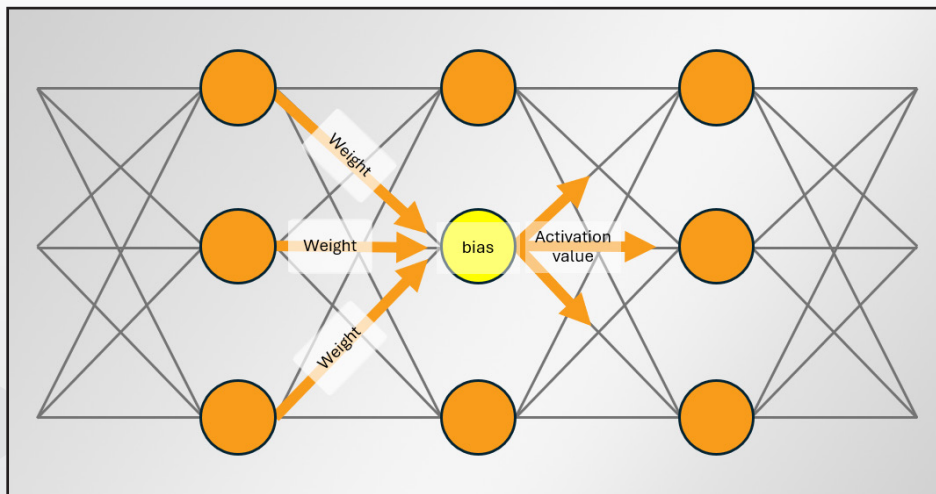
Neural Networks

Both SLMs and LLMs are based on neural networks. Neural networks are computational models that are based on the functioning of the human brain. They utilize interconnected nodes, called neurons, that are arranged in layers. Neural networks are called “deep” when they consist of multiple layers of neurons.

As shown in the diagram below [CT-AI], there are three types of layers: The input layer, the hidden layer(s), and the output layer. The input layer is responsible only for receiving the input. The output layer is responsible for presenting the outcome to the recipient. The hidden layer of neurons (sometimes called nodes) performs the computations.



The neurons in the hidden layer are connected to the next layer of neurons. The neurons perform their computations and send their output value to the connected neurons on the next layer. This continues as information is passed from the input layer, through the hidden layers, to the output layer.



Each neuron conducts its processing and generates an activation value. This value is computed by applying an activation function that receives the following inputs:

- The activation values from each of the neurons in the previous layer
- The weighting assigned to the connection between each neuron in the previous level and the receiving neuron
- The bias of the receiving neuron

Activation values, biases, and weights are key parts of how a neural network learns and makes decisions. Weights control how important each input is to a neuron, and biases let the neuron shift its decision boundary, giving it more flexibility. After combining inputs using weights and bias, the neuron applies an activation function, which decides what output (activation value) to send to the next layer. Activation values are the signals that carry information from one layer to the next, changing as the network learns.

The weighting of the connections changes as the neural network “learns”. The bias is a pre-set constant. The resulting activation value ranges from -1 (the neuron is not interested) and +1 (the neuron is very interested).

Neural networks are trained via an iterative process which requires feeding them known data and checking if the outputs provide the correct answers (predictions). The network’s internal parameters (weights and biases) are then adjusted to reduce the error. This process is repeated until the processing gives accurate results. Large training datasets are normally used for this training, which requires considerable processing power. Once the network is considered to be trained, the evaluation dataset is used to check its accuracy, which may lead to more adjustments and further evaluations.

If either the training or evaluation datasets are not accurate or not representative, the results may not produce the expected predictions. Ensuring the data for all three steps – training, evaluation, testing – is valid is necessary for a model to be trained correctly.

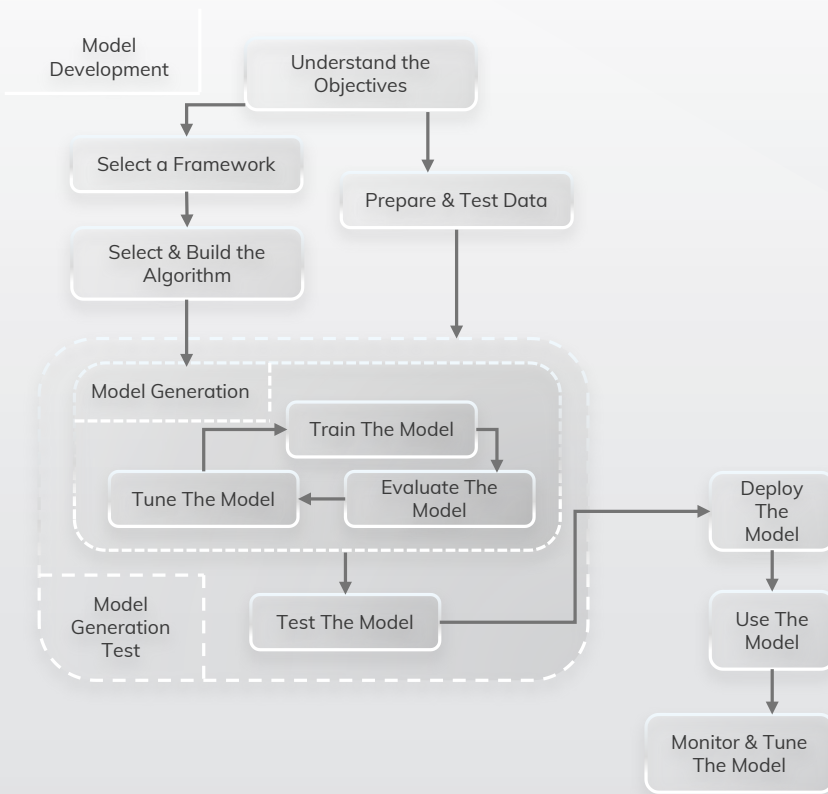
Testing of a neural network tends to focus on the neurons themselves:

- Was each neuron activated?
- Does each neuron return a value within an expected range (e.g., over a certain threshold)?
- Does each neuron have the ability to return a positive and negative activation value?

This level of white-box testing is usually conducted by the designers of the network. Testers usually concentrate on the outputs matching the expectations based on the inputs.

Machine Learning Workflow

The development of an ML model follows an established workflow, regardless of the type or purpose of the model. This model is shown below [CT-AI]:



These steps are explained in detail in the [CT-AI] syllabus. The following is a summary:

Understand the Objectives

Understanding the business priorities and getting agreement from the stakeholders is critical for a successful deployment of an AI model. It is also important to understand the full breadth of applications within an organization in order to adequately train the model. Acceptance criteria should be defined at this step.

Select a Framework

The appropriate framework should be selected based on the objectives defined in first step.

Select and Build the Algorithm

Based on the objectives, the defined acceptance criteria, and the data available for training, an algorithm is selected. This may be developed internally (manually coded) or may be obtained from a library of existing code.

Prepare and Test the Training Data

Data preparation includes acquisition, pre-processing, and feature engineering. Data analysis is often needed to determine the appropriateness of the data. This data will be used to train, tune, and test the model. It must be representative of the data that will be used in production (operational data). Data may already be available or it may need to be mined (and groomed).

Train the Model

The selected ML algorithm will use the training data to train the model. Guidelines may be passed to the model to help define the structure (such as layers of the neural network). Training data is sometimes processed multiple times as different features are being identified within the data.

Evaluate the Model

The trained model is tested using the evaluation dataset to determine if further tuning is needed. This is an iterative process that continues until the model has met the acceptance criteria.

Tune the Model

Based on the results of evaluation, the model may need to be adjusted to improve its accuracy. Additional tuning and evaluation may be needed to implement the necessary changes.

Test the Model

At this point, the model is tested using a separate test dataset (not the ones used for training or evaluation) to verify that the objectives are being met. Understanding the expected results is critical at this point to determine if the model is working correctly. In addition to testing the accuracy of the model, non-functional testing for performance (response time and resource requirements) and security are also evaluated with the final model.

Deploy the Model

When testing has been deemed adequate, the model is deployed into the production environment with the production data pipeline.

Use the Model

When the model is released into the system (usually a part of a larger AI-based system), it is used to either perform batch predictions at pre-determined intervals or it may run on request in real time.

Monitor and Tune the Model

It is important to remember that, unlike conventional software, once deployed the model continues to change as it learns. This may cause it to “drift” from its earlier results. In order to be sure this isn’t occurring, it’s important that the model is regularly evaluated to ensure the acceptance criteria are still being met. This is often done by reprocessing the test data and comparing the results. Drift is not necessarily bad, but it needs to be evaluated to ensure that the responses are still accurate, even if they are different. In the case that the drift is unacceptable, it may be necessary to re-train the model or create a new model that is more accurate.

Model Training

Tools and Hardware Requirements

There are tools that support the development of AI solutions. Frameworks such as TensorFlow and ApacheMXNet provide a platform that can be used to build, train, and deploy ML models. The tool sets are growing rapidly and the capabilities of the tools are continuing to advance. In addition to creating and training your own model, pre-trained models can be acquired to save time and money when launching an AI tool.

Hardware is another consideration for an AI implementation. The hardware required to train an ML model is usually considerably more powerful than the hardware needed to run the solution. For example, a text anticipation capability for a smartphone is small enough to run on the phone itself (the host device), but the systems

required to train that model probably used cloud computing. Hardware used for AI training usually uses low-precision arithmetic, an ability to work with large data structures, and “massively parallel (concurrent) processing” [CT-AI]. Hardware is being developed and refined that is particularly tuned for building ML models. This is an area where better, faster, and more efficient hardware will continue to become more affordable.

ML models can be created within an organization, assuming that the organization has the necessary hardware and knowledge. Pre-trained models can also be purchased from a third-party, eliminating the need for the assets required for training. Models can also be provided as a service (AlaaS). When used as a service, the access is provided via the Internet by the provider (e.g., Microsoft, AWS). The ability to get a trained ML and use it as a service allows organizations to get up and running with AI without requiring the extensive resources and expertise required to fully train a model.

An example of AlaaS is Microsoft's Azure AI Search, which can be used to develop AI apps that utilize enterprise data using AI agents to facilitate retrieval-augmented generation (RAG) workflows. This improves the adaptability and accuracy over traditional RAG systems. By using this as an AlaaS, the power of the RAG is available without having to create everything needed for it to work. Of course, any software like this has an inherent cost associated with it, but it allows AI applications to be developed much more quickly and accurately.

Pre-trained Models

It can be very expensive to train an ML model. Before the model training can commence, the training data has to be prepared. This can take a considerable amount of time to find and groom the data to be used. Once that's done, the training can occur, but this will consume large amounts of expensive computing resources. The solution to this problem is to use a pre-trained model that already has similar functionality to the required model. The pre-trained model is used as the basis to create a new model that either extends or specializes the base model.

Transfer Learning is the term used for taking a pre-trained model and modifying it for a different requirement. In this case, the first layers of the neural network are re-used because they perform generic tasks that are still needed, and the subsequent layers that are more specialized are modified for the specific problem to be solved. For example, a new AI system that will determine the best eyeglasses for a person could do the following:

- Use the first layers of an existing model that does facial recognition
- Isolate the capability of that model that processes the eyes
- Take that capability and transfer it to the new model (transfer the learning)
- Augment the new model with the capability to determine the optimal spectacles based on the positioning of the eyes

In this way, the training that is already in place for the recognition and classification of eyes can be leveraged, but used for a different application.

Transfer Learning is only cost-effective when the function of the models is similar. With the above example, starting with a model that determines speech patterns would not be helpful.

Risks with Models

The validity of a model, and the testability of a model, depend on having thorough documentation. If a pre-trained model is to be used, how was it trained? What is it supposed to do? Using a model that is not well understood can lead to data inaccuracies, invalid conclusions and a lack of testability.

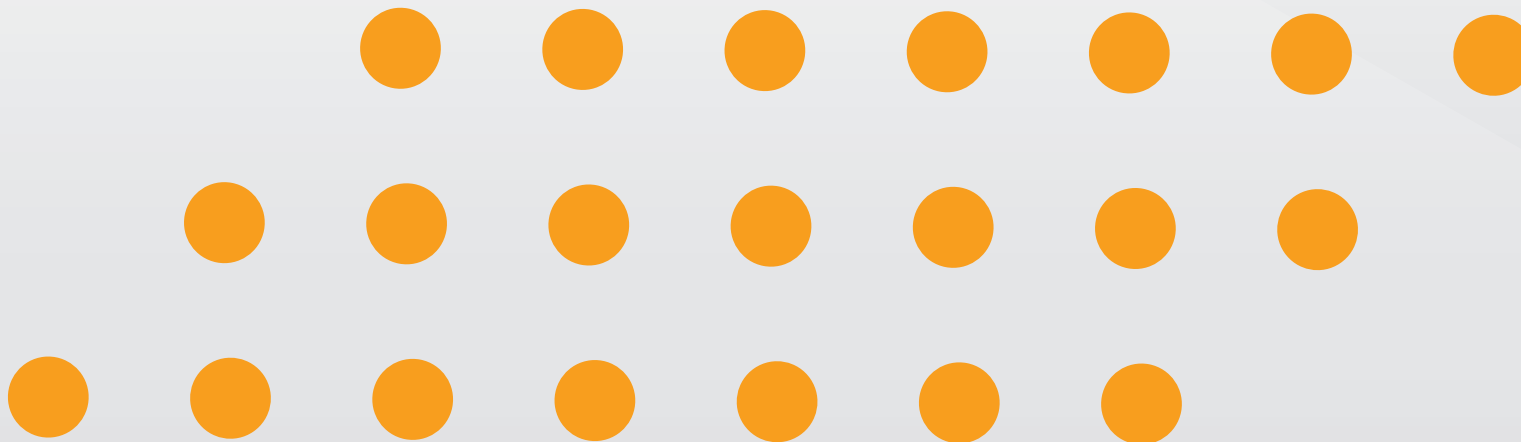
The risks as summarized in [CT-AI] are listed here with the testing concerns associated:

- Pre-trained models may lack transparency – how were they trained, what data was used, what is the expected processing?
- The intentions for the data may have differed between the original training and the customization, resulting in unpredictable outcomes.
- Shortcomings in the pre-trained model may not be documented and may be inherited. This will cause the testing effort to have to rediscover issues that were already known by the original training organization.
- When Transfer Learning is used, any vulnerabilities in the pre-trained model are likely still there. This creates unexpected security vulnerabilities that may be very difficult to detect during testing.



The more information available on the pre-trained model, the more accurate and focused the testing can be. Indeed, even the usage of the pre-trained model will be more aligned with the original intention if good documentation is available. It is

smart to understand that pre-trained models are probably not perfect and the use of that model, either as-is or via transfer learning, will inherit and potentially exacerbate those same flaws.



Machine Learning Algorithms

According to the [CT-AI] syllabus, there are three categories of ML algorithms. These are:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

The testing approach varies based on how a model was trained. When the tester knows how the model was trained, predicting the outcomes and creating the best test data set is more targeted and more reliable.

Supervised Learning

When a model is trained with a supervised approach, controlled data is fed to it for it to “learn”. The data is usually sent in pairs of inputs with the first item being the data (such as an image of a cat) and the second item being the label “cat”. This teaches the model the relationship between the image and the label. During testing of the model, data that was not used to train the model is used to test that the output classification is correct. If you use an image of a lion as input, you would still expect the result to be “cat”. If the result was “hyena” you will know the model hasn’t had sufficient training to ensure accuracy.

Data labelling can be a resource intensive activity. It is sometimes done by grouping similar data into folders (such as one folder for domestic cats and

another for wild cats) or by annotating an image so the software concentrates on the important aspects of the image. Data that is mis-labelled can introduce significant issues for the model. Imagine one dog image being labelled as a cat.

There are two main applications of this type of learning:

- **Classification** – this occurs when the input is expected to be classified into a pre-defined class. Classifying a picture of a lion as a cat is an example of this.
- **Regression** – this occurs when the model is expected to predict a numeric output. Predicting the age of death based on a person’s habits and medical data is an example of this.

Understanding which application of the learning will be used influences the test data that will be used and the expected results. These are relatively simple examples, but understanding how the model is expected to derive the result and how it was trained directly impacts the testing that will be needed.

Unsupervised Learning

In unsupervised learning, the labels are not provided – only the inputs. The algorithm takes the input data and looks for patterns in that data, then uses those patterns to classify it. The classification is not “supervised” and will depend heavily on the variety in the input data. When testing this type of model, a new set of inputs is provided to see if the classification of the data is done in an expected way.

There are two main applications of this type of learning:

- **Clustering** – this occurs when the goal is to identify similarities in the input data that result in it being “clustered” into groups depending on commonalities. Classifying pictures of human faces, cat faces, and dog faces into separate groups is an example of this.
- **Association** – this occurs when relationships and dependencies are identified among the data attributes. Associating such health factors as cholesterol level, blood pressure, and BMI with heart disease is an example of this.

Reinforcement Learning

This approach does not use training data. Instead, the “intelligent agent” learns by interacting with the environment iteratively. When given an input, the agent is “rewarded” for a correct decision. Chatbots are an example of this when they receive feedback from the human as to whether or not they have been helpful and if their answer is accurate. On a much larger scale, autonomous cars use this same technology to learn what objects require avoidance (such as a person in front of the car) or not (such as a shadow in the road).

Understanding the Learning

When the tester understands how the model was trained, the testing can be targeted correctly, input data can be developed, and outputs can be evaluated. Because some of these models will continue to learn, the answers they give may evolve, so the training data must be updated accordingly.

There are two areas to be aware of when determining the data used to train and test a model:

- **Overfitting** – This occurs when the model is unable to generalize from the training data to handle new data. This can occur when the training data was too specific or just too low in quantity. For example, if the model was trained on only one type of domestic cat, it may not be able to generalize cat characteristics to identify a lion as a type of cat.
- **Underfitting** – This occurs if the model is too simple and doesn’t accurately identify the trends in the training data. As a result, it has difficulty in making accurate predictions from a variety of input data. For example, if it was trained only on pictures of white kittens to determine what is a cat, it won’t have the capacity to understand that cats can be different sizes and colors. As a result, it won’t be able to recognize a black cat as a cat.

If a model is not trained properly, it won't work properly and may build on its own incorrect decisions. It is important for a tester to review the training data, understand its derivation, and compare that to the expected data that will be encountered in production. This is often an area handled by data scientists, but in order to know what to test, the tester needs to understand the training.

Data Considerations

As seen from the previous section, getting the right data for training, evaluation, and testing is critical to the success of an ML model.

Data Preparation

Data preparation is a critical component of the data pipeline which takes the input data and outputs the processed data in a form that is used to both train the model and for real processing by the released model.

Data preparation is composed of the following:

- **Data acquisition** – Identifying, gathering and labelling the data
- **Data pre-processing** – Cleaning, transforming, augmenting, sampling
- **Feature Engineering** – Selecting needed features from the data, extracting features to reduce the size of the data to be processed

Finding and acquiring the proper data requires good knowledge of the domain, the data itself, and how to do the data preparation. It is sometimes difficult to get the volume of good data that is needed. This sometimes results in having to groom the available data. Data preparation is usually the most time-consuming aspect of the ML workflow and is certainly one of the most critical for the accuracy of the resulting model.

Data Sets

It's important to remember that there must be at least three separate datasets:

- **Training dataset** – to train the model
- **Validation dataset** – to evaluate and tune the model
- **Test dataset** – to test the final version of the model

Because requirements may change over time, multiple versions of these datasets may be needed. It's important to remember that a test dataset may not produce the same results when used several times. This is because the model learns from the test data, just as it does from the real data it is processing. As a result, the results may change but are not necessarily incorrect. The differences must be analyzed to determine if the changes indicate improvements in the model or if invalid decisions are occurring.

Functional Performance and Accuracy

Performance, in AI terms, generally refers to the performance of the model itself in terms of the accuracy and speed of its data processing. Performance as experienced by the end user is a completely separate performance indicator.

It is important to remember that an AI model will rarely, if ever, be 100% accurate. Given the inherent inaccuracies, accuracy is often measured on the basis of a Confusion Matrix, shown here [CT-AI]:



This matrix is used to compute the following metrics [CT-AI]:

- **Accuracy** – the percentage of all correct classifications
 - o $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}) * 100\%$
- **Precision** – The proportion of predicted positives that are actually correct. It reflects the reliability of positive predictions.
 - o $\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) * 100\%$
- **Recall** (also known as Sensitivity or True Positive Rate) – the proportion of actual positives that were predicted correctly. It shows the model’s ability to detect positives.
 - o $\text{Recall} = \text{TP} / (\text{TP} + \text{FN}) * 100\%$
- **F1-score** – A combined measure of Precision and Recall that shows how well the model detects positive cases. A low F1-score means the model either misses many positives (low recall) or makes many incorrect positive predictions (low precision).
 - o $\text{F1-score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$

Different values are used to assess models depending on their acceptance criteria. For example, if the model is evaluating X-rays to check for signs of cancer, ensuring that all the positives are correctly reported is critical, more important than occasionally reporting a negative result as a positive (false positive). A low F1-score on the braking system of an autonomous car could be catastrophic.

In testing, it’s important to track the confusion matrix to check the accuracy of the model. This information should be added to defect tracking and should also be reported in all defect reports.

Additional metrics that can be determined and tracked during evaluation and testing are discussed in [CT-AI].

Generative AI

As mentioned in the introduction, Generative AI (GenAI) is the branch of Artificial Intelligence that uses trained LLMs or SLMs to create something new. This could be text or images, or something particularly targeted for a usage such as programming code. The training provides the necessary information to determine the context of the situation and to respond correctly to user prompts.

A common application for GenAI is to generate a document or anticipate text. In order to do this, the model has to be trained on a large set of text. To understand the text, it is broken down into small units, called tokens. A token can be a word, a subword, or an individual character. This is a part of natural language processing. With the tokens, the model can calculate the relationships between words and “understand” the context.

This knowledge is then used for the model to understand user prompts to create something new (i.e., a new document) using GenAI or to predict the next word in a sentence (i.e., filling in words in text messages) using Predictive AI. Predictive AI analyzes existing data to forecast future events and trends whereas GenAI creates new and original content.

LLMs use a transformer model, which is a type of neural network architecture. The transformer models can take tokenized text sequences and learn how the tokens relate to each other. From this information, the LLM can infer the next word in a sequence. This is all based on probability and may not be correct, or may be correct but awkward. There is inherent randomness in the responses, which means that even the exact same prompt may not produce the same output.

According to [CT-GenAI] there are training stages for the LLMs. The amount of training determines the applicability of the model to a task. The training stages are as follows:

- **Foundation LLMs:** These are general purpose models that have been trained on vast and diverse datasets containing text, code, images, etc. They can support tasks across different domains, such as processing natural language, speech recognition, etc. Because they are generalized, they need further training to meet the needs of a specific application such as test case development.
- **Instruction-tuned LLMs:** These build on Foundation LLMs by receiving further training by ingesting datasets with pairs of prompts and expected responses.
- **Reasoning LLMs:** These extend the instruction-tuned LLMs by building “structured cognitive abilities” such as the ability to infer logic, multi-step problem-solving and reasoning that results from a chain of thoughts. Training for these models concentrates on tasks that require contextual understanding, reasoning, and the synthesis of complex information.

For software testing applications, the instruction-tuned and reasoning LLMs are the most applicable because they can be targeted to understand software testing needs and terminology.

There are two other models that are particularly interesting for software testing. These are multimodal models and vision-language models. Multimodal models can process different types of data such as text, images, sound, and video. Vision-language models (which are a subset of multimodal models) can integrate visual and textual information. For example, the model could evaluate a screenshot against the requirements to ensure all the elements are present and correct. It could then write the defect report highlighting the differences. On the other end of testing, the vision-language models can create test cases that include textual and visual aspects.

References

ISTQB® Documents

[CT-AI] ISTQB Certified Tester – Artificial Intelligence Syllabus, v1.0, 2021

[CT-GenAI] ISTQB Certified Tester – Generative AI Syllabus, v1.0, 2025

Glossary References

ISTQB® Glossary <https://glossary.istqb.org/>

Web Pages

(IBM, 2025) IBM. What is Artificial Intelligence, <https://www.ibm.com/think/topics/artificial-intelligence>. Accessed 30 June 2025.

(Wikipedia, 2025) Wikipedia. Artificial Intelligence, https://en.wikipedia.org/wiki/Artificial_intelligence. Accessed 30 June 2025.

The previous references point to information available on the Internet and elsewhere. Even though those references were checked at the time of publication of this syllabus, AT*SQA cannot be held responsible if the references are not available anymore.

Glossary

accuracy - The ML functional performance metric used to evaluate a classifier, which measures the proportion of predictions that were correct (After ISO/IEC TR 29119-11)

activation function - The formula associated with a neuron in a neural network that determines the output of the neuron from the inputs to the neuron

activation value - The output of an activation function of a neuron in a neural network

artificial intelligence (AI) - The capability of an engineered system to acquire, process, create and apply knowledge and skills (ISO/IEC TR 29119-11)

association - An unsupervised learning technique that identifies relationships and dependencies between samples

bias - The systematic difference in treatment of certain objects, people or groups in comparison to others (ISO/IEC DIS 22989)

classification - A type of ML function that predicts the output class for a given input (After ISO/IEC TR 29119-11)

clustering - A type of ML function that groups similar data points together

confusion matrix - A technique for summarizing the ML functional performance of a classification algorithm

F1-score - An ML functional performance metric used to evaluate a classifier which provides a balance between recall and precision

foundation LLM - General-purpose models pre-trained on a wide range of text data, capable of predicting the next word based on learned linguistic patterns. Synonym: Base LLM

generative AI (GenAI) - An advanced technological approach that enables the creation of content including text, images, and videos. (the abbreviation for 'generative artificial intelligence').

hallucination - Wrong information created by an LLM.

intelligent agent - An autonomous program which directs its activity towards achieving goals using observations and actions

large language model (LLM) - A computer program that uses very large collections of language data in order to understand and produce text in a way that is similar to the way humans do.

machine learning (ML) - The process using computational techniques to enable systems to learn from data or experience (ISO/IEC TR 29119-11).

multimodal model - GenAI models that are capable of processing and generating content across multiple data types, such as text, images, and audio.

neural network - A network of primitive processing elements connected by weighted links with adjustable weights, in which each element produces a value by applying a nonlinear function to its input values, and transmits it to other elements or presents it as an output value (ISO/IEC 2382) Synonym: artificial neural network

neuron - A node in a neural network, usually receiving multiple input values and generating an activation value

overfitting - The generation of an ML model that corresponds too closely to the training dataset, resulting in a model that finds it difficult to generalize to new data (After ISO/IEC TR 29119-11)

precision - An ML functional performance metric used to evaluate a classifier, which measures the proportion of predicted positives that were correct (After ISO/IEC TR 29119-11)

reasoning LLM - An LLM building upon instruction-tuned models by refining their ability to emulate human-like reasoning processes

recall - An ML functional performance metric used to evaluate a classifier, which measures the proportion of actual positives that were predicted correctly (After ISO/IEC TR 29119-11) Synonym: sensitivity

regression - A type of ML function that results in a numerical or continuous output value for a given input (After ISO/IEC TR 29119-11)

reinforcement learning - The activity of building an ML model using a process of trial and reward to achieve an objective (After ISO/IEC TR 29119-11)

retrieval-augmented generation (RAG) - A technique combining LLM capabilities with a retriever to fetch relevant data for generating accurate, contextually relevant responses.

small language model (SLM) - Language models that are intentionally designed and trained to be small, offering a balance between efficiency and task-specific language understanding.

supervised learning - Training an ML model from input data and its corresponding labels

tokenization - The process of breaking down text into smaller units (tokens) for processing by language models.

training dataset - A dataset used to train an ML model

transfer learning - A technique for modifying a pre-trained ML model to perform a different related task

underfitting - The generation of an ML model that does not reflect the underlying trend of the training dataset, resulting in a model that finds it difficult to make accurate predictions (ISO/IEC TR 29119-11)

unsupervised learning - Training an ML model from input data using an unlabeled dataset

vision-language model - A GenAI system that jointly processes visual and textual data to perform tasks by linking and generating content across both modalities.

Purpose of this Document

This syllabus forms the basis of the AT*SQA certification for AI for Testers. AT*SQA is an International Standards Organization (ISO) compliant certification body for software testers. AT*SQA provides this syllabus as follows:

1. To training providers - to produce courseware and determine appropriate teaching methods.
2. To certification candidates - to prepare for the exam (as part of a training course or independently).
3. To the international software and systems engineering community - to advance the profession of software and systems testing, and as a basis for books and articles.

AT*SQA may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

This syllabus forms the basis of the AT*SQA certification for AI for Testers. AT*SQA is an International Standards Organization (ISO) compliant certification body for software testers. AT*SQA provides this syllabus as follows:

1. To training providers - to produce courseware and determine appropriate teaching methods.
2. To certification candidates - to prepare for the exam (as part of a training course or independently).
3. To the international software and systems engineering community - to advance the profession of software and systems testing, and as a basis for books and articles.

AT*SQA may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

This syllabus forms the basis of the AT*SQA certification for AI for Testers. AT*SQA is an International Standards Organization (ISO) compliant certification body for software testers. AT*SQA provides this syllabus as follows:

1. To training providers - to produce courseware and determine appropriate teaching methods.
2. To certification candidates - to prepare for the exam (as part of a training course or independently).
3. To the international software and systems engineering community - to advance the profession of software and systems testing, and as a basis for books and articles.

AT*SQA may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.



www.atsqa.org

