AiU Certified Tester in AI (CTAI) Syllabus

Version 1.0R 2019

Artificial Intelligence United

Copyright Notice

Thank you to the main authors:

- Vipul Kocher, Saurabh Bansal, Srinivas Padmanabhuni and Sonika Bengani

Thank you to the co-authors

- Rik Marselis and José M. Díaz Delgado

Thank you to the review committee

Amit Dang, Andreas Hetz, Ángel Rayo Acevedo, Aurelio Gandarillas, Baris Sarialioglu, Björn Lemke, Christine Green, Daniel Garcia Castillo, Dario Iván Rosas Miranda, Durga Mohapatra, Emilie Potin-Suau, Erik van Veenendaal, Girish Nuli, Girts Baltaisbrencis, Guino Henostroza, Gustavo Márquez Sosa, Héctor Ruvalcaba, Javier Alejandro Chávez Crivelli, Jeff Nyman, Joel Oliveira, José Antonio Rodríguez Gómez, Juan Pablo Rios Alvarez, Julie Gardiner, Kimmo Hakala, Kristine Corbus, Kyle Alexander Siemens, Lorena Parra Rubio,

Maarten-Jan van Gool, Márton Görög, Michaël Pilaeten, Michel Dussouchaud, Nora Alriyes, Paweł Noga, Petr Neugebauer, Ralf Pichler, Ram Shanmugam, Sammy Kolluru, Samuel Ouko, Santiago de Jesús González Medellín, Sebastia Małyska, Sergio Emanuel Cusmai, Serge Wolf, Shantel Yanique Stewart, Silvia Nane, Sunil Godse, Siva Prasad. B, Søren Wassard, Tariq King, Tetsu Nagata, Vikas Dhaka, Tom Van Ongeval, Werner Lieblang, Wim Decoutere,
Yogesh Ahuja, Young jae Choi

Revision History

Version	Date	Remarks
AiU A 2019	March 2, 2019	First beta release
AiU B 2019	May 8, 2019	Second beta release
AiU 1.0R 2019	July 16, 2019	First release

Table of Contents

Business Outcomes

BO-1	Understand current trends, industry applications of Artificial Intelligence (AI) using Machine Learning (ML).
BO-2	Compare different implemented ML algorithms to help choose the most suitable one.
BO-3	Evaluate models for both supervised and unsupervised learning.
BO-4	Design and execute test cases for AI systems.
BO-5	Use various methods for bringing transparency into model workings.
BO-6	Define a test strategy for testing of AI systems.
BO-7	Understand where AI can be used in manual testing and in test automation.
BO-8	Use AI based test execution tools to automate tests.

Learning Objectives/Cognitive Levels of Knowledge

Learning objectives (LOs) are brief statements that describe what you are expected to know after studying each chapter. The LOs are defined based on Bloom’s modified taxonomy as follows:

- K1: Remember. Some of the action verbs are Remember, Recall, Choose, Define, Find, Match, Relate, Select
- K2: Understand. Some of the action verbs are Summarize, Generalize, Classify, Compare, Contrast, Demonstrate, Interpret, Rephrase
- K3: Apply. Some of the action verbs are Implement, Execute, Use, Apply

For more details of Bloom's taxonomy please refer to [BT1] and [BT2] in References.

Hands-on Objectives

Hands-on Objectives (HOs) are brief statements that describe what you are expected to perform or execute to understand the practical aspect of Learning.

The HOs are defined as follows:

- HO-0: Live demo of an exercise or recorded video
- HO-1: Guided exercise. The trainees follow the sequence of steps performed by the trainer
- HO-2: Exercise with hints — Exercise to be solved by the trainee utilizing hints provided by the trainer
- HO-3: Unguided exercises without hints

Prerequisites

Mandatory

- None

Recommended

- ISTQB® Foundation level or equivalent
- Basic knowledge of any programming language - Java/Python/R
- Basic knowledge of statistics
- Some software development or testing experience

Chapter 1 - Introduction to Artificial Intelligence

Keywords

artificial intelligence, machine learning, supervised learning, unsupervised learning, supervised classification, supervised regression, unsupervised clustering, unsupervised association, reinforcement learning, deep learning, neural networks, CRISP-DM, ML life cycle.

LO-1.1.1	K2	Explain artificial intelligence (AI).
LO-1.1.2	K1	Recall various types of AI — Narrow, General and Super AI.
LO-1.2	K2	Explain machine learning (ML) and the fact that ML is one way to achieve AI.
LO-1.2.1.1	K2	Explain Supervised ML and the difference between Supervised Classification and Regression.
LO-1.2.1.2	K2	Explain Unsupervised ML and compare Unsupervised Clustering and Association.
LO-1.2.1.3	K1	Recall the definition and applications of Reinforcement Learning (RL).
LO-1.3	K2	Explain Deep Learning (DL) and types of Neural Networks.
LO-1.4.1	K2	Explain various stages of CRISP-DM — a process for the ML lifecycle.
LO-1.4.2	K3	Apply the steps involved in identifying the appropriate ML problem type.

1.1 Artificial Intelligence (AI)

Artificial intelligence is the intelligence acquired by a machine to solve problems usually solved by humans.
IBM Watson, MS Cortana, Apple Siri, Autonomous Vehicles are some of the examples of a large number of existing well-known AI applications.

1.1.1 Definition of Artificial Intelligence (AI)

LO-1.1.1	K2	Explain artificial intelligence (AI).

AI is an art of creating machines that perform functions that require intelligence when performed by people. [KUR]AI is playing a leading role in healthcare, manufacturing, e-commerce, retail, social Media, logistics and other industry sectors. Some of the reasons for increasing use of AI are the rise in the processing power, availability of data, and new technologies. AI use cases span from monitoring one's home, determining which stock to invest in, helping to decide which recipe to make to helping in choosing your life partner!

AI is an umbrella term which covers the science of making machines intelligent, whether it is a robot, a refrigerator, a television, a car, a firmware or a software component. ML is the subset of AI. ML and AI are often used interchangeably, but they are not the same thing.

ML is explained in more details in 1.2 Machine Learning (ML)

1.1.2 Types of AI

LO-1.1.2	K1	Recall various types of AI — Narrow, General and Super AI.

AI can broadly be categorized as Narrow, General or Super AI.

- **Narrow AI**: Machines that are programmed for carrying out a specific task with limited context. For example, game playing machines, voice assistants and all AI currently.
- **General AI**: Machines with general cognitive abilities are popularly called as Strong AI cases. These AIs can reason and understand their environment as humans do, and act accordingly. For instance, common-

sense reasoning. Currently, General AI has not been realized and nobody knows when or if it will become a reality at all.

- **Super AI**: Machines that are capable of replicating human thoughts, ideas and emotions. It is that super state of intelligence where machines will become smarter and wiser than humans. Considering the current state of AI developments, Super AI will not become a reality anytime soon.

1.2 Machine Learning (ML)

1.2.1 Definition of ML

LO-1.2.1	K2	Explain machine learning (ML) and the fact that ML is one way to achieve AI.

Arthur Samuel defined ML as, “A field of study that gives computers the ability to learn without being explicitly programmed.” ML systems learn and improve with experience, and, with time, refine a model that can be used to predict the outcome of questions, based on the previous learning [JB1].

Beyond ML, AI uses concepts of knowledge representation and reasoning for handling diverse scenarios. Notion of searching, scheduling and optimizing fall under the scope of AI, but not necessarily ML.

Some of the technologies used to accomplish AI are:

- Machine Learning (ML)
- Natural Language Processing (NLP)
- Robotics
- Speech Processing
- Computer Vision

There are a few ways in which ML algorithms can be categorized:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

1.2.2 Supervised Learning - Classification and Regression

LO-1.2.2	K2	Explain Supervised ML and the difference between Supervised Classification and Regression.

Supervised Learning: In this kind of learning, the model learns from labeled data during the training phase. The labeled data acts as a trainer/supervisor for the mapping function which infers the relationship between input data and the output label during the training. During the testing phase, the mapping function is then applied to a new set of unseen data to predict the output which is also labeled. The model is deployed once the output accuracy level is satisfactory.

Problems solved by Supervised Learning are further divided into two categories:

Classification: When the problem requires classifying an input into one of a few pre-decided classes, supervised learning is used. This kind of model is used when the output data is discrete or when the output falls among the number of classes fed during training. Face recognition or object detection in an image are examples of problems that can use classification. Some other applications of classification are spam detection (spam or no spam), the diagnosis of a disease on the basis of the likes of an X-ray, correct identification of road signs by a driver assistance system, etc. Some of the commonly used algorithms for classification are logistic regression, nearest neighbor, support vector machine, and neural nets.

Regression: When the output data is continuous or numeric in nature, e.g., predicting the age/weight of a person, predicting the future price of the stock, etc., regression learning is used. The most commonly used algorithm for this kind of problems is linear regression, a simple algorithm which explains the relation between inputs and the output and inputs as a linear equation. Some other algorithms are logistic regression, support vector machines, Lasso regression, etc.

1.2.3 Unsupervised Learning — Clustering and Association

LO-1.2.3	K2	Explain Unsupervised ML and compare Unsupervised Clustering and Association.

Unsupervised Learning: Clean, labeled data are not readily available all the time, so that certain problems need to be solved without an explicitly labeled training set. This kind of ML where no labeled data is provided explicitly is called Unsupervised Learning. The objective in such problems is to learn the pattern and structure of input data without any associated labels.

Unsupervised Learning is further classified in the following two methods, based on the type of outputs:

Clustering: This Unsupervised Learning model groups the input data based on some common characteristics or attributes. Input data with similar attributes (not labeled) are grouped in one cluster. Thus, the outputs are clusters of input data. For instance, customer segmentation in market analysis.

Association: Association Rule Mining finds interesting relationships or dependencies among the data attributes. The discovery of interesting associations provides a source of information often used for decision making. For example, market-basket data analysis, product recommendation system based on learnings derived from customer shopping behavior are good examples of association rule-based modeling.

1.2.4 Reinforcement Learning

LO-1.2.4	K1	Recall the definition and applications of Reinforcement Learning (RL).

It is a type of ML where an agent (algorithm) learns by interacting with the environment in an iterative manner and thereby learns from experience. The agent is rewarded when it makes a right decision and penalized when it makes a wrong one. This reward and penalty-based learning is thus defined as ‘reinforcement learning’ (RL). Setting up the proper environment, choosing the right strategy for the agent to meet the desired goal, and designing a reward function, are some of the key challenges in implementing RL. Robotics, Autonomous Vehicles, and Chatbots are examples of applications that can use RL.

1.3 Deep Learning (DL)

1.3.1 Deep Learning and the Types of Neural Networks

LO-1.3.1	K2	Explain Deep Learning (DL) and the types of Neural Networks (NN).

Deep Learning (DL) refers to the systems gaining experience from massive data sets. DL uses Artificial Neural Networks (ANN) to analyze large data sets, e.g. Autonomous Vehicles, Large Text Processing, and Computer Vision applications among others. [AG1]

DL is a subset of ML and ML is a subset of AI. DL uses the same types of learning (Supervised, Unsupervised and Reinforcement Learning) as ML.

Artificial Neural Networks: Artificial Neural Networks (ANN) are inspired by the architecture of the human brain. 'Neurons', as the basic unit of ANN, act upon the input stimulus and produce the output signal. The input goes through the layers of activation functions to generate the output. These layers form a mesh like network.

Every ANN has at least two layers — input and output layers. All the layers between these two layers are called hidden layers. Some of the various types of neural networks are:

Deep Neural Network (DNN): Deep Neural Network (DNN) is an ANN with two or more hidden layers.

Convolutional Neural Network (CNN): Convolutional Neural Network (CNN) is an ANN that emerged from the study of the brain's visual cortex, and they have been used in image recognition since the 1980s. Unlike other neural networks, CNNs work directly on input images without serializing/ vectorizing an input image and extracting features by filters. CNNs power image search services, autonomous vehicles, automatic video classification systems, and more.

Recurrent Neural Network (RNN): These ANNs can predict the future of time series problems. They follow a sequential approach on series of input data of arbitrary length rather than inputs of fixed length as in other neural networks. Each input and output are independent of all the other layers. The feedback from the output layer is fed to the same network recurrently, till the right level of confidence is achieved. RNNs can analyze time series data such as stock prices, and tell you when to buy or sell. In autonomous vehicles, they can anticipate trajectories and help avoid accidents.

1.4 Stages of the ML Process

A typical ML project follows all the stages of the Cross Industry Standard Process for the Data Mining (CRISP-DM) framework — an industry standard, and a flexible framework.

1.4.1 Stages of the ML Process — CRISP-DM Process

LO-1.4.1	K2	Explain various stages of CRISP-DM — a process for the ML lifecycle.

CRISP-DM has traditionally six stages in the data mining life cycle. It has been customized to meet the requirements of ML projects, by adding a seventh stage.

The seven stages of the CRISP-DM framework for ML are: [DSC1] [SMU]

1. **Data acquisition**: Gather data from all internal and external sources (for example databases, CSV files, social media, etc.)

2. **Data preparation**: Clean the raw data and reshape it. New attributes are created with feature engineering, a process for creating new variables from existing data. Dimensionality reduction, data imputation, null value treatment for the missing values, etc., are some of the methods involved in data preparation.

3. **Modeling**: Select the model or algorithm, divide the available data into training set and testing set. Models are obtained by executing ML algorithms on the training data set. Use the testing data set to evaluate and enhance the performance of the model until satisfactory performance is achieved.

4. **Evaluation**: Evaluate the model on various metrics (discussed in 3.2 Metrics) and baseline it before it goes for final deployment.

5. **Deployment**: Deploy and monitor the baselined model for metrics in the production environment.

6. **Operations**: Carry out regular maintenance and operations. Regenerate and refine the model when the metrics fall below a certain threshold.

7. **Optimization**: The deployed solution may be replaced due to concept drift (see 6.1.3 Risk of Concept Drift (CD)), as better algorithms become available, or because of some major failures in performance.

Steps 1-4 can be classified as part of the **offline** phase, the output of which is the trained model. Steps 5-6 are a part of the **online** phase, where the model trained in the offline phase is integrated with the rest of the system and deployed in a production environment. The optimization step involves re-execution of steps 1-6.

1.4.2 Steps for the Identification of the ML Problem Type

LO-1.4.2	K3	Apply the steps involved in identifying the appropriate ML problem type.

It is important to understand the problems that we are trying to solve and the type of learning required to solve those problems. One way we can identify the ML problem type is discussed below:

1. If the problem involves the notion of multiple states, and involves moves at each state, then explore RL.
2. If there is an output variable it is supervised learning.
   2.1. In the case that the output is discrete and categorical, it is a classification problem.
   2.2. In the case that the output is numeric and continuous in nature, then it is a regression problem
3. If the output is not provided in the given data set, then explore the unsupervised learning.
   3.1. If the problem involves grouping similar data, then it is a clustering problem.
   3.2. If the problem involves finding co-occurring data items, then apply association rule mining
   3.3. If the raw data is unstructured, extracting features automatically can be explored with deep learning algorithms.

The prerequisite to the above steps is that there should be enough data available for the analysis of the appropriate ML problem type.

Chapter 2 - Overview of Testing AI Systems

Keywords: offline phase, online phase, ISO25010

LO-2.1.1	K2	Explain the tests performed in AI-ML systems in the training (offline) phase and post-training integration (online) phase.
LO-2.2.1	K2	Compare testing of AI systems with non-AI systems.
LO-2.3.1	K1	Recall the ISO 25010 quality characteristics, especially functional suitability, performance efficiency, reliability, maintainability and other parameters such as complexity, scalability and continuous learning as the parameters to be used for the evaluation of the trained model.
LO-2.3.2	K1	Recall the quality characteristics specific to AI testing in addition to those mentioned in ISO25010 — intelligent behavior, morality and personality.

2.1 AI Testing Phases

2.1.1 Offline and Online Testing of AI Systems

LO-2.1.1	K2	Explain the tests performed in AI-ML systems in the training (offline) phase and post-training integration (online) phase.

The ML lifecycle, as described in 1.4 Stages of the ML Process, can be divided into two phases: offline and online. Distinct types of tests performed during these phases are

Offline phase testing: In this phase, the trained model is tested. Various metrics are used for evaluation parameters for a trained model to verify how far it has achieved the objectives. There are different parameters for both supervised and unsupervised learning. Typically, the model is tested for functional behavior but non-functional characteristics are not tested. Since the model is deployed in an environment which is different from the training environment, doing performance testing of the model in this phase does not make much sense. Model training time is a parameter that is evaluated as a non-functional parameter. For models that will need to be retrained frequently, this can be an important parameter. Offline testing is covered in Chapter 3 - Offline Testing of AI Systems.

Another Important aspect of offline testing is whether it is possible to be able to explain the behavior of the model. There are various methods and algorithms that can be used for it. This aspect of testing is covered in Chapter 5 - Explainable AI.

Online phase testing: In this phase, the integration of the trained model with the rest of the system, including all other AI and non-AI components, is tested. Both functional and non-functional tests such as performance tests can be performed.

Since the inputs to the system can be non-textual, unstructured inputs, as well automation support for some of the tests related to the ML part, may be limited, based on the tool being used.

Online testing is covered in Chapter 4 - Online Testing of AI Systems.

2.2 AI vs. Non-AI Testing

2.2.1 Testing of AI systems vs. Traditional (non-AI) Systems

LO-2.2.1	K2	Compare testing of AI systems with non-AI systems.

- Test oracles for AI systems are not easily available
- Unlike in traditional testing, the outcomes of testing AI systems are non-deterministic. The output of an AI system has probabilities thus, the result of a test case is also probabilities rather than a definite pass/fail.
- AI systems logic is generated based on the data used to train the mode. That logic is not available for examination, especially neural nets. This makes it difficult to understand why a particular output was produced. A correct or desired answer doesn’t guarantee correct functioning.
- Testing in the offline phase is an additional phase which requires specialized skills and techniques, such as those related to data cleaning and preprocessing, and testing the trained model.
- Testing in the online phase requires a deep understanding of how AI systems work. The integration of AI systems with other AI and non-AI systems increases the need for different test design techniques.

- Similar to non-AI systems, both functional and non-functional tests need to be executed for AI systems.
- Online phase testing can be performed as normal black-box system and system integration testing without worrying whether there are one or more AI components in the mix.

2.3 AI Quality Characteristics

2.3.1 Quality Characteristics for Evaluating AI Systems

LO-2.3.1	K1	Recall the ISO 25010 quality characteristics, especially functional suitability, performance efficiency, reliability, maintainability and other parameters such as complexity, scalability and continuous learning as the parameters to be used for the evaluation of the trained model.

The quality characteristics of an AI system can be evaluated based on a combination of quality characteristics from ISO 25010 and other quality characteristics. Some of the important characteristics from the AI testing perspective are

- **Functional suitability** - Functional correctness, completeness and appropriateness.

- **Reliability**
  - Availability- Availability of the system during normal operations.
  - Fault tolerance - Ability of the system to handle corrupt, incomplete, or irrelevant data without breaking down.

- **Performance efficiency**
  - Time behavior - how quickly the system responds to the demands made from it.
  - Resource utilization - Which and how many resources are used by the system to perform a function.

- **Maintainability**
  - Analyzability - The degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to

be modified. In the case of AI, analyzability also refers to the ability to be able to understand why the system took the decision that it took. Ideally, explainablity (or examinability) should be a separate quality characteristic for ISO 25010 for AI systems.

- Testability - The degree with which the AI component supports testing in a given context. The higher the degree/the testability, the easier it is to find bugs by means of testing

Additional important parameters are:

- **Complexity** — Time and space complexity
- **Scalability** — The ability of the system to handle more load by adding additional resources to it
- **Continuous learning** — The ability of the system to continuously learn from new data, especially from real time environment data

2.3.2 Extended Quality Characteristics Specific to AI

LO-2.3.2	K1	Recall the quality characteristics specific to AI testing in addition to those mentioned in ISO25010 - intelligent behavior, morality and personality.

The use of AI has required an extension of the standard quality characteristics.

An intelligent machine should also bear the following quality characteristics, apart from those mentioned in ISO25010. [TDA]

- Intelligent behavior — Intelligent behavior is the ability to comprehend or understand. It is basically a combination of reasoning, memory, imagination, and judgment; each of these faculties relies upon the others. Intelligence is a combination of cognitive skills and knowledge made evident by behaviors that are adaptive. The sub-characteristics are: Ability to learn, Improvisation, Transparency of choices, Collaboration and Natural interaction.
- Morality — Morality, in relation to AI, is about the principles concerning the distinction between right and wrong or good and bad behavior. The sub-characteristics are: Ethics, Privacy and Human friendliness.

- Personality — Personality is the combination of characteristics or qualities that form an individual's distinctive character. The sub-characteristics are: Mood, Empathy, Humor and Charisma.

Chapter 3 - Offline Testing of AI Systems

Keywords: structured data, unstructured data, dimensionality, metrics, cross-validation, underfitting, overfitting, analytics

LO-3.1.1	K1	Recall the steps involved in the data preparation and preprocessing, and the need for the comprehensive clean-up of data.
LO-3.1.2	K3	Apply the data reading and manipulation, including filtering steps for structured data.
HO-3.1.2	H2	Make use of code (language of choice) to read data from various data sources such as an excel file, a CSV file or a database and clean up a given dataset by dropping the least important column(s), adding columns and cleaning up the data for structured text data (data manipulation).
LO-3.1.3	K2	Summarize various data preprocessing steps for unstructured data (images).
LO-3.1.4	K2	Summarize various data preprocessing steps for unstructured data (text).
HO-3.1.4	H1	Perform data preprocessing steps on unstructured data (text).
LO-3.1.5	K3	Apply various data imputation methods to different types of problems.
HO-3.1.5	H3	Fill missing data values for given dataset using mean, mode, KNN method.
LO-3.1.6	K1	List various types of plots (uni, bi and multivariate) used for data visualization.
LO-3.1.7	K2	Explain the concept of outliers and various reasons for their presence.
LO-3.1.8	K3	Apply visual box plot method to determine outliers from a given dataset.
HO-3.1.8	H2	Draw a box plot for the given dataset to identify outliers.
LO-3.1.9	K3	Apply dimensionality reduction techniques - irrelevant feature elimination, principal component analysis (PCA).
HO-3.1.9	H3	Perform irrelevant feature elimination, PCA for

		dimensionality reduction on a set of given data.
LO-3.2.1	K2	Explain the role of metrics in ML systems.
LO-3.2.2	K1	List various model evaluation parameters for both supervised and unsupervised learning.
LO-3.2.3	K2	Compare inertia and adjusted Rand score as metrics for unsupervised clustering.
HO-3.2.3	H3	Apply inertia - within-cluster-sum-of-squares (WCSS) — and use the elbow method to determine the optimum number of clusters.
LO-3.2.4	K2	Compare support, confidence and lift metrics for the unsupervised association rule mining.
HO-3.2.4	H3	Calculate support, confidence and lift metrics for the unsupervised association rule mining.
LO-3.2.5	K2	Explain the confusion matrix.
LO-3.2.6	K3	Apply the formula to calculate accuracy, precision, recall, specificity and F1-score metrics for supervised classification and compare them.
HO-3.2.6	H3	Calculate accuracy, precision, recall, specificity and F1-score for supervised classification.
LO-3.2.7	K3	Apply the formula to calculate various metrics (root mean square error (RMSE) and R-square) for supervised regression.
HO-3.2.7	H2	Calculate RMSE and R-square errors for supervised regression.
LO-3.3.1	K1	Define the need for validating the models by splitting the available dataset into training, validation and testing datasets.
LO-3.3.2	K2	Summarize the concept of underfitting and overfitting and bias-variance tradeoff.
HO-3.3.2	H0	Demonstrate underfitting and overfitting to show bias-variance tradeoff.
LO-3.3.3	K3	Apply split test, K-fold cross-validation, bootstrap and leave-one-out cross-validation methods.
HO-3.3.3	H1	Apply the K-fold cross-validation method on an algorithm

		and compare the various outcomes.
LO-3.4.1	K1	Recall the characteristics of the four types of analytics — descriptive, exploratory, predictive, prescriptive — and their use in ML.

3.1 Data Preparation and Preprocessing

3.1.1 Steps of Data Preparation and Preprocessing

LO-3.1.1	K1	Recall the steps involved in data preparation and preprocessing and the need for comprehensive clean-up of data..

Input data could be in the form of database tables, CSV (comma separated values) files, or it could be unstructured data such as images, audios, videos or running texts. The required data is acquired from various sources, internal and external.

After acquisition, the data needs thorough cleanup and some processing before it can be fed to the algorithms for training and testing.

The following steps are performed for data preparation and preprocessing

- Data manipulation
- Data filtering
- Data preprocessing activities include
    - Data Imputation, i.e. Dealing with Missing Values
    - Data visualization to get a big picture and Treating Anomalies and Outliers
    - Correlation Analysis and Reducing Dimensions

Several data format (e.g., image, text) specific preprocessing steps need to be performed to make data format suitable for training. When the data volume is too large, perform data-reduction without losing the information. For structured data missing, data values may need to be filled. All such data preprocessing steps are required to obtain desired accuracy and better predictability in the models.

3.1.2 Data Preparation

LO-3.1.2	K3	Apply the data reading and manipulation, including filtering steps for structured data.

HO-3.1.2	H1	Make use of code (language of choice) to read data from various data sources such as an excel file, a CSV file or a database and clean up a given dataset by dropping the least important column(s), adding columns and cleaning up the data for structured text data (data manipulation).

Data preparation includes:

1. **Data manipulation**: Changing the structure of the given data for example, adding a new column, dropping some rows, etc.
2. **Data filtering**: Reducing the size of both structured (table/matrix) and unstructured (image/text) data for improving data quality.

3.1.3 Processing of Unstructured Data (Images)

LO-3.1.3	K2	Summarize various data preprocessing steps for unstructured data (images).

Removing noise from the image and resizing it are the common operations carried out on images for designing computer vision algorithms. [UDI]

3.1.4 Processing of Unstructured Data (Text)

LO-3.1.4	K2	Summarize various data preprocessing steps for unstructured data (text).
HO-3.1.4	H1	Perform data preprocessing steps on unstructured data (text).

Text data preprocessing can be done in multiple steps of syntactic changes depending on the need of the ML model. For example, removing numerals, conversion of uppercase to lowercase, removing punctuations, white spaces, removing stop words, perform stemming/lemmatization, etc. [UDT]

3.1.5 Data Imputation

LO-3.1.5	K3	Apply various data imputation methods to different types of problems.
HO-3.1.5	H3	Fill missing data values for given dataset using mean, mode, KNN method.

Data collected from the field may have null or missing values requiring replacing null values with some appropriate values. Null or missing values can be imputed with measures of central tendency (mean, median or mode), K-Nearest-Neighbor [DI1] method, or a regression-based approach.

3.1.6 Data Visualization

LO-3.1.6	K1	List various types of plots (uni, bi and multivariate) used for data visualization.

Visualizing the data helps understanding its structure and the relationship among its attributes, which is not possible by merely looking at the numbers or text provided. There are various types of visualizations. The most commonly used visualization methods are: line plots for continuous values, histograms for discrete ones, box plots, bar charts, pie charts, etc. They give meaningful insight of the available data at the first go.

Types of plots:
Univariate: The simplest form of analysis, in which the data being analyzed is a single variable. For example, the age of a population or the weight of a population, etc. They are analyzed individually and their relationship is never considered. Line plots, histograms, frequency distribution, bar charts and box plots are utilized for the analysis of univariate data.

Bivariate: This kind of analysis is carried out to find the relationship between two variables in the given data set. Plotting one variable against another on a XY plane helps to find the first hand relationship between two variables. For example, the relationship between the age and weight of the population under consideration. For this kind of analysis, you can use scatter plots or correlograms.

Multivariate: The analysis of three or more variables. Mesh plots and 3D plots are some of the ways one can visualize multivariate data and discover relationships among them.

3.1.7 Anomaly/Outliers Detection

LO-3.1.7	K2	Explain the concept of outliers and the various reasons for the presence of outliers.

Observations which do not follow the expected pattern for a given data set fall into the category of outliers, for example, fraud detection or hack attacks.
If outliers are not frequent and do not contribute in the critical events, then they can be removed. But in practice, they should be thoroughly investigated before pruning them from the dataset.

Causes of Outliers

- Error: Outliers in this case are outcomes of an error in the measurement, data entry and sampling, e.g. the temperature data recorded in Celsius for most records but for a few others, in Fahrenheit, by mistake.
- Natural: Some outliers can occur in a natural situation, e.g., if a flood incident occurs once in 100-years, it is a natural outlier.
- Intentional: Dummy outliers made to validate detection methods, e.g., lab-grown artificial records used for testing corner scenarios.

3.1.8 Outliers Detection Techniques

LO-3.1.8	K3	Apply visual box plot method to determine outliers from a given dataset.
HO-3.1.8	H2	Draw a box plot for the given dataset to identify outliers.

While computing various statistics with the given data set, often it is helpful to make use of a visual box-plot to see the data distribution. Box-plots help determine the outlier position of the given data set.

The ends of the box are the upper and lower quartiles. The whiskers are the two lines outside the box that extend to the highest and lowest data thresholds beyond which datapoints are considered to be outliers.

3.1.9 Dimensionality Reduction

LO-3.1.9	K3	Apply dimensionality reduction techniques — irrelevant feature elimination, principal component analysis (PCA).
HO-3.1.9	H3	Perform irrelevant feature elimination, PCA for dimensionality reduction on a set of given data.

ML problems often have large numbers of input features but not all of them contribute to the classification or regression output. The higher the number of features, the harder it is to visualize the training set. The technique of reducing the number of variables under consideration is called dimensionality reduction.

The need to work with fewer dimensions than the original dimensions arises from:

- Cost and speed factors
- Memory requirement
- Avoiding redundancy
- Identifying the most relevant part of the data for further processing

Methods of Dimensionality Reduction:

- Irrelevant feature elimination is removing columns that are not contributing to the output variable:
    - Univariate analysis helps to remove the columns whose value does not change across rows. For instance, consider the sales transaction data from a single retail store, then columns such as store name, store location would not change across rows and should be removed.

    - Too low-density data can easily be dropped after investigation to save space. For example, database row ID based columns have a unique value for every row and should be removed.

- Bivariate analysis removes one among the pair of highly correlated input attributes (thus, it is also known as correlation analysis), for instance, in a store inventory data, 'item price' and 'item quantity' are highly correlated attributes.

- Principal component analysis: PCA reduces the dimensions of larger datasets extensively, yet preserves the information to the maximum. It deduces a new set of independent variables (called principal components) and puts them in order of reducing significance. The required number of top principal components (thus, reduced number of variables or dimensions) can then be selected still preserving the maximum possible information of the original dataset.

3.2 Metrics

3.2.1 Role of Metrics

LO-3.2.1	K2	Explain the role of metrics in ML systems.

Metrics are evaluation parameters for a trained model and can be seen as the measurement of how far the trained model delivers accurate and reliable results. For a given type of algorithm, metrics can be used to compare trained models with each other.

3.2.2 Metrics for Supervised and Unsupervised Learning

LO-3.2.2	K1	List various model evaluation parameters for both supervised and unsupervised learning.

The problem objectives for supervised and unsupervised learning models are different. Thus, the metrics to evaluate the models are different.

Learning type	**Model type**	**Metrics used**
Unsupervised	Clustering	• Inertia • Adjusted Rand score
	Association	• Support • Confidence • Lift
Supervised	Classification	• Accuracy • Precision • Recall/Sensitivity • Specificity • F1-score
	Regression	• Root-mean-square-error (RMSE) • R-square error

3.2.3 Inertia and Adjusted Rand Score

LO-3.2.3	K2	Compare inertia and adjusted Rand score as metrics for unsupervised clustering.
HO-3.2.3	H3	Apply inertia — within-cluster-sum-of-squares (WCSS) — and use the elbow method to determine the optimum number of clusters.

For an unsupervised clustering-based model, inertia or WCSS (within-cluster-sum-of-squares) is the average spread of a cluster across all the discovered clusters. The smaller Inertia value means better clustering, as it means that the data points within a cluster are closer to each other. The cluster size (and thus, the value of Inertia) will go down naturally as the number of clusters goes up. However, Inertia stops decreasing significantly beyond a certain number of clusters; this point shows the optimum value for Inertia and the number of clusters for a given dataset — this method is known as the elbow method.

When the actual values of the labels are available for every data point, the **adjusted Rand score** is preferred over Inertia. It is a measure of similarity between the cluster assignments (by the model) and the actual separate classes.

3.2.4 Support, Confidence and Lift metrics

LO-3.2.4	K3	Apply formula to calculate support, confidence and lift metrics for unsupervised association rule mining.
HO-3.2.4	H3	Calculate support, confidence and lift metrics for unsupervised association rule mining.

Support for an itemset measures how frequently it is appearing in transactions. For example, if the item ‘bread’ is present in 7 out of 10 total transactions at a retail store, its support is 70%.

Confidence measures the likelihood of itemset Y appearing, given that X has appeared.

Lift is used to eliminate scenarios where two itemset are occurring together very frequently (thus, the confidence metric value will be high). However, the two itemset may not have any interdependency. Besides, this metric can reveal if more occurrence of itemset X means more or less occurrence of itemset Y (i.e., positive or negative association between X and Y).

3.2.5 Confusion Matrix

LO-3.2.5	K2	Explain confusion matrix.

Supervised classification metrics are computed using a confusion matrix made-up of the counts of true positives, false positives, true negatives, false negatives.

Confusion Matrix	Target Positive	Target Negative
Model Positive	True Positive TP	False Positive FP
Model Negative	False Negative FN	True Negative TN

3.2.6 Accuracy, Precision, Recall, Specificity and F1-Score

LO-3.2.6	K3	Apply the formula to calculate accuracy, precision, recall, specificity and F1-score metrics for supervised classification and compare them.
HO-3.2.6	H3	Calculate accuracy, precision, recall, specificity and F1-score for supervised classification.

The **Accuracy** of a model shows what percentage or fraction of total classifications were done accurately by the trained model on a test dataset. This metric becomes a poor choice of metric if one class of data dominates over the others.
Accuracy = (TP + TN) / (TP +TN + FP +FN).

Precision measures how accurately the model classifies true positives.
Precision = TP/ (TP+FP).

Recall measures how far the model failed or missed to detect the positives.
Recall = TP/ (TP+FN).

In order to minimize false positives, high precision is required, whereas to minimize false negatives, recall should be high.

Like precision but opposite to recall, **specificity** measures how accurately the model classifies true negatives.

Specificity = TN / (TN + FP)

F1-score is computed as the harmonic mean of precision and recall. A low F1-score represents the poor quality of the model at detecting positives. F1-score will have a value between 0 and 1. Close to 1 means that there is good quality and no false data disturbing the result.

3.2.7 RMSE and R-Square

LO-3.2.7	K3	Apply the formula to calculate various metrics (root mean square error (RMSE) and R-square) for supervised regression.
HO-3.2.7	H2	Calculate RMSE and R-square errors for supervised regression.

The supervised regression model metrics represent how well the regression line fits the actual data points.

RMSE (root-mean-square-error) is a measure of how far the data points are from the regression line. It is measured as the standard deviation of prediction errors. The value of RMSE changes if the same dataset is measured in a different unit.

R-square is a measure of how better the predictions by the regression line are compared to using mean as a predictor. Its value ranges from 0 to 1 and is independent of the unit used to measure data points.

3.3 Model Evaluation

The metrics values depend on how the datapoints for training and validation are chosen. Thus, it becomes the key aspect for model evaluation that datapoints for training and validation are selected in a completely unbiased way.

3.3.1 Training, Validation and Testing Datasets

LO-3.3.1	K1	Define the need for validating the models by splitting available dataset into training, validation and testing datasets.

The training dataset contains the data that the model is trained on. The validation dataset is used by the machine learning algorithm to evaluate if the training was effective. In every run of ML (which is a process of many iterations), the training dataset and validation dataset are combined again and split in a different way so that the algorithm uses different combinations of data to learn from.

The testing dataset is a separate dataset that is used after the ML process is finished to validate whether the algorithm has been adequately trained. The testing dataset should not be used during the training process. [Wiki1]

In the post-training phase, an ML model is evaluated and tested with a dataset different from its training dataset. However, the dataset for training, validation and testing phases must come from the same or similar source(s) to ensure the effectiveness of metrics. A model trained using one kind of dataset may perform very poorly on a dataset originated from very different source(s).

3.3.2 Underfitting and Overfitting

LO-3.3.2	K2	Summarize the concept of underfitting and overfitting and bias-variance tradeoff.
HO-3.3.2	H0	Demonstrate underfitting and overfitting to show bias-variance tradeoff.

If a supervised ML model is too simplistic to fit the training data points (i.e., it fails to represent the data trend) it is an example of underfitting. Conversely, an overfitting model tries to fit the training data points too much and this often results in poor prediction accuracy during the subsequent validation or testing phases.

The underfitting and overfitting nature of a model can also be explained in terms of Bias and variance errors. If a model is oversimplified and does not learn from all the provided features to represent, it is said to have high-bias and suffers from poor prediction accuracy.

If the model prediction performance varies highly by changing the training dataset slightly, it is said to be a model of high variance (too dependent on the training dataset). A good model has to achieve low bias and low variance. This is known as the bias-variance tradeoff.

3.3.3 Cross-validation methods

LO-3.3.3	K2	Explain split test, K-fold cross-validation, bootstrap and leave-one-out cross-validation methods.
HO-3.3.3	H1	Apply K-fold cross-validation method on an algorithm and compare various outcomes.

The way the available dataset is split into training and validation datasets may lead to high bias or high variance. In order to overcome this, multiple split combinations must be tried out before concluding model metrics. Some useful methods are split-test, bootstrap, K-fold cross-validation and leave-one-out cross-validation. Each of these methods repeats the training and validation process several times and the model performance is averaged out across all runs.

Split-test divides the data into parts for training and testing datasets but in each iteration in different ratios. This helps revealing how different splits can produce different results.

Bootstrap works by picking random data points from the entire dataset for training and uses the remaining dataset for validation.

K-fold cross-validation divides the dataset into k parts and uses k-1 subsets for training and remaining subset for validation.

Leave-one-out cross-validation is similar to K-fold cross-validation, except that each part has one data point, so it requires more execution time.

3.4 Analytics

3.4.1 Types of Analytics

LO-3.4.1	K1	Recall the characteristics of the four types of analytics — descriptive, exploratory, predictive, prescriptive — and their use in ML.

Analytics is one of the primary tasks when it comes to understanding the available dataset. Data analytics can be done at four levels and each subsequent level is a natural extension to the previous level.

Descriptive analytics is about deducing the statistical summary of data in terms of measures for central tendency (mean, median, mode) as well as measures for dispersion (variance, standard deviation). This helps getting insight about the past and answer: "What has happened?" [DA1]

Exploratory analytics is about visualizing the dataset at a high level to see its patterns and variations.

Predictive analytics is modeling the input variables and predicting the probability of outcomes.

Prescriptive analytics is comparing all viable predictions resulting from predictive analytics and choosing/prescribing the best among them.

Chapter 4 - Online Testing of AI Systems

Keywords: Linguistic Analysis method, chatbot

LO-4.1.1	K2	Explain the AI and non-AI parts of an intelligent application and its functional and non-functional testing needs.
LO-4.1.2	K1	Show the information-oriented and action-oriented interactions of AI and non-AI parts.
LO-4.2.1	K3	Make use of the linguistic analysis test design method to generate test scenarios.
HO-4.2.1	H1	Demonstrate the use of the linguistic analysis test design method to generate test scenarios.
LO-4.3.1	K3	Make use of the test cases derived from the given requirements and architecture to test a chatbot.
HO-4.3.1	H3	Test a given chatbot at system level and report bugs.

4.1 Architecture of an AI application

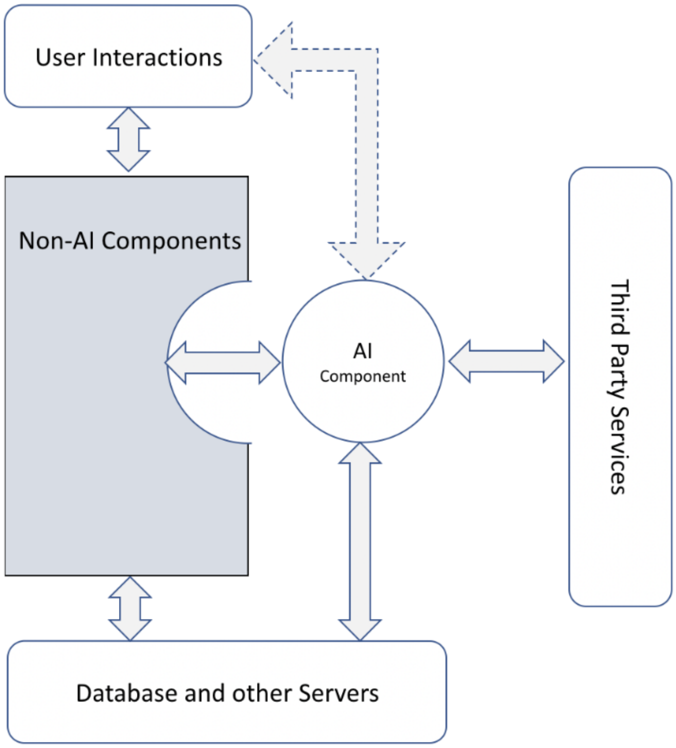
4.1.1 Components of an Intelligent Application and their Testing Needs

LO-4.1.1	K2	Explain the AI and non-AI parts of an intelligent application and its functional and non-functional testing needs.

A typical AI application needs to be examined, whether it is a monolithic AI application, or an overall system comprised of a smaller set of AI components invoked from a larger non-AI application. This analysis is required to understand how various components, AI and non-AI, work together to provide the desired functionality. The interfaces between these various components need to be understood from the point of view of:

- Input data being passed
- Output generated
- Actions taken by the respective components
- Direct user interactions, if any

- Third-party system interactions, if any
- Frequency of invocation
- Number of parallel invocations, if possible
- Constraints such as
  - Timing
  - Duration
  - Ranges of inputs and outputs
- Necessary pre-conditions
- Assumptions/common settings
- Visualizing the chain of inputs and outputs with relevant transformations, if any
- Error and exceptions and their handling by
  - The component
  - Chain of components and final handling
- Logging



Furthermore, there is a need for the explainability of the reason for the system to arrive at the given solution. In addition, there is a need for identifying test scenarios for these AI systems using techniques such as linguistic analysis and exploratory testing.

The other issue of complexity which needs to be ensured is the presence or absence of combinations of AI systems. Single standalone AI systems may be insufficient to completely specify a real-world problem. To handle these scenarios, combinations of AI systems are used to model the problem. To that end, a test strategy to handle combinations of AI systems is required.

In AI - non-AI interaction, there is a need to ensure the test coverage of the AI component, the non-AI component, and the interaction involving handover between the two parts.

An example of testing such a system is provided here:
In an email application, sentence completion is provided by an AI component. The user interface of the email system represents the non-AI part and the AI component interacts with it to provide the suggested text. The non-AI part displays the suggestion and acts on the user input. If the suggestion is accepted, this might result in the storage of some data for future learning.

The same email system also provides inline spell checking, provided either by an AI or non-AI component.

If we analyze the interfaces, we find that the UI (non-AI component) is passing on the partially written text to the AI component. If there is a suggestion from the AI component, the text is displayed by the UI. Constraints related to timing and duration are that the suggestion has to be made within a few hundred milliseconds of the text being entered and this has to be a continuous process. The tester needs to determine if the system is responding fast enough. If there are timing and duration issues, such as the model predictions being slow, by the time the system shows the suggested sentence, the user may have written additional text. This may make the suggested sentence incorrect grammatically or the wrong suggestion all together.

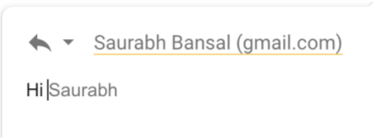
An example of issues in the interaction of the GUI and the non-GUI (AI) parts of an output can be related to the display of the suggested text. The suggested text may be displayed at an incorrect place in the GUI. The suggested sentence needs to be visually differentiated from the actual text being written and needs to change based on user actions. Any errors in these behaviors would also fall under the GUI and non-GUI (AI) interactions.

An example of the error/exception handling part is the scenario where the sentence completion component fails (for some reason) and the GUI is able to handle it.

The settings of the email system include language settings and dictionary settings. For example, English as the language setting, and US English as the dictionary setting. The sentence completion system should provide US English suggestions. The spell and grammar check system should ideally use US English as well, in this case. Which means that the sentences formed by the AI component should not be marked incorrect text by the spell check component because of a mismatch in the assumption/settings of the two interacting AI or AI - non-AI components.

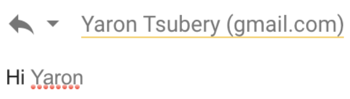
The images below show some examples of issues discovered on such a system, A and B show the marking of one name as a spelling error and not the other, whereas both names are part of the address book.

On the other hand, B and C show the difference of speed and caching. When typed for the first time, the name completion happens but the name is not marked as an error for a few seconds and then it gets marked as an error. However, once marked as an error, then even deletion of the name and subsequent auto-completion marks this as an error immediately.


Saurabh Bansal (gmail.com)

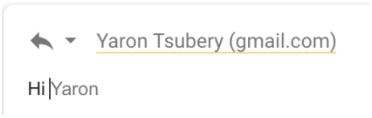
Hi Saurabh

Image A


Yaron Tsubery (gmail.com)

Hi Yaron

ImageB


Yaron Tsubery (gmail.com)

Hi Yaron

ImageC

4.1.2 Interaction of AI and Non-AI Parts

LO-4.1.2	K1	Show the information-oriented and action-oriented interaction of AI and non-AI parts.

In the information-oriented calls, the API or service of the AI application is invoked from an end application which can be non-AI. Typically, the AI component is invoked for a response. For example, a fraud detection algorithm works on the idea of calling a pre-defined trained model to return whether an input transaction is a fraudulent one or not.

Some of the important tests for testing the interactions are related to:

- Boundary value-based tests — both input and output
- Unusual test cases (a.k.a. Corner test cases)
- Tests related to the size and type of data to be passed
- Exception handling tests
    - Response not received
    - Broken request-response feedback loop for actions
- Erroneous input data tests
- Erroneous output data tests
- Request could not be completed
- Response not received
- Performance related tests
- Security related tests
- Robustness related tests

The interactions between the AI and non-AI components may have an impact on the experience of the user. The interactions can be of the following type:

- Flag only (just feed information to the system)
- Action oriented, e.g., generate a response for the user, OR classify the problem
- Scenarios of failure of interacting APIs where the APIs fail to return a result
- Handover from AI to non-AI components and vice-versa

When tests are designed for the AI systems, these are of following test levels:

- Tests that just test the AI part
- Tests that just test the non-AI part
- Tests for integration of both parts
- Cached response vs. learned responses

In a larger non-standalone system, there is a need to look at the coverage of the deployed AI model as well as the performance management of the deployed AI model. After the development of AI component, the system needs to be tested in a deployment environment.

4.2 Linguistic Analysis Test Design Method

4.2.1 Linguistic Analysis-Based Test Design

LO-4.2.1	K3	Make use of the linguistic analysis test design method to generate test scenarios.

Linguistic Analysis is used to design a large number of scenarios even when the requirements are poorly documented [LA1]. A linguistic analysis of the requirements identifies the test objects, and the actions to be taken on them. This method helps finding corner (unusual) test cases, a key requirement for testing AI systems.

The method works as below:

1. Identify the nouns which represent the test objects and the verbs that represent the actions.

2. Identify the properties of the noun.

3. Identify the properties of the properties and repeat it until no more properties can be discovered or you think sufficient depth has been reached.

4. Identify adverbs and adjectives applicable to the nouns and verbs to identify more properties.

5. Use 5W1H (What, Why, Where, When, Which, How) on the verbs to identify the noun-verb combinations giving you the functional and non-functional tests.

6. Ask the following questions of every scenario:
   a. Who/what is the agent?
   b. What are the instruments/means to accomplish the scenario?
   c. What is the purpose of the action?
   d. What is the direction of the action?
   e. What is the source of the action?
   f. What is the motive of the action?

g. Who possesses the means and the results of the action?
h. Where does the action take place?
i. Who is the receiver of the action?

These questions follow the grammar rules of Sanskrit [LA2].

In case of testing AI systems, the data is the test case. The given method allows the identification of both, the data as well as the combination of actions (scenarios).

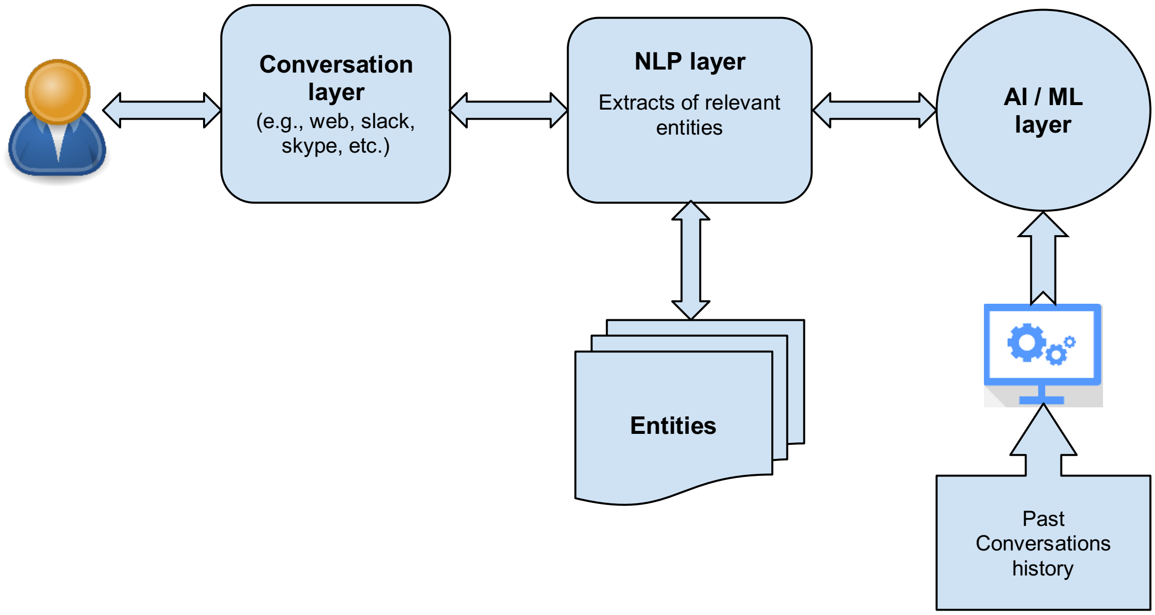
4.3 Testing AI systems

4.3.1 Test a Chatbot

LO-4.3.1	K3	Make use of the test cases derived from given requirements and architecture to test a chatbot.
HO-4.3.1	H3	Test a given chatbot at system level and report bugs.

Chatbots are used today in several contexts, ranging from customer service bots, information bots, to support bots.

The logical architecture shows the key layers of a chatbot, including the conversation layer, NLP layer, AI/ML layer and connectivity layer. The conversation layer manages the front-end interaction with the end user for example, Skype, Slack, Facebook messenger, or a web frontend. The NLP layer extracts the relevant entities and information from the end user utterances. The AI/ML layer decodes the actual intent of the utterance, having been trained on past historical conversations fed as data to the system.


Conversation layer
(e.g., web, slack, skype, etc.)
NLP layer
Extracts of relevant entities
AI / ML layer
Entities
Past Conversations history

Test cases should cover input data variations at each individual layer. The conversation layer needs to be tested for the right data transmission from front-end to the backend and vice-versa. At NLP layer, one should test if the data-cleaning and preprocessing of raw input text utterances is happening properly and if the extracted entities are enough to convey the context in an ongoing conversation. The AI/ML layer or model (trained on historical conversations) should be tested for its accuracy of predicting the correct intent based on extracted entities (by the NLP layer), e.g., for an input text 'Hi' / 'Hello' / 'Greetings', etc. 'Welcome' intent should be predicted.

In addition, the chatbot should be able to jump between various intents seamlessly if the conversation flow sequence changes abruptly. So, the test cases covering dynamic switching of intent should be identified.

Nonfunctional requirements are to be tested for the performance of the frontend primarily, whether it is able to withstand many concurrent customer invocations.

Chapter 5 - Explainable AI

Keywords: interpreting/explaining the ML model, explainable, LIME, CAM

LO-5.1.1	K2	Explain the need for explainable AI (XAI).
LO-5.1.2	K3	Make use of LIME to explain an AI model.
HO-5.1.2	H1	Make use of LIME on an image classifier as well as a text classifier.
LO-5.1.3	K3	Make use of Grad-CAM to explain CNN models.
HO-5.1.3	H1	Make use of Grad-CAM to make CNN more transparent.

5.1 Explainable AI (XAI)

5.1.1 Explainable AI and its Need

LO-5.1.1	K2	Explain the need for explainable AI (XAI).

Once an ML model is trained, it should work with a defined level of accuracy and for all the defined variations of scenarios. If the quality of the predictions by the model is insufficient for some scenarios and work fantastically for others, it is likely to be a biased model. A dataset for which the model quality goes below the defined level of quality represents a defect.
As a tester, it is very difficult to discover all such defects without using a systematic approach. One needs to examine the behavior of the model and its variation with change in input factors to deduce an approximation of the relationship between input and output. This kind of deduction is called interpreting or explaining the ML model. Such approximate relationship may not be a replacement of the actual model, but it may be good enough to reveal possible biasing. An insight into the model behavior helps evaluating its overall quality and the model's viability for deployment. Other reasons for requiring explaining or interpreting models can be safety measures, social acceptance, detecting bias, or human curiosity and learning about the model [EA1].

Not all models can be explained. The more complex the model, the less likely it is to interpret or explain it. The output of non-DL models, e.g. random forest

[Wiki2], decision trees [Wiki3], linear regression [Wiki4] etc., can conveniently be explained in terms of the input variables.

The DL models are inherently complex in their implementation, so it is better to examine the model as a black-box, i.e. observe the variations of outcomes by small perturbations of the input variables and approximate the underlying model by a simple, interpretable model.

Some of the popular and easy-to-use algorithms, tools and approaches for model interpretation include: Local Interpretable Model-agnostic Explanations (LIME) [EA2] and Class Activation Maps (CAM) [EA3].

5.1.2 LIME

LO-5.1.2	K3	Make use of LIME to explain an AI model.
HO-5.1.2	H1	Make use of LIME on an image classifier as well as a text classifier.

LIME is fed a sample to study model predictions for the sample and its closer variations and it reveals the input features responsible for the model predicted output. LIME generates enough close variants of the sample input and gets outcome for each of the variants. Thus, it attempts to approximate how small variations in input are changing the output variable. Since all variants generated and studied by LIME are in close vicinity of the given sample, LIME explanations are called ‘local’ (to the sample input). Also, it is a model-agnostic method, as LIME does not require to peek into the model or algorithm.

LIME can be used on an image classifier to generate explanations. It deduces image portions that play a significant role in determining the outcome. It allows a tester to relate as well as to rule out that the model is not basing its deductions on irrelevant features.

Similarly, for a text classifier, LIME can point out the words (text portions) that lead to the categorization of the sample text into one of the predefined classes. This helps a tester to assess whether the model is basing its deductions on irrelevant words.

5.1.3 CAM for Neural Networks

LO-5.1.3	K3	Make use of Grad-CAM to explain CNN models.
HO-5.1.3	H1	Make use of Grad-CAM to make CNN more transparent.

CNNs are very useful but they are not transparent about why they reached a particular conclusion. In order to bring transparency to such models, Gradient-based Class Activation Mapping (Grad-CAM) visualizes the input regions that are important for predictions from these models. It uses the class-specific gradient information flowing into the final convolutional layer of a CNN and produces a coarse localization map of the important regions in the image.

Chapter 6 - Risks and Test Strategy for AI Systems

Keywords: pre-trained model, concept drift

LO-6.1.1	K1	Recall the risks of testing AI systems.
LO-6.1.2	K2	Explain the issues associated with the use of third-party pre-trained models in production systems.
LO-6.1.3	K2	Explain the need for testing the trained model again because of concept drift.
LO-6.1.4	K1	Define test environment for AI systems and recall its challenges.
LO-6.2.1	K1	Recall test strategies for AI applications, especially considering types, phases, and levels of testing.

6.1 Risks in testing AI

6.1.1 Risks of Testing AI Systems

LO-6.1.1	K1	Recall the risks of testing AI systems.

Testing AI systems poses some additional risks, in addition to the risks associated with non-AI systems. Some of these risks are

- Test condition related challenges
  - With large number of tests (each dataset being a test), how does one verify
    - correctness of tests
    - completeness of tests
  - Lack of a reliable test oracle to indicate what the correct output should be for and arbitrary input
  - Identification of false positives and false negatives. False positives are still easier to identify because failures are to be investigated. False negatives are difficult to identify because these tests are shown as pass
- Test data related challenges
  - In AI test data is equal to a test case, especially for offline testing. A huge amount of test data is ideally required to test.
  - Data availability can be a problem
  - Data quality can be bad, requiring data cleanup

  - Tagged/annotated data is required for training and testing. Getting this type of data is expensive.
  - Corner cases of ML systems are tough and costly to generate
- Cost related challenges
  - For complex applications, the type of hardware required to train and test the offline model may be expensive
  - Energy costs of training DNN is generally very high
- Skill and tool related challenges
  - Skill requirements for testing AI systems are high. The tester needs to understand how AI systems are built and how they need to be tested
  - Lack of structured information, tools and frameworks for AI systems testing
- Domain understanding and bias related challenges
  - Formal proofs of an algorithm's optimal quality do not guarantee that an application implements or uses the algorithm correctly
  - Need for thorough coverage of input cases, based on domain, are required to avoid biases, incomplete coverage and potential for accidents

6.1.2 Risk of Using Pre-Trained Models

LO-6.1.2	K2	Explain the issues associated with the use of third-party pre-trained models in production systems.

Some organizations have made their pre-trained models available and many AI developers make use of them.

For example, ImageNet models such as Inception, VGG, AlexNet, etc. The models may have their own biases and shortcomings which may need to be discovered by testing. Since these are either unknown or undocumented, systems built using pre-trained models may fail in unanticipated ways or produce results that may be sub-optimal in some situations.

6.1.3 Risk of Concept Drift (CD)

LO-6.1.3	K2	Explain the need for testing the trained model again because of concept drift.

Working models may degrade over time because of the change in the relationship between input features and output. For example, the impact of

commercials and various other types of marketing campaigns results in a change of the behavior of potential customers every few months. This is known as Concept Drift (CD). To find out the occurrence of concept drift, periodic testing of working and integrated models with recent data samples is required. For this purpose, retraining the model and repeating the cycle of various tests and validations is performed in the offline and integrated phases. [JB2]
There are various ways to handle concept drift, such as:

- Do nothing
- Retrain the model with recent data
- Periodically update the model by using the latest data on the current model
- Learn the change — An ensemble approach where the current model is left unchanged. A new model takes the output of the current model and learns to correct the predictions
- Detect and choose model — detect the concept drift, if possible, and choose an appropriate model

6.1.4 Challenges of AI Test Environment

LO-6.1.4	K1	Define test environment for AI systems and recall its challenges.

The test environments for AI systems can be very complex, owing to possible different use-cases, contexts, and various ways and steps of data preprocessing. From the offline testing point of view, the environment needs are more demanding than from the online testing point of view. There is a need for a large size of data storage, high network bandwidth requirement and also for greater computational power to train/run the model.

6.2 Test Strategy

6.2.1 Test Strategy for Testing AI Applications

LO-6.2.1	K1	Recall test strategies for AI applications especially considering types, phases and levels of testing.

A real-world AI based application might employ one or more AI and non-AI components. The test strategy for such a system will include conventional test dimensions as well as new factors specific to AI components and their integration with other system components.

Some of the conventional test strategy considerations are:

- **Level of testing required for the system** - unit testing, integration testing, system testing, acceptance testing and system integration testing as described in ISTQB® FL syllabus. [ISTQB-FL2018]
- **Test techniques** - white-box, black-box and gray-box
- **Functional and non-functional tests** - especially security, performance, robustness and scalability tests
- **Use of test automation**

For testing AI based applications, one needs to take into account the following additional aspects:

- **Offline testing (functional)** - Testing the trained AI model is an additional step that needs to be performed as part of SDLC. The skill requirements for testing in this phase also needs to be considered.

- **Offline testing (non-functional)** - Testing the trained AI model for various non-functional aspects. The model needs to be tested for speed, resource utilization, concurrency and load, scalability before deployment.

- **Black-box testing** - most AI models are exposed as APIs and invoked as black-box generally. A critical component missing in AI systems is the lack of test oracle making black-box testing difficult. Techniques like metamorphic testing are getting popular for ML as it does not require an oracle. Given a successful run of test cases, variation to the test cases are performed and the outputs examined. The need for the output to only satisfy the relationships does not require any oracle. [Wiki5]

- **White-box testing** - In general, white-box testing for ML systems is difficult as, except for some simple models, the inner working of the model is neither clear nor accessible. However, recently, some white box techniques have emerged for ML systems. An example is DeepXplore, an automated white-box testing product. Using neuron coverage, the researchers were able to expose thousands of unique incorrect corner case behaviors. [DX1]

- **Data acquisition and preprocessing** - The available data may not always be in shape to be fed to a model for training or testing. This has been discussed in detail in 3.1 Data preparation and processing.

- **Converting of development environment implementations to production** - for example, converting Jupyter notebook into Python code that runs on the server inside a Docker container on a cloud instance. Once it has been done, the model generated can be saved as a file and used in the application.

Chapter 7 - AI in Testing

Keywords: Test automation

LO-7.1.1	K2	Explain how AI can assist in various testing activities —test planning and estimation, risk analysis, test case design and test data generation, defect assignment, impact analysis, coverage analysis.
LO-7.1.2	K2	Explain how AI can assist in reporting and smart dashboards.
LO-7.2.1	K2	Illustrate the use of various AI based test execution automation tools commercially available.
HO-7.2.1	H2	Make use of demo/trial versions of AI based test execution automation tools.

AI can be used for improving existing processes of testing in SDLC. The idea is to make use of the data generated in test processes to provide detailed analytics, more automation, deeper insights and patterns and ability to predict and take corrective actions.

7.1 AI for Software Testing Life Cycle (STLC)

7.1.1 AI for STLC Methods

LO-7.1.1	K2	Explain how AI can assist in various testing activities —test planning and estimation, risk analysis, test case design and test data generation, defect assignment, impact analysis, coverage analysis.

In the context of test planning and estimation, overall software project cost can be approximated, taking into account different inputs related to the AI projects, including data size, effort involved, platform choice, application type, data preparation time, training time, and testing time. Given the historical data of these inputs, ML models can be prepared to give more accurate estimations.

AI can help prioritize risks, obtain more accurate metrics associated with schedule adherence, and help identifying the performance metrics of applications more accurately.

For test design, using AI technologies, like natural language processing (NLP) and text mining, can help the automated test case generation from textual requirements documents. Additionally, AI applied on code analysis (both static and dynamic), along with the analysis of the data collected from tests, can flag potential issues of performance and other non-functional requirements. Moreover, running ML on past data can help identifying test data patterns and helps generating automated test data for both component tests and system tests.

In particular, for image data and GUI elements, AI can help identify incorrectly rendered elements automatically. Additionally, by an ML based data centric analysis of different possible flows, the right flows of data can be automated.

For automated defect prediction, models using ML can predict defects based on code quality metrics.

Impact analysis using ML on code can help automate the identification of impacted modules and files based on change.

In terms of coverage analysis using AI can help achieve comprehensive test and code coverage via the analysis of the data flows captured.

7.1.2 AI for Reporting and Smart Dashboards

LO-7.1.2	K2	Explain how AI can assist in reporting and smart dashboards.

In the context of reporting and dashboards, using ML helps to generate focused insights and summarized data for presentation in smart dashboards, instead of pure analytics.

7.2 AI based automation tools

7.2.1 Tools

LO-7.2.1	K2	Illustrate the use of various AI based test execution automation tools commercially available.
HO-7.2.1	H2	Make use of demo/trial versions of AI based test execution automation tools.

There are a large number of automation tools available in the market. Some of these tools use AI for making automation easier or more maintainable.

AI tools could use GUI Spiders which traverse the complete GUI and record the app. Over iterations, they are able to learn, compare and identify bugs.

Some tools combine elements from visual GUI tests and use ML to figure out the changes and possible correlation between locators and elements and whether a change is a bug or an expected change.

Some tools use NLP, some work at DOM level, some at UI, some go to the log, some combine functional and performance testing and some combine some of the above-mentioned approaches.

Image based tools may use AI based image classification tools to flag UI defects. These may be working as a web browser plug-in with record and playback. AI based image comparison can be superior to simple image comparison.

References

[AG1] Neural Networks and Deep Learning - By Aurélien Géron (O'Reilly)

[BT1] A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives - By L. Anderson, P. W. Airasian, and D. R. Krathwohl (Allyn & Bacon 2001)

[BT2]
https://www.apu.edu/live_data/files/333/blooms_taxonomy_action_verbs.pdf

[DA1] 4 types of data analytics to improve decision-making
https://www.scnsoft.com/blog/4-types-of-data-analytics

[DI1] Introduction to k-Nearest Neighbors
https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

[DSC1] https://www.datasciencecentral.com/profiles/blogs/the-data-science-project-lifecycle

[DX1] DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP '17 Proceedings of the 26th Symposium on Operating Systems Principles (Pages 1-18) http://www.cs.columbia.edu/~junfeng/papers/deepxplore-sosp17.pdf

[EA1] Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (Section 2.1 Importance of Interpretability) - By Christoph Molnar
https://christophm.github.io/interpretable-ml-book/agnostic.html

[EA2] "Why Should I Trust You?" Explaining the Predictions of Any Classifier
https://arxiv.org/pdf/1602.04938v1.pdf

[EA3] Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization
https://arxiv.org/pdf/1610.02391.pdf

[ISTQB-FL 2018] ISTQB Foundation Level Syllabus version 2018. Available at https://www.istqb.org/downloads/category/51-ctfl2018.html

[JB1] Machine Learning: Hands-On for Developers and Technical Professionals - By Jason Bell (WILEY 2014)

[JB2] A Gentle Introduction to Concept Drift in Machine Learning
https://machinelearningmastery.com/gentle-introduction-concept-drift-machine-learning

[KUR] The Age of Intelligent Machines - By Ray Kurzweil (MIT Press 1990)

[LA1] Visual Modeling for Test Idea Generation - By Vipul Kocher
https://www.testnet.org/testnet/download/preview-voorjaar-2015/visual-modeling-for-test-idea-generation-v2.pdf

[LA2] Vibhakti - By Ujjwol Lamichhane
https://www.scribd.com/doc/30377592/Vibhakti-%E0%A4%B5%E0%A4%BF%E0%A4%AD%E0%A4%95-%E0%A4%A4%E0%A4%BF

[SMU]The CRISP-DM User Guide
https://s2.smu.edu/~mhd/8331f03/crisp.pdf

[TDA] Testing in the digital age: AI makes the difference - By Tom van de Ven, Rik Marselis and Humayun Shaukat (Sogetibooks 2018)

[UDI] Image Preprocessing
https://towardsdatascience.com/image-pre-processing-c1aec0be3edf

[UDT] Text Preprocessing in Python: Steps, Tools, and Examples
https://medium.com/@datamonsters/text-preprocessing-in-python-steps-tools-and-examples-bf025f872908

[ULA] Unsupervised Learning: Association Rules - By Krzysztof J. Cios, Roman W. Swiniarski, Witold Pedrycz, Lukasz A. Kurgan
[Wiki1] https://en.wikipedia.org/wiki/Training,_validation,_and_test_sets

[Wiki2] Random forest
https://en.wikipedia.org/wiki/Random_forest

[Wiki3] Decision tree learning
https://en.wikipedia.org/wiki/Decision_tree_learning

[Wiki4] Linear regression
https://en.wikipedia.org/wiki/Linear_regression

[Wiki5] Metamorphic testing
https://en.wikipedia.org/wiki/Metamorphic_testing