Strategy for the GxP-validation of AI/Machine Learning solutions.

Scope of this document

This document applies to all types of AI / machine learning (AI/ML) algorithms intended to be used in GxP-regulated environments and need to be validated. It covers the process of training, deployment, and retraining under the constraint that the model is “frozen” before it is deployed. Continuous learning is outside the scope of this document.

Introduction

The technology is ready. There are use cases, which show the power of AI/machine learning (AI/ML). The next step is industrialization. For a regulated industry, this is more than a technical challenge. This document outlines a strategy of how AI/ML algorithms can be validated and put into action in a GxP environment. In principle, this is possible using existing methods and tools for CSV (computer system validation) and process validation. This green paper formulates a proposal for a framework in which this can be done in a systematic and reliable way.

It starts with outlining the differences in vocabulary between data science and quality. Next it explores how the difference between traditional and AI/ML applications translate to different risks. These different risks are the basis for the GxP-Readiness Level (GxP-RL) framework which describes the stepwise implementation and validation approach for AI/ML solutions.

Different vocabularies

The fields of data science and quality (CSV) use similar words for different things. To some extent, this is inevitable, but in this case, one needs to be prudent as the differences have the potential to overcomplicate the application of these novel technologies in a regulated environment.

The most striking difference is the usage of the word "validation." For a data scientist “validation” is the model performance evaluation after a single training cycle. It is used in the process of tuning the hyper-parameters of the model and has nothing to do with assessing the quality of the result.

For quality people, validation is the process of establishing documentary evidence, which demonstrates that a system does with a high degree of probability what it is designed to do in a consistent and reproducible manner.

We recommend the following approach to avoid unnecessary misunderstandings.

- To facilitate the validation of an AI/ML solution all reports should use GAMP5 terminology. Internally, the machine learning community can continue to use their own terms (at the risk of creating confusion.)
- Train all team members of an AI/ML project at the beginning and introduce them to the basics of validation, data science, and other team members to create a common understanding of the terms used within the project.

Differences between traditional and AI/ML applications

What are the key differences between classical rule based and AI/ML applications from a validation perspective? Contrary to what many people think, it is not so much the probabilistic nature of these models versus the deterministic nature of classical software coding. From a validation perspective, the final applications are very similar. Both are lines of code that are executed. The associated risks are covered by established CSV practices. The difference is in the role data plays. In deterministic applications data is processed, in AI/ML applications data used for training is part of the code.

Traditional software applications

The development of a traditional software can be summarized in following three steps:

1. User requirements: a user specifies the requirements / functionalities that the software needs to have.
2. Programming: Based on user requirements/specification a computer scientist develops a computer script that fulfills all requirements and specifications. After the development (but often also during) the code is versioned and tested.
3. Deployment and usage: after the user acceptance tests are passed successfully (the application is ready for its intended use) and documented this version of the code is deployed to the production system and executed by users.

As computer code is nothing else than a set of rules, the source of every error is traceable and has its source in the code or logic. These errors are also reproducible as they occur every time the program is executed (no accidental errors). Only deterministic errors occur.

This type of error can be detected by executing well-designed tests. Once the computer code passes these tests, one can be sure that it will consistently generate correct results. To do this systematically the GAMP5 framework provides the necessary guidance for CSV-activities and necessary documentation.

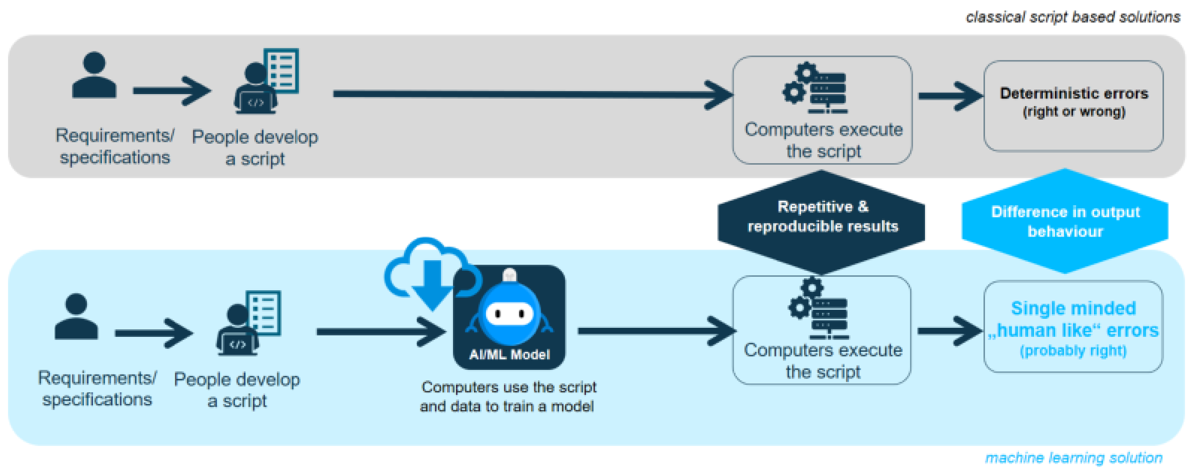

classical script based solutions
Requirements/ specifications
People develop a script
Computers execute the script
Deterministic errors (right or wrong)
Repetitive & reproducible results
Difference in output behaviour
Requirements/ specifications
People develop a script
AI/ML Model
Computers use the script and data to train a model
Computers execute the script
Single minded „human like" errors (probably right)
machine learning solution

Figure 1 What is different from traditional software development?

AI/ML applications

AI/ML applications are to a great extent similar to traditional software applications. The first and the last step of the software development process are identical to traditional software (see Figure 1). A trained AI/ML application is nothing else than a static script. This means that in general the same validation approaches apply to these types of solutions as well.

The difference lays in how the application is generated and that difference creates a new type of risk. The computer program behind an AI/ML application has three components.

1. The first component describes the model that needs to be trained.
2. The second component defines which data will be used to train the model.
3. Whereas the last component provides a script which defines how the model parameters will be updated when processing the data.

The output of this program is a model with “frozen” parameters. From a software perspective, this is a repeatable script, not different from a traditional program.

The main difference is the fact that (training) data are part of the code of AI/ML applications. Which means that the quality of an AI/ML application depends on the quality of the data. Even when the code is correct, an AI/ML application can return the wrong result when it processes data that are not reflected in the training data. This risk cannot be eliminated. It would take an infinite amount of data to prevent it. A risk management strategy for AI/ML applications cannot rely on CSV alone.

This means that there are three types of risks to the quality of the AI / ML application:

1. The quality of the data determines the quality of the application. Therefore, it is important to validate the systems and processes for handling data early on. This creates transparency of your ground truth.
2. Risk that the model does not function properly. Since deployed AI/ML applications are – just like any other computer program – lines of code that are executed. There is the risk of bugs in the code, issues with the infrastructure, etc. This risk can be mitigated by designing proper tests. Existing computer system validation approaches do this systematically.
3. Then there is the risk that the data used for training does not sufficiently cover real life situations. There is the risk that a properly trained model (no issues with the ground truth), executed correctly (all tests passed) returns the wrong results. This can have multiple reasons: perhaps for a specific case there are not enough data in the training data set, a new situation occured which was not considered when compiling the ground truth, etc. This risk is comparable to the risk of an error made by a trained operator. To mitigate this, risk mechanisms such as monitoring the solution in real life conditions need to be in place. This monitoring triggers established processes such as deviation and change management.

Validation framework for AI/ML applications

The proposed framework for the validation of AI/ML applications tackles the maturity of these solutions in five consecutive steps. Analogous to the technology readiness levels introduced by NASA we call them the GxP-readiness levels (GxP-RL) and allow you to position the GxP maturity of the application on this scale. Instead of focusing on the technology, the validation status is at its center. In the subsequent text, we give a short description of each level, more concrete deliverables can be found in a separate table.

Readiness level 1: Ground truth – data system validation

As data is a critical part of every model that will be generated, building a data set that fits to the intended use of the AI/ML application is the first step

This covers following activities:

1. Define user requirements and the intended use.
2. Analyze the possible GMP-risks of the use case and the intended solution.
3. Build a data set for training and testing (ground truth). This data set should represent real world conditions as close as possible and of high quality (as the quality of the data will determine the quality of the model).
4. Define, implement, and validate the systems and processes (e.g. data collection process, annotation process, handling of raw data etc.) that manage the data. Verify quality of the data and show, how the chosen data set fit to the intended use.

Readiness level 2: Build model – application verification

This level is reached when there is a validated application that fulfills the acceptance criteria when applied to the test data. Following activities need to be completed:

1. Translate the user requirements into acceptance criteria for a model (precision, recall, …).
2. Develop a model that fulfills the specifications.
3. Define, implement and validate (CSV) the application based on the risk analysis (model + infrastructure + UX).

Readiness level 3: Apprentice – model evaluation in real life situation

This level is reached when the predictive power of the model also applies in real life conditions. For that a monitoring system and process is designed and implemented. To get to this level two essential tasks need to be completed:

1. Define (incl. acceptance criteria), execute, and evaluate an experiment to provide evidence that the model works well in real life conditions.
2. Deploy the system to run in the production environment.
3. Design and implement a monitoring system and processes to track the model while running in the background.

Readiness level 4: Advisor – evaluation that the monitoring system captures all

After there is evidence that the model works well in real life conditions, it can be put into action parallel to a person. During this phase, evidence is gathered that the process runs stable enough and no situations occur that are not reflected in the training / test data set.

In case differences occur, they create deviations, which need to be processed following established CAPA processes with review of the original risk analysis (close the risk management cycle).

Readiness level 5: Decider – AI/ML solution takes over the role of an operator

At this level there is evidence that both the AI/ML application infers according to the set acceptance criteria and that the process is stable enough so that the AI/ML solution can make GxP decisions independently.

The following table summarizes the 5 steps:

GxP-RL	Key tasks
1 Ground truth	• Define user requirements and intended use. • Set of annotated training and test data.
2 Prototype	• Well defined model acceptance criteria. • Trained model. Ready application. • Evidence that the model fulfils the acceptance criteria for the available test data.
3 Apprentice	• Monitoring system • Deploy the system to run in the production environment. • Evidence that the model performs in real life conditions.
UNDER GxP CONTOLS	
4 Advisor	• Evidence that the process in which the AI/ML solution is applied, is stable.
5 Decider	• AI/ML solution is trusted and takes decisions

Development, learning and deployment of AI/ML applications

As with the NASA TRL scale, some applications will enter at different levels of the GxP-RL scale. In case there is an already existing validated system that manages your data, the entry point is GxP-RL1. It is also possible that you have some existing models, established test procedures. In that case you can enter the scale at GxP-RL2.

When everything needs to be built one can choose alternative ways to develop the solution. The first alternative is to move sequentially from GxP-RL1 to 3. Another is to develop the solution as a complete package and aim to enter the GxP-RL3 when the validation of the developed application is finalized. This allows a more agile application development

For AI/ML-application there are a couple of recommendations for an efficient development and deployment

- For the development and deployment of an AI/ML-application work in 4 systems
  - Model training systems:
    - The development system – used to develop and train the solutions.
    - The master system (copy of the development system) - used for a last clean training of the model, before deploying the final code to the QA system. This ensures that the quality of the training.
  - Model deployment systems
    - The Q system (copy of the production system) to test and validate the "frozen" model which got pushed from the master system
    - The Production system where the model infers using real life data.
- Use a data repository with audit trail capabilities from day 1
- Automate testing of requirements and the documentation of tests as much as possible. This will shorten the time needed to finalize the validation of the application in the Q-system.
- Building AI/ML solutions requires the collaboration of many functions as of day one. At the beginning following parties should be present: user, business data specialist, quality engineer, data scientist, and system architect. Make sure the team has the time to work on the development.

Note on explainable AI (XAI)

Explainable AI helps to understand why a model made a specific prediction based on the data it is trained with, but it adds limited value to manage the risks of AI/ML-applications. The reason is that it tells why it inferred, it does not tell you how it will infer incorrectly when given data that are not covered in the training / test data. This additional risk for AI/ML applications compared to rule-based applications, cannot be mitigated by XAI.

Note on continuous learning

We excluded continuous learning where the model is automatically retrained with additional real life data the moment a predefined threshold is reached for the following two reasons.

1. Difference in raw data: the training and testing data sets are curated. As the quality of the data determines the quality of the application, all these data points went through a quality control process. For training purposes data becomes raw data when it fulfills the quality criteria.
   In production this is different, all generated data is raw data. This means that also data that does not fulfill the quality criteria for training and test data are contained in this data set. Including all production data in the training set would lead to models with poor performance.
2. The second reason only applies to supervised learning where the data in the training and test set are annotated. In case the model encounters new situations, they need to be labeled accordingly. A process that cannot be automated.

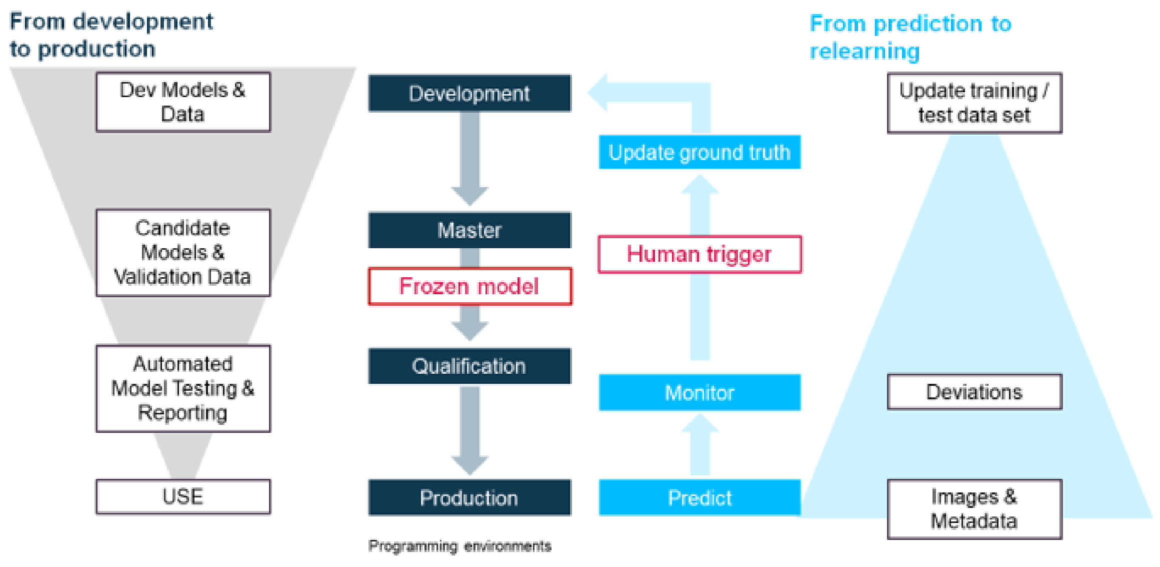


Figure 2 Environment architecture and how to treat model refinement

Authors & contributors

This document resulted out of a project where Daphne Tsatsoulis, Jim Bailey, Julia Ertl, Julia Fix, Oliver Winkelmann, and Tom Maes contributed most. Further we got input and comments from many colleagues during either the project or the generation of this discussion paper. We would like to thank all for their valuable contributions. That list is incredibly long, and we hope it keeps growing as we are looking forward to your comments and input as well.

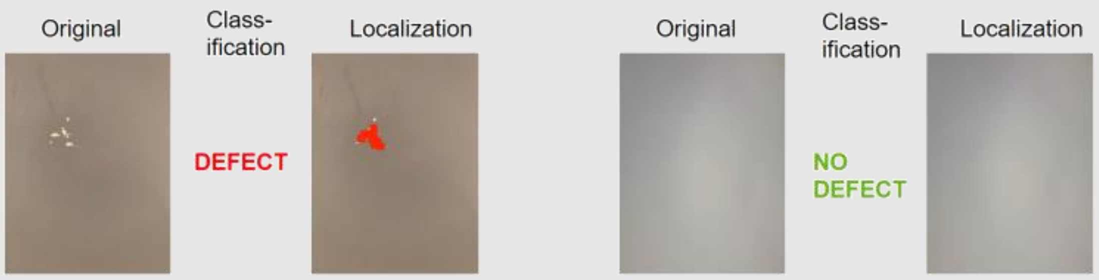
Contact: tom.maes@bayer.com

What do we mean with "single minded human like errors"?

We trained an algorithm to detect white powder stains on a polished metal surface. The algorithm should give us the answer to two questions:

1. Is it free from white stains?
2. If not, where are the stains located?

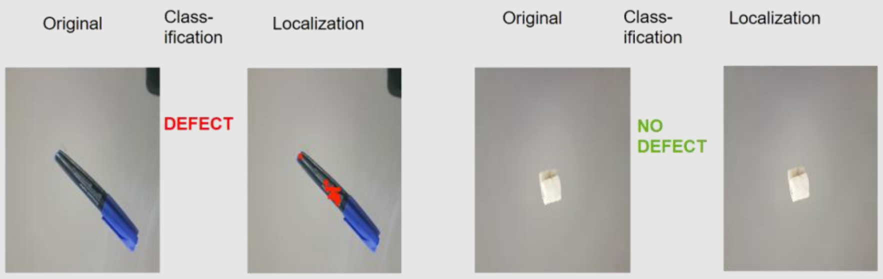
For that, we collected and annotated 80 images to train a deep neural network and 35 "test images" the AI/ML solution never had seen.



It correctly predicted if the surface was clean or not:

		Predicted	
		Defect	No Defect
Actual	Defect	30	0
	No Defect	0	5

Then we gave it two other "wrong scenarios" which any human would spot immediately, a pen and white candy:



The system classified the pen as “defect”, because it has small white letters (similar to the powder stains). However, it failed to recognize the white candy as it has a completely other shape as the powder stains we generated to train the algorithm.

The algorithm did what it was trained for. In a "single minded" way it classified every image in the way it made sense. This is what we mean with “single minded human like errors”.