Microsoft logo white text version

Zzz

Zzz

Une image contenant bleu, brouillard, capture d’écran, bleu vert

Description générée automatiquement

Responsible AI Workshop

Framing a (more) Trustworthy AI Lifecycle for your non-Generative AI-powered solutions

An illustration guide for data engineers, data scientists, AI developers, and other AI practitioners to help building trust around a Machine Learning based project.

Version 1.1 - August 2022 (Updated: October 2024)

This page is intentionally left blank.

Table of contents

Notice 4

About this guide and the learning objectives 5

Objectives of this guide 6

Non-objectives of this guide 6

Guide elements 6

Guide prerequisites 7

Module 1: Coping with the cybersecurity challenges posed by AI systems 8

Recognizing the significance of cybersecurity for AI systems 8

Conducting an adversarial attack as an illustration 8

Establishing your “North Star” 13

Securing AI systems by following a structured lifecycle approach 15

Module 2: A today vulnerable AI lifecycle for the ML models 24

An analysis of the so-called “classic” AI lifecycle 24

Understanding implied specific AI threats 30

Modeling AI threats with MITRE ATLAS™ 33

Defining a suitable taxonomy for AI threats 34

Module 3: Strengthening the “classic” (non-Generative) AI lifecycle 38

“Shift left security” in the “classic” AI lifecycle 38

Discussing complementary activities and practices 38

Considering tools, frameworks, and techniques of interest 45

Towards a (more) Trustworthy AI lifecycle 56

As a conclusion 60

To go beyond 61

Notice

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user.

A drawing of a face

Description automatically generated

Attribution 4.0 International (CC BY 4.0)

Microsoft and any contributors grant you a license to this document under the [Creative Commons Attribution 4.0 International Public License](https://creativecommons.org/licenses/by/4.0/legalcode), see the [LICENSE](https://github.com/microsoft/responsible-ai-workshop/blob/main/LICENSE) file, and grant you a license to any code in the repository under the [MIT License](https://opensource.org/licenses/MIT), see the [LICENSE-CODE](https://github.com/microsoft/responsible-ai-workshop/blob/main/LICENSE-CODE) file.

Microsoft, Windows, Microsoft Azure and/or other Microsoft products and services referenced in the document may be either trademarks or registered trademarks of Microsoft in the United States and/or other countries. The license for this document does not grant you rights to use any Microsoft names, logos, or trademarks. Microsoft's general trademark guidelines can be found at <http://go.microsoft.com/fwlink/?LinkID=254653>.

Privacy information can be found at <https://privacy.microsoft.com/en-us/>

Microsoft and any contributors reserve all other rights, whether under their respective copyrights, patents, or trademarks, whether by implication, estoppel or otherwise.

# About this guide and the learning objectives

Welcome to this guide Framing a (more) Trustworthy AI Lifecycle for your non-Generative AI-powered solutions for data engineers, data scientists, AI developers, and other AI practitioners.

As its name indicates, this guide is part of the Responsible AI Workshop and the related tutorials & walkthroughs.

Machine Learning (ML) is a subset of artificial intelligence (AI) that deals with the design of algorithms that have the ability to learn from data and improve on their own. They are not explicitly programmed to perform a specific task. Instead, they are given a set of training data that they can use to learn.

The main types of ML are as follows:

* Supervised learning is where the data is labeled, and the algorithm is given a set of training data to learn from. For example, if you wanted to build a ML model to predict the price of a house, you would give it a set of data that includes the price of the house and a set of features about the house (e.g., square footage, number of bedrooms, etc.). The algorithm would then learn to predict the price of a house based on its features.
* Unsupervised learning is where the data is not labeled, and the algorithm is given a set of data to learn from. For example, if you wanted to build a ML model to cluster data points, you would give it a set of data points and the algorithm would learn to cluster them based on their similarity.
* Reinforcement learning is where the algorithm is given a set of data to learn from and is also given feedback on its performance. For example, if you wanted to build a ML model to play a game, you would give it a set of data about the game, and it would learn to play the game by trial and error. As it plays the game, it would receive feedback on its performance and learn to improve its strategy.

Like all new technologies, the evolution of Machine Learning and more generally speaking artificial intelligence (AI) is happening faster than the development of controls and measures to secure their (sensitive) use cases. This is perfectly understandable in so far as one must first get used to this technology to fully understand its ins and outs.

This said, today, one should state that the field is sufficiently advanced for us to think about the risks weighing on systems powered by the use of ML model(s). Throughout this guide, we will refer to such systems as AI systems.

This field is definitely changing the world as we know it. There are applications in very different fields. However, there are still many industry practitioners and other actors in our world who are reluctant to use this technology for the simple reason that they aren’t confident in their ability to ensure the proper functioning and security of these AI systems. And this, even if it could drastically accelerate their development and the outcome of the considered systems.

Despite the compelling reasons to secure AI systems, a Microsoft’s survey entitled “[Adversarial Machine Learning - Industry Perspectives](https://arxiv.org/pdf/2002.05646.pdf)” spanning twenty-eight businesses found that most industry practitioners have yet to come to terms with adversarial Machine Learning. Twenty-five out of the 28 businesses indicated that they don’t have the right tools in place to secure their AI systems. What’s more, they are explicitly looking for guidance to do so.

The survey found that the lack of preparation is not just limited to smaller organizations – they range from Fortune 500 companies, governments to non-profit organizations.

**Organizations of any size, and regardless of the industry vertical, acknowledge the need to secure AI systems but simply do not know how.**

## Objectives of this guide

This guide aims at being a first element of answers towards the questions of confidence that one can have towards a non-Generative AI system, i.e., a system that leverage one or multiple ML models, with considerations that pertain to the related ML based projects and their development lifecycle.

By the end of the guide, you will be able to:

* Explain why it is necessary to evolve cybersecurity practices in the new context, in which Machine Learning is developing, with a concrete example.
* Recognize which parts of the development pipeline of a ML based project are vulnerable and in what way(s) and at what stage(s).
* Introduce (and discuss) a set of good practices in terms of development to ensure a minimum of confidence in the final project.
* Put forward a set of useful tools that can support new cybersecurity practices for ML-based project.

## Non-objectives of this guide

This guide is neitheraimed at introducing the building blocks of Responsible AI nor at giving a complete overview of RAI tooling aimed at helping to protect and control non-Generative AI systems.

For an introduction to RAI, and notably through Microsoft’s ongoing journey in the field, please refer to the guide [Establishing your own Responsible AI journey for your (non-Generative vs. Generative) AI-powered solutions](https://github.com/microsoft/responsible-ai-workshop/blob/main/responsible-ai-journey/docs/establishing-your-own-responsible-ai-journey.docx).

For tutorials of the most prominent tools we open-sourced for non-Generative AI, please refer to the guide [Leveraging Responsible AI Tooling for your non-Generative AI-powered solutions](https://github.com/microsoft/responsible-ai-workshop/blob/main/nongen-ai-tooling-tutorials/docs/leveraging-responsible-ai-tooling.docx).

These two guides are also part of this Responsible AI Workshop, which is available on GitHub at <https://github.com/microsoft/responsible-ai-workshop>.

**Note** For a complete overview of Microsoft’s resources designed to help you responsibly implement (non-Generative vs. Generative) AI systems, please refer to the [Microsoft Responsible AI resources page](https://aka.ms/rairesources).

## Guide elements

The document is organized as follows.

Module 1 will first illustrate why it is absolutely necessary to consider the cybersecurity aspect in the development of ML-based projects.

To do so, we will implement an adversarial attack on a ML model designed for the occasion and supposed to represent a critical use case of a company.

Then, Module 2 will analyze the classical development lifecycle of AI in order to:

* Understand how it is built.
* Determine where potential vulnerabilities may lie.
* Establish a "North Star" to (better) guide us in securing this cycle.

Finally, Module 3 will consider how to strengthen such a development lifecycle in order to harden it against the possible threats we will highlight, and ideally against those we do not yet know, and raise the bar. For that, we will put forward a certain number of good practices and tools which will allow you to a (more) trustworthy AI Lifecycle.

## Guide prerequisites

The Responsible AI Workshop repo on GitHubcontains a hands-on tutorial for this guide: *Adversial-attack.ipynb*.

If you would like to use it, we recommend that you read the following notes:

* [Cloning this workshop GitHub repo](https://github.com/microsoft/responsible-ai-workshop/blob/main/perequisites/cloning-the-repo.md).
* [Fulfilling the prerequisites for the workshop](https://github.com/microsoft/responsible-ai-workshop/blob/main/perequisites/fulfilling-prerequisites.md).
* [Getting started with Azure for your environment](https://github.com/microsoft/responsible-ai-workshop/blob/main/perequisites/getting-started-with-azure.md).
* [Using the Jupyter notebook in Azure Data Studio](https://github.com/microsoft/responsible-ai-workshop/blob/main/perequisites/using-jupyter-notebooks-in-azure-data-studio.md).
* [Setting up an Azure DevOps environment](https://github.com/microsoft/responsible-ai-workshop/blob/main/perequisites/setting-up-an-azure-devops-env.md).
* [Leveraging Counterfit](https://github.com/microsoft/responsible-ai-workshop/blob/main/perequisites/leveraging-counterfit.md).

**With that being said, let’s start by first considering the (many) cybersecurity challenges posed by AI systems.**

# Module 1: Coping with the cybersecurity challenges posed by AI systems

## Recognizing the significance of cybersecurity for AI systems

Let's take the example of autonomous cars which is a domain where ML is often associated. Computer vision models are indeed very well adapted to these problems. They can for example be assigned to traffic sign recognition, obstacle detection on the road and even to the driving part itself.

However, we do not yet have autonomous cars in the streets because we can’t yet measure the confidence we can have in this technology. A ML model, even if it gives very good performances on a never seen data set, is still subject to a lot of attacks that we must absolutely consider if we want to give more responsibilities to the ML model and somehow rely on its outcomes.

The list of attacks that can be articulated against a ML model is already large and is an ever-expanding field. However, to start illustrating the above, we will consider here what we call adversarial attacks.

An adversarial attack is an attempt to fool a ML model by providing it with input that is purposely designed to create an incorrect output. These attacks can be used to cause a model to misclassify data, which can lead to security vulnerabilities or other problems.

Adversarial attacks can be generated using a variety of methods, including manipulating the training data that a model is using, or creating new data that is designed to fool the model.

Here are few examples of type of adversarial attacks :

* Confidence reduction when we make the model less confident in its predictions.
* Misclassification when we make the model no longer able to correctly classify an input.
* Targeted Misclassification when we force to model to misclassify an input with a different, but selected, target.

**Note** The attack we will show you here is a shortened adaptation of [this notebook](https://github.com/microsoft/responsible-ai-workshop/blob/main/nongen-ai-tooling-tutorials/hands-on-tutorials/adverserial_attacks_counterfit/adverserial_attacks_counterfit.ipynb) that we have adapted to our use case.

## Conducting an adversarial attack as an illustration

For the sake of this illustration, we will show you the result of an adversarial attack on a computer vision model trained on traffic sign recognition. We will show you how we were able to compromise the model, although it is considered as perfect according to the usual metrics, by making it confuse a STOP sign with a 60 km/h speed limit sign. You call do it yourself with the hands on tutorial associated: *Adversial-attack.ipynb*

**Note** At the time of reviewing (23.10.2024), the attack on traffic signs is no more available on Counterfit. However, you can try with another type of images listed [here](https://github.com/Azure/counterfit/wiki/Basic-Use). You will just have to retrain a model with these data, no change is needed, and then generate your own attack with Counterfit. The notebook still work for the training and the test of the attack.

### Performing the attack

#### Defining the attack’s objectives

The first step is to determine the attack’s objectives.

For the sake of this illustration, and as introduced above, we want to compromise a computer vision model trained to classify traffic signs. More precisely, we want to create an adversarial image from an image correctly classified as a STOP sign and transform it into a speed limit sign in the eyes of the model while leaving it with the overall appearance of a STOP sign.

#### Training the model - optional, just for the sake of the illustration -

Then, we need to define and train a computer vision model on a dataset composed of traffic signs.

The model is built in a relatively simple way. It is a succession of convolutional layers separated by Batch Normalization layers. The convolutional neural network (CNN) layers are connected to dense layers, the last of which contains as many neurons as existing classes. We use the rectified linear activation function or ReLU as short as activation function except for the last layer where we use SoftMax to recover probabilities in output.

The training is done with [Adam as optimizer](https://arxiv.org/abs/1412.6980). We leave the default settings of TensorFlow. For the loss, we work with the Categorical Cross entropy and the metric we want to improve is the Accuracy.

model = keras.models.Sequential([

keras.layers.Conv2D(filters=16, kernel\_size=(3,3), activation='relu', input\_shape=(IMG\_HEIGHT,IMG\_WIDTH,channels)),

keras.layers.Conv2D(filters=32, kernel\_size=(3,3), activation='relu'),

keras.layers.MaxPool2D(pool\_size=(2, 2)),

keras.layers.BatchNormalization(axis=-1),

keras.layers.Conv2D(filters=64, kernel\_size=(3,3), activation='relu'),

keras.layers.Conv2D(filters=128, kernel\_size=(3,3), activation='relu'),

keras.layers.MaxPool2D(pool\_size=(2, 2)),

keras.layers.BatchNormalization(axis=-1),

keras.layers.Flatten(),

keras.layers.Dense(512, activation='relu'),

keras.layers.BatchNormalization(),

keras.layers.Dropout(rate=0.5),

keras.layers.Dense(43, activation='softmax')

])

opt = Adam()

model.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])

The dataset used is the German Traffic Sign Recognition Benchmark available [here](https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign). We split the dataset in a training dataset and a test dataset with a ratio of 80% / 20%.

At the end of the training, we have our final model which is really good according to our metric.

Epoch 30/30

858/858 [==============================] - 12s 14ms/step - loss: 9.8685e-04 - accuracy: 0.9998 - val\_loss: 0.0109 - val\_accuracy: 0.9976

The model has nearly 100% accuracy in predicting the output of never seen data.

But now, let’s prepare the real attack.

Analyzing the entry points

To compromise the model, we need to put ourselves in the attacker's shoes and understand what entry points we have.

*How much access do we have access to the model we want to attack? Do we have access to its weights?*

The answer to these questions will allow us to know what type of attack we will be able to implement.

If we don't have access to the structure of the model nor to its parameters, we can carry out a black-box attack by observing the answer of the model to an input. After some iterations of a black-box attack algorithm, we are likely to converge to an input which will be misclassified by the model. On the other hand, if we have access to the model and its parameters, it will be possible to implement a white-box attack which is more specific.

In our case, we will perform a black-box attack and assume we don’t have access to the model but only the inference part. We can only feed the network with an input and see the prediction.

#### Selecting the black-box attack algorithm

Counterfit offers a plethora of different attacks of all kinds. Both in black-box and white-box attacks. In our case, we will launch on a HopSkipJump attack which will be perfect.

#### Selecting the target image

We are going to create an adversarial image based on this one:

A picture containing text, stop, sign

Description automatically generated

Figure : Our target image which is correctly predicted as a STOP sign by the model

The idea will be to modify the image slightly using the HopSkipJump algorithm so that it is misclassified by the model but still looks like a STOP sign to human eyes.

#### Running the attack

Counterfit allows you to perform the attack very simply, you just have to give it the right input parameters. Below are the few commands needed to perform the attack.

**# Select target**

counterfit> set\_target traffic\_signs

**# Select the attack we will perform**

traffic\_signs> set\_attack HopSkipJump

**# Setting up some parameters**

traffic\_signs>HopSkipJump> set\_params --sample\_index 0 --max\_eval 1250 --max\_iter 5 --norm inf

**# And that’s all**

$ run

And that’s it ! Counterfit created our adversarial image.

#### Analyzing the results

Now, we need to verify that the resulting adversarial image meets the following requirements:

1. The adversarial image has to look like a stop sign
2. The adversarial image has to be classified by the model as another sign

Here is the resulting adversarial image:

A picture containing text, stop, sign

Description automatically generated

Figure : Our final adversarial image generated with Counterfit which is wrongly classified as a “Speed limit (60 kmph)” sign.

Both requirements are validated. The adversarial image still looks like a STOP sign to human eyes when it is incorrectly classified as a “Speed limit (60 kmph)” sign.

Qr code

Description automatically generated

Figure : A better view of what happened

### What can we learn from this?

The above example helps explaining why it is necessary to both consider and address the security part of an AI system.

For this specific adversarial attack, we exploited a weakness of the AI system deployment to conduct an adversarial attack on the model. As such, the adversarial attack that we just set up works because we have in place two crucial elements:

1. An unlimited access to the inference part of the ML model, which is important because the more inferences we do, the more we can adjust our initial image and make it converge to our final adversarial image.
2. Enough data returned by the ML model. In our case, we were on a black-box attack, but we still had access to the probability distribution given by the SoftMax layer in the model output which was enough to perform the attack.

So, as a user, we were therefore in perfect conditions to perform the attack. To correct this problem, you can consider several options:

* **Limiting the number of calls made to the API**: you can set up a minimal delay to wait between two calls to the API. With this approach, you can drastically increase the minimal duration of the attack to reach an acceptable adversarial image. This however alters the responsiveness of the systems...
* **Reducing the amount of information returned to the user after the inference**: instead of returning the output of the SoftMax layer that contains the probabilities of belonging to each class, you could instead return only the class that has the highest probability.
* **Making our ML model more robust**: you can do data augmentation on your initial dataset by adding adversarial images.

Nevertheless, this attack shows how relatively easy it is to compromise a ML model. Therefore, if you want to (continue to) potentially benefit from both the advances and thus the performances of ML in your systems whatever they are, then you need to tackle with such security issues and improve the security posture of your systems throughout the development lifecycle in place.

The case of autonomous cars highlighted here is just one among many other examples. Here are some other critical decisions that a ML model can be asked to make:

* Deciding whether or not to grant a loan to a customer.
* Deciding whether or not to offer a job to a candidate.
* Deciding whether or not to provide insurance to a customer.
* And a lot more…

Therefore, many important responsibilities weigh on this technology. We are talking about human, financial, environmental, and even societal responsibilities.

However, because of the lack of trust, this technology is either put aside in favor of (possibly) less efficient solutions – This can constitute by itself a good option if AI isn’t absolutory a necessity for the considered use case -, or it is used while taking more or less measured risks if at least already identified... and without any sort of security controls.

## Establishing your “North Star”

First, it is necessary to determine whether or not you really need this technology. Since AI is very trendy these days, there is a tendency to use it a little bit inappropriately, i.e., to phrase it in another way, in a non-responsible manner.

The easiest way to know if you really need a ML model in your system is to (always) follow a simple rule of thumb:

Don't build a ML model when a simpler approach could be just as successful.

And even if the answer if positive, you should always prefer a model that is interpretable rather than a more accurate model that might be more difficult to interpret.

After validating the need to work/cope with ML, you then need to highlight a number of principles that you must respect to ensure the development of a trustworthy AI.

The best practices can vary a lot depending on the specific application domain and context of the considered AI system(s). However, among the [core AI principles](https://www.microsoft.com/en-us/ai/responsible-ai?activetab=pivot1%3aprimaryr6) established by Microsoft as the foundation for a responsible and trustworthy approach to AI, we can highlight those below that would be our “North Star” towards designing a (more) trustworthy AI lifecycle.

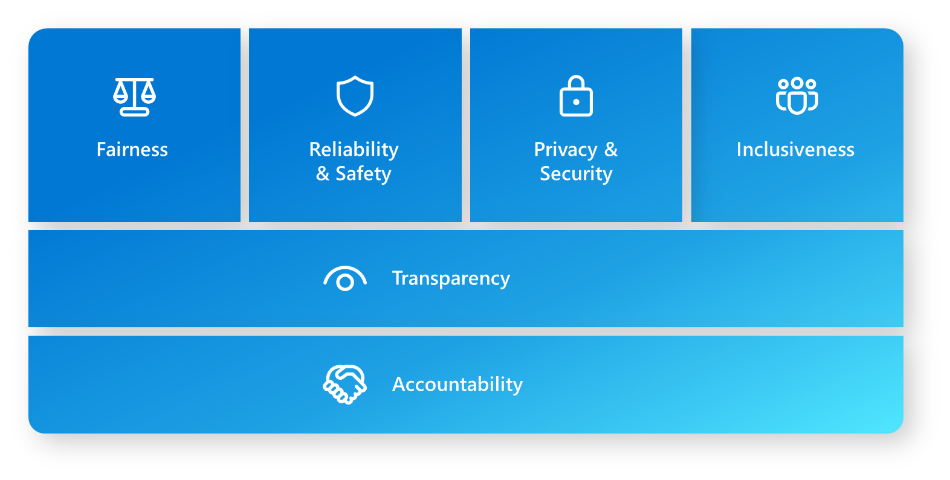


Figure . Microsoft Responsible AI Principles

You will have to define yours. See guide [Establishing your own Responsible AI journey for your (non-Generative vs. Generative) AI-powered solutions](https://github.com/microsoft/responsible-ai-workshop/blob/main/responsible-ai-journey/docs/establishing-your-own-responsible-ai-journey.docx), also part of this Responsible AI Workshop. You might also consider the [Asilomar AI Principles](https://futureoflife.org/ai-principles/). Such principles illustrate the complexity of delivering on AI in a fashion that consistently benefits humanity.

With that, let’s consider them.

#### Reliability and safety

AI/Machine Learning is increasingly used in support of high-value decision-making processes in medicine and other industries where the wrong decision may result in serious injury or death.

To drive trust, it is essential that AI systems operate safely, reliably, and consistently under normal circumstances and unexpected conditions.

These systems must be able to perform as intended when designed, respond safely to unexpected conditions, and withstand unsafe handling. It is also important to verify that:

1. These systems behave as expected under actual operating conditions.
2. These systems are able to identify abnormal behaviors and prevent manipulation or coercion outside of normal boundaries of acceptable behavior in relation to what these systems are and the specific task(s) they conduct/handle.

As previously introduced, there are new types of attacks specific to the AI/ML space. Such systems should be designed to resist inputs that would otherwise conflict with local laws, ethics and values held by the community and its creators.

Their behaviors, as well as the variety of conditions they can reliably and safely handle, reflects the range of situations and circumstances that developers anticipate during design and testing.

#### Privacy and security

**As AI becomes more widespread, protecting the privacy and security of important personal and business information is an increasingly complex and critical task.**

With AI, data privacy and security issues require special attention, as access to (sensitive) data is essential for such AI systems to make accurate and informed predictions and decisions about people, textual situations, etc., and these systems should be a responsible and trustworthy custodian of any information they have access to.

For personally identifiable information (PII), linked vs. linkable information, etc. AI systems must increasingly comply with privacy laws that apply. As an example, the EU’s General Data Protection Regulation (GDPR) that require transparency in data collection, use, and storage and require that consumers have appropriate controls over how their data is used.

These principles aim, among other things, to ensure that humans always have control over AI and highlight the need to define appropriate objectives from a cybersecurity perspective.

Now that our list of key principles is defined, and for the sake of the illustration, let's get to the heart of the subject and see how to try building a trustworthy AI system.

These principles might not be the same for you and your systems, or at least will not be given the same weight.

## Securing AI systems by following a structured lifecycle approach

Let’s start with the obvious: Every AI system is a software system at its core.

Despite seeming overly sophisticated, AI systems, just like any other computer system, remain vulnerable to security issues. Some of them don’t even require any prior knowledge of machine learning as these files store generic programs. Traditional software attack vectors and traditional hacking tools, some as simple as injecting malicious code into model repositories, exploiting poor encryption, improperly configured workspaces and broad access to the training data, or utilizing pickle serialization threats, can be abused and seen as risks to any AI system if not considered properly. This [proof of concept](https://github.com/Azure/counterfit/wiki/Abusing-ML-model-file-formats-to-create-malware-on-AI-systems:-A-proof-of-concept) shows how machine learning model file formats can be used to create malware.

Such system exposes in turn an API or a façade to send (interactive vs. batch) request to ML model(s) or invoke ML model(s) as part of their logic processing.

Consequently, **all the already existing, well-recognized and established secure software development lifecycles (SSDLC) models and frameworks apply at least for the classic software part of these AI systems**.

### Leveraging the Microsoft Security Development Lifecycle (SDL)

The [Microsoft Security Development Lifecycle (SDL)](https://www.microsoft.com/en-us/securityengineering/sdl/) is a software development process that helps architects and developers create more secure software.

As such, SDL includes a set of activities and tools that can be used at each stage of the software development process to help ensure the security of the software. SDL has been successful in helping to reduce the number of vulnerabilities in Microsoft products and has been adopted by many other organizations.

This development process will be the keystone of our development lifecycle. Thanks to SDL, we will be able to structure the development of the project for the software part in a coherent way, considering the security aspect that we are trying to put in place.

SDL consists of a set of practices that support security assurance and compliance requirements. Below, we'll look at and review what these practices are and how they can help us establish the basis towards a (more) trustworthy AI lifecycle.

#### Providing training

**Security is everyone’s job.** Architects, developers, DevOps engineers, etc. must understand security basics and know how to build security into software and services to make (AI) systems, more secure while still addressing business needs and delivering user value.

As you might expect and anticipate, such a consideration will also have an implication for data engineers, data scientists, ML developers, and other AI practitioners involved in the portion of the development lifecycle devoted the ML models and the data that comes along.

Effective security training will complement and re-enforce security policies, SDL practices, standards, and requirements of software security, and be guided by insights derived through data or newly available technical capabilities. See section Modeling AI threats with MITRE ATLAS™ below.

Suitable training materials could be developed to deliver on goals while mitigating the challenges discussed in this guide for a (more) trustworthy AI lifecycle. As such, AI-specific security training must ensure that the architects, developers engineers, etc. as well as AI practitioners are aware of the threats and subsequent risks posed to their AI systems and the resources at their disposal. These materials need to be delivered in conjunction with current training on protecting applications and systems in general, and the data they have access to and manipulate.

**Although security is everyone’s job, it’s important to remember that not everyone needs to be a security expert nor strive to become a proficient penetration tester.**

However, ensuring everyone understands the attacker’s perspective, their goals, and the art of the possible will help capture the attention of everyone and raise the collective knowledge bar. So, instead, the focus should be placed on educating on the “North Star” principles (see section Establishing your “North Star” above), as applied to their AI use cases.

In addition, beyond AI practitioners, architects, developers, etc. will need to understand the secure “building blocks” of AI systems that will be reused across their organization. There will need to be an emphasis on fault-tolerant design with subsystems which can be easily turned off (e.g., video or image processors).

#### Defining security (and reliability, safety, and privacy) requirements

The need to consider security is a fundamental aspect of developing highly secure (AI) systems in and regardless of development methodology being used, i.e., ~~“classic”~~ legacy waterfall methodology vs. modern agile methodology, security requirements must be continually updated to reflect changes in required functionality and changes to the threat landscape and the “state-of-the-art” in this space. Obviously, the optimal time to define the security requirements is during the initial design and planning stages as this allows development teams to integrate security in ways that minimize disruption.

Factors that influence security requirements include (but are not limited to) the legal and industry requirements, internal standards and coding practices, review of previous incidents, and known threats. These requirements should be tracked through either a work-tracking system or through telemetry derived from the engineering pipeline.

**AI/ML-specific pivots are required to existing security practices are required to mitigate the new types of security issues previously outlined.** Areas to covered includes in a non-exhaustive manner the authentication, the separation of duty, the input validation and the Denial of Service (DoS). Without attentions and investments in these areas, released AI systems will continue to fight an uphill battle against adversaries of all skill levels.

In addition, and beyond the “sole” security aspect, the requirements must address and cover all the principles an AI system must conform to as per the above-mentioned “North Star”, starting by the goals to pursue.

#### Defining metrics and compliance reporting

**It is essential to define the minimum acceptable levels of security quality, referred as to the SDL bug bar, and to hold engineering teams accountable to meeting that criterion.**

Defining them early helps a team understand risks associated with security issues, identify and fix security defects during development, and apply the standards throughout the entire project. Setting a meaningful bug bar involves clearly defining the severity thresholds of security vulnerabilities - for example, all known vulnerabilities discovered with a “critical” or “important” severity rating must be fixed with a specified time frame -, and never relaxing it once it's been set. See [SDL Security Bug Bar (Sample)](https://docs.microsoft.com/en-us/security/sdl/security-bug-bar-sample) for an illustration.

For AI systems, existing bug bar used to triage traditional security vulnerabilities must be extended to also serve as a reference for the triage of AI/ML-related security issues. See sections Understanding failures modes and **Understanding implied specific AI threats** below. See [AI/ML Pivots to the Security Development Lifecycle Bug Bar](https://docs.microsoft.com/en-us/security/engineering/bug-bar-aiml) for more information.

In order to track key performance indicators (KPIs) and ensure security tasks are completed, the bug tracking and/or work tracking mechanisms used by an organization (such as [Azure DevOps](https://azure.microsoft.com/en-us/services/devops/) or [GitHub Actions](https://github.com/features/actions)) should allow for security defects and security work items to be clearly labeled as security and marked with their appropriate security severity. This allows for accurate tracking and reporting of security work.

#### Performing threat modeling

[Threat modeling](https://strikecommunity.azurewebsites.net/articles/1941/course-threat-modeling-101.html) should be used in environments where there is meaningful security risk. Threat modeling can be applied at the component, application, or system level. **It is a practice that allows development teams to consider, document, and (importantly) discuss the security implications of designs in the context of their planned operational environment and in a structured fashion.**

Applying a structured approach to threat scenarios helps a team more effectively and less expensively identify security vulnerabilities, determine risks from those threats, and then make security feature selections and establish appropriate mitigations.

SDL provides a tool to perform this modeling: the [Microsoft Threat Modeling Tool](https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling). It makes threat modeling easier for all developers through a standard notation for visualizing system components, data flows, and security boundaries. It also helps threat modelers identify classes of threats they should consider based on the structure of their software design for the considered system. The tool has been designed so that non-security experts can also use it. It makes threat modeling easy for all developers by providing clear guidance on creating and analyzing threat models.

The perform a threat modeling, it is necessary to follow the following steps:

1. Defining security requirements.
2. Creating an application architecture diagram.
3. Identifying threats.
4. Mitigating threats.
5. Validating that threats have been mitigated.

To clarify how this crucial SDL practice works, we will carry out a simplified threat modeling on the system that was previously considered, and in which we compromised a computer vision model trained to classify traffic signs.

##### Step #1: Defining security requirements

We work with a computer vision model trained to classify images of road signs. This classification part being relatively critical, we want the system in general to be robust to external attacks. More specifically, we want the model to be resistant to adversarial attacks like the ones we have done in the PoC.

##### Step #2: Creating an application architecture diagram

In this step we have to make a diagram of the project structure. In the interest of this demonstration, we will simplify the overall architecture of the project.

Let's say that the architecture consists of two main parts:

1. An IoT part that represents the camera installed on a vehicle.
2. A web server on which the model has been deployed behind an API allowing inference.

A picture containing chart

Description automatically generated

*Figure 5: PoC diagram inside the threat modeling tool*

##### Step #3: Identifying threats

Now we can run the threat modeling using the same tool.

Here are the results:

A picture containing chart

Description automatically generated

*Figure 6: Running threat modeling on the PoC diagram*

The image above shows the result of threat modeling on a very simplified version of our system where we would deploy our model in the cloud as an API. This API would expose endpoints that would be callable by IoT devices like connected cameras.

We see that there are several important information points to consider ensuring a minimum of security in our system. If we want to avoid adversary attacks, we can focus our attention on these lines:

* An adversary can inject malicious inputs into an API and affect downstream processes. (1)
* If proper authentication is not in place, an adversary can spoof a source process or an external entity and gain unauthorized access to the web application. (2)

The threat modeling tool shows us that we can run into security problems if we don't control user input (1) and run the service without authentication (2).

##### Step #4 and #5: Mitigating threats and validating that threats have been mitigated.

An adversarial attack involves the misuse of the inference part of a ML model. The first step to prevent this is to detect these abusive uses. To do this, we can imagine several solutions, but the simplest is simply to observe when the number of requests to the model exceeds a certain threshold to be determined beforehand. If the number of requests exceeds the threshold, we can assume that the model is under attack and that we must act accordingly.

In this type of attack, it is important to quickly restrict or even cut the connection between the attacker and the model to preserve the system. However, we do not want to cut off access to the model to all users but only to the attacker. Hence the interest in implementing an authentication system to allow case-by-case decisions.

That said, it appears to be a good idea to restrict access to the model itself by authenticating each user and setting a limit on the number of requests to the model to limit the possibility of adversarial attacks.

**The above only constitutes both an illustration and an introduction of the subject.**

See [Threat Modeling AI/ML Systems and Dependencies](https://docs.microsoft.com/en-us/security/engineering/threat-modeling-aiml) for more information. Also watch the video [AI Security Engineering—Modeling/Detecting/Mitigating New Vulnerabilities](https://www.youtube.com/watch?v=SiACfPJblAs).

#### Establishing design requirements

SDL is typically thought of as assurance activities that help engineers implement “secure features”, in that the features are well engineered with respect to security.

To achieve this, engineers will typically rely on security features, such as cryptography, authentication, logging, and others. In many cases, the selection or implementation of security features has proven to be so complicated that design or implementation choices are likely to result in vulnerabilities. Therefore, it’s crucially important that these are applied consistently and with a consistent understanding of the protection they provide.

Such design requirements as expressed above should be extended in such a way to embrace all the principles of our “North Star” (see section Establishing your “North Star” above), and thus should also cover the reliability, the safety, and the privacy, with first the definition of the goals to pursue.

#### Defining and using cryptography standards

With the rise of mobile and cloud computing, it’s critically important to ensure all data, including security-sensitive information and management and control data, is protected from unintended disclosure or alteration when it’s being transmitted or stored. Encryption is typically used to achieve this.

Making an incorrect choice in the use of any aspect of cryptography can be catastrophic, and it’s best to develop clear encryption standards that provide specifics on every element of the encryption implementation. This should be left to experts. A good general rule is to only use industry-vetted encryption libraries and ensure they’re implemented in a way that allows them to be easily replaced if needed.

See [Microsoft SDL Cryptographic Recommendations](https://docs.microsoft.com/en-us/security/sdl/cryptographic-recommendations) for more information.

#### Managing the security risk of using third-party components

Today, the vast majority of software projects are built using third-party components (both commercial and open source). When selecting third-party components to use, it’s important to understand the impact that a security vulnerability in them could have to the security of the larger system into which they are integrated. Having an accurate inventory of third-party components and a plan to respond when new vulnerabilities are discovered will go a long way toward mitigating this risk, but additional validation should be considered, depending on your organization's risk appetite, the type of component used, and potential impact of a security vulnerability.

**Such considerations have to be extended to the third-party data for AI systems.** ML models are indeed largely unable to discern between malicious input and benign anomalous data. A significant source of training data is derived from un-curated, unmoderated public datasets that may be open to third-party contributions.

**Attackers don’t need to compromise datasets when they are free to contribute to them. Such dataset poisoning attacks can go unnoticed while model performance inexplicably degrades. Over time, low-confidence malicious data becomes high-confidence trusted data, provided that the data structure/formatting remains correct, and the quantity of malicious data points is sufficiently high.**

**We will come back to that later in this guide.**

#### Using approved tools

Define and publish a list of approved tools and their associated security checks, such as compiler/linker options and warnings. Engineers should strive to use the latest version of approved tools, such as compiler versions, and to take advantage of new security analysis functionality and protections.

On this subject, Microsoft SDL has published a [list of tools and resources](https://www.microsoft.com/en-us/securityengineering/sdl/resources) to address some of the issues that can be encountered in software development projects.

As you can anticipate, this calls for an equivalent for the ML platform(s), tooling, and libraries. The development environment/workspace must also be setup and configured in a securely manner while respecting the governance in place. A secure configuration that is compliant with your organization policies supposes to address considerations such as:

* Restricting access to AI/ML resources and operations by user accounts, groups, roles (role-based access-control), or attributes (attribute-based access control),
* Restricting incoming and outgoing network communications to the development environment/workspace,
* Encrypting data in transit and at rest,
* Scanning for vulnerabilities,
* Applying and auditing configuration policies,
* Etc.

For considerations that pertain to [Azure Machine Learning](https://azure.microsoft.com/en-us/services/machine-learning/), see [Secure Azure Machine Learning Service (AMLS) Environment](https://techcommunity.microsoft.com/t5/fasttrack-for-azure/secure-azure-machine-learning-service-amls-environment/ba-p/3162297). Specific details for the enterprise security and governance for Azure Machine Learning can be found in [Enterprise security and governance](https://docs.microsoft.com/en-us/azure/machine-learning/concept-enterprise-security).

Azure Machine Learning is for individuals and teams implementing MLOps within their organization to build and bring ML models into production in a secure and auditable production environment.

#### Performing Static Analysis Security Testing (SATS)

Analyzing the source code prior to compilation provides a highly scalable method of security code review and helps ensure that secure coding policies are being followed.

With Dev(Sec)Ops, SAST is typically integrated into the commit pipeline to identify vulnerabilities each time the software is built or packaged. However, some offerings integrate into the developer environment to spot certain flaws such as the existence of unsafe or other banned functions and replace those with safer alternatives as the developer is actively coding.

There is no one size fits all solution and development teams should decide the optimal frequency for performing SAST and maybe deploy multiple tactics - to balance productivity with adequate security coverage.

#### Performing Dynamic Analysis Security Testing (DAST)

Performing run-time verification of your fully compiled or packaged software checks functionality that is only apparent when all components are integrated and running. This is typically achieved using a tool or suite of prebuilt attacks or tools that specifically monitor application behavior for memory corruption, user privilege issues, and other critical security problems.

Similar to SAST, there is no one-size-fits-all solution and while some tools, such as web app scanning tools, can be more readily integrated into the continuous integration / continuous delivery pipeline, other DAST testing such as fuzzing requires a different approach.

If Machine Learning techniques are increasingly introduced as new method into fuzz testing to overcome some of the challenges in the fuzzing process (see [A systematic review of fuzzing based on machine learning techniques](https://pubmed.ncbi.nlm.nih.gov/32810156/) as an illustration), conversely, a so-called “Machine Learning Fuzzing Framework” could be created, which provides data engineer and data scientists with the ability to inject various types of attacks into test training sets for AI to evaluate. This could focus on different data types depending on the considered AI systems.

#### Performing penetration testing

Penetration testing is a security analysis of a software system performed by skilled security professionals simulating the actions of a hacker. The objective of a penetration test is to uncover potential vulnerabilities resulting from coding errors, system configuration faults, or other operational deployment weaknesses, and as such the test typically finds the broadest variety of vulnerabilities.

Penetration tests are often performed in conjunction with automated and manual code reviews to provide a greater level of analysis than would ordinarily be possible.

One should consider extending the framework in in place if any with AI/ML-focused activities.

#### Establishing a Standard Incident Response Process

Preparing an Incident Response Plan is crucial for helping to address new threats that can emerge over time. It should be created in coordination with your organization’s dedicated Product Security Incident Response Team (PSIRT) if any.

The plan should include who to contact in case of a security emergency, and establish the protocol for security servicing, including plans for code inherited from other groups within the organization and for third-party code. The incident response plan should be tested before it is needed!

Furthermore, AI/ML is increasingly used in support of high-value decision-making processes in medicine and other industries where the wrong decision may result in serious injury or death.

A lack of forensics reporting capabilities in an AI system will prevent these high-value conclusions from being defensible in both the court of law and court of public opinion. AI must have built-in forensic capabilities. This enables to provide transparency and accountability for the considered AI system, ensuring its actions are not only verifiably correct but also legally defensible. These capabilities also function as an early form of “AI intrusion detection”, allowing to determine the exact point in time that a decision was made by a classifier, what data influenced it, and whether or not that data was trustworthy.

One should note that data visualization capabilities in this area are rapidly advancing and show promise to help security engineers/experts identify and resolve root causes for these complex issues.

**At this stage, we reviewed how SDL practices applies to AI systems, outlined some AI/ML pivots when relevant. With that, let’s now consider more specifically the AI lifecycle of the ML models.**

### The need to also address the specific AI lifecycle of the ML models

Every AI system also comprises as its name suggests one or several ML models.

The development cycle of a ML model can be complex depending on the size of the project and the many disciplines that may be involved.

**With the complexity of the project also comes the** greater risk of introducing vulnerabilities throughout the development process.

**The reason why these AI risks exist is relatively simple but crucial to understand:**

* + **Nor the professionals involved in the development of the ML models, i.e., data scientists and the data engineers, are trained in cybersecurity.**
  + **Neither the ones in charge of the operationalization, the deployment, and the maintenance of the ML models, i.e., ML developers, MLOps engineers, other AI practitioners, Dev(Sec)Ops engineers, etc. are necessary aware of the suitable controls to implement, or any other relevant measures to be adopted for the sake of cybersecurity, even though some of them might potentially trained in cybersecurity in general.**

This highlights a clear need for appropriate and necessary AI/ML-oriented cybersecurity training for the various disciplines involved in the development lifecycle of a ML-based project as already outlined.

As already stressed a number of times, same considerations apply to other dimensions, such as the reliability, the safety, and the privacy. referring back to our “North Star”.

In order for such a training to be as effective as possible, it should start from the study of the classic development cycle that has already proven its effectiveness.

In this way, we will be able to:

* **Understand how the classic development lifecycle is built**: we first need to understand from what we are starting from, and what can be considered as “mainstream”.

*What are the main stages of development? Who are the different skills needed? Who works on each of these bricks?*

* **Analyze the threats to the development lifecycle**: once we have identified the different building blocks or bricks of the development lifecycle, we can then see how they might introduce vulnerabilities at some stage(s) in the system.
* **Strengthen the development lifecycle**: this consists in putting forward a certain number of activities and principles. We will also complete this by highlighting tools and technologies that may be useful in our context.

# Module 2: A today vulnerable AI lifecycle for the ML models

The first requirement to forward for establishing the basis for a (more) trustworthy AI lifecycle is starting from an AI development lifecycle that has already proven itself several times and is widely adopted.

Additional activities, tooling and practices must indeed complement/enrich a mainstream lifecycle. This is the bare condition to have them in turn be (widely) adopted.

## An analysis of the so-called “classic” AI lifecycle

Before considering how this lifecycle poses a problem from a security point of view in terms of threats’ exposure, let’s review this lifecycle along with the various artefacts/components that compose an AI lifecycle.

### Main stages

As one can imagine, an AI lifecycle like any other lifecycle is classically made of/structured as a set of main stages, nine here, through which we have to go. This constitutes as a whole what we can refer as to a workflow.

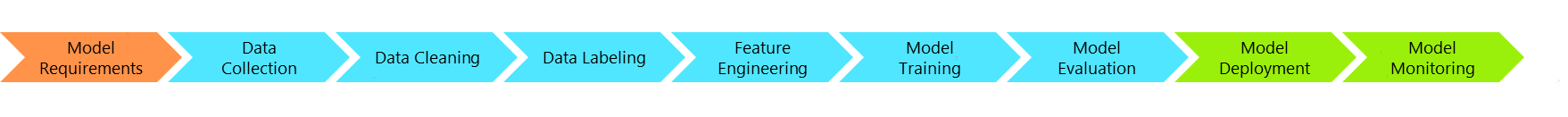


Figure : Main stages of AI lifecycle

Let’s shortly introduce these main stages in order.

Stage #1: Model requirements

The model requirements stage is the first stage in the AI lifecycle. In this stage, data scientists work with business stakeholders to understand the problem that they want to solve with ML. They also identify the data that will be used to train the ML model.

Stage #2: Data collection

The data collection stage is the process of gathering data that will be used to train the ML model. This data can come from a variety of sources, such as databases, text files, images, and so on.

Once the data is collected, it must be prepared for use in training the model. This preparation may include cleaning the data, formatting it into a specific structure, and so on.

Stage #3: Data cleaning/cleansing

The data cleaning stage is when you prepare your data for modeling. This stage can involve a variety of tasks, such as imputing missing values, standardizing data, and creating new features.

The goal of this stage is to get your data into a form that is ready for modeling. This stage can be time-consuming, but it is important to do a thorough job so that your models will be as accurate as possible.

Stage #4: Data labeling

The data labeling stage in ML is the process of assigning labels to data points. This is usually done by humans, although there are some automated methods. The labels can be anything, but they are typically classes or categories. The data labeling stage is important because it is the input to the ML algorithm(s). If the labels are incorrect, the results of the algorithm will be incorrect.

Stage #5: Feature engineering

Feature engineering is the stage where data is transformed, and new features are created. This stage is important because the features created will be used by the chosen ML algorithm(s) to learn the relationship between the input data and the target variable.

Feature engineering can be used to create features that are more informative, reduce the dimensionality of the data, or make the data more amenable to ML algorithms. For example, feature engineering can be used to transform raw data into features that better represent the underlying relationships in the data, or to create new features that are more predictive of the target variable.

It is a creative process, and there is no single right way to do it. The goal is to transform the data in a way that makes it more useful for the ML algorithm. The best way to do this will vary depending on the data and the ML algorithm being used.

Stage #6: Model training

The training stage of a ML model is the process of teaching the model to recognize patterns in data. This phase is important because it allows the model to learn the relationships between input and output, so it can generalize on data never seen before.

Stage #7: Model evaluation

The evaluation stage of a ML model is when the model is evaluated on a dataset that it has not seen before. This is important in order to ensure that the model has not overfit to the training data. If the model performs well on the test data, it is likely that it will generalize well to new data.

Stage #8: Model deployment

The deployment stage of a ML model is when the model is deployed in a production environment. This can be done in a number of ways, but typically it involves deploying the model on a server that can be accessed by users.

The model may be deployed as a web service, which can be accessed by a web browser, as a standalone application, or as part of a microservice application. In either case, the model will need to be configured to work with the data and infrastructure of the production environment.

Stage #9: Model monitoring

The model monitoring stage in Machine Learning is the stage where the model is constantly monitored and updated. This stage is important because it helps to ensure that the model is constantly improving and that any issues with the model are quickly identified and corrected.

This stage also helps to identify when the model is no longer useful and needs to be replaced.

### Pipelines

Some of the above main stages constitutes in turn pipelines, to stream in a reproducible manner the workflow and the related processes : data pipeline vs. modeling pipeline.

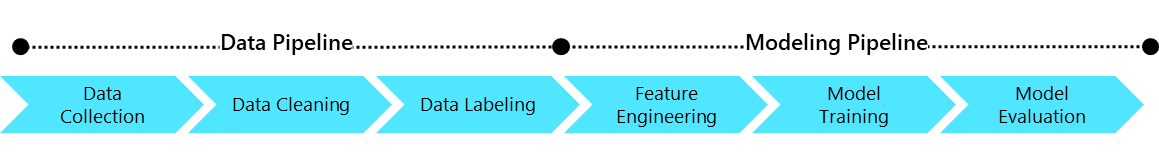


Figure : Pipelines which compose AI lifecycle

* Data pipeline: this data is a series of stages that data goes through in order to be transformed from raw data to useful features that can be used in a model. The data pipeline is often referred to as the data preprocessing stage.

This is an important part of the AI lifecycle because it can have a significant impact on the performance of a ML model. A well-designed data pipeline can help to improve the accuracy of a model, while a poorly designed data pipeline can lead to decreased accuracy.

It includes the following stages:

1. **Data collection**.
2. **Data cleaning/cleansing**.
3. **Data labelling**.

* Modeling pipeline: this pipeline in ML can be generally described as a process of iteratively improving a model. This typically involves starting with a simple model, then making it more complex as more data is collected and more experience is gained. The goal is to find the simplest model that accurately captures the underlying relationships in the data.

The modeling pipeline in Machine Learning is composed of the following stages:

1. **Feature engineering**.
2. **Model training**.
3. **Model evaluation**.

### Phases

The above main stages can be grouped in three phases:

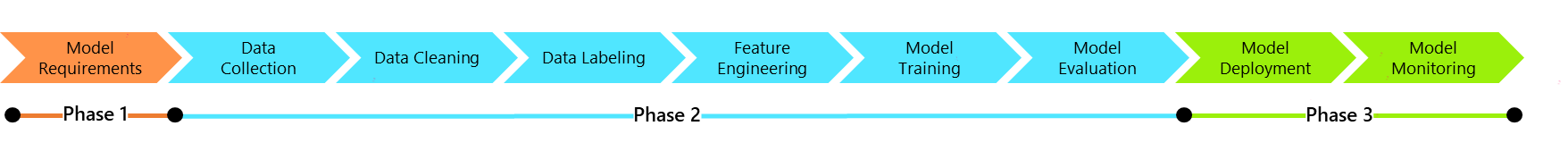


Figure : Phases which compose AI lifecycle

* Phase 1: this phase includes the very first “Model requirements” stage of the workflow presented above.
* Phase 2: this phase encompasses the data pipeline starting with data collection as well as the modeling pipeline up to model evaluation, which can feed back into each of the previous stages. This phase is an iterative one and constitutes a loop, see below.

This phase also implies the availability a local deployment part for testing the model before the real deployment, which we consider as part of model evaluation stage below - the last step of the modeling pipeline.

* Phase 3: this phase comprises the deployment and monitoring stages.

### Loops

From a MLOps/DevOps perspective, the iterative phase 2 further constitutes the inner loop of the AI lifecycle, while the phase 1 and phase 3 are part an outer loop.

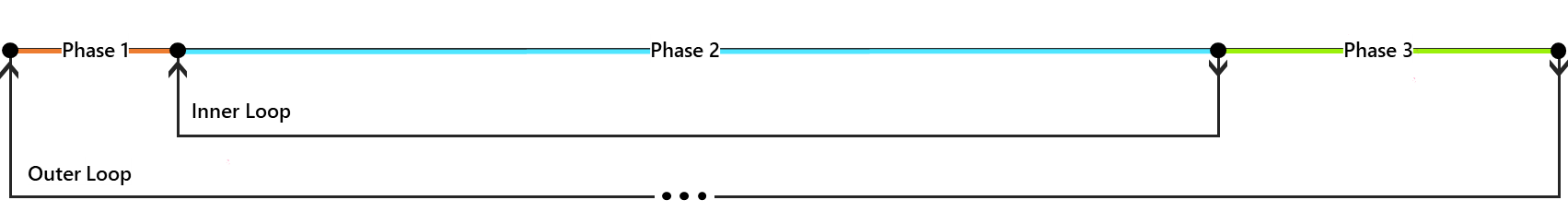


Figure : Loops which compose AI lifecycle

* Inner loop: this loop is the core inner development loop, which takes as input the model requirements and produces a ML model satisfying those requirements.

This is usually what induces the most attention around it as it includes implementing the model itself and tuning it.

* Outer loop: this consists of deploying the considered ML model in production and monitoring it to ensure that it meets all requirements.

This loops also includes all DevOps tasks which are not specific to ML-based projects such as Continuous Integration (CI) or Continuous Deployment/Delivery (CD).



Figure : Outer loop DevOps stages

### Persona

There are a wide variety of persona involved in the development lifecycle of a ML model. Here is a description of the main ones. They are all stakeholders of this endeavor.

Table

Description automatically generated

Figure : AI lifecycle persona

Data engineer

A data engineer is responsible for designing, building, and maintaining the data infrastructure that allows an organization to effectively collect, store, and process data for the intended AI systems, and their ML models.

A data engineer must have a strong understanding of data management principles and be able to work with a variety of data-related technologies.

Data analyst

A data analyst is a professional who is responsible for analyzing data and providing insights to help businesses make better decisions.

A data analyst may work with a variety of data, including financial data, customer data, and marketing data. A data analyst may use a variety of tools and techniques to analyze data, including statistical analysis, data mining, and data visualization.

Data scientist

A data scientist is responsible for extracting insights from data that can be used to improve decision-making. This may involve developing statistical models, applying ML algorithms, or performing exploratory data analysis.

A data scientist must have strong analytical and problem-solving skills, as well as experience with statistical software and programming languages.

MLOps engineer

A MLOps engineer is responsible for managing and maintaining the above ML workflows and pipelines. This includes ensuring that data is properly formatted, managing model training and deployment, and monitoring model performance.

The MLOps engineer also works with other teams to ensure that the ML pipelines are integrated with the rest of the fabric of the considered AI systems, and the related organization’s infrastructure.

Dev(Sec)Ops engineer

A DevOps engineer is someone who specializes in the development and operation of software part of the considered AI systems.

They are responsible for the deployment and maintenance of software applications, as well as the management of the associated infrastructure. DevOps engineers typically have a background in software development and are familiar with a variety of programming languages and frameworks. They are also skilled in system administration and automation tools.

Note that there is also the term "DevSecOps" for development, security and operations which is an approach that integrates security as a shared responsibility throughout the lifecycle, thus making security principles and practices an integral part of DevOps while maintaining improved efficiency and productivity

As new types of cybersecurity attacks rise, harden your development environment and data/software supply chain by integrating security early in the development cycle. See [Microsoft Security DevOps](https://www.microsoft.com/en-us/securityengineering/devsecops) for more information.

In Azure, [DevSecOps combines GitHub and Azure products and services](https://azure.microsoft.com/en-us/solutions/devsecops/#overview) to help DevOps and SecOps teams collaborate in building more secure (AI) systems.

How can MLOps and Dev(Sec)Ops can work (better) together ?

MLOps is a practice for operationalizing and managing ML workflows that pertains to the above depicted stages, pipelines, and phases. It indeed aims to streamline and automate the process of building, training, and deploying ML models.

As outlined before, Azure Machine Learning can help implementing these pipelines and putting MLOps into practices into your organization.

Dev(Sec)Ops is a set of practices that aims to automate the process of software delivery and infrastructure management. It aims to improve the collaboration between software developers and operations staff, and to reduce the time it takes to deliver new features and updates to users.

MLOps and Dev(Sec)Ops share many common goals and practices. Both aim to improve the collaboration between different teams, and to automate repetitive tasks. Both also place an emphasis on monitoring and logging to help identify and debug problems.

The two disciplines can complement each other well. ML Ops can help to automate the process of training and deploying ML models, while DevOps can help to automate the process of delivering new features and updates to users.

This brings us to the following complete development AI lifecycle:

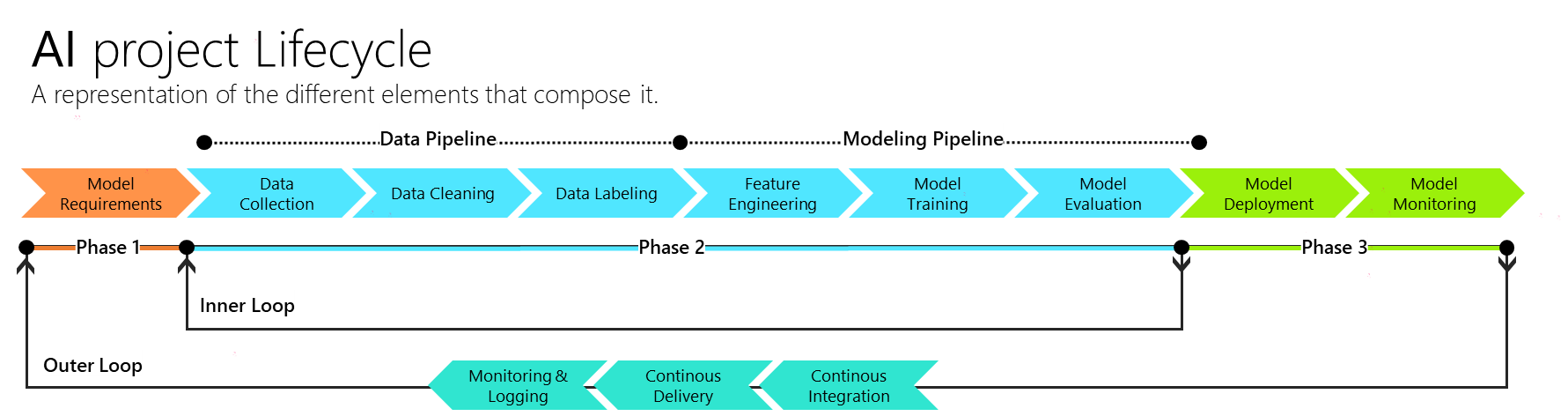


Figure : The complete end-to-end AI lifecycle

MLOps handle the stages Model requirements, Model deployment, and Model monitoring.

While Dev(Sec)Ops focus on the stages Continuous Integration (CI), Continuous Delivery/Deployment (CD), and Monitoring & Logging.

Note that MLOps and Dev(Sec)Ops can work side-by-side on the part Model Deployment stage.

Such considerations are extensively covered in the guide [Implementing a Responsible AI Lifecycle for MLOps processes](https://github.com/microsoft/responsible-ai-workshop/blob/main/nongen-ai-lifecycle-walkthrough/docs/implementing-responsible-ai-lifecycle.docx), also part of this Responsible AI Workshop.

### Understanding failures modes

**With the already introduced adversarial attacks on both the ML algorithms and data that keep increasing, and as these ML-powered features and/or AI systems become more pervasive, the need to understand how and why they fail, whether by the hand of an adversary or due to the inherent design of a system, will only become more pressing to leverage the suitable techniques as part of the design, the development, the deployment, along with the monitoring of these features and/or systems.**

Regarding the failure modes, they range:

* From *intentional failures* wherein the failure is caused by an active adversary attempting to subvert the system to attain her goals – either to misclassify the result, infer private training data, or to steal the underlying algorithm. See next section on AI threats.
* To *unintentional failures* wherein the failure is because an ML feature or AI system produces a formally correct but completely unsafe outcome.

See [Failure modes in Machine Learning](https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning) for more information. There have been hundreds of research papers dedicated to this topic, but inconsistent vocabulary from paper to paper has limited the usefulness of important research to data scientists, security engineers, and incident responders. This article includes vocabulary that can be used to describe intentional failure caused by an adversary attempting to alter results or steal a ML model as well as vocabulary for unintentional failures like an AI system that produces results that might be unsafe.

## Understanding implied specific AI threats

Due to the complexity of Machine Learning and the fact that it is a relatively new field, there is a wide variety of threats that can be found throughout the above “classic” AI lifecycle of a ML model.

### Categorizing AI threats

If some of them are documented, there are still many that will emerge in the coming years as we understand what is going on behind this technology.

Diagram

Description automatically generated

Figure . Representation of some of the most known threats against ML model

In the figure above, we categorize some types of attacks. Let’s consider them in order and share some explanations for each of them.

#### Data poisoning

Data poisoning is the introduction of malicious data into a training dataset for a Machine Learning algorithm. The purpose of data poisoning is to cause the algorithm to learn a false model from the data, which can be used to cause the algorithm to make inaccurate predictions. Data poisoning can be used to cause a Machine Learning algorithm to misclassify data, or to cause the algorithm to make predictions that are biased in favor of the attacker.

#### Adversarial attack

An adversarial attack is a malicious attempt to subvert the normal operation of a system. In the context of Machine Learning, an adversarial attack is an attempt to fool a ML model into making an incorrect prediction. Adversarial attacks can be used to exploit weaknesses in ML models to cause them to misclassify data. For example, an attacker could create a fake image that is designed to fool an image recognition model into thinking it is a picture of a cat when it is actually a picture of a dog.

#### Transfer learning attack

Transfer learning attack is a type of attack where the attacker uses a model trained on one dataset to attack a different but related dataset. This is possible because the knowledge learned by the model can be transferred to the new dataset, allowing the attacker to bypass training a model from scratch. This attack can be used to target any type of ML model, including deep neural networks.

#### Data extraction

A data extraction attack is the extraction of private data from a ML model, such as training data. This can be done by reverse engineering the model to understand how it works, or by using the model to make predictions and interpret them using algorithms to retrieve sensitive data.

#### Model extraction

A model extraction attack is a type of attack against a ML model where an attacker tries to reconstruct the model from its output. This can be done by reverse-engineering the model, or by using a technique called query synthesis. As such, query synthesis is a technique where the attacker generates a set of queries that are likely to produce the same output from the model, and then uses these queries to reconstruct the model.

See [AI/ML Pivots to the Security Development Lifecycle Bug Bar](https://docs.microsoft.com/en-us/security/engineering/bug-bar-aiml) for more information.

### Mitigating specific AI threats

In the above section, we highlighted several types of attacks against the development pipeline of a Machine Learning based project.

To fight against these types of threats, address the associated vulnerabilities, put in place the needed cybersecurity controls and thus adequately mitigate the associated risks, the general idea is not to fight against each one of them individually but rather to put into practice general concepts, activities, and practices that allow to get rid/mitigate of them in a grouped and systematic way to encompass a yet evolving complete AI threat landscape.

In a classical software development cycle, we could for example quote the rule "Never trust a user's input" which by the way still applies in our case.

*Do similar rules of thumb exit and apply to AI systems?*

And starting from the beginning:

* *Is there a taxonomy of ML techniques and core functionalities to support the identification of which specific AI threats target ML algorithms?*

Specifically, data scientists (and other AI practitioners) and security engineers/expert need to share a common language, allowing them to more effectively threat model their AI systems before deploying to production; Security Incident Responders also need to have a bug bar to triage these net-new threats specific to AI/ML, see above.

* *What are the associated vulnerabilities?*
* *What are the techniques and the security controls to apply in order to address those vulnerabilities without jeopardizing the expected level of performance?*

## Modeling AI threats with MITRE ATLAS™

“If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.”

*-Sun Tzu, The Art of War*

Modeling AI specific threats can not only be quite tedious to do but also requires mastering the state-of-the-art of attack vectors. Indeed, it is a relatively new field, and the list of existing attacks is constantly evolving with new research publications.

This is why we can work with a framework that does this census work.

[MITRE ATLAS™](https://atlas.mitre.org/) (Adversarial Threat Landscape for Artificial-Intelligence Systems), is a knowledge base of adversary tactics, techniques, and case studies for Machine Learning (ML) systems based on real-world observations, demonstrations from ML red teams and security groups, and the state of the possible from academic research.

Une image contenant texte, ligne, diagramme, Police

Description générée automatiquement

*Figure 15: Global view of the ATLAS framework*

The image above represents the distribution of Machine Learning threats according to the type of attack executed. Each of the referenced threats is illustrated by a scenario in which the protocol of the attack execution is explained.

Let's take the example of yet another [adversarial attack](https://atlas.mitre.org/studies/AML.CS0012) carried out by MITRE's AI Red Team back in 2020. They demonstrated a physical-domain evasion attack on a commercial face identification service with the intention of inducing a targeted misclassification.

|  |  |  |
| --- | --- | --- |
| # | Technique | Description |
| 1 | [Search for Victim's Publicly](https://atlas.mitre.org/techniques/AML.T0000)  [Available Research Materials](https://atlas.mitre.org/techniques/AML.T0000) | The team first performed reconnaissance to gather information about the target ML model. |
| 2 | [Valid Accounts](https://atlas.mitre.org/techniques/AML.T0012) | The team gained access via a valid account. |
| 3 | [ML Model Inference API Access](https://atlas.mitre.org/techniques/AML.T0040) | The team accessed the inference API of the target model. |
| 4 | [Discover ML Model Ontology](https://atlas.mitre.org/techniques/AML.T0013) | The team identified the list of identities targeted by the model by querying the target model's inference API. |
| 5 | Acquire Public ML Artifacts: [Datasets](https://atlas.mitre.org/techniques/AML.T0002.000) | The team acquired representative open-source data. |
| 6 | [Create Proxy ML Model](https://atlas.mitre.org/techniques/AML.T0005) | The team developed a proxy model using the open-source data. |
| 7 | [Craft Adversarial Data: White-Box Optimization](https://atlas.mitre.org/techniques/AML.T0043.000) | Using the proxy model, the red team optimized a physical domain patch-based attack using expectation over transformation. |
| 8 | [Physical Environment Access](https://atlas.mitre.org/techniques/AML.T0041) | The team placed the physical countermeasure in the physical environment. |
| 9 | [Evade ML Model](https://atlas.mitre.org/techniques/AML.T0015) | The team successfully evaded the model using the physical countermeasure and causing targeted misclassifications. |

MITRE ATLAS™ allows to have a quick overview of potential threats around our model and especially of how they can be put into practice in the context of a realistic scenario.

However, even if having an idea of the threats to our model allows us to greatly reduce the number of vulnerabilities present in the AI system, the fact remains that the system in question is never 100% protected and that we must be prepared to respond according to the behavior of the AI system.

We first showed how a classical AI development lifecycle is built. Then, on such a foundation, we both considered and categorized the main threats specifically applying to these AI systems, and we finally introduced the more than welcome MITRE ATLAS™ knowledge base as a reference.

## Defining a suitable taxonomy for AI threats

During this chapter, we looked at the typical development cycle for ML-based projects and highlighted a number of types of attacks that can occur throughout the development lifecycle. We then saw the MITRE ATLAS™ framework that helps structure these attacks in a context where each of them is illustrated with a proof of concept that shows the ways in which these attacks can be implemented.

However, these attacks are part of a more general framework in which we can identify threats to AI systems.

Before we get into the details, it is worth defining the difference between a threat and an attack.

* A threat is a potential danger or harm that could happen.
* An attack is when that potential danger or harm actually happens.

Note that an attack can also be a form of threat, but the two notions are not interchangeable because a threat is not necessarily an attack.

Now that we have highlighted this duality between attack and threat, we will first see why it may be interesting to draw up a more complete picture of what may exist in terms of AI threats, and then we will see how the [European Union Agency for Cybersecurity (ENISA)](https://www.enisa.europa.eu/) proceeded to build its AI threat taxonomy.

### Why build a taxonomy of AI threats?

So far, we have highlighted a number of attacks that can target different places in the development pipelines of a ML-based project. Some of these attacks, depending on the context in which the project operates, can be devastating. The adversarial attack we performed on a computer vision model in charge of traffic sign recognition shows this perfectly.

Therefore, we will naturally want to defend ourselves against these potential attacks. We will want to reduce the risks of adversarial attacks, data poisoning attacks, etc. But *is this really the right way to proceed?*

**Machine Learning is a constantly evolving field, and it is even one of the fastest growing fields in terms of scientific advances these days. This means that today's attacks are probably not tomorrow's attacks. Or at least, they won't be the only ones. If we know how to protect ourselves against the attacks already discovered today, it does not necessarily mean that we will be able to do so for those discovered in the next few years.**

**This is exactly why having an AI threat taxonomy is useful. The idea of building this categorization of threats allows to put forward general concepts of types of threats in which today's attacks fit but especially in which tomorrow's discovered attacks could also fit.**

From there, rather than reinforcing the development cycle according to the attacks, we will rather have in mind the different categories of threats and try to limit their associated risks in order to reduce the impact of the attacks already existing but also of the attacks that will be discovered in the future.

### AI threat taxonomy according to ENISA’s report

“The significance and impact of AI in society nowadays cannot be overstated. It permeates every aspect of our daily lives and therefore it is of paramount importance to ensure the cybersecurity of AI to ensure that AI and the set of associated technologies will be trustworthy, reliable and robust.”

*-ENISA’s Artificial Intelligence Cybersecurity Challenge report*

For this purpose, we will refer to a report published by the above-mentioned ENISA entitled [Artificial Intelligence Cybersecurity Challenge](https://www.enisa.europa.eu/publications/artificial-intelligence-cybersecurity-challenges), which presents the agency's active mapping of the AI cybersecurity ecosystem and its Threat Landscape.

To build this AI threat taxonomy, ENISA has set up a high-level categorization of the different existing threats:

* Nefarious activity/abuse (NAA): “intended actions that target ICT systems, infrastructure, and networks by means of malicious acts with the aim to either steal, alter, or destroy a specified target”.
* Eavesdropping/Interception/ Hijacking (EIH): “actions aiming to listen, interrupt, or seize control of a third party communication without consent”.
* Physical attacks (PA): “actions which aim to destroy, expose, alter, disable, steal or gain unauthorised access to physical assets such as infrastructure, hardware, or interconnection”.
* Unintentional Damage (UD): unintentional actions causing “destruction, harm, or injury of property or persons and results in a failure or reduction in usefulness”.
* Failures or malfunctions (FM): “Partial or full insufficient functioning of an asset (hardware or software)”.
* Outages (OUT): “unexpected disruptions of service or decrease in quality falling below a required level“.
* Disaster (DIS): “a sudden accident or a natural catastrophe that causes great damage or loss of life”.
* Legal (LEG): “legal actions of third parties (contracting or otherwise), in order to prohibit actions or compensate for loss based on applicable law”.

Below, each category of threats from the previous list is illustrated with different examples of threats.



Figure : ENISA AI threat taxonomy ([source](https://www.enisa.europa.eu/publications/artificial-intelligence-cybersecurity-challenges))

This construction of the AI threat taxonomy allows to structure in distinct categories the different existing threats towards ML based projects.

The above report is further complemented by yet another report entitled [Securing Machine Learning Algorithm](https://www.enisa.europa.eu/publications/securing-machine-learning-algorithms). This report not only highlights core functionalities and critical stages, but also presents a detailed analysis of above AI threats targeting AI systems and proposes in turn concrete and actionable security controls described in relevant literature and security frameworks and standards.

**With that, and from there, we can eventually define and implement a set of additional activities to strengthen the previously discussed and described development lifecycle in order to raise the bar and move towards a (more) the trustworthy AI lifecycle, keeping in mind that it isn’t so much against the attacks themselves that we want to fight against but rather against the type of above threats in general.**

**One can’t stress enough that this doesn’t exclude an ongoing investments in both understanding the state-of-the-art and the know-how about attacks against AI systems and ensuring an adequate training of the involved team(s).**

If tomorrow an attack that does not belong to any category were to occur, it would be an imperative to be in a position to detect and respond to such an attack, by notably breaking the kill chain – According to [Wikipedia](https://en.wikipedia.org/wiki/Kill_chain), this is a military concept, which identifies the structure of an attack -.

# Module 3: Strengthening the “classic” (non-Generative) AI lifecycle

With all the previous considerations, we will consider how to strengthen our so-called “classic” (non-Generative) AI development lifecycle in order to raise the bar and move towards a (more) trustworthy AI lifecycle.

To do so, we will first look at a number of tools that will be useful before seeing how to apply them along with a list of best practices to be implemented in the development of projects based on Machine Learning.

## “Shift left security” in the “classic” AI lifecycle

“Shift left security” is at the heart of DevSecOps and aims at integrating security considerations and related activities and practices as early as possible into the development lifecycle with the end goal of improving the security posture of the released application and/or system.

A typical software development lifecycle starts from the requirements definition and the planning and comprises on that foundation five steps, namely the Design, the Development, the Integrating and Test, and the Deployment steps. The Requirements definition and the planning being at the left-most part of the cycle, anything that is moved towards them is shift left.

In the next sections we will look at all the activities that can be carried out during the development cycle in order to enforce our two principles, which are “Reliability and Safety” and “Privacy and Security”

## Discussing complementary activities and practices

This part aims to highlight a certain number of activities that can be implemented throughout the development cycle of a project based on artificial intelligence.

This list has (partially) been extrapolated from the [AI Risk Assessment - Best practices and guidance to secure AI systems](https://github.com/Azure/AI-Security-Risk-Assessment/blob/main/AI_Risk_Assessment_v4.1.4.pdf) document and it is NOT in any manner an exhaustive one.

It depends mainly on the state of progress of research in the field of AI security.

### Stage: Model requirements [MR]

#### Goal #1: Identifying the criticality level of the AI system

##### Activity [MR-G1-1]: Answering the right questions

Answering the right questions is important to have a good comprehension of the project for the continuation when it will be necessary to make decisions in terms of technological choices, development and more generally of the place of cybersecurity - vs. safety, a.k.a. functional security - in the AI system project.

###### How:

The first step is to determine the criticality level of the AI system. To do this, it is necessary to define the responsibilities of the project:

Towards the subject affected by the predictions:

* *Human?*
* *Company?*
* *Environment?*
* Etc.

Towards the data:

* *What is the level of criticality of the data?*
* *What about privacy?*
* *Is this data shared between several actors?*

Towards the actors involved

* *Who is involved in the development of the project?* Developers, data providers, private companies, ...
* *What are their constraints?*

###### What it allows?

Although everyone is aware of the importance of cybersecurity vs. safety, answering these questions early on can help avoid poor decision making that may be difficult to correct in the future.

#### Goal 2: Ensuring to react quickly in the case of a problem

##### Activity [MR-G2-1]: Document the boundaries of the model

During the Model requirements stage, the team in charge of the project has to document the boundaries of the model to define its expected functioning. This is necessary to know when the model's decisions can be questioned, and it improves both the reliability and safety of the overall AI system.

###### How:

Finding the boundaries of the model means answering some questions such as:

* *What are the characteristic values for normal operation?*
* *What are acceptable error rates for the overall performance of the system in the context of expectations?*
* *Etc.*

##### Activity [MR-G2-2]: Handling failures and remedies

When the model goes out of its nominal behavior, it is necessary to take a decision quickly. To do this, the team in charge of the project has to analyze the way the model leaves its normal operation and determine the appropriate actions to implement.

###### How:

* Defining predictable failures, including false positive, false negative and how they would impact users and stakeholders for each intended use.
* For each case of a predictable failure, document it:
  + When possible, build the system to avoid this failure. Describe the solution. Estimate the time range for resolving predictable failures for each designed solution.
  + When a failure cannot be prevented by design, build a fallback option. Describe the fallback option and document the estimated time required to invoke and use it.
  + Provide training and documentation for stakeholders accountable for system oversight that supports their resolution of the failure.
* Providing training and documentation for system owners, customer support and any other stakeholders responsible for managing the AI system to support their remediation and mitigation of predictable failures identified. Document the training and documentation provided.

#### Goal 3: Reducing the attack surface

##### Activity [MR-G3-1]: Ensuring proper rights management

Before starting the development of the AI system itself, it is necessary to define the access rights and limits correctly.

###### How:

To define these rights and limitations, some questions such as the following have to be answered:

Development side:

* *Who should have access to which stage of the model development pipeline (data collection, preprocessing, model training, etc.)?*
* *Who can have access to the model parameters?*
* *Etc.*

Deployment side:

* *What rights should the user be granted with respect to the model (access to the model inference, last layer, etc.)?*
* *Do they need to be authenticated? Do they have a limited number of possible calls per time unit?*
* *Who will be entitled to monitor (the users using) the AI system?*
* *Etc.*

##### Activity [MR-G3-2]: Choosing the right technologies

Yet another step before proceeding with the actual development is to determine the right technologies to use and this has to be done with the criticality of the project in mind.

###### How:

Depending on the characteristics of the project and its level of criticality, it is important to make an appropriate choice of technologies to be implemented.

* *Which Machine Learning Framework and platform will be used?*
* *Should the intended AI system be run in a cloud? on-premises? at the edge?*
* *Which Machine Learning Framework will be used?*
* *Is the* protection *of the confidentiality of the training data required? From whom? Regarding what identified risk(s) if any?*
* *Ibid for the production data?*
* If yes, for the state of data:

1. Data at rest: *i) where to store/access the data, ii) how to encrypt the data, is a multiple layer approach required to mitigate different identified risks? and iii) how to securely protect, release, and renew the encryption key(s), who will own and manage these keys, etc.*
2. Data in transit: *how to encrypt the data?*
3. Data in use: *when, how, and form whom to protect the data? With what level of granularity?*

* *Is there a requirement to (also) protect the IP of the model during the inference?*

### Stage: Data Collection [DC]

#### Goal 1: Ensuring data quality and availability

##### Activity [DC-G1-1]: Be demanding with the choice of data sources

For the model to work properly, it is important to carefully select the data used for training and evaluation.

###### How:

Some possible strategies could be applied to this end:

* Limit the number of data sources as appropriate.
* Implement anomaly detection with statistical tests.
* Do manual moderation.
* Use a Golden dataset which refers to data of very high quality.

#### Goal 2: Improving the level of privacy

##### Activity [DC-G2-1]: Leverage privacy-preserving machine learning (PPML) techniques

The model sees a large amount of data flowing through. It is possible, depending on the criticality of the project, that this data is sensitive, confidential, etc. In this case, it is advisable to use adapted technologies.

###### How:

* Leverage privacy-preserving machine learning (PPML) techniques such as differential privacy in the early stage of model development. See [A Brief Survey of Privacy Preserving Technologies](https://www.statcan.gc.ca/en/data-science/network/privacy-preserving) for more information.
* Use data lineage process.
* Use datasheets to document used datasets.

### Stage: Data Cleaning [DCL]

#### Goal 1: Ensuring data quality

##### Activity [DCL-G1-1]: Do not trust your data or where it comes from

The data cleaning stage is one of the few stages in which we are interested in the constitution of the data. It is therefore necessary to take advantage of it to make sure that the dataset does not present any compromising entries.

###### How:

* Determine the types of outliers in the dataset and their proportion.
* Determine the threat level of these outliers: *Were they introduced intentionally or is it a recording error?*
* Make an appropriate decision regarding the dataset: *Can we continue to work with it or not?*

### Stage: Data Labelling [DL]

#### Goal 1: Ensuring data privacy

##### Activity [DL-G1-1]: Do not trust your data or where it comes from

When receiving data that is not already labeled, it may be tempting to hire specialized companies to perform this task as it is often time consuming and repetitive. However, this represents a potential risk of data leakage.

###### How:

* Determine the size of your dataset.
* Approximate how long it will take to label the data in question.
* Decide whether or not outsourcing this task can be cost-effective after taking these different aspects into account.

### Stage: Model Training [MT]

#### Goal 1: Ensuring the model is robust

##### Activity [MT-G1-1]: Strengthening the model

To protect the model against external threats such as adversarial attacks, the development team may need to strengthen it.

###### How:

* Introduce statistical noise in the training dataset to make it less sensitive to noise and therefore to adversarial attacks.
* Train the model on a dataset augmented with adversarial input.
* Use Randomized Discretization (or other methods) to defend against white-box adversarial attacks

### Stage: Model Evaluation [ME]

#### Goal 1: Ensure that metrics are relevant

##### Activity [ME-G1-1]: Use metrics in addition with performance metrics

The choice of metrics to be observed during the evaluation is important because it conditions part of the adequacy of the model with the security requirements.

###### How:

* Do not be satisfied with accuracy and/or mean squared error but also implement metrics and statistical tests to follow the evolution of model predictions distribution.

### Stage: Model Deployment [MD]

#### Goal 1: Ensuring a successful deployment

##### Activity [MD-G1-1]: Verify that the requirements are met

Between the establishment of the requirements and the deployment phase, it can take quite some time. Time during which the project itself can mature and evolve.

The deployment phase is a critical phase since it is this stage that sees the model reach the user and therefore potentially open to external risks. Therefore, it is important to check that all the requirements previously established are verified before deployment.

###### How:

* Retrieve the list of previously established requirements and ensure that the project validates them all

#### Goal 2: Ensuring project availability

##### Activity [MD-G2-1]: Perform stress testing

Once in production, the model will have to hold the load in terms of uptime and response capacity. We should be able to ensure this beforehand.

###### How:

* Perform **a progressive stress test on the system’s endpoints, whether it is the model inference or other endpoints that are not specific to ML and ensure it can reach its theorical load.**
* Make sure that the project can operate in good condition when the maximum theoretical load is reached.

### Stage: Model Monitoring [MM]

#### Goal 1: Ensuring that the model is still relevant

##### Activity [MM-G1-1]: Monitoring metrics over time

In the case of a data poisoning attack, the model can become less and less relevant until it is totally compromised. It is important to monitor the relevance of predictions over time.

###### How:

* Re-evaluate the metrics used for the evaluation stage throughout the life of the model.
* Implement statistical tests to measure data shift.
* Use monitoring tools like RiverML particularly suited for data stream analysis.

#### Goal 2: Ensure that the architecture is still fully operational

##### Activity [MM-G2-1]: Monitor the whole system over time

For a ML model to work in good conditions, it is necessary to have a healthy environment in which the model can run. To ensure the environment is functional in respect to any prior expressed KPIs, along with the ability to examine the recorded state of specific classifiers which may have led to a decision, it is necessary to implement AI/ML event tracing facilities.

Such event tracing facilities are needed to prove the correctness and transparency of AI/ML-generated decisions whenever called into question.

In addition, in high-value scenarios, as already stressed, security logging on what is going on to prevent potential attacks and/or to break the kill chain if an attack occurs to aid in detecting, responding to/recovering from such attack, preventing it to occur again as well as appropriate AI/ML forensic.

###### How:

* Log all events taking place in the system keeping track of components, infrastructure, and network status.

[Azure Monitor](https://azure.microsoft.com/fr-fr/services/monitor/) can help you maximize the availability and performance of your systems. It delivers a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. This information helps you understand how your systems are performing and proactively identify issues that affect them and the resources they depend on.

* Implement and leverage AI/ML event tracing facilities. This can start with the correlation of basic decision-making information such as – in a non-exhaustive manner - :
* The timeframe in which the last training event occurred.
* The timestamp of the most recent dataset entry trained upon.
* Weights and confidence levels of key classifiers used to arrive at high-impact decision.
* The classifiers or components involved in the decision.
* The final high-value decision reached by the model.

One should say that such tracing is overkill for the majority of algorithm-assisted decision making. However, having the ability to identify the data points and algorithm metadata leading to specific results will be of great benefit in high-value decision making. Such capabilities will not only demonstrate trustworthiness and integrity through the algorithm’s ability to “show its work”, but this data could also be used for fine-tuning as well.

* Implement security logging and appropriate forensic to provide integrity, transparency, accountability, and in some instances, evidence where civil or criminal liability may arise.

One should underline that these capabilities will be of tremendous value when paired with data visualization techniques allowing the auditing, debugging and tuning of algorithms/models for more effective results.

* Implement [extended detection and response (XDR)](https://www.microsoft.com/en-us/security/business/security-101/what-is-xdr) capabilities, which notably includes abnormal behavior detection, to deliver intelligent, automated, and integrated security across domains to help connect seemingly disparate alerts and get ahead of attackers, as well as threat hunting as part of your SIEM infrastructure if any.

[Microsoft Defender for Cloud](https://azure.microsoft.com/en-us/services/defender-for-cloud/) and [Microsoft Sentinel](https://azure.microsoft.com/en-us/services/microsoft-sentinel/?culture=en-us&country=US) can be help you implementing such capabilities.

## Considering tools, frameworks, and techniques of interest

### Designing Human–AI interaction with HAX Toolkit

In Machine Learning, incident response is the process of identifying, assessing, and responding to incidents that involve ML models. This includes identifying the root cause of an incident, assessing the impact of the incident, and taking steps to mitigate or resolve the incident.

Incident response in AI/ML is important because ML models might be used in high-stakes situations, such as decision-making in healthcare or finance. If an incident occurs, it is important to respond quickly and effectively to minimize the impact.

There are a few steps that should be taken in incident response :

1. First, the root cause of the incident should be identified. This may involve looking at the data that was used to train the model, the model itself, and the environment in which the model is deployed.
2. Second, the impact of the incident should be assessed. This includes understanding how the incident will affect the accuracy of the model and the business that is using the model.
3. Finally, steps should be taken to mitigate or resolve the incident. This may involve retraining the model, changing the data that is used to train the model, or changing the environment in which the model is deployed.

Incident response in AI/ML is important to ensure that ML models are used effectively and safely. By taking steps to identify, assess, and mitigate incidents, we can ensure that ML models are used in the best way possible.

The [Human-AI eXperience (HAX) Toolkit](https://www.microsoft.com/en-us/haxtoolkit/) is a set of practical tools for creating human-AI experiences with people in mind from the beginning. Each tool is designed to help AI creators, including UX, AI, project management, and engineering teams, take this human-centered approach in their day-to-day work.

Among the components of the toolkit, several guidelines address how to interact with an AI model in its state. These are divided into several categories that we will investigate further.

#### Initially



*Figure 5: Guidelines for stage: "Initially"*

Guidelines describing the behaviors to be adopted before the development of the model. These guidelines emphasize the need to clearly define the objective of the model and its scope of action. Where it is supposed to be good and where it is not.

#### During interaction



*Figure 6: Guidelines for stage: "During interaction"*

Guidelines describing the behaviors to adopt during the interaction. These guidelines emphasize the need to ensure that the model works properly. This is done by managing the running time and by monitoring the model in order to highlight potential bias problems for example.

#### When wrong



*Figure 7: Guidelines for stage: "When wrong"*

These guidelines emphasize the need to always have control over the model. It is necessary to be able to deactivate, reactivate and debug the model easily. It is also important to be able to log all the activity in order to understand what went wrong.

In our case, this section is particularly of interest as it highlights a number of concepts to be applied to ensure the best possible response to a problem situation so let’s dive into the actual recommendations:

* Support efficient invocation. To ensure people can recover if an AI system does not activate when needed or as expected, make it easy for the user to manually invoke its services. For example, for an AI-powered writing assistant that can suggest more inclusive phrasing, make it easy for the user to manually request the assistant’s suggestions in case it fails to trigger when desired.
* Support efficient dismissal. To ensure people can recover if the AI system activates when not needed or expected, make it easy for the user to dismiss or ignore its services. For example, for an AI-powered voice assistant, make it easy for the user to dismiss the assistant in case it mistakenly triggers or acts in inappropriate situations.
* Support efficient correction. Sometimes AI systems will be only partially correct. To ensure people can still achieve their goals when this happens, make it easy for the user to edit the AI system’s outputs. For example, for an AI-powered navigation app that can recommend fast routes to a user’s destination, make it easy for the user to manually edit a recommended route or part of the route (e.g., if the user wants to avoid an area they know is under construction).
* Scope services when in doubt. In ambiguous situations, less can be more. For example, for an AI-powered assistant that can call people on demand, if the assistant is unsure whom to call, requesting clarification (e.g., “Do you mean Bill G. or Bill C.?”) can be less costly than calling the wrong person. In these situations, build the AI such that it can compute its own uncertainty and use this information to gracefully degrade or scope its services when in doubt.
* Make clear why the system did what it did. Make available an explanation for the AI system’s actions/outputs as appropriate.

#### Over time



*Figure 8: Guidelines for stage: "Over time"*

Guidelines describing the behaviors to be adopted over time. These guidelines mainly emphasize the need to listen to user feedback in order to monitor the model and its evolution in the best possible conditions.

#### Using the guidelines

The use of the tool is relatively simple. Each guideline comes with a detailed explanation that specifies the answer to be given depending on the context.

Graphical user interface, text, application

Description automatically generated

*Figure 9: Content of guideline n°1*

[Guideline No. 1](https://www.microsoft.com/en-us/haxtoolkit/guideline/make-clear-what-the-system-can-do/), deals with the fact that the scope of the model must be well defined before its deployment in production. By exploring the content of the guideline, more information on what this means can be found.

### Protecting the data privacy

Everyone agrees that protecting the privacy of user data is necessary. In the era of the explosion of data science, data is considered as gold and the different market players are fond of it. But users, on an individual level, are more reticent about how that data can be used. No one wants to see their bank details or medical records leaked on the internet. Therefore, dealing with the privacy aspect of the data used seems inevitable. But one must know how to do it properly.

Let’s see the old way of doing anonymization. When we want to anonymize a dataset, one of the first things to do is to remove (or not ask for) any data that directly identifies a user such as a name, address, or phone number. This way, every entry appears anonymous since there is seemingly no direct way to associate an entry with a user. This is exactly the decision taken by Netflix to anonymize the dataset used in the Netflix Prize competition in 2006.

The dataset contained more than 100,000,000 ratings from about 500,000 different users on more than 17,000 different movies. The dataset was divided into a training dataset and a test dataset. It was composed of the following features:

* User : anonymized variable, it was not the real pseudonyms used
* Movie: the name of the movie that was graded
* Date of grade: the date of when the movie was graded
* Grade: the grade given to the movie, it is a number of stars between 1 and 5. This is also the feature to predict, therefore this column is in the training dataset but not in the test dataset.

The objective of the competition was to predict the rating that a movie would get for a given user. For this purpose, the database was anonymized mainly by replacing the users' names with a string of characters. But this wasn’t enough.

In 2008, a research paper entitled “[How To Break Anonymity of the Netflix Prize Dataset](https://arxiv.org/abs/cs/0610105)” is released. This paper explains how they were able to identify the allegedly anonymized users in the Netflix competition dataset using an auxiliary dataset from IMDb performing what is called a Linkage attack. This type of attack occurs when several seemingly anonymous datasets are combined to reveal real identities.

The Netflix example is interesting because it highlights the reasons why this aspect of the data scientist's work should be taken seriously. It's a pretty funny example and without serious consequences but things could have been more complicated if instead of movie notes, it was medical data we were talking about.

Now let's see what we can do about it. First, it is worth remembering that traditional methods are still valid. To protect the confidentiality of the data used, it is important to avoid retrieving identifying data such as gender, postal code, language spoken, etc., as these data can be used to re-identify a person when combined. This is therefore an important first step. But in this part, we go a little further by presenting an additional method to reinforce this protection.

Today, there are techniques to anonymize a dataset more efficiently and among them, we are going to focus on differential privacy.

To introduce the concept of differential privacy, we will illustrate this with an example.

Let’s imagine that we would like to know how many people children read books. To do this, we could simply ask the following question to a representative sample:

*Do you read books?* The answer to this question could only be “Yes” or “No”.

But storing the answer to this question as such poses privacy problems as we saw with the example above.

To deal with this problem, let's imagine that after getting the answer to this question we randomly flip a coin.

* If the coin lands on tails: we record the given answer.
* If the coin lands on heads: we record a random answer between Yes or No independently of the given answer.

In this way, after collecting enough data, if we focus our attention on only one entry in the dataset, it is impossible to know whether the given answer is the real answer of the respondent or a random answer. But when we take the data as a whole, it remains coherent.

Graphical user interface

Description automatically generated

Figure : A simplified representation of the entry registration process

Differential privacy is based on the idea of adding noise to data in order to protect the privacy of individuals. The amount of noise added is based on the sensitivity of the data. The more sensitive the data, the more noise is added.

It has a number of advantages over other privacy protecting methods.

* It is very difficult to attack.
* It provides a way to measure the privacy of individuals in a data set.
* It is possible to use differential privacy in a way that does not require any changes to the data collection process.

To implement differential privacy, there are tools that can help. Here are some tools that might interest you.

* [SmartNoise](https://smartnoise.org/) is a toolkit that uses state-of-the-art differential privacy (DP) techniques to inject noise into data, to prevent disclosure of sensitive information and manage exposure risk. Note that the Smart Noise project has been renamed to [OpenDP](https://opendp.org/) for Open Differential Privacy. The project is open source, and you can find the repository by following this [link](https://github.com/opendp).
* [TensorFlow Privacy](https://github.com/tensorflow/privacy) is a Python library that includes implementations of TensorFlow optimizers for training ML models with differential privacy.
* [Opacus](https://github.com/pytorch/opacus) is a library that enables training PyTorch models with differential privacy. It supports training with minimal code changes required on the client, has little impact on training performance, and allows the client to online track the privacy budget expended at any given moment.

#### Implementing privacy by design

Finally, when we work with data, we should assume that we are not the only ones to have access to it, and even if this is the case at a given moment, we must anticipate a potential leakage of this data. To do this, we can implement privacy by design by following these basic principles:

* Develop rigorous access control.
* Use robust de-identification techniques with technologies such as differential privacy.
* Collect and process the smallest possible amounts of data in terms of personal identifiers associated with the data.
* Avoid quasi-identifiers and non-unique identifiers so that individuals can be re-identified when combined.

#### Data Lineage

In this guide we mainly talk about the AI development lifecycle in its globality. But as illustrated by the notions of pipeline and loop, there are other development cycles included in the AI development lifecycle. One of the most important is the data lifecycle. To understand how data lineage can be interesting to put into practice, let's briefly study the functioning of the data lifecycle.

##### Data lifecycle

The data lifecycle is a process that data goes through from its creation to its eventual deletion. It is a cycle because once data is deleted, the loop can start again. The data lifecycle has mainly four stages: creation, storage, processing, and deletion.

* Creation is the first stage of the data lifecycle. This is when data is first collected and created. It can be created manually, such as when a user fills out a form, or automatically, such as when a sensor collects data.
* Storage is the second stage of the data lifecycle. This is when data is saved so that it can be accessed later. Data can be stored locally, such as on a user's computer, or remotely, such as in the cloud.
* Processing is the third stage of the data lifecycle. This is when data is analyzed and transformed into useful information. Processing can be done manually, such as when a user sorts through a list of data, or automatically, such as when a computer program runs a complex analysis such as machine learning algorithms.
* Deletion is the fourth and final stage of the data lifecycle. This is when data is no longer needed and is removed from storage. Data can be deleted manually, such as when a user deletes a file, or automatically, such as when a computer program deletes old data such as temporary files that are no longer needed.

Figure : Data lifecycle

Since data is the very reason why machine learning technologies work, special attention must be paid in terms of security since it is also a gateway for the user to the heart of the system. This is where data lineage comes in.

##### Data lineage concept

Data lineage is the process of tracking the data as it moves through the different stages of the data lifecyle. It includes the origin of the data, the transformation that it undergoes, and the destination of the data. Data lineage provides a way to trace the data back to its source, and to understand how it has been transformed over time.

This process is very important for a number of reasons.

1. First, it can help to identify errors in the data.
2. Second, it can help to understand the impact of changes to the data.
3. Finally, it can help to ensure that the data is used correctly.

Data lineage is typically represented as a directed graph. The nodes in the graph represent the different stages in the data lifecyle, and the edges represent the data flow between the stages.

The data lineage graph can be used to answer a number of questions, such as:

* *What is the source of the data?*
* *What are the transformations that the data has undergone?*
* *What is the destination of the data?*
* *What is the impact of a change to the data?*

##### Document dataset

The machine learning community currently has no standardized process for documenting datasets, which can lead to severe consequences in high-stakes domains. To address this gap, Microsoft Research published a research paper entitled [Datasheet for Datasets](https://arxiv.org/abs/1803.09010).

In the electronics industry, every component, no matter how simple or complex, is accompanied with a datasheet that describes its operating characteristics, test results, recommended uses, and other information. By analogy, it is proposed that every dataset be accompanied with a datasheet that documents its motivation, composition, collection process, recommended uses, and so on. Datasheets for datasets is built to facilitate better communication between dataset creators and dataset consumers and encourage the machine learning community to prioritize transparency and accountability.

The research paper is mainly written around questions on different topics that need to be addressed. Let's take the example of motivation. Here are some questions that should be answered.

* *For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled?*
* *Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?*
* *Who funded the creation of the dataset?* *Is there an associated grant?*

For each of the categories mentioned you can find a list of questions relevant to the use of data sets.

##### Use appropriate tools.

Among the different data lineage tools, [Microsoft Purview](https://www.microsoft.com/en-us/security/business/microsoft-purview) provides a unified data governance solution to help manage and govern your on-premises, multicloud, and software as a service (SaaS) data. It is possible to easily create a holistic, up-to-date map of your data landscape with automated data discovery, sensitive data classification, and end-to-end data lineage.

See [Data lineage in Microsoft Purview](https://docs.microsoft.com/en-us/azure/purview/concept-data-lineage) for more information.

### Raising the level of guarantees of AI system development vs. deployment environment

Data privacy and security is important when it comes to business and consumer data. Protecting data in use is one way to increase data privacy and security.

This unlocks new scenarios such as multiparty data analytics where Machine Learning can help you discover new opportunities while keeping data private among participants.

[Azure Confidential Computing](https://azure.microsoft.com/en-us/solutions/confidential-compute/) provides data security using trusted execution environments or encryption, providing protection of sensitive data across the AI development lifecycle.

* Protecting data in use: Secure your sensitive and regulated data while it's being processed in the cloud. Azure confidential computing encrypts data in memory in hardware-based trusted execution environments and processes it only after the cloud environment is verified, helping prevent data access by cloud providers, administrators, and users. Build on top of secure hardware using familiar tools, software, and cloud infrastructure, or migrate your existing workloads and applications.
* Sharing AI insights confidentially: Combine datasets confidentially, without exposing your data to other contributing organizations. Benefit from confidential computing and great AI and Machine Learning insights. Upload encrypted data to a secure enclave in a virtual machine and perform algorithms on datasets from multiple sources.
* Owning your data: Migrate to the cloud and keep full control of your data in a trusted execution environment. Specify the hardware and software that have access to your data and code, and verifiably enforce this guarantee. Customers retain control over their protected information so they can meet government regulations and compliance needs. Customize your confidential computing path using tools and solutions built in Azure, in open-source frameworks, and by independent software vendor partners.
* Protecting the IP of your model.

See [Azure confidential computing](https://docs.microsoft.com/en-us/azure/confidential-computing/) and [Use confidential computing with Azure Machine Learning](https://azure.microsoft.com/en-us/blog/azure-confidential-computing/) for more information.

Among the broad range of confidential compute offerings, including hardware, services, SDKs, and deployment tools offered by Azure Confidential Computing, here are some relevant to a project based on Machine Learning:

* **Application Enclaves. Enclaves are secured portions of the hardware's processor and memory. You can't view data or code inside the enclave, even with a debugger. If untrusted code tries to change content in enclave memory, the environment is disabled, and operations are denied. These unique capabilities help you protect your secrets from being accessible in the clear.**

Think of an enclave as a secured lockbox. You put encrypted code and data inside the lockbox. From the outside, you can't see anything. You give the enclave a key to decrypt the data. The enclave processes and re-encrypts the data, before sending the data back out. Such enclaves can be leveraged for inferencing a ML model.

See [Application enclave development](https://docs.microsoft.com/en-us/azure/confidential-computing/application-development) for more information.

* Confidential containers. Azure provides confidential containers which offer a secure and isolated environment with attestation to improve the security of your overall container deployment.

With Azure's confidential container offerings, you can verify that your container applications are running in a verifiable execution environment. Additionally, these containers support custom applications written in any programming language, as well as off-the-shelf Docker container applications. As an illustration, confidential containers can also be leveraged for inferencing.

See [Confidential containers](https://docs.microsoft.com/en-us/azure/confidential-computing/confidential-containers) for more information.

* **Confidential VMs**. Azure confidential VMs offer a new and enhanced disk encryption scheme. This scheme protects all critical partitions of the disk. It also binds disk encryption keys to the virtual machine's TPM and makes the protected disk content accessible only to the VM. These encryption keys can securely bypass Azure components, including the hypervisor and host operating system. To minimize the attack potential, a dedicated and separate cloud service also encrypts the disk during the initial creation of the VM.

See [Latest innovations in Azure confidential computing](https://techcommunity.microsoft.com/t5/azure-confidential-computing/latest-innovations-in-azure-confidential-computing/ba-p/3573389) and [DCasv5 and ECasv5 series confidential VMs](https://docs.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview) for more information.

**Confidential GPUs** (in preview). NVIDIA and Microsoft are collaborating to enable GPU-accelerated computing with confidential computing. With support for [Ampere Protected Memory (APM) in NVIDIA A100 Tensor Core GPUs](https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/) and hardware-protected VMs, enterprises will be able use their preferred machine learning frameworks with an added layer of security that allows the used VM to boot and run in a trusted environment. As a result, you know that the confidentiality of your data remains encrypted while you leverage the performance of the GPU for your workloads.

See [Azure confidential computing with NVIDIA GPUs for trustworthy AI](https://azure.microsoft.com/en-us/blog/azure-confidential-computing-with-nvidia-gpus-for-trustworthy-ai/) and [Powering the next generation of trustworthy AI in a confidential cloud using NVIDIA GPUs](https://www.microsoft.com/en-us/research/blog/powering-the-next-generation-of-trustworthy-ai-in-a-confidential-cloud-using-nvidia-gpus/) for more information.

#### An illustration of how to leverage these tools

The development phase and the deployment phase of a model are two very different things. However, although they are not the same, both must be carried out in good security conditions if we want to avoid bad surprises. A simple example to illustrate this is enterprise computers.

Using confidential VMs as a development environment. Sometimes, when a company hires a developer, it provides him with a work computer. Among the reasons for doing this, there is the separation of private and professional life, which is a good reason, but there is another important one. Security. By having an environment specially adapted for work, the risks of getting infected are reduced.

If a data scientist gets infected during the data aggregation phase, i.e., before he or she can apply anonymization methods, this can have dramatic consequences. Therefore, it is necessary to work in a secure environment and even more so when working with potentially sensitive data.

Confidential VMs can be useful for this purpose. A data scientist could create his development environment in a secure virtual machine so that he can properly control his development environment and ensure for example that the operating system he is working on has not been compromised. In this environment, developers will be able to work on each step of the inner loop with a reduced risk of infection.

Using confidential containers for model inference. Using confidential containers can be beneficial when deploying the model to production. This improves the success rate of the model deployment and limits the dependencies and access rights to the image/container to the necessary. There are predefined docker images compatible with confidential containers for all use cases. Below are some of them you could use depending on the libraries and frameworks you would like to see pre-installed.

##### Predefined Docker images for TensorFlow models inference

|  |  |  |  |
| --- | --- | --- | --- |
| Framework version | CPU/GPU | Pre-installed packages | MCR Path |
| 1.15 | CPU | pandas==0.25.1  numpy=1.20.1 | mcr.microsoft.com/azureml/tensorflow-1.15-ubuntu18.04-py37-cpu-inference:latest |
| 2.4 | CPU | numpy>=1.16.0  pandas~=1.1.x | mcr.microsoft.com/azureml/tensorflow-2.4-ubuntu18.04-py37-cpu-inference:latest |
| 2.4 | GPU | numpy >= 1.16.0  pandas~=1.1.x  CUDA==11.0.3  CuDNN==8.0.5.39 | mcr.microsoft.com/azureml/tensorflow-2.4-ubuntu18.04-py37-cuda11.0.3-gpu-inference:latest |

##### Predefined Docker images for PyTorch models inference

|  |  |  |  |
| --- | --- | --- | --- |
| Framework version | CPU/GPU | Pre-installed packages | MCR Path |
| 1.6 | CPU | numpy==1.20.1  pandas==0.25.1 | mcr.microsoft.com/azureml/pytorch-1.6-ubuntu18.04-py37-cpu-inference:latest |
| 1.7 | CPU | numpy>=1.16.0  pandas~=1.1.x | mcr.microsoft.com/azureml/pytorch-1.7-ubuntu18.04-py37-cpu-inference:latest |

##### Predefined Docker images for SciKit-Learn models inference

|  |  |  |  |
| --- | --- | --- | --- |
| Framework version | CPU/GPU | Pre-installed packages | MCR Path |
| 0.24.1 | CPU | scikit-learn==0.24.1  numpy>=1.16.0  pandas~=1.1.x | mcr.microsoft.com/azureml/onnxruntime-1.6-ubuntu18.04-py37-cpu-inference:latest |

##### Minimal predefined Docker image

|  |  |  |  |
| --- | --- | --- | --- |
| Framework version | CPU/GPU | Pre-installed packages | MCR Path |
| / | CPU | No packages | mcr.microsoft.com/azureml/minimal-ubuntu18.04-py37-cpu-inference:latest |

### Other useful tools you may want to consider

The list of tools is obviously not exhaustive, but here are some other tools you might want to consider.

#### ONNX: an open format built to represent ML models

[ONNX](https://onnx.ai/) is an open format built to represent ML models using a common set of operators, some building blocks of machine learning and deep learning, and a common file format to enable AI developers to use models with a variety of frameworks, tools, runtimes, and compilers.

ONNX is also a runtime is a high-performance inference engine for deploying ONNX models to production.

You might want to use ONNX if you’re looking for:

* Interoperability: ONNX offers the ability to use any framework of your choice to develop your ML model
* Hardware optimization : ONNX eases access to hardware optimizations.

See [Confidential ONNX inference server](https://github.com/microsoft/onnx-server-openenclave), a collaboration between Microsoft Research, Azure Confidential Compute, Azure Machine Learning, and Microsoft’s ONNX Runtime project to showcase a hosting possibility which restricts the ML hosting party from accessing both the inferencing request and its corresponding response.

#### RiverML: a python library for online Machine Learning

[River](https://riverml.xyz/0.11.1/) is a Python library that allows for online Machine Learning. This means that the library can be used to process streaming data, which can be more convenient than using a batch model. The library is also designed to be robust against concept drift in dynamic environments. River supports different Machine Learning tasks, including regression, classification, and unsupervised learning.

#### AI-specific security scanner

Anyone can download machine learning models embedded with malware from repositories online, which could lead to hackers taking hold of a company’s environment. To avert this, Microsoft collaborated with Hugging Face on an [AI-specific security scanner](https://github.com/mmaitre314/picklescan) that detects Python Pickle files performing suspicious actions.

#### Software Bill of Materials (SBOM)

SBOM, a list of all the components and dependencies that make up a software product, is recommended for AI systems. SPDX and CycloneDX are the leading SBOM standards which Identify known vulnerabilities in components and track ML models using [purl](https://github.com/package-url/purl-spec), that now include Hugging Face and MLFlow specifications.

## Towards a (more) Trustworthy AI lifecycle

In this guide you will have understood that we were particularly interested in strengthening the different phases of the lifecycle that make up the development and deployment of the ML model in accordance to our “North Star”. Let’s wrap up everything here to have a better understanding on what we’ve worked on.

### Improving the inner loop

As a reminder, the inner loop is the part that consists in developing the model itself. It consists of the "Data Pipeline" in which the data is received and then processed; and the "Modeling Pipeline" which consists of building the machine learning model by optimizing the results of a previously chosen metric that depends on the problem. It is a loop because the stages that make it up are repeated until we obtain a model that satisfies the business analysis.

In this guide, we have highlighted a certain number of vulnerabilities specific to these steps and, based on this, we have been able to develop activities to reduce the risks.

Here is a summary of activities to strengthen the inner loop.

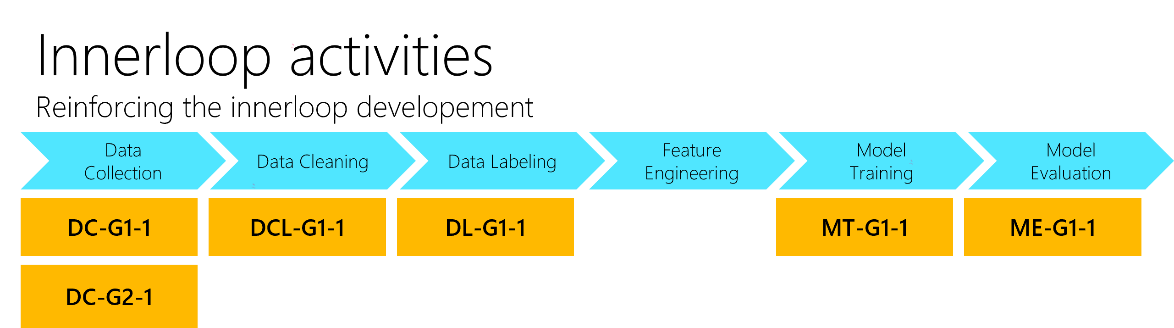


Figure : Activities to reinforce the inner loop

### Improving the outer loop

As a reminder, the outer loop is the part that consists of establishing the requirements of the model itself as well as its deployment in production and its monitoring. This loop also includes the development of functionalities around the model and their delivery to the user through the stages of Continuous Integration (CI) and Continuous Deployment/Delivery (CD).

In this guide, we also have highlighted a number of vulnerabilities specific to these stages and, on this basis, we have been able to develop activities to reduce the risks.

Below is a summary of the activities to strengthen the outer loop.

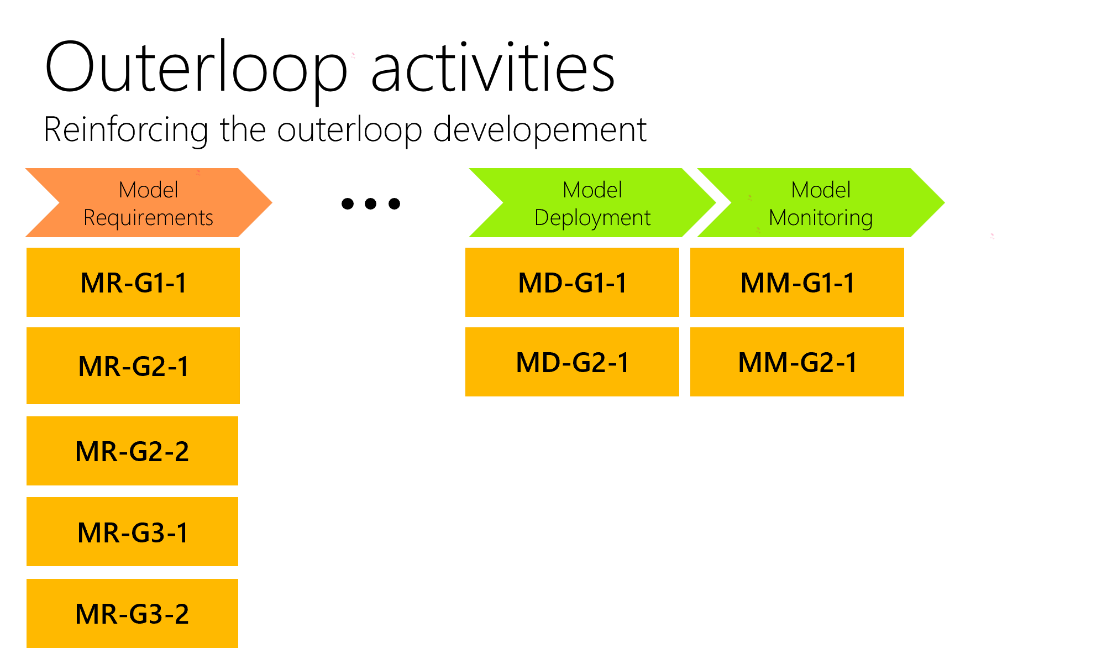


Figure : Activities to reinforce the outer loop

### Rules of thumb

There is no single trustworthy AI lifecycle. Depending on the needs you have identified, you can build a lifecycle adapted to your own projects and their specifics. In fact, you should establish your own “North Star” in the same way we were able to do in this guide. This involves finding answers to questions such as:

* *Does my project really need this technology?*
* *How important are the Reliability and Safety aspects?*
* *How much importance do I want to give to Privacy and Security aspects?*

The answer to these questions depends on several parameters of your model, especially the level of criticality and responsibilities that will fall on the future model.

To help you build your North Star and adapt your development cycle to the best practices we have highlighted in this guide, here are some rules of thumb that you can keep in mind during the development of the project.

#### Rule #1: Security from the very beginning

A (more) trustworthy AI lifecycle takes into account cybersecurity issues from the very beginning of the project.

This corresponds to our moto “Shift left security” as already adopted by Dev(Sec)Ops practices.

It asks the right questions at the right time. It defines the boundaries of the project to anticipate future problems. For each anticipated problem, it provides solutions or alternatives in advance in order to bring a quick answer and reduce the impact of the problem on the user, the company or the stakeholders.

#### Rule #2: Use proven tools

A (more) trustworthy AI lifecycle uses proven tools rather than reinventing the wheel.

In fact, it can be tempting to reinvent your own solutions to an existing problem, but this is often an excellent way to introduce vulnerabilities into the core of the project. When developing a trusted AI, one will tend to prefer to use tools that have been proven, audited, tested and shown to be robust against cybersecurity attack over time.

#### Rule #3: Cybersecurity activities

A Trustworthy AI lifecycle implements a set of security-oriented activities throughout the development process. Indeed, it is not only talking about security from the beginning of the project and using the right tools. Each stage of the development cycle must come with its own set of activities to be implemented to ensure continuity in cybersecurity considerations. The activities we have discussed in this guide can be applied throughout the AI development cycle but feel free to supplement them with other sources according to your own requirements.

The aforementioned document [AI Risks Assessment - Best practices and guidance to secure AI system](https://github.com/Azure/AI-Security-Risk-Assessment/blob/main/AI_Risk_Assessment_v4.1.4.pdf) provides you with a suitable approach and additional activities to enable you to perform an advanced risk analysis on your AI systems, and the related development lifecycle.

#### Rule #4: Perfectly secure development cycle doesn’t exist

There is no things such as a perfectly secure development lifecycle. Just as we recommend doubting everything and everyone from a security point of view in a classic development cycle, we can also apply this ideology in the context of a ML-based project development.

Assume that there is a non-zero probability that your model and the confidential data you use could be compromised one day or revealed to the world by techniques that may not even exist today. Your goal is simply to postpone this date as long as possible, or better still, to make sure that in the worst-case scenario where this situation should happen, to minimize the consequences of such an event beforehand.

Challenging this suggested approach

The approach to strengthen the development lifecycle may be subject to variation or further development depending on the importance you place on it.

You may have heard of the terms blue team and red team in cybersecurity. This is a concept to enhance the security level of an IT project.

* Blue team refers to the group of people protecting an organization's infrastructure, its (AI) systems data.
* Conversely, red team refers to the group of people who attack these assets and find to the ends weaknesses and vulnerabilities to exploit in a certain way.

See “The dynamic duo: How to build a red and blue team to strengthen your cybersecurity”, [part 1](https://www.microsoft.com/security/blog/2021/01/05/the-dynamic-duo-how-to-build-a-red-and-blue-team-to-strengthen-your-cybersecurity-part-1/) and [part 2](https://www.microsoft.com/security/blog/2021/01/21/the-dynamic-duo-how-to-build-a-red-and-blue-team-to-strengthen-your-cybersecurity-part-2/). For more information.

This concept can also be applied to our suggested and so-called (more) trustworthy AI lifecycle. The blue team would be responsible for securing the lifecycle itself and the resulting AI systems, which fits quite well with the (performance) metrics, (AI/ML) event traces, security logs and other forensics capabilities we have already highlighted so far.

But the red team part is what we can add to complete our development lifecycle. The red team executes on this mission by finding and exploiting security weaknesses to provide information on the security stance of any AI system at the organization. By simulating real security incidents, red teaming assesses service readiness to the detection of, investigation of, and recovery from a breach and simulates real-world targeted and persistent advisories. See [How Microsoft and Google use AI red teams to “stress test” their systems](https://www.emergingtechbrew.com/stories/2022/06/14/how-microsoft-and-google-use-ai-red-teams-to-stress-test-their-system).

At Microsoft, today, we routinely use tools such as Counterfit (see section Conducting an adversarial attack as an illustration above) as part of our AI red team operations. We have found it helpful to automate techniques in [MITRE’s Adversarial ML Threat Matrix](https://github.com/mitre/advmlthreatmatrix/) and replay them against Microsoft’s own production AI services to proactively scan for AI-specific vulnerabilities. Counterfit is also being piloted in the AI development phase to catch vulnerabilities in AI systems before they hit production. Counterfit has also been integrated into MITRE CALDERA, a framework for simulating adversary behavior in a lab environment, to enable security practitioners to test for vulnerabilities of machine learning models using a tool that they are already familiar with. See [AI security risk assessment using Counterfit](https://www.microsoft.com/security/blog/2021/05/03/ai-security-risk-assessment-using-counterfit/) for more information on how the AI red team and security analysts are working “together” as part of their daily operations.

This is now time to conclude our journey towards a (more) trustworthy AI lifecycle.

# As a conclusion

**We hope you have enjoyed the tour and you are not too tired with all those considerations and technical details.**

In this guide, we analyzed how the classical AI development life cycle are organized and built. We investigate the main stages of development that compose it as well as the corresponding persona who make it live within an organization.

Then we highlighted a number of threats and failure modes that can be found throughout the development lifecycle and/or that result from it. This consequently explains in part the reluctance of some actors to adopt this (new) technology that has not yet been sufficiently tested and/or isn’t appropriately mastered with the relevant guardrails in place, would it be in terms of reliability, safety, security, and/or privacy controls.

To complete this picture, we shared and iterated on a PoC that shows how easy it can be to compromise an AI system based on a ML model while the task in question being performed could be considered critical from a safety perspective: autonomous driving. This (basic) illustration justifies by itself the necessity of a (more) trustworthy AI lifecycle to build trust around ML based projects.

With that, we finally discussed a number of considerations that pertains to the definition of this (more) trustworthy AI lifecycle. We introduced on purpose a series of valuable activities to be implemented throughout the classical AI development lifecycle to strengthen it, highlighted in this context a number of existing practices and tools that could help to sustain these activities, and finally lead to this (more) trustworthy AI lifecycle.

One of the recommendations from Gartner’s [Top 5 Priorities for Managing AI Risk Within Gartner’s MOST Framework](https://blogs.gartner.com/avivah-litan/2021/01/21/top-5-priorities-for-managing-ai-risk-within-gartners-most-framework/) published back in January 2021 is that organizations “Adopt specific AI security measures against adversarial attacks to ensure resistance and resilience,” noting that “By 2024, organizations that implement dedicated AI risk management controls will successfully avoid negative AI outcomes twice as often as those that do not.”

In the meantime, The NIST published its [Artificial Intelligence Risk Management Framework (AI RMF 1.0)](https://www.nist.gov/itl/ai-risk-management-framework) in January 2023.

# To go beyond

In this guide, we went through several steps before defining the incremental contours at a (more) trustworthy development lifecycle. In this way we tried to follow our North Star, and thus to fulfill the requirements deriving from two of the Microsoft's six principles of Responsible AI.

To complete this initiated work, which **is NOT a destination but a journey** like security, other constraints and areas should be without any doubt embraced in order to build a more generic AI lifecycle. If all of the above constitute a subject of considerations/investments, and if you would like to learn more, here are some additional links beyond the ones already shared that might be of interest to you:

* [Adversarial Machine Learning - Industry Perspectives](https://arxiv.org/pdf/2002.05646.pdf).
* [ENISA - Artificial Intelligence Cybersecurity Challenge](https://www.enisa.europa.eu/publications/artificial-intelligence-cybersecurity-challenges).
* [ENISA - Securing Machine Learning Algorithms](https://www.enisa.europa.eu/publications/securing-machine-learning-algorithms).
* [ENISA - Multilayer Framework for Good Cybersecurity Practices for AI](https://www.enisa.europa.eu/publications/multilayer-framework-for-good-cybersecurity-practices-for-ai).

To continue learning about the passionate subject of Responsible AI, you can follow the other tutorials and walkthroughs available in this workshop.

Une image contenant texte, motif, point

Description générée automatiquementYou can also scan this code or visit <https://aka.ms/RAIresources> where you can access the entirety of already available tools, guidelines, and other additional resources that will help you create your next AI solution in a (more) responsible manner.

Une image contenant texte, capture d’écran, Site web, Page web

Description générée automatiquement

Une image contenant bleu, brouillard, capture d’écran, bleu vert

Description générée automatiquement