Trust in AI, trust in public processes

S. Daniil

S. Vethman

M. Molhoek

OUR OBJECTIVE

To enable trustworthy use of AI by public authorities and support effective oversight.

The use of AI tooling can facilitate the design and application of efficient and socially aware policies by governmental bodies, but involves the risk of disregarding *trustworthiness* as a priority, thus risking losing the trust of society towards the government. Guidelines for Trustworthy AI have been proposed, but require practical understanding from developers of AI systems in order to be adequately implemented.

Menu

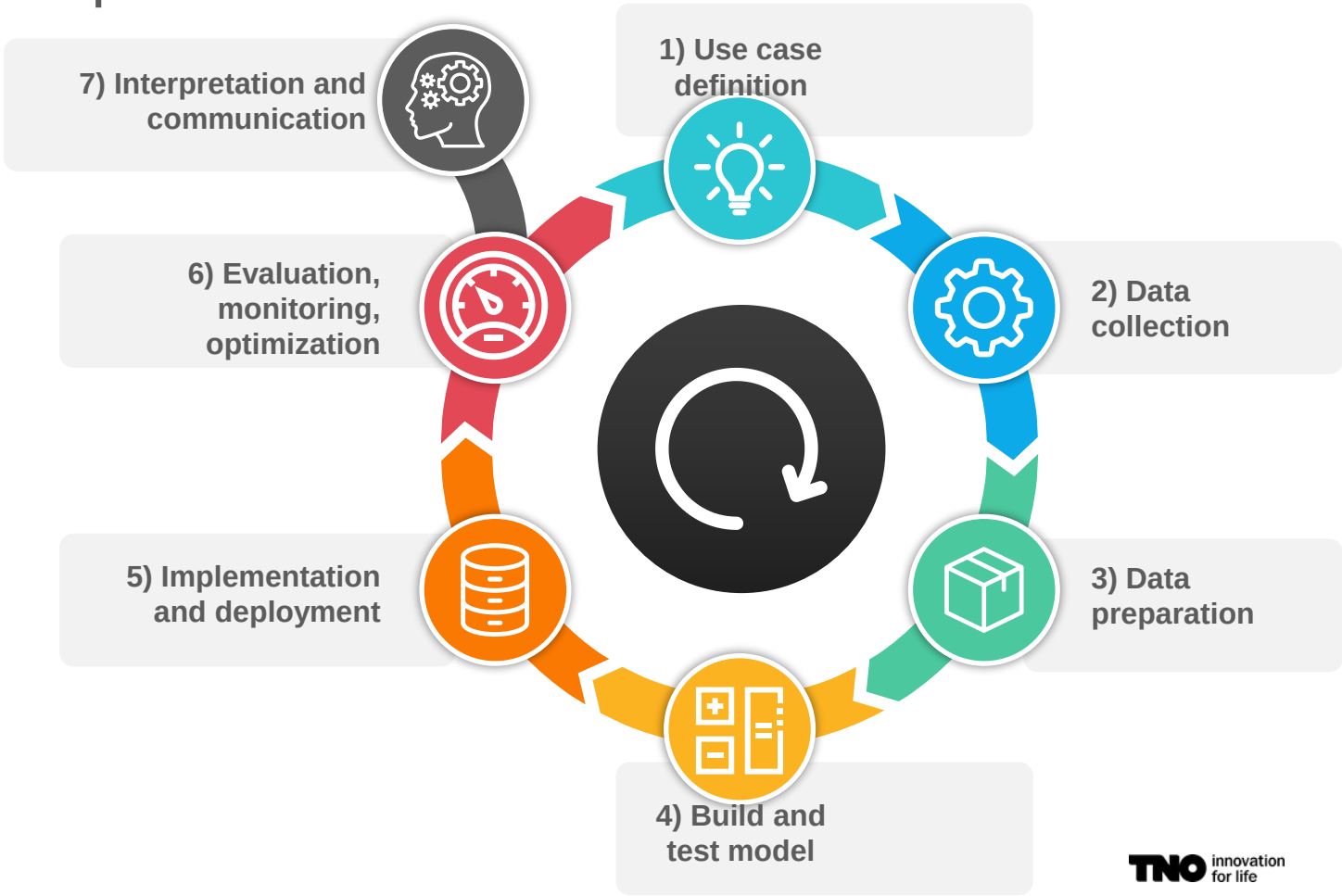
AI life cycle and trustworthy AI principles	• AI lifecycle • Overview of important Trustworthy AI principles • Selection of principle Reproducibility
Reproducibility	• Importance of Reproducibility • Views on Reproducibility • Required Randomness

Handbook	• Reproducibility framework mapped to the AI lifecycle
	• Best practices for implementation
	  • Step 3: Data pre-processing
	  • Step 4: Build and test model

Other causes	• Inherent randomness in hardware
Next steps	• Vision of the future of AI Oversight Lab handbook

AI life cycle components

The AI development cycle passes through 7 stages. From project scoping (1-2), design and build phase (3-4) up to continuous improvement in production (5-6). The 7th stage is specifically added as interpretation and communication by operators is society-dependent and can change over time.

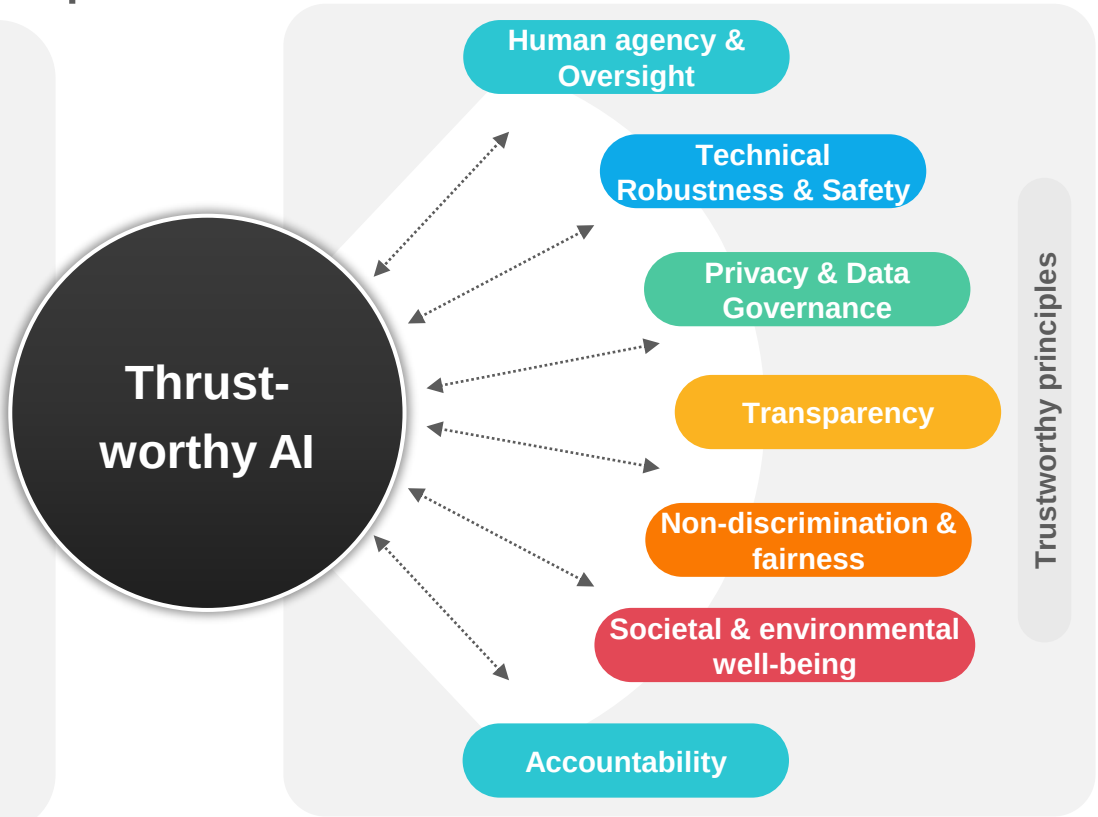


Trustworthy AI principles:

The European ethics guidelines on trustworthy AI state that it should be:

- Lawful – respecting all applicable laws and regulations
- Ethical - respecting ethical principles and values
- Robust – both from a technical perspective while taking into account its social environment

These guidelines translate into 7 key requirements that AI systems should meet , the principles of trustworthy AI.

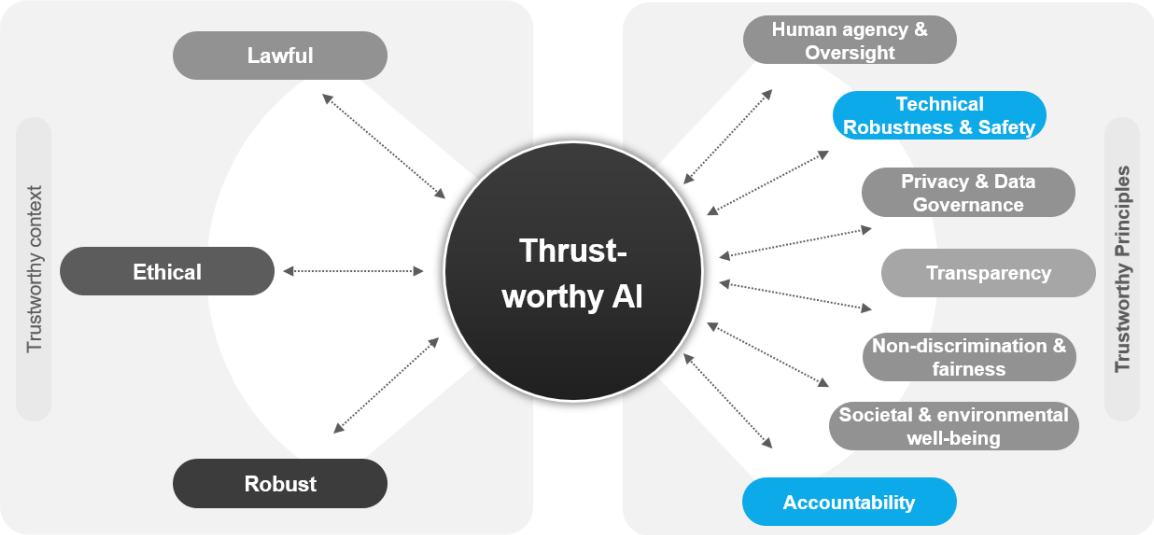


Begin with … Reproducibility

Reproducibility serves as the foundation for the Trustworthy principles *Robustness* and *Accountability*. Its relevance is often underestimated and has not been thoroughly investigated in the context of AI, despite the crisis that it seems to face.

Reproducibility of AI systems highly influences their performance and applicability in social context, since:

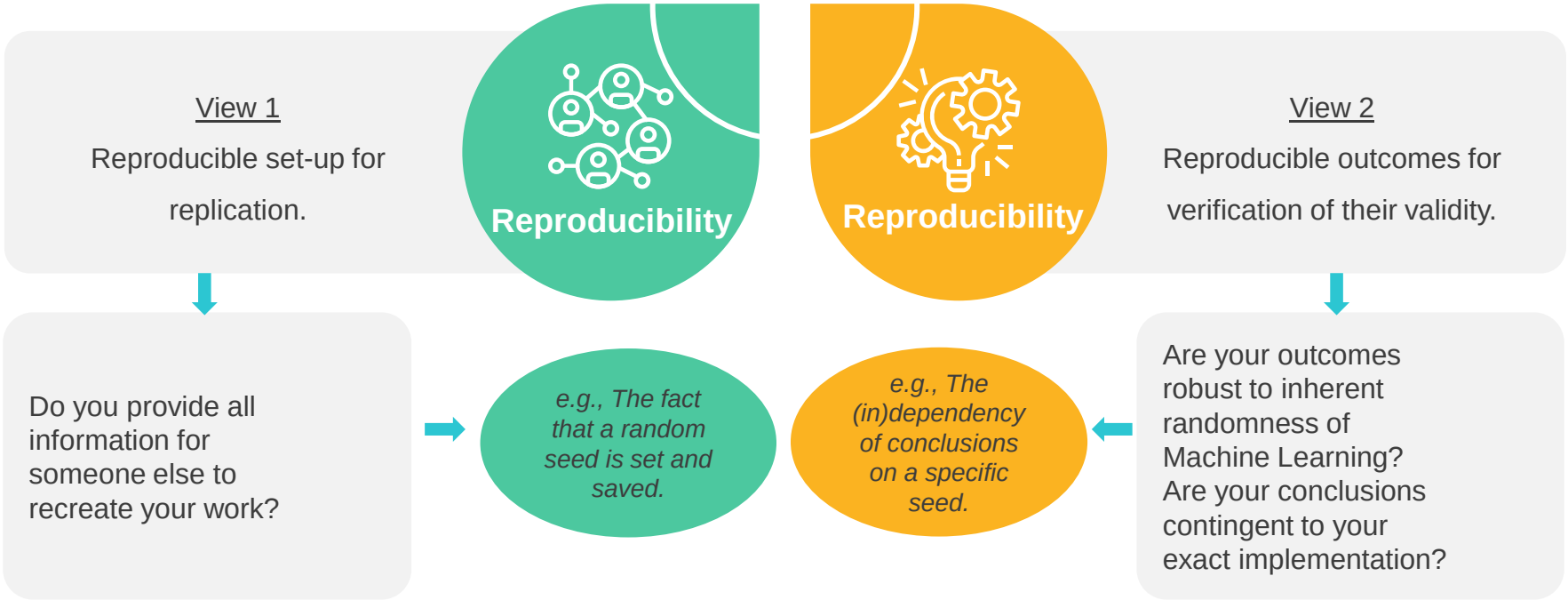
1. It is a prerequisite for technical robustness, and
2. It is crucial in rendering systems verifiable and thus fit for criticism and improvement.



TNO innovation for life

The importance of reproducibility

What is reproducibility?



The reproducible scale

There are various strictness levels of reproducibility for another person to get to the same results. All variations can occur in between. What is the reproducibility if different hardware is used? What is the effect of coding in R versus in Python? What is the result of different library versions? How do random forest and gradient boost affect reproducibility? Depending on the need of the user, the strictness of reproducibility can be chosen.

Reproducibility
With **the same** hardware,
the **original** code,
the **same** language,
the **same** AI methods,
the result is reproducible for
another person.

Reproducibility
With **various** hardware,
no original code available,
different languages,
comparable AI methods,
the result is reproducible for
another person

Reproducibility: Complementing views

Easy to reproduce

Hard to reproduce

View 1: Replicability

REQUIRES VALIDATION

The use of the algorithm is **accountable**, as the settings and outcomes can be repeated such that previous execution can be corroborated.

However, the outcome **may be dependent on the exact implementation** of the code, **more validation is needed** before one considers the outcomes of the algorithm in process affecting society.

IDEAL

The algorithm is **accountable** as information for reproducibility for all stages in the AI life cycles is appropriately stored.

The **outcomes are verifiable or are verified** with different models and settings such that they can be **interpreted with appropriate level of confidence**.

NO GO

Outcomes of the algorithm cannot be reproduced by any (3rd) party. Randomness in the algorithm or contingencies of the algorithm are ignored in the interpretation of the outcomes. Hence, application of the algorithm is in many real life processes ill-advised as **the outcomes do not have a robust interpretation nor are they accountable.**

REQUIRES TRUST

Outcomes of the algorithm cannot be reproduced by 3rd parties due to omission of information on parameters and packages or privacy aspects of even the characteristics of the data.

The outcomes are presented to be verified by multiple approaches and classifiers, however **due to the inability to exactly replicate the outcomes, the developer need be trusted**.

View 2: Robustness/Stability/Reliable

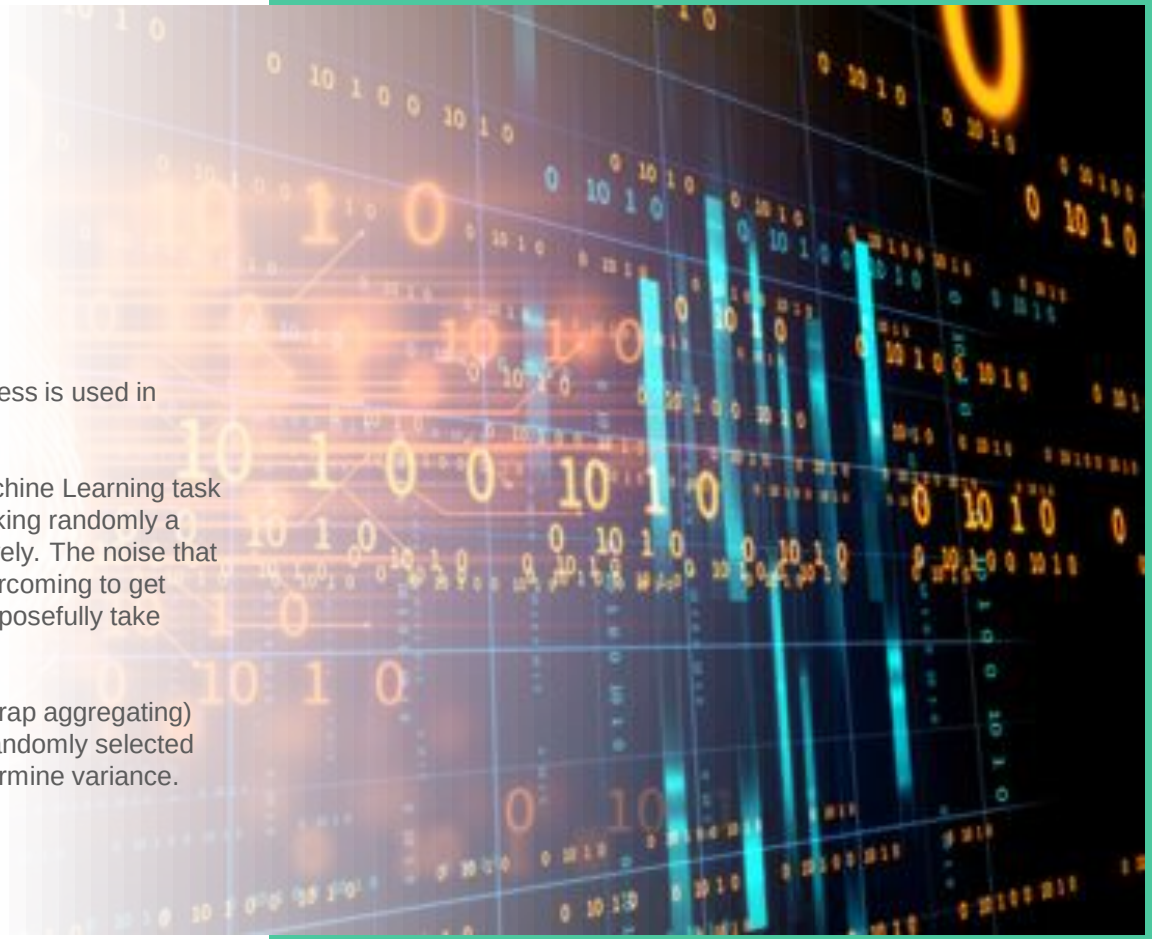
Contingent **Verified/Verifiable**

Importance of randomness

Due to memory and time constraints randomness is used in machine learning tasks.

To calculate the gradient in optimizing the Machine Learning task often Stochastic Gradient Descent is used picking randomly a part of the data to update its parameter iteratively. The noise that is added during these updates can help in overcoming to get stuck in a local minimum. The model may purposefully take random steps to seek a better state.

Another example is the use of bagging (bootstrap aggregating) which trains multiple models on overlapping randomly selected subsets of data to increase accuracy and determine variance.

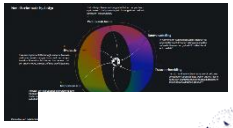


TNO's focus in the landscape

Existing frameworks/tools

EU ethics guidelines and frameworks WP1

TILT handbook



Trustworthy AI Questionnaire

V29 questionnaire

Trustworthy AI guidelines and handbooks

High level

In-depth

Technical

Juridical

Organizational

Technical principles of trustworthy AI

Juridical implications for implementation

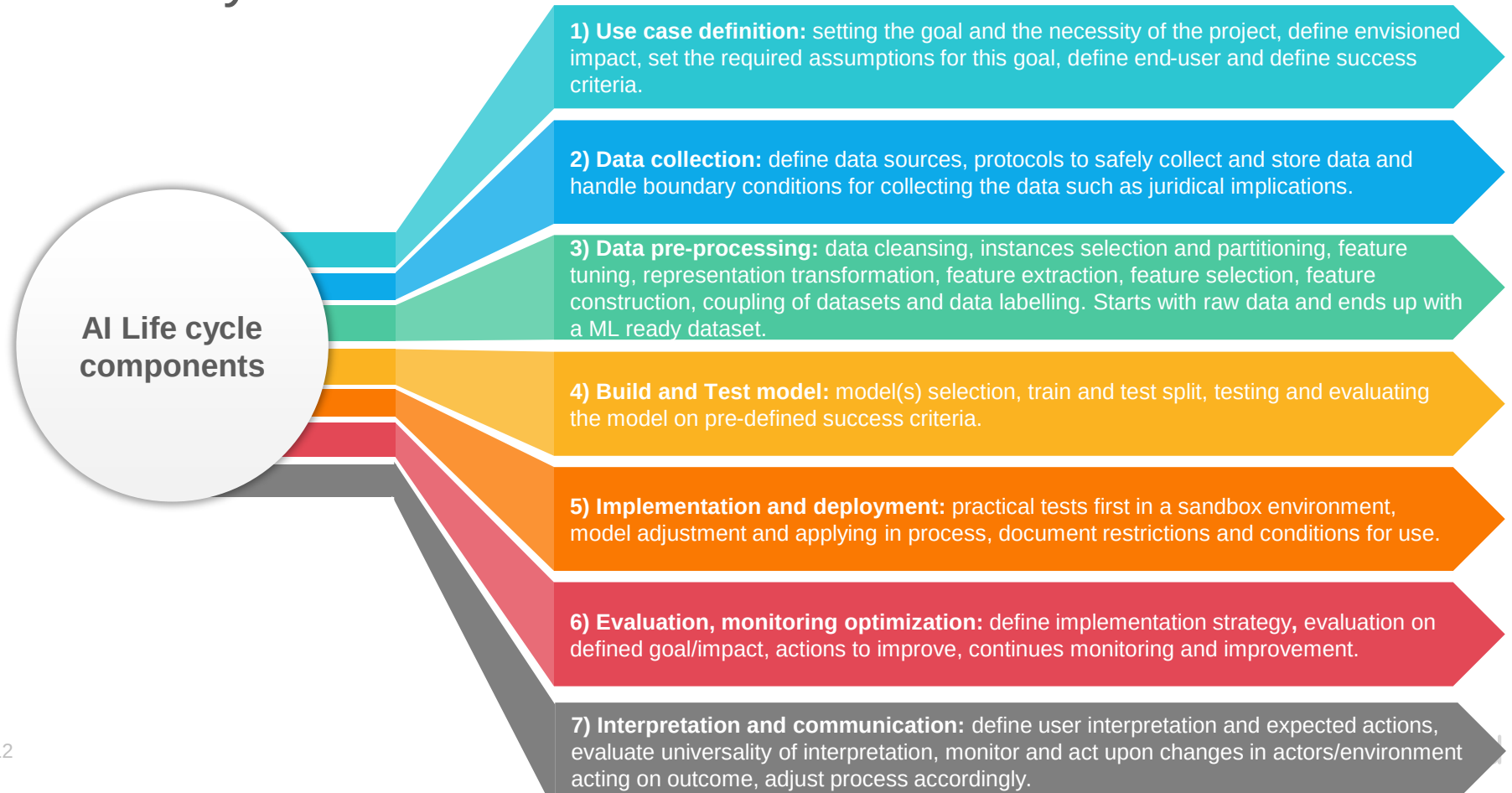
Organizational implementations in processes

Break-down high level concepts into technical implementations of trustworthy AI including specific coding guidelines for developers.

Focus of TNO's handbook

TNO innovation for life

AI life cycle



Framework Reproducibility

Level	Life cycle components						
	Use case	**Data collection**	**Data preparation**	**Build and test model**	**Implementation and deployment**	**Evaluation, monitoring optimization**	**Interpretation and communication**
Definition	• Problem statement • Define goal and scope • Define intended use	• Define data sources • Procedure and boundaries of data collection	• From raw data to ML input • Data cleaning and preprocessing	• Model selection • Test procedures including train test split.	• Sandbox first • Practical application • Restrictions and conditions for use	• Evaluation with defined goal • Actions to improve • Continuous monitoring	• User interpretation and expected actions.
Problems sources	• What level and type of reproducibility is necessary?	• Data size • Computing power requirements	• Dynamic data • Feature engineering • Missing data	• Random sampling • Parameter choices • Inherent randomness • Libraries	• Documentation by user	• Metrics	• Visualization • Non-universal standardization
Solutions	• Use our handbook to select your needs for your use case.	• Learning Curve • Benchmark set	• Timestamps • Recording of preliminary data sets.	• Documentation of seed. • Version control. • Automated testing • Environment and docker files.	• Automated documentation.	• Checklist • Dynamic dashboard.	• Benchmark • Examples

Build and test model:

	Problems / Sources	Solutions

Train-test split	• **A random allocation of each data point or subject into two separate sets** is executed to split the data into a train or test set. This randomness may influence the results of the algorithm.
	  • In python, this can e.g. be performed by a function of the package scikit learn called: train_test_split.
		• **Setting and saving a seed** to the train test split ensures that creation of the train and test is repeatable.
		  • In python, the package scikit learn often allows the random state to be given to the function.
		• **Cross validation** is a more extensive version on a train-test split with multiple splits per execution such that the influence of the randomness becomes smaller. This increases the reproducibility w.r.t. View 2 of the results as it decreases the reliance of a specific seed.
		• **Saving (and sharing) the index of the train and test splits** (folds) increase the reproducibility of results w.r.t. a train-test split further. In this way, you can exactly recreate the splits irrespective of another random number generator and if applicable others may test the findings on the exact same data splits.

Parameter choice	• **Unknown parameter choices**, i.e. missing records of the settings used in an algorithmic application, are a risk for reproducibility.
	• **Default parameters in packages** are an even larger risk for the reproducibility in terms of View 2. These are often unacknowledged and not recorded, while default parameters differ between (versions of) packages and programming languages.
	  • In python, the default number of splits in function StratifiedShuffleSplit of package scikit learn has changed over versions from 3 to 5. E.g. the solver for RandomForest in python has a default depth of 100 branches while the package for R has a default of 200 branches.
		• **Configuration** files or scripts aid in the recording of the parameter settings applied in testing the model. This is in contrast to parameters being set somewhere in the collection of scripts that the eventual algorithm will consist of.
		  • It is common practice to use .yml files for saving parameter choices and data paths for referencing all relevant directories.
		• **Version control** is in turn able to save the configuration files and as such the chosen parameters of each test. For this solution there is a distinction for the developer (elaborated here) and the user (elaborated in the deployment phase).

Build and test model

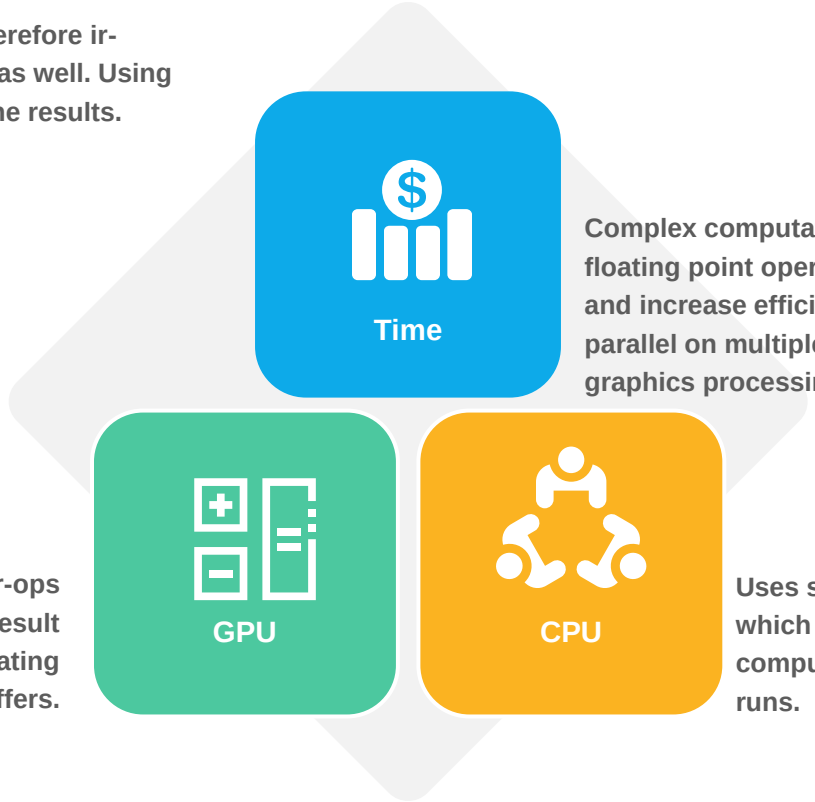
	Problems / Sources	Solutions

Inherent algorithmic randomness (non-deterministic)	• There are **many reasons for algorithms to be non-deterministic**, see slide 19 for elaboration on inherent randomness in algorithms. Some specific origins of randomness in machine learning algorithms are listed here:
	  • Many models optimize **weights** such that they capture the targeted relation in the data. To acquire a starting point of the optimization, weights are often randomly initialized.
	  • Multiple features of **Neural Networks** use random sampling, one of them is the random selection in the application of dropout layers.
	  • For **stochastic optimization** to find (local) optima, random sampling is used. An example for which this holds is stochastic gradient descent.
	  • Randomness in **random forest** models. Each tree in the random forest selects random features on which the model is trained.
		• **Test automation:** To gain the benefits from algorithmic randomness and limit the drawbacks, automatic testing of the algorithm for e.g. a range of seeds may be performed.
		  • Do the results for different seeds fall into an acceptable range?
		  • Are the conclusions the same for all most seeds?
		• **Continuous integration:** A step further in test automation is the use of continuous integration of testing during the building and testing of the model. Before any version of the model is approved or even any merge from branch to main is requested, automated testing can be integrated. As such, version updates and merge request are accompanied with a testing report.
		• **Alternative solver, same results?** In many applications, there are many options to solve the objective function, i.e. to optimize the classifier. To increase the reproducibility of outcomes w.r.t. View 2, it is advised to see the outcomes for multiple solvers with similar functionalities s.t. the dependency on one component is limited.
		• **Statistical significance tests** between results and benchmark take into account random noise in data and if applied correctly contribute to robustness of a model's results.

Library	• The use of **libraries** simplifies the use of AI and machine learning but often does not guarantee an 100% reproducibility. Libraries and the underlying functions change each version, including their dependencies to other libraries.	• **Environment,** in python, it is common practice to create a designated virtual environment per project or per version of the model. These virtual environment constitute a.o. the python version and the versions of all installed packages. The creation and preservation of python environments enhance the reproducibility by ensuring execution of the code on similar software settings. The operation system is however not included in the virtual environment. • **Docker file,** the operation system (Linux, Windows 2000) on which is directly related to whether certain algorithms can be executed. To increase reproducibility, a docker file, "a virtual operating system often containing a virtual environment" can be created and shared.

Inherent randomness in Hardware

There is inherent randomness and therefore ir-reproducibility in hardware selection as well. Using different hardware types may affect the results.

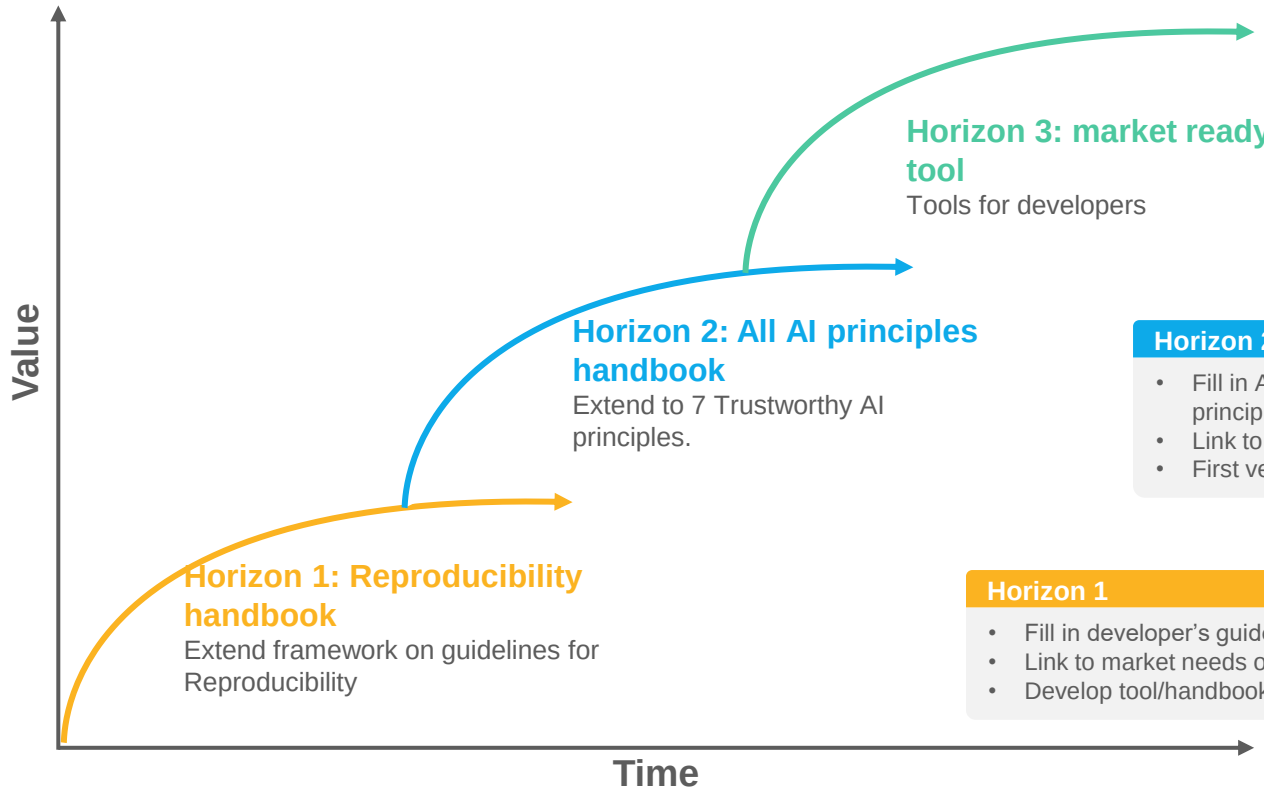


Complex computations require Giga or even Tetra floating point operations. To decrease the timeframe and increase efficiency the processes are run in parallel on multiple central processing units and/or graphics processing units.

Uses Intra-Ops and Inter-ops parallelism on CPU, which can result in different results. By using floating points in solving the result differs.

Uses stream multiprocessing units which performs asynchronous computation which results in different runs.

Next steps towards trustworthy AI



Horizon 3

- Market ready tool for developers
- Practical use cases
- Cross links between AI life cycle components

Horizon 2

- Fill in AI lifecycle components for the other Trustworthy AI principles (explainability, transparency, privacy, bias)
- Link to market needs
- First version of tool

Horizon 1

- Fill in developer's guidelines for every AI lifecycle component
- Link to market needs on reproducibility
- Develop tool/handbook set-up

Trust in AI, trust in public processes

S. Daniil

S. Vethman

M. Molhoek

OUR OBJECTIVE

To enable trustworthy use of AI by public authorities and support effective oversight.

The use of AI tooling can facilitate the design and application of efficient and socially aware policies by governmental bodies, but involves the risk of disregarding *trustworthiness* as a priority, thus risking losing the trust of society towards the government. Guidelines for Trustworthy AI have been proposed, but require practical understanding from developers of AI systems in order to be adequately implemented.

References

[1] P. Hemant, “Reproducible machine learning,” Apr 2020.
[2] T. Allard, “10 top tips for reproducible machine learning,” Apr 2020.
[3] W. D. Heaven, “Ai is wrestling with a replication crisis,” Nov 2020
[4] “Reproducibility challenge @ neurips 2019.
[5] P. Sugimura and F. Hartl, “Building a reproducible machine learning pipeline,” 2018.
[6] P. Warden, “The machine learning reproducibility crisis,” Mar 2018.
[7] jennifervilla,“Reproducibilityinml:whyitmattersandhowtoachieveit,”May 2018.
[8] J. Fjeld, N. Achten, H. Hilligoss, A. Nagy, and M. Srikumar, “Principled artificial intelligence: Mapping consensus in ethical and rights-based approaches top principles for ai,” SSRN Electronic Journal, 2020.
[9] B. Shneiderman, “Bridging the gap between ethics and practice,” ACM Trans-actions on Interactive Intelligent Systems, vol. 10, no. 4, p. 1–31, 2020.
[10] J. Brownlee, “Embrace randomness in machine learning,” Aug 2019
[11] S. Mall, “Realizing reproducible machine learning - with tensorflow,” Dec 2019
[12] C. Shao, “Properly setting the random seed in ml experiments. not as simple as you might imagine,” April 2019.
[13] C. Ayuya, “Reproducibility to improve machine learning,” January 2021.
[14] EU, “Ethics guidelines for trustworthy ai,” April 2019.
[15] N. Koleva, “Ai and data science lifecycle: Key steps and considerations,” May2020.
[16] Bart van der Sloot, Esther Keymolen, Merel Noorman, Yvette Wagensveld(Tilburg University), College voor de rechten van de Mens, Hilde Weerts (TU E) Bram Visser (VU Brussel) “Handreiking non-discriminatie by design,” 2021
[17] Interviews with NFI, VORtech, UvA, Leiden University