How to use the Mapping Toolkit

In this document we will show you how to utilise the mapping toolkit which is meant to aid you in choosing a sensible testing technique for testing your machine learning models.

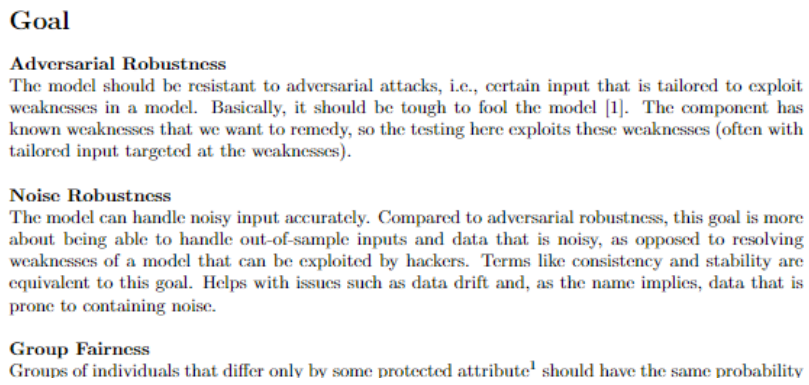
Step 0: Learning what you have available

Before getting started with using the mapping you are recommended to take a look at the **Mapping Description (README)** document. This document will give you some insight on the mapping itself as well as the extra documents that are provided to you to aid you in properly utilising the mapping.

Step 1: Before you get started

Step 1.1: Learning the definitions

The mapping utilises tags that you as a developer are able to filter on so that the mapping can recommend a good testing technique for your project. To be able to properly select the tags most relevant to your project, it is important that you understand what each and every tag means. For this reason we provide a document called the **Definitions Document** that contains all the definitions used in the mapping. Many of the terms will most likely be familiar, but since some of them have varying definitions across the literature, like robustness and fairness, we would recommend still skimming this document so that you are sure of the definitions used for this mapping.



Goal

Adversarial Robustness
The model should be resistant to adversarial attacks, i.e., certain input that is tailored to exploit weaknesses in a model. Basically, it should be tough to fool the model [1]. The component has known weaknesses that we want to remedy, so the testing here exploits these weaknesses (often with tailored input targeted at the weaknesses).

Noise Robustness
The model can handle noisy input accurately. Compared to adversarial robustness, this goal is more about being able to handle out-of-sample inputs and data that is noisy, as opposed to resolving weaknesses of a model that can be exploited by hackers. Terms like consistency and stability are equivalent to this goal. Helps with issues such as data drift and, as the name implies, data that is prone to containing noise.

Group Fairness
Groups of individuals that differ only by some protected attribute[1] should have the same probability

Figure 1: A preview of the **Definitions Document**.

Step 1.2: Getting an overview

Before getting started with using the mapping, you can use the **Mapping Overview Trees** to get a quick overview of what the mapping will be able to provide you with. It's formatted so that you can see which goals are applicable to which components under test, and which testing techniques the literature recommends applying to achieve these goals. In the image below you can see an example of this, with the green "component under test" node in this

case being DATA. The goals are the red nodes, which are for example completeness and correctness. The blue nodes are then the testing techniques that have been recommended in the literature for achieving the goals that they are connected to.

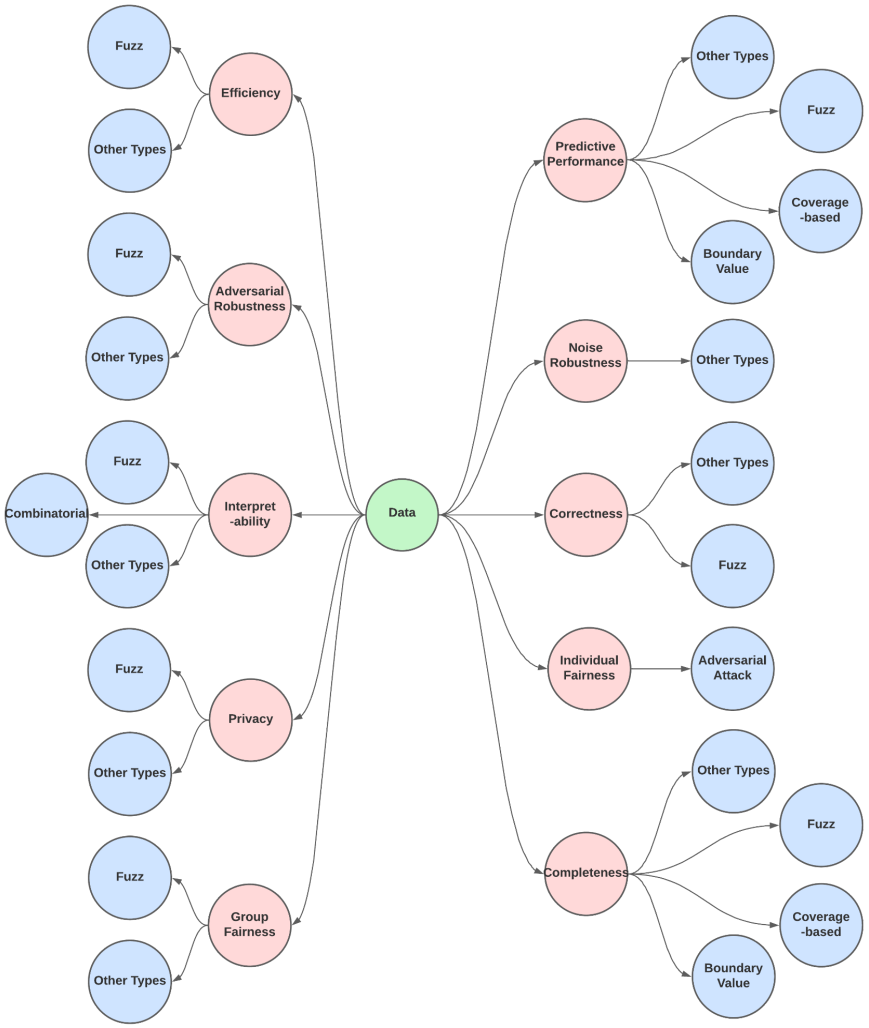


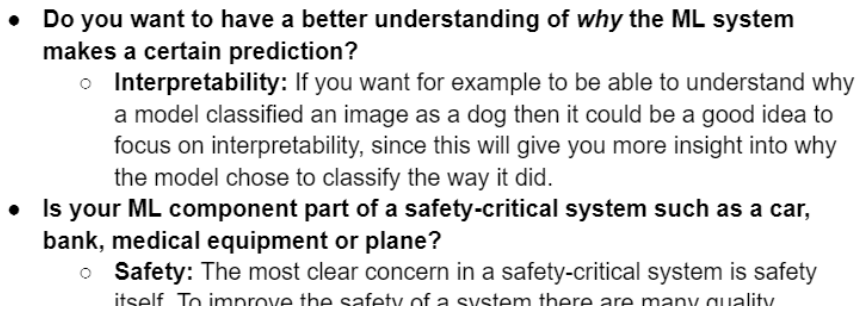
Figure 2: The overview tree diagram for the Data CUT.

Step 1.3: Choosing Goals

You might already have decided what goals you want to test for in your project, which means that you won't be needing to use the **Goal Identification Document**. Do note however, that you must translate your own goals so that they are the same as the ones used as tags in the mapping. For example, if you have robustness as a goal you must determine if it is Noise Robustness or Adversarial Robustness (or both). Once that is done you are ready to use the mapping, although going over the **Goal Identification Document** to find additional goals to test for might also be a good idea.

If you have not decided which goals to test for, then it is heavily recommended that you use the **Goal Identification Document** to give yourself and idea of what could be relevant for you to test for. The goals in this document are already recommended in the same form as

the tags, so there is no need to convert them like you would have to do with your own predefined goals. Once you have identified some goals you want to test for you are ready to use the mapping.



- **Do you want to have a better understanding of *why* the ML system makes a certain prediction?**
  - **Interpretability:** If you want for example to be able to understand why a model classified an image as a dog then it could be a good idea to focus on interpretability, since this will give you more insight into why the model chose to classify the way it did.
- **Is your ML component part of a safety-critical system such as a car, bank, medical equipment or plane?**
  - **Safety:** The most clear concern in a safety-critical system is safety itself. To improve the safety of a system there are many quality

Figure 3: A preview of the **Goal Identification Document**.

Step 1.4: Understanding the Project's Context

The more knowledge you have about your project's context, the more accurate recommendations you will be able to get from the mapping. Do note however that not all combinations of contexts are available in the mapping. This means that it can be better to be more lenient in how context tags are selected, and select more rather than less since this will give more options to choose from. We would recommend including any context tags that you feel could be relevant to you, and only excluding ones that you know will not be of any relevance to you. If you then end up with a lot of suggestions, you can gradually become more strict with the tags. Do note that not selecting any tags for a column is equivalent to selecting all the tags for that column, so having zero tags on a column is more lenient than having one. In all other cases more tags in a single column means being more lenient however (two is more lenient than one, three is more lenient than two, and so on).

Step 2: Using the Mapping

After all the preparation has been done you are ready to get started with using the actual mapping. At this point you should know what your goals are, and have a good idea of how your context can be tagged (lenient at first, strict if many recommendations remain).

Title	Pap...	Goal	CUT	ML Task	Access Levels	Offline/Online	Testing Workflo...	Testing W...	Limitation	Testing Technique	Comment
"Why should i trust you?" Explaining the predictions of any classifier	ACM	Interpretability	Model	Classification	Black-box	Offline	Test Result Analysis		None	Other Type	Not testing per say any prediction fron
A systematic approach for evaluating artificial intelligence models in industrial settings	MDPI	Adversarial Rob...	Model	Classification	White-box	Offline	Test Case Design		None	Adversarial Attack	
Application of metamorphic testing to supervised classifiers	IEEE	Correctness	Learning Program	Classification	Black-box	Offline	Test Result Analysis		None	Metamorphic	
Assessing the Robustness of Conversational Agents using Paraphrases	IEEE	Adversarial Rob...	MLS	Classification Regression	Black-box	Online	Test Case Design	Input Data Prep... Test Result Anal...	NLP	Metamorphic	A variation of meta identifying metam divergent example meaning-preservin predicts the same
ASTRAEA: Grammar-based Fairness Testing	IEEE	Individual Fairn... Group Fairness	Model	Classification	Data-box	Offline	Test Case Design	Test Case Analysis Input Data Prep... Test Result Anal...	NLP	Adversarial Attack Metamorphic Mutation	
Automated directed fairness testing	ACM	Individual Fairn...	Data Model	Classification	Data-box	Offline	Input Data Preparation	Test Case Design	None	Adversarial Attack	
Automatic Fairness Testing of Machine Learning Models	Springer	Individual Fairn...	Model	Classification	Black-box	Offline	Test Case Design		None	Other Type	Uses verification-b approximating a bl and then applies ve violations
Automatic system testing of programs without test oracles	ACM	Noise Robustne...	Model	Classification	Black-box	Offline	Test Case Design	Test Case Analysis Test Result Anal...	None	Metamorphic Mutation	

Figure 4: A preview of the main mapping tool.

The tool has been implemented in Microsoft Lists, but no previous experience with Lists framework is necessary to be able to use the mapping efficiently. Each row in the mapping table represents a single recommended testing technique based on the paper that is named in the very first column. The columns are categories for which tags can be applied, with the categories representing the goals and different parts of the context. To add filter tags to a column you can either press the column directly or press the filter button that can be seen in the image below.

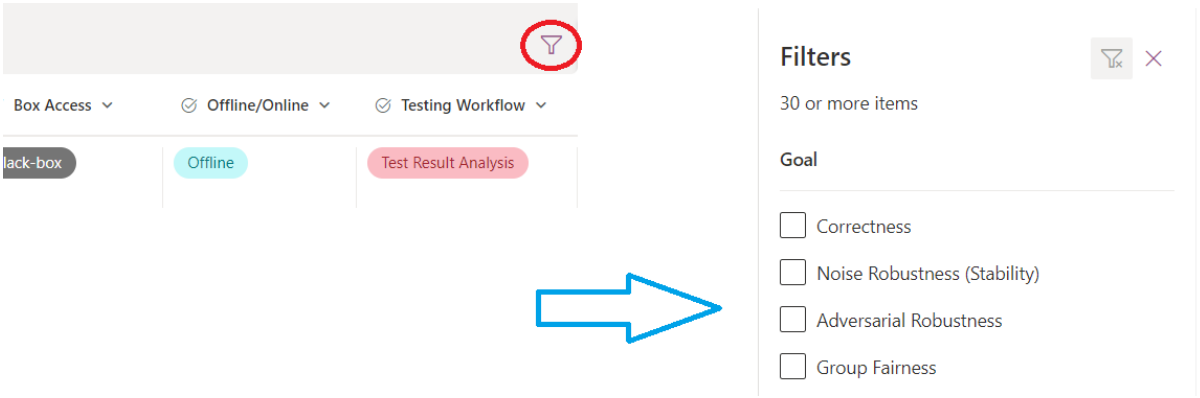


Figure 5: Where to find the filter button.

Once the filters have been applied you should have a list of possible testing recommendations. If there are too many recommendations you can be more strict with your context. If there are too few recommendations you can instead try to be more lenient with the context if possible. Remember that being lenient means having more tags and being strict means having less.

Once you have a reasonable number of suggestions you can take a look at the **Paper Summaries Document** to get a first indication about what each paper proposes. If a particular technique sounds interesting then the next step would be to take a look at the actual paper. If it sounds like a sensible suggestion then your usage of the mapping is complete and you can move on to the next phase of your project which will be planning and implementation of the testing technique.

Astraea: Grammar-based fairness testing: Soremekun et al. [4] proposes another testing approach called ASTAREA that is meant to help developers identify issues with fairness in their NLP models. ASTRAEA uses context-free grammars (CFG) to generate input that include biases that can reveal these issues. The authors state that the idea is that in the generated input there is sentence a and b and the only difference between these is a protected attribute, such as gender or occupation. If the output of a model differs for the input of sentence a or b, then that would constitute a fairness violation where a protected attribute affects the models prediction. ASTRAEA is evaluated against 18 models, and the results show that it on average improves the fairness of those models by 76% according to the authors. One of the limitations of ASTRAEA is that it is only able to detect fairness violations that are in the grammar. This means that the developer needs to have an idea of what fairness violations might occur.

Automated directed fairness testing: Udeshi et al. [5] implemented the automated tool Aequitas that can generate test inputs to check the predictive fairness of a model, and consecutively retrain the model, with the gathered inputs, to improve its fairness. To find an initial set of discriminatory

Figure 6: A preview of the **Paper Summaries Document**.