## A guide to identify your testing goal

Below is a list of questions you can ask yourself to get a better idea of what testing goal is the most important to your ML system. In that way you can more easily identify what testing techniques to use.

- **Does the ML system use data on customers as an input?**
  - **Individual, Group, Counter-factual Fairness:** If the system deals with customer data as an input, then it is important to make sure that the system does not discriminate against customers based on protected attributes such as gender or ethnicity.
  - **Privacy:** If the system deals with customer data it is important to think about privacy concerns since it can otherwise lead to issues with for example EU laws like GDPR.
- **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 attributes that must be looked at, such as robustness, correctness, and completeness.
  - **Adversarial and Noise Robustness:** For the safety-critical system it is important that the robustness is high, since otherwise there might be inputs (either adversarial or just random) that make the system behave weirdly and therefore in a worst case scenario put lives at risk.
  - **Correctness:** It is extremely important that the system behaves according to the specification when the system is safety-critical. In the case of a ML model we might have specified a minimum accuracy for a certain task, and if this is not achieved that can lead to big issues.
  - **Completeness:** Similarly to correctness it is important that all of the specification is covered by the safety-critical system. If the system lacks functionality, or the ability to predict a certain scenario, even though it is meant to, then that can lead to big safety issues.

- **Is the ML system being used directly by the customers?**
    - **Security:** If the system can be used by customers directly it is important to make sure that the access levels are set up properly, so a customer can’t change more than they should. It’s always good to follow a principle of only giving a user access to exactly what they need and nothing else.
- **Do you have existing knowledge about some of the features you are trying to model?**
    - **Monotonicity:** If you have knowledge about some of the relationships that the features might have, then it can be useful to test the ML model's monotonicity so that it is properly reflecting these relationships in its predictions.
- **Will you apply your system in a wide array of scenarios?**
    - **Completeness:** If your requirements specification lists many different application scenarios, then you want to verify that as many of them as possible are covered. For example, the requirements of a self-driving car could state that the car should be able to drive in many different weather conditions.
    - **Noise Robustness:** If we know that the model will have to deal with a lot of varied input that might be quite different from the training data, then it’s important to make sure that the model is stable and can handle these new inputs. It is not enough that the model has high accuracy on the training/test set, we also need to make sure that it can be trusted for out-of-sample data.
- **Is the model or MLS likely to be targeted by malicious actors? I.e. people want to take advantage of the model for some gain**
    - **Security:** If the system is likely to be targeted for attacks it’s important to try to not leave any vulnerabilities that do not have to exist. Having proper access levels is one example that is essential to maintaining a secure system, so that the malicious actors can not take advantage of information they should not have access to.
    - **Adversarial Robustness:** The malicious actors should not be able to attack the model or MLS with targeted input, since this can make it easy to take advantage of. Therefore it’s important to make sure that the model's adversarial robustness is high so that it is able to deal with input that was meant to target its weaknesses.
- **Does your model require vast amounts of data and/or be retrained often?**
    - **Efficiency:** You could then decrease the time or amount of data needed to train the model to maximise its predictive performance. The learning program could also be optimised to reduce the training time.
- **Does your enterprise/institution have a very strict limit on how much resources can be spent on testing the models?**

    - **Efficiency:** Making the test and training data examples more efficient is an excellent way of making sure that less time and other resources have to be expended on the testing, and is the easiest way of managing to make the best use out of a limited resource budget.
    - **Interpretability:** Having an interpretable model is another way of trying to mitigate the amount of tech debt that can be acquired while creating models. An interpretable model is easier to test and debug since we have more insight into how it works and why it predicts in the manner that it does.
- **Does the data change a lot over time? (i.e. do you have data drift?)**
    - **Completeness:** By anticipating a wide range of inputs that the model should be able to handle and by including this in the training data, the model might be able to better handle the changing aspects of the data over time.
    - **Noise Robustness:** If the data drifts a lot over time then it is especially important to ensure that the model has noise robustness, since it needs to be able to deal with the data drift that happens between training and retraining.