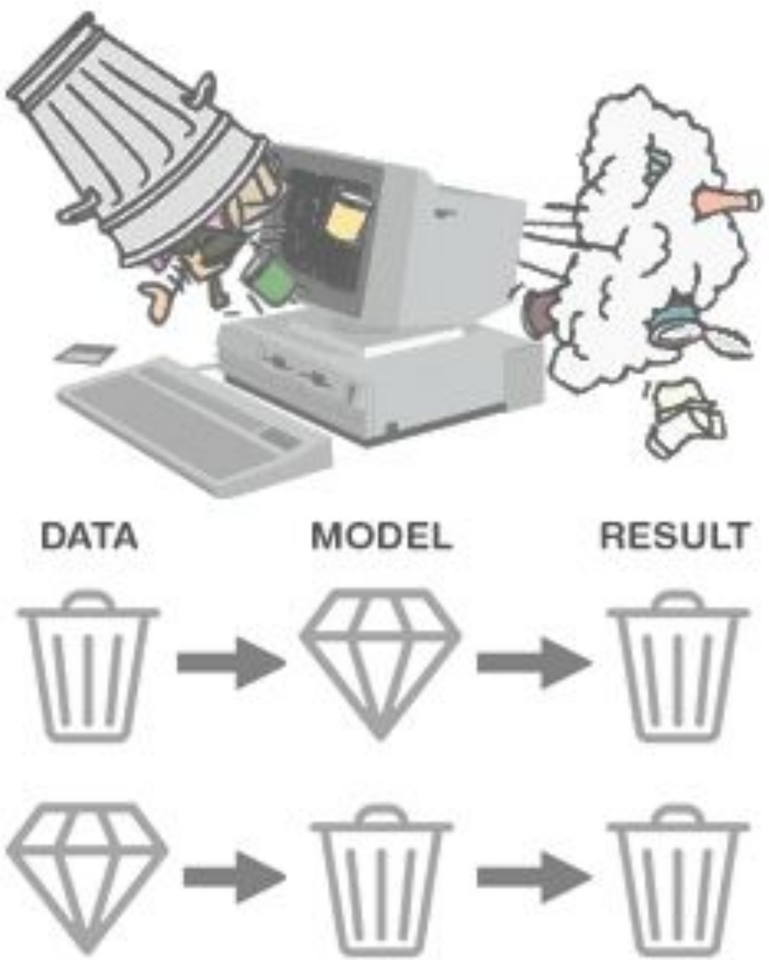
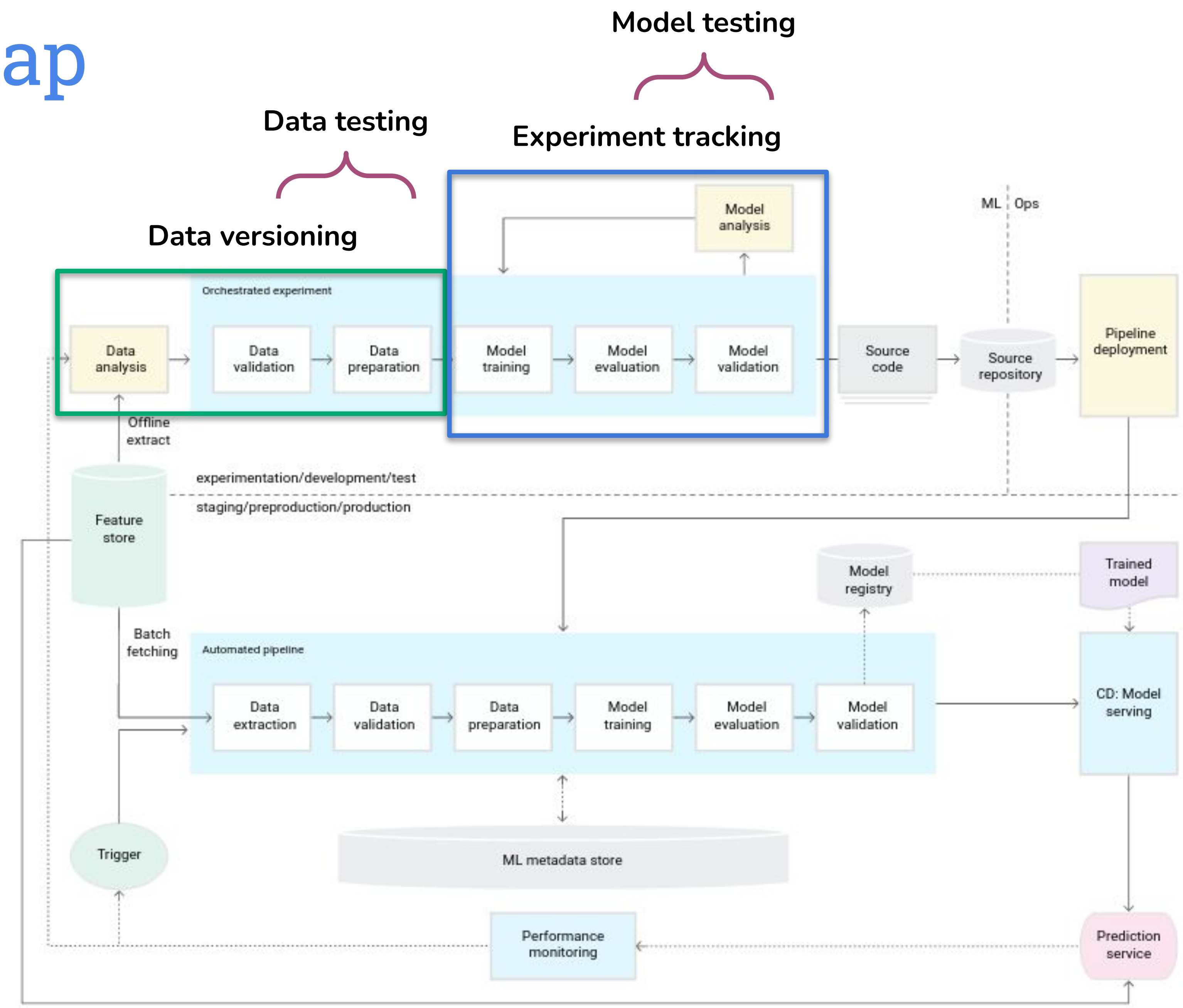
4. Testing

Does the code behave as intended?

Extension of GIGO to ML systems



Roadmap



Model testing
Data testing
Experiment tracking
Data versioning
Model analysis
ML
Ops
Orchestrated experiment
Data analysis
Data validation
Data preparation
Model training
Model evaluation
Model validation
Source code
Source repository
Pipeline deployment
Offline extract
experimentation/development/test
staging/preproduction/production
Feature store
Model registry
Trained model
Batch fetching
Automated pipeline
Data extraction
Data validation
Data preparation
Model training
Model evaluation
Model validation
CD: Model serving
Trigger
ML metadata store
Performance monitoring
Prediction service

Tests everywhere...

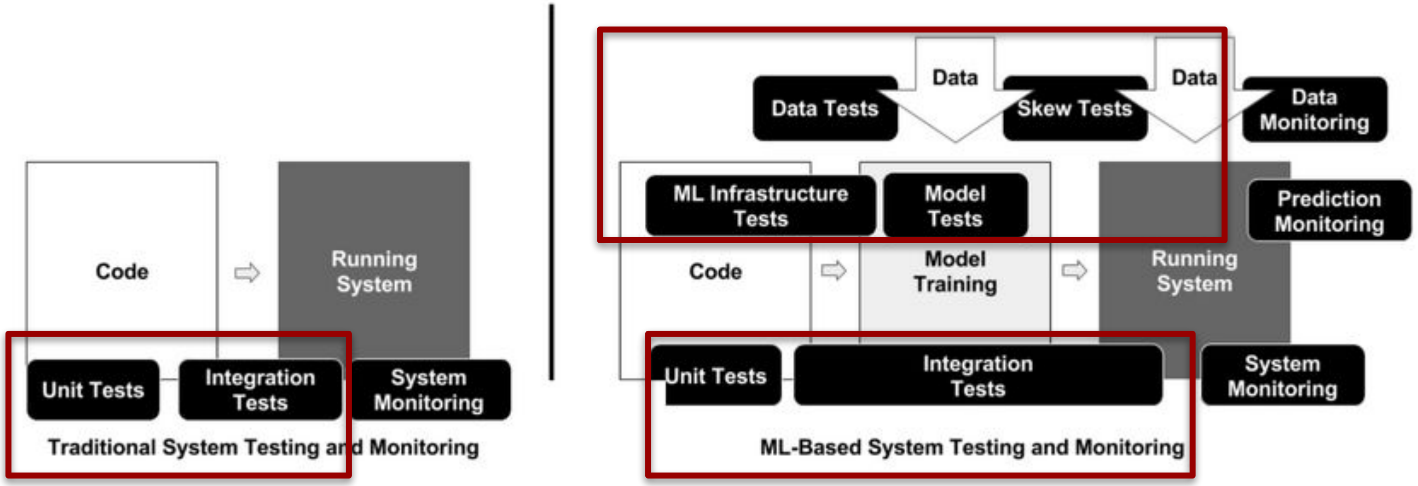


Figure 1. **ML Systems Require Extensive Testing and Monitoring.** The key consideration is that unlike a manually coded system (left), ML-based system behavior is not easily specified in advance. This behavior depends on dynamic qualities of the data, and on various model configuration choices.

Figure source: "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction" by E.Breck et al. 2017

… but every test has its time

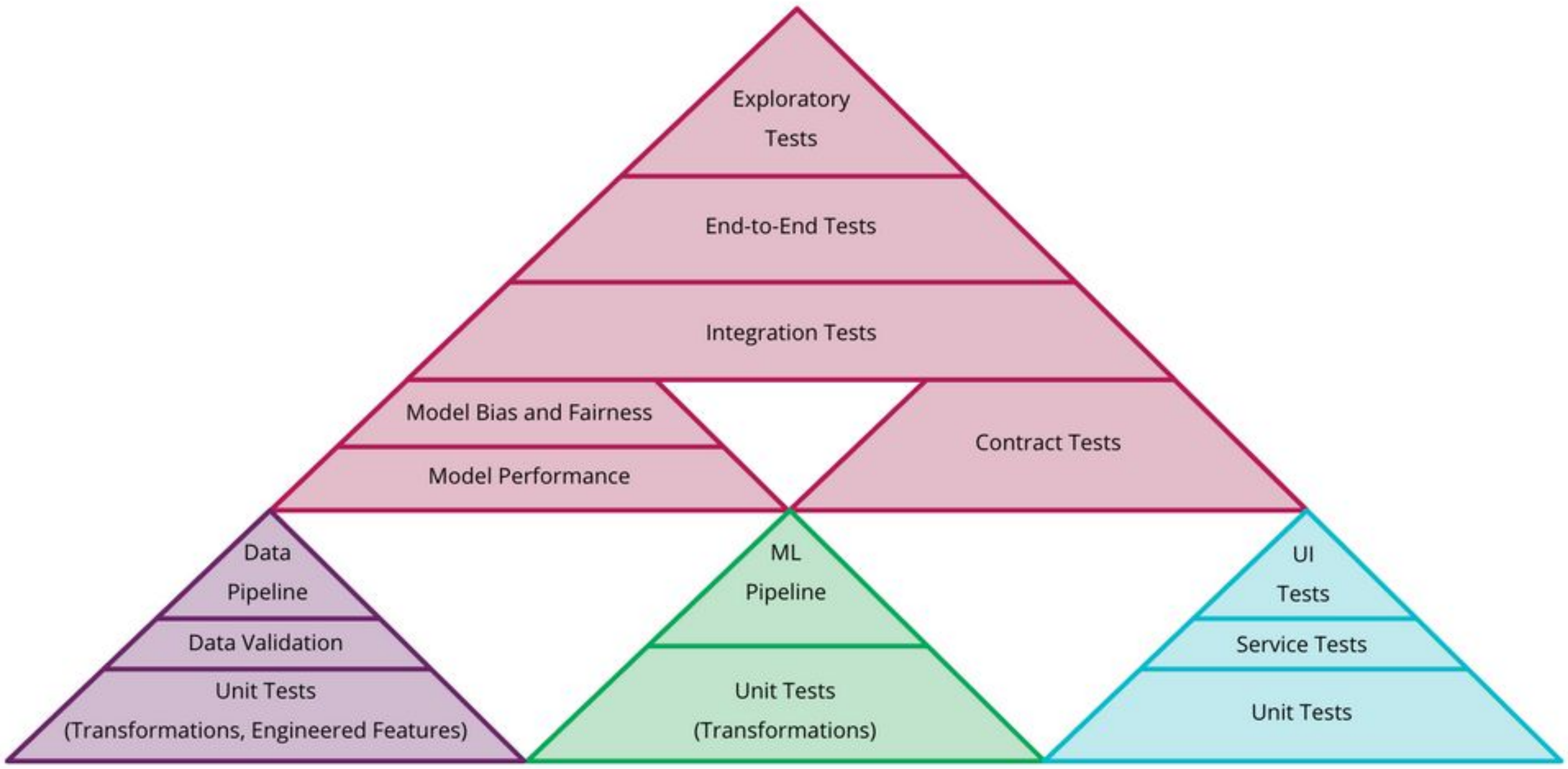


Figure 6: an example of how to combine different test pyramids for data, model, and code in CD4ML

https://martinfowler.com/articles/cd4ml.html

ML pipeline testing

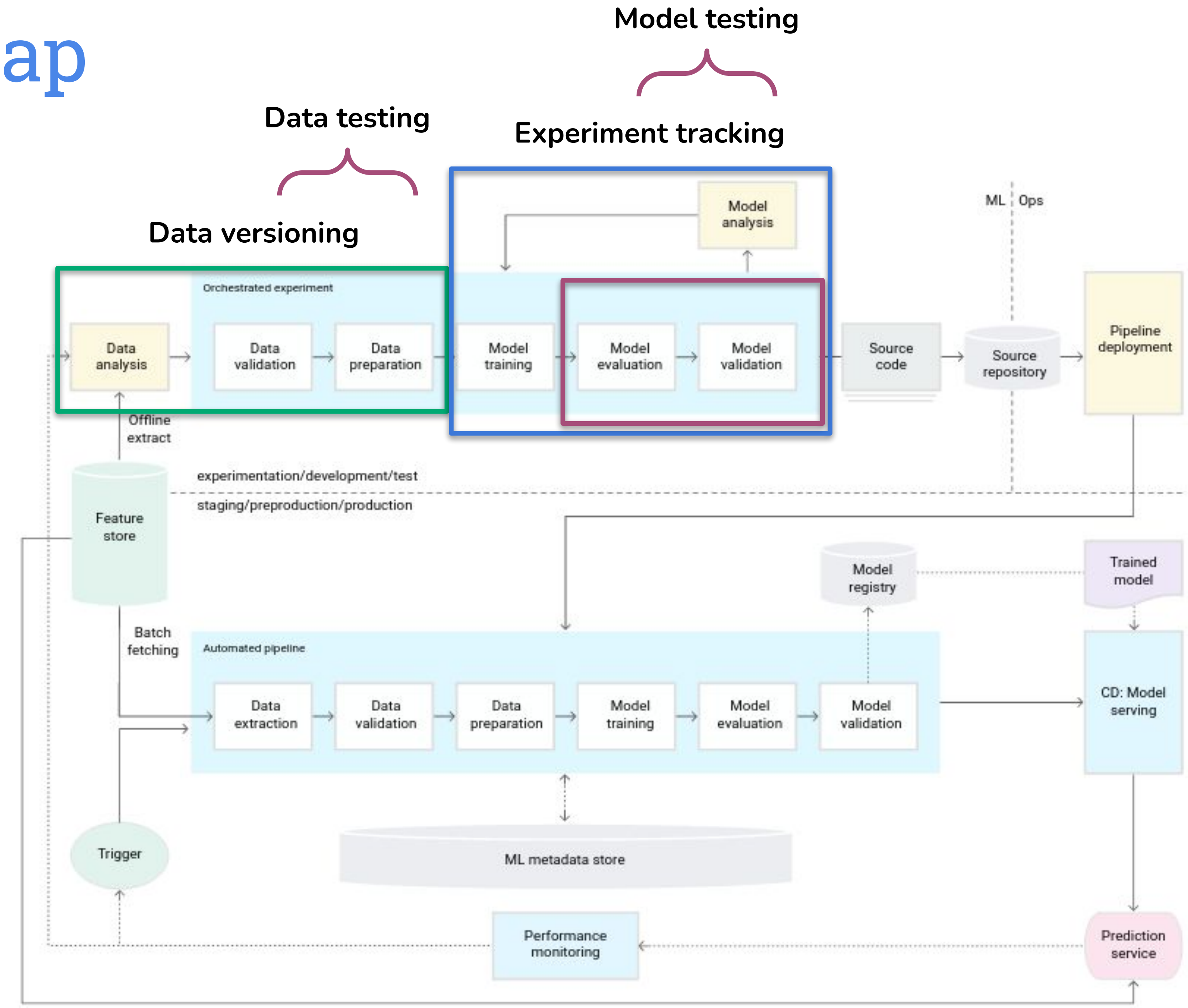
ADALTAS

- Goals of software testing:
  - Catch bugs early on
  - Assure that the software is **behaving** according to the requirements

    => behavior of the model is data-dependant and is based on learning (it’s not hard-coded)!

- Addition for ML systems:
  - The behavior needs to remain consistent during many calls
    - non-deterministic algorithms (deep learning)

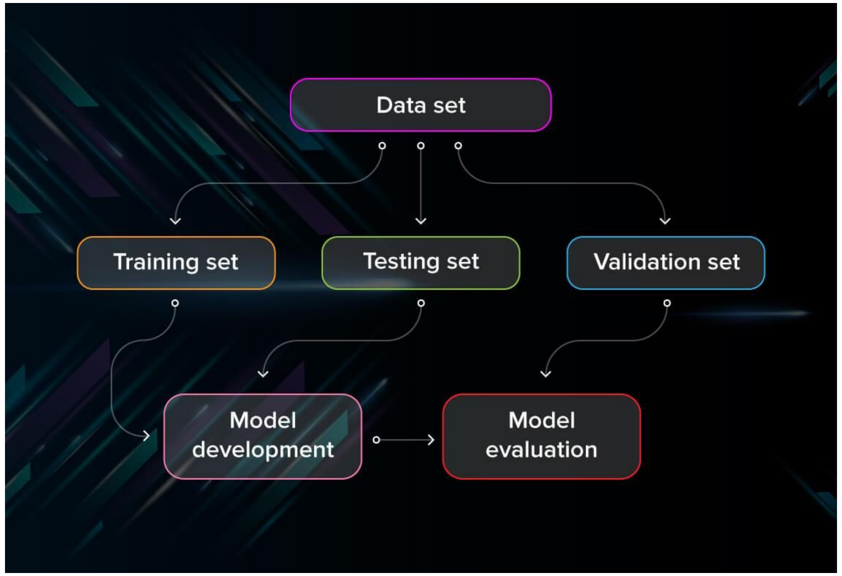
Roadmap

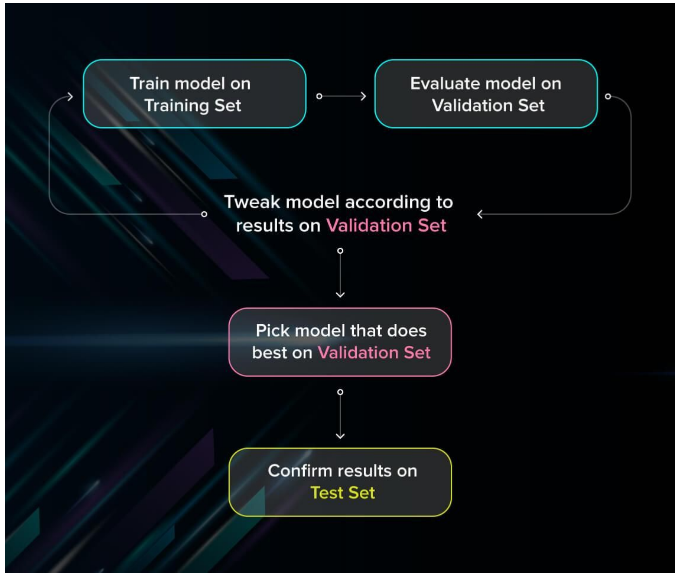


Model testing
Data testing
Experiment tracking
Data versioning
Model analysis
ML
Ops
Orchestrated experiment
Data analysis
Data validation
Data preparation
Model training
Model evaluation
Model validation
Source code
Source repository
Pipeline deployment
Offline extract
experimentation/development/test
staging/preproduction/production
Feature store
Model registry
Trained model
Batch fetching
Automated pipeline
Data extraction
Data validation
Data preparation
Model training
Model evaluation
Model validation
CD: Model serving
Trigger
ML metadata store
Performance monitoring
Prediction service

Model evaluation

- Evaluate the capabilities of generalisation (usual practice):
  - cross-validation
  - choose appropriate metric
- Good practice: split your dataset on 3 sets (not on 2 which you always see in class!)





https://serokell.io/blog/machine-learning-testing

Model validation

ADALTAS

- Logic of the algorithm - do we always test it? (output length, range, intended overfitting…)
- Compare to a baseline model (better than random?)
- Invariance tests - how much we can change the input without it affecting the performance of the model. For example, if we run a pattern recognition model on two different photos of red apples, we expect that the result will not change much.
- Directional expectation tests - how perturbations in input will change the behavior of the model. For example, when building a regression model that estimates the prices of houses and takes square meters as one of the parameters, we want to see that adding extra space makes the price go up.
- Minimum functionality tests - test the components of the program separately just like traditional unit tests. For example, you can assess the model on specific cases found in your data.

https://serokell.io/blog/machine-learning-testing

Data testing

- Raw data:
    - null values - not allowed in target column
    - schema
    - range - outliers
    - text - special symbols, capitalized letters
    - content, language - NLP

    => corrections before nonconform data enters the pipeline and breaks it

- Processed data:
    - distribution/range of engineered features
        - normalization: range 0-1
        - standardization: mean approx. 0

  => correctness of the transformations

What will we do?

To get the feeling for the tests, we will test the following aspects of data:

- Raw data: null values, distributions, range, schema
- Processed data: engineered features

Pytest

ADALTAS

```python
# content of test_sample.py
def inc(x):
    return x + 1


def test_answer():
    assert inc(3) == 5
```

To execute it:

```
$ pytest
=========================== test session starts ============================
platform linux -- Python 3.x.y, pytest-6.x.y, py-1.x.y, pluggy-0.x.y
cachedir: $PYTHON_PREFIX/.pytest_cache
rootdir: $REGENDOC_TMPDIR
collected 1 item

test_sample.py F                                                     [100%]

================================= FAILURES =================================
_______________________________ test_answer ________________________________

    def test_answer():
>       assert inc(3) == 5
E       assert 4 == 5
E        +  where 4 = inc(3)

test_sample.py:6: AssertionError
========================= short test summary info ==========================
FAILED test_sample.py::test_answer - assert 4 == 5
============================ 1 failed in 0.12s =============================
```

https://docs.pytest.org/en/6.2.x/

To go from here

ADALTAS

- Advanced data testing framework: [Great Expectations]
- Generating synthetic data: [Trumania] (and many others)