Day3 - Reproducibility and Software

02476 Machine Learning Operations

Nicki Skafte Detlefsen, Associate Professor, DTU Compute

January 2026

Whats in this presentation?

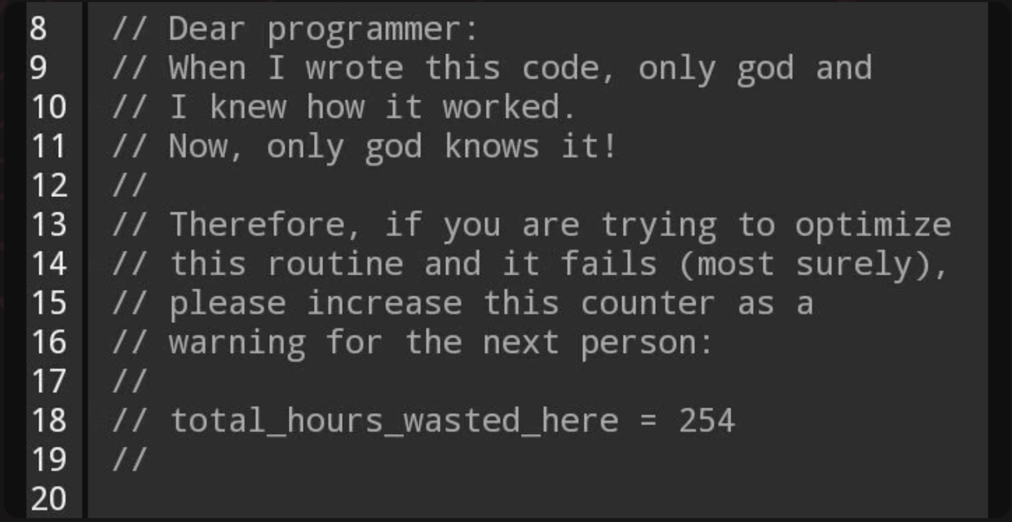
♻️ The reproducibility crisis

What is it?

How bad is it?

💻How can software help?

What to use when?



```
// Dear programmer:
// When I wrote this code, only god and
// I knew how it worked.
// Now, only god knows it!
//
// Therefore, if you are trying to optimize
// this routine and it fails (most surely),
// please increase this counter as a
// warning for the next person:
//
// total_hours_wasted_here = 254
//

```

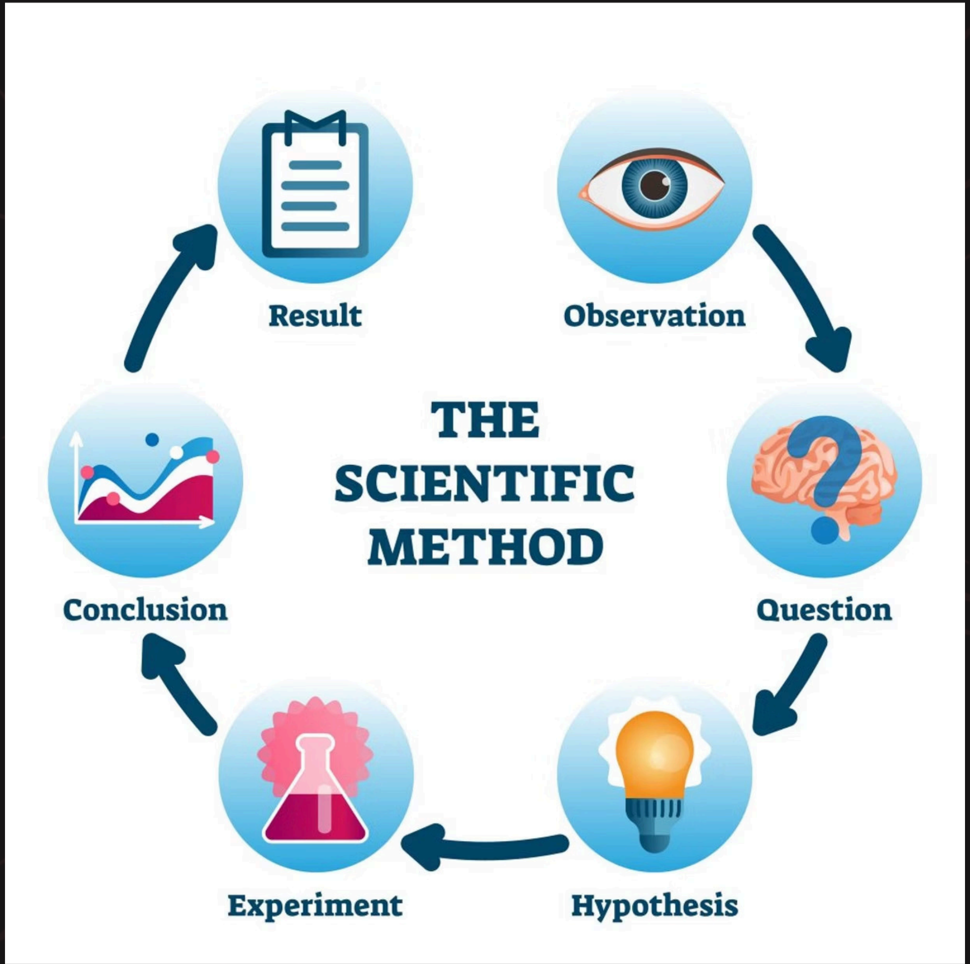
What is reproducibility?

💡 Reproducibility is the ability of **an entire experiment** or study to be duplicated, either by the same researcher or **by someone else working independently**.

💡 Reproducible data - **repeatability** which is the degree of agreement of tests or measurements on replicate specimens by the same observer in the same laboratory.

💡 Computationally reproducible research - the idea that the ultimate product of **academic research** is the paper along with the **full computational environment** used to produce the results in the paper such as the code, data, etc. that can be used to reproduce the results and create new work based on the research.

Why do we need it? (research perspective)



Why do we need it? (industry perspective)

🚩Knowledge Preservation	If other cannot reproduce your work, knowledge can be lost
🚩Transparency and Accountability	To make sure that others can verify your claims before going into production
🚩Regulatory Compliance	To secure the correct documentation to make sure everything is compliant
🚩Continuous Improvement	Making sure that improvements to the pipeline are real and not artifacts of random effects

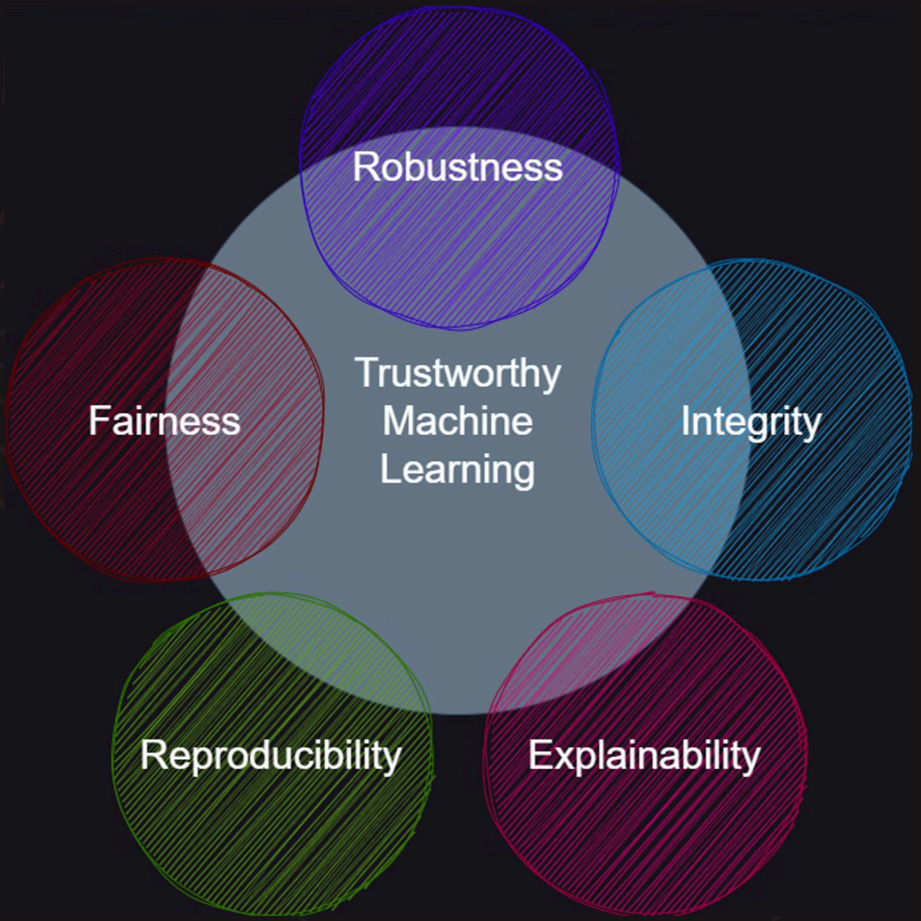
Trustworthy ML/AI

Trustworthy ML/AI

Reproducibility is a key component in *Trustworthy ML*

⚠️Case:

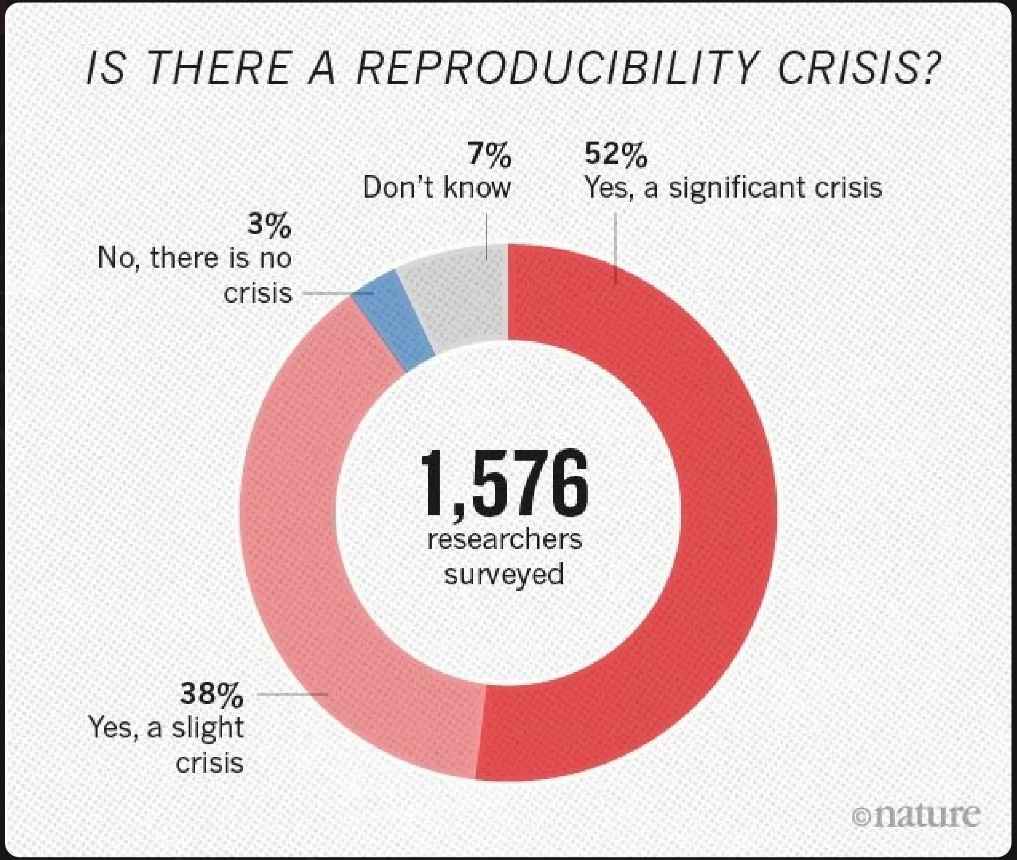
Imaging an AI agent used for diagnostics. Without reproducibility two persons with the exact same symptoms could get different diagnosis



We are in a crises!

🚩 There is growing alarm about results that cannot be reproduced. Explanations include

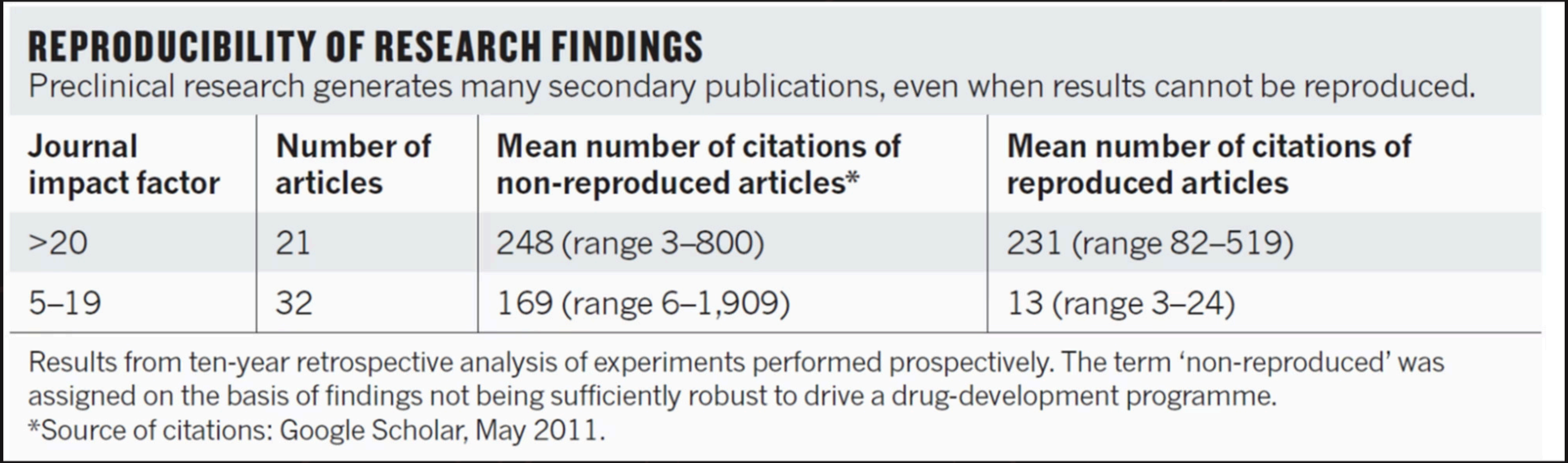
- Increased levels of scrutiny
- Complexity of experiments and statistics
- Pressures on researchers



An example: Trouble in the lab

The biotech company Amgen had a team of about 100 scientists trying to reproduce the findings of 53 “landmark” articles in cancer research published by reputable labs in top journals.

Only 6 of the 53 studies were reproduced (about 10%).



REPRODUCIBILITY OF RESEARCH FINDINGS

Preclinical research generates many secondary publications, even when results cannot be reproduced.

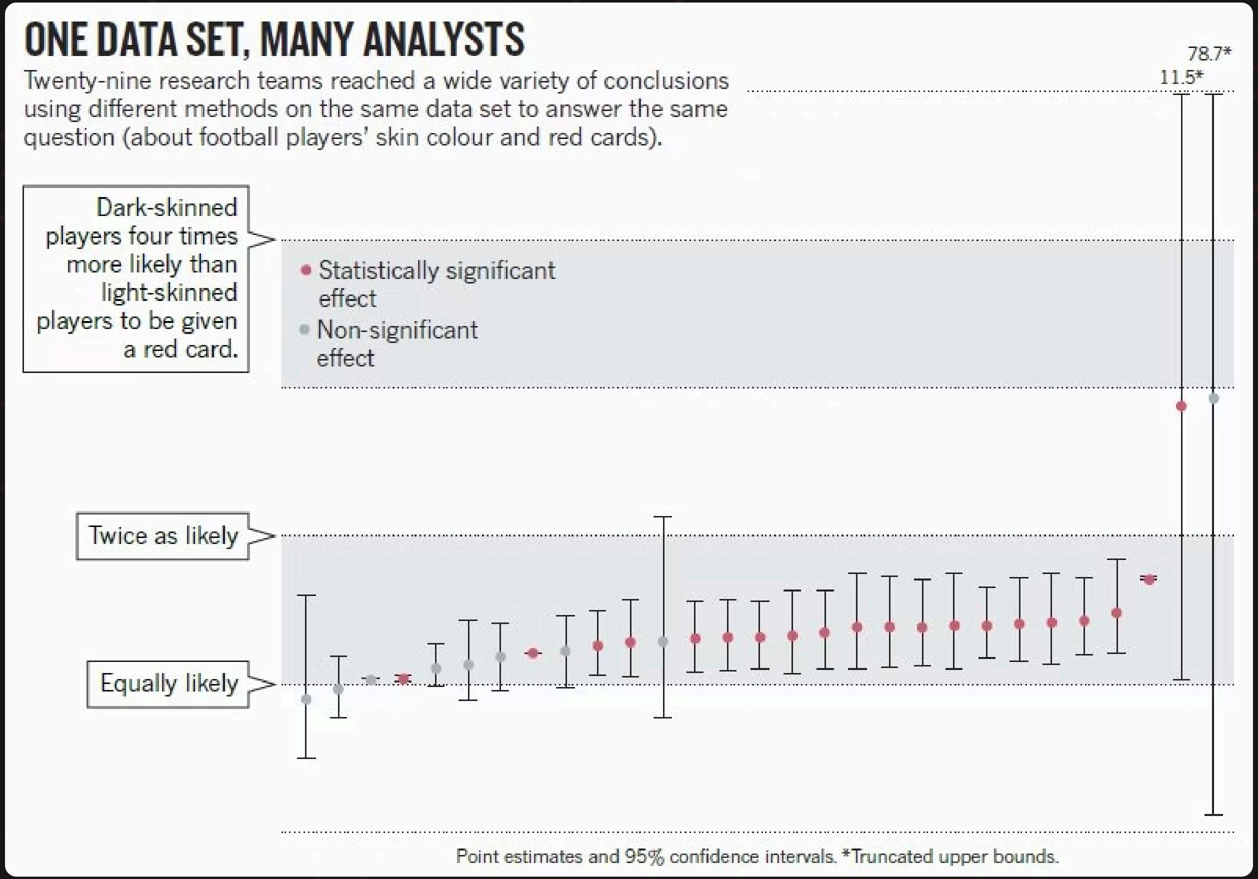
Journal impact factor	Number of articles	Mean number of citations of non-reproduced articles*	Mean number of citations of reproduced articles
>20	21	248 (range 3–800)	231 (range 82–519)
5–19	32	169 (range 6–1,909)	13 (range 3–24)

Results from ten-year retrospective analysis of experiments performed prospectively. The term ‘non-reproduced’ was assigned on the basis of findings not being sufficiently robust to drive a drug-development programme.

*Source of citations: Google Scholar, May 2011.

Another example: We are only humans

💭 The (subjective) choices we take during research may impact the conclusion

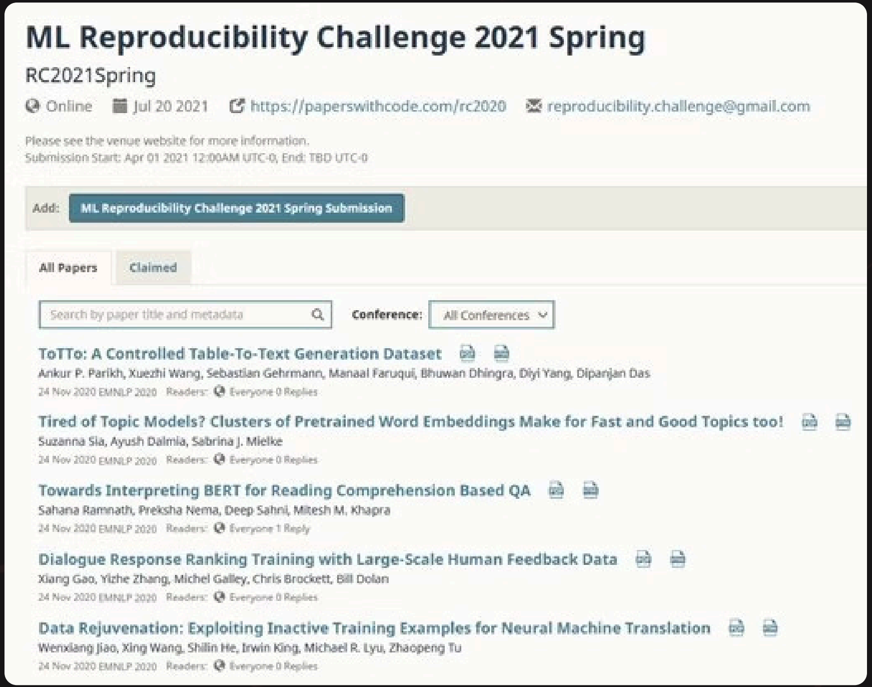


ONE DATA SET, MANY ANALYSTS

Twenty-nine research teams reached a wide variety of conclusions using different methods on the same data set to answer the same question (about football players' skin colour and red cards).

Point estimates and 95% confidence intervals. *Truncated upper bounds.

What are we trying to do within research?



ML Reproducibility Challenge 2021 Spring

RC2021Spring

Online Jul 20 2021 https://paperswithcode.com/rc2020 reproducibility.challenge@gmail.com

Please see the venue website for more information.
Submission Start: Apr 01 2021 12:00AM UTC-0, End: TBD UTC-0

Add:
ML Reproducibility Challenge 2021 Spring Submission
All Papers
Claimed
Search by paper title and metadata
Conference:
All Conferences

ToTTo: A Controlled Table-To-Text Generation Dataset
Ankur P. Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, Dipanjan Das
24 Nov 2020 EMNLP 2020 Readers: Everyone 0 Replies

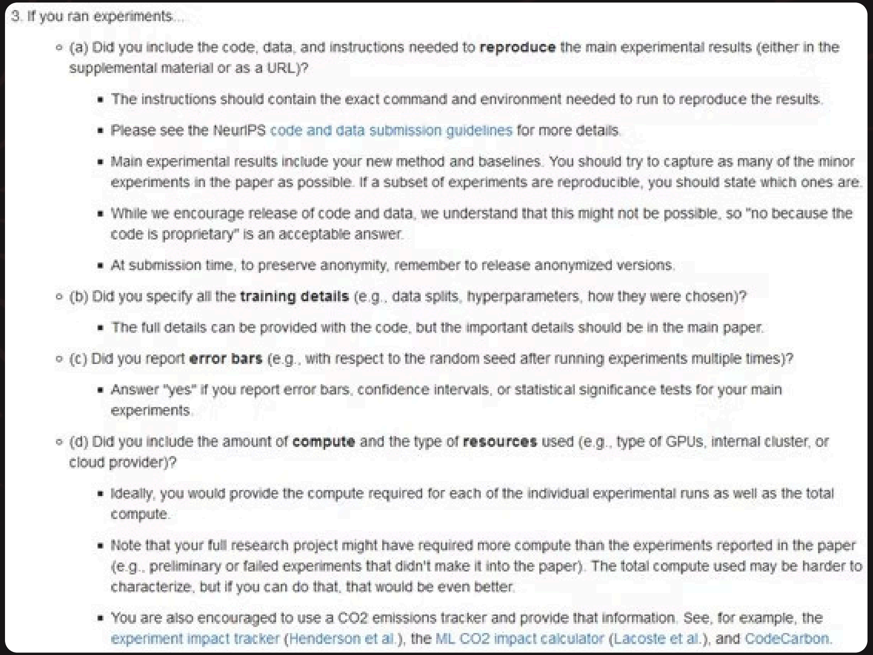
Tired of Topic Models? Clusters of Pretrained Word Embeddings Make for Fast and Good Topics too!
Suzanna Sia, Ayush Dalmia, Sabrina J. Mielke
24 Nov 2020 EMNLP 2020 Readers: Everyone 0 Replies

Towards Interpreting BERT for Reading Comprehension Based QA
Sahana Ramnath, Preksha Nema, Deep Sahni, Mitesh M. Khapra
24 Nov 2020 EMNLP 2020 Readers: Everyone 1 Reply

Dialogue Response Ranking Training with Large-Scale Human Feedback Data
Xiang Gao, Yizhe Zhang, Michel Galley, Chris Brockett, Bill Dolan
24 Nov 2020 EMNLP 2020 Readers: Everyone 0 Replies

Data Rejuvenation: Exploiting Inactive Training Examples for Neural Machine Translation
Wenxiang Jiao, Xing Wang, Shilin He, Irwin King, Michael R. Lyu, Zhaopeng Tu
24 Nov 2020 EMNLP 2020 Readers: Everyone 0 Replies

https://paperswithcode.com/



3. If you ran experiments...
   - (a) Did you include the code, data, and instructions needed to **reproduce** the main experimental results (either in the supplemental material or as a URL)?
     - The instructions should contain the exact command and environment needed to run to reproduce the results.
     - Please see the NeurIPS code and data submission guidelines for more details.
     - Main experimental results include your new method and baselines. You should try to capture as many of the minor experiments in the paper as possible. If a subset of experiments are reproducible, you should state which ones are.
     - While we encourage release of code and data, we understand that this might not be possible, so "no because the code is proprietary" is an acceptable answer.
     - At submission time, to preserve anonymity, remember to release anonymized versions.
   - (b) Did you specify all the **training details** (e.g., data splits, hyperparameters, how they were chosen)?
     - The full details can be provided with the code, but the important details should be in the main paper.
   - (c) Did you report **error bars** (e.g., with respect to the random seed after running experiments multiple times)?
     - Answer "yes" if you report error bars, confidence intervals, or statistical significance tests for your main experiments.
   - (d) Did you include the amount of **compute** and the type of **resources** used (e.g., type of GPUs, internal cluster, or cloud provider)?
     - Ideally, you would provide the compute required for each of the individual experimental runs as well as the total compute.
     - Note that your full research project might have required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper). The total compute used may be harder to characterize, but if you can do that, that would be even better.
     - You are also encouraged to use a CO2 emissions tracker and provide that information. See, for example, the experiment impact tracker (Henderson et al.), the ML CO2 impact calculator (Lacoste et al.), and CodeCarbon.

Checklist for conferences

Reproducibility is not binary

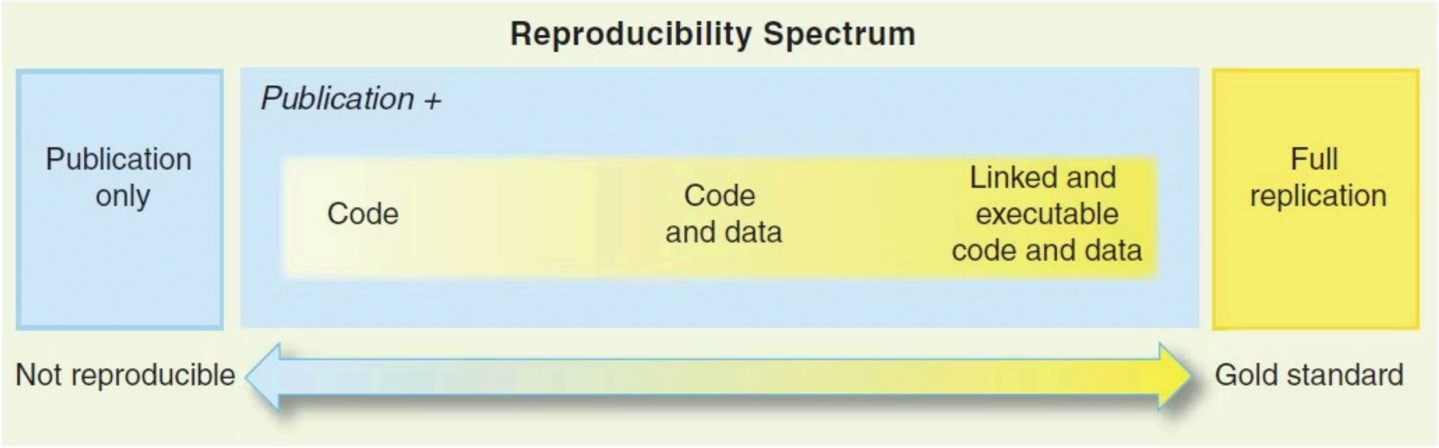


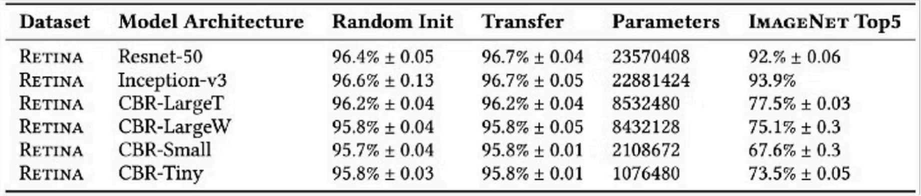
Fig. 1. The spectrum of reproducibility.

Example from ML

$$\begin{Bmatrix} w_1 \\ \vdots \\ w_n \end{Bmatrix} == \begin{Bmatrix} w'_1 \\ \vdots \\ w'_n \end{Bmatrix}$$

m1.ckpt m2.ckpt

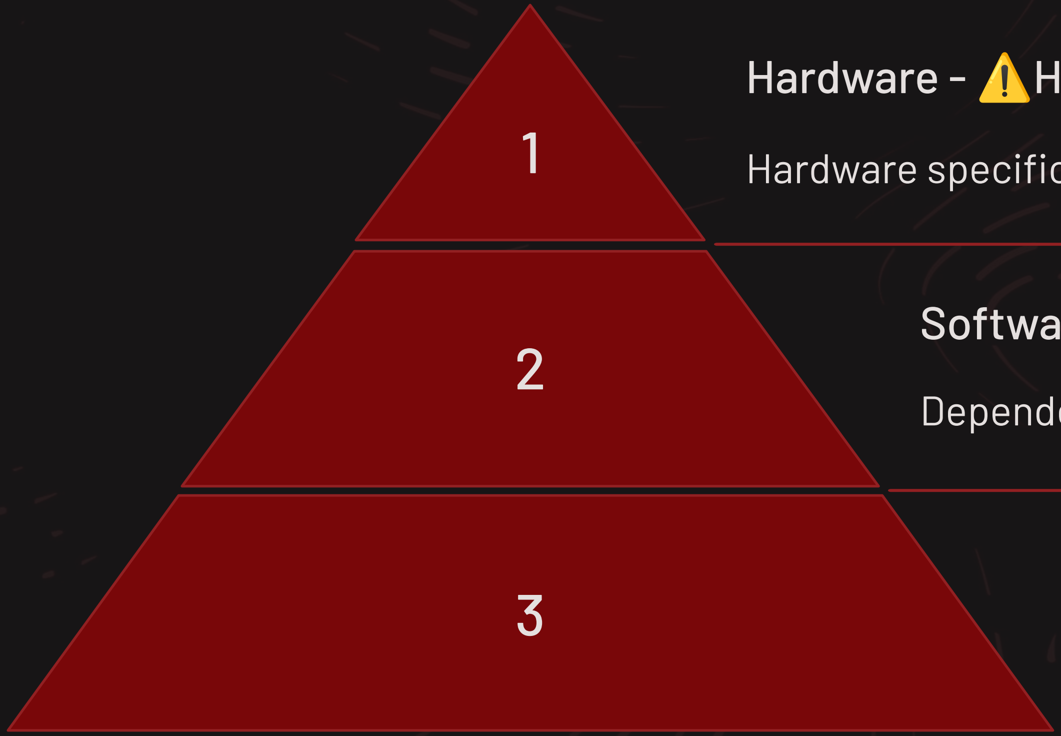
VS



Dataset	Model Architecture	Random Init	Transfer	Parameters	ImageNet Top5
Retina	Resnet-50	96.4% ± 0.05	96.7% ± 0.04	23570408	92.% ± 0.06
Retina	Inception-v3	96.6% ± 0.13	96.7% ± 0.05	22881424	93.9%
Retina	CBR-LargeT	96.2% ± 0.04	96.2% ± 0.04	8532480	77.5% ± 0.03
Retina	CBR-LargeW	95.8% ± 0.04	95.8% ± 0.05	8432128	75.1% ± 0.3
Retina	CBR-Small	95.7% ± 0.04	95.8% ± 0.01	2108672	67.6% ± 0.3
Retina	CBR-Tiny	95.8% ± 0.03	95.8% ± 0.01	1076480	73.5% ± 0.05

What can we do about it ?

🔥Nicki’s hierarchy of reproducibility of ML🔥 → Make sure that you document everything about your experiments



Hardware - ⚠️Hard to ship/✅Highly reproducible

Hardware specifications

Software - Tradeoff between the two

Dependency specifications

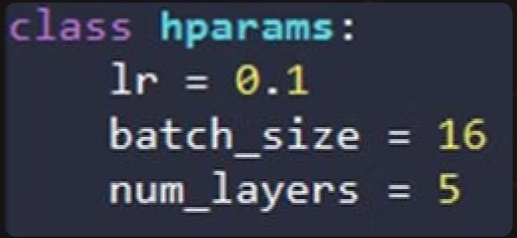
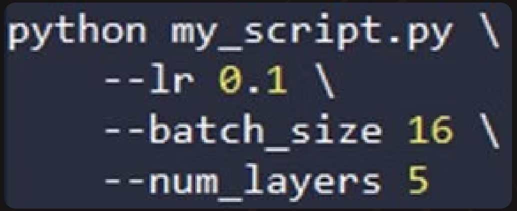
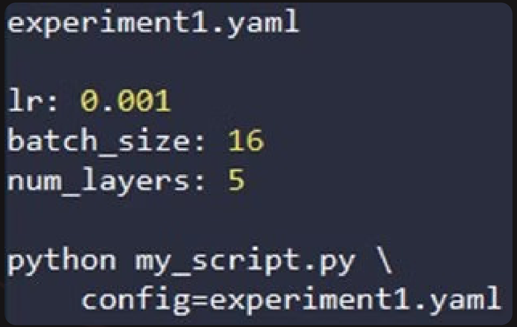
Code - ✅Easy to ship/⚠️Less reproducible

Configs, data, runnable code

Reproducibility level 1

There is a lot of subjective choices that we do when running experiments in machine learning, most notable the hyperparameters

Parameters in scripts	Argument parser	Config files

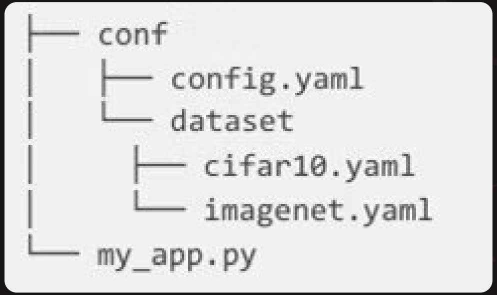

`class hparams:`
`    lr = 0.1`
`    batch_size = 16`
`    num_layers = 5`
	
	`python my_script.py \`
	`    --lr 0.1 \`
	`    --batch_size 16 \`
	`    --num_layers 5`
		
		`experiment1.yaml`
		
		`lr: 0.001`
		`batch_size: 16`
		`num_layers: 5`
		
		`python my_script.py \`
		`    config=experiment1.yaml`

✅Easy to code ⚠️Not easy to configure on the run ⚠️Experimental info may be lost if not careful	✅Easy to configure ⚠️Falls on user to save the config	✅Highly configurable ✅Configuration is systematically saved (and version controlled)

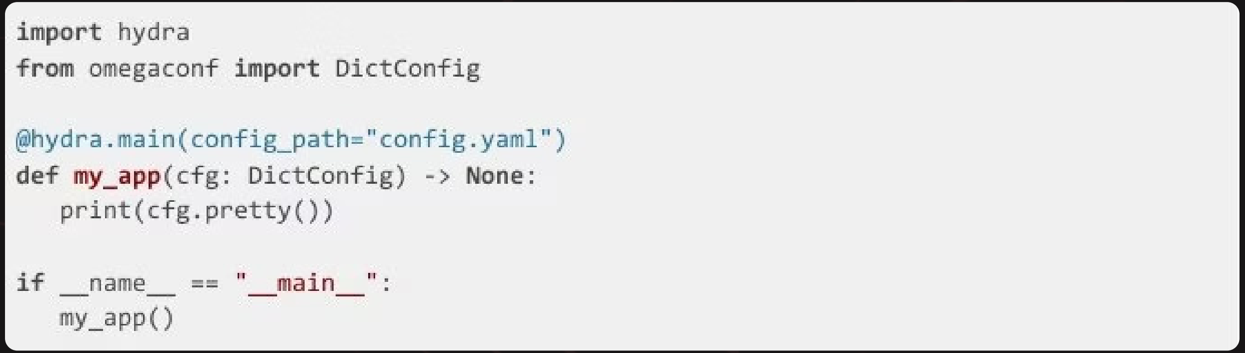
Reproducibility level 1

Hydra is a framework for elegantly configuring complex (ML) applications

https://github.com/facebookresearch/hydra



```
├── conf
│   ├── config.yaml
│   └── dataset
│       ├── cifar10.yaml
│       └── imagenet.yaml
└── my_app.py
```



```python
import hydra
from omegaconf import DictConfig

@hydra.main(config_path="config.yaml")
def my_app(cfg: DictConfig) -> None:
    print(cfg.pretty())

if __name__ == "__main__":
    my_app()
```

Other options

💡 https://github.com/IDSIA/sacred

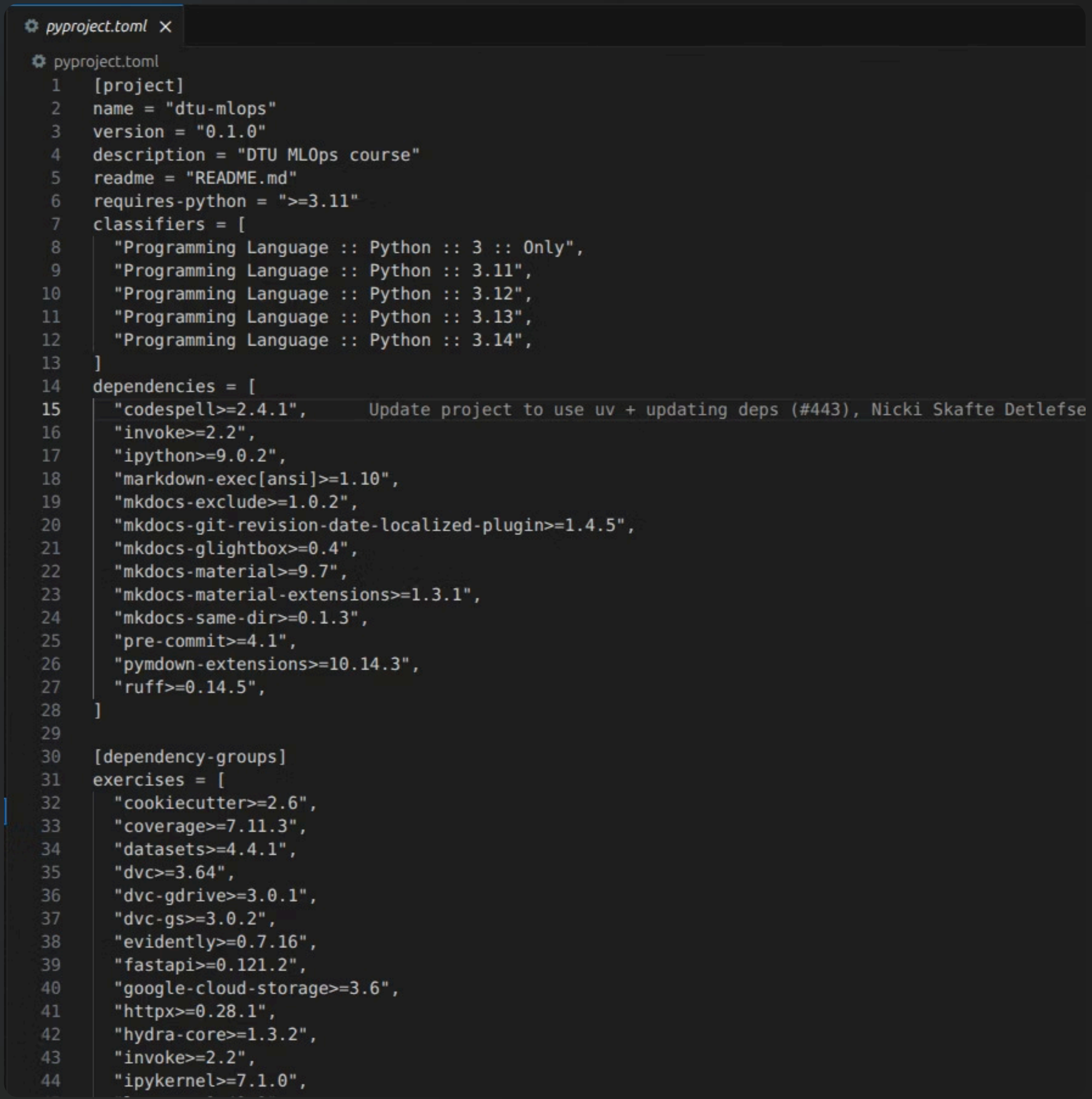
💡 https://mlflow.org/

Reproducibility level 2

For python: Just use a package management system

Examples:

- 💡 Conda
- 💡 Pipenv
- 💡 venv
- 💡 pyenv
- 💡 uv



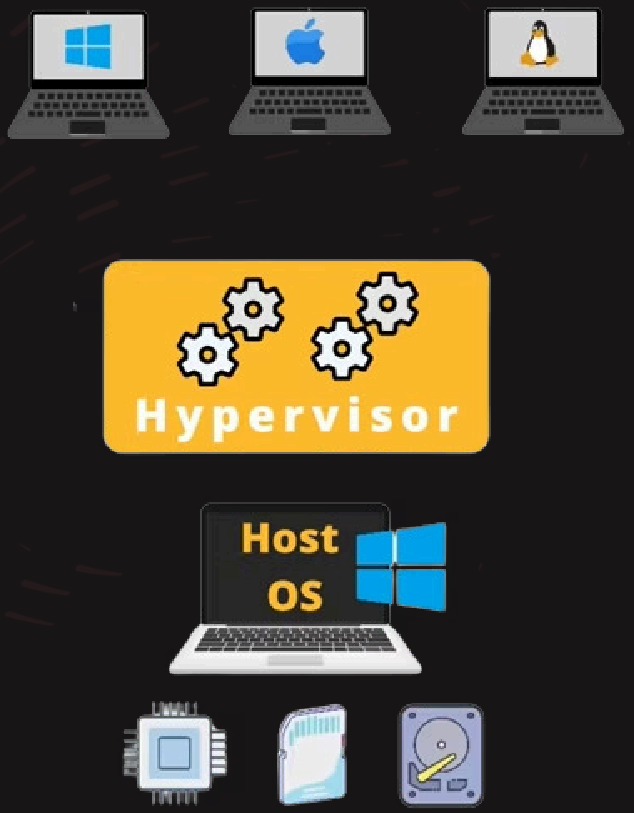
```toml
[project]
name = "dtu-mlops"
version = "0.1.0"
description = "DTU MLOps course"
readme = "README.md"
requires-python = ">=3.11"
classifiers = [
  "Programming Language :: Python :: 3 :: Only",
  "Programming Language :: Python :: 3.11",
  "Programming Language :: Python :: 3.12",
  "Programming Language :: Python :: 3.13",
  "Programming Language :: Python :: 3.14",
]
dependencies = [
  "codespell>=2.4.1",       Update project to use uv + updating deps (#443), Nicki Skafte Detlefse
  "invoke>=2.2",
  "ipython>=9.0.2",
  "markdown-exec[ansi]>=1.10",
  "mkdocs-exclude>=1.0.2",
  "mkdocs-git-revision-date-localized-plugin>=1.4.5",
  "mkdocs-glightbox>=0.4",
  "mkdocs-material>=9.7",
  "mkdocs-material-extensions>=1.3.1",
  "mkdocs-same-dir>=0.1.3",
  "pre-commit>=4.1",
  "pymdown-extensions>=10.14.3",
  "ruff>=0.14.5",
]

[dependency-groups]
exercises = [
  "cookiecutter>=2.6",
  "coverage>=7.11.3",
  "datasets>=4.4.1",
  "dvc>=3.64",
  "dvc-gdrive>=3.0.1",
  "dvc-gs>=3.0.2",
  "evidently>=0.7.16",
  "fastapi>=0.121.2",
  "google-cloud-storage>=3.6",
  "httpx>=0.28.1",
  "hydra-core>=1.3.2",
  "invoke>=2.2",
  "ipykernel>=7.1.0",
```

Reproducibility level 3

💡 The easiest way for someone to reproduce your work, would be to just hand over your computer.

💡 Instead of doing this, lets hand over a virtual copy of our machine

Virtual machine works by taking hardware from the host and creates virtual CPU, RAM, storage for each virtual machine. The virtual machines are completely independent from the host

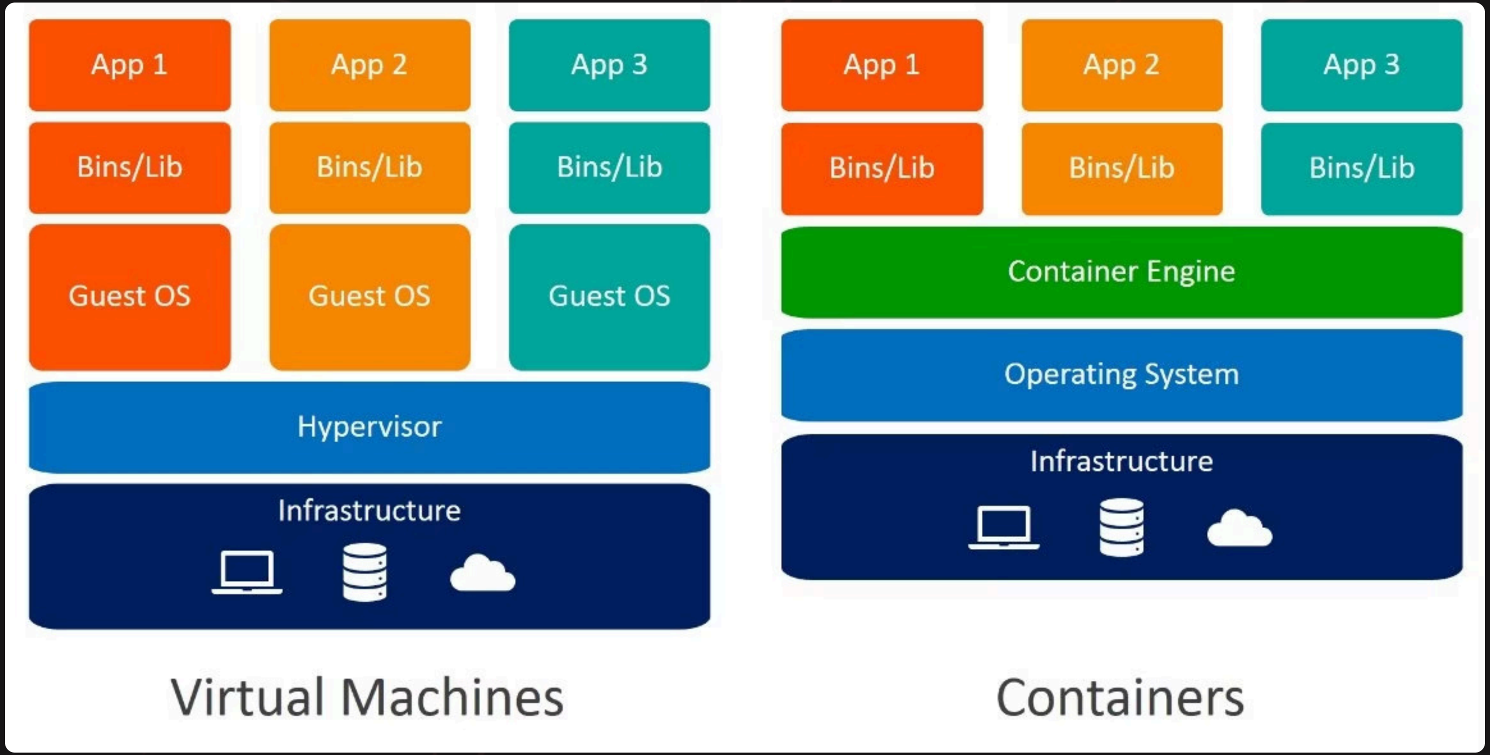


Reproducibility level 3

💡 The core advantage of a VM is that it in principal can run on any host without changes, because it is independent.

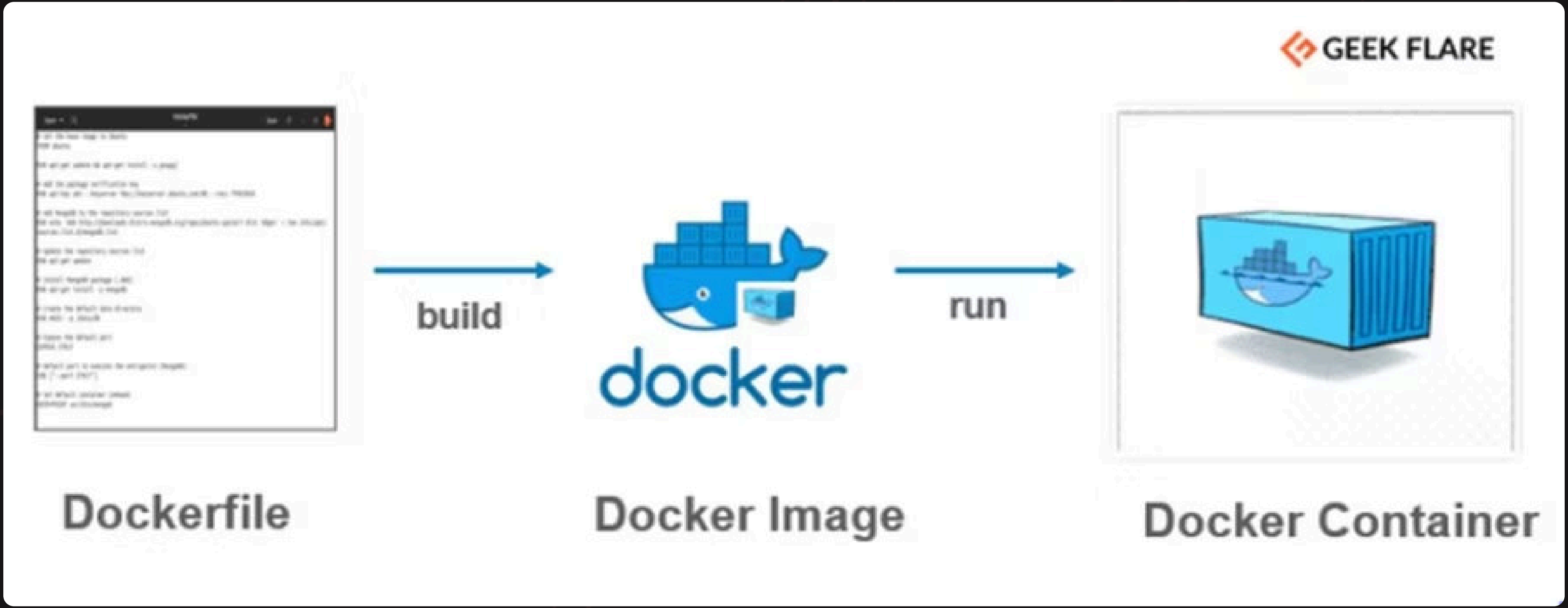
💡 Docker can be seen as an lightweight version of full VMs.

💡 *VM is isolation of machines, while Containers is isolation of processes*



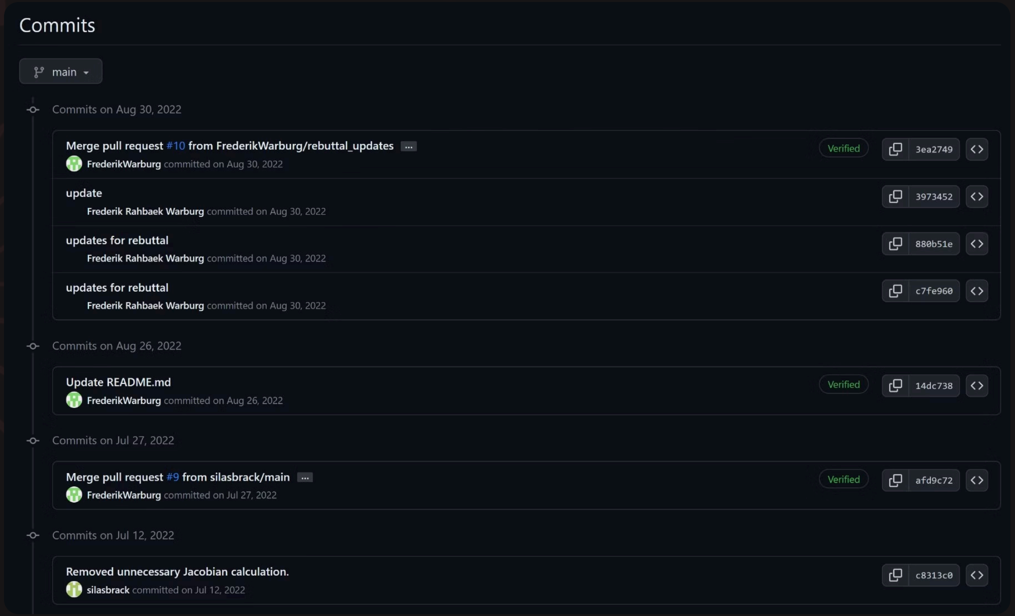
Reproducibility level 3

A way to create containerized applications = low overhead virtual machines (VMs)



A 6-step process for reproducible software

Use version control



Commits

main

Commits on Aug 30, 2022

- Merge pull request #10 from FrederikWarburg/rebuttal_updates …
  FrederikWarburg committed on Aug 30, 2022 — Verified — 3ea2749
- update
  Frederik Rahbaek Warburg committed on Aug 30, 2022 — 3973452
- updates for rebuttal
  Frederik Rahbaek Warburg committed on Aug 30, 2022 — 880b51e
- updates for rebuttal
  Frederik Rahbaek Warburg committed on Aug 30, 2022 — c7fe960

Commits on Aug 26, 2022

- Update README.md
  FrederikWarburg committed on Aug 26, 2022 — Verified — 14dc738

Commits on Jul 27, 2022

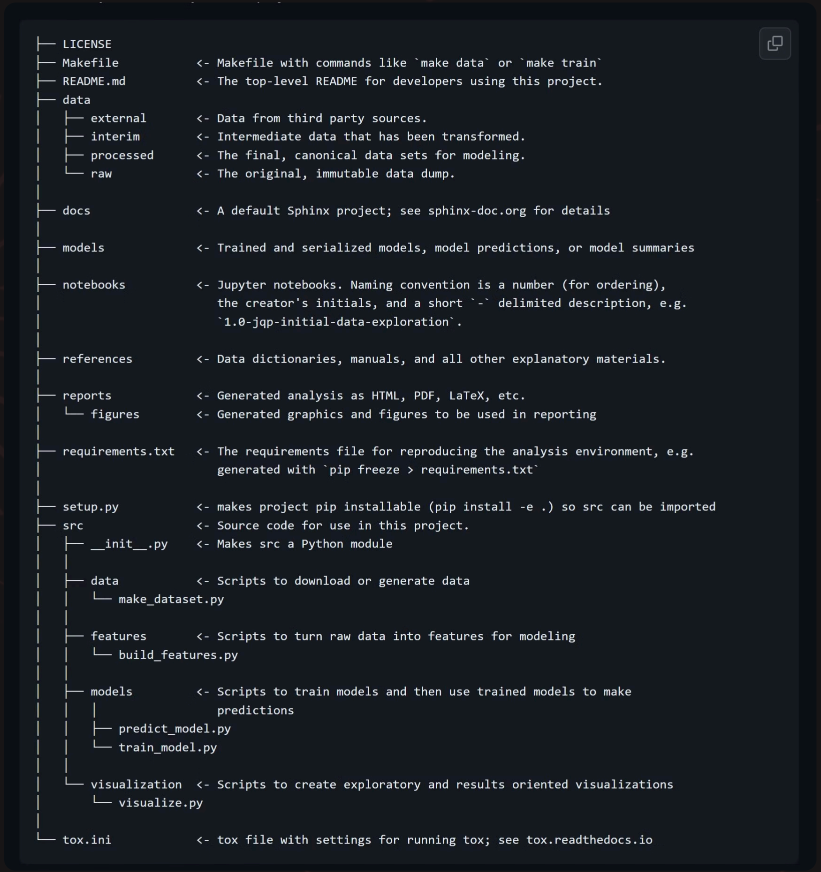
- Merge pull request #9 from silasbrack/main …
  FrederikWarburg committed on Jul 27, 2022 — Verified — afd9c72

Commits on Jul 12, 2022

- Removed unnecessary Jacobian calculation.
  silasbrack committed on Jul 12, 2022 — c8313c0

A 6-step process for reproducible software

- Use version control
- Use templates

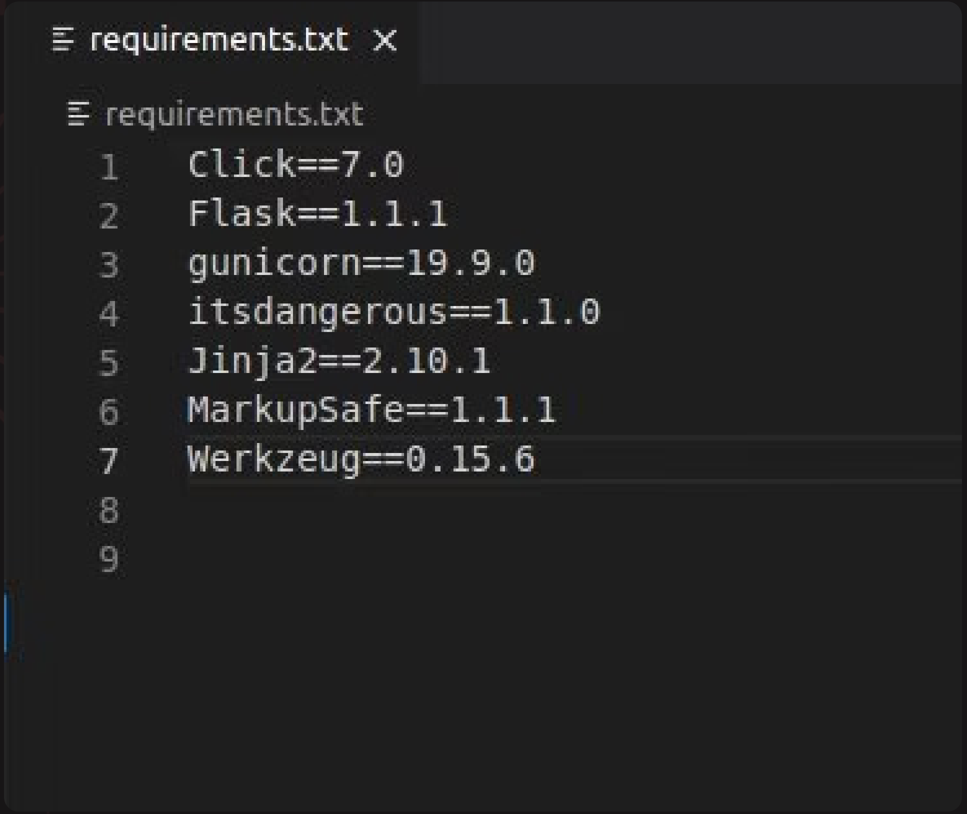


```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.py           <- makes project pip installable (pip install -e .) so src can be imported
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```

A 6-step process for reproducible software

- Use version control
- Use templates
- Write down your dependencies

  (and use virtual environments)

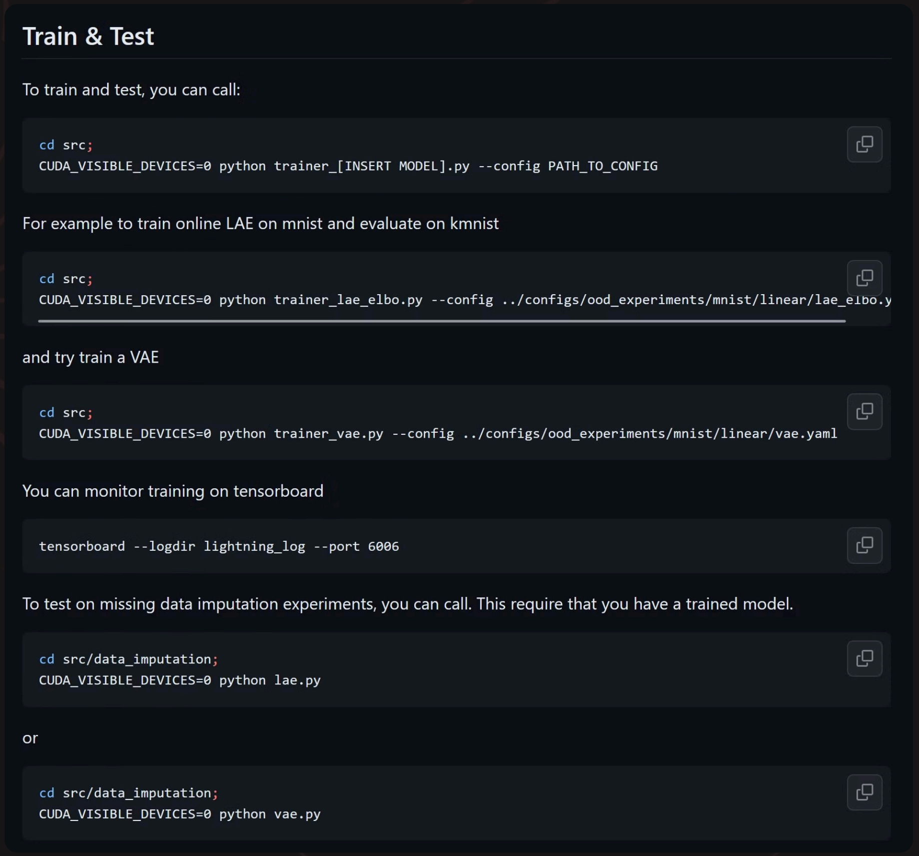


```
Click==7.0
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.6
```

A 6-step process for reproducible software

- Use version control
- Use templates
- Write down your dependencies

  (and use virtual environments)
- Document your code!



Train & Test

To train and test, you can call:

```
cd src;
CUDA_VISIBLE_DEVICES=0 python trainer_[INSERT MODEL].py --config PATH_TO_CONFIG
```

For example to train online LAE on mnist and evaluate on kmnist

```
cd src;
CUDA_VISIBLE_DEVICES=0 python trainer_lae_elbo.py --config ../configs/ood_experiments/mnist/linear/lae_elbo.y
```

and try train a VAE

```
cd src;
CUDA_VISIBLE_DEVICES=0 python trainer_vae.py --config ../configs/ood_experiments/mnist/linear/vae.yaml
```

You can monitor training on tensorboard

```
tensorboard --logdir lightning_log --port 6006
```

To test on missing data imputation experiments, you can call. This require that you have a trained model.

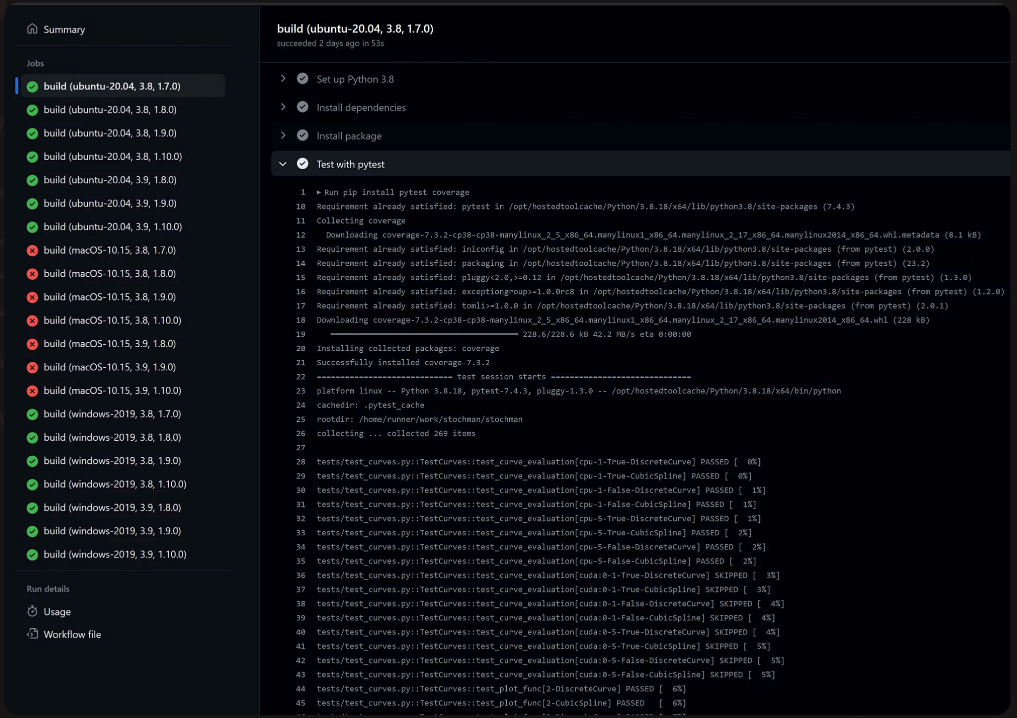
```
cd src/data_imputation;
CUDA_VISIBLE_DEVICES=0 python lae.py
```

or

```
cd src/data_imputation;
CUDA_VISIBLE_DEVICES=0 python vae.py
```

A 6-step process for reproducible software

- Use version control
- Use templates
- Write down your dependencies (and use virtual environments)
- Document your code!
- Test your code



Summary

Jobs

- ✓ build (ubuntu-20.04, 3.8, 1.7.0)
- ✓ build (ubuntu-20.04, 3.8, 1.8.0)
- ✓ build (ubuntu-20.04, 3.8, 1.9.0)
- ✓ build (ubuntu-20.04, 3.8, 1.10.0)
- ✓ build (ubuntu-20.04, 3.9, 1.8.0)
- ✓ build (ubuntu-20.04, 3.9, 1.9.0)
- ✓ build (ubuntu-20.04, 3.9, 1.10.0)
- ✗ build (macOS-10.15, 3.8, 1.7.0)
- ✗ build (macOS-10.15, 3.8, 1.8.0)
- ✗ build (macOS-10.15, 3.8, 1.9.0)
- ✗ build (macOS-10.15, 3.8, 1.10.0)
- ✗ build (macOS-10.15, 3.9, 1.8.0)
- ✗ build (macOS-10.15, 3.9, 1.9.0)
- ✗ build (macOS-10.15, 3.9, 1.10.0)
- ✓ build (windows-2019, 3.8, 1.7.0)
- ✓ build (windows-2019, 3.8, 1.8.0)
- ✓ build (windows-2019, 3.8, 1.9.0)
- ✓ build (windows-2019, 3.8, 1.10.0)
- ✓ build (windows-2019, 3.9, 1.8.0)
- ✓ build (windows-2019, 3.9, 1.9.0)
- ✓ build (windows-2019, 3.9, 1.10.0)

Run details

- Usage
- Workflow file

build (ubuntu-20.04, 3.8, 1.7.0)

succeeded 2 days ago in 53s

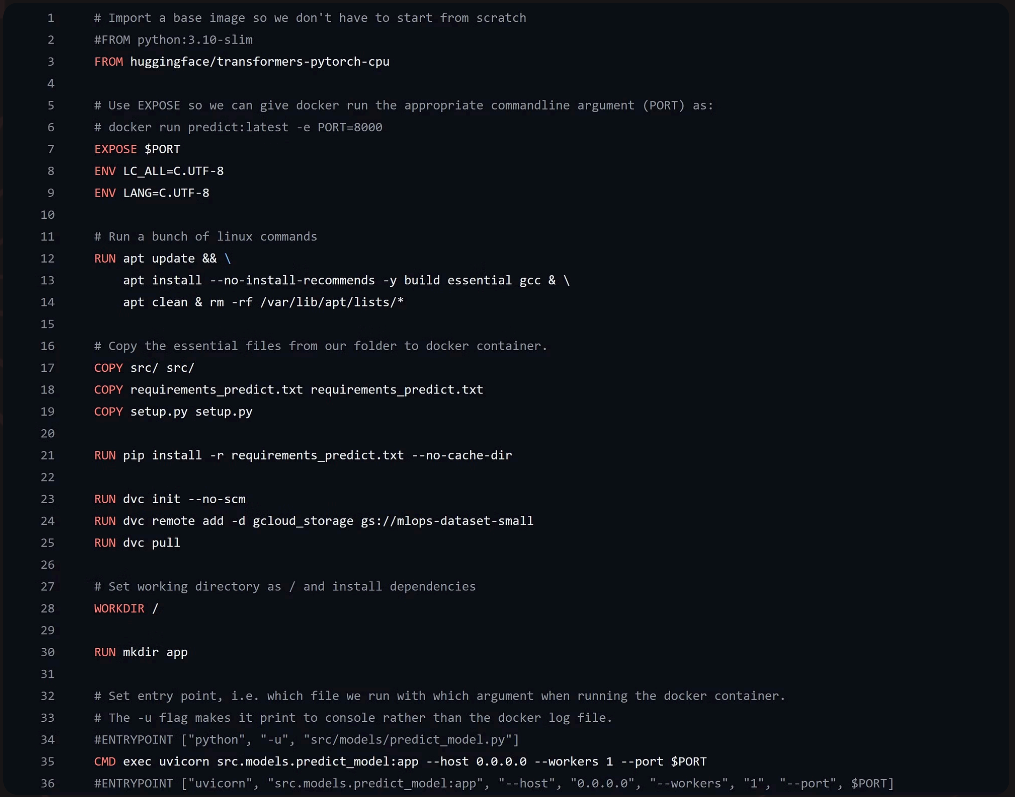
- ✓ Set up Python 3.8
- ✓ Install dependencies
- ✓ Install package
- ✓ Test with pytest

```
▸ Run pip install pytest coverage
Requirement already satisfied: pytest in /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages (7.4.3)
Collecting coverage
  Downloading coverage-7.3.2-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (8.1 kB)
Requirement already satisfied: iniconfig in /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages (from pytest) (2.0.0)
Requirement already satisfied: packaging in /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages (from pytest) (23.2)
Requirement already satisfied: pluggy<2.0,>=0.12 in /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages (from pytest) (1.3.0)
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages (from pytest) (1.2.0)
Requirement already satisfied: tomli>=1.0.0 in /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages (from pytest) (2.0.1)
Downloading coverage-7.3.2-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (228 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 228.6/228.6 kB 42.2 MB/s eta 0:00:00
Installing collected packages: coverage
Successfully installed coverage-7.3.2
============================= test session starts ==============================
platform linux -- Python 3.8.18, pytest-7.4.3, pluggy-1.3.0 -- /opt/hostedtoolcache/Python/3.8.18/x64/bin/python
cachedir: .pytest_cache
rootdir: /home/runner/work/stochman/stochman
collecting ... collected 269 items

tests/test_curves.py::TestCurves::test_curve_evaluation[cpu-1-True-DiscreteCurve] PASSED [  0%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cpu-1-True-CubicSpline] PASSED [  0%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cpu-1-False-DiscreteCurve] PASSED [  1%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cpu-1-False-CubicSpline] PASSED [  1%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cpu-5-True-DiscreteCurve] PASSED [  1%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cpu-5-True-CubicSpline] PASSED [  2%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cpu-5-False-DiscreteCurve] PASSED [  2%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cpu-5-False-CubicSpline] PASSED [  2%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cuda:0-1-True-DiscreteCurve] SKIPPED [  3%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cuda:0-1-True-CubicSpline] SKIPPED [  3%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cuda:0-1-False-DiscreteCurve] SKIPPED [  4%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cuda:0-1-False-CubicSpline] SKIPPED [  4%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cuda:0-5-True-DiscreteCurve] SKIPPED [  4%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cuda:0-5-True-CubicSpline] SKIPPED [  5%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cuda:0-5-False-DiscreteCurve] SKIPPED [  5%]
tests/test_curves.py::TestCurves::test_curve_evaluation[cuda:0-5-False-CubicSpline] SKIPPED [  5%]
tests/test_curves.py::TestCurves::test_plot_func[2-DiscreteCurve] PASSED [  6%]
tests/test_curves.py::TestCurves::test_plot_func[2-CubicSpline] PASSED   [  6%]
```

A 6-step process for reproducible software

- Use version control
- Use templates
- Write down your dependencies

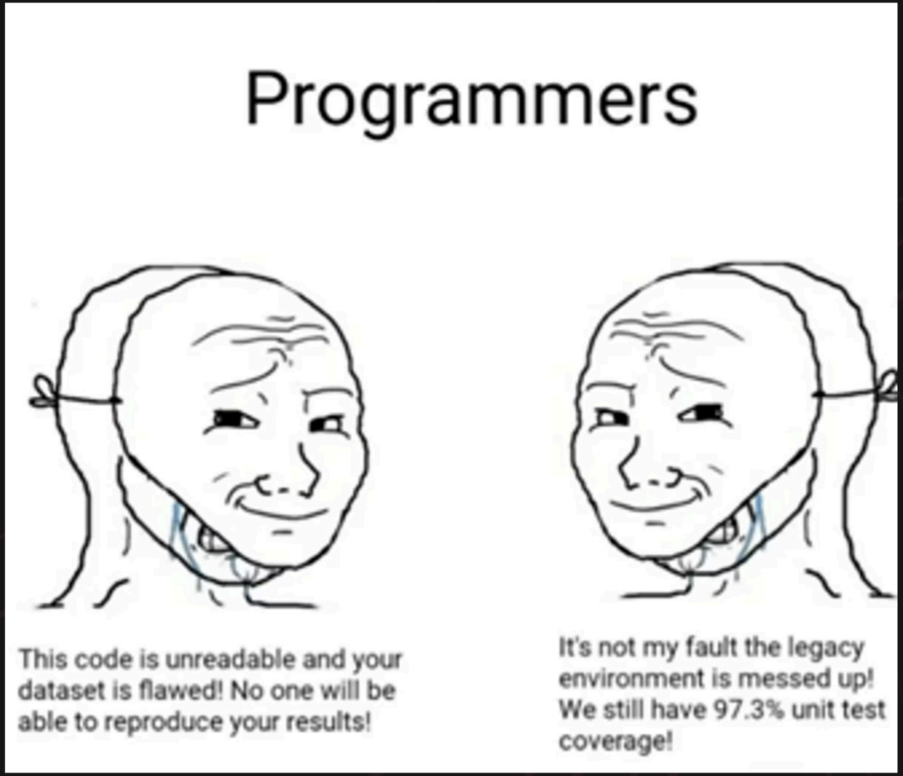
  (and use virtual environments)
- Document your code!
- Test your code
- Containerize your code

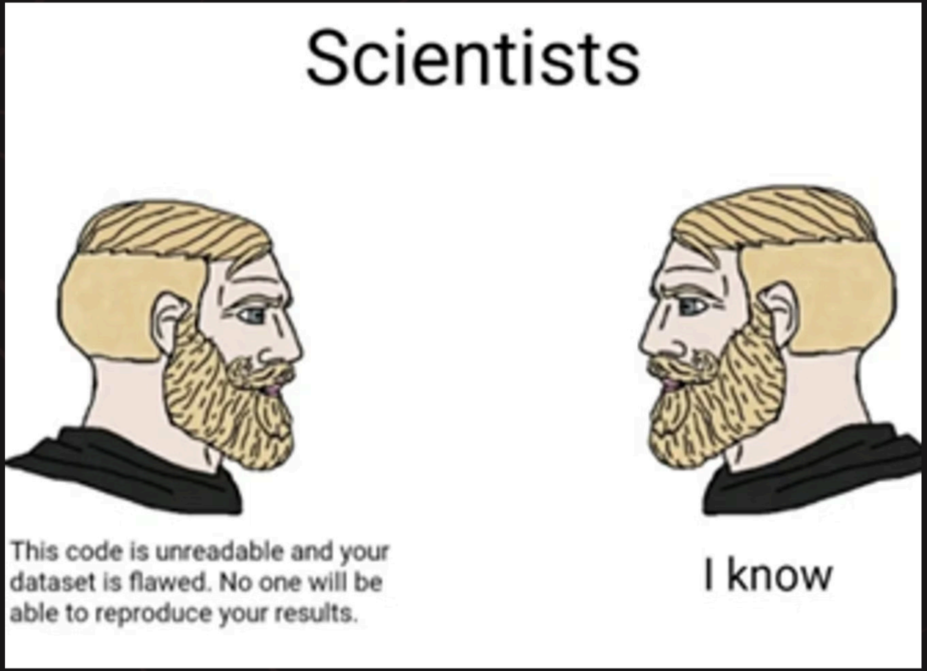


```
# Import a base image so we don't have to start from scratch
#FROM python:3.10-slim
FROM huggingface/transformers-pytorch-cpu

# Use EXPOSE so we can give docker run the appropriate commandline argument (PORT) as:
# docker run predict:latest -e PORT=8000
EXPOSE $PORT
ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8

# Run a bunch of linux commands
RUN apt update && \
    apt install --no-install-recommends -y build essential gcc & \
    apt clean & rm -rf /var/lib/apt/lists/*

# Copy the essential files from our folder to docker container.
COPY src/ src/
COPY requirements_predict.txt requirements_predict.txt
COPY setup.py setup.py

RUN pip install -r requirements_predict.txt --no-cache-dir

RUN dvc init --no-scm
RUN dvc remote add -d gcloud_storage gs://mlops-dataset-small
RUN dvc pull

# Set working directory as / and install dependencies
WORKDIR /

RUN mkdir app

# Set entry point, i.e. which file we run with which argument when running the docker container.
# The -u flag makes it print to console rather than the docker log file.
#ENTRYPOINT ["python", "-u", "src/models/predict_model.py"]
CMD exec uvicorn src.models.predict_model:app --host 0.0.0.0 --workers 1 --port $PORT
#ENTRYPOINT ["uvicorn", "src.models.predict_model:app", "--host", "0.0.0.0", "--workers", "1", "--port", $PORT]
```

Meme of the day

https://skaftenicki.github.io/dtu_mlops/s3_reproducibility/


Programmers
This code is unreadable and your dataset is flawed! No one will be able to reproduce your results!
It's not my fault the legacy environment is messed up! We still have 97.3% unit test coverage!


Scientists
This code is unreadable and your dataset is flawed. No one will be able to reproduce your results.
I know