7. Publishing, interoperability and responsible AI (RAI)

Software publication

What is this?

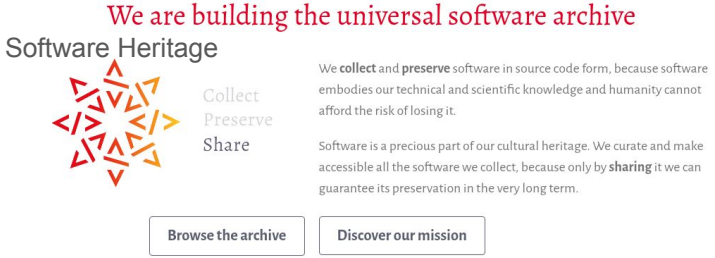
A software publication associates a DOI or other persistent identifier with your software.

Why is this important?

- This makes your software citable in scientific publications.
- You can reference your software plus providing the version number that you used.
- Your contribution to the scientific community becomes more visible.
- The publication helps you to structure the code and documentation for a user’s perspective.

Software publication in practice

- A list of software-specific journals: https://www.software.ac.uk/top-tip/which-journals-should-i-publish-my-software
- Write a short paper detailing your software, the architecture and design, in- and output, and purpose of the software
- Another option is to place a snapshot of your repo on zenodo, this also allows you to obtain a DOI
- Another option is to upload your project to Software Heritage and obtain a persistent SWHID https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html



Data publication

What is this?

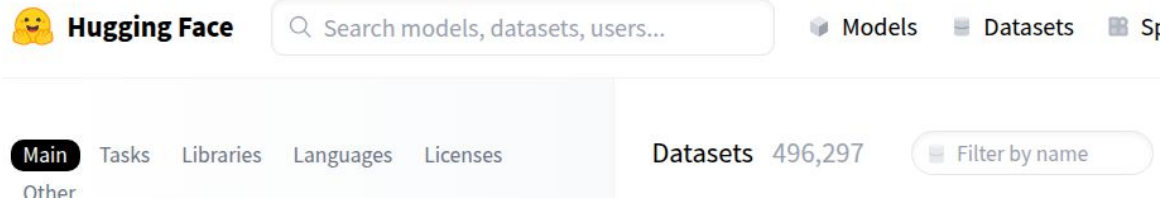
Data can be published in dedicated data repositories and obtain a DOI.

Why is this important?

- You can publish your datasets, making them available to the community.
- By associating a DOI, you may garner citations.
- Data often is the foundation for reproducing results.
- Data should be stored and preserved separately from the software once it reached a stage that it does not change anymore.

Data publication in practice

- Option A: Publish your data on zenodo and obtain a DOI
- Option B: Publish your data in your institutionally provided data repository, ie heiDATA and obtain a DOI
- Option C: Publish your data on community platforms such Hugging Face *if these provide a DOI*
- Make sure your data is clean and does not need to be updated anymore
- Cite your own data repository in your publications
- You can also download your own data dynamically within your code, ie,. using pooch https://www.fatiando.org/pooch/dev/ or datasets https://huggingface.co/docs/datasets/en/index



Why share your data?

- Create a larger impact of your scientific work
- Increase transparency and trust in your work
- Enable others to reproduce and build upon you
- Contribute to science as a whole
- BUT: Be careful with legal (copyright) and ethic advisor/institution.
- Follow the FAIR principle: Findable, Accessible

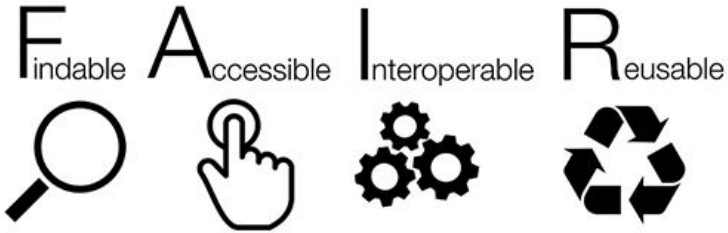


Image credit: SangyaPundir shared under CC-BY-SA 4.0
The FAIR Guiding Principles for scientific data management and stewardship
https://doi.org/10.1038/sdata.2016.18

Where to share your data

- HeiDATA
- Open Science Framework
- Zenodo
- Figshare
- Dryad
- Hugging face
- Kaggle: no DOI!
- Domain-specific repositories
  - i.e. https://lindat.mff.cuni.cz/repository/xmlui/
  - see https://www.nature.com/sdata/policies/repositories for a list
- For a comparison, see DOI: 10.5281/zenodo.3946720

Model sharing platforms

You can make models available for others on model sharing platforms like

- Hugging face,
- OpenML,
- Kaggle.

Advantages: Public platform with version control and model cards, you can link the data into the repo, allows others to use your model for production or fine-tuning.

Model card

- Model details
  - Architecture, parameters, citation information, license information
- Intended use
  - Use cases within the model's scope
- Performance metrics
  - Intended performance on given data
- Training data
  - Description of training data and data distribution
- Quantitative analysis
  - Potential biases and limitations
- Ethical consideration
  - Privacy and fairness concerns, impact on society
- https://huggingface.co/spaces/huggingface/Model_Cards_Writing_Tool
- https://github.com/openai/gpt-3/blob/master/model-card.md

Model deployment

In addition to making models and software available for others to use in their own code, you can also directly deploy the model - together with your code - directly so that it can be used.

Examples:

- Diffusers: google colab https://colab.research.google.com/github/huggingface/notebooks/blob/main/diffusers/stable_diffusion.ipynb
- https://lightning.ai/ for paid service and deployable models
- See https://www.freecodecamp.org/news/deploy-your-machine-learning-models-for-free/ for tutorials and services

REFORMS checklist

To ensure reproducibility, include a checklist such as the REFORMS checklist into the publication of your results.

There are a couple of domain-specific checklists, especially in health and life sciences (STARD, CLAIM, see the EQUATOR* network), that may be more appropriate - REFORMS aims to be most general.

REFORMS checklist: DOI 10.48550/arXiv.2308.07832, https://reforms.cs.princeton.edu/
*EQUATOR Network, https://www.equator-network.org/reporting-guidelines/

Interoperability and RAI

Interoperability

- Ability to interact with other systems interchangeably (i.e. transformers, pytorch)
- Exchange information with other systems
- Integrate into other systems

⇩

Use data and models in different contexts

RAI

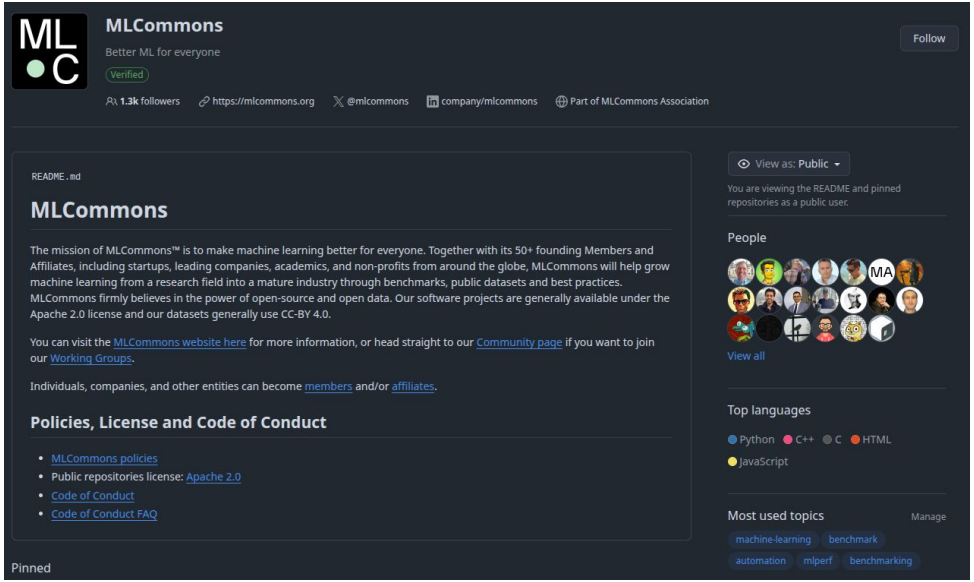
- Develop, assess, deploy AI systems safe and ethically
- Prioritize equitable and beneficial outcome of the AI system in the system development
- people and their goals at the center of design

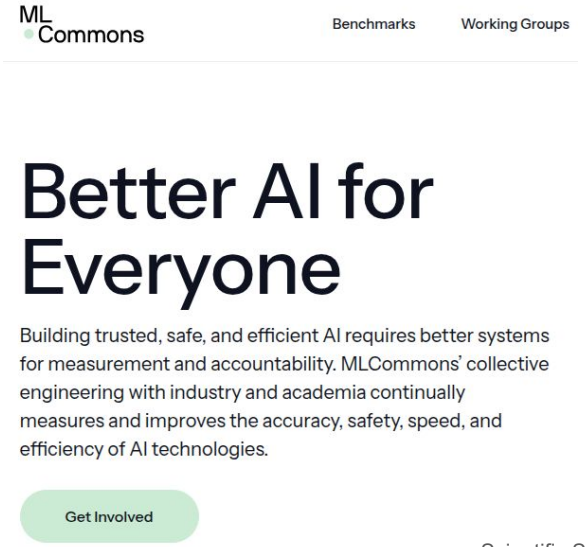
⇩

fairness, reliability, transparency, safety, privacy, security, inclusiveness, accountability

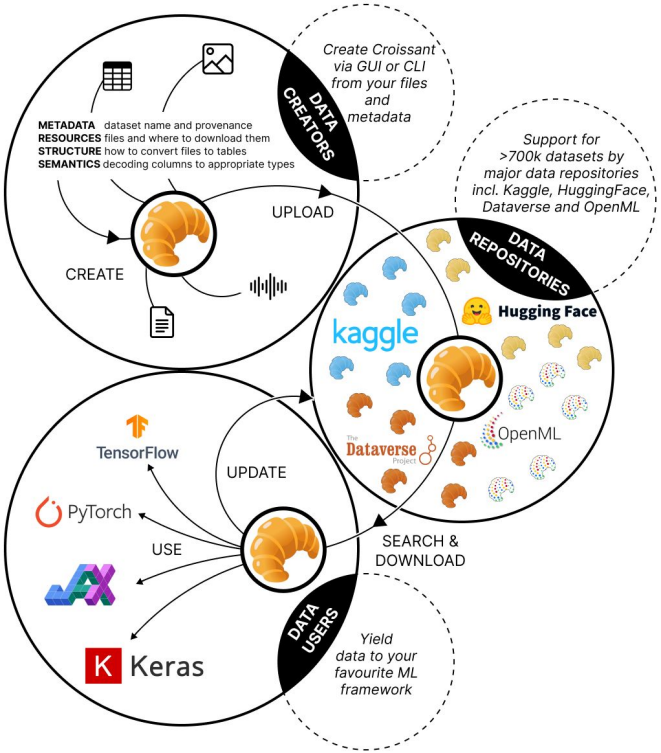
MLCommons

- To accelerate artificial intelligence innovation and increase its positive impact on society
- Targets industry and academia
- Democratize machine learning through open industry-standard benchmarks that measure quality and performance and build open, large-scale, and diverse datasets to improve AI models





Croissant: an interoperable data format



Croissant 🥐 is a high-level format for machine learning datasets

Metadata: standardized description of the dataset, including responsible ML aspects

Resources: one or more files or other sources containing the raw data

Structure: how the raw data is combined and arranged into data structures for use

ML semantics: how the data is most often used in an ML context

Find, inspect, and use the data in your favorite ML framework!

https://github.com/mlcommons/croissant
https://doi.org/10.52202/079017-2610

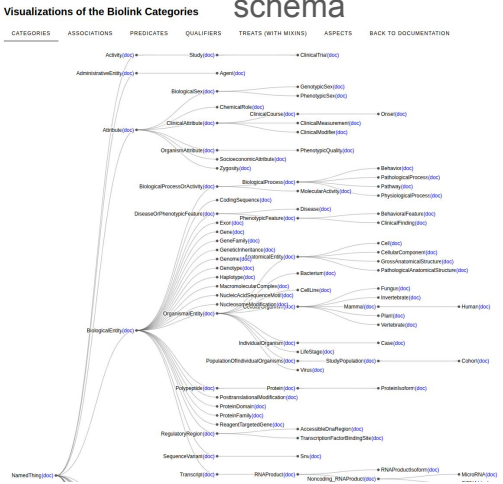
Croissant file

```
"@type": "sc:Dataset",
"dct:conformsTo": "http://mlcommons.org/croissant/1.0",
"name": "Collectri",
"description": "CollecTRI dataset",
"version": "0.0.1",
"license": "https://github.com/saezlab/CollecTRI/blob/main/LICENSE",
"url": "https://github.com/saezlab/CollecTRI",
"keywords": [
  "multi-omics",
  "regulatory interactions"
],
"datePublished": "2024-05-12",
"creator": [
  {
    "@type": "sc:Person",
    "name": "Sebastian Lobentanzer",
    "affiliation": "Helmholtz Munich",
    "identifier": "https://orcid.org/0000-0003-3399-6695"
  }
],
"citeAs": "This adapter was developed by Sebastian Lobentanzer to load and transform the Collectri dataset. For dataset c
"conformsTo": "http://mlcommons.org/croissant/1.0",
"distribution": [
  {
    "@type": "cr:FileObject",
    "@id": "collectri.csv",
    "name": "CollecTRI.csv",
    "contentUrl": "https://rescued.omnipathdb.org/CollecTRI.csv",
    "encodingFormat": "text/csv",
    "sha256": "86c90b30f2cc75c189da1f0a8c353d1547287cd656a9fac1c678634285bcb4e0"
  }
],
"recordSet": [
  {
```

Metadata layer (name, description, license)

Resources layer (source data included in dataset)

an ontology for a schema

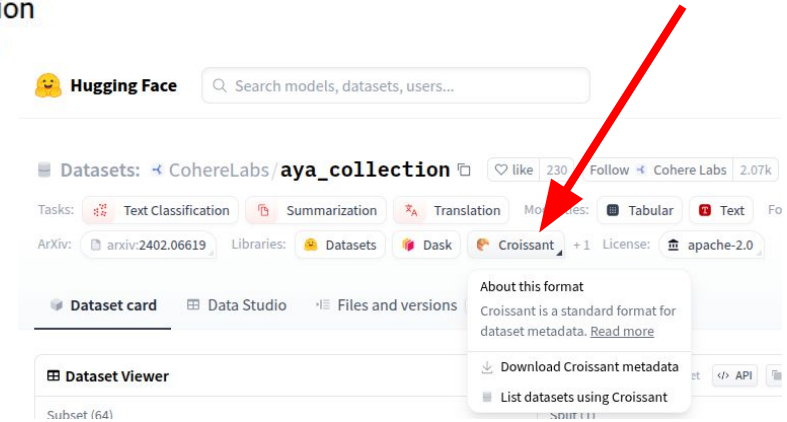


Croissant file


```
"recordSet": [
  {
    "@type": "cr:RecordSet",
    "@id": "collectri",
    "name": "CollecTRI",
    "description": "Transcription factor → target gene regulatory interactions with supporting evidence.",
    "field": [
      {
        "@type": "cr:Field",
        "@id": "collectri/source",
        "name": "source",
        "description": "Gene symbol of transcription factor (TF).",
        "dataType": "sc:Text",
        "examples": [
          "MYC"
        ],
        "source": {
          "fileObject": {
            "@id": "collectri.csv"
          },
          "extract": {
            "column": "source"
          }
        }
      },
      {
        "@type": "cr:Field",
        "@id": "collectri/target",
        "name": "target",
        "description": "Gene symbol of the regulated target gene.",
        "dataType": "sc:Text",
        "examples": [
          "TERT"
        ],
        "source": {
          "fileObject": {
            "@id": "collectri.csv"
          },
          "extract": {
            "column": "target"
          }
        }
      },
```


Structure Layer (data as RecordSets, data mapping and transformations)

Semantic layer (custom data types, dataset organization methods)



Work with croissant datasets

Loading:

```python
import mlcroissant as mlc

# load the collectri dataset
collectri_dataset = mlc.Dataset("../biocypher-components-registry/external_repos/adapter_collectri/collectri.json")
```

Using as a base for training:

```python
import tensorflow_datasets as tfds
import torch
from tqdm import tqdm

# Use Croissant dataset in your ML workload
builder = tfds.core.dataset_builders.CroissantBuilder(
    jsonld="../biocypher-components-registry/external_repos/adapter_collectri/collectri.json",
    record_set_ids=["collectri"],
    file_format='array_record',
)
builder.download_and_prepare()

# Split for training/testing and use as_dataset directly from the builder
train = builder.as_data_source(split='default[:80%]')
test = builder.as_data_source(split='default[20%:]')

ds = builder.as_data_source()
```

```python
# now load into pytorch
batch_size = 128
train_sampler = torch.utils.data.RandomSampler(train, num_samples=5_000)
train_loader = torch.utils.data.DataLoader(
    train,
    sampler=train_sampler,
    batch_size=batch_size,
)
test_loader = torch.utils.data.DataLoader(
    test,
    sampler=None,
    batch_size=batch_size,
)

class LinearClassifier(torch.nn.Module):
  def __init__(self, length, num_classes):
    super(LinearClassifier, self).__init__()
    self.classifier = torch.nn.Linear(length, num_classes)

  def forward(self, feature_x):
    # Convert to float32 if needed
    feature_x = feature_x.to(torch.float32)
    # Pass through the linear layer
    return self.classifier(feature_x)

# Create label mapping
print('Creating label mapping...')
label_mapping = {}
for example in train_loader:
  for label in example['sign_decision']:
    if label not in label_mapping:
      label_mapping[label] = len(label_mapping)
print(f"Label mapping: {label_mapping}")

# Update model with correct number of classes
num_classes = len(label_mapping)
model = LinearClassifier(length=1, num_classes=num_classes)
optimizer = torch.optim.Adam(model.parameters())
loss_function = torch.nn.CrossEntropyLoss()

print('Training...')
model.train()
```

- use tfds builder for loading
- builder compatible with tensorflow, pytorch, JAX
- eclair MCP server to use croissant with an AI agent
- Very active and open group of

Contributors

Albert Villanova (Hugging Face), Andrew Zaldivar (Google), Baishan Guo (Meta), Carole Jean-Wu (Meta), Ce Zhang (ETH Zurich), Costanza Conforti (Google), D. Sculley (Kaggle), Dan Brickley (Schema.Org), Eduardo Arino de la Rubia (Meta), Edward Lockhart (Deepmind), Elena Simperl (King's College London), Goeff Thomas (Kaggle), Joan Giner-Miguelez (UOC), Joaquin Vanschoren (TU/Eindhoven, OpenML), Jos van der Velde (TU/Eindhoven, OpenML), Julien Chaumond (Hugging Face), Kurt Bollacker (MLCommons), Lora Aroyo (Google), Luis Oala (Dotphoton), Meg Risdal (Kaggle), Natasha Noy (Google), Newsha Ardalani (Meta), Omar Benjelloun (Google), Peter Mattson (MLCommons), Pierre Marcenac (Google), Pierre Ruyssen (Google), Pieter Gijsbers (TU/Eindhoven, OpenML), Prabhant Singh (TU/Eindhoven, OpenML), Quentin Lhoest (Hugging Face), Steffen Vogler (Bayer), Taniya Das (TU/Eindhoven, OpenML), Michael Kuchnik (Meta)

Thank you for supporting Croissant! 🙂

RAI

Croissant RAI specification https://docs.mlcommons.org/croissant/docs/croissant-rai-spec.html

Use case 1: The data life cycle (level: dataset)

Key stages of the dataset life cycle include **motivation, composition, collection process, preprocessing/cleaning/labeling, uses, distribution, and maintenance** [5]. Documenting RAI-related properties of the dataset can encourage its creators to reflect on the process and improve understanding for users.

Information generated throughout the cycle addresses different aspects requiring consideration for responsible data usage, including (1) information regarding who created the dataset for which purpose, (2) information when the dataset was created, (3) which data sources were used, (4) information on the versioning of the dataset with timestamps for each version (5) how the data is composed and if it contains noise, redundancies, privacy-critical information, etc. (6) how data was processed (e.g. also containing information on crowdsourcing - see use case 2), (7) how the data is intended to be used, (8) how the dataset will be maintained. In conjunction, properties for documenting the provenance and lineage of the datasets that are derived from revision, modification or extension of existing datasets are also relevant for this use case.

We anticipate that this use case will be covered by the core vocabulary. The main purpose of the use case is to understand if there are any additional properties currently not included in Croissant.

Use case 2: Data labeling (level: dataset or record)

A portion of the metadata at dataset level will be aggregated from record-level annotations. These can be achieved either through some form of human input, in particular labels and annotations created via labeling services, including, but not restricted to crowdsourcing platforms (e.g. what platform has been used), how many human labels were extracted per record, any demographics about the raters if available, or by machine annotations (e.g. concept extraction, NER, and additional characteristics of the tools used for annotation to allow for replication or extension). These are important to automatically create distribution characteristics of datasets to better understand its composition, but also to be able to efficiently sample from different datasets. Information about the labeling process helps **understand how the data was created, the sample the labels apply to,** hence making the process easier to assess, repeat, replicate, and reproduce. This increases the reliability of the resulting data.

Software security

- Threat modelling: who, what, how
- Data-oriented attack: Access training data, poison data, inject trojan data
- Model-oriented attack: Modify training process (pre-trained malicious models), manipulate the deployed model (model patches, privacy information leakage, model inversion attacks)
- System-oriented attack: specialised hardware accelerators for ML software (SOC, trojan in GPU/TPU, for model corruption, backdoor insertion, model extraction, spoofing, information extraction, sybil attack)
- Possibility to carry out *pentests*

Security: best practices

Phases	Vulnerabilities Causes
Analysis phase	No risk analysis/ No security policy
	Biased risk analysis
	Unanticipated risks
Design phase	Relying on non-secure abstractions
	Security/Convenience tradeoff
	No logging
	Design does not capture all risks
Implementation phase	Insufficiently defensive input checking
	Non-atomic check and use
	Access validation errors
	Incorrect crypto primitive implementation
	Insecure handling of exceptional conditions
	Bugs in security logic
Deployment phase	Reuse in more hostile environments
	Complex or unnecessary configuration
	Insecure defaults
Maintenance phase	Feature interaction
	Insecure fallback

Chen, Barbar, Security for Machine Learning-based Software Systems, DOI 10.1145/3638531

Legal aspects

Legal aspects

- If you reuse data / models / code: Make sure the license terms allow this and that your license(s) is (are) compatible
- Make sure you do not violate the DSGVO / GDPR / European Data Act / Copyright / European AI Act
- Once a model is made available, it is impossible to restrict its use!
- Examples:
  - Models can put out near-exact copies of images/text in training data, ie Dall-E/Stable Diffusion generating images with Shutterstock/Getty Images watermarks, or reproducing artist's work
  - Code-generating tools such as GitHub copilot allow recreation of code, that is already contained in other software, regardless of the license terms of that software

Ethical aspects

Ethical aspects

- Be aware of people's tendency of overreliance!
  - ELIZA effect: the tendency of users to project human traits onto interactive software
  - Trusting into predictions above one's own assessment
- Misuse of AI by bad actors
  - Face recognition used to detect Uyghur population (China), Clearview used to track people's movement and employment status by police officers (even though use was prohibited)
  - Facial analysis used to track people's attention on billboards, change advertisement based on their demographic
- Source of truth during training
  - Data that foundation models are trained on does also contain false statements, ie law professor incorrectly accused of sexual harassment
  - Data in foundation models contains toxicity of the internet; human labor is used to label / annotate toxic text and images ie subcontracting workers in Kenya (OpenAI)

Infamous AI mistakes

AI in general

- **Automatization at amazon:** Experimental hiring tool, developed by a team of five, used artificial intelligence to give job candidates scores ranging from one to five stars
- Tool was trained on all resumes of the last 10 years
- Tool preferably suggested male candidates, penalyzing resumes that contained the word “women’s” or graduates from all-female colleges
- ***Why?***

https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G/ (2018)

Automated resume selection at amazon

- The dataset consisted of the resumes of the last 10 years: Predominantly male applicants
- Women are underrepresented in tech:

  https://fingfx.thomsonreuters.com/gfx/rngs/AMAZON.COM-JOBS-AUTOMATION/010080Q91F6/index.html
- Thus, the model learnt it was more often correct if it suggested male candidates: Unbalanced representation of the dataset

AI in general

- U.S. healthcare system uses commercial algorithms to guide health decisions
- Algorithm (Optum's Impact Pro) to target patients for "high-risk care management" programs
- Identify patients who benefit the most: https://www.science.org/doi/10.1126/science.aax2342
- Model is trained on healthcare spendings to determine the healthcare need
- Algorithm was much more likely to recommend white patients for these programs than black patients, even though the black patients were evidently sicker
- ***Why?***

Bias in healthcare need estimation

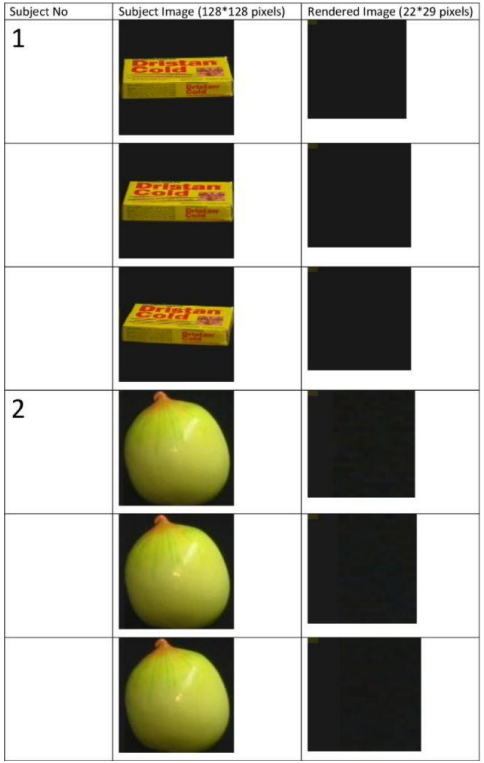
- Training based on healthcare spendings: But people of color are more likely to have lower incomes - making them less likely to access medical care even if they are insured
- Also, they may experience higher barriers to accessing health care (geography, transportation, work/childcare constraints), in addition to direct doctor-patient bias
- Data shows that race is correlated with substantial differences in health-care spendings: This results in a bias of the trained model

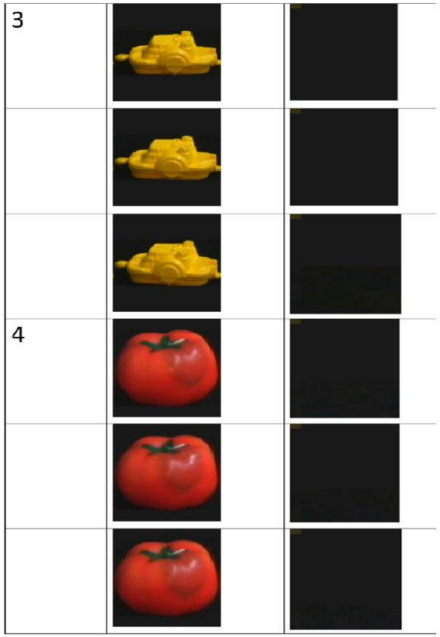
AI mistakes in research software

- Applying machine learning methods to COVID-19 radiological imaging for improving the accuracy of diagnosis
- Distinguish patients with COVID-19 from patients without COVID-19 but also bacterial pneumonia
- Tools were trained on public datasets with CT and CXR images
- Predictive tools failed practical tests: https://www.nature.com/articles/s42256-021-00307-0
- ***Why?***

Learning from the image background

- Image dataset was collected under controlled conditions: Does not represent the target distribution of interest
- CNN learnt to distinguish the image background rather than the image content
- Actually quite prevalent problem in computer vision





COIL-100 dataset https://www1.cs.columbia.edu/CAVE/publications/pdfs/Nene_TR96_2.pdf
https://doi.org/10.1016/j.visinf.2021.10.001

AI mistakes in research software

- Predict whether a country is likely to slide into civil war based on GDP, poverty rates, type of government structure, etc.
- Complex models using Random Forests and Adaboost outperform more standard statistical approaches like logistic regression by far
- Missing values in the dataset were constructed using imputation on the complete dataset
- Models proved to be over-optimistic and erroneous https://doi.org/10.1016/j.patter.2023.100804
- ***Why?***

Civil war predictions: Data leakage

- Data leakage: The data was imputed for missing values using the whole dataset
- Thus, the training dataset contained information about the test dataset
- This leads to an inflated estimate of the model performance

https://doi.org/10.1016/j.patter.2023.100804 supplemental material

Classification of failures/errors

Data leakage

Spurious relationship between independent variables and target variable

Artifact of collection, sampling, pre-processing

Leads to inflated estimates of model performance

Lack of clean separation training/test

- no test set
- pre-processing on training and test set (over/under sampling, imputation)
- feature selection on entire dataset
- duplicates in dataset

Kapoor, Narayanan, DOI: 10.1016/j.patter.2023.100804

Data leakage

Model uses features that are not legitimate

- for example, use of a certain drug when predicting illness (hypertensive drug, antibiotics)

Test set is not drawn from distribution of scientific interest

- temporal leakage (test set must not contain data from before the training set)
- non-independence between training and test samples (same people/units in both sets - use block cross-validation)
- sampling bias in test distribution (spatial bias, age, image settings)

Kapoor, Narayanan, DOI: 10.1016/j.patter.2023.100804

Resources

- Good practices in machine learning (Mathieu Bauchy) (https://www.youtube.com/watch?v=WScUQnU-ozQ&t=3213s
- Roadmaps https://roadmap.sh/roadmaps
- Kaggle https://www.kaggle.com/learn
- Hugging Face https://huggingface.co/learn
- REFORMS checklist https://reforms.cs.princeton.edu/
- MLCommons https://mlcommons.org/
- Scikit-learn resources https://scikit-learn.org/stable/common_pitfalls.html
- Testing of non-deterministic software https://bssw.io/blog_posts/testing-non-deterministic-research-software