MEDS ML4H 2025 Tutorial

Matthew McDermott
Assistant Professor, Columbia University Department of Biomedical Informatics

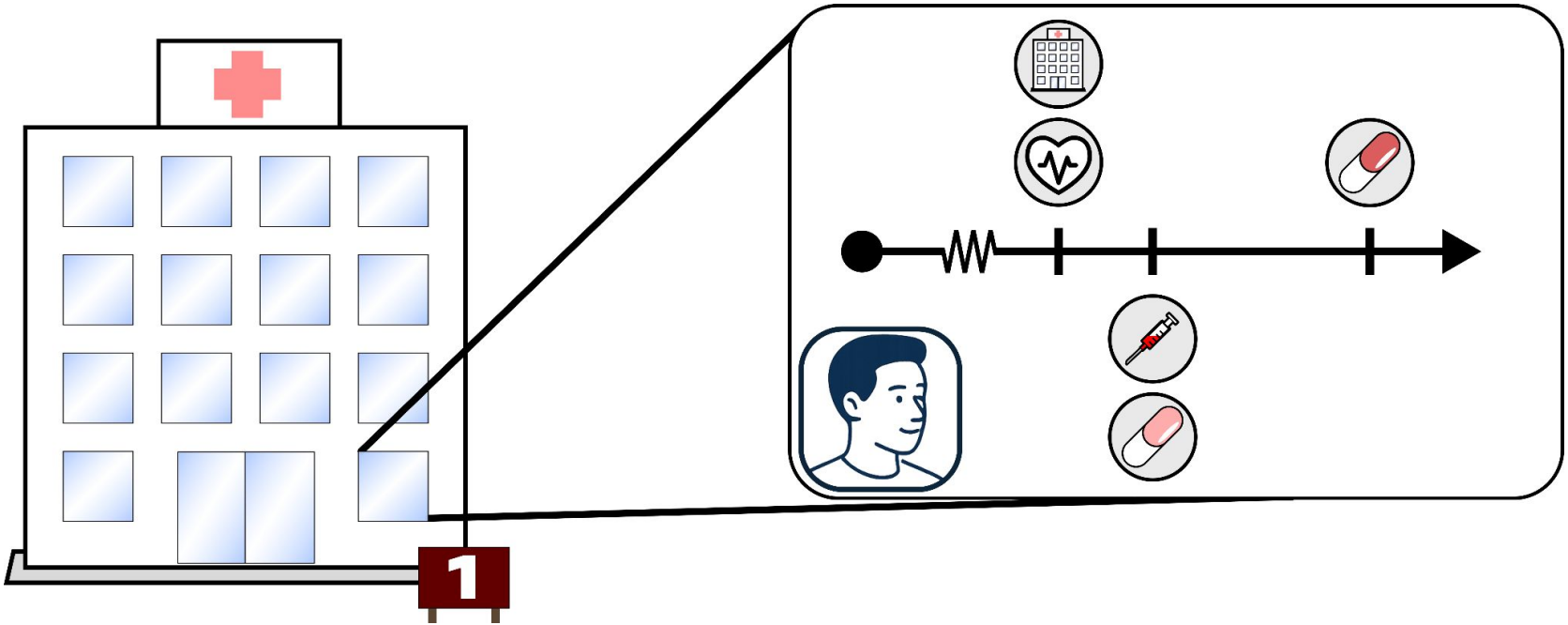
Preparation: Suhana Bedi, Teya Bergamaschi, Hyewon Jeong, Simon Lee, Nassim Oufattole, Patrick Rockenschaub, Kamilė Stankevičiūtė, Ethan Steinberg, Jimeng Sun, Robin van de Water, Michael Wornow, John Wu, Zhenbang Wu, and Justin Xu

MEDS

mattmcdermott8@gmail.com

https://medical-event-data-standard.github.io/

Area: AI for Structured Electronic Health Record (EHR) Data



Learning Goals

1. Understand the core concepts and design principles of the MEDS ecosystem
    a. *Apply:* Be able to transform data into the MEDS format.
    b. *Analyze:* Be able to break down a problem into different parts that can leverage different tools.
2. Understand how to build models over MEDS data
    a. *Understand:* Identify how to map data into the necessary formats for effective ML/AI
3. Identify how to participate in the MEDS community and promote reproducible research
    a. *Apply:* Use existing models via MEDS-DEV to compare to the state-of-the-art

Outline

	✓	*9:15 - 9:20*	Opening remarks
CO	1.	*9:20 - 9:40*	What is MEDS?
	2.	*9:40 - 9:45*	Guiding Problem Setup
CO	3.	*9:45 - 10:05*	Convert your data into MEDS
CO	4.	*10:05 - 10:10*	Extract a prediction task cohort with ACES
CO	5.	*10:10 - 10:15*	Develop a predictive model 1: Tabular Baseline
CO	6.	*10:15 - 10:35*	Develop a predictive model 2: Neural Network
	7.	*10:35 - 10:40*	Closing Remarks

www.tinyurl.com/
MEDS-ML4H-pres

Shared Disclaimers

Disclaimer S1: MEDS is a “data-as-interface” ecosystem. This means that, in general, rather than operating through an interface of python objects, MEDS tools operate through an interface of data in a format. This makes the ecosystem very open, enabling many ways of doing things and many tools that could be used for the same job, not just the tools we show here.

Disclaimer S2: Most MEDS tools have been designed by academic volunteers, and it is more than plausible that there are better, faster, or more efficient ways to do things. If you think you can find one -- great! Put it out there in the community, and we’ll gladly use it.

Part 1: What is MEDS and why use it?

Driving Problem: Health AI has a Reproducibility Crisis

We reproduced datasets for 38 experiments corresponding to 28 published studies using MIMIC. In half of the experiments, the sample size we acquired was 25% greater or smaller than the sample size reported.

- Alistair Johnson et. al., 2017
https://proceedings.mlr.press/v68/johnson17a.html

MLH papers scored even more poorly when it came to code release... with only ~21% of the papers we analyzed releasing their code publicly

- Matthew McDermott et. al., 2021
https://doi.org/10.1126/scitranslmed.abb1655

Of the 218 included articles, 73 (34%) shared code, with 24 (33% of code sharing articles and 11% of all articles) sharing reproducible code

- Kesavan Venkatesh et. al., 2022
https://doi.org/10.1148/ryai.220081

The (lack of?) Science of Machine Learning for Healthcare.

McDermott, M. (2025). *Proceedings of the 4th Machine Learning for Health Symposium*, in *Proceedings of Machine Learning Research* 259:19-29

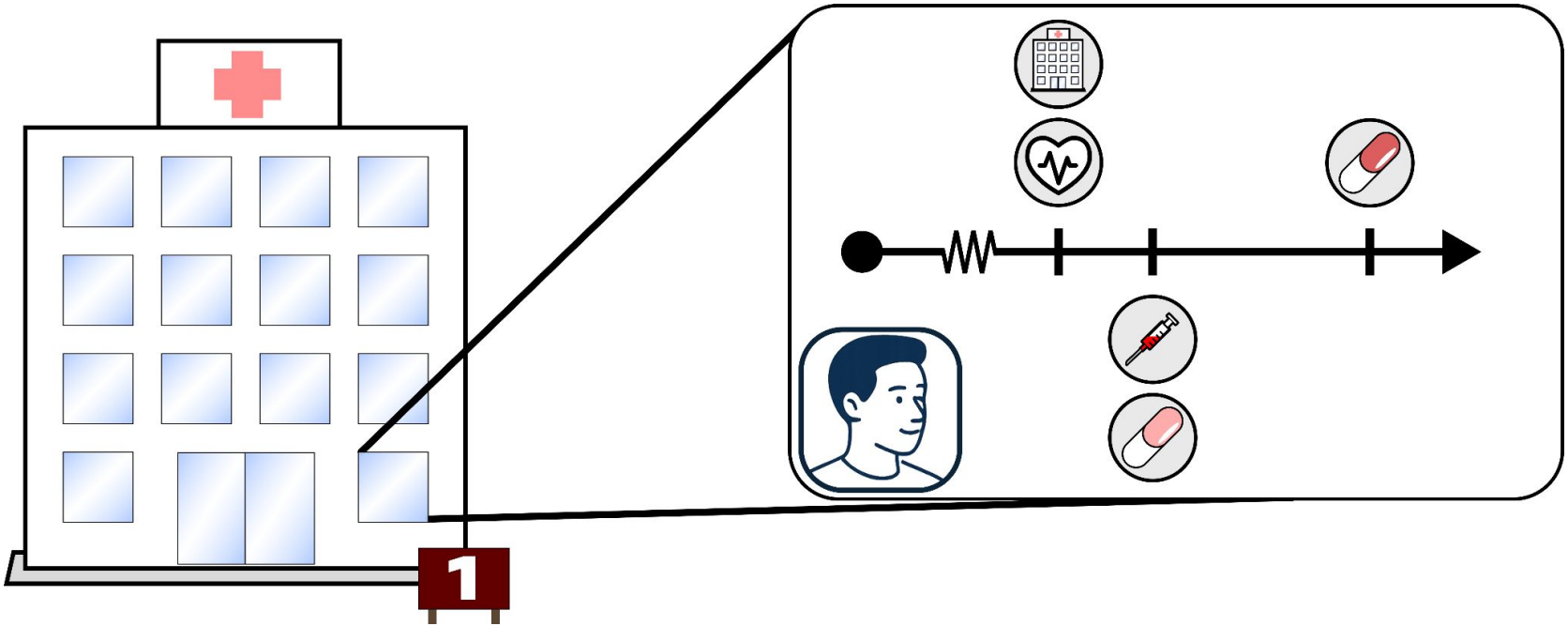
Available from https://proceedings.mlr.press/v259/mcdermott25a.html

To solve this reproducibility crisis, we need a data standard designed for health AI.

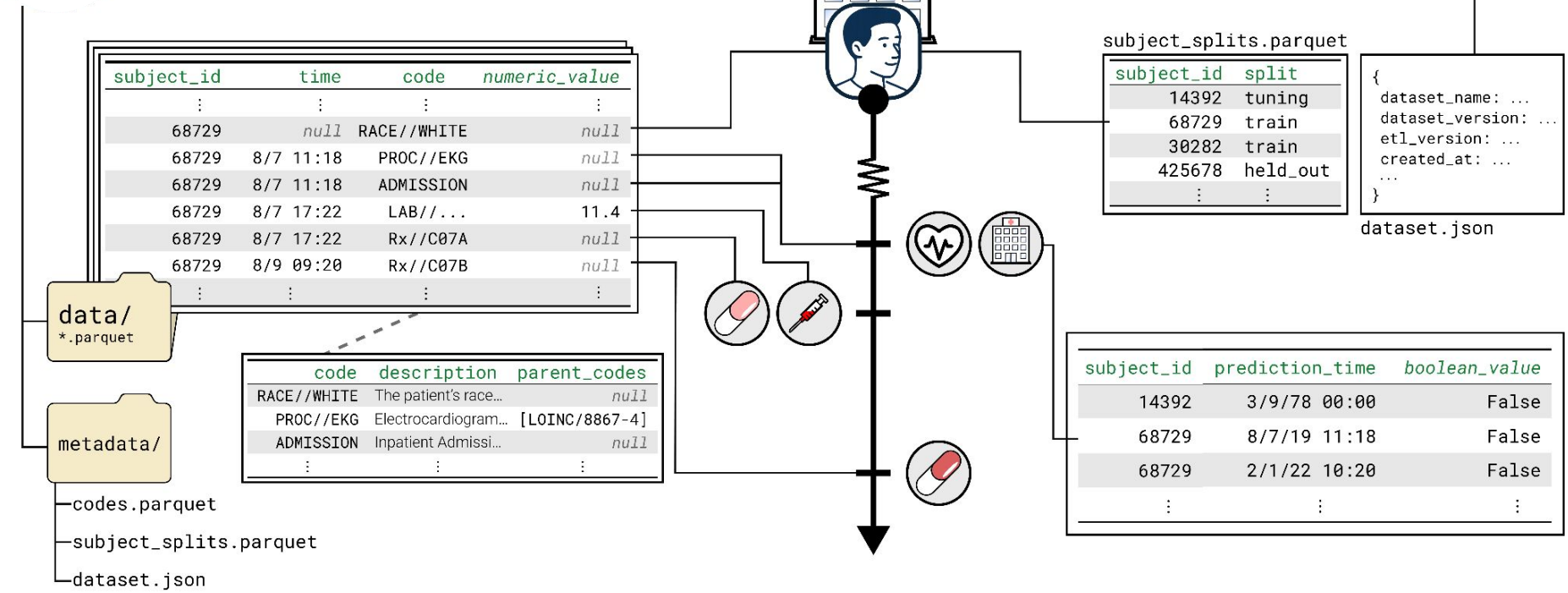
Design Principles for a Standard for Health AI

1. Transportability of model training algorithms
   The fundamental goal of MEDS is to enable *frictionless* transportability of algorithms first and foremost.
2. Capture the simplest view of the fundamental structure of health data
   MEDS asserts that EHR data can be most simply and fundamentally represented by capturing the longitudinal timeline of patient observations. Emphasizing simplicity in this way empowers and simplifies downstream use.
3. Empower development of an open-source ecosystem for health AI
   MEDS must enable development of an ecosystem of models, tools, & datasets.
4. Support Computational Performance for Foundation Models
   MEDS should support foundation-model scale systems.

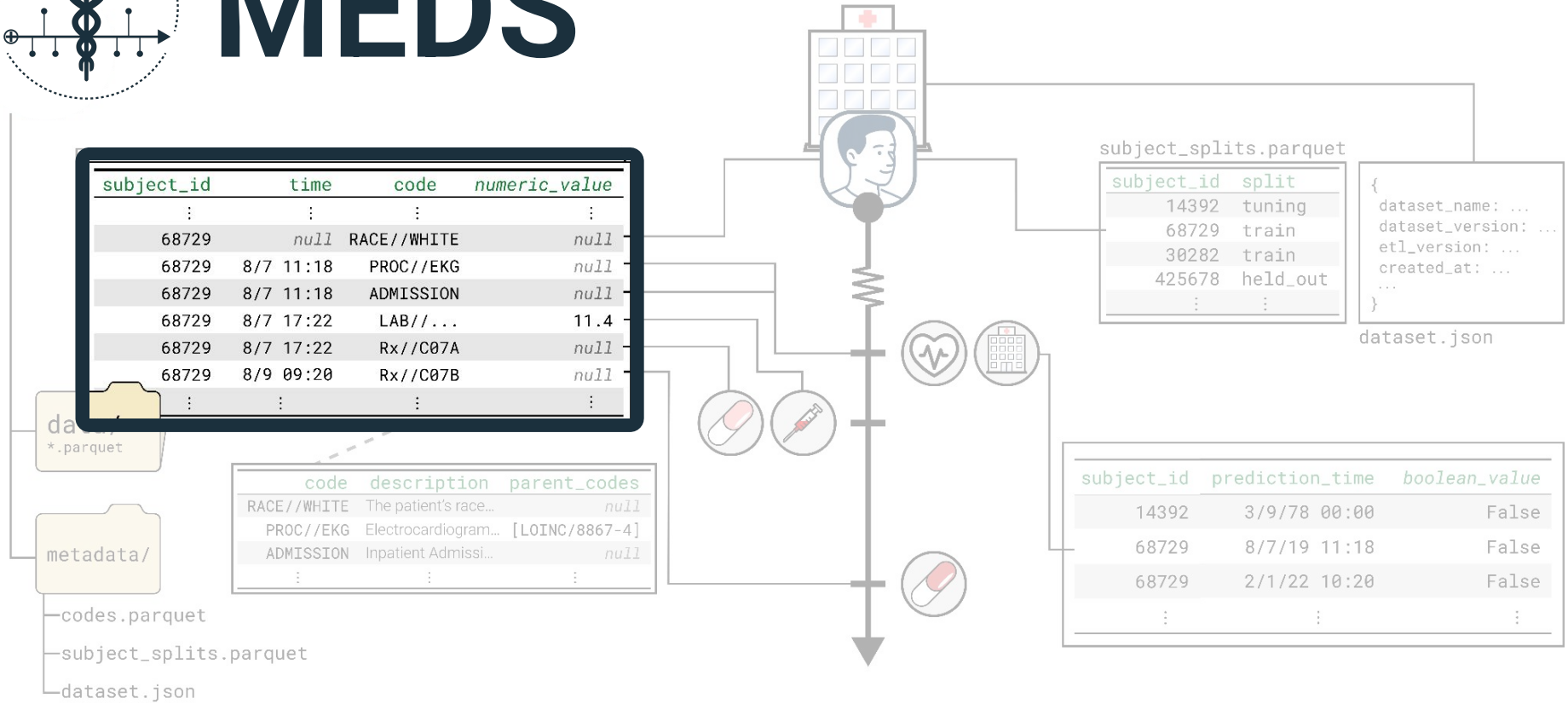
A Patient's Data can be represented as a **Timeline**



MEDS


subject_id time code numeric_value
68729 null RACE//WHITE null
68729 8/7 11:18 PROC//EKG null
68729 8/7 11:18 ADMISSION null
68729 8/7 17:22 LAB//... 11.4
68729 8/7 17:22 Rx//C07A null
68729 8/9 09:20 Rx//C07B null
data/
*.parquet
code description parent_codes
RACE//WHITE The patient's race... null
PROC//EKG Electrocardiogram... [LOINC/8867-4]
ADMISSION Inpatient Admissi... null
metadata/
codes.parquet
subject_splits.parquet
dataset.json
subject_splits.parquet
subject_id split
14392 tuning
68729 train
30282 train
425678 held_out
{
dataset_name: ...
dataset_version: ...
etl_version: ...
created_at: ...
...
}
dataset.json
subject_id prediction_time boolean_value
14392 3/9/78 00:00 False
68729 8/7/19 11:18 False
68729 2/1/22 10:20 False

MEDS


subject_id time code numeric_value
68729 null RACE//WHITE null
68729 8/7 11:18 PROC//EKG null
68729 8/7 11:18 ADMISSION null
68729 8/7 17:22 LAB//... 11.4
68729 8/7 17:22 Rx//C07A null
68729 8/9 09:20 Rx//C07B null
*.parquet
code description parent_codes
RACE//WHITE The patient's race... null
PROC//EKG Electrocardiogram... [LOINC/8867-4]
ADMISSION Inpatient Admissi... null
metadata/
codes.parquet
subject_splits.parquet
dataset.json
subject_splits.parquet
subject_id split
14392 tuning
68729 train
30282 train
425678 held_out
{
dataset_name: ...
dataset_version: ...
etl_version: ...
created_at: ...
...
}
dataset.json
subject_id prediction_time boolean_value
14392 3/9/78 00:00 False
68729 8/7/19 11:18 False
68729 2/1/22 10:20 False

Data Shards

subject_id	time	code	numeric_value
⋮	⋮	⋮	⋮
68729	*null*	RACE//WHITE	*null*
68729	8/7 11:18	PROC//EKG	*null*
68729	8/7 11:18	ADMISSION	*null*
68729	8/7 17:22	LAB//...	11.4
68729	8/7 17:22	Rx//C07A	*null*
68729	8/9 09:20	Rx//C07B	*null*
⋮	⋮	⋮	⋮

- Each shard contains all data for a patient, in time order.

Data Shards

subject_id	time	code	*numeric_value*
⋮	⋮	⋮	⋮
68729	*null*	RACE//WHITE	*null*
68729	8/7 11:18	PROC//EKG	*null*
68729	8/7 11:18	ADMISSION	*null*
68729	8/7 17:22	LAB//...	11.4
68729	8/7 17:22	Rx//C07A	*null*
68729	8/9 09:20	Rx//C07B	*null*
⋮	⋮	⋮	⋮

- Each shard contains all data for a patient, in time order.
- The code column is *unconstrained*.

Data Shards

subject_id	time	code	*numeric_value*
⋮	⋮	⋮	⋮
68729	*null*	RACE//WHITE	*null*
68729	8/7 11:18	PROC//EKG	*null*
68729	8/7 11:18	ADMISSION	*null*
68729	8/7 17:22	LAB//...	11.4
68729	8/7 17:22	Rx//C07A	*null*
68729	8/9 09:20	Rx//C07B	*null*
⋮	⋮	⋮	⋮

- Each shard contains all data for a patient, in time order.
- The code column is *unconstrained*.
- Additional columns can be included as needed.

MEDS data properties and examples

sujbect_id	time	code	numeric_value
68729		EYE_COLOR//HAZEL	
68729		HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	MEDS_BIRTH	
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	
68729	05/26/2010, 2:30:56	HR	86
68729	05/26/2010, 2:30:56	TEMP	97.8
68729	05/26/2010, 4:51:52	DISCHARGE	
239684		EYE_COLOR//BROWN	
239684		HEIGHT	175.2711152
239684	12/28/1980, 0:00:00	MEDS_BIRTH	
239684	05/11/2010, 17:41:51	ADMISSION//CARDIAC	
239684	05/11/2010, 17:41:51	HR	102.6
239684	05/11/2010, 17:41:51	TEMP	96
239684	05/11/2010, 17:48:48	HR	105.1
239684	05/11/2010, 17:48:48	TEMP	96.2

Static observations have null timestamps

sujbect_id	time	code	numeric_value
68729		EYE_COLOR//HAZEL	
68729		HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	MEDS_BIRTH	
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	
68729	05/26/2010, 2:30:56	HR	86
68729	05/26/2010, 2:30:56	TEMP	97.8
68729	05/26/2010, 4:51:52	DISCHARGE	
239684		EYE_COLOR//BROWN	
239684		HEIGHT	175.2711152
239684	12/28/1980, 0:00:00	MEDS_BIRTH	
239684	05/11/2010, 17:41:51	ADMISSION//CARDIAC	
239684	05/11/2010, 17:41:51	HR	102.6
239684	05/11/2010, 17:41:51	TEMP	96
239684	05/11/2010, 17:48:48	HR	105.1
239684	05/11/2010, 17:48:48	TEMP	96.2

Static observations

Observations at the same time share timestamps

Timestamps will be duplicated!

sujbect_id	time	code	numeric_value
68729		EYE_COLOR//HAZEL	
68729		HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	MEDS_BIRTH	
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	
68729	05/26/2010, 2:30:56	HR	86
68729	05/26/2010, 2:30:56	TEMP	97.8
68729	05/26/2010, 4:51:52	DISCHARGE	
239684		EYE_COLOR//BROWN	
239684		HEIGHT	175.2711152
239684	12/28/1980, 0:00:00	MEDS_BIRTH	
239684	05/11/2010, 17:41:51	ADMISSION//CARDIAC	
239684	05/11/2010, 17:41:51	HR	102.6
239684	05/11/2010, 17:41:51	TEMP	96
239684	05/11/2010, 17:48:48	HR	105.1
239684	05/11/2010, 17:48:48	TEMP	96.2

MEDS mandates *standardization*, not *harmonization*

sujbect_id	time	code	numeric_value
68729		EYE_COLOR//HAZEL	
68729		HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	MEDS_BIRTH	
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	
68729	05/26/2010, 2:30:56	HR	86
68729	05/26/2010, 2:30:56	TEMP	97.8
68729	05/26/2010, 4:51:52	DISCHARGE	
239684		EYE_COLOR//BROWN	
239684		HEIGHT	175.2711152
239684	12/28/1980, 0:00:00	MEDS_BIRTH	
239684	05/11/2010, 17:41:51	ADMISSION//CARDIAC	
239684	05/11/2010, 17:41:51	HR	102.6
239684	05/11/2010, 17:41:51	TEMP	96
239684	05/11/2010, 17:48:48	HR	105.1
239684	05/11/2010, 17:48:48	TEMP	96.2

Non-standard vocabularies

MEDS mandates *standardization*, not *harmonization*

sujbect_id	time	code	numeric_value
68729		EYE_COLOR//HAZEL	
68729		HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	MEDS_BIRTH	
239684	12/28/1980, 0:00:00	MEDS_BIRTH	
239684	05/11/2010, 17:41:51	ADMISSION//CARDIAC	
239684	05/11/2010, 17:41:51	HR	102.6
239684	05/11/2010, 17:41:51	TEMP	96
239684	05/11/2010, 17:48:48	HR	105.1
239684	05/11/2010, 17:48:48	TEMP	96.2

Non-standard vocabularies

Mandate *Standardization* while remaining agnostic to *Harmonization*

We can support model training recipe transportability and health AI development without requiring concepts to be aligned with a particular vocabulary.

MEDS: minimal standard, no harmonization

sujbect_id	time	code	numeric_value
68729		EYE_COLOR//HAZEL	
68729		HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	MEDS_BIRTH	
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	
68729	05/26/2010, 2:30:56	HR	86
68729	05/26/2010, 2:30:56	TEMP	97.8
68729	05/26/2010, 4:51:52	DISCHARGE	
239684		EYE_COLOR//BROWN	
239684		HEIGHT	175.2711152
239684	12/28/1980, 0:00:00	MEDS_BIRTH	
239684	05/11/2010, 17:41:51	ADMISSION//CARDIAC	
239684	05/11/2010, 17:41:51	HR	102.6
239684	05/11/2010, 17:41:51	TEMP	96
239684	05/11/2010, 17:48:48	HR	105.1
239684	05/11/2010, 17:48:48	TEMP	96.2

Intervals via start...

... and end events

What is MEDS missing?

1. Additional data modalities (e.g., waveforms, images).
    a. Images are relatively easy to support if needed.
    b. Waveforms are more challenging as they don’t conform to the core MEDS standard in general.

What is MEDS missing?

1. Additional data modalities (e.g., waveforms, images).
    a. Images are relatively easy to support if needed.
    b. Waveforms are more challenging as they don't conform to the core MEDS standard in general.
2. Complex representations of "interval" events - instead, separate "start" and "end" events are used.

What is MEDS missing?

1. Additional data modalities (e.g., waveforms, images).
    a. Images are relatively easy to support if needed.
    b. Waveforms are more challenging as they don't conform to the core MEDS standard in general.
2. Complex representations of "interval" events - instead, separate "start" and "end" events are used.
3. Guarantees of harmonization to external ontologies.

What is MEDS missing?

1. Additional data modalities (e.g., waveforms, images).
    a. Images are relatively easy to support if needed.
    b. Waveforms are more challenging as they don't conform to the core MEDS standard in general.
2. Complex representations of "interval" events - instead, separate "start" and "end" events are used.
3. Guarantees of harmonization to external ontologies.
4. Canonical representations of different sites for multi-site datasets.

What is MEDS missing?

1. Additional data modalities (e.g., waveforms, images).
    a. Images are relatively easy to support if needed.
    b. Waveforms are more challenging as they don't conform to the core MEDS standard in general.
2. Complex representations of "interval" events - instead, separate "start" and "end" events are used.
3. Guarantees of harmonization to external ontologies.
4. Canonical representations of different sites for multi-site datasets.
5. Canonical representations of providers for provider-ID augmented datasets

What is MEDS missing?

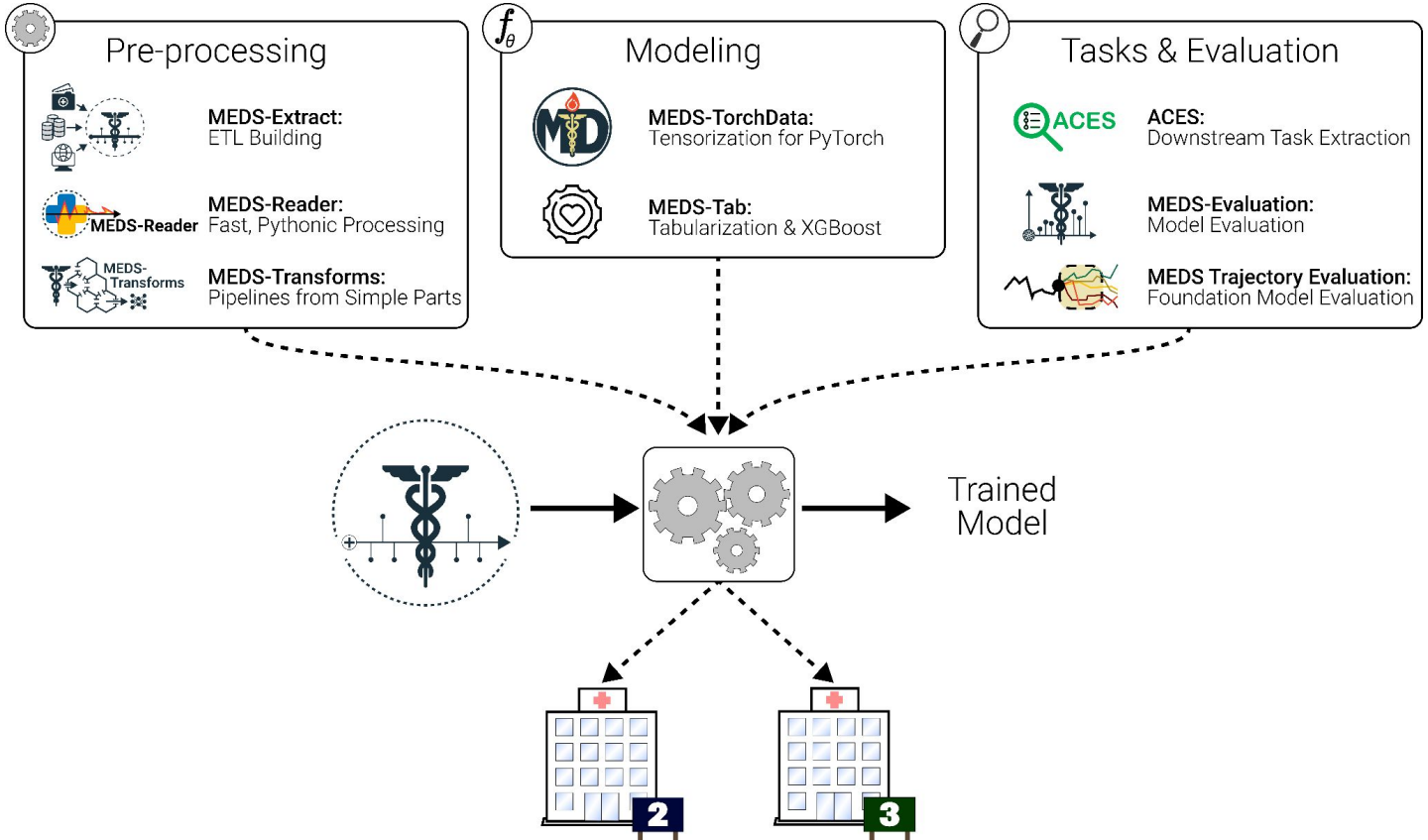
1. Additional data modalities (e.g., waveforms, images).
    a. Images are relatively easy to support if needed.
    b. Waveforms are more challenging as they don't conform to the core MEDS standard in general.
2. Complex representations of "interval" events - instead, separate "start" and "end" events are used.
3. Guarantees of harmonization to external ontologies.
4. Canonical representations of different sites for multi-site datasets.
5. Canonical representations of providers for provider-ID augmented datasets.

What is MEDS missing?

1. Additional data modalities (e.g., waveforms, images).
    a. Images are relatively easy to support if needed.
    b. Waveforms are more challenging as they don't conform to the core MEDS standard in general.
2. Complex representations of "interval" events - instead, separate "start" and "end" events are used.
3. Guarantees of harmonization to external ontologies.
4. Canonical representations of different sites for multi-site datasets.
5. Canonical representations of providers for provider-ID augmented datasets.
6. Natural ways to capture multi-part numeric values (e.g., medication dose, frequency, duration).

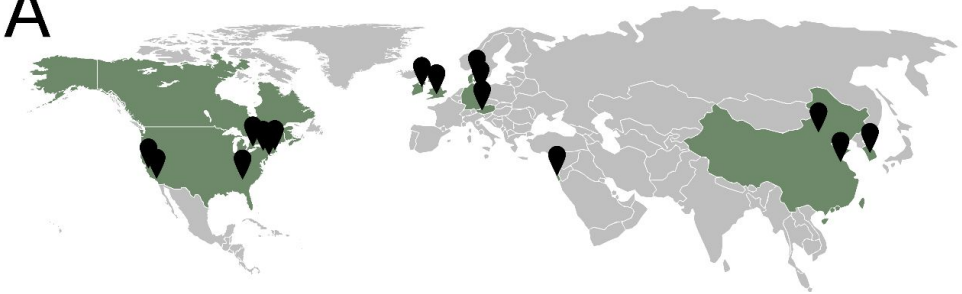
… Why should you use it?

MEDS


Pre-processing
MEDS-Extract:
ETL Building
MEDS-Reader
MEDS-Reader:
Fast, Pythonic Processing
MEDS-Transforms
MEDS-Transforms:
Pipelines from Simple Parts
Modeling
MEDS-TorchData:
Tensorization for PyTorch
MEDS-Tab:
Tabularization & XGBoost
Tasks & Evaluation
ACES
ACES:
Downstream Task Extraction
MEDS-Evaluation:
Model Evaluation
MEDS Trajectory Evaluation:
Foundation Model Evaluation
Trained
Model
2
3

MEDS

A



Used in at least 19 institutions

B

14 Papers

17 Datasets

11 Models

C

Up to 100x Faster

Up to 70% fewer lines of code

More than 13 tools available

Part 1 Colab Notebook: Introduction to MEDS

Time: 5 Minutes

Learning Goals:

1. Ensure the colab notebook setting is familiar.
2. Gain familiarity with the file structure and schema contents of MEDS.

www.tinyurl.com/
MEDS-ML4H-p1

Part 2: Problem Set-up

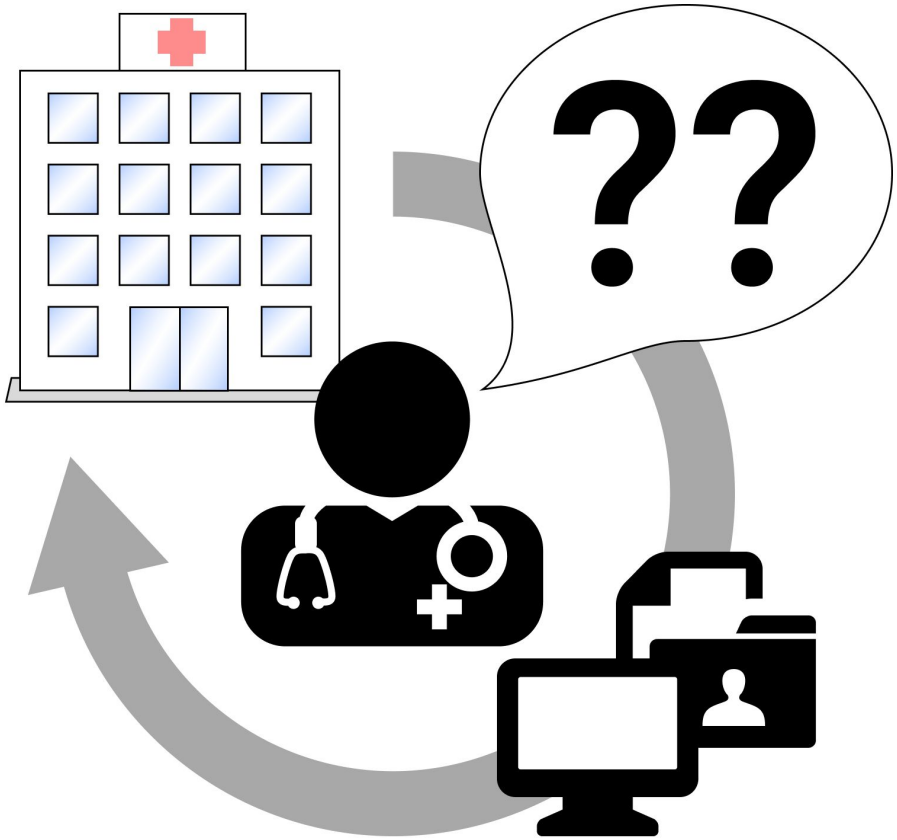
Disclaimers and Goal

This tutorial will be structured around a hypothetical problem. But...

1. The sample problem and setting were chosen for their utility in this tutorial, not for their utility as true clinical tasks!
2. This tutorial will work over *demo data only* -- you should not expect results to be individually meaningful or generalize.
3. Tools and pipelines in this tutorial are configured as they are for educational utility -- do not assume they would necessarily be appropriate in a real modeling task!

Question:

A new colleague asks you: What model should they build for their data?



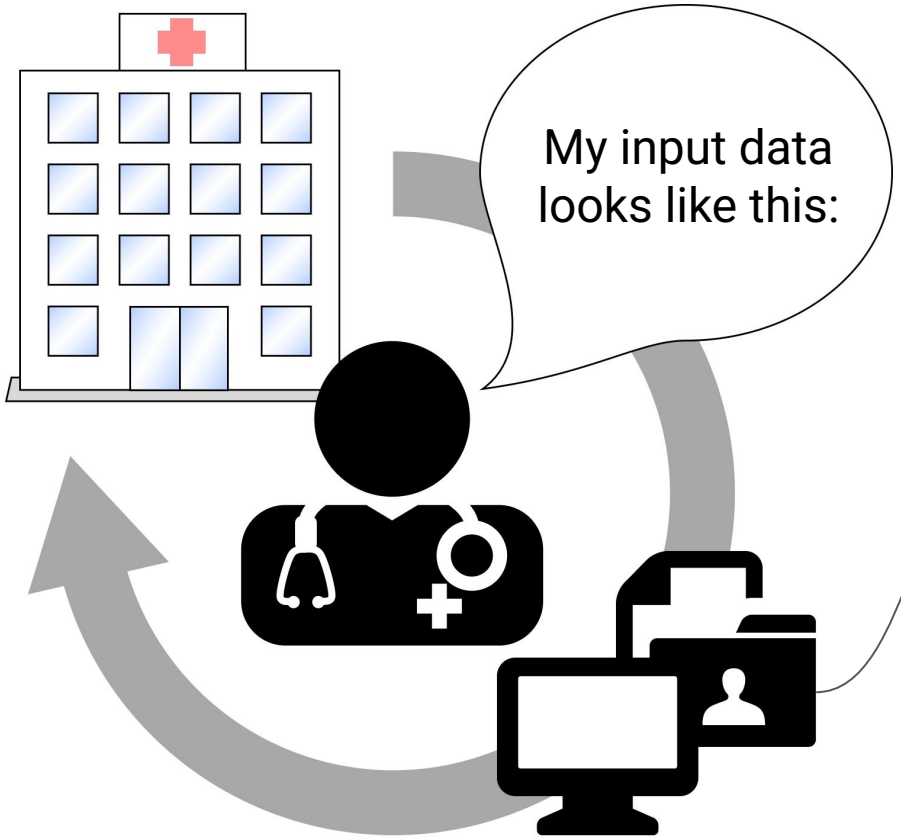
This tutorial

Their data: *MIMIC-IV Demo Dataset* (https://physionet.org/content/mimic-iv-demo/2.2/)

Their problem: *Identify patients who will have a long length of stay in the ICU*

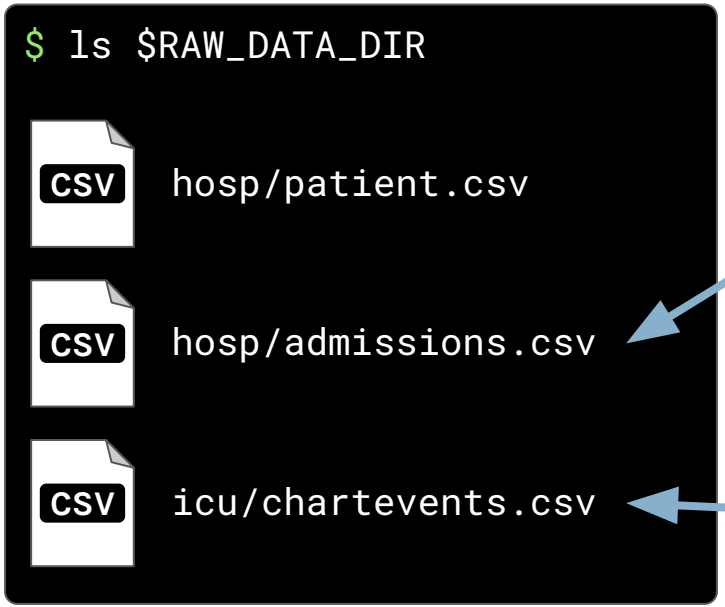
Your goal: *Help them perform this modeling task and identify the right problem framing and model to perform this prediction task, in a computational (i.e., non-deployment) setting*

Part 3: Convert your data into MEDS



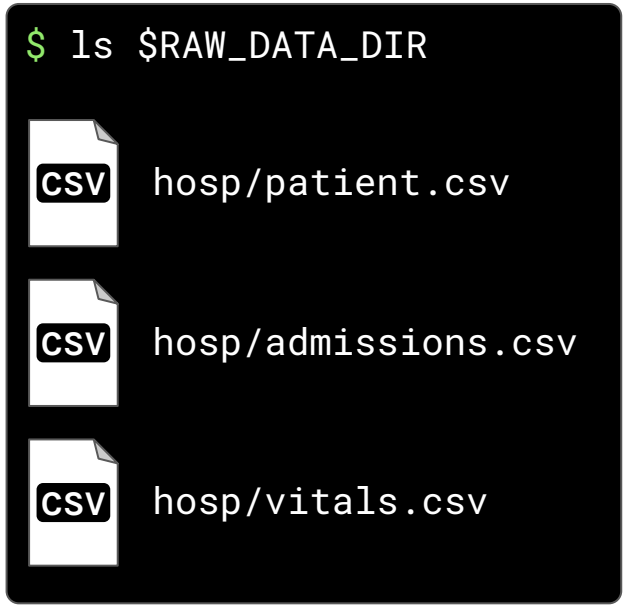

```
$ ls $RAW_DATA_DIR

CSV  hosp/patient.csv.gz

CSV  hosp/admissions.csv.gz

CSV  icu/chartevents.csv.gz
```

MEDS' Simplicity Enables Easy Extraction Pipelines



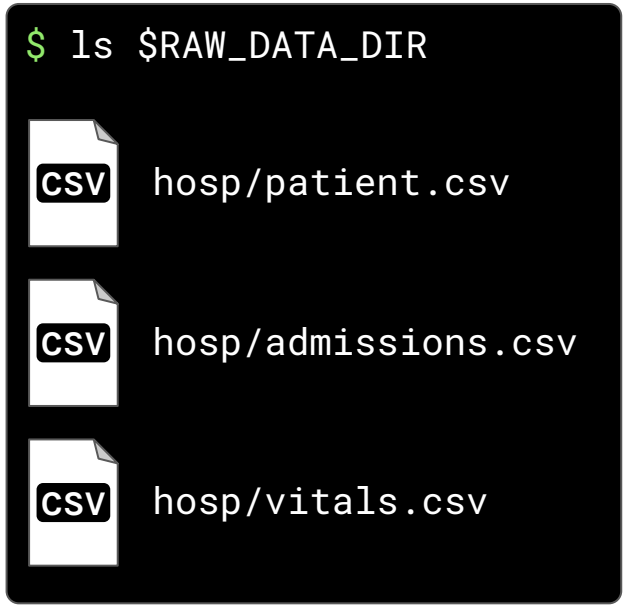

```yaml
subject_id_col: subject_id
hosp/admissions:
  ed_registration:
    code: ED_REGISTRATION
    time: col(edregtime)
    time_format: "%Y-%m-%d %H:%M:%S"
  admission:
    code:
      - HOSPITAL_ADMISSION
      - col(admission_type)
      - col(admission_location)
    time: col(admittime)
    time_format: "%Y-%m-%d %H:%M:%S"
    insurance: insurance
    language: language
    marital_status: marital_status
    race: race
    hadm_id: hadm_id

...

icu/chartevents:
  event:
    code:
      - LAB
      - col(itemid)
      - col(valueuom)
    time: col(charttime)
    time_format: "%Y-%m-%d %H:%M:%S"
    numeric_value: valuenum
    text_value: value
    hadm_id: hadm_id
    icustay_id: stay_id
    _metadata:
      d_labitems_to_loinc:
        description: ["omop_concept_name", "label"]
        itemid: "itemid (omop_source_code)"
        parent_codes: "{omop_vocabulary_id}/{omop_concept_code}"
        valueuom: "valueuom"
```

Extracting Data to MEDS with MEDS-Extract



Assumption:
Each row of each file corresponds to one or more *complete* measurements in the output MEDS data view.

Extracting Data to MEDS with MEDS-Extract

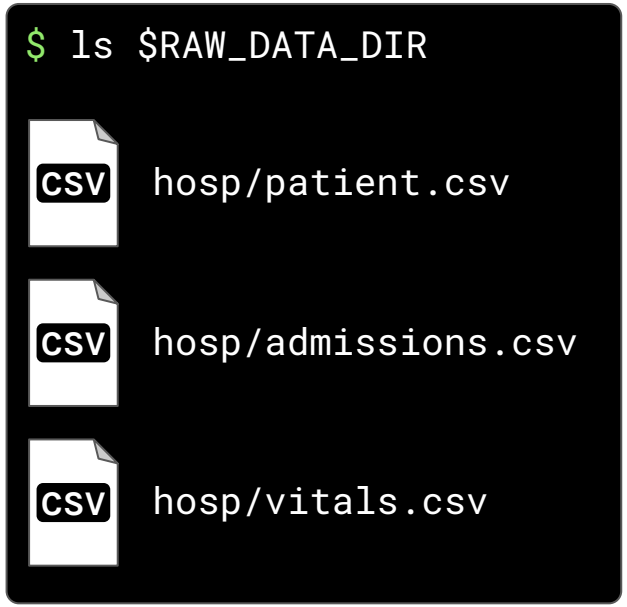


Assumption:
Each row of each file corresponds to one or more *complete* measurements in the output MEDS data view.

May require some "*pre-MEDS*" work to validate.

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/pre_MEDS.py

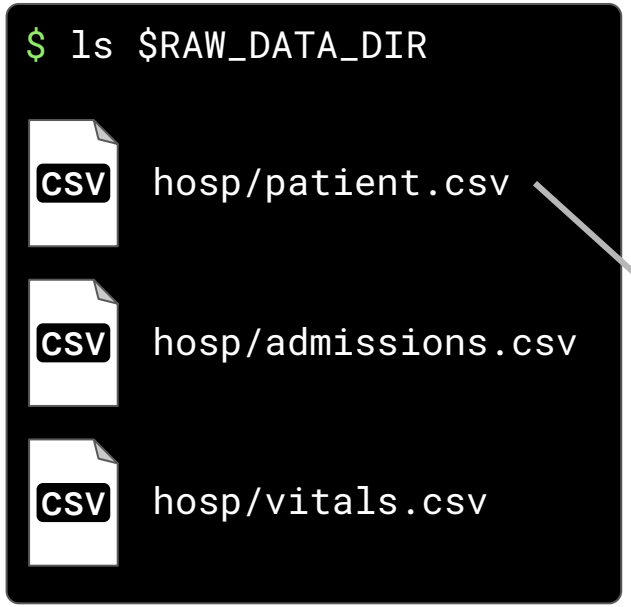
Every row of a MEDS data shard maps to a given input file:



subject_id	time	code	numeric_value
68729	*null*	EYE_COLOR//HAZEL	*null*
68729	*null*	HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	DOB	*null*
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	*null*
68729	05/26/2010, 2:30:56	HR	86

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/configs/event_configs.yaml

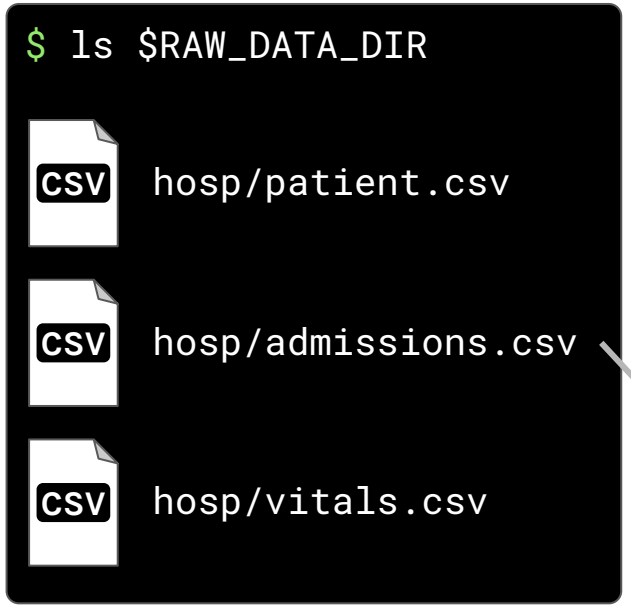
Every row of a MEDS data shard maps to a given input file:



subject_id	time	code	numeric_value
68729	*null*	EYE_COLOR//HAZEL	*null*
68729	*null*	HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	DOB	*null*
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	*null*
68729	05/26/2010, 2:30:56	HR	86

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/configs/event_configs.yaml

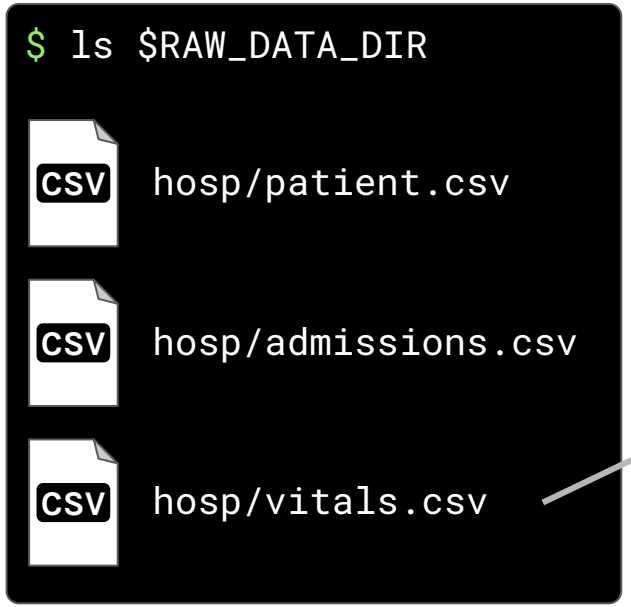
Every row of a MEDS data shard maps to a given input file:



subject_id	time	code	numeric_value
68729	*null*	EYE_COLOR//HAZEL	*null*
68729	*null*	HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	DOB	*null*
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	*null*
68729	05/26/2010, 2:30:56	HR	86

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/configs/event_configs.yaml

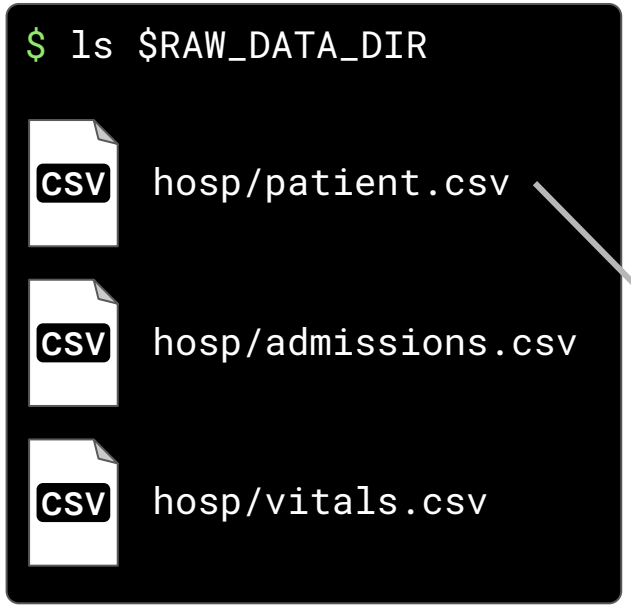
Every row of a MEDS data shard maps to a given input file:



subject_id	time	code	numeric_value
68729	*null*	EYE_COLOR//HAZEL	*null*
68729	*null*	HEIGHT	160.3953106
68729	03/09/1978, 0:00:00	DOB	*null*
68729	05/26/2010, 2:30:56	ADMISSION//PULMONARY	*null*
68729	05/26/2010, 2:30:56	HR	86

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/configs/event_configs.yaml

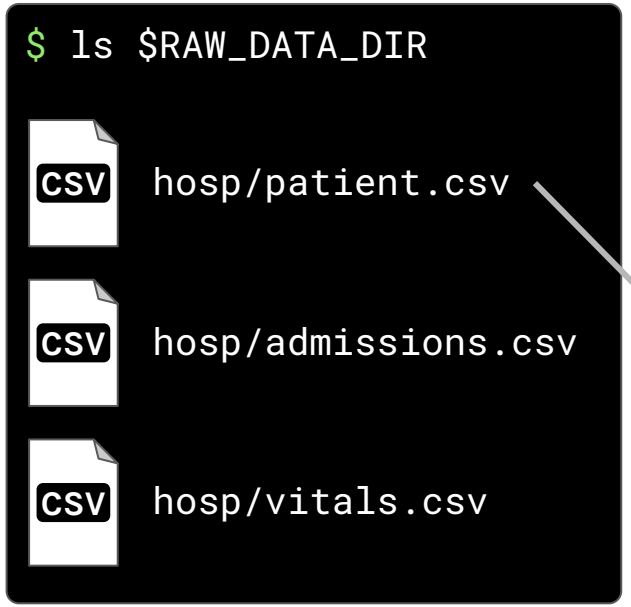
Reversed: Every row of each input file maps to one or more MEDS rows



subject_id	gender	anchor_age	anchor_year	anchor_year_group	dod
100	F	27	2153	2010-2015	*null*
101	M	54	2124	2015-2020	2144
102	M	78	2155	2010-2015	2160
...	...	...	...	...	...

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/configs/event_configs.yaml

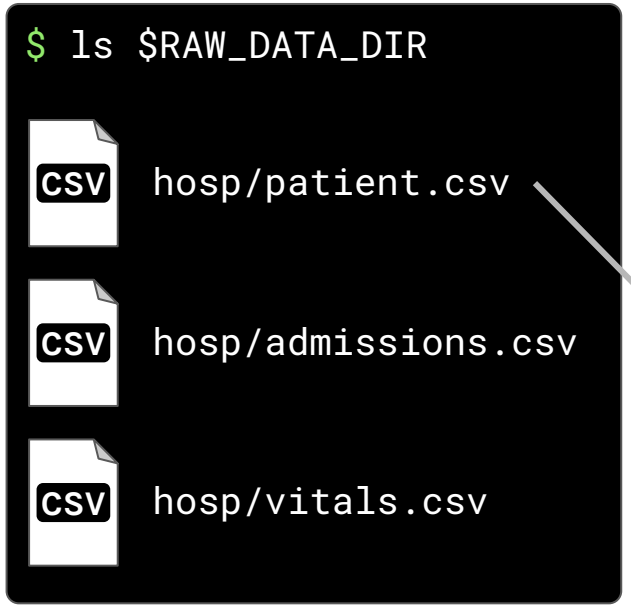
Reversed: Every row of each input file maps to one or more MEDS rows



subject_id	gender	anchor_age	anchor_year	anchor_year_group	dod
100	F	27	2153	2010-2015	*null*
101	M	54	2124	2015-2020	2144
102	M	78	2155	2010-2015	2160
...	...	...	...	...	...

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/configs/event_configs.yaml

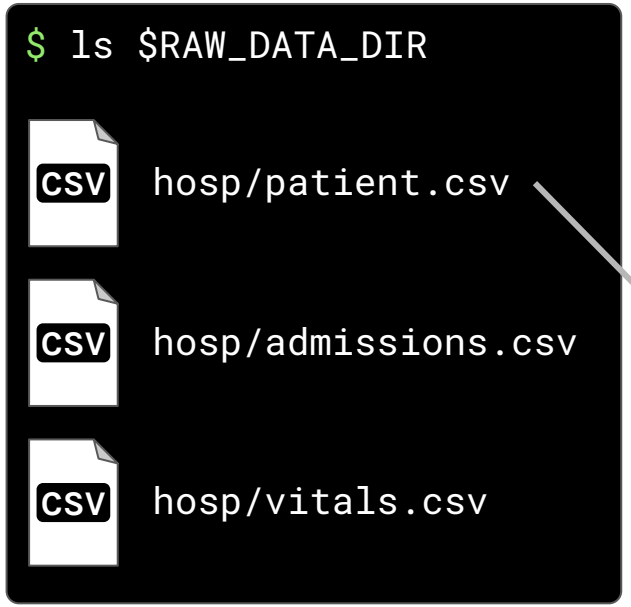
Reversed: Every row of each input file maps to one or more MEDS rows



subject_id	gender	anchor_age	anchor_year	anchor_year_group	dod
100	F	27	2153	2010-2015	*null*
101	M	54	2124	2015-2020	2144
102	M	78	2155	2010-2015	2160
...	...	...	...	...	...

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/configs/event_configs.yaml

Reversed: Every row of each input file maps to one or more MEDS rows



subject_id	gender	anchor_age	anchor_year	anchor_year_group	dod
100	F	27	2153	2010-2015	*null*
101	M	54	2124	2015-2020	2144
102	M	78	2155	2010-2015	2160
...	...	...	...	...	...

https://github.com/mmcdermott/MEDS_transforms/blob/main/MIMIC-IV_Example/configs/event_configs.yaml

MEDS: Extract your data by asking “who”, “what”, and “when”

1. You can think about MEDS data extraction as a repeated task of asking: “To whom is this happening?”, “What is happening?”, and “When is it happening?”
2. This maps to the MEDS-Extract Specification Syntax YAML (MESSY) format: allowing you to communicate and define your extraction parameters.
3. *Bonus:* You can also extract MEDS datasets using other tools or custom pipelines -- only the MEDS output matters!
4. *Bonus:* Lots of datasets already have publicly available MEDS ETLs! Check out this link for more information:

https://tinyurl.com/MEDS-Datasets

Part 3 Colab Notebook: Convert to MEDS

Time: 40 Minutes

Learning Goals:

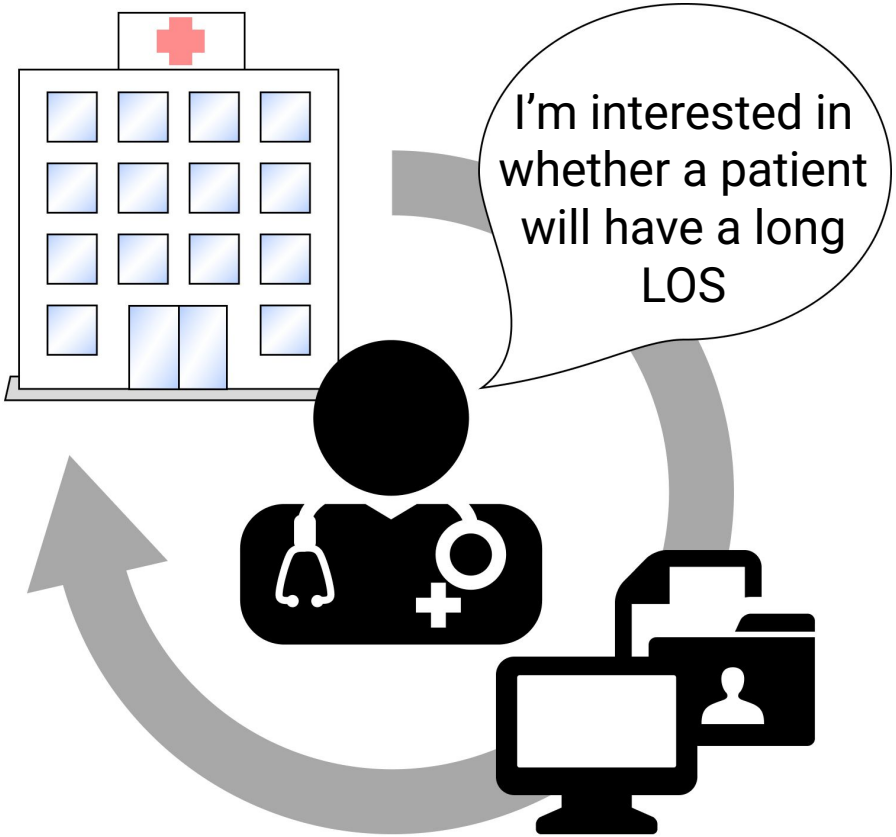
1. *Identify* MEDS observations in real data.
2. *Create* a specification using MEDS-Extract that realizes those MEDS observations.

Disclaimers:

1. S1 & S2: Many ways, many tools, and all can be improved!
2. MEDS conventions are not set in stone -- if you participate in the MEDS community, you can help define new conventions for how to do things properly!

www.tinyurl.com/
MEDS-ML4H-p3

Part 4: Identify your prediction task


I’m interested in whether a patient will have a long LOS

MEDS Label Schema

The MEDS label schema requires an index (a `subject_id`, and `prediction_time`) and permits optional labels of type including *`boolean_value`*, *`integer_value`*, *`float_value`*, and *`categorical_value`*. These labels can be predicted using any of the data of the indexed subject that occurred anytime at or before the indexed prediction time.

subject_id	prediction_time	*boolean_value*
68729	3/9/78 00:00	False
68729	5/2/10 14:22	False
68729	5/2/10 14:34	False
125829	4/9/18 18:19	True

MEDS Label Schema

Index DataFrame

The MEDS label schema requires an index (a `subject_id`, and `prediction_time`) and permits optional labels of type including *`boolean_value`*, *`integer_value`*, *`float_value`*, and *`categorical_value`*. These labels can be predicted using any of the data of the indexed subject that occurred anytime at or before the indexed prediction time.

subject_id	prediction_time	*boolean_value*
68729	3/9/78 00:00	False
68729	5/2/10 14:22	False
68729	5/2/10 14:34	False
125829	4/9/18 18:19	True

MEDS Label Schema

"Task" or "Label" DataFrame

The MEDS label schema requires an index (a `subject_id`, and `prediction_time`) and permits optional labels of type including *`boolean_value`*, *`integer_value`*, *`float_value`*, and *`categorical_value`*. These labels can be predicted using any of the data of the indexed subject that occurred anytime at or before the indexed prediction time.

subject_id	prediction_time	*boolean_value*
68729	3/9/78 00:00	False
68729	5/2/10 14:22	False
68729	5/2/10 14:34	False
125829	4/9/18 18:19	True

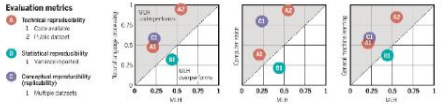
ACES: Reproducible Extraction of Task Cohorts for EHRs

Justin Xu, Jack Gallifant, Alistair E. W. Johnson, Matthew B. A. McDermott

ICLR International Conference On Learning Representations
UNIVERSITY OF OXFORD · MIT · HARVARD MEDICAL SCHOOL

Health AI has a Reproducibility Crisis

Health AI faces a **systemic reproducibility crisis**, limiting our ability to do effective science. We need to build a health AI ecosystem to change that, and ACES builds on that foundation.



ACES Leverages Event-Stream Schemas

The MEDS data schema epitomizes simplicity and has only 4 required columns: `subject_id`, `time`, `code`, and `numeric_value`.

subject_id	time	code	numeric_value
68729	null	RACE//WHITE	null
68729	null	SEX//M	null
68729	3/9/78 00:00	MEDS_BIRTH	null
68729	5/2/10 14:22	ED//REG	null
68729	5/2/10 14:34	HR//bpm	93.0
68729	5/2/10 20:00	ED//OUT	null
125829	null	SEX//F	null
125829	4/9/10 18:19	ADMISSION//CARDIAC	null

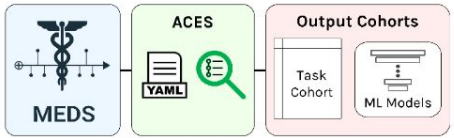
code	description	parent_codes
RACE//WHITE	The patient's race is...	null
SEX//M	The patient's biologic...	null
SEX//F	The patient's biologic...	null
MEDS_BIRTH	null	null
MEDS_DEATH	null	null
HR//bpm	Heart rate, measured...	[LOINC/8867-4]
ED//REG	Emergency departme...	null

The MEDS codes schema may contain descriptions and links to external ontologies for elements of the `code` vocabulary.

- Single stream of events!
- Simple and flexible to use!
- Easy transformations!

Learn More about MEDS:

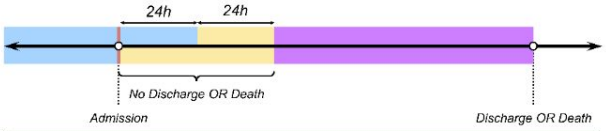
Task Extraction Made Easy!



- Transparent
- Reproducible
- Extractable on diverse datasets

ACES Configuration Files

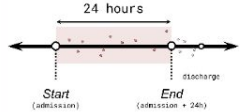
In-hospital Mortality: Given the first 24 hours of a patient's stay, predict whether or not they will die within this hospital admission, with a gap time of 48h.




```yaml
predicates:
  admission:
    code: ADMISSION
  discharge:
    code: DISCHARGE
  death:
    code: DEATH
  discharge_or_death:
    expr: or(discharge, death)

trigger: admission
```

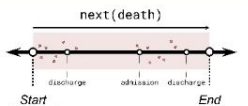
```yaml
windows:
  input:
    start: NULL
    end: trigger + 24h
    start_inclusive: True
    end_inclusive: True

  gap:
    start: trigger
    end: start + 48h
    start_inclusive: False
    end_inclusive: True

  target:
    start: gap.end
    end: start -> discharge_or_death
    start_inclusive: False
    end_inclusive: True
    label: death
```

Temporal Aggregation



*Predicates are bounded by **time***

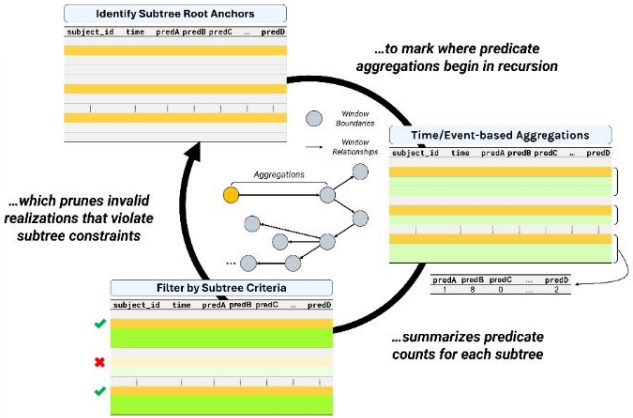
Event-based Aggregation



*Predicates are bounded by **specific events***

ACES Demo & MEDS Tutorial:

Recursive Algorithm for Extraction



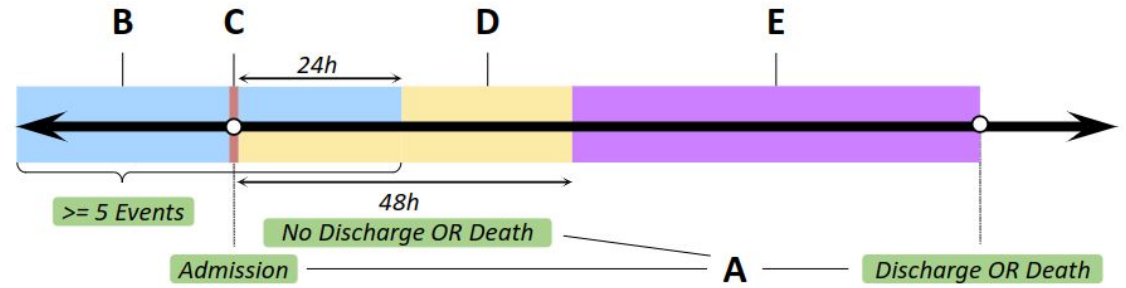
How can ACES help you?

- For a variety of pre-defined tasks, check out (or contribute to) MEDS-DEV:
- For more info on how to write your own ACES configs, check out our documentation:

Acknowledgements

MIRAM gratefully acknowledges support from a Berkowitz Postdoctoral Fellowship at Harvard Medical School. JG is funded by the National Institutes of Health through NIH-USA RDTCA294033. JX greatly appreciates support from supervisors David Eyre (University of Oxford) and Curtis Langlotz (Stanford University). We also acknowledge valuable contributions by Tom Pollard (Massachusetts Institute of Technology) and by the broader MEDS ecosystem of contributors and users.

In-hospital Mortality Prediction



Task Configuration

A

```
predicates:
    admission:
        code: ADMISSION
    discharge:
        code: DISCHARGE
    death:
        code: DEATH
    discharge_or_death:
        expr: or(discharge, death)
```

B

```
input:
    start: NULL
    end: trigger + 24 hours
    start_inclusive: True
    end_inclusive: True
    has:
        _ANY_EVENT: (5, None)
```

C

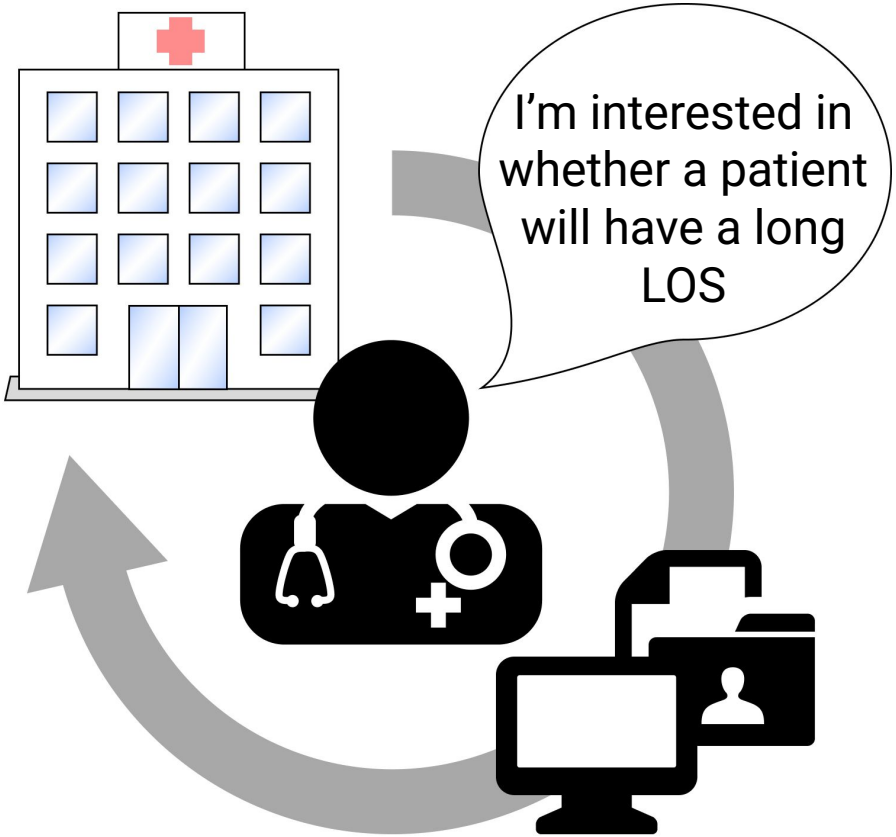
```
trigger: admission
```

D

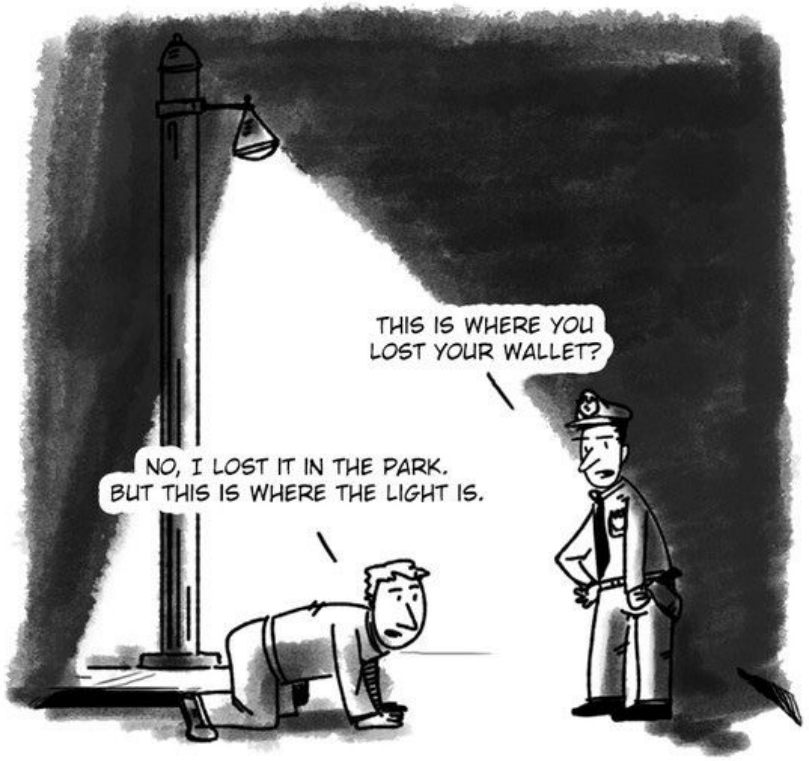
```
gap:
    start: trigger
    end: start + 48 hours
    start_inclusive: False
    end_inclusive: True
    has:
        admission: (None, 0)
        discharge: (None, 0)
        death: (None, 0)
```

E

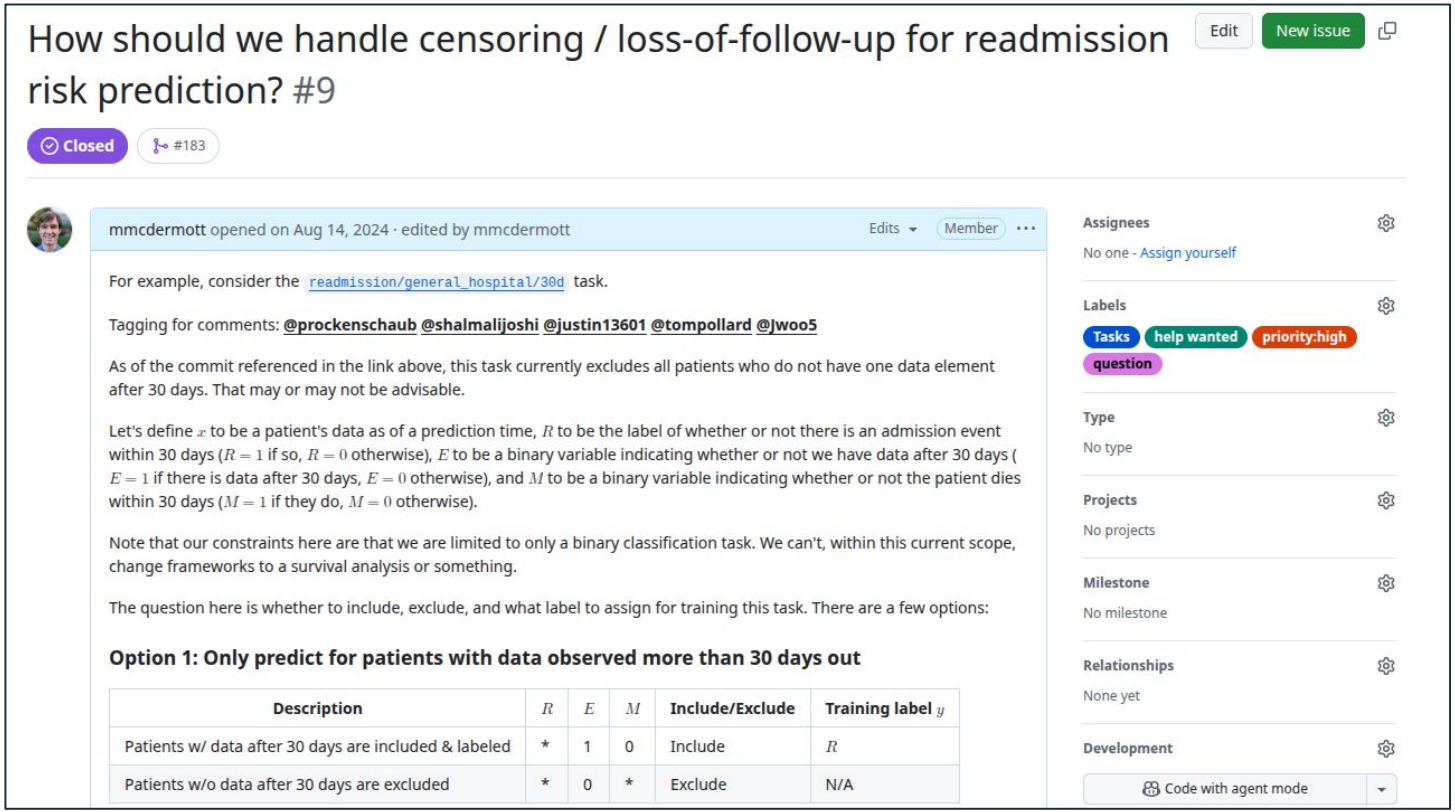
```
target:
    start: gap.end
    end: start -> discharge_or_death
    start_inclusive: False
    end_inclusive: True
    label: death
```


I’m interested in whether a patient will have a long LOS

Measure what you value



Tasks can be discussed by the community!



How should we handle censoring / loss-of-follow-up for readmission risk prediction? #9

Edit | New issue

Closed | #183

mmcdermott opened on Aug 14, 2024 · edited by mmcdermott — Edits | Member

For example, consider the `readmission/general_hospital/30d` task.

Tagging for comments: **@prockenschaub @shalmalijoshi @justin13601 @tompollard @Jwoo5**

As of the commit referenced in the link above, this task currently excludes all patients who do not have one data element after 30 days. That may or may not be advisable.

Let's define x to be a patient's data as of a prediction time, R to be the label of whether or not there is an admission event within 30 days ($R = 1$ if so, $R = 0$ otherwise), E to be a binary variable indicating whether or not we have data after 30 days ($E = 1$ if there is data after 30 days, $E = 0$ otherwise), and M to be a binary variable indicating whether or not the patient dies within 30 days ($M = 1$ if they do, $M = 0$ otherwise).

Note that our constraints here are that we are limited to only a binary classification task. We can't, within this current scope, change frameworks to a survival analysis or something.

The question here is whether to include, exclude, and what label to assign for training this task. There are a few options:

Option 1: Only predict for patients with data observed more than 30 days out

Description	R	E	M	Include/Exclude	Training label y
Patients w/ data after 30 days are included & labeled	*	1	0	Include	R
Patients w/o data after 30 days are excluded	*	0	*	Exclude	N/A

Assignees: No one - Assign yourself

Labels: Tasks, help wanted, priority:high, question

Type: No type

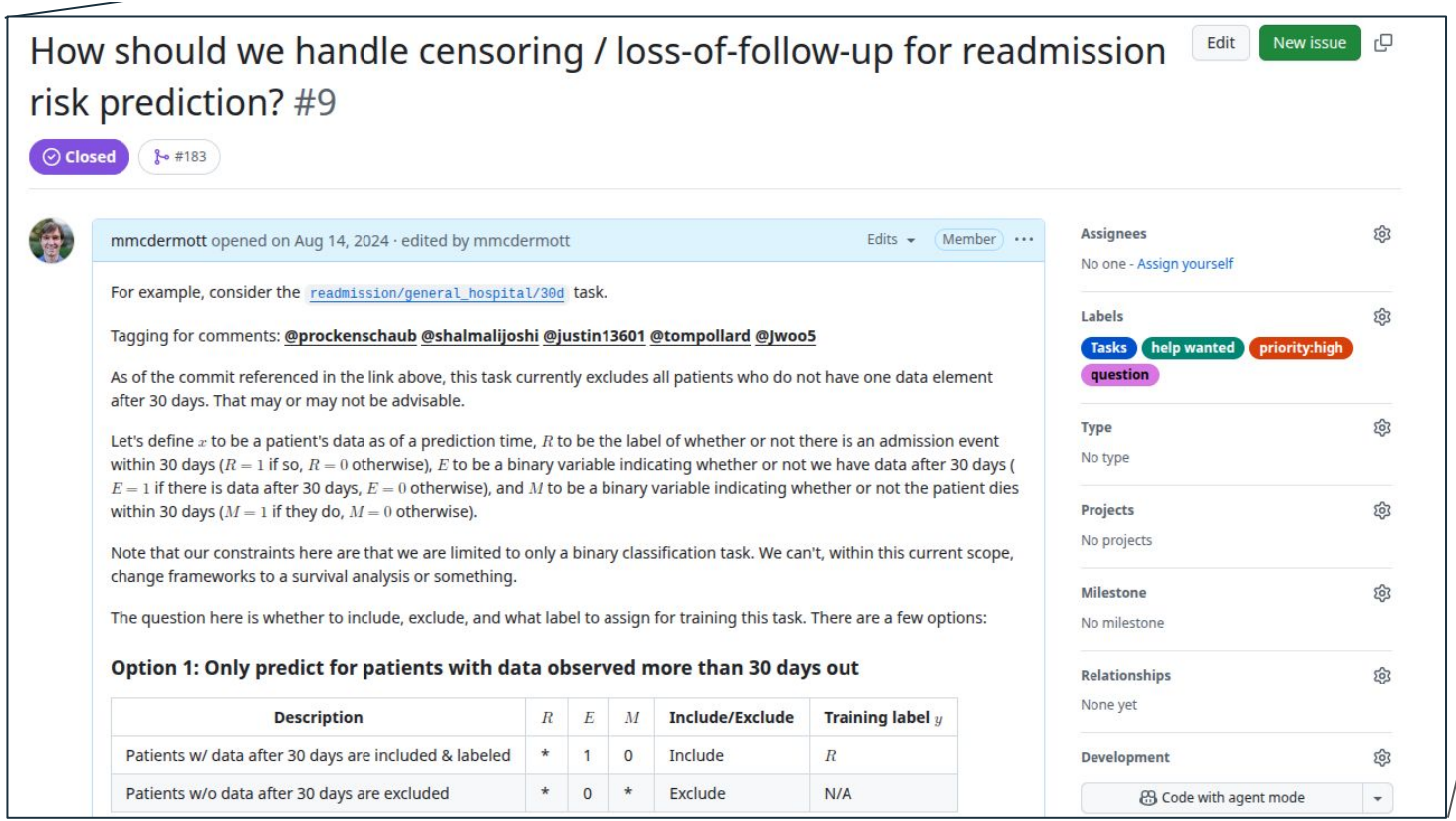
Projects: No projects

Milestone: No milestone

Relationships: None yet

Development: Code with agent mode

Tasks can be discussed by the community!



How should we handle censoring / loss-of-follow-up for readmission risk prediction? #9

Edit | New issue

Closed | #183

mmcdermott opened on Aug 14, 2024 · edited by mmcdermott — Edits | Member

For example, consider the `readmission/general_hospital/30d` task.

Tagging for comments: **@prockenschaub @shalmalijoshi @justin13601 @tompollard @Jwoo5**

As of the commit referenced in the link above, this task currently excludes all patients who do not have one data element after 30 days. That may or may not be advisable.

Let's define x to be a patient's data as of a prediction time, R to be the label of whether or not there is an admission event within 30 days ($R = 1$ if so, $R = 0$ otherwise), E to be a binary variable indicating whether or not we have data after 30 days ($E = 1$ if there is data after 30 days, $E = 0$ otherwise), and M to be a binary variable indicating whether or not the patient dies within 30 days ($M = 1$ if they do, $M = 0$ otherwise).

Note that our constraints here are that we are limited to only a binary classification task. We can't, within this current scope, change frameworks to a survival analysis or something.

The question here is whether to include, exclude, and what label to assign for training this task. There are a few options:

Option 1: Only predict for patients with data observed more than 30 days out

Description	R	E	M	Include/Exclude	Training label y
Patients w/ data after 30 days are included & labeled	*	1	0	Include	R
Patients w/o data after 30 days are excluded	*	0	*	Exclude	N/A

Assignees: No one - Assign yourself

Labels: Tasks, help wanted, priority:high, question

Type: No type

Projects: No projects

Milestone: No milestone

Relationships: None yet

Development: Code with agent mode

Final task:

Predict using the first 24 hours of data if the ICU stay will be less than 3 days long.

```yaml
predicates:
  icu_admission:
    code: { regex: "^ICU_ADMISSION//.*" }
  icu_discharge:
    code: { regex: "^ICU_DISCHARGE//.*" }
  death:
    code: { regex: "MEDS_DEATH.*" }

  # CMO predicates
  cmo_1:
    code: { any: ["LAB//220001//UNK", "LAB//223758//UNK"] }
    text_value: "Comfort measures only"
  cmo_2:
    code: { any: ["LAB//220001//UNK", "LAB//223758//UNK"] }
    text_value: "Comfort care (CMO, Comfort Measures)"

  # DNR predicates
  dnr_1:
    code: { any: ["LAB//220001//UNK", "LAB//223758//UNK"] }
    text_value: "DNR / DNI"
  dnr_2:
    code: { any: ["LAB//220001//UNK", "LAB//223758//UNK"] }
    text_value: "DNAR (Do Not Attempt Resuscitation)  [DNR]"
  dnr_3:
    code: { any: ["LAB//220001//UNK", "LAB//223758//UNK"] }
    text_value: "DNAR (Do Not Attempt Resuscitation) [DNR] / DNI"
  dnr_4:
    code: { any: ["LAB//220001//UNK", "LAB//223758//UNK"] }
    text_value: "DNR (do not resuscitate)"

  # derived predicates
  cmo:
    expr: or(cmo_1, cmo_2)
  dnr:
    expr: or(dnr_1, dnr_2, dnr_3, dnr_4)

trigger: icu_admission

windows:
  input:
    start: null
    end: trigger + 24h
    start_inclusive: True
    end_inclusive: True
    index_timestamp: end
    has:
      cmo: (None, 0) # Exclude patients on comfort measures only
      dnr: (None, 0) # Exclude patients with DNR orders
  gap:
    start: trigger
    end: start + 30h
    start_inclusive: False
    end_inclusive: True
    has:
      cmo: (None, 0)
      dnr: (None, 0)
      icu_discharge: (None, 0)
  target:
    start: trigger
    end: start + 3d
    start_inclusive: True
    end_inclusive: True
    label: icu_discharge
    has:
      death: (None, 0)
```

Part 4 Colab Notebook: Develop an ACES File

Time: 20 Minutes

Learning Goals:

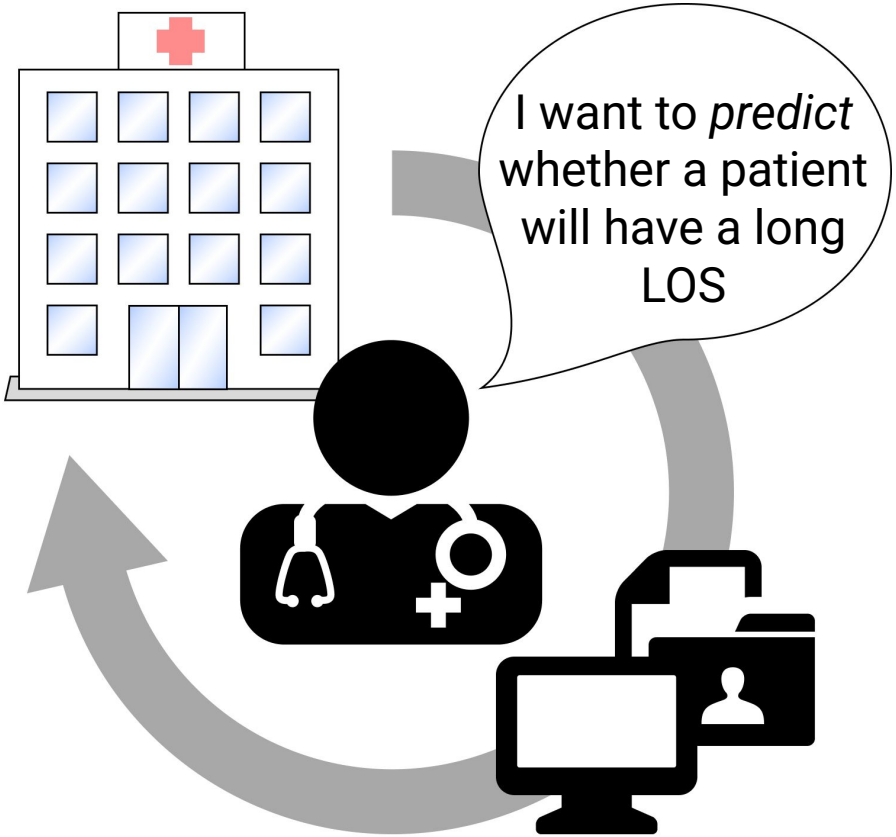
1. *Examine*, *question*, and *refine* a stated prediction goal into a meaningful predictive cohort.
2. *Create* an ACES configuration file to extract that cohort.

Disclaimers:

1. S1 & S2: Many ways, many tools, and all can be improved!
2. This task and sample “conversation” is not necessarily representative of a real discussion with a clinician.

www.tinyurl.com/
MEDS-ML4H-p4

Part 5: Building a tabular baseline model


I want to *predict* whether a patient will have a long LOS

Part 5 Colab Notebook: Build a Tabular Model

Time: 30 Minutes

Learning Goals:

1. *Explain* tabular baseline modeling needs and how MEDS supports them.
2. *Assemble* a tabular baseline model for the MEDS data *using* default MEDS tools to make a prediction.

Disclaimers:

1. S1 & S2: Many ways, many tools, and all can be improved!
2. What model will work best on your data? Nobody knows but you (eventually)

www.tinyurl.com/
MEDS-ML4H-p5

Part 5 Takeaways:

1. Manual tabularization is not the primary goal of MEDS, but it is very tractable with sufficient data expertise.
2. Tabularization can be seen as asking “What measurements, over what windows relative to the prediction time, with what aggregation functions?”
3. Tabularization is challenging to do efficiently at large scale.

Part 6: Neural Network Models

README License

orchData: A PyTorch Dataset Class for MEDS
ts

thon 3.11+ | pypi v0.6.1 | MEDS 0.4 | docs passing | Tests passing | codecov 100%
in passing | Config Hydra 1.3 | contributors 7 | PRs welcome | License MIT

Start

all

```
meds-torch-data
```

a Tensorization

set is not sharded by split, you need to run a reshard to split stage first! You can enable this by
`do_reshard=True` argument to the command below.

If your input MEDS dataset lives in `$MEDS_ROOT` and you want to store your pre-processed files in `$PYD_ROOT`, you run:

```
MTD_preprocess MEDS_dataset_dir="$MEDS_ROOT" output_dir="$PYD_ROOT"
```

Step 3: Use the dataset

To use a dataset, you need to (1) define your configuration object and (2) create the dataset object. The only required configuration parameters are `tensorized_cohort_dir`, which points to the root directory containing the pre-processed data on disk (`$PYD_ROOT` in the above example), and `max_seq_len`, which is the maximum sequence length you want to use for your model. Here's an example:

```
import os
from meds_torchdata import MEDSPytorchDataset, MEDSTorchDataConfig

cfg = MEDSTorchDataConfig(tensorized_cohort_dir=os.environ["PYD_ROOT"], max_seq_len=512)
pyd = MEDSPytorchDataset(cfg, split="train")
```

```
stages:
  - fit_normalization:
      aggregations:
        - "code/n_occurrences"
        - "code/n_subjects"
        - "values/n_occurrences"
        - "values/sum"
        - "values/sum_sqd"
  - fit_vocabulary_indices
  - normalization
  - tokenization
  - tensorization
```

Part 6 Colab Notebook: Build a Neural Network Model

Time: 30 Minutes

Learning Goals:

1. *Explain* longitudinal neural network modeling needs and how MEDS supports them.
2. *Assemble* a neural network models *using* default MEDS tools to make a prediction.

Disclaimers:

1. S1 & S2: Many ways, many tools, and all can be improved!
2. What model will work best on your data? Nobody knows but you (eventually)

www.tinyurl.com/
MEDS-ML4H-p6

Part 6 Takeaways:

1. Neural network models have very different needs than tabular models.
2. For longitudinal neural network models, you need to ask how you pad or truncate sequences, how you sample sequences during training, and how you tokenize and tensorize your data.
3. Efficiency of data loading and processing is highly important.
4. MEDS-TorchData lets you build PyTorch models over MEDS datasets out of the box.

Conclusion

Part 1	Learn about the MEDS ecosystem	Explain why MEDS enables a collaborative, reproducible, open-source ecosystem in Health AI
Part 3	Convert new data into the MEDS format	Understand the conceptual and technical specification of MEDS-Extract
Part 4	Learn how to define prediction tasks with ACES	● Identify and decompose a task into operationalized targets ● Identify necessary predicates and criteria to define a common prediction task.
Part 5	Learn how to build baseline and neural network models using MEDS tools	● Separate tabularization and baseline model needs from NN needs. ● Leverage MEDS ecosystem tools like MEDS-Tab, MEDS Transforms, and MEDS TorchData to accelerate modeling.
Part 6	Learn how to leverage the research of the community to empower your own	

MEDS-DEV:

The MEDS Decentralized, Extensible, Validation Benchmark

If reproducibility is made trivial, we can realize all aspects of assessing a Health AI algorithm under a simple, easy to use interface

```
# Build datasets:
meds-dev-dataset dataset=$DATASET_NAME output_dir=$DATASET_DIR

# Extract tasks:
meds-dev-task task=$TASK_NAME dataset=$DATASET_NAME output_dir=$LABELS_DIR dataset_dir=$DATASET_DIR

# Train models:
# 1. Pre-train a model on unsupervised data
meds-dev-model model=$MODEL_NAME dataset_dir=$DATASET_DIR mode=train dataset_type=unsupervised
output_dir=$PRETRAINED_MODEL_DIR

# 2. Fine-tune a model on supervised data
meds-dev-model model=$MODEL_NAME dataset_dir=$DATASET_DIR labels_dir=$LABELS_DIR mode=train dataset_type=supervised
output_dir=$FINETUNED_MODEL_DIR model_initialization_dir=$PRETRAINED_MODEL_DIR

# 3. Make predictions with a model for the held-out set:
meds-dev-model model=$MODEL_NAME dataset_dir=$DATASET_DIR labels_dir=$LABELS_DIR mode=predict dataset_type=supervised
split=held_out output_dir=$PREDICTIONS_DIR model_initialization_dir=$FINETUNED_MODEL_DIR
```

Acknowledgements

Kamilė Stankevičiūtė
Ph.D. Student, University of Cambridge

Justin Xu
Ph.D. Student, University of Oxford

Ethan Steinberg
Researcher, Prealize Health

Acknowledgements

Edward Choi
Ethan Steinberg
Jason A. Fries
Jungwoo Oh
Matthew McDermott

Michael Wornow
Nigam H. Shah
Patrick Rockenschaub
Robin P. van de Water
Tom J. Pollard

Aleksia Kolo
Chao Pang
Edward Choi
Ethan Steinberg
Hyewon Jeong
Jack Gallifant
Jason A. Fries
Jeffrey N. Chiang

Jungwoo Oh
Justin Xu
Kamilė Stankevičiūtė
Kiril V. Klein
Matthew McDermott
Mikkel Odgaard
Nassim Oufattole
Patrick Rockenschaub
Pawel Renc

Robin van de Water
Shalmali Joshi
Simon A. Lee
Teya S. Bergamaschi
Tom J. Pollard
Vincent Jeanselme
Young Sang Choi

HARVARD MEDICAL SCHOOL

Stanford University

KAIST AI
Kim Jaechul Graduate School

COLUMBIA

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

UNIVERSITY OF OXFORD

UNIVERSITY OF CAMBRIDGE

prealize

MEDS Publications

MEDS Working Group: Bert Arnrich, Edward Choi, Jason A. Fries, Matthew B. A. McDermott, Jungwoo Oh, Tom J. Pollard, Nigam Shah, Ethan Steinberg, Michael Wornow, Robin van de Water. 2024. “Medical Event Data Standard (MEDS): Facilitating Machine Learning for Health.” In ICLR 2024 Workshop TS4H. https://openreview.net/forum?id=IsHy2ebjIG&referrer=%5BAuthor%20Console%5D(%2Fgroup%3Fid%3DICLR.cc%2F2024%2FWorkshop%2FTS4H%2FAuthors%23your-submissions).

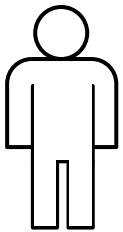
Oufattole, Nassim, Teya Bergamaschi, Aleksia Kolo, Hyewon Jeong, Hanna Gaggin, Collin M. Stultz, and Matthew B. A. McDermott. 2024. “MEDS-Tab: Automated Tabularization and Baseline Methods for MEDS Datasets.” *arXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2411.00200.

Steinberg, E., Michael Wornow, Suhana Bedi, J. Fries, Matthew B. A. McDermott, and Nigam H. Shah. 2024. “Meds_reader: A Fast and Efficient EHR Processing Library.” In *Machine Learning for Health Symposium (Findings Track)*. Vol. abs/2409.09095. https://doi.org/10.48550/arXiv.2409.09095.

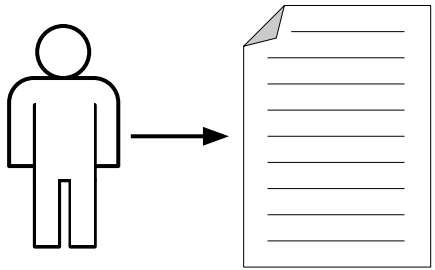
Xu, Justin, Jack Gallifant, Alistair E. W. Johnson, and Matthew B. A. McDermott. 2025. “ACES: Automatic Cohort Extraction System for Event-Stream Datasets.” In *Proceedings of the International Conference on Learning Representations (in Press)*.

Reproducibility Crisis

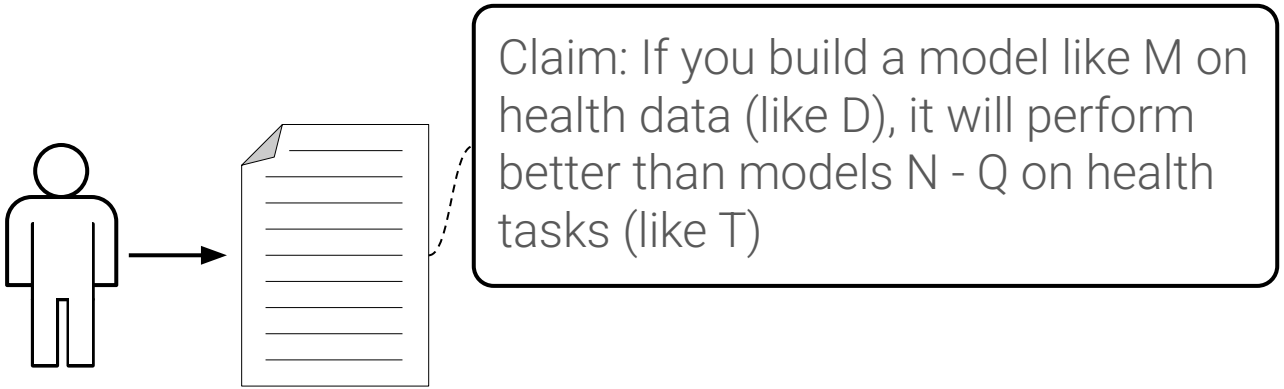
Scientific Process of Health AI



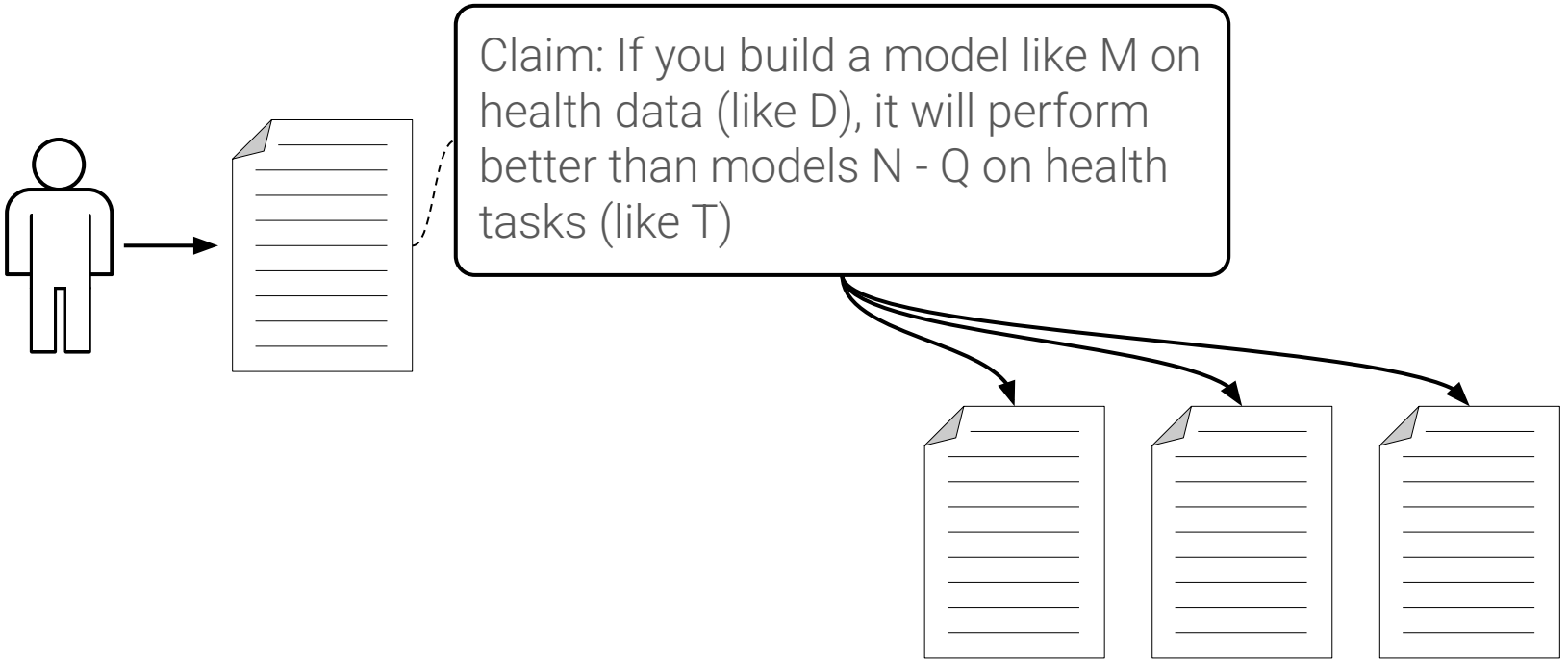
Scientific Process of Health AI



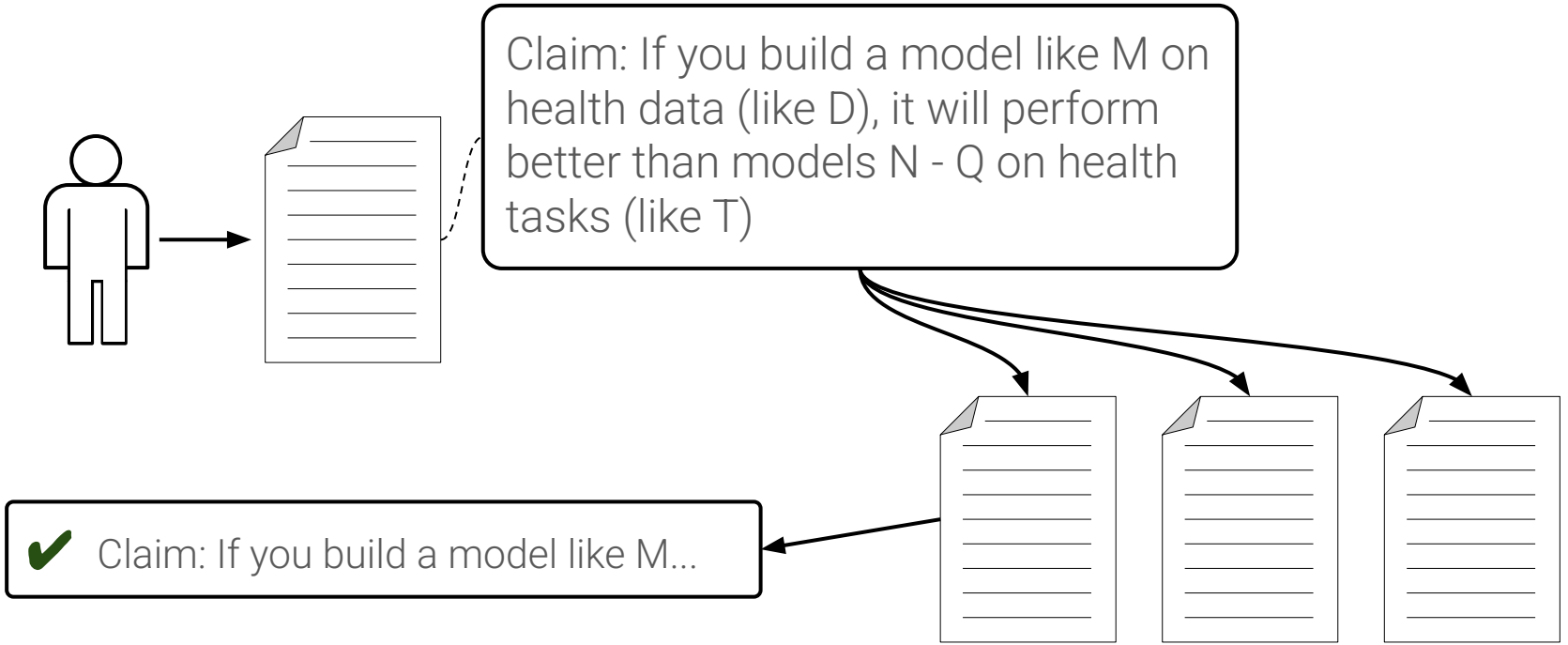
Scientific Process of Health AI: Claims over model training recipe superiority



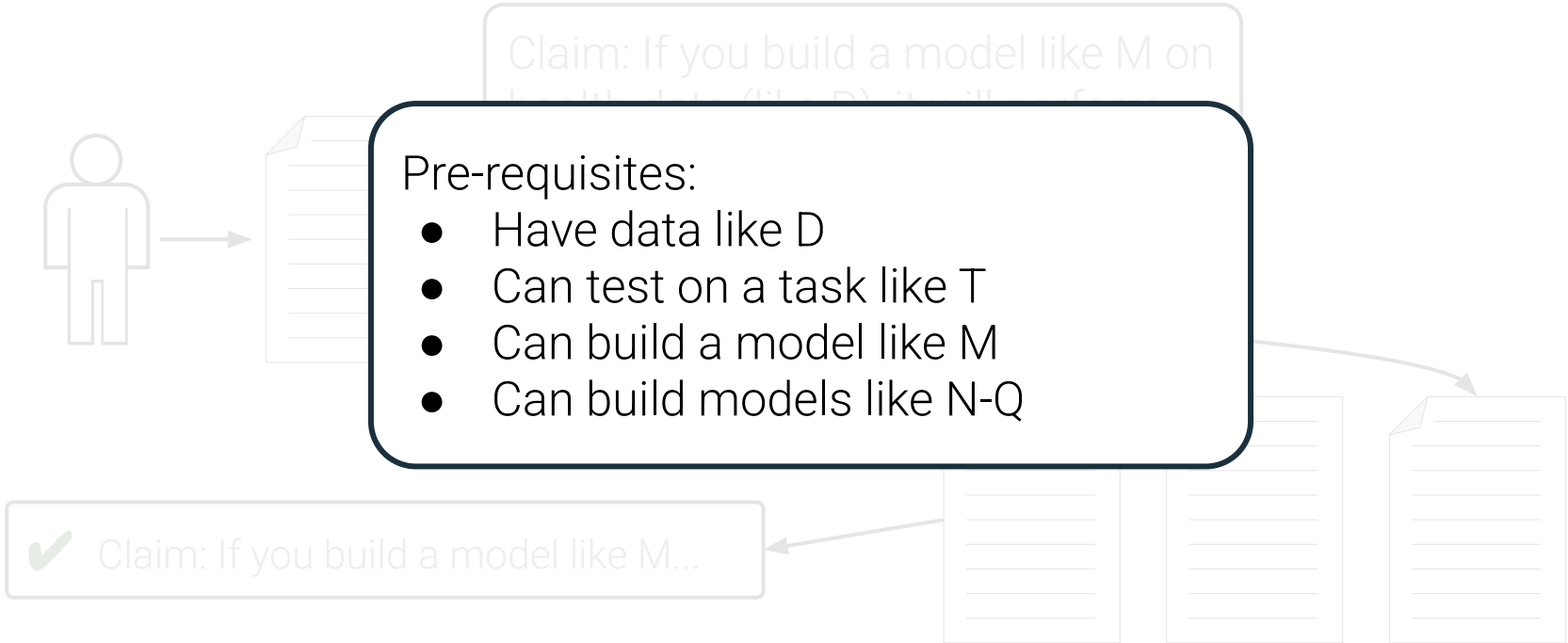
Scientific Process of Health AI: Claims over model training recipe superiority



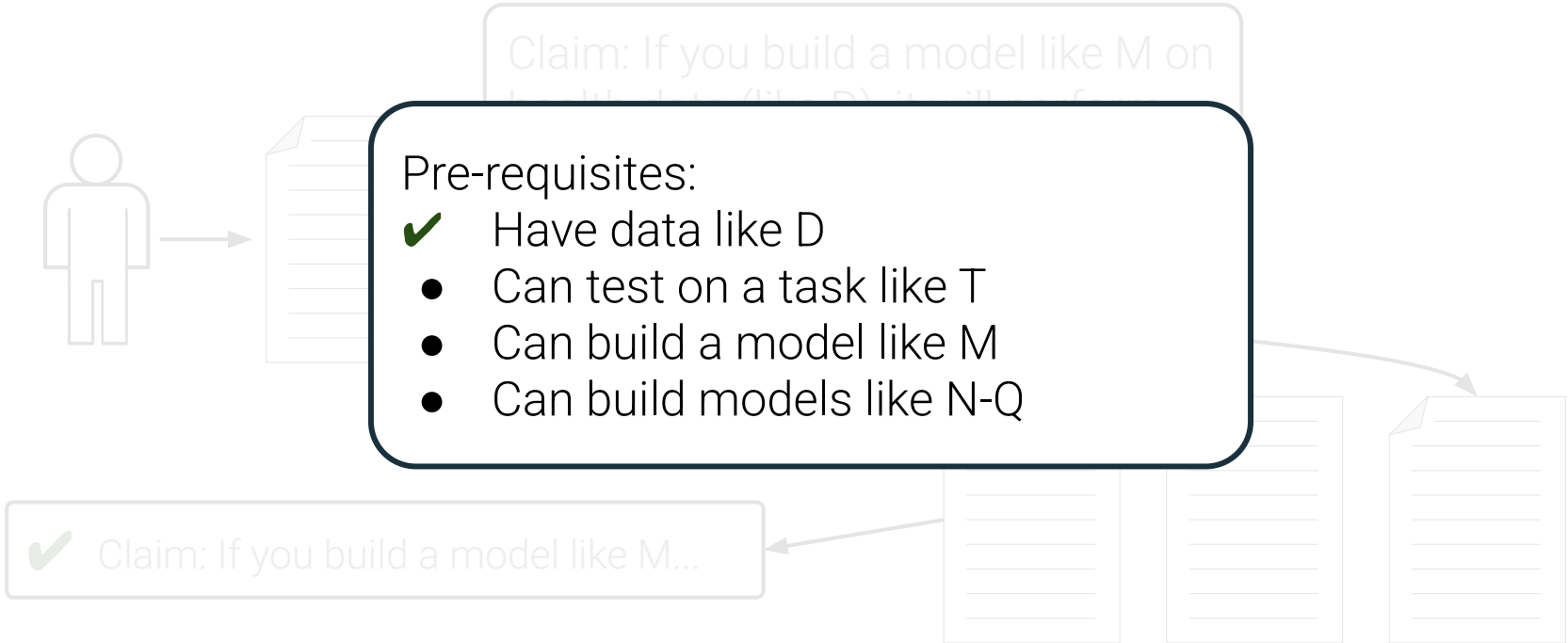
Scientific Process of Health AI: Reproducibility implies that the same training recipes are superior on new health data



What do we need to check reproducibility?



Assume access to data (or public data)



We can’t reliably define prediction tasks!

Pre-requisites:

✔ Have data like D

✘ Can test on a task like T

- Can build a model like M
- Can build models like N-Q

We reproduced datasets for 38 experiments corresponding to 28 published studies using MIMIC. In half of the experiments, the sample size we acquired was 25% greater or smaller than the sample size reported.

- Alistair Johnson et. al., 2017

We can’t reliably re-train models!

Pre-requisites:

✔ Have data like D

✘ Can test on a task like T

✘ Can build a model like M

✘ Can build models like N-Q

✔ Claim: If you build a model like M...

Of the 218 included articles, 73 (34%) shared code, with 24 (33% of code sharing articles and 11% of all articles) sharing reproducible code

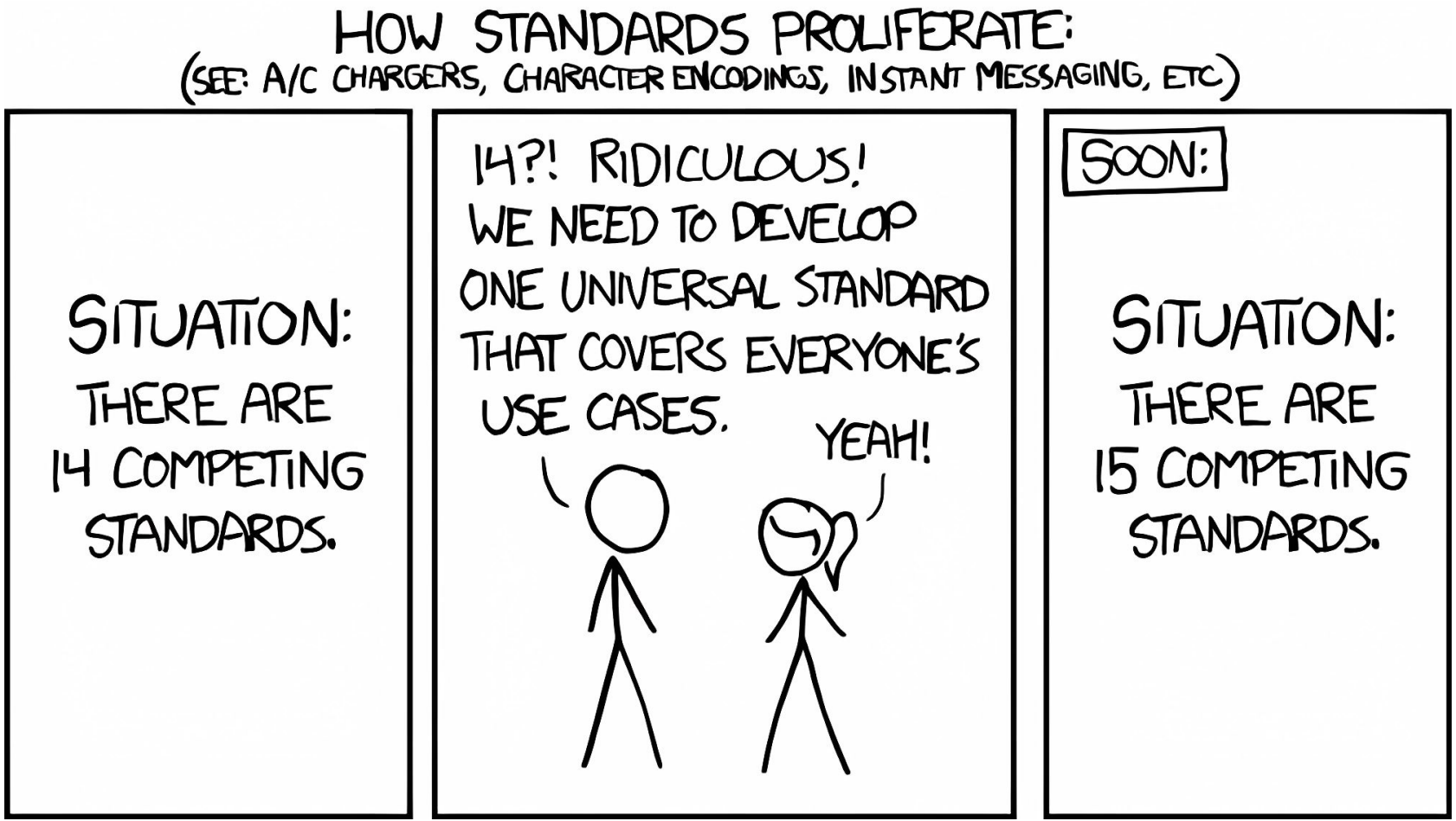
- Kesavan Venkatesh et. al., 2022
https://doi.org/10.1148/ryai.220081

MLH papers scored even more poorly when it came to code release... with only ~21% of the papers we analyzed releasing their code publicly

- Matthew McDermott et. al., 2021
https://doi.org/10.1126/scitranslmed.abb1655

The Health AI “transportability” crisis: We can’t transport our code, tasks, or experiments from one paper to another

Why not use other standards?



Source: https://imgs.xkcd.com/comics/standards.png

OMOP Common Data Model

The Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM) is an open community data standard, designed to standardize the structure and content of observational data and to enable efficient analyses that can produce reliable evidence.

"The OMOP Common Data Model serves as the foundation of all our work in the OHDSI community, and I'm proud that our open community data standard has been so widely adopted and so extensively used to generate reliable evidence."

- Clair Blacketer
2020 Titan Award for Data Standards recipient

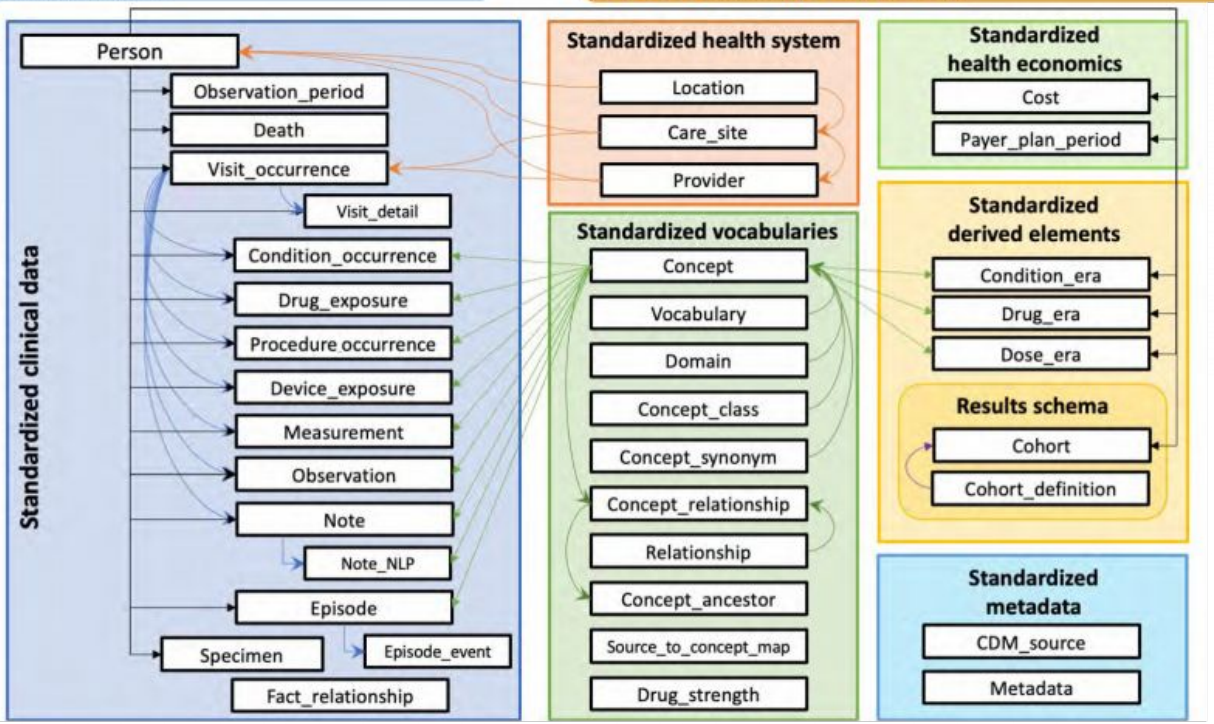
DATA STANDARDS

OMOP CDM By The Numbers

37 tables
- 17 to standardize clinical data
- 10 to standardize vocabularies

394 fields
- 193 with _id to standardize identification
- 101 with _concept_id to standardize content
- 43 with _source_value to preserve original data

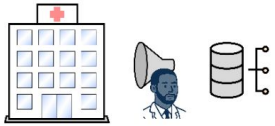
1 Open Community Data Standard



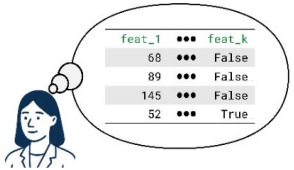
Classical Observational Informatics

Observational informatics communities draw much more from traditional statistics and informatics backgrounds, and often have much greater clinical expertise and more targeted health problems in mind.

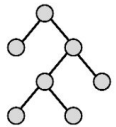
Cohorts in observational informatics studies are typically narrower, and designed via expert input to target the specific problem of interest.



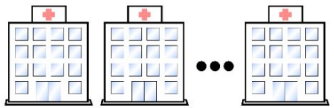
Features are often selected via manual feature engineering efforts to maximize performance, explainability, and robustness for low capacity models.



Models are often lower capacity in traditional observational informatics studies, providing generalizability, robustness, and alignment to understandable clinical mechanisms at the expense of performance.

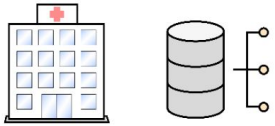


Specific analytic findings or models are expected to generalize across sites of care by virtue of causal, biological grounding.

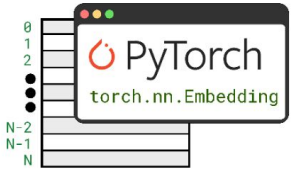


Health AI

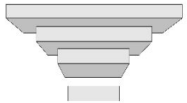
Health AI communities draw heavily from the computer science & AI communities, and leverage the well developed Python ecosystem for AI development in their work. Their research often focuses more on method development, without as concrete a clinical target in mind.



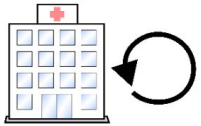
Health AI studies often leverage as much data in as close to the raw form as is possible, so the model can learn relationships directly from the source, across all patients.



Health AI models typically omit manual feature engineering, and represent features via learned embedding layers that are randomly initialized, updated through model training, and cover almost all features in the data.



Models are generally high capacity, with model capacity often only limited by data and computational resource availability. Higher capacity models offer higher performance, but are less explainable and more specialized to local data context.



As models are locally specialized and may be too sensitive for release, models are often best for local use, though as centralized data availability increases, this may change.

MEDS-EIC-AR Model

MEDS_EIC_AR Public

Pin | Unwatch 1 | Fork 0 | Star 1

main | 10 Branches | 13 Tags | Go to file t | Add file | Code

mmcdermott Merge pull request #63 from mmcdermott/codex/fix-issues-... c36f1dd · 2 weeks ago 148 Commits

.github/workflows	Updated PR action. Closes #44.	2 weeks ago
src/MEDS_EIC_AR	Added datamodule config mod for generation to genera...	2 weeks ago
tests	Add non-empty check for generated trajectory DataFram...	2 weeks ago
.gitignore	Initial commit	2 months ago
.pre-commit-config.yaml	Updated pre-commit config to remove shfmt which caus...	2 months ago
LICENSE	Initial commit	2 months ago
README.md	Fixed tests and removed unused sample ID	2 weeks ago
conftest.py	Something's off with dependencies, trying github CI	2 weeks ago
pyproject.toml	Something's off with dependencies, trying github CI	2 weeks ago

README | MIT license

MEDS "Everything-is-code" Autoregressive Model

pypi v0.1.10 | Python 3.12 | codecov 95% | Tests passing | Code Quality Main passing | License MIT | PRs welcome | contributors 1

A MEDS, "Everything-is-code" style Autoregressive Generative Model, capable of zero-shot inference.

This is based on the MEDS-Torch model of the same name.

Installation

```
pip install MEDS-EIC-AR
```

About

A MEDS, "Everything-is-code" style Autoregressive Generative Model, capable of zero-shot inference.

Readme
MIT license
Activity
1 star
1 watching
0 forks

Releases 13

0.1.10 Latest
2 weeks ago

+ 12 releases

Packages

No packages published
Publish your first package

Deployments 13

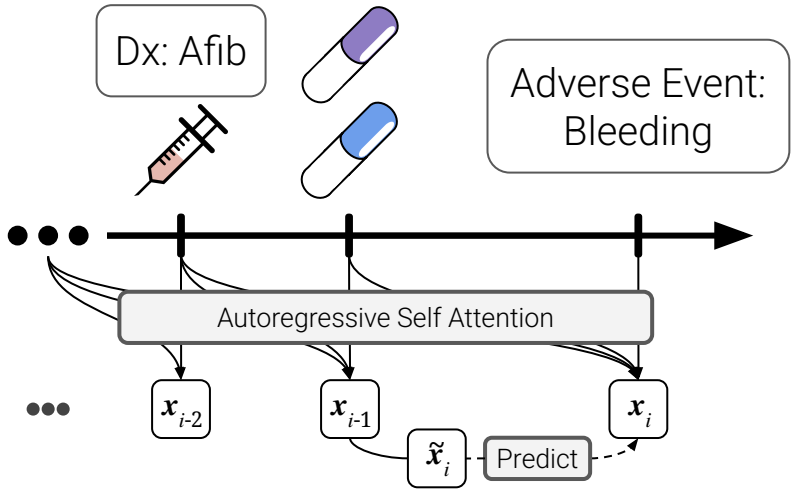
pypi 2 weeks ago

+ 12 deployments

Languages

Python 100.0%

Autoregressive models predict the next MEDS measurement given the patient's history:



Installation and Usage

```
# 1. Install
pip install MEDS-EIC-AR

# 2. Data Pre-processing
MEICAR_process_data input_dir="$RAW_MEDS_DIR" \
    intermediate_dir="$INTERMEDIATE_DIR" \
    output_dir="$FINAL_DATA_DIR"

# 3. Pre-train the model
MEICAR_pretrain datamodule.config.tensorized_cohort_dir="$FINAL_DATA_DIR" \
    output_dir="$PRETRAINED_MODEL_DIR" \
    datamodule.batch_size=32

# 4. Generate Trajectories
MEICAR_generate_trajectories \
    output_dir="$GENERATED_TRAJECTORIES_DIR" \
    model_initialization_dir="$PRETRAINED_MODEL_DIR" \
    datamodule.config.tensorized_cohort_dir="$FINAL_DATA_DIR" \
    datamodule.config.task_labels_dir="$TASK_ROOT_DIR/$TASK_NAME" \
    datamodule.batch_size=32
```

The Data Preprocessing

```
# 2. Data Pre-processing
MEICAR_process_data input_dir="$RAW_MEDS_DIR" \
    intermediate_dir="$INTERMEDIATE_DIR" \
    output_dir="$FINAL_DATA_DIR"
```

```
stages:
  - add_time_derived_measurements:
      timeline_tokens:
        time_unit: years
  - count_codes:
      aggregations:
        - code/n_occurrences
        - code/n_subjects
  - filter_measurements:
      min_subjects_per_code: 1000
  - filter_subjects:
      min_events_per_subject: 10
  - fit_quantile_binning:
      aggregations:
        - name: values/quantiles
          quantiles: [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
  - bin_numeric_values:
      custom_bins:
        TIMELINE//DELTA//years:
          1_minute: 0.00000190258
          5_minutes: 0.00000951293
          10_minutes: 0.00001902587
          30_minutes: 0.00005707762
          ...
          1_year: 1
          2_years: 2
          5_years: 5
          10_years: 10
          20_years: 20
          40_years: 40
```

README MIT license

MEDS Transforms: Build and run complex pipelines over MEDS datasets via simple parts

pypi v0.5.1 | Python 3.12 | MEDS 0.4 | docs passing | codecov 100% | Tests passing | Code Quality Main passing | Config Hydra 1.3 | License MIT | PRs welcome | contributors 7

🚀 Quick Start

1. Install via `pip`:

```
pip install MEDS-transforms
```

2. Craft a pipeline YAML file:

```yaml
input_dir: $MEDS_ROOT
output_dir: $PIPELINE_OUTPUT

description: Your special pipeline

stages:
  - filter_subjects:
      min_events_per_subject: 5
  - add_time_derived_measurements:
      age:
        DOB_code: MEDS_BIRTH
        age_code: AGE
        age_unit: years
      time_of_day:
        time_of_day_code: TIME_OF_DAY
        endpoints: [6, 12, 18, 24]
  - fit_outlier_detection:
      _base_stage: aggregate_code_metadata
      aggregations:
        - values/n_occurrences
        - values/sum
        - values/sum_sqd
  - occlude_outliers:
      stddev_cutoff: 1
  - fit_normalization:
      _base_stage: aggregate_code_metadata
      aggregations:
        - code/n_occurrences
        - code/n_subjects
        - values/n_occurrences
        - values/sum
        - values/sum_sqd
  - fit_vocabulary_indices
  - normalization
```

Save your pipeline YAML file on disk at `$PIPELINE_YAML`.

3. Run the pipeline

In the terminal, run

```
MEDS_transform-pipeline pipeline_fp="$PIPELINE_YAML"
```

After you do, you will see output files stored in `$PIPELINE_OUTPUT` with the results of each stage of the pipeline, stored in stage specific directories, and the global output in `$PIPELINE_OUTPUT/data` and `$PIPELINE_OUTPUT/metadata` (for data and metadata outputs, respectively). That's it!

The Model

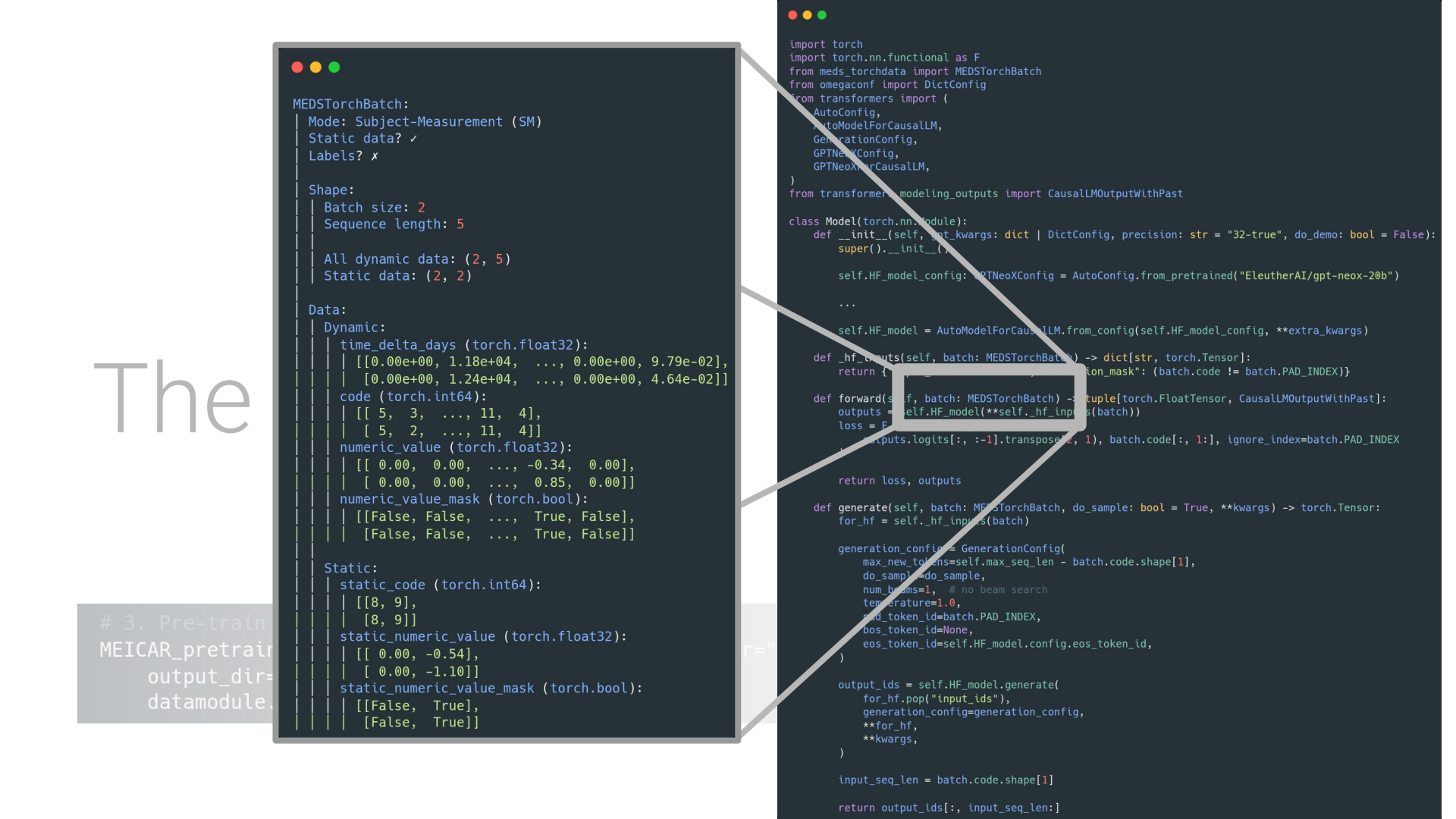
```
# 3. Pre-train the model
MEICAR_pretrain datamodule.config.tensorized_cohort_dir="
    output_dir="$PRETRAINED_MODEL_DIR" \
    datamodule.batch_size=32
```

```python
import torch
import torch.nn.functional as F
from meds_torchdata import MEDSTorchBatch
from omegaconf import DictConfig
from transformers import (
    AutoConfig,
    AutoModelForCausalLM,
    GenerationConfig,
    GPTNeoXConfig,
    GPTNeoXForCausalLM,
)
from transformers.modeling_outputs import CausalLMOutputWithPast

class Model(torch.nn.Module):
    def __init__(self, gpt_kwargs: dict | DictConfig, precision: str = "32-true", do_demo: bool = False):
        super().__init__()

        self.HF_model_config: GPTNeoXConfig = AutoConfig.from_pretrained("EleutherAI/gpt-neox-20b")

        ...

        self.HF_model = AutoModelForCausalLM.from_config(self.HF_model_config, **extra_kwargs)

    def _hf_inputs(self, batch: MEDSTorchBatch) -> dict[str, torch.Tensor]:
        return {"input_ids": batch.code, "attention_mask": (batch.code != batch.PAD_INDEX)}

    def forward(self, batch: MEDSTorchBatch) -> tuple[torch.FloatTensor, CausalLMOutputWithPast]:
        outputs = self.HF_model(**self._hf_inputs(batch))
        loss = F.cross_entropy(
            outputs.logits[:, :-1].transpose(2, 1), batch.code[:, 1:], ignore_index=batch.PAD_INDEX
        )

        return loss, outputs

    def generate(self, batch: MEDSTorchBatch, do_sample: bool = True, **kwargs) -> torch.Tensor:
        for_hf = self._hf_inputs(batch)

        generation_config = GenerationConfig(
            max_new_tokens=self.max_seq_len - batch.code.shape[1],
            do_sample=do_sample,
            num_beams=1,  # no beam search
            temperature=1.0,
            pad_token_id=batch.PAD_INDEX,
            bos_token_id=None,
            eos_token_id=self.HF_model.config.eos_token_id,
        )

        output_ids = self.HF_model.generate(
            for_hf.pop("input_ids"),
            generation_config=generation_config,
            **for_hf,
            **kwargs,
        )

        input_seq_len = batch.code.shape[1]

        return output_ids[:, input_seq_len:]
```

The Model

```
# 3. Pre-train the model
MEICAR_pretrain datamodule.config.tensorized_cohort_dir="
    output_dir="$PRETRAINED_MODEL_DIR" \
    datamodule.batch_size=32
```

```python
import torch
import torch.nn.functional as F
from meds_torchdata import MEDSTorchBatch
from omegaconf import DictConfig
from transformers import (
    AutoConfig,
    AutoModelForCausalLM,
    GenerationConfig,
    GPTNeoXConfig,
    GPTNeoXForCausalLM,
)
from transformers.modeling_outputs import CausalLMOutputWithPast

class Model(torch.nn.Module):
    def __init__(self, gpt_kwargs: dict | DictConfig, precision: str = "32-true", do_demo: bool = False):
        super().__init__()

        self.HF_model_config: GPTNeoXConfig = AutoConfig.from_pretrained("EleutherAI/gpt-neox-20b")

        ...

        self.HF_model = AutoModelForCausalLM.from_config(self.HF_model_config, **extra_kwargs)

    def _hf_inputs(self, batch: MEDSTorchBatch) -> dict[str, torch.Tensor]:
        return {"                                    ion_mask": (batch.code != batch.PAD_INDEX)}

    def forward(self, batch: MEDSTorchBatch) -> tuple[torch.FloatTensor, CausalLMOutputWithPast]:
        outputs = self.HF_model(**self._hf_inputs(batch))
        loss = F.
            outputs.logits[:, :-1].transpose(2, 1), batch.code[:, 1:], ignore_index=batch.PAD_INDEX
        )

        return loss, outputs

    def generate(self, batch: MEDSTorchBatch, do_sample: bool = True, **kwargs) -> torch.Tensor:
        for_hf = self._hf_inputs(batch)

        generation_config = GenerationConfig(
            max_new_tokens=self.max_seq_len - batch.code.shape[1],
            do_sample=do_sample,
            num_beams=1,  # no beam search
            temperature=1.0,
            pad_token_id=batch.PAD_INDEX,
            bos_token_id=None,
            eos_token_id=self.HF_model.config.eos_token_id,
        )

        output_ids = self.HF_model.generate(
            for_hf.pop("input_ids"),
            generation_config=generation_config,
            **for_hf,
            **kwargs,
        )

        input_seq_len = batch.code.shape[1]

        return output_ids[:, input_seq_len:]
```


The
MEDSTorchBatch:
Mode: Subject-Measurement (SM)
Static data? ✓
Labels? ✗
Shape:
Batch size: 2
Sequence length: 5
All dynamic data: (2, 5)
Static data: (2, 2)
Data:
Dynamic:
time_delta_days (torch.float32):
[[0.00e+00, 1.18e+04, ..., 0.00e+00, 9.79e-02],
[0.00e+00, 1.24e+04, ..., 0.00e+00, 4.64e-02]]
code (torch.int64):
[[5, 3, ..., 11, 4],
[5, 2, ..., 11, 4]]
numeric_value (torch.float32):
[[0.00, 0.00, ..., -0.34, 0.00],
[0.00, 0.00, ..., 0.85, 0.00]]
numeric_value_mask (torch.bool):
[[False, False, ..., True, False],
[False, False, ..., True, False]]
Static:
static_code (torch.int64):
[[8, 9],
[8, 9]]
static_numeric_value (torch.float32):
[[0.00, -0.54],
[0.00, -1.10]]
static_numeric_value_mask (torch.bool):
[[False, True],
[False, True]]
import torch
import torch.nn.functional as F
from meds_torchdata import MEDSTorchBatch
from omegaconf import DictConfig
from transformers import (
AutoConfig,
AutoModelForCausalLM,
GenerationConfig,
GPTNeoXConfig,
GPTNeoXForCausalLM,
)
from transformers.modeling_outputs import CausalLMOutputWithPast
class Model(torch.nn.Module):
def __init__(self, gpt_kwargs: dict | DictConfig, precision: str = "32-true", do_demo: bool = False):
super().__init__()
self.HF_model_config: GPTNeoXConfig = AutoConfig.from_pretrained("EleutherAI/gpt-neox-20b")
...
self.HF_model = AutoModelForCausalLM.from_config(self.HF_model_config, **extra_kwargs)
def _hf_inputs(self, batch: MEDSTorchBatch) -> dict[str, torch.Tensor]:
return {
def forward(self, batch: MEDSTorchBatch) -> tuple[torch.FloatTensor, CausalLMOutputWithPast]:
outputs = self.HF_model(**self._hf_inputs(batch))
loss = F
outputs.logits[:, :-1].transpose(2, 1), batch.code[:, 1:], ignore_index=batch.PAD_INDEX
return loss, outputs
def generate(self, batch: MEDSTorchBatch, do_sample: bool = True, **kwargs) -> torch.Tensor:
for_hf = self._hf_inputs(batch)
generation_config = GenerationConfig(
max_new_tokens=self.max_seq_len - batch.code.shape[1],
do_sample=do_sample,
num_beams=1, # no beam search
temperature=1.0,
pad_token_id=batch.PAD_INDEX,
bos_token_id=None,
eos_token_id=self.HF_model.config.eos_token_id,
)
output_ids = self.HF_model.generate(
for_hf.pop("input_ids"),
generation_config=generation_config,
**for_hf,
**kwargs,
)
input_seq_len = batch.code.shape[1]
return output_ids[:, input_seq_len:]
3. Pre-train
MEICAR_pretrain
output_dir=
datamodule.
r="

README License

MEDS TorchData: A PyTorch Dataset Class for MEDS Datasets

🚀 Quick Start

Step 1: Install

```
pip install meds-torch-data
```

Step 2: Data Tensorization

> ⚠ Warning
>
> If your dataset is not sharded by split, you need to run a reshard to split stage first! You can enable this by adding the `do_reshard=True` argument to the command below.

If your input MEDS dataset lives in `$MEDS_ROOT` and you want to store your pre-processed files in `$PYD_ROOT`, you run:

```
MTD_preprocess MEDS_dataset_dir="$MEDS_ROOT" output_dir="$PYD_ROOT"
```

Step 3: Use the dataset

To use a dataset, you need to (1) define your configuration object and (2) create the dataset object. The only required configuration parameters are `tensorized_cohort_dir`, which points to the root directory containing the pre-processed data on disk (`$PYD_ROOT` in the above example), and `max_seq_len`, which is the maximum sequence length you want to use for your model. Here's an example:

```
import os
from meds_torchdata import MEDSPytorchDataset, MEDSTorchDataConfig

cfg = MEDSTorchDataConfig(tensorized_cohort_dir=os.environ["PYD_ROOT"], max_seq_len=512)
pyd = MEDSPytorchDataset(cfg, split="train")
```

README License

orchData: A PyTorch Dataset Class for MEDS
ts

thon 3.11+ | pypi v0.6.1 | MEDS 0.4 | docs passing | Tests passing | codecov 100%
in passing | Config Hydra 1.3 | contributors 7 | PRs welcome | License MIT

Start

all

```
meds-torch-data
```

a Tensorization

set is not sharded by split, you need to run a reshard to split stage first! You can enable this by
`do_reshard=True` argument to the command below.

If your input MEDS dataset lives in `$MEDS_ROOT` and you want to store your pre-processed files in `$PYD_ROOT`, you run:

```
MTD_preprocess MEDS_dataset_dir="$MEDS_ROOT" output_dir="$PYD_ROOT"
```

Step 3: Use the dataset

To use a dataset, you need to (1) define your configuration object and (2) create the dataset object. The only required configuration parameters are `tensorized_cohort_dir`, which points to the root directory containing the pre-processed data on disk (`$PYD_ROOT` in the above example), and `max_seq_len`, which is the maximum sequence length you want to use for your model. Here's an example:

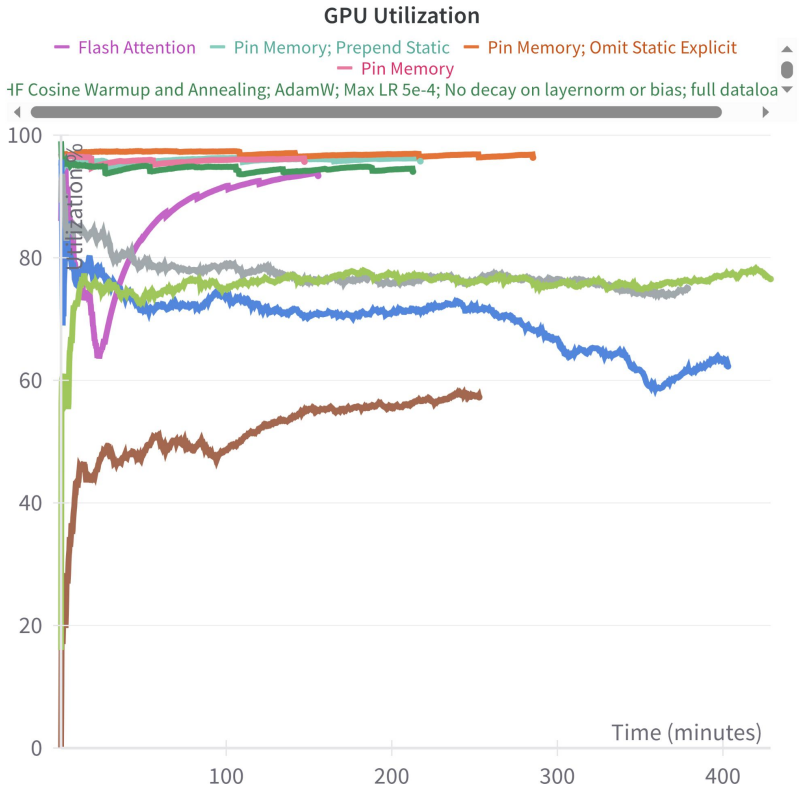
```
import os
from meds_torchdata import MEDSPytorchDataset, MEDSTorchDataConfig

cfg = MEDSTorchDataConfig(tensorized_cohort_dir=os.environ["PYD_ROOT"], max_seq_len=512)
pyd = MEDSPytorchDataset(cfg, split="train")
```

```
stages:
  - fit_normalization:
      aggregations:
        - "code/n_occurrences"
        - "code/n_subjects"
        - "values/n_occurrences"
        - "values/sum"
        - "values/sum_sqd"
  - fit_vocabulary_indices
  - normalization
  - tokenization
  - tensorization
```

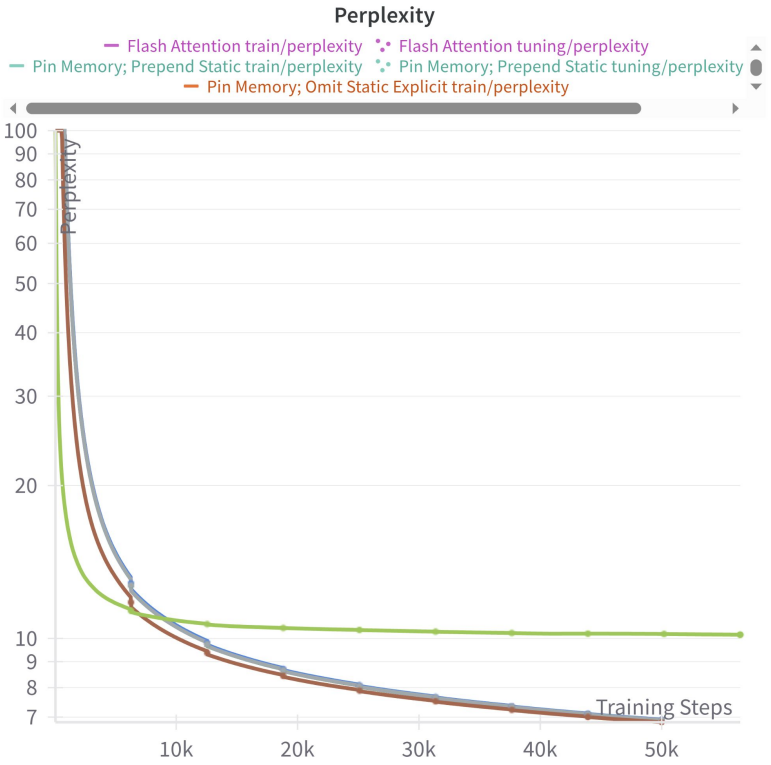
Batches

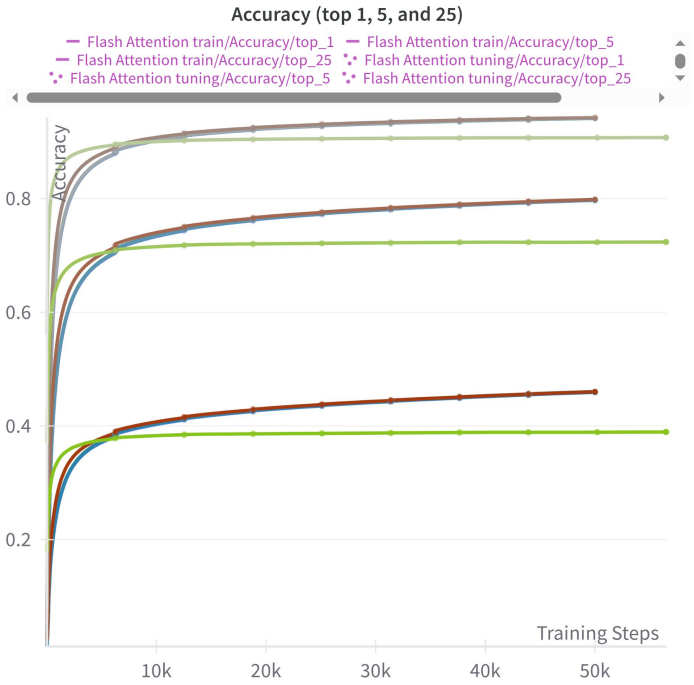
```
MEDSTorchBatch:
| Mode: Subject-Measurement (SM)
| Static data? ✓
| Labels? ✗
|
| Shape:
| | Batch size: 2
| | Sequence length: 5
| |
| | All dynamic data: (2, 5)
| | Static data: (2, 2)
|
| Data:
| | Dynamic:
| | | time_delta_days (torch.float32):
| | | | [[0.00e+00, 1.18e+04,  ..., 0.00e+00, 9.79e-02],
| | | |  [0.00e+00, 1.24e+04,  ..., 0.00e+00, 4.64e-02]]
| | | code (torch.int64):
| | | | [[ 5,  3,  ..., 11,  4],
| | | |  [ 5,  2,  ..., 11,  4]]
| | | numeric_value (torch.float32):
| | | | [[ 0.00,  0.00,  ..., -0.34,  0.00],
| | | |  [ 0.00,  0.00,  ...,  0.85,  0.00]]
| | | numeric_value_mask (torch.bool):
| | | | [[False, False,  ...,  True, False],
| | | |  [False, False,  ...,  True, False]]
| | |
| | Static:
| | | static_code (torch.int64):
| | | | [[8, 9],
| | | |  [8, 9]]
| | | static_numeric_value (torch.float32):
| | | | [[ 0.00, -0.54],
| | | |  [ 0.00, -1.10]]
| | | static_numeric_value_mask (torch.bool):
| | | | [[False,  True],
| | | |  [False,  True]]
```

GPU Utilization (Training Speed)

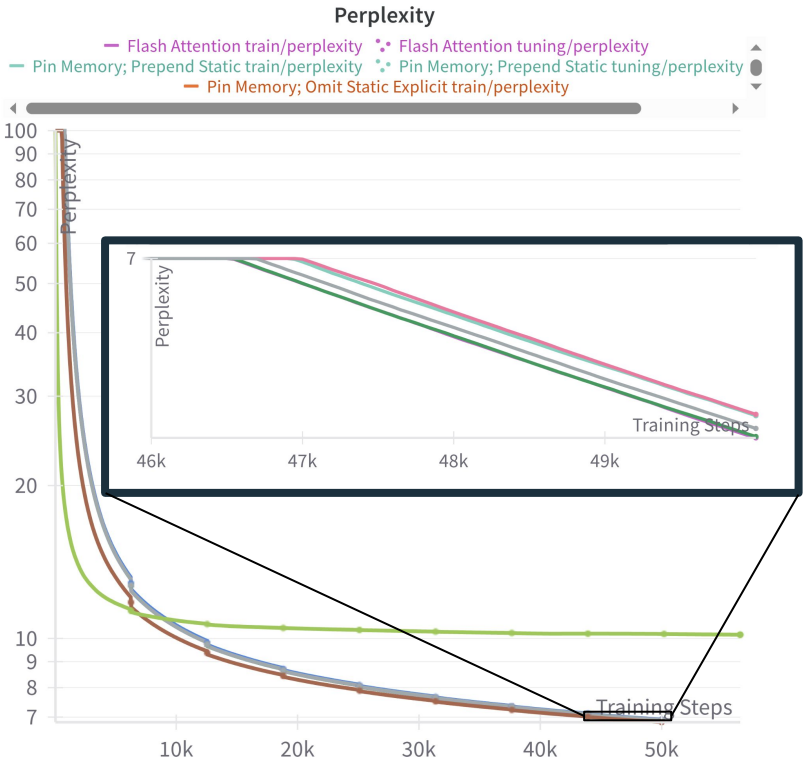


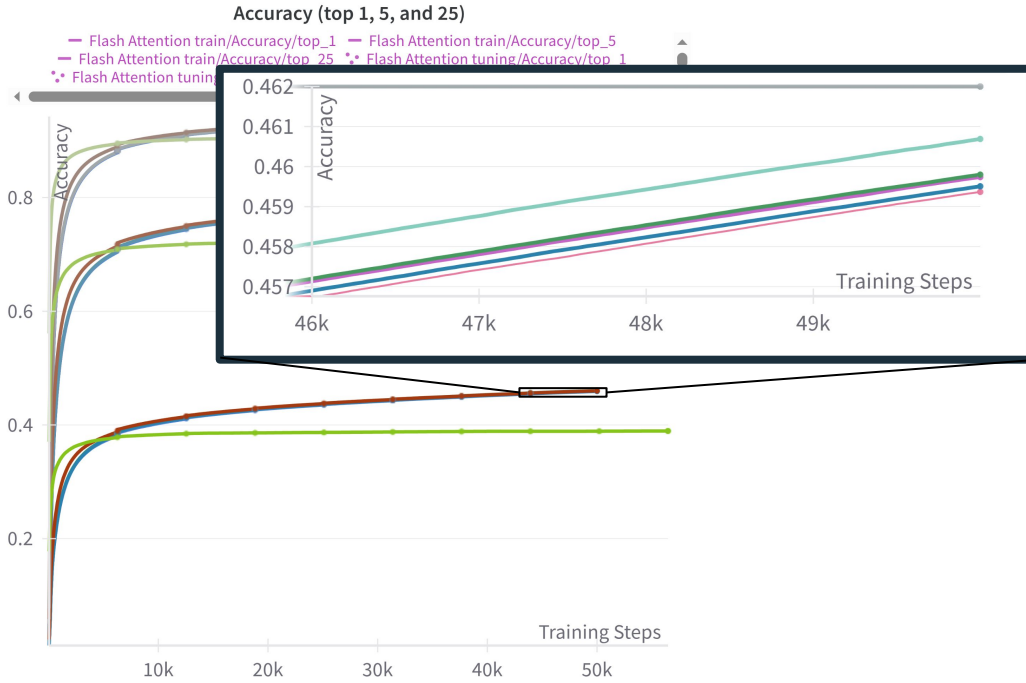
Training Convergence and Stability





Training Convergence and Stability





Determining Parameters?

Determining Parameters?

```
>>> print_directory("./src/MEDS_EIC_AR/configs", config=PrintConfig(file_extension=".yaml"))
├── _demo_generate_trajectories.yaml
├── _demo_pretrain.yaml
├── _generate_trajectories.yaml
├── _pretrain.yaml
├── datamodule
│   ├── default.yaml
│   ├── generate_trajectories.yaml
│   └── pretrain.yaml
├── inference
│   ├── default.yaml
│   └── demo.yaml
├── lightning_module
│   ├── LR_scheduler
│   │   ├── cosine_annealing_warm_restarts.yaml
│   │   ├── get_cosine_schedule_with_warmup.yaml
│   │   ├── one_cycle_LR.yaml
│   │   └── reduce_LR_on_plateau.yaml
│   ├── default.yaml
│   ├── demo.yaml
│   ├── metrics
│   │   └── default.yaml
│   ├── model
│   │   ├── default.yaml
│   │   ├── demo.yaml
│   │   └── small.yaml
│   └── optimizer
│       ├── adam.yaml
│       └── adamw.yaml
└── trainer
    ├── callbacks
    │   ├── default.yaml
    │   ├── early_stopping.yaml
    │   ├── learning_rate_monitor.yaml
    │   └── model_checkpoint.yaml
    ├── default.yaml
    ├── demo.yaml
    └── logger
        ├── csv.yaml
        ├── mlflow.yaml
        └── wandb.yaml
```

Other Tool Posters

MEDS-Reader

A blazing fast EHR processing library

Ethan Steinberg, Michael Wornow, Suhana Bedi, Jason Alan Fries, Matthew B. A. McDermott, Nigam Shah

prealize · Stanford University · HARVARD MEDICAL SCHOOL

Introduction

With the rise of foundation models, EHR research needs to process increasing amounts of data. However, existing tooling, which is optimized for tabular data, images, and text, has not kept up.

To fix this we introduce meds_reader, a blazing fast EHR processing library.

Contributions

We introduce meds_reader, which has:

- A simple Python interface, easily installable with pip
- A flexible interface, usable for
  Featurization, labeling, tensorization, and much more!
- 10-100x speed and memory improvements over alternatives
- Support for a variety of EHR datasets through the MEDS standard

Links

Demo: bit.ly/4ir2vbg

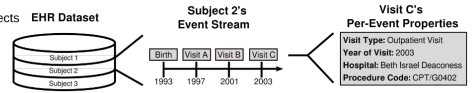
Code: bit.ly/49memmF

EHRs as Event Streams

EHR Dataset = Collection of subjects

Subject = Series of events

Event = Collection of properties



Python API

Core Design:
Subject transformation functions

Features:

- Parallelization across threads
- Uses incremental algorithms
- Has a C++ backend
- Supports arbitrary properties
- Memory and disk compression

pip install meds_reader

```python
def find_stroke(subjects):
    subject_ids = set()
    for subject in subjects:
        for event in subject.events:
            if event.code.startswith('ICD10/I63'):
                subject_ids.add(subject.subject_id)
    return subject_ids

subject_database = meds_reader.SubjectDatabase('...', num_threads=10)
all_subjects = set()
for partial_result in subject_database.map(find_stroke):
    all_subjects |= partial_result
```

Benchmarks

We can test the performance of meds_reader by reimplementing two other data processing pipelines.

We observe huge improvements in memory, speed and disk usage.

Dataset	#Subjects	Library	Memory (GB) ↓			CPU (secs) ↓			Disk (GB) ↓		
			Orig.	Ours	Δ×	Orig.	Ours	Δ×	Orig.	Ours	Δ×
eICU	200,859	PyHealth	16.80	**0.67**	25.1×	70.44	**1.91**	36.9×	1.4	**0.095**	14.7×
MIMIC-III	46,520	PyHealth	20.97	**0.54**	38.8×	57.55	**1.58**	36.4×	1.8	**0.042**	42.9×
MIMIC-IV	299,712	MDP	14.32	**2.09**	6.9×	293.261	**7.44**	39.4k×	0.64	**0.33**	1.9×

Table 1: Compute resources used by various data processing pipelines compared to our reimplementations using **meds_reader**. We measure peak memory usage, wall clock CPU time, and the disk space used for cache directories. We compare the original resources used to the reimplementation and report the multiplicative change (Δ×). Orig. = Original pipeline. MDP = MIMIC-IV Data Pipeline.

Dataset Support

Supports any dataset has been converted to MEDS (Medical Event Data Standard)

OMOP · MIMIC-IV · eICU · EHRSHOT · Your Custom Dataset

README · MIT license

MEDS Transforms: Build and run complex pipelines over MEDS datasets via simple parts

pypi v0.5.1 · Python 3.12 · MEDS 0.4 · docs passing · codecov 100% · Tests passing · Code Quality Main passing · Config Hydra 1.3 · License MIT · PRs welcome · contributors 7

🚀 Quick Start

1. Install via `pip`:

```
pip install MEDS-transforms
```

2. Craft a pipeline YAML file:

```yaml
input_dir: $MEDS_ROOT
output_dir: $PIPELINE_OUTPUT

description: Your special pipeline

stages:
  - filter_subjects:
      min_events_per_subject: 5
  - add_time_derived_measurements:
      age:
        DOB_code: MEDS_BIRTH
        age_code: AGE
        age_unit: years
      time_of_day:
        time_of_day_code: TIME_OF_DAY
        endpoints: [6, 12, 18, 24]
  - fit_outlier_detection:
      _base_stage: aggregate_code_metadata
      aggregations:
        - values/n_occurrences
        - values/sum
        - values/sum_sqd
  - occlude_outliers:
      stddev_cutoff: 1
  - fit_normalization:
      _base_stage: aggregate_code_metadata
      aggregations:
        - code/n_occurrences
        - code/n_subjects
        - values/n_occurrences
        - values/sum
        - values/sum_sqd
  - fit_vocabulary_indices
  - normalization
```

MEDS-Torch: A Machine Learning Pipeline for Inductive Experiments on EHR Data

Nassim Oufattole, Teya Bergamaschi, Pawel Renc, Aleksia Kolo, Matthew B.A. McDermott, Collin Stultz

Paper: Demo:

Streamlining Health AI Benchmarking

Challenges of Reproducible benchmarking across health AI systems

1. Complex data preprocessing requirements (↑ User Burden)
2. Efficient loading of large irregularly sampled Time Series data

MEDS-Torch Key Features

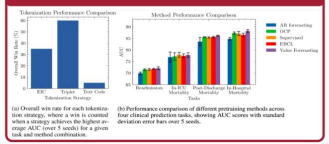
- Multiple tokenization and pretraining methods to enable benchmarking
- Generalizability across datasets via the MEDS data standard
- Minimal code and user burden
- GPU-optimized processing via MEDS-torch Tensorization
- Memory and CPU efficient Dataloaders for contrastive learning, supervised, and forecasting workflows

MEDS-Torch Usage

Stage 1 (Tensorization)
What your need:
- Data in MEDS format
- Task labels

Stage 2 (Training/Prediction)
What you need:
- Tensorized Data
- Task labels

MEDS-Torch Results

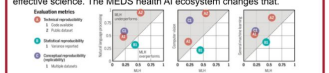


(a) Overall win rate for each tokenization strategy, where a win is counted when a method achieves the highest average AUC (over 3 seeds) for a given task and method combination.

(b) Performance comparison of different pretraining methods across four clinical prediction tasks, showing AUC scores with standard deviation error bars over 5 seeds.

The Science of ML in Healthcare

Health AI faces a systemic reproducibility crisis, limiting our ability to do effective science. The MEDS health AI ecosystem changes that.

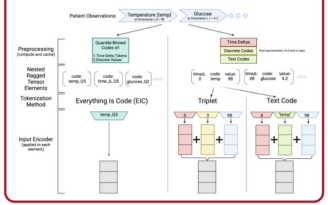


- MEDS enables a robust ecosystem of health AI tools.
- This allows groups to share software, models, & results!
- Check out the link below to learn more!

To Learn More About MEDS Visit:

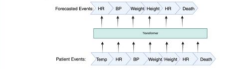
Supported Tokenization Methods

We support (1) Everything is Code, (2) Triplet, and (3) Text Code

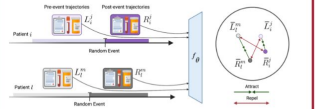


Supported Transfer Learning Methods

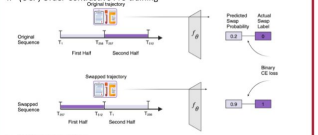
1. Supervised Baseline
2. Autoregressive Next Observation Forecasting



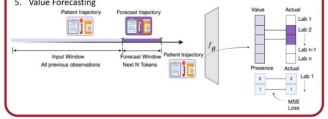
3. (EBCL) Event-Based Contrastive Learning



4. (OCP) Order-Contrastive Pre-training



5. Value Forecasting



Future Roadmap

- Multimodal Support (+ ECGs, Notes, etc.)
- Zero-Shot Autoregressive Generation and Evaluations
- CPU and Memory Benchmarking Scripts

README | License

MEDS TorchData: A PyTorch Dataset Class for MEDS Datasets

PyTorch | Python 3.11+ | pypi v0.6.1 | MEDS 0.4 | docs passing | Tests passing | codecov 100%
Code Quality Main passing | Config Hydra 1.3 | contributors 7 | PRs welcome | License MIT

🚀 Quick Start

Step 1: Install

```
pip install meds-torch-data
```

Step 2: Data Tensorization

> ⚠ Warning
>
> If your dataset is not sharded by split, you need to run a reshard to split stage first! You can enable this by adding the `do_reshard=True` argument to the command below.

If your input MEDS dataset lives in `$MEDS_ROOT` and you want to store your pre-processed files in `$PYD_ROOT`, you run:

```
MTD_preprocess MEDS_dataset_dir="$MEDS_ROOT" output_dir="$PYD_ROOT"
```

Step 3: Use the dataset

To use a dataset, you need to (1) define your configuration object and (2) create the dataset object. The only required configuration parameters are `tensorized_cohort_dir`, which points to the root directory containing the pre-processed data on disk (`$PYD_ROOT` in the above example), and `max_seq_len`, which is the maximum sequence length you want to use for your model. Here's an example:

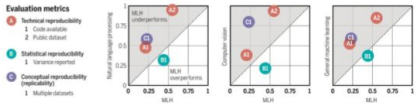
```
import os
from meds_torchdata import MEDSPytorchDataset, MEDSTorchDataConfig

cfg = MEDSTorchDataConfig(tensorized_cohort_dir=os.environ["PYD_ROOT"], max_seq_len=512)
pyd = MEDSPytorchDataset(cfg, split="train")
```

MEDS-DEV: Enabling Reproducible Science of AI for EHR Data

Aleksia Kolo, Chao Pang, Edward Choi, Ethan Steinberg, Hyewon Jeong, Jack Gallifant, Jason A. Fries, Jeffrey N. Chiang, Jungwoo Oh, Justin Xu, Kamilė Stankevičiūtė, Kiril V. Klein, Matthew McDermott, Mikkel Odgaard, Nassim Oufattole, Patrick Rockenschaub, Pawel Renc, Robin van de Water, Shalmali Joshi, Simon A. Lee, Teya S. Bergamaschi, Tom J. Pollard, Vincent Jeanselme, Young Sang Choi

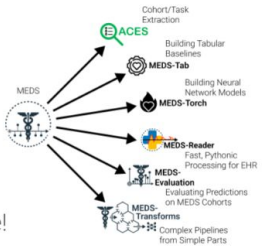
Health AI has a Reproducibility Crisis

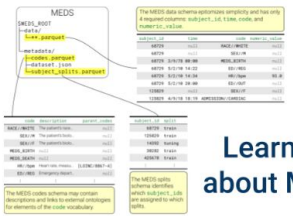
Health AI faces a systemic reproducibility crisis, limiting our ability to do effective science. The MEDS health AI ecosystem, including ACES and MEDS-DEV, changes that.



MEDS: A Health AI Data Ecosystem

- MEDS enables a robust ecosystem of health AI tools.
- This allows groups to share software, models, & results!
- Check out the link below to learn more!





- MEDS is simple and easy to use.
- Converting to MEDS is easy!

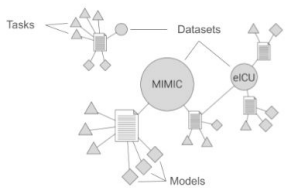
Learn More about MEDS:

MEDS-DEV
Decentralized Extensible Validation

Meaningful shared tasks in healthcare require:

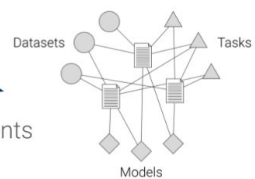
- Decentralized evaluation over private data.
- *Frictionlessly reproducible* task definitions, data preprocessing pipelines, model training recipes, and evaluation protocols.
- A community curated set of tasks spanning clinical problems, settings, and data needs.

MEDS-DEV enables meaningful benchmarking and reproducible AI in ML4H by satisfying these needs.



Benchmarks today are disconnected and irreproducible, leading to uninformative results and a lack of progress.

MEDS-DEV enables more interconnected, comparable experiments and better science



How do you use MEDS-DEV?

Need	Tool
Reproducible, transparent task definitions	ACES task configs are transparent, reproducible, & can be shared across datasets
Shared evaluation protocols	MEDS-Evaluation ensures evaluation is consistent and aspects such as fairness and equity can be included
Communal contributions on tasks & models Decentralized, extensible data and task support	MEDS-DEV configuration files and results are stored & versioned in an open-source GitHub, enabling communal development and decentralization of results

- Adding new results and models is easy!
- Preliminary results showcase adding new models, tasks, and datasets!

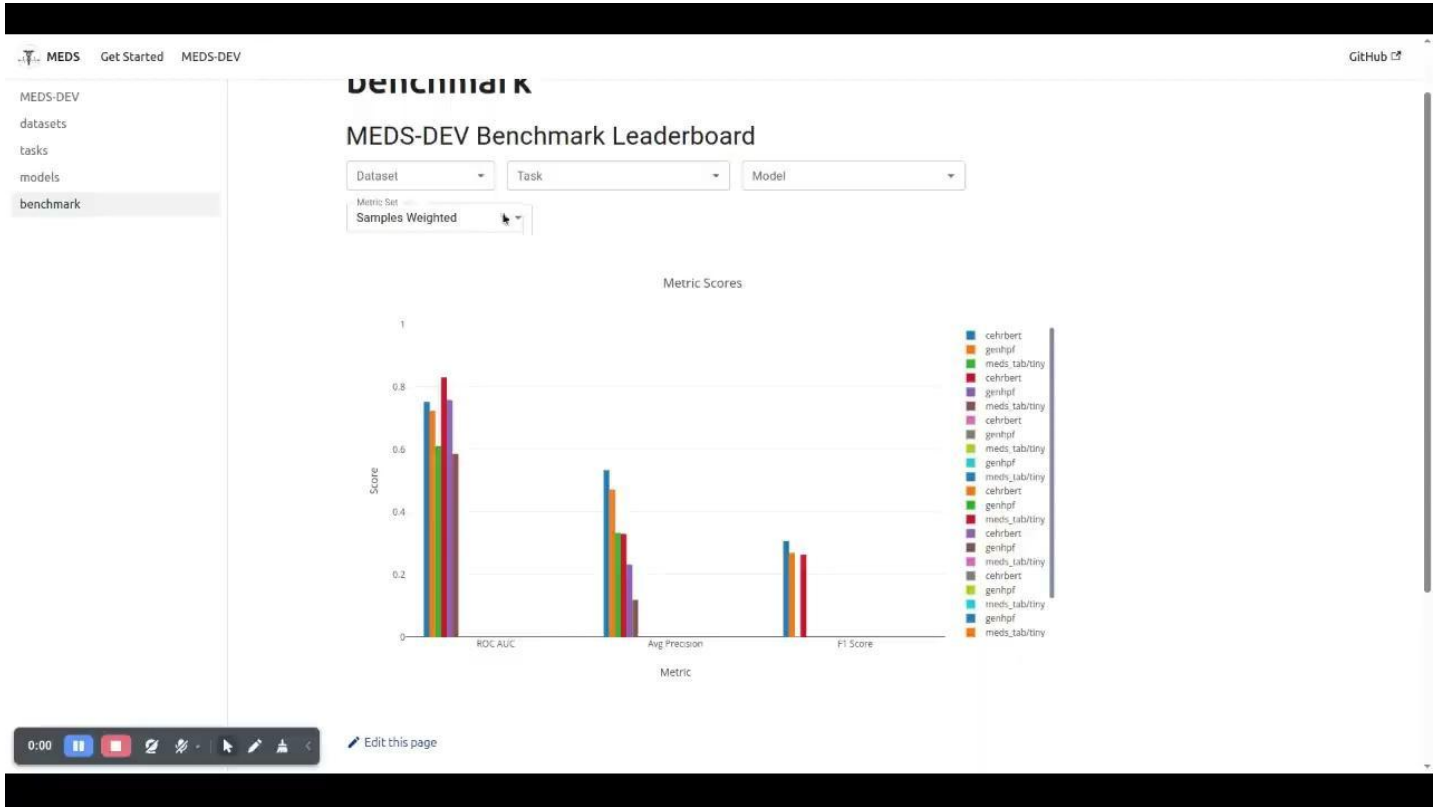
Model	MIMIC-IV/Columbia Long LOS	MIMIC-IV/Columbia ICU Mortality
Log. Reg.	0.752/0.677	0.754/0.509
LightGBM	0.783/0.757	0.798/0.661
MOTOR	0.804/0.735	**0.854**/0.727
CEHR-BERT	0.808/0.741	0.845/0.726
MEDS-Tab	**0.811**/**0.761**	0.830/**0.785**
GenHPF	0.779/0.662	0.790/0.633

Check out detailed MEDS-DEV Tutorials Here:

Acknowledgements

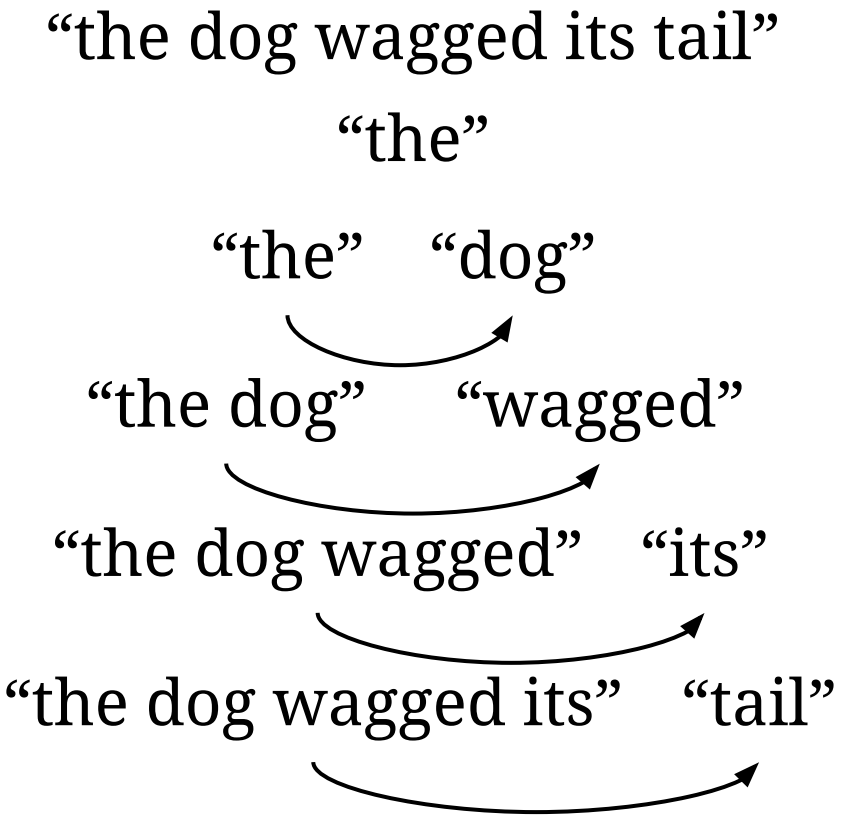
MBAM gratefully acknowledges support from a Berkowitz Postdoctoral Fellowship; KS acknowledges support from AstraZeneca; JAF acknowledges support for this effort from the Debra and Mark Leslie fund for supporting AI in Healthcare research; TP is supported by the National Institutes of Health (NIH OT2OD032701 and NIH R01EB030362); Edward Choi is supported by the IITP grant (No.RS-2019-II190075) and the NRF grant (NRF-2020H1D3A2A03100945) funded by the Korean government; Part of the PHAIR project supported by the Innovation Fund Denmark; NHS acknowledges support for this effort from the Debra and Mark Leslie fund for supporting AI in Healthcare researchPioneer Centre for AI, DNRF Grant P1; Novo Nordisk Foundation

Sharing Results

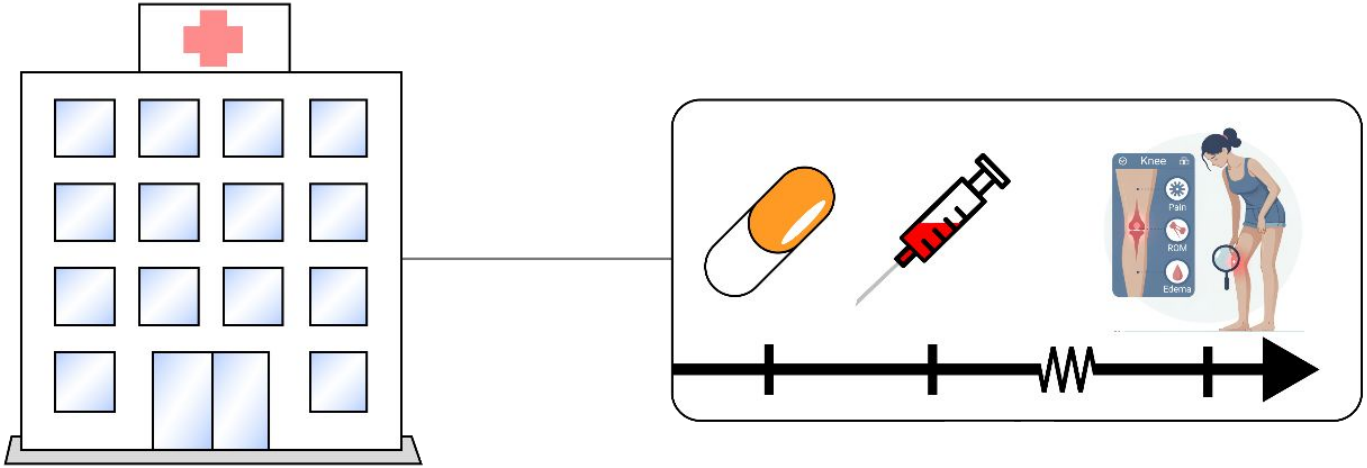


Language Models:
What word comes next?

Language Models: What word comes next?

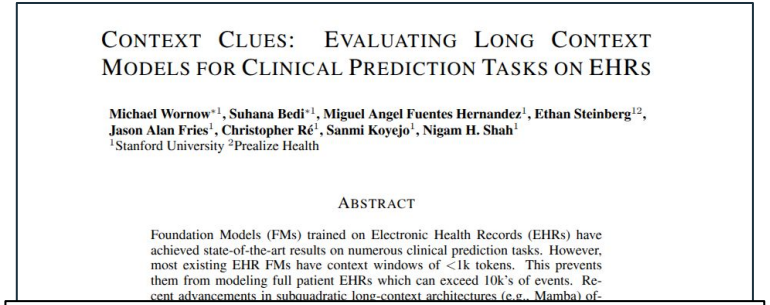


EHR Data is a Continuous Longitudinal Timeline of Complex Observations



Autoregressive Foundation Models: What observation comes next?

Such models have been widely validated in the literature



CONTEXT CLUES: EVALUATING LONG CONTEXT MODELS FOR CLINICAL PREDICTION TASKS ON EHRS

Michael Wornow*[1], Suhana Bedi*[1], Miguel Angel Fuentes Hernandez[1], Ethan Steinberg[1,2], Jason Alan Fries[1], Christopher Ré[1], Sanmi Koyejo[1], Nigam H. Shah[1]
[1]Stanford University [2]Prealize Health

ABSTRACT

Foundation Models (FMs) trained on Electronic Health Records (EHRs) have achieved state-of-the-art results on numerous clinical prediction tasks. However, most existing EHR FMs have context windows of <1k tokens. This prevents them from modeling full patient EHRs which can exceed 10k's of events. Recent advancements in subquadratic long-context architectures (e.g., Mamba) of-

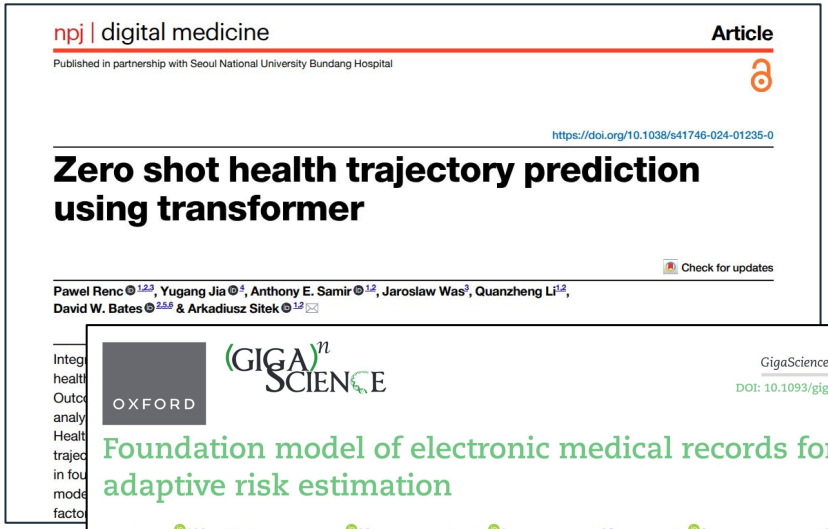
Generative Medical Event Models Improve with Scale

Shane Waxler*[1], Paul Blazek*[1], Davis White*[1], Daniel Sneider*[1], Kevin Chung[1], Mani Nagarathnam[1], Patrick Williams[1], Hank Voeller[1], Karen Wong[1], Matthew Swanhorst[1], Sheng Zhang[2], Naoto Usuyama[2], Cliff Wong[2], Tristan Naumann[2], Hoifung Poon[2], Andrew Loza[3], Daniella Meeker[3, 4], Seth Hain[1], and Rahul Shah[†1]

[1]Epic Systems
[2]Microsoft Research
[3]Yale School of Medicine
[4]Cosmos Governing Council

Abstract

Realizing personalized medicine at scale calls for methods that distill insights from longitudinal patient journeys, which can be viewed as a sequence of medical events. Foundation models pretrained on large-scale medical event data represent a promising direction for scaling real-world evidence generation and generalizing to diverse downstream tasks. Using Epic Cosmos, a dataset with medical events from de-identified longitudinal health records for 16.3 billion encounters over 300 million unique patient records from 310 health systems, we introduce the Comet models, a family of decoder-only transformer models pretrained on 118 million patients representing 115 billion discrete medical events (151 billion tokens). We present the largest scaling-law study of medical event data, establishing a methodology for pretraining and revealing power-law scaling relationships for compute, tokens, and model size. Consequently, we pretrained a series of compute-optimal models with up to 1 billion parameters. Conditioned on a patient's real-world history, Comet autoregressively predicts the next medical event to simulate patient health timelines. We studied 78 real-world tasks, including diagnosis prediction, disease prognosis, and healthcare operations. Remarkably for a foundation model with generic pretraining and simulation-based inference, Comet



npj | digital medicine

Article

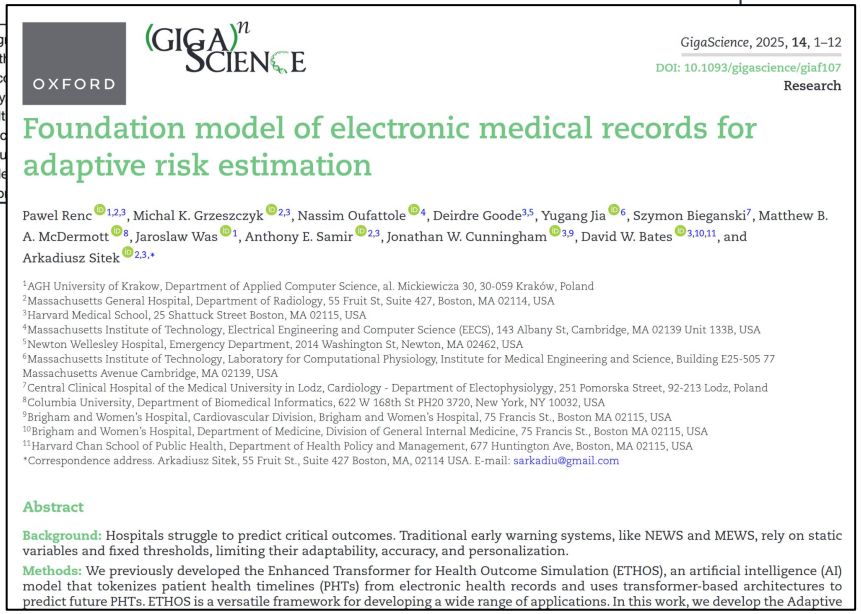
Published in partnership with Seoul National University Bundang Hospital

https://doi.org/10.1038/s41746-024-01235-0

Zero shot health trajectory prediction using transformer

Check for updates

Pawel Renc[1,2,3], Yugang Jia[1], Anthony E. Samir[1,2], Jaroslaw Was[3], Quanzheng Li[1,2], David W. Bates[2,5,6] & Arkadiusz Sitek[1,2]



OXFORD (GIGA)n SCIENCE

GigaScience, 2025, **14**, 1–12
DOI: 10.1093/gigascience/giaf107
Research

Foundation model of electronic medical records for adaptive risk estimation

Pawel Renc[1,2,3], Michal K. Grzeszczyk[2,3], Nassim Oufattole[4], Deirdre Goode[3,5], Yugang Jia[6], Szymon Bieganski[7], Matthew B. A. McDermott[8], Jaroslaw Was[1], Anthony E. Samir[2,3], Jonathan W. Cunningham[3,9], David W. Bates[3,10,11], and Arkadiusz Sitek[2,3,*]

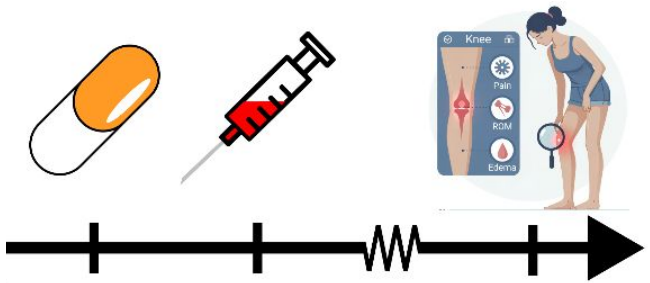
[1]AGH University of Krakow, Department of Applied Computer Science, al. Mickiewicza 30, 30-059 Kraków, Poland
[2]Massachusetts General Hospital, Department of Radiology, 55 Fruit St, Suite 427, Boston, MA 02114, USA
[3]Harvard Medical School, 25 Shattuck Street Boston, MA 02115, USA
[4]Massachusetts Institute of Technology, Electrical Engineering and Computer Science (EECS), 143 Albany St, Cambridge, MA 02139 Unit 133B, USA
[5]Newton Wellesley Hospital, Emergency Department, 2014 Washington St, Newton, MA 02462, USA
[6]Massachusetts Institute of Technology, Laboratory for Computational Physiology, Institute for Medical Engineering and Science, Building E25-505 77 Massachusetts Avenue Cambridge, MA 02139, USA
[7]Central Clinical Hospital of the Medical University in Lodz, Cardiology - Department of Electophysiolygy, 251 Pomorska Street, 92-213 Lodz, Poland
[8]Columbia University, Department of Biomedical Informatics, 622 W 168th St PH20 3720, New York, NY 10032, USA
[9]Brigham and Women's Hospital, Cardiovascular Division, Brigham and Women's Hospital, 75 Francis St., Boston MA 02115, USA
[10]Brigham and Women's Hospital, Department of Medicine, Division of General Internal Medicine, 75 Francis St., Boston MA 02115, USA
[11]Harvard Chan School of Public Health, Department of Health Policy and Management, 677 Huntington Ave, Boston, MA 02115, USA
*Correspondence address. Arkadiusz Sitek, 55 Fruit St., Suite 427 Boston, MA, 02114 USA. E-mail: sarkadiu@gmail.com

Abstract

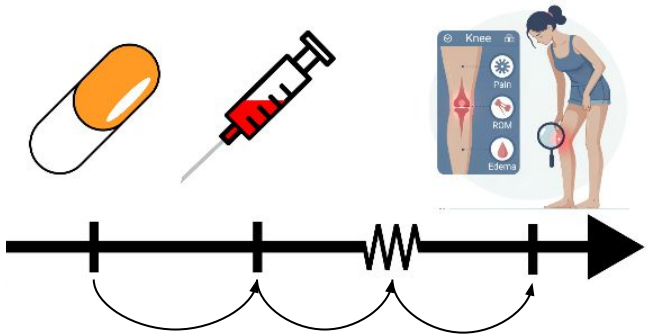
Background: Hospitals struggle to predict critical outcomes. Traditional early warning systems, like NEWS and MEWS, rely on static variables and fixed thresholds, limiting their adaptability, accuracy, and personalization.

Methods: We previously developed the Enhanced Transformer for Health Outcome Simulation (ETHOS), an artificial intelligence (AI) model that tokenizes patient health timelines (PHTs) from electronic health records and uses transformer-based architectures to predict future PHTs. ETHOS is a versatile framework for developing a wide range of applications. In this work, we develop the Adaptive

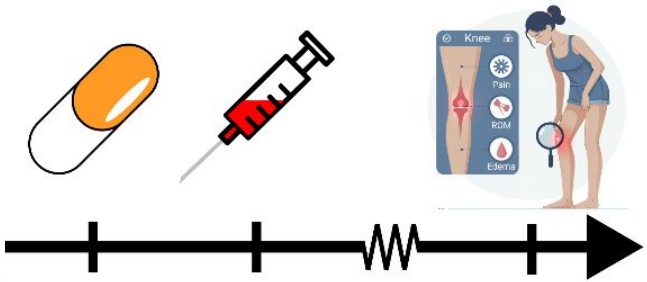
What observation comes next?

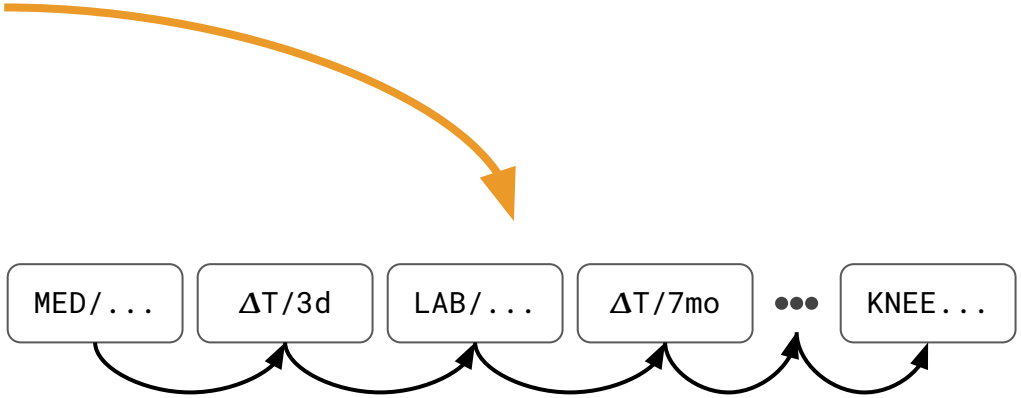


What observation comes next?

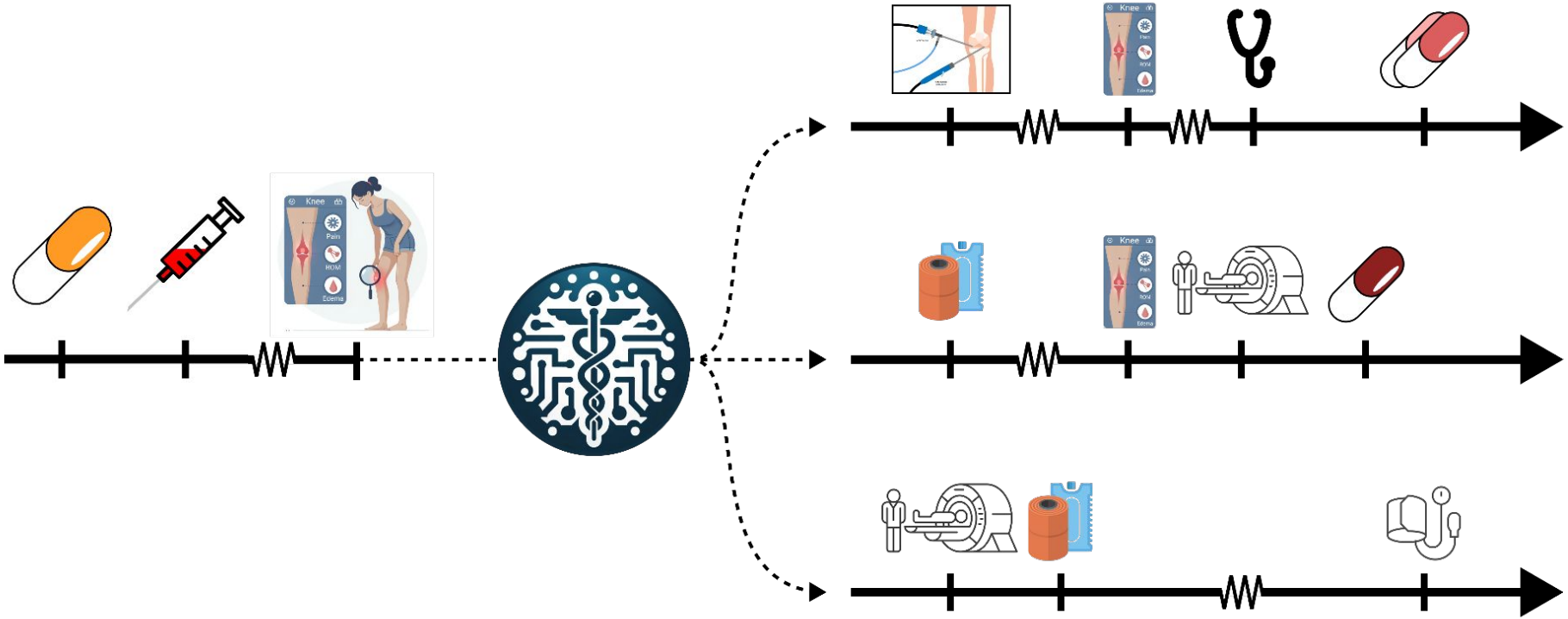


What observation comes next?



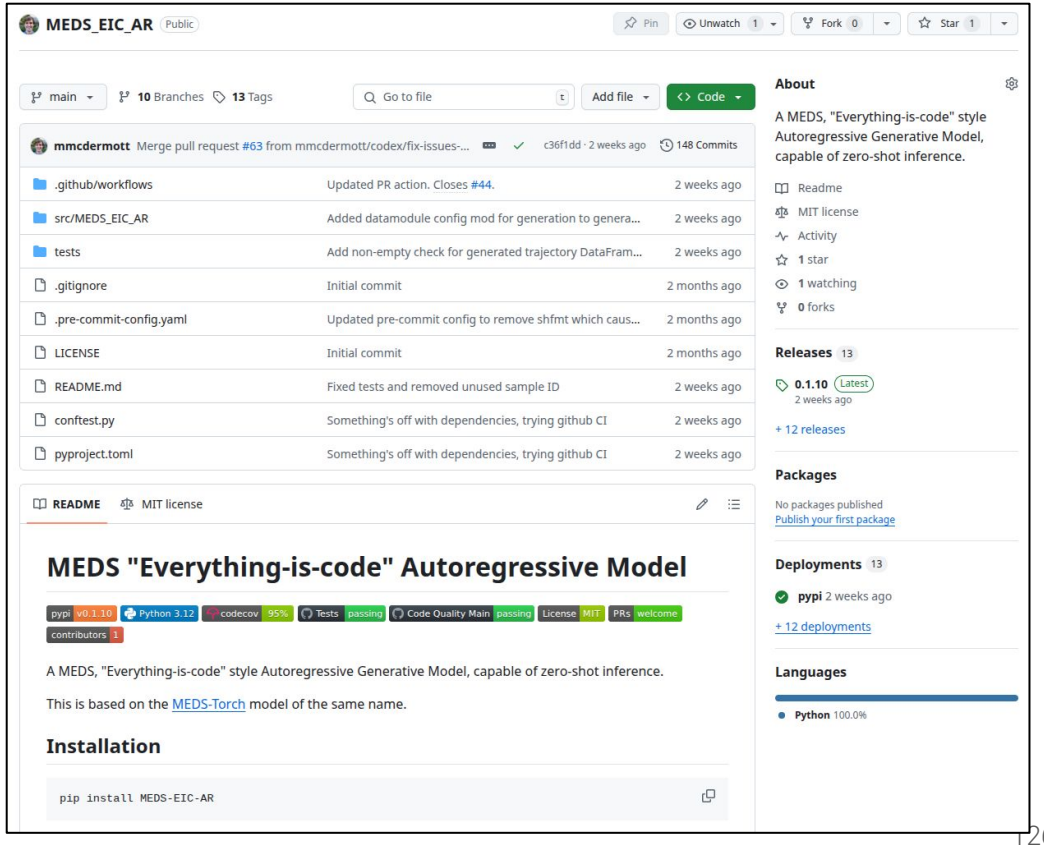


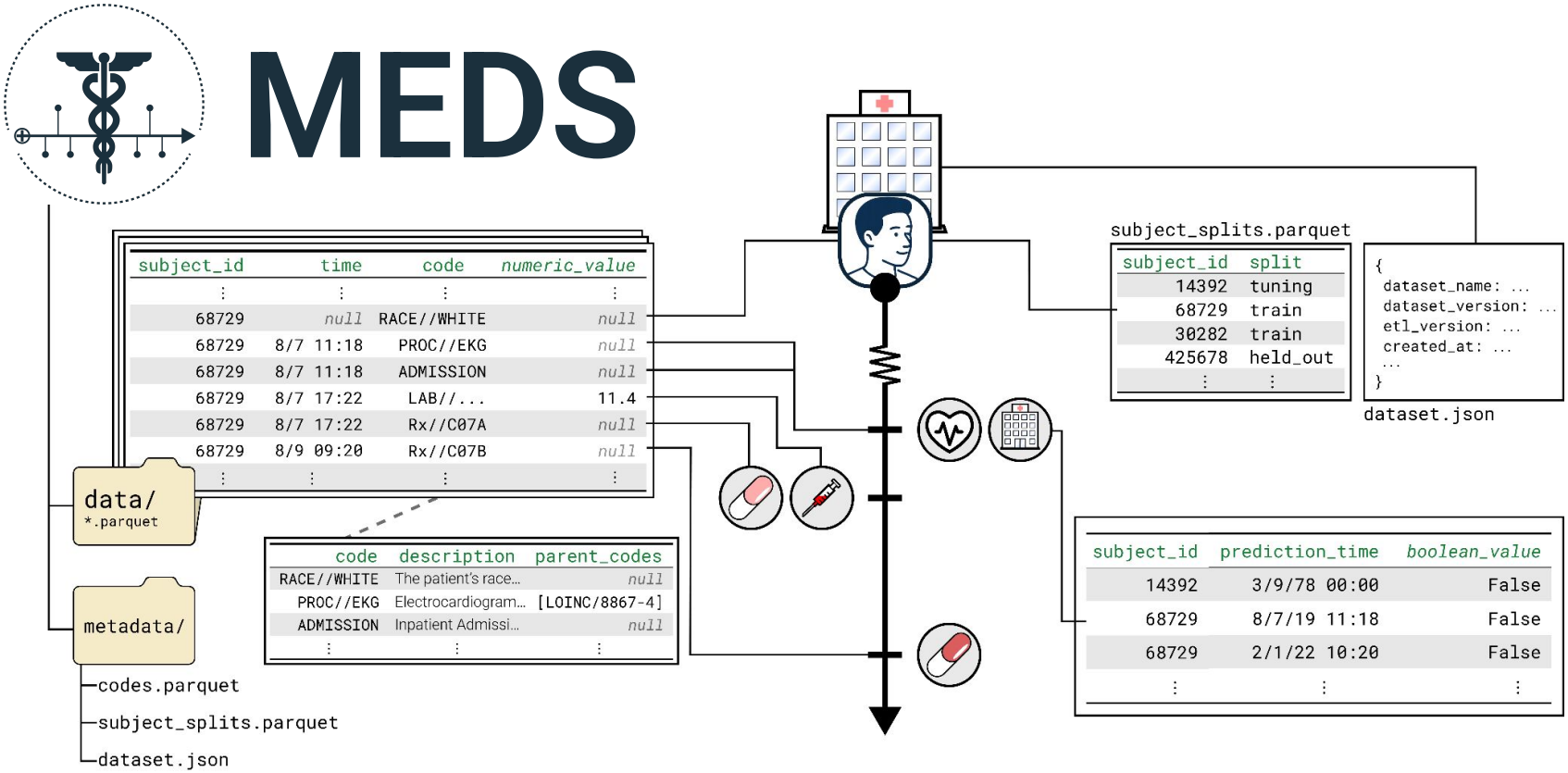
This is a foundation model because we can reason over synthetic futures:



An EHR Foundation Model Recipe

1. Convert to MEDS.
2. Use MEDS-EIC-AR.

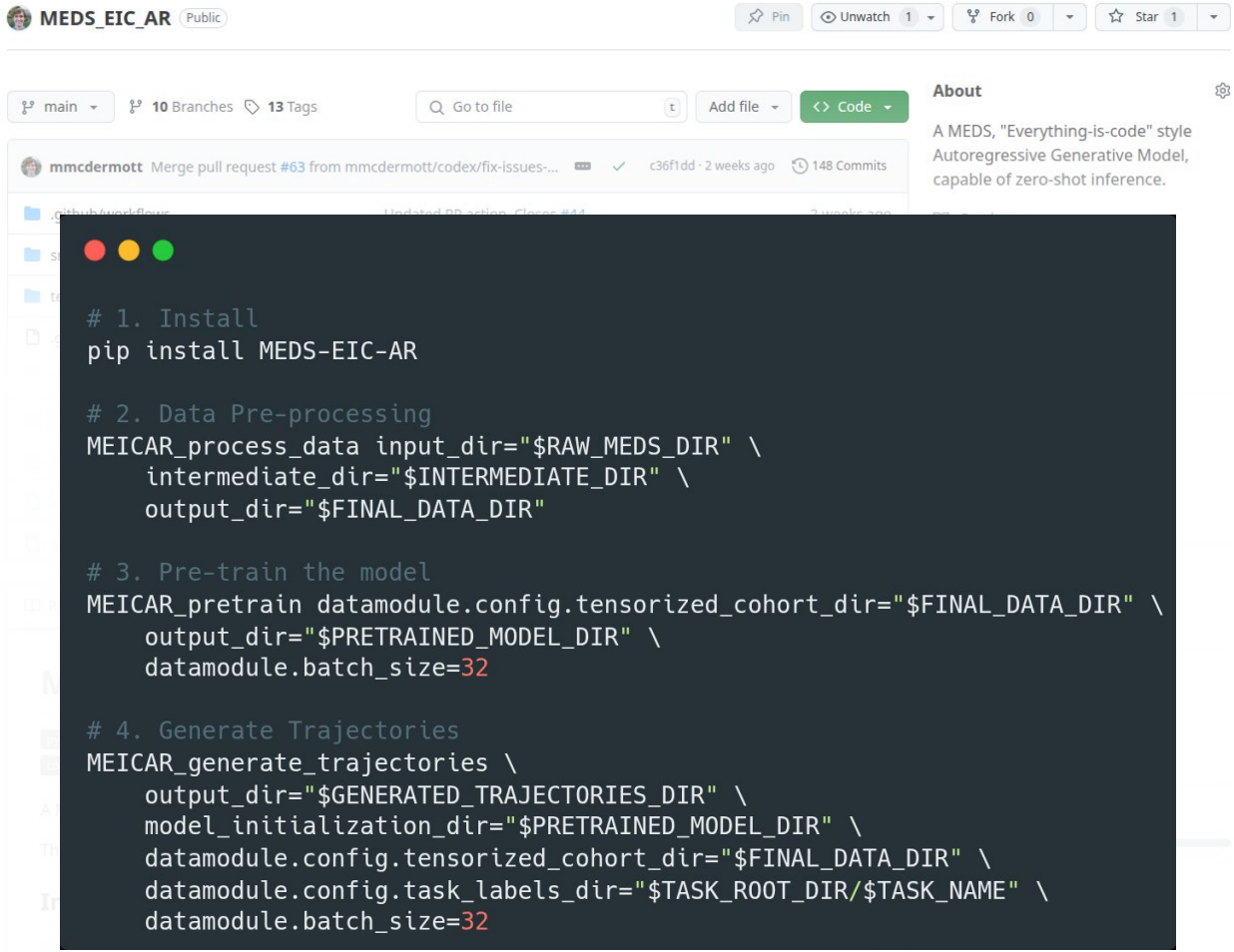



MEDS
subject_id time code numeric_value
68729 null RACE//WHITE null
68729 8/7 11:18 PROC//EKG null
68729 8/7 11:18 ADMISSION null
68729 8/7 17:22 LAB//... 11.4
68729 8/7 17:22 Rx//C07A null
68729 8/9 09:20 Rx//C07B null
data/
*.parquet
code description parent_codes
RACE//WHITE The patient's race... null
PROC//EKG Electrocardiogram... [LOINC/8867-4]
ADMISSION Inpatient Admissi... null
metadata/
codes.parquet
subject_splits.parquet
dataset.json
subject_splits.parquet
subject_id split
14392 tuning
68729 train
30282 train
425678 held_out
{
dataset_name: ...
dataset_version: ...
etl_version: ...
created_at: ...
...
}
dataset.json
subject_id prediction_time boolean_value
14392 3/9/78 00:00 False
68729 8/7/19 11:18 False
68729 2/1/22 10:20 False

MEDS Reveals a Recipe for Autoregressive Foundation models on EHR Data

subject_id	time	code	*numeric_value*
⋮	⋮	⋮	⋮
68729	*null*	RACE//WHITE	*null*
68729	8/7 11:18	PROC//EKG	*null*
68729	8/7 11:18	ADMISSION	*null*
68729	8/7 17:22	LAB//...	11.4
68729	8/7 17:22	Rx//C07A	*null*
68729	8/9 09:20	Rx//C07B	*null*
⋮	⋮	⋮	⋮

Given a sequence of MEDS rows (ordered by time), can I predict the next row?


MEDS_EIC_AR
Public
Pin
Unwatch 1
Fork 0
Star 1
main
10 Branches
13 Tags
Go to file
Add file
Code
About
A MEDS, "Everything-is-code" style Autoregressive Generative Model, capable of zero-shot inference.
mmcdermott Merge pull request #63 from mmcdermott/codex/fix-issues-...
c36f1dd · 2 weeks ago
148 Commits
1. Install
pip install MEDS-EIC-AR
2. Data Pre-processing
MEICAR_process_data input_dir="$RAW_MEDS_DIR" \
intermediate_dir="$INTERMEDIATE_DIR" \
output_dir="$FINAL_DATA_DIR"
# 3. Pre-train the model
MEICAR_pretrain datamodule.config.tensorized_cohort_dir="$FINAL_DATA_DIR" \
output_dir="$PRETRAINED_MODEL_DIR" \
datamodule.batch_size=32
# 4. Generate Trajectories
MEICAR_generate_trajectories \
output_dir="$GENERATED_TRAJECTORIES_DIR" \
model_initialization_dir="$PRETRAINED_MODEL_DIR" \
datamodule.config.tensorized_cohort_dir="$FINAL_DATA_DIR" \
datamodule.config.task_labels_dir="$TASK_ROOT_DIR/$TASK_NAME" \
datamodule.batch_size=32