MEDS KDD 2025 Tutorial

Presenters: Matthew McDermott and Suhana Bedi

Preparation: 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

Presenters

Matthew McDermott

Assistant Professor

Columbia Department of Biomedical Informatics

Suhana Bedi

Ph.D. Student

Stanford Department of Biomedical Data Science

Additional Tutorial Notebook Authors

Kamilė Stankevičiūtė
Ph.D. Student, University of Cambridge

Justin Xu
Ph.D. Student, University of Oxford

Ethan Steinberg
Researcher, Prealize Health

Help you to think about AI/ML over EHR data in an insightful, responsible, & empowering way.

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

Our Expectations of You

1. Follow the ACM KDD code of conduct
2. Be here to learn
   This will be interactive, both in terms of discussion and implementation.
   Ask questions
3. Be here to teach
   Work with those around you to share knowledge and ideas
   Share your experiences and expertise
4. Have fun!

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.

What's next?

	✓	*8:05 - 8:10*	Opening remarks
CO	1.	*8:10 - 8:20*	What is MEDS?
	2.	*8:20 - 8:25*	Guiding Problem Setup
CO	3.	*8:25 - 9:05*	Convert your data into MEDS
	4.	*9:05 - 9:10*	Extract a prediction task cohort with ACES
CO	5.	*9:10 - 9:30*	Develop a predictive model 1: Tabular Baseline
☕		*9:30 - 10:00*	Coffee Break
CO	6.	*10:00 - 10:40*	Develop a predictive model 2: Neural Network
	7.	*10:40 - 10:45*	Participate in the open-source community
	8.	*10:45 - 11:00*	Closing Remarks

REPLACE

www.tinyurl.com/
MEDS-KDD-pres

Part 1: What is MEDS and why use it?

Driving Problem: Health AI has a Reproducibility Crisis

Table 3: Comparison of results shown from original studies ("study") and the reproduction here ("repro."). While the model used in studies varied, we grouped them as either linear (Lin) or non-linear (NonLin). We evaluated two models: a linear model (logistic regression, LR) and a non-linear model (gradient boosting, GB). The outcome was in-hospital mortality for all studies.

	Sample size		Outcome (%)			AUROC		
Cohort	Study	Repro.	Study	Repro.	Model	Study	GB	LR
Caballero Barajas and Akella (2015), W=24	11,648	11,648	-					
Caballero Barajas and Akella (2015), W=48	11,648	11,648	-					
Caballero Barajas and Akella (2015), W=72	11,648	11,648	-					
Calvert et al. (2016b)	3,054	1,985	12.84					
Calvert et al. (2016a)	9,683	18,396	10.68					
Celi et al. (2012), AKI	1,400	4,741	30.7					
Celi et al. (2012), SAH	223	350	25.6					
Che et al. (2016) (b)	4,000	4,000	13.85					
Ding et al. (2016)	4,000	4,000	13.85	14.35	NonLin	0.8177	0.8461	0.8273
Ghassemi et al. (2014), W=12	19,308	28,172	10.84	12.2	Lin	0.84	0.8846	0.8609
Ghassemi et al. (2014), W=24	19,308	23,442	10.80	12.92	Lin	0.841	0.8841	0.8651
Ghassemi et al. (2015)	10,202	21,969	-	13.51	NonLin	0.812	0.8781	0.8591
Grnarova et al. (2016)	31,244	29,572	13.82	12.49	NonLin	0.963	0.9819	0.9765
Harutyunyan et al. (2017)	42,276	45,493	-	10.54	NonLin	0.8625	0.9406	0.9286

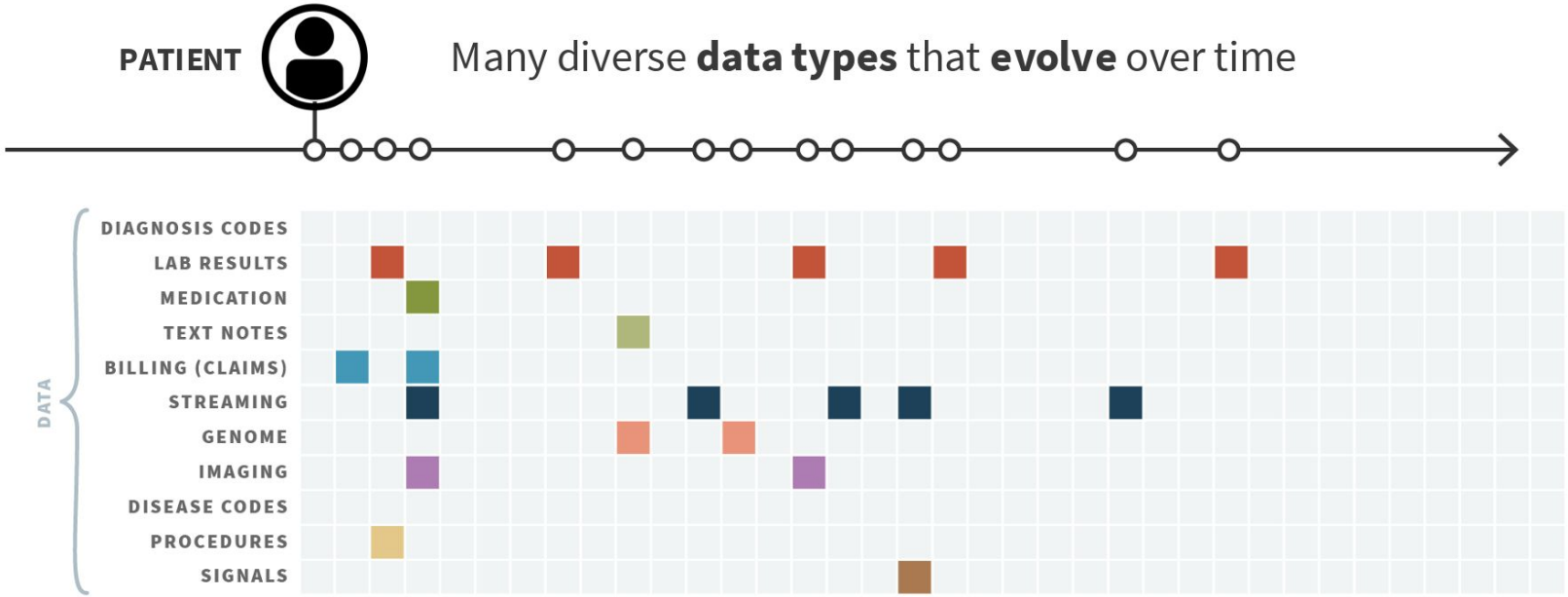
We reproduced datasets for 38 experiments corresponding to 28 published studies using MIMIC. In half of the experiments, the sample size we acquired was 25% greater or smaller than the sample size reported.

- Alistair Johnson et. al., 2017

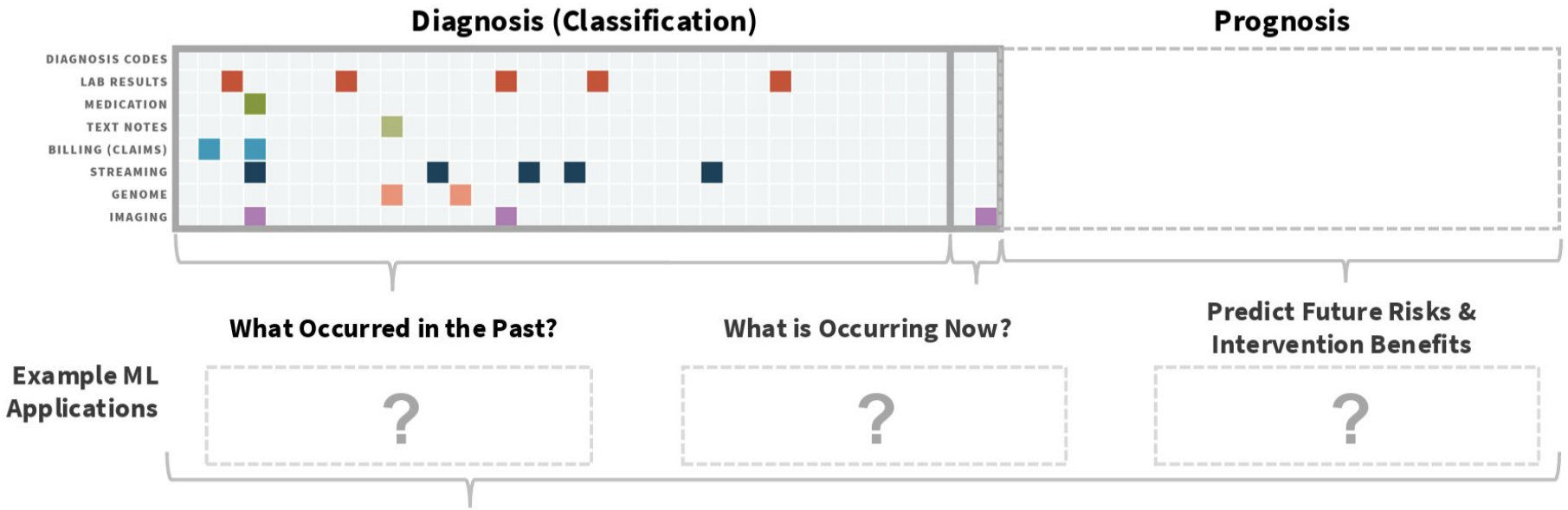
Johnson, A.E.W., Pollard, T.J. & Mark, R.G.. (2017). Reproducibility in critical care: a mortality prediction case study. *Proceedings of the 2nd Machine Learning for Healthcare Conference*, in *Proceedings of Machine Learning Research* 68:361-376 Available from https://proceedings.mlr.press/v68/johnson17a.html.

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

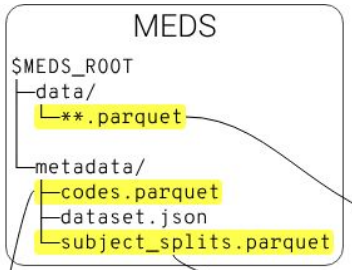
A Patient's Data can be represented as a **Timeline**



This is an opportunity to pursue many **ML applications**



MEDS



The MEDS data schema epitomizes simplicity and has only 3 required columns: `subject_id`, `time`, and `code`. It supports two optional value modalities: *`numeric`* and *`text`*.

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

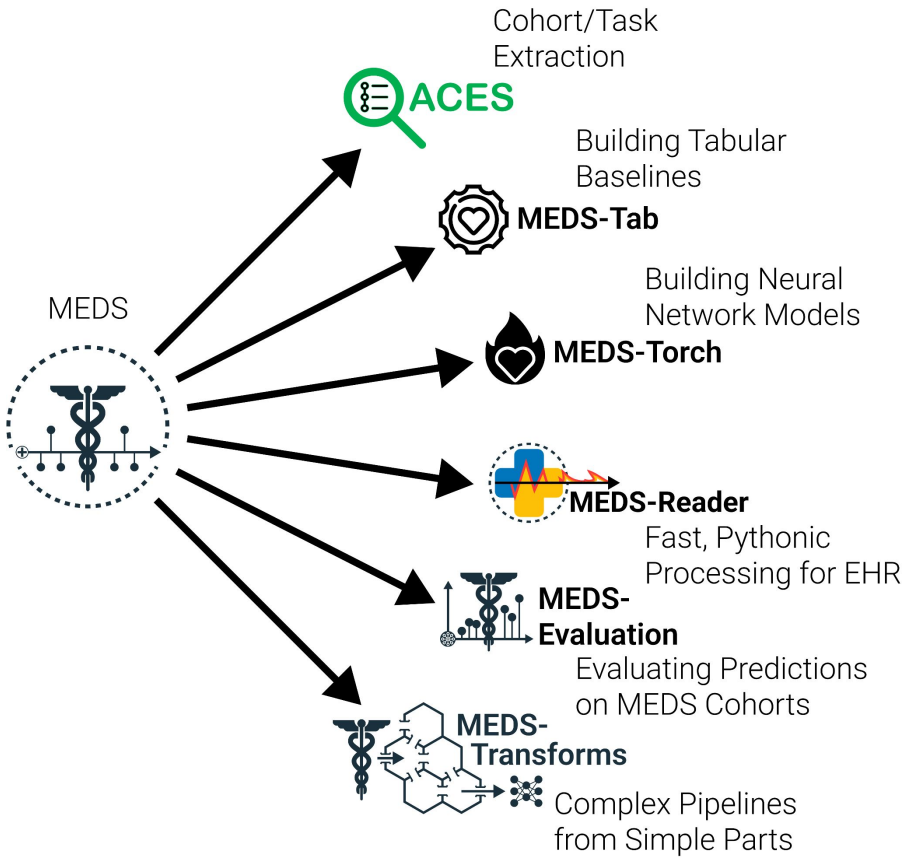
code	description	parent_codes
RACE//WHITE	The patient's race...	*null*
SEX//M	The patient's biolo...	*null*
SEX//F	The patient's biolo...	*null*
MEDS_BIRTH	*null*	*null*
MEDS_DEATH	*null*	*null*
HR//bpm	Heart rate, measu...	[LOINC/8867-4]
ED//REG	Emergency depart...	*null*
⋮	⋮	⋮

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

subject_id	split
68729	train
125829	train
14392	tuning
30282	train
425678	train
⋮	⋮

The MEDS splits schema identifies which `subject_id`s are assigned to which splits.

MEDS


Cohort/Task Extraction
ACES
Building Tabular Baselines
MEDS-Tab
Building Neural Network Models
MEDS
MEDS-Torch
MEDS-Reader
Fast, Pythonic Processing for EHR
MEDS-Evaluation
Evaluating Predictions on MEDS Cohorts
MEDS-Transforms
Complex Pipelines from Simple Parts

MEDS

A

Used at more than 19 institutions

B

12 Papers

17 Datasets

10 Models

C

Up to 100x Faster

Up to 80% fewer lines of code

More than 13 tools available

Part 1 Interactive Content

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.

Part 2: Problem Set-up

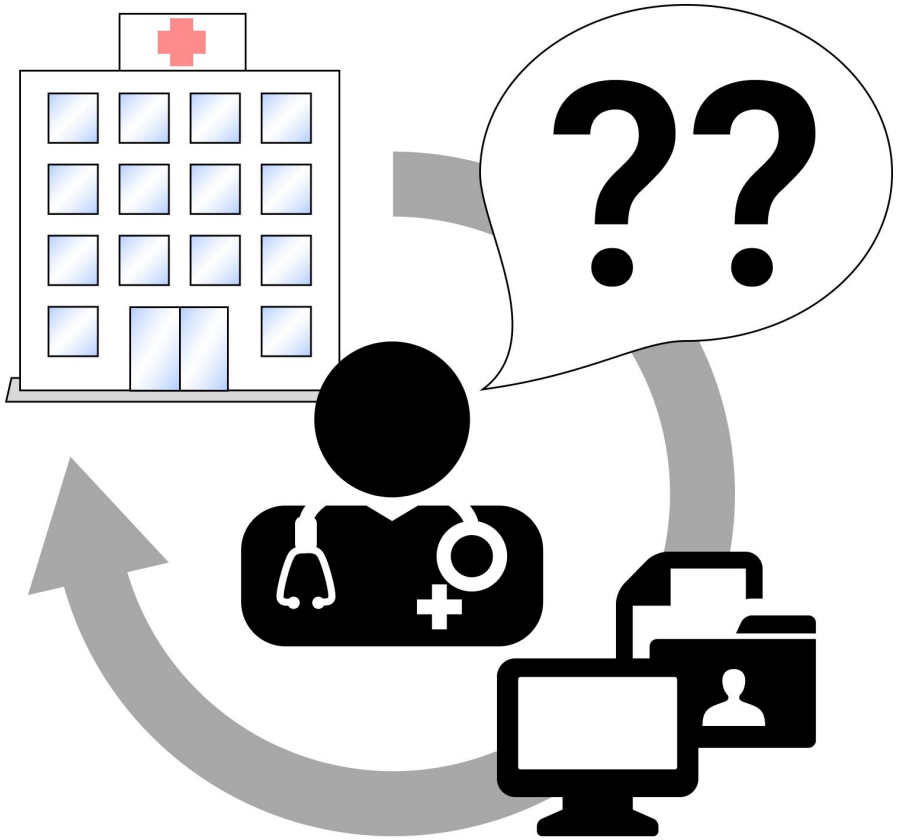
Disclaimers and Goal

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*

What's next?

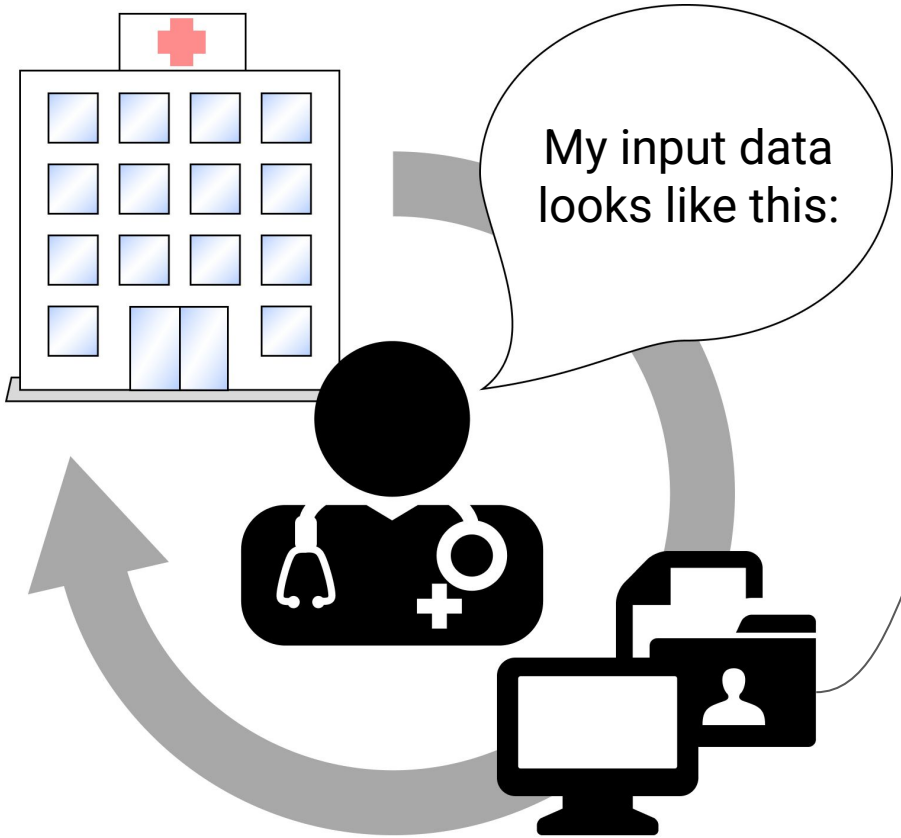
Parts:

3. Convert your data into MEDS *(Interactive)*
4. Extract a prediction task cohort with ACES *(Offline)*
5. Develop a predictive (tabular) model *(Interactive)*
6. Develop a predictive (NN) model *(Interactive)*
7. Participate in the open-source community *(Offline)*

Resources:

- All interactive sessions will leverage Google Colab notebooks. We will provide templates, and you will have to discuss and fill in cells amongst yourselves.
- All notebooks are further embedded in complete (e.g., non-interactive) form on the MEDS website: https://medical-event-data-standard.github.io/

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
```

Part 3 Interactive Content: 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!

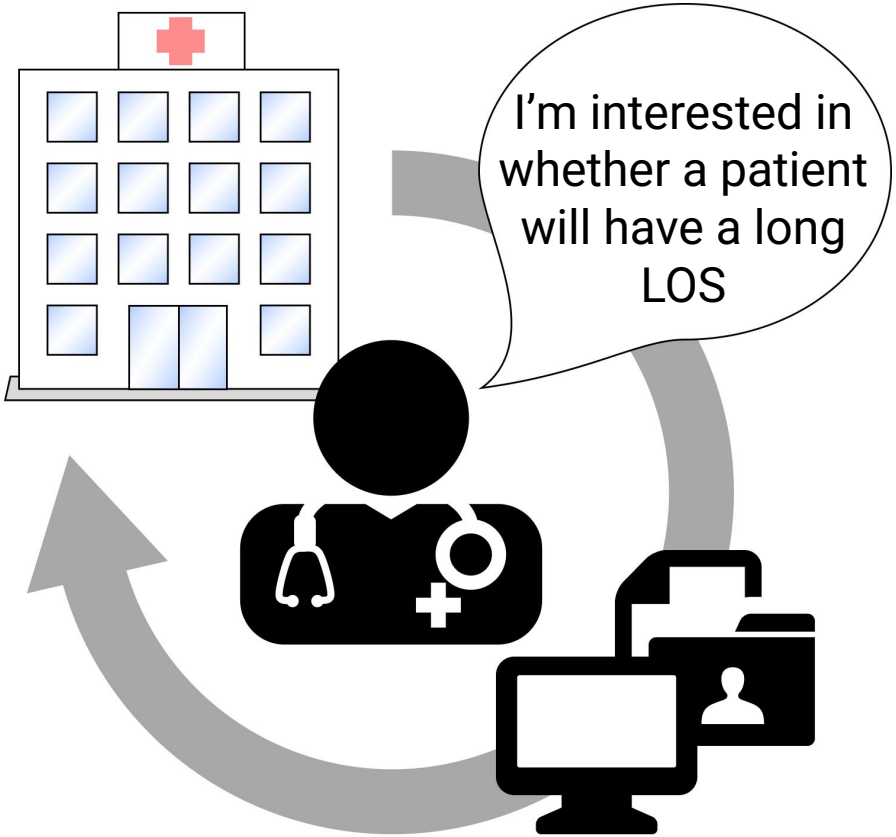
www.tinyurl.com/
MEDS-KDD-p3

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 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

Part 4 ***Offline, Asynchronous Content***

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-KDD-p4

ACES: Reproducible Extraction of Task Cohorts for EHRs

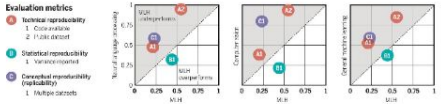
Justin Xu, Jack Gallifant, Alistair E. W. Johnson, Matthew B. A. McDermott

UNIVERSITY OF OXFORD · MIT · HARVARD MEDICAL SCHOOL

ICLR International Conference On Learning Representations

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*

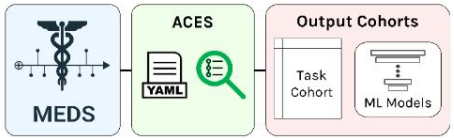
code	description	parent_codes
RACE//WHITE	The patient's race is...	*null*
SEX//M	The patient's biologic...	*null*
SEX//F	The patient's biologic...	*null*
MEDS_BIRTH	*null*	*null*
MEDS_DEATH	*null*	*null*
HR//bpm	Heart rate, measured...	[LOINC/8867-4]
ED//REG	Emergency departme...	*null*

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:

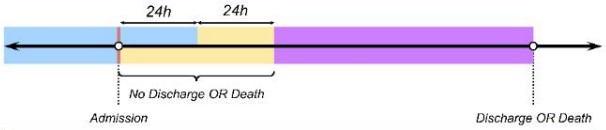
Task Extraction Made Easy!



- Transparent
- Reproducible
- Extractable on diverse datasets

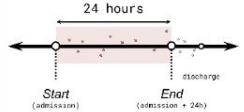
ACES Configuration Files

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.



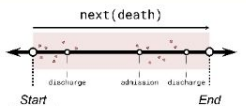

```yaml
predicates:
  admission:
    code: ADMISSION
  discharge:
    code: DISCHARGE
  death:
    code: DEATH
  discharge_or_death:
    expr: or(discharge, death)

trigger: admission

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***

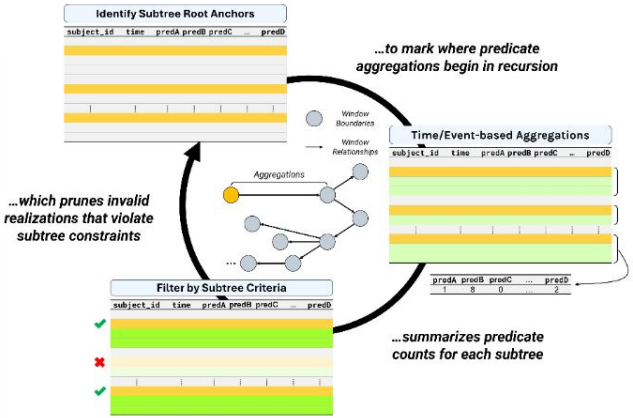
Event-based Aggregation



*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

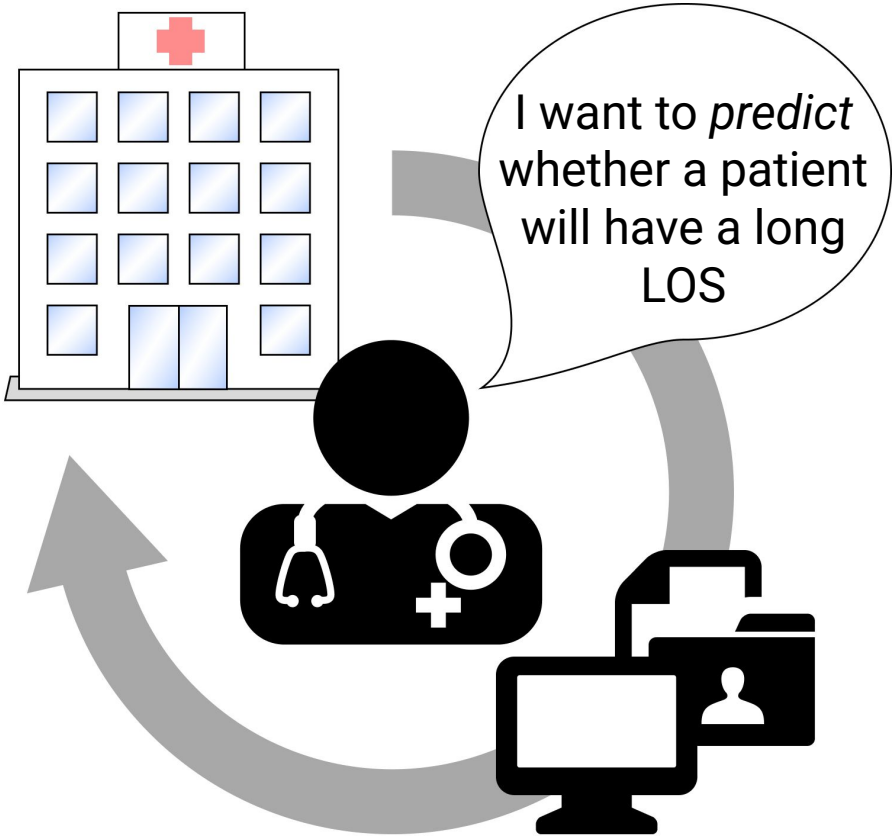
MIRAM gratefully acknowledges support from a Berkowitz Postdoctoral Fellowship at Harvard Medical School. JG is funded by the National Institutes of Health through NIH-USA RO1CA294033. JX greatly appreciates support from supervisors David Eyre (University of Oxford) and Curtis Langlotz (Stanford University). We also acknowledge valuable contributions by Tom Pollard (Massachusetts Institute of Technology) and by the broader MEDS ecosystem of contributors and users.

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 5: Building a model


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

Part 5: Tabular Baseline Interactive Content

Time: 30 Minutes *Note:* May Span Coffee Break!

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-KDD-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.

Coffee Break!

(Be back by 10:00)

Part 6: Neural Network Interactive Content

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-KDD-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.

Part 7: Leveraging and Contributing to the Community

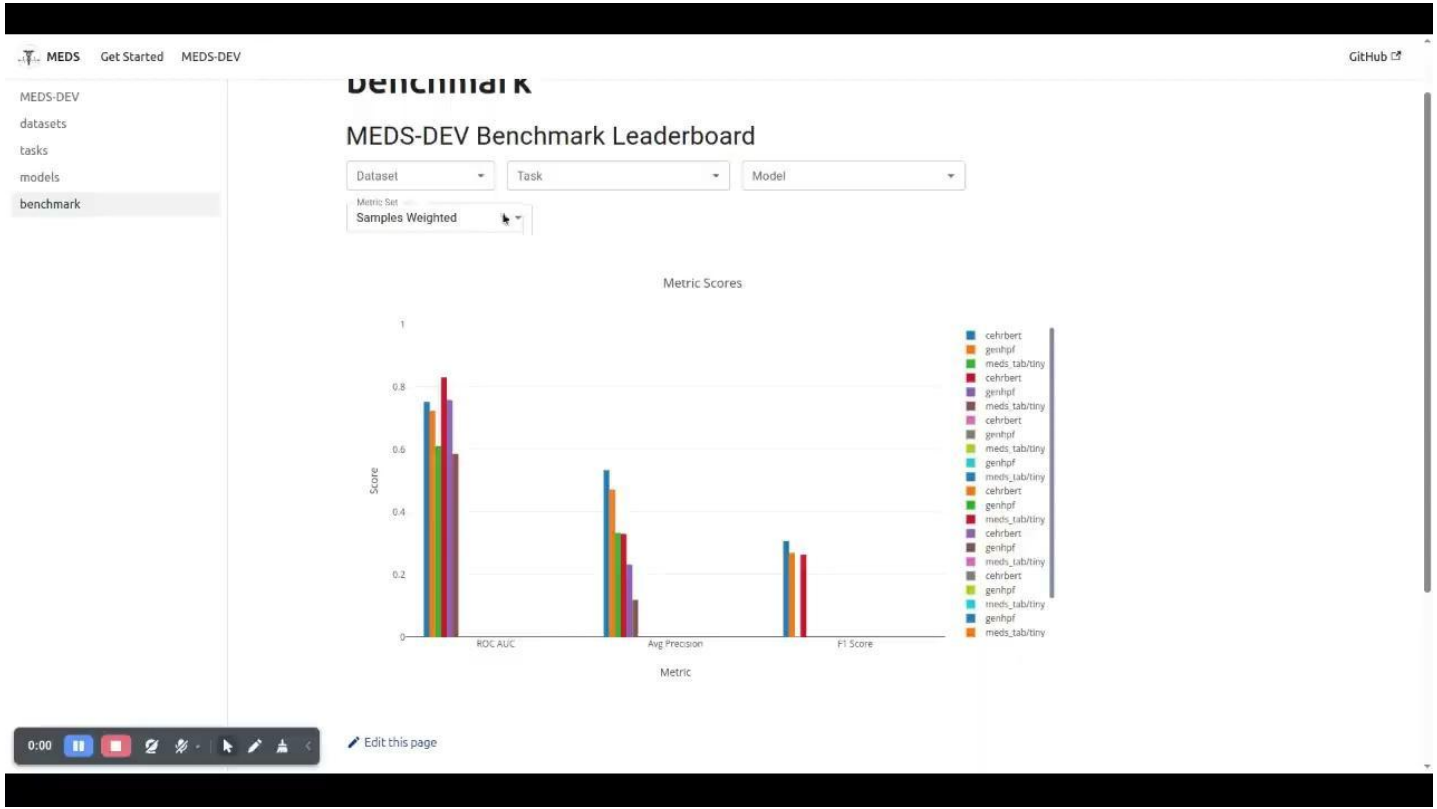
MEDS-DEV:

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
```

Sharing Results



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	

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)*.

However, the **key challenges** are:

Non-transferable Models

Models trained at one hospital can't run at another

Incomparable Results

Can't tell which mortality prediction model actually works best

Reinventing the Wheel

Every team rebuilds the same data pipelines from scratch

Introducing **MEDS**

Non-transferable Models

Models trained at one hospital can't run at another

Incomparable Results

Can't tell which mortality prediction model actually works best

Reinventing the Wheel

Every team rebuilds the same data pipelines from scratch

MEDS:

Universal format enables seamless model transfer across institutions

Standardized tasks enable fair, direct model comparisons

Pre-standardized data eliminates redundant preprocessing

Even our *task selection on public data* isn't reproducible!

Table 3: Comparison of results shown from original studies ("study") and the reproduction here ("repro."). While the model used in studies varied, we grouped them as either linear (Lin) or non-linear (NonLin). We evaluated two models: a linear model (logistic regression, LR) and a non-linear model (gradient boosting, GB). The outcome was in-hospital mortality for all studies.

	Sample size		Outcome (%)			AUROC		
Cohort	Study	Repro.	Study	Repro.	Model	Study	GB	LR
Caballero Barajas and Akella (2015), W=24	11,648	11,648	-	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Caballero Barajas and Akella (2015), W=48	11,648	11,648	-					
Caballero Barajas and Akella (2015), W=72	11,648	11,648	-					
Calvert et al. (2016b)	3,054	1,985	12.84					
Calvert et al. (2016a)	9,683	18,396	10.68					
Celi et al. (2012), AKI	1,400	4,741	30.7					
Celi et al. (2012), SAH	223	350	25.6					
Che et al. (2016) (b)	4,000	4,000	13.85					
Ding et al. (2016)	4,000	4,000	13.85	14.35	NonLin	0.8177	0.8461	0.8273
Ghassemi et al. (2014), W=12	19,308	28,172	10.84	12.2	Lin	0.84	0.8846	0.8609
Ghassemi et al. (2014), W=24	19,308	23,442	10.80	12.92	Lin	0.841	0.8841	0.8651
Ghassemi et al. (2015)	10,202	21,969	-	13.51	NonLin	0.812	0.8781	0.8591
Grnarova et al. (2016)	31,244	29,572	13.82	12.49	NonLin	0.963	0.9819	0.9765
Harutyunyan et al. (2017)	42,276	45,493	-	10.54	NonLin	0.8625	0.9406	0.9286

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

Johnson, A.E.W., Pollard, T.J. & Mark, R.G.. (2017). Reproducibility in critical care: a mortality prediction case study. *Proceedings of the 2nd Machine Learning for Healthcare Conference*, in *Proceedings of Machine Learning Research* 68:361-376 Available from https://proceedings.mlr.press/v68/johnson17a.html.

The Reproducibility Crisis in ML for Health is Unambiguous

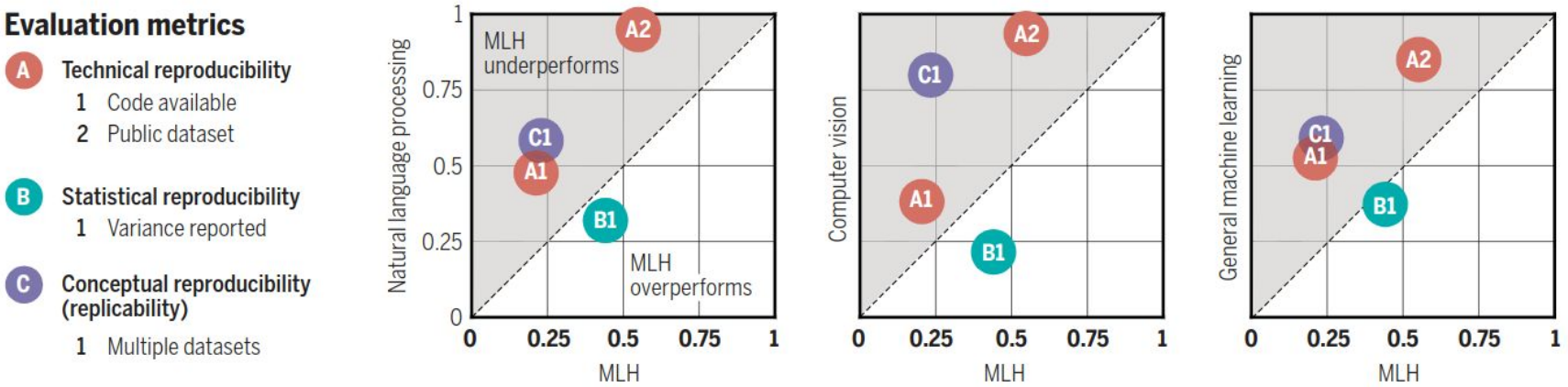


Fig. 1. Reproducibility metrics for machine learning applications. Shown are reproducibility metrics (A, B, and C) for evaluating scientific papers from four machine learning subspecialties: machine learning in health (MLH), natural language processing, computer vision, and general machine learning. Presented is the fraction of papers in a given subspecialty (*y* axis) versus those in MLH (*x* axis) that release their code (A1), release their data (A2), report their variance (B1), and leverage multiple datasets (C1). MLH consistently lags other subfields of machine learning on all measures of reproducibility apart from inclusion of proper statistical variance.

Matthew B. A. McDermott et al., Reproducibility in machine learning for health research: Still a ways to go. Sci. Transl. Med. **13**, eabb1655 (2021). DOI: 10.1126/scitranslmed.abb1655

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."

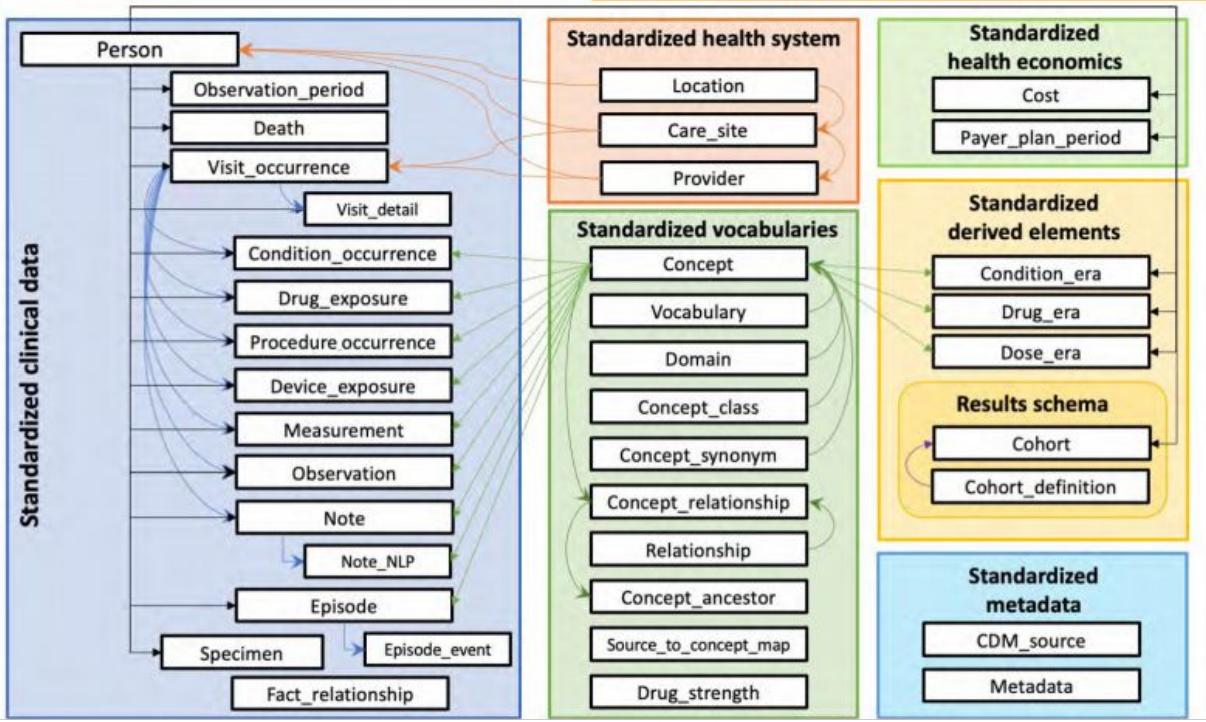
- Clair Blacketer
2020 Titan Award for Data Standards recipient

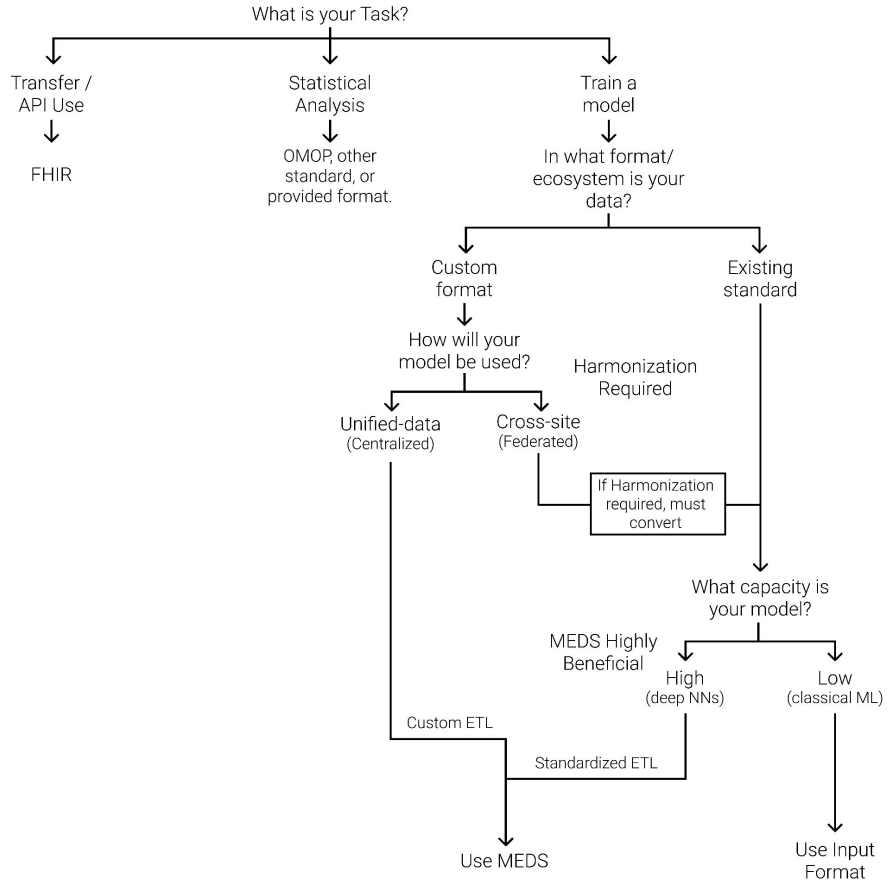
OMOP CDM By The Numbers

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

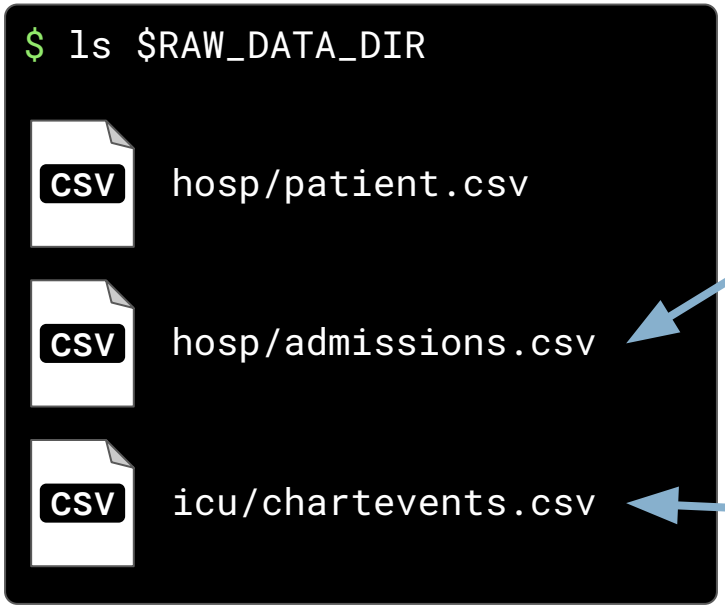
394 fields
- 193 with _id to standardize identification
- 101 with _concept_id to standardize content
- 43 with _source_value to preserve original data

1 Open Community Data Standard



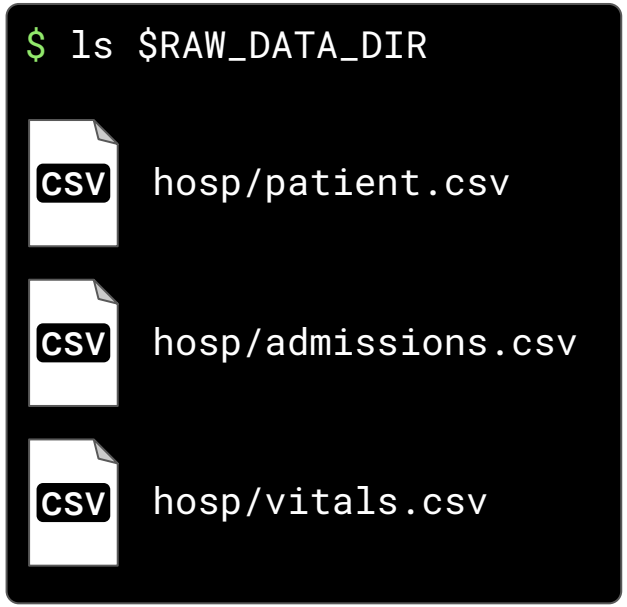

What is your Task?
Transfer / API Use
FHIR
Statistical Analysis
OMOP, other standard, or provided format.
Train a model
In what format/ ecosystem is your data?
Custom format
How will your model be used?
Unified-data (Centralized)
Cross-site (Federated)
Harmonization Required
If Harmonization required, must convert
Existing standard
What capacity is your model?
MEDS Highly Beneficial
High (deep NNs)
Low (classical ML)
Custom ETL
Standardized ETL
Use MEDS
Use Input Format

MEDS' Simplicity Enables Easy Extraction Pipelines



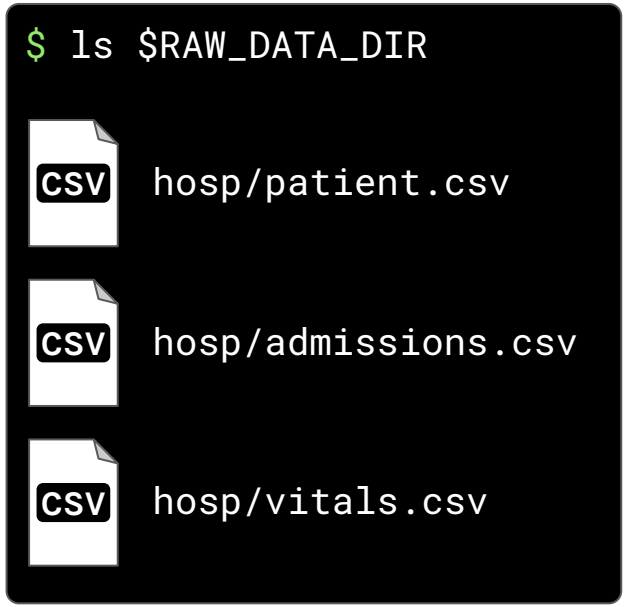

```yaml
subject_id_col: subject_id
hosp/admissions:
  ed_registration:
    code: ED_REGISTRATION
    time: col(edregtime)
    time_format: "%Y-%m-%d %H:%M:%S"
  admission:
    code:
      - HOSPITAL_ADMISSION
      - col(admission_type)
      - col(admission_location)
    time: col(admittime)
    time_format: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

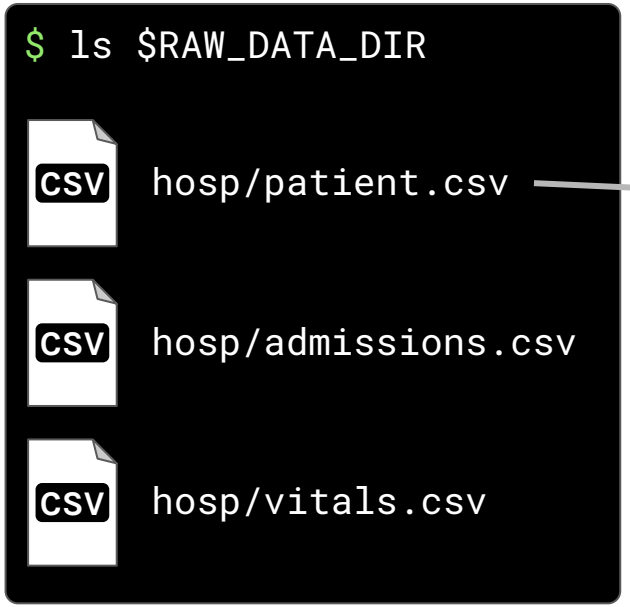
Extracting Data to MEDS with MEDS-Extract



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

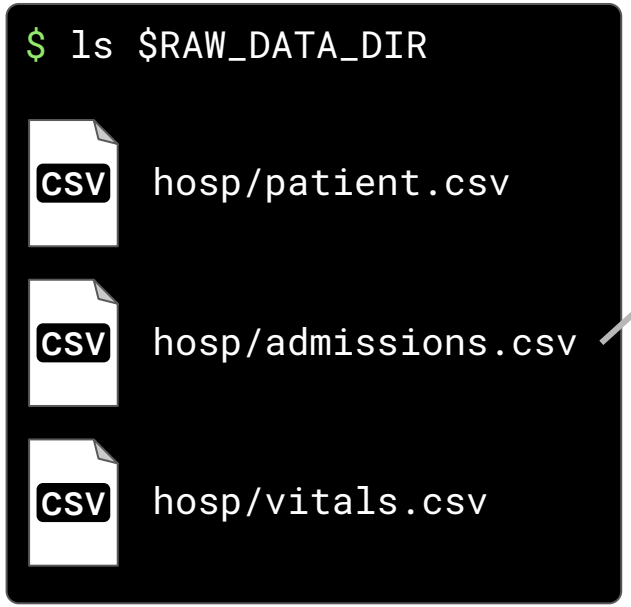
Extracting Data to MEDS with MEDS-Extract



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

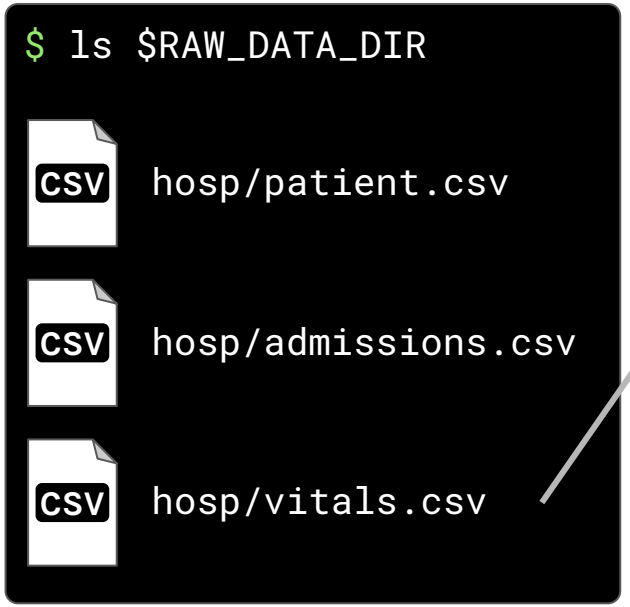
Extracting Data to MEDS with MEDS-Extract



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

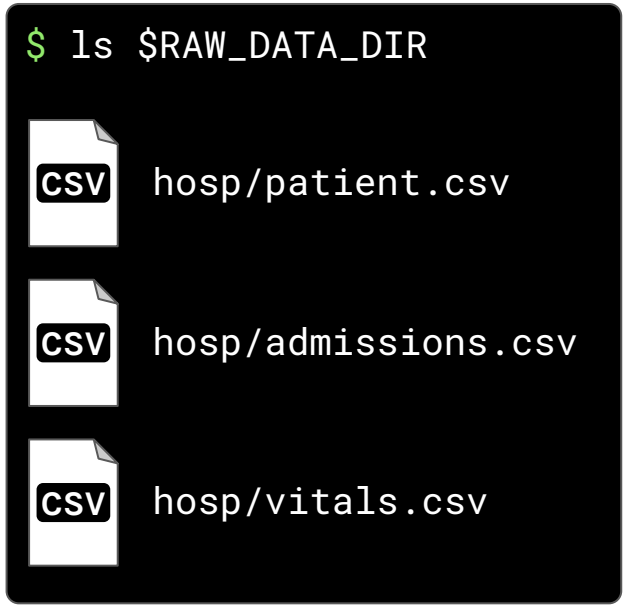
Extracting Data to MEDS with MEDS-Extract



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

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

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

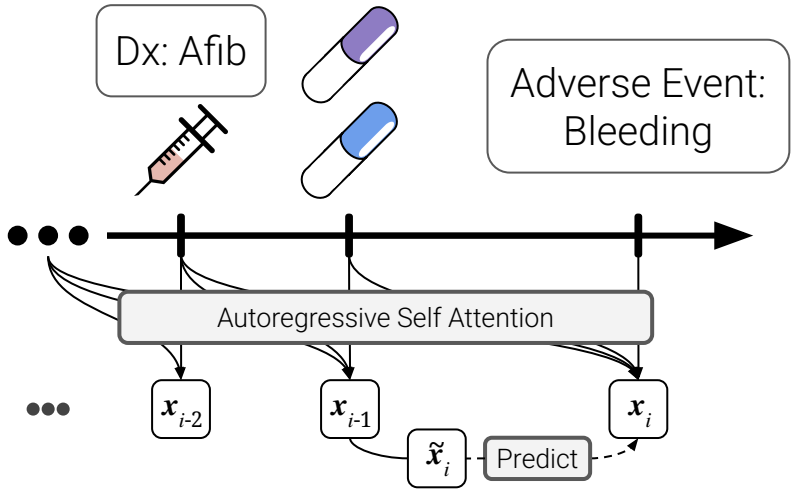
pypi 2 weeks ago

+ 12 deployments

Languages

Python 100.0%

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



Installation and Usage

```bash
# 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"
```

```yaml
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

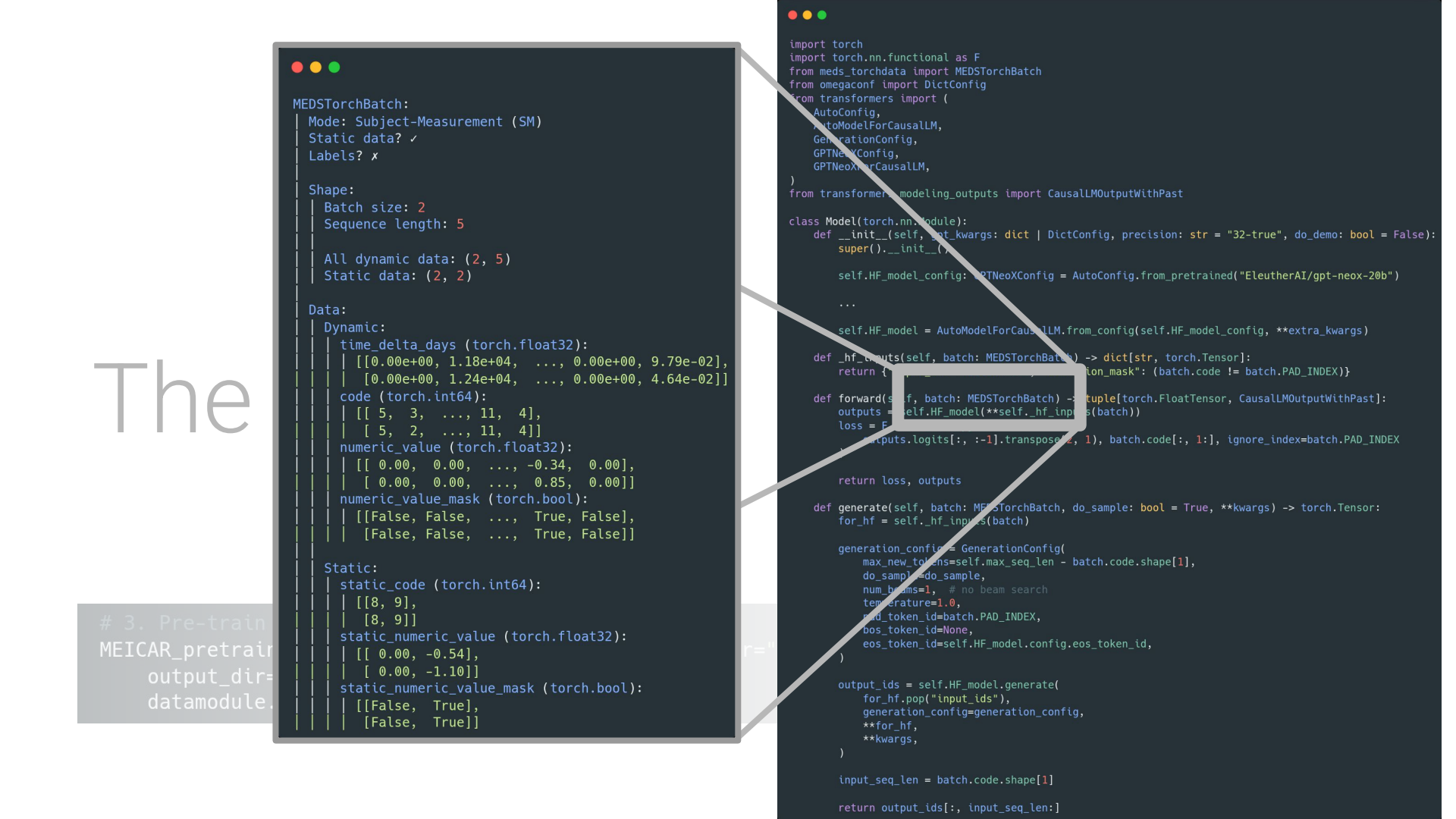
```
# 3. Pre-train the model
MEICAR_pretrain datamodule.config.tensorized_cohort_dir="
    output_dir="$PRETRAINED_MODEL_DIR" \
    datamodule.batch_size=32
```

```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:

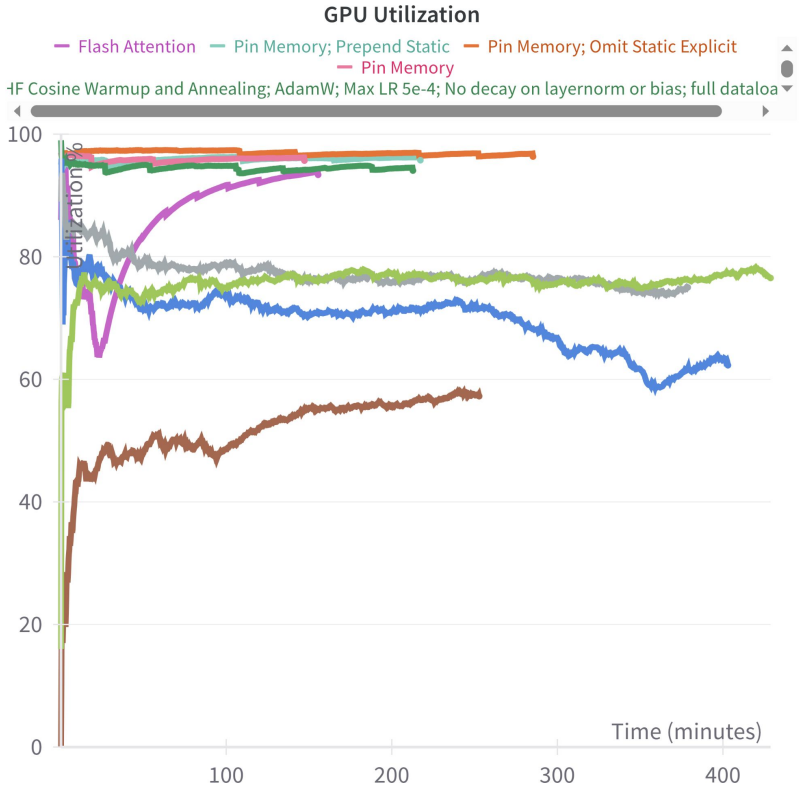
```
import os
from meds_torchdata import MEDSPytorchDataset, MEDSTorchDataConfig

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
```

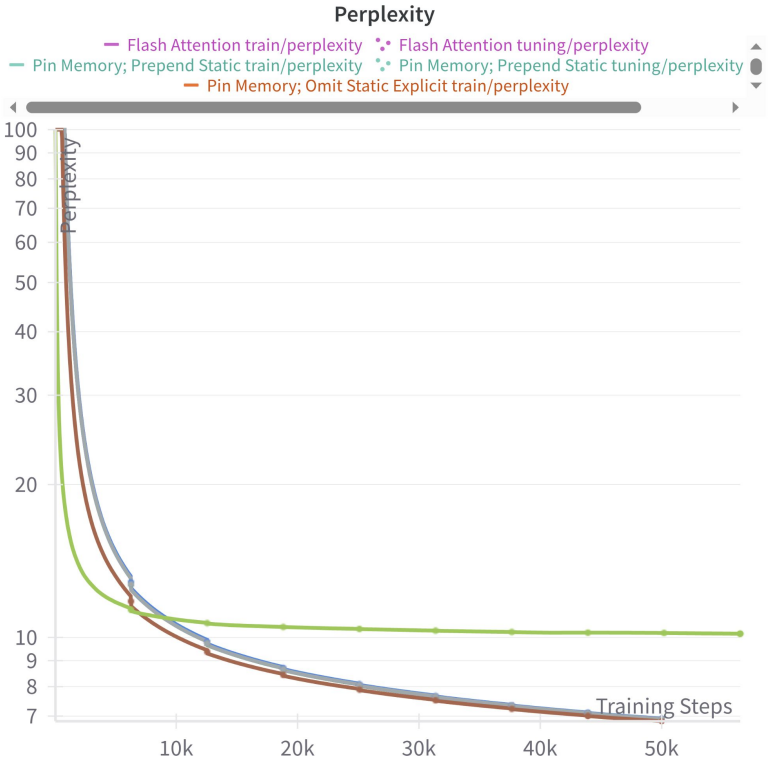
Batches

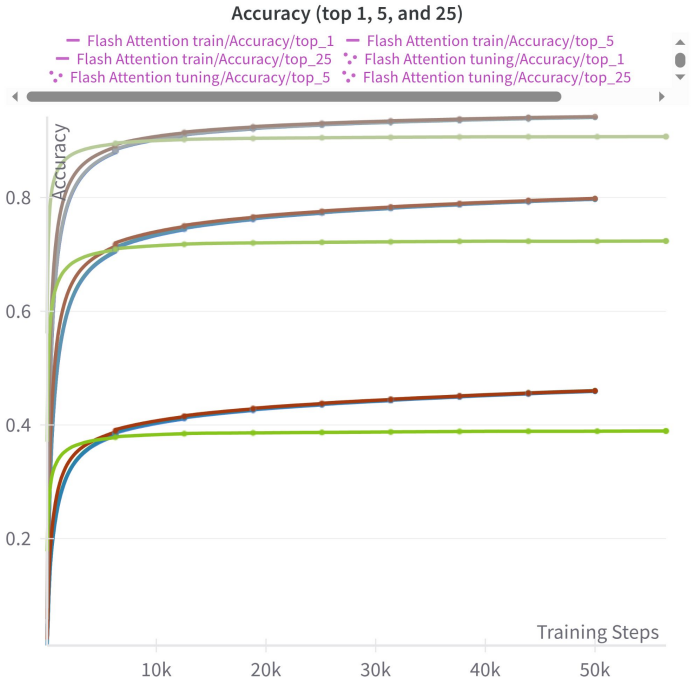
```
MEDSTorchBatch:
| Mode: Subject-Measurement (SM)
| Static data? ✓
| Labels? ✗
|
| Shape:
| | Batch size: 2
| | Sequence length: 5
| |
| | All dynamic data: (2, 5)
| | Static data: (2, 2)
|
| Data:
| | Dynamic:
| | | time_delta_days (torch.float32):
| | | | [[0.00e+00, 1.18e+04,  ..., 0.00e+00, 9.79e-02],
| | | |  [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)

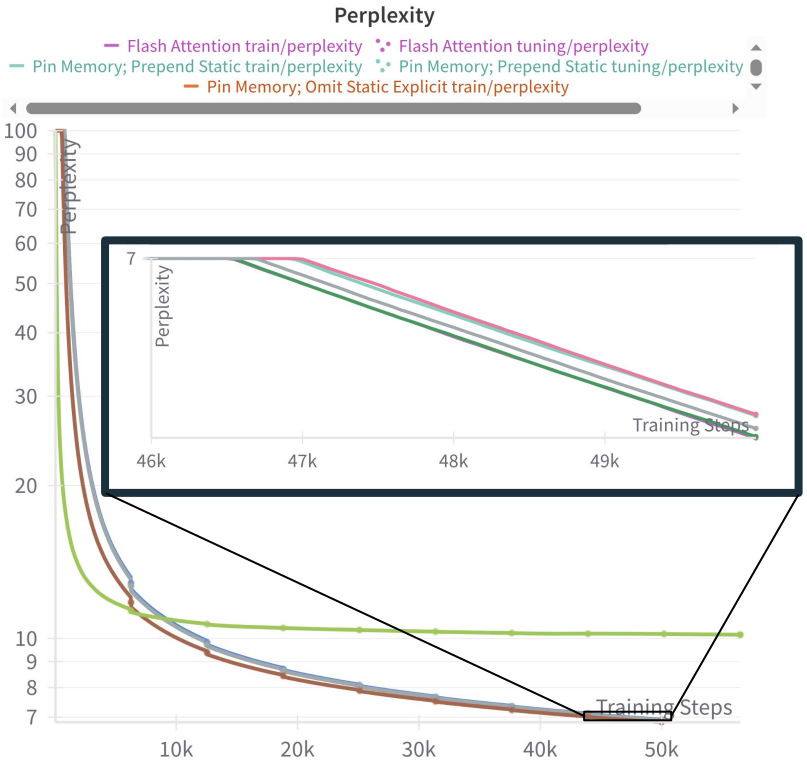


Training Convergence and Stability





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
```

What's next?

✓	*8:05 - 8:10*	Opening remarks
1.	*8:10 - 8:25*	What is MEDS?
2.	*8:25 - 8:35*	Guiding Problem Setup
3.	*8:35 - 9:00*	Convert your data into MEDS
4.	*9:00 - 9:25*	Extract a prediction task cohort with ACES
♨️	***9:25 - 9:40***	**Coffee Break [KDD COFFEE BREAK 9:30 - 10:00]**
5.	*9:40 - 10:30*	Develop a predictive model
6.	*10:30 - 11:00*	Participate in the open-source community

What's next?

✓	*8:05 - 8:10*	Opening remarks
1.	*8:10 - 8:20*	What is MEDS?
2.	*8:20 - 8:25*	Guiding Problem Setup
3.	*8:25 - 9:05*	Convert your data into MEDS
4.	*9:05 - 9:10*	Extract a prediction task cohort with ACES
5.	*9:10 - 9:30*	Develop a predictive model 1: Tabular Baseline
☕	*9:30 - 10:00*	Coffee Break
6.	*10:00 - 10:40*	Develop a predictive model 2: Neural Network
7.	*10:40 - 10:50*	Participate in the open-source community
8.	*10:50 - 11:00*	Closing Remarks

What's next?

✓	*8:05 - 8:10*		Opening remarks
1.	*8:10 - 8:20*		What is MEDS?
2.	*8:20 - 8:25*		Guiding Problem Setup
3.	*8:25 - 9:05*		Convert your data into MEDS
4.	*9:05 - 9:10*		Extract a prediction task cohort with ACES
5.	*9:10 - 9:30*		Develop a predictive model 1: Tabular Baseline
	*9:30 - 10:00*		Coffee Break
6.	*10:00 - 10:40*		Develop a predictive model 2: Neural Network
7.	*10:40 - 10:50*		Participate in the open-source community
8.	*10:50 - 11:00*		Closing Remarks