ACES: AUTOMATIC COHORT EXTRACTION SYSTEM FOR EVENT-STREAM DATASETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reproducibility remains a significant challenge in machine learning (ML) for healthcare. Datasets, model pipelines, and even task/cohort definitions are often private in this field, leading to a significant barrier in sharing, iterating, and understanding ML results on electronic health record (EHR) datasets. This paper addresses a significant part of this problem by introducing the Automatic Cohort Extraction System (ACES) for event-stream data. This library is designed to simultaneously simplify the development of task/cohorts for ML in healthcare and also enable the reproduction of these cohorts, both at an exact level for single datasets and at a conceptual level across datasets. To accomplish this, ACES provides (1) a highly intuitive and expressive configuration language for defining both dataset-specific concepts and dataset-agnostic inclusion/exclusion criteria, and (2) a pipeline to automatically extract patient records that meet these defined criteria from real-world data. ACES can be automatically applied to any dataset in either the Medical Event Data Standard (MEDS) or EventStreamGPT (ESGPT) formats, or to ***any*** dataset in which the necessary task-specific predicates can be extracted in an event-stream form. ACES has the potential to significantly lower the barrier to entry for defining ML tasks that learn representations, redefine the way researchers interact with EHR datasets, and significantly improve the state of reproducibility for ML studies in this modality. ACES is available at (anonymous repository link): `https://anonymous.4open.science/r/ACES-B5FB/`. A short video demonstration of ACES is available at (anonymous video link): `https://streamable.com/6og6l1`.

1 INTRODUCTION

Machine learning (ML) for healthcare suffers from a severe and systemic reproducibility crisis (McDermott et al., 2021b). This challenge is further exacerbated by the need to maintain private and secure datasets, but even with public datasets, ML pipelines are not reliably reproducible from published papers alone. For instance, in numerous attempts to reproduce ML for healthcare studies using the MIMIC-III dataset (Johnson et al., 2016), Johnson et al. found that more than half the time, the cohorts described in the studies could not be reliably reconstructed. Specifically, experiments led to many discrepencies of up to 25% with cohort sizes, with one study reaching as high as 11,767 patients (Johnson et al., 2017). This is primarily due to sparse descriptions of cohorts in the study methods, with essential details for reproducibility often omitted, along with the absence of openly available code.

This burden in reproducing even the basic task and problem definitions in ML for healthcare studies is profoundly detrimental. Beyond the obvious concerns it raises around the robustness of reported results and their readiness for deployment, our inability to reliably define shared, canonical, reproducible task definitions limits our capacity to perform meaningful model comparisons during methodological development. This is particularly notable in settings where not all researchers have mutual access to all datasets, as is common in healthcare. Given the critical role that open benchmarks play in the advancement of ML methods (Zhang & Hardt, 2024; Salaudeen & Hardt, 2024; Shirali et al., 2023), this deficit directly translates to a significant barrier in our ability, as a research community, to effectively experiment, iterate, and develop new ML methodologies in the healthcare space.

Given the clear import of this problem, the research community has naturally explored a number of prospective solutions. These can be largely categorized into two areas: (1) leveraging existing common data models (CDMs) to define reproducible task cohorts only for datasets within these schemas, and (2) defining static benchmarking tasks on individual public datasets. Both of these areas have generated numerous high-impact works. For example, in the area of CDM-driven tools, systems such as the ATLAS tool (OHDSI et al., 2024) for OHDSI's OMOP CDM (OHDSI, 2021) and i2b2's query tool for the i2b2 CDM (Murphy et al., 2010), as well as institution-specific tools such as the Stanford Research Repository (STARR) OMOP system and the NIHR Infections in Oxfordshire Database (IORD) cohort discovery platform have all been used to drive numerous new lines of inquiry. Unfortunately, these tools are also extremely limited in that they can only apply to the specific CDM or institutional data warehouse for which they have been defined. Further, because many of these CDMs have had limited penetrance into healthcare's high-capacity, deep learning ecosystems, they are particularly ill-posed for task and cohort extraction within the healthcare deep learning communities. Conversely, public static benchmarks (Harutyunyan et al., 2019; McDermott et al., 2021a; van de Water et al., 2024) over datasets such as MIMIC-IV (Johnson et al., 2023) or eICU (Pollard et al., 2018) have also been extremely impactful for the ML community. However, they are all tied to only a single or small set of datasets and tasks. Given the highly dynamic nature of clinical data and healthcare requirements, this is insufficient for the benchmarking and reproducibility needs faced by the ML for healthcare community.

When considering these existing solutions alongside the realities of healthcare data access and methodological development, it is clear that they are insufficient for three key reasons:

1. The Need for Interoperability The limited public datasets and only partially used CDMs cannot capture the diverse clinical populations, needs, and model capacities necessary for tangible ML progress in healthcare. To address this, systems for automated task extraction must be *meaningfully interoperable* across both public and private datasets with diverse input schemas.

2. The Need for Flexibility A single, static benchmark cannot encompass the variety of clinical tasks relevant to clinicians and informaticians. As existing tools (with only limited interfaces to define queries using per-set vocabularies) may struggle to generalize to new clinical tasks, ideal solutions must be *sufficiently flexible* to accommodate a myriad of new task definitions, criteria formats, and disease or deployment areas.

3. The Need for Accessibility, Usability, & Applicability in Deep Learning Workflows While many existing tools present with no-code interfaces (*e.g.*, web platforms to build queries) that are essential for less technically-literate audiences, integrating such tools with deep learning workflows can prove challenging. Deep learning systems are often run in a semi-programmatic manner on siloed, private computational clusters where researchers have minimal control. Hence, existing tools can cause significant hindrance. Instead, successful software must be able to provide a Python and command-line interface (CLI) that offer *significant ease of use* to deep learning researchers, alongside shareable and readable configuration files that specify task definitions in a manner that can be *readily ported* across datasets and environments.

Our Solution: Automatic Cohort Extraction System for Event-Stream Datasets In this work, we solve these problems with the Automatic Cohort Extraction System for Event-Stream Datasets (ACES). ACES offers a simple, expressive, and shareable configuration language for task definitions, as well as a reliable command-line interface and Python-accessible library for extracting labeled task dataframes (Figure 2).

Task definitions in ACES are naturally separated into simple **dataset-specific** event *predicates* and **dataset-agnostic** inclusion/exclusion *criteria*, thereby permitting the same task to be used in a *conceptually identical* manner across diverse datasets. By requiring users to specify predicates to realize their ML tasks on their specific datasets, ACES allows users to produce precise, locally specific, validatable cohorts that harmonize only the data elements needed for their task, regardless of how their input dataset is aligned or misaligned with existing ontologies or CDMs. Further, for datasets that are fully harmonized (*e.g.*, through OHDSI vocabularies), ACES predicate definitions can be reused across datasets without any loss of utility. In this way, ACES accommodates diverse datasets at various levels of data harmonization in a flexible, transparent manner. Overall, this approach

not only enhances reproducibility but also facilitates community collaboration on task definitions, inclusion/exclusion criteria, and evaluation metrics for specific clinical use cases.

In contrast to prior task-definition systems such as ATLAS, ACES makes minimal assumptions about the input data structure or source vocabularies. In particular, ACES can be run on ***any*** dataset, provided the necessary task-specific predicates can be pre-extracted in an "event-stream" format (Figure 1), and can further be run from raw data directly for any dataset in the relatively low-level and flexible Medical Event Data Standard (MEDS) (Arnrich et al., 2024) or EventStreamGPT (ESGPT) (McDermott et al., 2023) formats in approximately five lines of template code, offering high efficiency.To demonstrate the utility and flexibility of ACES, we also release a collection of task definitions (Section 3) based on prior ML for healthcare works at both the dataset-agnostic criteria level and with dataset-specific predicates for the widely-used MIMIC-IV dataset (Johnson et al., 2023).

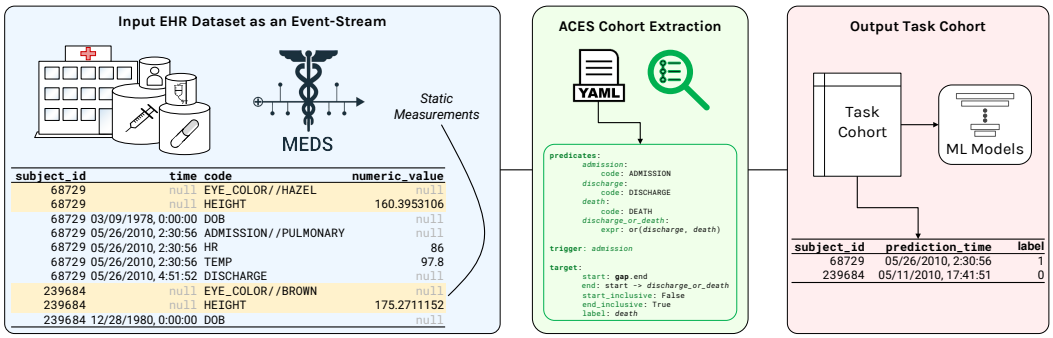

Figure 1: ACES workflow. The pipeline shows the expected format for ACES-supported event-stream datasets and outcome cohorts. The transformation of raw data into the event-stream format is intentionally designed to be straightforward–primarily merging relational database tables–minimizing data loss risks associated with other CDMs like OMOP.

Further, we align ACES with Donoho's 2023 concept of frictionless reproducibility for shared tasks, especially in the clinical domain (Donoho, 2023). This addresses the "Bring-Your-Own-Data Challenge", where research focuses on private patient outcomes data accessible to only a few credentialed researchers under usage agreements. This barrier remains problematic even when benchmarking platforms and shared code exist, as the lack of direct data sharing often stifles progress.

ACES seeks to overcome these challenges by offering novel infrastructure to ensure reproducibility without necessitating data sharing. Instead of relying on public datasets or reconfiguring code for diverse environments, ACES enables researchers to distribute task definitions through configuration files. These files provide a standardized way to conceptually reproduce cohorts on private datasets or exactly reproduce them on public datasets.

In sum, ACES represents 3 key contributions:

1. ACES defines a shareable, simple, and flexible task configuration language that can define diverse sets of prediction tasks for ML in healthcare on any event-stream dataset.
2. ACES provides an easy-to-use library to automatically extract these tasks from diverse sources of real-world, structured, and longitudinal electronic health record (EHR) data.
3. Through these advancements, ACES is poised to significantly advance the state of reproducibility, interoperability, and effective development of ML methods for healthcare.

In the rest of this work, we will do the following. First, in Section 2, we will go through ACES in more depth, beginning by illustrating the key concepts of ACES using a running example with ACES CLI. We then briefly overview its core recursive algorithm. Subsequently, in Section 3, we will demonstrate the use of ACES on diverse problem areas over real-world data, before ultimately discussing the limitations and future roadmap of ACES in Section 4, and offering concluding thoughts in Section 5.

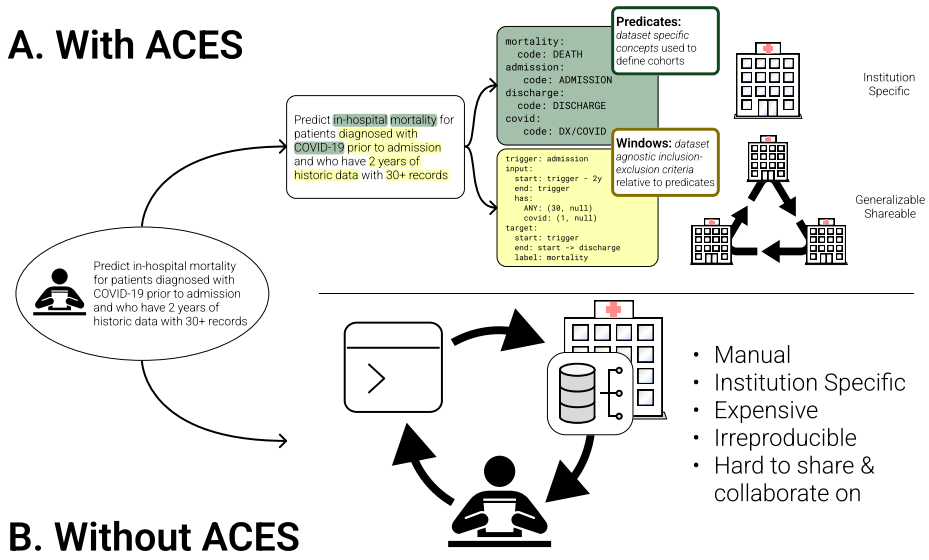

Figure 2: ML task cohort extraction process **(A)** with and **(B)** without ACES. Predicates are dataset-specific concepts that are needed to conceptually capture a machine learning task. Windows are temporal segments on a patient's health record and are dataset-agnostic, as they are defined relative to the predicates. This distinction allows researchers to share the more complex task logic that is independent of datasets, enabling conceptual reproducibility for ML tasks in healthcare.

2 Automatic Cohort Extraction System for Event-Stream Datasets (ACES)

In this section, we introduce ACES, a novel automated task and cohort definition and extraction system that fills the key gaps in *interoperability*, *flexibility*, and *accessibility* left by the existing tools outlined in Section 1.

To use ACES and extract a cohort for downstream ML tasks, a user only needs to do the following simple steps:

1. Install ACES: A fully functional version of ACES is pushed to PyPI, and any user can easily install it by simply running `pip install es-aces`. All dependencies are automatically set up with no further actions needed.

2. Define Dataset: A dataset in a permitted format, such as MEDS, ESGPT, or as direct predicates, is required. More information on the data formats is available in Section 2.1.

3. Define Task: A task configuration file is required to define the task that the user wishes to extract. This configuration language is simple, clear, yet flexible, permitting users to rapidly share and iterate over task definitions for their clinical settings. Configuration specification is given in Section 2.2.

4. Run the ACES CLI: ACES can be directly run from the command-line using:

```
$ aces-cli cohort˙name="$TASK" data.path="$DATA˙PATH"
```

Additional details about the possible command-line arguments are detailed in Section 2.3. ACES can also be used as a direct Python import, as detailed in Section 2.4.

5. Get Outputs: The resulting output from ACES is a single unified dataframe with all valid patient instances extracted according to task specifications. Users can subsequently leverage the

returned columns with original patient identifiers and health record timestamps for downstream ML tasks and benchmark creation. ACES also returns additional columns for the task label, as well as summaries of each of the predicates over windows of the patient record. Further information is available in Section 2.5.

Critically, after *only these five simple steps*, a user can immediately, reproducibly extract a full cohort from their source dataset that matches their task definition, and begin using this task for downstream representation learning. Due to space constraints, full technical details on all aspects of ACES and on the precise details of the recursive algorithm used to extract task cohorts given a configuration file are limited to the Appendix and the ACES public documentation, available here: **link redacted for anonymity, please refer to the PDF compilation of the technical documentation in the Supplementary Material (Pages 41-51)**.

2.1 Dataset Configurations

ACES is extremely flexible and can handle different data formats, including `data.standard=meds` (Arnrich et al., 2024), `data.standard=esgpt` (McDermott et al., 2023), or `data.standard=direct`, where event-stream predicate values are pre-extracted by the user from any given dataset schema. Other CDMs are interchangeable with these formats, such as OHDSI OMOP through the MEDS OMOP ETL, which transforms OMOP-compliant datasets into MEDS without data loss or scalability issues[1].

The use of direct predicates for extraction from datasets in a format that ACES does not natively support still significantly reduces the burden on users. Simply creating predicates is much less cumbersome than either fully converting the dataset to a CDM in order to use existing tools like ATLAS or i2b2's platform, or performing the entire task extraction from scratch by writing in-house dataframe querying code. Additionally, as ACES configuration files are shareable and easily portable to other datasets (by simply swapping out predicate defintions), we believe ACES will offer long-term efficiency benefits. This demonstrates the significant improvement in utility that ACES brings across diverse data schemas compared to existing tools.

2.2 Task Configurations

In ACES, tasks are specified through configuration files that define a collection of dataset-specific event predicates, which are simple functions evaluated on individual events within a structured event-stream dataset. Predicate definitions can be stored in a central "database" file specific to each dataset, such that previously defined features could be easily reused for a variety of downstream tasks without further effort. Community predicate contributions for public datasets also streamline collaborative efforts for reproducibility. Additionally, task criteria are defined in a dataset-agnostic manner through a collection of interrelated windows, which specify regions of a patient's record and are constrained by certain relationships. Please see Figure 3 for an example of a task configuration.

2.3 Command-Line Interface Arguments

Hydra Arguments The Hydra framework (Yadan, 2019) enhances the CLI by enabling flexible run configurations and argument parsing for cohort extractions. For instance, specific arguments are required to define the external source dataset for data loading. Depending on the chosen format (the `data.standard` argument), either the path to the data file (for `meds` or `direct`) or the path to the dataset directory (for `esgpt`) must be specified to indicate the external source data from which ACES will extract the cohort. Additionally, `cohort_dir` and `cohort_name` are essential for locating and loading the task configuration file, as well as for saving results and logging operational data.

Scaling to Large Datasets An overview of the computational profile of ACES is available in Section 3.1. Additionally, for users dealing with large datasets, ACES can also be run over a collection of sharded files, extracting and storing the matching cohort for each shard individually in match-

[1] `https://github.com/Medical-Event-Data-Standard/meds_etl/tree/main`

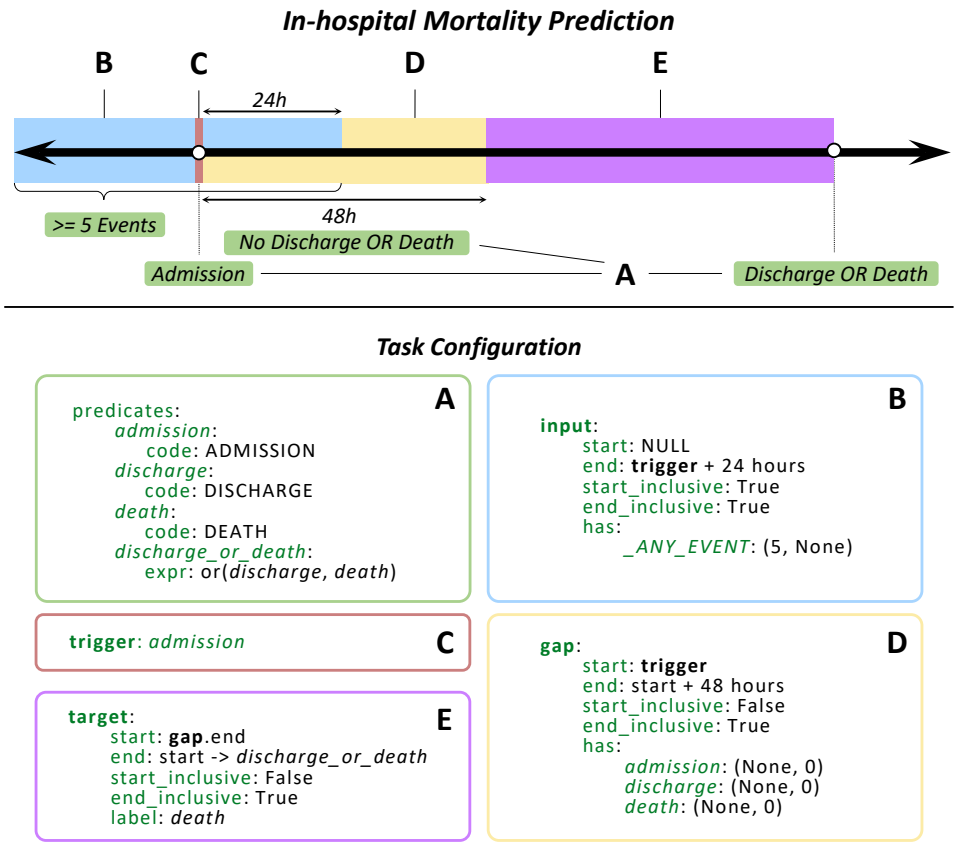

Figure 3: Example of a configuration file for the binary prediction of in-hospital mortality 48 hours after admission. References to predicates and windows are italicized and bolded, respectively. **(A)** Dataset-specific task predicates. These concepts are needed to conceptually capture this task and are used as constraints and boundaries for windows of the patient record. For instance, in this example, the value of "ADMISSION" denotes a hospital admission event in the external source dataset. **(B)** A window of the task specifying the task inputs for downstream models. Suppose we'd like to use all historic patient data up to and including 24 hours past the admission. An arbitrary criterion requiring more than 5 records can then be placed on this window to ensure that the extracted cohort contains sufficient input data. **(C)** The trigger events for the task, which are hospital admissions as we'd like to make a mortality prediction for each admission. **(D)** A window of the task specifying a gap in the patient timeline. Suppose we'd like to set a minimum length of admission for our cohort (*e.g.*, 48 hours). A temporal constraint (minimum window duration) of 48 hours could then be set to represent this requirement. **(E)** A window of the task specifying the task target, which is set from the end of *(D)* to the immediately subsequent "discharge" or "death" predicate. This creates our binary classes for the task (*i.e.*, $discharge = 0$, $death = 1$). All windows are interrelated on the patient timeline, as shown by how each window references another in the configuration file.

ing file paths. This can greatly increase computational efficiency by facilitating the processing of different shards in parallel via Hydra's multi-run launchers[2].

2.4 Python Usage

In addition to the command-line tool, we also provide a Python interface to allow researchers to easily leverage ACES for cohort extraction in their code pipelines. A full tutorial is provided at: **link redacted for anonymity, please refer to the PDF compilation of the tutorial notebook in the Supplementary Material (Pages 31-36)**.

[2] `https://hydra.cc/docs/1.0/plugins/joblib_launcher/`

```
$ aces-cli "
    --multirun "
    cohort_name="¡task_config_name¿" "
    cohort_dir="/directory/to/task/config/" "
    data=sharded "
    data.standard=meds "
    data.root="/directory/to/dataset/shards/" "
    "data.shard="$(expand_shards ¡folder¿/¡num¿)" # Sweeps over shards
```

2.5 Extraction Output

Finally, with a dataset configured for predicates and a task configuration, ACES will execute the extraction for the cohort and return a table where each row is a valid instance as per the criteria defined in the configuration file. Hence, each instance can be included in our cohort used for the downstream ML task. At the most basic level, the table contains the patient identifiers of our cohort, a user-defined timestamp that indexes prediction time, and a task label derived from a user-specified predicate. In addition, for each of the interrelated windows, a `start` and `end` timestamp is provided to segment the patient record, along with a summary of the number of predicates evaluated in said window.

2.6 Algorithm Design

ACES addresses the challenge of extracting meaningful windows of data from patient records by using a recursive approach grounded in a tree-structured configuration file. Each task is represented as a hierarchy of constraints, with nodes defining windows of interest and edges specifying temporal or event-based relationships between these windows. The algorithm begins by identifying root anchor events in the dataset that correspond to the triggering criteria of the task. It then recursively evaluates subtrees of constraints, aggregating predicate counts over defined windows either through temporal aggregations (e.g., over fixed time intervals) or event-bound aggregations (e.g., over windows bounded by specific events, such as discharges or diagnoses). Each step ensures that the criteria of the subtree are met, filtering out invalid realizations before proceeding to child nodes. This recursive process guarantees that the specified configuration can always be resolved into valid windows that meet the task's constraints. The final output is a dataframe containing all valid windows, task-specific labels, and prediction timestamps, and optionally, window start and end times as well as aggregated predicate counts. This ensures systematic and deterministic extraction of datasets for ML tasks. It also maintains flexibility and leverages ACES' simple, transparent, and highly expressive domain-specific language (DSL) Figure 4).

3 Using ACES: A Repository of Task Configuration Examples

To demonstrate the flexibility and utility of ACES, we define and publicly release the task configurations described in Table 1, both with dataset-agnostic criteria and with dataset-specific predicate realizations based on the MEDS version of the public MIMIC-IV dataset. Additional usage profiles, tasks, and areas, as well as details of the task cohorts extracted from MIMIC-IV, can be found in the Appendix and documentation. These various tasks have been previously studied, and ACES will facilitate their conceptual reproducibility to encourage benchmarking efforts and ensure robustness in ML for healthcare.

3.1 Computational Profile

To establish an overview of the computational profile of ACES, a collection of various common tasks was queried on MIMIC-IV (Johnson et al., 2023) in the MEDS format. The MIMIC-IV MEDS schema has approximately 50,000 patients *per shard* with an average of approximately 80,500,000 total event rows *per shard* over seven shards.

A **single** MEDS shard was used to provide a bounded computational overview of ACES, which is applicable even when scaled to larger datasets using Hydra[2]. For instance, if one shard costs M

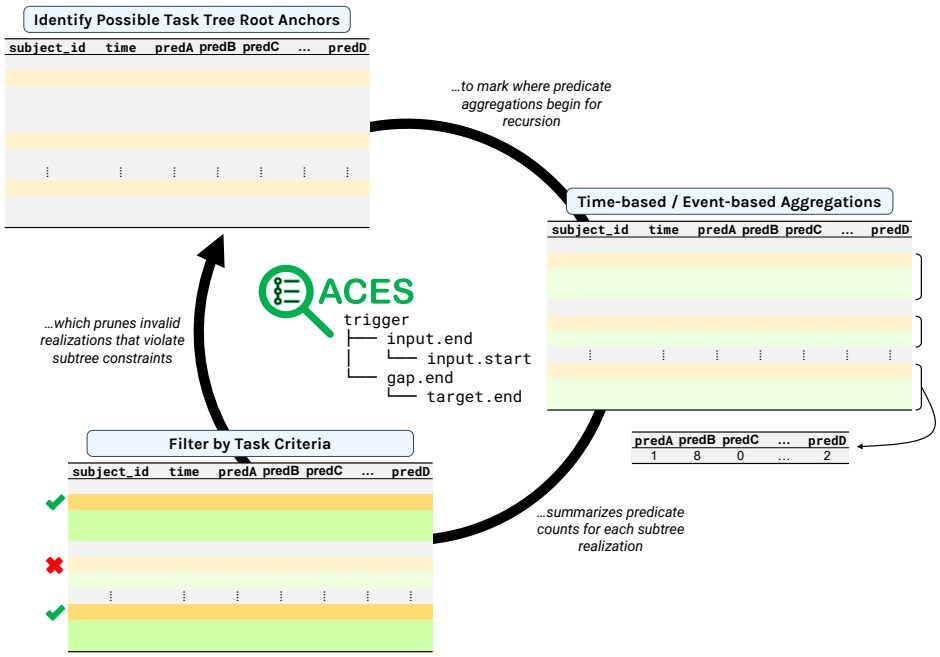

Figure 4: ACES recursive algorithm overview. Given a task tree generated from a configuration file, ACES first identifies possible roots of the tree (task triggers) based on the associated predicate. It then does aggregations of predicate counts over time-based (ie. time interval) or event-based (ie. window until a specified event) periods to summarize predicates over edges between nodes. Finally, invalid branches are filtered out if their predicate counts do not meet the specified criteria. This process is recursed for each child node of the original task tree.

Table 1: A collection of sample configuration files for various common predictive tasks on MIMIC-IV (Johnson et al., 2023). These tasks can be easily generalized to other datasets, such as e-ICU (Pollard et al., 2018) or other private intensive care unit (ICU) and inpatient datasets by simply swapping out appropriate predicate definitions. **Anonymized *pastecode.io* links have been embedded in place of identifiable links to GitHub repositories**.

Task Name	Description
First 24h in-hospital mortality	Predict **mortality** within a *hospital admission* using the first 24 hours of data from that admission.
First 48h in-hospital mortality	Predict **mortality** within a *hospital admission* using the first 48 hours of data from that admission.
First 24h in-ICU mortality	Predict **mortality** within an *ICU admission* using the first 24 hours of data from that admission.
First 48h in-ICU mortality	Predict **mortality** within a *ICU admission* using the first 48 hours of data from that admission.
30d post-hospital-discharge mortality	Predict **mortality** within 30 days of *discharge*.
30d re-admission	Predict **hospital readmission** within 30 days of *discharge*.
Myocardial infarction 1-5Y phenotyping	Predict **myocardial infarction (MI) incidence** 1-5 years after *hospital admission*.
Reduced echo-derived LVEF 9m post-ECG	Predict **reduced echo-derived left ventricular ejection fraction (LVEF)** within 9 months of any *ECG*.
CKD onset in diabetics 5Y from kidney panel	Predict **chronic kidney disease (CKD) onset** in diabetic patients within 5 years of any *kidney panel laboratory test*.

memory and T time, then N shards may be executed in parallel with $N * M$ memory and T time, or in series with M memory and $T * N$ time.

Table 2: Performance statistics for various common predictive tasks on a **single** MEDS shard of MIMIC-IV (Johnson et al., 2023), including the number of patients, the number of samples, total runtime (in seconds), and maximum memory usage (in MiBs). All experiments were executed on a Linux server with 36 cores and 340 GBs of RAM available.

Task	# Patients	# Samples	Total Time (s)	Max Memory (MiBs)
First 24h in-hospital mortality	20,971	58,823	363.09	106,367.14
First 48h in-hospital mortality	18,847	60,471	364.62	108,913.95
First 24h in-ICU mortality	4,768	7,156	216.81	39,594.37
First 48h in-ICU mortality	4,093	7,112	217.98	39,451.86
30d post-hospital-discharge mortality	28,416	68,547	182.91	30,434.68
30d re-admission	18,908	464,821	367.41	106,064.04
Myocardial Infarction 1-5Y phenotyping	3,329	8,319	198.04	33,427.70
Reduced echo-derived LVEF 9m post-ECG	14	17	210.02	35,385.79
CKD onset in diabetics 5Y from kidney panel	736	3,503	238.65	44,221.81

4 Discussion

4.1 Additional Related Work

In addition to the existing tools discussed in Section 1, there are several other areas of related work relevant to ACES. Firstly, ACES serves as a middle ground between solutions that focus on specific CDMs, such as i2b2 PIC-SURE hms (2024) and OHDSI ATLAS ohd (2023). Compared to these tools, ACES balances capability with greater ease of use and improved communicative value. ACES is also not tied to a particular CDM. Built on a flexible event-stream format, ACES is a no-code solution with a descriptive input format, permitting easy and wide iteration over task definitions. It can be applied to a variety of schemas, making it a versatile tool suitable for diverse research needs. Additionally, ACES could be directly connected with and support various health CDMs, such as OMOP, FHIR, PCORNET, and i2b2 (OHDSI, 2021; Bender & Sartipi, 2013; PCORnet, 2023; Murphy et al., 2010), rather than through ETLs. These models provide already-accepted standardized frameworks for organizing and analyzing healthcare data, and integrating ACES directly with them could greatly enhance its utility and interoperability. Similarly, frameworks such as DescEmb (Hur et al., 2022) and GenHPF (Hur et al., 2024) hold great synergistic potential with ACES, and we believe that they can be complementary in enabling new kinds of cross-dataset training, transfer learning, and evaluation. Static benchmarks that provide standardized datasets, evaluation metrics, and baseline methods for a range of clinical problems, such as YAIB (van de Water et al., 2024), Harutyunyan et al.'s multitask learning clinical prediction benchmarks (Harutyunyan et al., 2019), and EHR-PT (McDermott et al., 2021a), can also be directly integrated with ACES to facilitate robust ML in healthcare. Lastly, ACES can be used in conjunction with various health data management tools, such as ESGPT (McDermott et al., 2023), TemporAI (Saveliev & van der Schaar, 2023), PyHealth (Yang et al., 2023), and OMOPLearn (Kodialam et al., 2021). These tools offer functionalities for pre-processing, managing, and analyzing health data for downstream tasks, and integrating ACES with them directly can streamline ML workflows.

Beyond healthcare, ACES is applicable to data from a variety of other domains, such as for financial, climate, or social media data—essentially, ACES could be used for **any** *structured*, *longitudinal* data that can be reformatted as an event-stream. This versatility makes ACES a powerful library for extracting and analyzing complex event-based datasets across different fields.

4.2 Limitations & Future Roadmap

ACES has several key limitations that can be addressed in future work. Firstly, while already very expressive, ACES's task configuration language can still be further expanded. Expressing more complex kinds of predicates, window aggregations, labeling functions, and criteria would expand the scope of ACES significantly. ACES also seeks to provide direct support for cohort extraction based on unstructured data (notes and memos) in the future. Currently, such predicates need to be manually extracted by the user, but with the help of community contributions, we hope to be able to

incorporate automatic feature extraction from clinical notes, or even images, and integrate them into configuration files for cohort extraction.

ACES is also very well poised to capture more complex patterns of task and cohort relationships, including prescribed systems of case/control matching, automated bias analyses, or propensity re-weighting over excluded populations. It is also possible to enable users to nest ACES configuration files to leverage extracted task labels as new predicates in more complex ACES extraction processes.

To enhance the scalability of ACES, we will ensure that the expanding MEDS standard will **always** be supported. Direct interoperability with existing resources in this space, in particular ATLAS and OHDSI vocabulary-derived cohort definitions, is a high priority area for future work.

Finally, with the standardization that ACES offers, new opportunities for human interaction with data are also made available, such as via a natural language interface to define ACES predicates or configuration files and, thus, to extract downstream tasks, patient cohorts, or derived datasets in a **code-free manner** on diverse input EHR formats. We aim to also explore the viability of leveraging large language models (LLMs) to directly format predicate and criteria definitions given a data dictionary.

4.3 ACES for Enabling a New Kind of Benchmark

In addition to the clear impact of ACES on reproducibility, robustness, and accessibility of health datasets and ML for healthcare, we also feel that ACES is critical for a "new kind of benchmark" in the field—and, in so being, is a portent of what **needs to come** should ML for healthcare progress to a more productive, communal, and impactful stage.

In particular, we argue that for this field to progress in the manner desired by the community, and most likely to be maximally positively impactful for all patients, we need to develop methodologies to test, share, and develop ML solutions across diverse datasets in a meaningful and reproducible manner, **even without said datasets being publicly available** to general researchers. This capability is critical because, without it, we will never be able to offer new inductive insights about which methods are most likely to work best on novel, private data. In other words, if we cannot test our model training recipes across the diverse sets of clinical care settings, populations, and conceptual dataset schemas that exist in the real world, we similarly cannot expect those training recipes to generalize to said set of myriad downstream deployment areas.

Libraries like ACES, which make it as easy as possible for users to share the *conceptual* definitions of their tasks and prediction areas across datasets—in such a way that their colleagues can use them **even over independent, private datasets**—can help transform the kinds of benchmarking studies we can perform in ML for healthcare towards those that permit generalizable assessment of ML training recipes across datasets, clinical areas, and more.

5 Conclusion

In this work, we present the Automatic Cohort Extraction System for Event-Stream Datasets (ACES). ACES is a system designed to intuitively define cohorts and downstream tasks of interest for learning representations and reliably and automatically extract those cohorts from arbitrary datasets that are in an event-stream format. This system enables significantly greater shareability of task definitions, reproducibility of ML training and evaluation recipes, and is as easy to use as installing a package via `pip` and running a simple command-line tool. We feel that ACES will be integral in the development of new kinds of benchmarks in ML for healthcare, which can be explored across both public and private datasets alike, and characterize the populations and tasks of interest in a manner that cleanly separates dataset-specific components from shareable dataset-agnostic components. To learn more about ACES and use it today in your work, please visit our GitHub repository at (anonymous repository link): `https://anonymous.4open.science/r/ACES-B5FB/`, and the online ACES Documentation at (**link redacted for anonymity, please refer to the PDF compilation of the documentation in the Supplementary Material**).

REFERENCES

Dec 2023. URL https://github.com/OHDSI/Atlas.

2024. URL https://github.com/hms-dbmi/Access-to-Data-using-PIC-SURE-API.

Bert Arnrich, Edward Choi, Jason Alan Fries, Matthew B.A. McDermott, Jungwoo Oh, Tom Pollard, Nigam Shah, Ethan Steinberg, Michael Wornow, and Robin van de Water. Medical event data standard (MEDS): Facilitating machine learning for health. In *ICLR 2024 Workshop on Learning from Time Series For Health*, 2024. URL https://openreview.net/forum?id=IsHy2ebjIG.

Duane Bender and Kamran Sartipi. Hl7 fhir: An agile and restful approach to healthcare information exchange. In *Proceedings of the 26th IEEE international symposium on computer-based medical systems*, pp. 326–331. IEEE, 2013.

David Donoho. Data science at the singularity, 2023. URL https://arxiv.org/abs/2310.00865.

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96, June 2019. ISSN 2052-4463. doi: 10.1038/s41597-019-0103-9. URL https://www.nature.com/articles/s41597-019-0103-9. Publisher: Nature Publishing Group.

Kyunghoon Hur, Jiyoung Lee, Jungwoo Oh, Wesley Price, Younghak Kim, and Edward Choi. Unifying heterogeneous electronic health records systems via text-based code embedding. *PMLR*, pp. 183–203, Apr 2022. URL https://proceedings.mlr.press/v174/hur22a.html.

Kyunghoon Hur, Jungwoo Oh, Junu Kim, Jiyoun Kim, Min Jae Lee, Eunbyeol Cho, Seong-Eun Moon, Young-Hak Kim, Louis Atallah, and Edward Choi. Genhpf: General healthcare predictive framework for multi-task multi-source learning. *IEEE Journal of Biomedical and Health Informatics*, 28(1):502–513, Jan 2024. doi: https://doi.org/10.1109/jbhi.2023.3327951. URL https://arxiv.org/abs/2207.09858.

Alistair E. W. Johnson, Tom J. Pollard, and Roger G. Mark. Reproducibility in critical care: a mortality prediction case study. In *Proceedings of the 2nd Machine Learning for Healthcare Conference*, pp. 361–376. PMLR, November 2017. URL https://proceedings.mlr.press/v68/johnson17a.html. ISSN: 2640-3498.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1, January 2023. ISSN 2052-4463. doi: 10.1038/s41597-022-01899-x. URL https://www.nature.com/articles/s41597-022-01899-x. Publisher: Nature Publishing Group.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Rohan Kodialam, Rebecca Boiarsky, Justin Lim, Aditya Sai, Neil Dixit, and David Sontag. Deep contextual clinical prediction with reverse distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 249–258, 2021.

Matthew McDermott, Bret Nestor, Evan Kim, Wancong Zhang, Anna Goldenberg, Peter Szolovits, and Marzyeh Ghassemi. A comprehensive ehr timeseries pre-training benchmark. In *Proceedings of the Conference on Health, Inference, and Learning*, pp. 257–278, 2021a.

Matthew B. A. McDermott, Shirly Wang, Nikki Marinsek, Rajesh Ranganath, Luca Foschini, and Marzyeh Ghassemi. Reproducibility in machine learning for health research: Still a ways to go. *Science Translational Medicine*, 13(586):eabb1655, March 2021b. ISSN 1946-6242. doi: 10.1126/scitranslmed.abb1655.

Matthew B.A. McDermott, Bret Nestor, Peniel N Argaw, and Isaac S. Kohane. Event stream GPT: A data pre-processing and modeling library for generative, pre-trained transformers over continuous-time sequences of complex events. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=hiO0735tmc.

Shawn N Murphy, Griffin Weber, Michael Mendis, Vivian Gainer, Henry C Chueh, Susanne Churchill, and Isaac Kohane. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17 (2):124–130, 2010.

OHDSI. Omop common data model. https://www.ohdsi.org/data-standardization/, 2021. Accessed: 2024.

OHDSI, Sigfried Gold, Chris Knoll, Anthony Sena, Frank DeFalco, Anton Abushkevich, Pavel Grafkin, Vlad Belousov, Alex Saltykov, Vitaly Koulakov, Sergey Suvorov, Anastasiia Klochkova, Maria Pozhidaeva, Jungmi Han, Anton Gackovka, Anton Stepanof, Mark Velez, Vlad Belousov, Chen Regen, Ekaterina Krivets, Semyon Titarenko, chgl, Joris Borgdorff, Valeri Antonov, Kwang Soo Jeong, Gowtham Rao, Alex Odysseus, Ajit Londhe, Alex Cumarav, Kai Kewley, Marc Suchard, Wonjun Hong, Tom White, Konstantin Yaroshovets, Shaun Turner, Taha Abdul-Basser, Richard D. Boyce, Tiago Novo, and Rowan Parry. Atlas, June 2024. URL https://github.com/OHDSI/Atlas. original-date: 2015-07-08T16:26:35Z.

PCORnet. pcornet: Common data model (cdm) specification, version 6.1, 2023. URL https://pcornet.org/wp-content/uploads/2023/04/PCORnet-Common-Data-Model-v61-2023_04_031.pdf.

Tom J Pollard, Alistair E W Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific data*, 5(1):1–13, 2018.

Olawale Salaudeen and Moritz Hardt. Imagenot: A contrast with imagenet preserves model rankings. *arXiv preprint arXiv:2404.02112*, 2024.

Evgeny S Saveliev and Mihaela van der Schaar. Temporai: Facilitating machine learning innovation in time domain tasks for medicine. *arXiv preprint arXiv:2301.12260*, 2023.

Ali Shirali, Rediet Abebe, and Moritz Hardt. A theory of dynamic benchmarks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=i8L9qoeZOS.

Robin van de Water, Hendrik Schmidt, Paul Elbers, Patrick Thoral, Bert Arnrich, and Patrick Rockenschaub. Yet Another ICU Benchmark: A Flexible Multi-Center Framework for Clinical ML, March 2024. URL http://arxiv.org/abs/2306.05109. arXiv:2306.05109 [cs].

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

Chaoqi Yang, Zhenbang Wu, Patrick Jiang, Zhen Lin, Junyi Gao, Benjamin P. Danek, and Jimeng Sun. Pyhealth: A deep learning toolkit for healthcare applications. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 5788–5789, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599178. URL https://doi.org/10.1145/3580305.3599178.

Guanhua Zhang and Moritz Hardt. Inherent trade-offs between diversity and stability in multi-task benchmark. *arXiv preprint arXiv:2405.01719*, 2024.

A DOCUMENTATION

The full ACES Online Documentation is available at (**link redacted for anonymity**). We have also included a compiled PDF version of this documentation in the Supplementary Material.

To answer specific questions about ACES, please see the below index (links to the external online documentation will be provided after de-anonymization; references to chapters and page numbers of the included PDF documentation are shown):

How do you use ACES?

1. *What is a task and how do you specify one?*

   Sample task descriptions and specifications are provided on the Task Examples page of the online documentation. Please refer to Chapter 3, Pages 21-26.

   1.1. *What are predicates and how do you specify them?*

   For an overview of predicates and how they form the foundation of ACES, please refer to the Predicates DataFrame section in Chapter 4, Pages 27-30.

   1.2. *What are windows and how do you specify them?*

   A window in ACES is defined as a segment in the patient record. For details on how to define a window, please refer to Chapter 1.3.3, Page 9.

2. *How do you extract a task from a dataset?*

   For general ACES usage instructions, please refer to Chapter 2.1, Pages 13-16. Additionally, brief end-to-end instructions are also available in Chapters 1.2 and 1.3, Pages 4-10.

   2.1. *Detailed Usage Instructions for ACES CLI*

   For detailed instructions on using ACES CLI, please refer to the Usage Guide in Chapter 2.2, Pages 16-20.

   2.2. *Tutorial for the ACES Python API*

   For a step-by-step tutorial on using the ACES Python API, please refer to the Code Example Notebook in Chapter 5, Pages 31-41.

3. *ACES with & vs. Other Tools*

   For an overview of how ACES could be used with other existing complementary tools for reproducible machine learning, please refer to Chapter 1.4.2, Page 10.

   For an overview of ACES and other existing alternative tools for semi- or fully-automated cohort extraction, please refer to Chapter 1.4.3, Page 11.

How does ACES work?

1. *What is the formal configuration language specification for ACES?*

   For technical details on the ACES configuration language, please refer to the Configuration Language Specification section in Chapter 6.1, Pages 41-44.

2. *Glossary of ACES Terminology*

   For a glossary of terminology used throughout ACES, please refer to the Algorithm Terminology section in Chapter 6.3, Pages 46-47.

3. *What is the ACES extraction algorithm?*

   For technical details on the ACES algorithm, please refer to the Algorithm Design section in Chapter 6.4, Pages 48-51.

4. *Full ACES Module API Documentation*

   For the complete ACES module documentation, including *doctests* that ensure algorithm correctness, please refer to the Module API sections in Chapter 8, Pages 55-125.

How well does ACES work?

1. *Computational Profile*

   For an overview of the computational profile of ACES, please refer to the Computational Profile section in Chapter 7, Page 53.

2. *Further Examples*

   For additional examples of configuration files and criteria of different machine learning for health tasks, please refer to the MEDS-DEV benchmarking effort on GitHub (**link redacted for anonymity**).