Pre-print version.To appear in PMLR, Volume 298, Machine Learning for Healthcare, 2025. 1–31

Iterative Learning of Computable Phenotypes for Treatment Resistant Hypertension using Large Language Models

Guilherme Seidyo Imai Aldeia guilherme.aldeia@ufabc.edu.br
Federal University of ABC, Santo André, São Paulo, Brazil

Daniel S. Herman[*] Daniel.herman2@pennmedicine.upenn.edu
Department of Pathology and Laboratory Medicine, University of Pennsylvania, Philadelphia, PA, USA

William G. La Cava[*] william.lacava@childrens.harvard.edu
Computational Health Informatics Program, Boston Children's Hospital, Harvard Medical School, Boston, MA, USA

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities for medical question answering and programming, but their potential for generating interpretable computable phenotypes (CPs) is under-explored. In this work, we investigate whether LLMs can generate accurate and concise CPs for six clinical phenotypes of varying complexity, which could be leveraged to enable scalable clinical decision support to improve care for patients with hypertension. In addition to evaluating zero-short performance, we propose and test a *synthesize, execute, debug, instruct* strategy that uses LLMs to generate and iteratively refine CPs using data-driven feedback. Our results show that LLMs, coupled with iterative learning, can generate interpretable and reasonably accurate programs that approach the performance of state-of-the-art ML methods while requiring significantly fewer training examples.

1. Introduction

Recent advances in neural network architectures, particularly transformers (Vaswani et al., 2017), have led to groundbreaking innovations in large language models (LLM), enabling commercial products such as ChatGPT (OpenAI, 2024) and Claude (Anthropic, 2024). As a result, LLMs have been widely applied across various domains, demonstrating their effectiveness in solving domain-specific problems, including in coding (Jiang et al., 2024), law (Lai et al., 2024), and healthcare (Zhou et al., 2023; Thirunavukarasu et al., 2023).

Given the complexity of LLMs and other artificial intelligence (AI) models, the robustness and intelligibility of their responses requires scrutiny. The interpretability of the recommendations of AI tools is an important factor considered by regulatory agencies, such as the US FDA (Food and Administration, 2022) and the European AI Office (Goodman and Flaxman, 2017), in assessing the risks of deploying AI-targeted decision support in healthcare. Post-hoc explanation methods (Lundberg and Lee, 2017), including recent methods applying representation engineering (Zou et al., 2023) and sparse auto-encoders (NG et al.,

[*] These authors contributed equally

2011), are fundamentally limited in that they cannot fully explain models' behavior (Ghassemi et al., 2021; Rudin, 2019). Additionally, LLM models are prone to *hallucinations* (*i.e.*, generating fabricated information) (Ouyang et al., 2022; Singh et al., 2024), and can show bias across demographic subgroups (Chen et al., 2024).

While previous research has explored LLMs for natural language processing (NLP) tasks, an overlooked question is whether they can generate useful computable phenotypes (CPs) (Banda et al., 2018). A CP is an algorithmic construct that identifies observable traits from patient electronic health record data (He et al., 2023), enabling the identification of patients with a shared condition of interest (Tasker, 2017). CPs are desirable for decision support because they are machine-executable and can also be both intuitively interpretable to clinicians and can precisely identify specific patient populations (Mo et al., 2015). However, conventional manual approaches to the construction of CPs require considerable time and effort from clinical experts and data analysts, and as such, they do not scale well across phenotypes and struggle to adapt to differences across clinical practices or changes over time. In fact, He et al. (2023) reviewed CP generation literature, finding that 40% of the CP does not use ML — and when CP generation relies on ML, authors often choose models such as SVM and random forests, and perform feature selection manually, by choosing features related to previous studies, or guidelines, but rarely use automated generation processes. As LLMs are trained on medical literature and structured datasets (Liu et al., 2024) and perform well on medical Q&A tasks, they likely encode sufficient information regarding the relationships among clinical concepts to connect clinical concepts provided in prompts to available EHR data elements, enabling the composition of CPs. As opposed to the direct usage of LLMs for making clinical predictions (Jiang et al., 2023), using LLMs to compose CPs may yield efficient, standalone models that can be transparently assessed for performance and intelligibility.

Computable phenotyping is inherently a *program synthesis* problem, for which the goal is to solve a computer programming problem autonomously. The task of generating CPs as a program synthesis problem was studied using supervised machine learning (ML) methods, such as symbolic regression (SR), which simultaneously optimizes the model structure and parameters to produce simple and interpretable models (La Cava et al., 2023). Recent work has shown the potential for LLMs to tackle general program synthesis (Liventsev et al., 2023), especially when used as components in an iterative process including the steps of synthesizing, executing, and debugging (Gupta et al., 2020).

As a new technology, it is largely unknown how to best apply current LLMs for CP development. To that end, this paper aims to address the following research questions:

1. Can LLMs generate clinically meaningful CPs for hypertension-related conditions? If so, how much detail does the prompt require?

2. How accurate and concise are LLM-generated models compared to those produced by interpretable ML methods?

3. Can an iterative refinement improve LLM-generated CPs?

We investigate the capability of LLMs to generate CPs for three phenotypes of increasing complexity: hypertension (HTN), hypertension with unexplained hypokalemia (HTN-HypoK), and apparent treatment-resistant hypertension (aTRH). Because LLMs are trained on large corpuses of code, we decide to explore their ability to compose CPs as simple Python programs. Our approach differs from other CP generation ones by leveraging natural language descriptions of the phenotypes to automatically guide CP construction, using less data to perform iterative optimization. We acknowledge LLMs are speculative, but as an intermediate step, we can focus on the final CP for using in clinical practice.

We targeted hypertension and these sub-phenotypes because of the substantial opportunity for leveraging such CPs for clinical decision support. One critical need is to improve screening for primary aldosteronism, which is recommended for patients with HTN-HypoK or aTRH by specialty society guidelines (Funder et al., 2016). Primary aldosteronism causes cardiovascular disease at rates that are even higher than other causes of hypertension (Milliez et al., 2005). It is thought to affect up to ≈ 1% of US adults and is treatable, and sometimes curable, when identified but it is only diagnosed in fewer than 10% of affected patients (Käyser et al., 2016; Funder, 2016; Cohen et al., 2020; Reincke et al., 2012; Lin et al., 2012; Catena et al., 2007). Pilot studies have demonstrated that targeted decision support can increase screening of eligible patients from 2% to at least 16% (Passman et al., 2025), but it is difficult to manually adapt manually curated CPs from one clinical practice to another. By developing methodologies for automatically generating CPs, we can create a largely automated pipeline for adapting CPs across different localities or cohorts and adapting them over time to maintain performance.

We evaluate various aspects of CP generation, including: *i)* different LLMs, *ii)* different degrees of detail in the specification of the desired CP, *iii)* the number of features provided to the LLM, and *iv)* an iterative *synthesize, execute, debug, and instruct* (SEDI) strategy, wherein misclassified patients are provided as feedback to the LLM to refine the CP.

Our experiments are conducted using pre-processed EHR data. We compare LLM-generated models to interpretable ML methods, including decision trees, logistic regression, and a symbolic regression framework called FEAT (La Cava et al., 2019).

Our results showed that the LLMs generated concise CPs for all phenotypes analyzed. As expected, LLM-generated CPs were more accurate when the prompt included a more detailed description of the desired phenotype. The SEDI strategy improved performance, even when prompts did not include detailed phenotype definitions. To some extent, GPT-based models under-performed relative to the best supervised ML approaches in terms of cross-validated area under the precision-recall curve (AUPRC). However, the performance of the best LLM-generated CP (`gpt-4o+SEDI`) exhibited similar AUPRC and AUROC to a recently published, ML-based CP for predicting aTRH (La Cava et al., 2023) in held-out testing data. When performing parameter optimization on the final model, we could further increase its AUPRC performance in held-out testing data, outperforming the ML-based CP.

Generalizable Insights about Machine Learning in the Context of Healthcare

Computable phenotypes are useful tools for clinicians whose development traditionally demands a large amount of clinician time spent in chart review and model refinement. The expensive nature of the gold-standard labeling process for this task presents a general challenge for ML solutions. Our results suggest that LLMs can be leveraged to automate both the generation of computable phenotypes and the review and refinement process using a smaller amount of expert-curated samples than is typically required to train ML models.

We further find that, by using LLMs to generate CPs rather than black-box predictions, the model behavior becomes interpretable.

2. Related work

In healthcare, LLMs have been employed as NLP tools for tasks such as extracting concepts from clinical notes for postpartum hemorrhage patients without task-specific training (zero-shot) (Alsentzer et al., 2023), making meaningful inferences from physiological and behavioral time-series data by providing a few relevant examples in the prompt (few-shot) (Liu et al., 2023), and extracting logical conditions of inclusion/exclusion for phenotypes using concept codes (Tekumalla and Banda, 2024). LLMs are trained on vast amounts of data (Hoffmann et al., 2022), and can learn complex linguistic structures and rich representations of real-world information, thereby achieving remarkable performance in zero-shot or few-shot tasks. Jiang et al. (2023) described LLMs as "all-purpose prediction engines" for health systems due to their ability to perform well across five seemingly disparate tasks (30-day all-cause readmission prediction, in-hospital mortality prediction, comorbidity index prediction, length of stay prediction, and insurance denial prediction). The performance of LLMs on medical exam question-answering has become a *de facto* benchmark to track AI model advances (Jin et al., 2020; Singhal et al., 2023), although such benchmarking has recently been called into question given its specious relationship to usefulness in real-world tasks (Raji et al., 2025).

More detailed investigations into specific clinical applications of LLMs for EHR data, reviewed by Wornow et al. (2023), suggest that most meaningful clinical applications of LLMs, outside of high-level prediction tasks, are under-explored.

LLMs were recently evaluated for their ability to draft CPs in the form of SQL queries for three phenotypes: type 2 diabetes mellitus, dementia, and hypothyroidism (Yan et al., 2024). Another promising evaluation was done by Tekumalla and Banda (2024) of usefulness and reliability in retrieving phenotype concept codes was done by comparing LLM-generated phenotypes with OHDSI phenotype library (Banda et al., 2017) and HDRUK phenotype library (HRDUK Phenotype Library, 2024) by domain-experts.

Beyond clinical applications, LLMs have also been employed to generate concepts for discriminating input texts, then aggregating the concept scores into a white-box prediction model (Ludan et al., 2024). With a focus on medical calculations, MedCalc-Bench (Khandekar et al., 2024) evaluated the task of predicting common medical scores based on textual descriptions of patient records.

While these studies show LLMs in healthcare applications and computable phenotyping, to the best of our knowledge, no prior work directly aligns with our approach. Existing research has primarily focused on using LLMs for prediction tasks, medical question-answering, and phenotype retrieval, largely in the zero-shot context. In contrast, our study provides a detailed investigation into generating CPs with LLMs, including through automated refinement, in comparison to expert and ML-based solutions.

3. Methods

LLMs generate text by predicting the next token in a sequence, using as context the previous tokens. A token is a fundamental unit of text that can be a word or sub-word, encoded in a numerical embedding. The novelty introduced by transformers is the efficient processing of long-range token-token dependencies. By default, LLMs select the most probable next token given the specified preceding context. Some parameters, such as *temperature* (which perturbs the probability space when generating the next token) (Grubisic et al., 2024) or *nucleus sampling* (which restricts token choices to within a cumulative probability threshold) (Ravfogel et al., 2023) adjust the variability in selection of the next token.

To enhance their ability to follow structured instructions, LLMs can be fine-tuned to the *instruct* task (Wei et al., 2022), using datasets containing instruction-response pairs. In this setup, prompts are categorized into *system prompts* (guiding overall model behavior) and *user prompts* (task-related queries). Instruct models have no memory, so the entire conversation is used as context when generating a new response.

3.1. LLM-Guided CP Generation

Fig. 1 illustrates our approach for leveraging LLMs to construct CPs. We explore three target phenotypes, specify these phenotypes in the prompt using two levels of detail, and provide either the full set of features or a smaller, pre-selected set of features.

We explore two modes for prompt composition:

Zero-shot prompts The LLM generates a Python function that predicts phenotype probabilities based on the available features without receiving feedback.

SEDI prompts The process follows a synthesize-execute-debug-instruct (SEDI) loop (Gupta et al., 2020) iteratively receiving feedback regarding the CP's performance on training dataset. If the CP fails to execute, the LLM receives a message containing the error traceback (Debug). If the CP is successful, the LLM receives performance metrics, as well as example false positive (FP) and example false negative (FN) cases, and is instructed to refine its phenotype definition to improve the program's performance (Instruct). When providing FP and FN cases, information about the features being currently used by the phenotype, and additionally randomly sampled features are provided. We set to 10 false positive and 10 false negative examples per iteration, capped at 10 iterations — using at most 200 samples. In practice, samples may be resampled, and some iterations may contain fewer false positive or false negative examples, with 200 serving as a strict upper bound.

The SEDI prompts relies on giving examples for the model to perform small adjustments with supervised feedback, as including examples of inputs and desired outputs can improve overall performance (Murr et al., 2023). Our integration of the SEDI strategy significantly reduces reliance on labeled data compared to traditional supervised machine learning methods. In our approach, we retain the history of all models generated during the iterative phase, and at the end select the model that performs the best on training data as the final CP.

The *problem definition* prompts are presented below. The additional SEDI prompts are detailed in Appendix A.

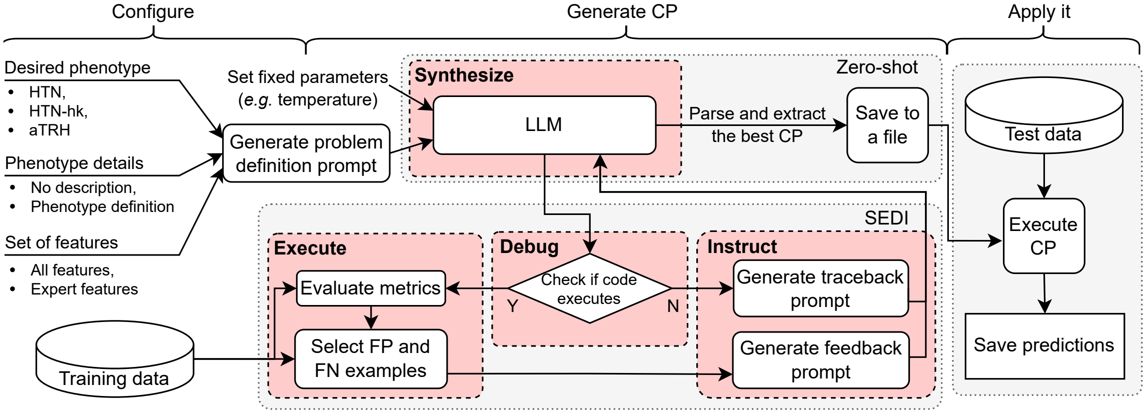

Figure 1: Overview of our proposed method for generating computable phenotypes with LLMs. A prompt is constructed by selecting a phenotype, specifying the level of description detail, and choosing the set of available features. In the zero-shot approach, the LLM generates a Python function to estimate phenotype probability using the listed features. With the SEDI strategy, a loop performs the following iteration: the CP is applied to the training dataset and the LLM is informed regarding the overall performance metrics and provided with false positive (FP) and false negative (FN) examples. Finally, after saving the CP to a file, it can be used to predict probabilities for held-out data as a standalone function.

System prompt You are an AI assistant that generates Python code based on a plain-text description of a function's purpose. You will receive a statement describing what the function should do. Your response must contain only a Python function, with no comments or explanations, that strictly follows the given description.

User prompt Please create a Python function named 'predict_hypertension' that takes a pandas DataFrame named 'df' as input. The function should assess whether each patient (represented as rows) has evidence of <`phenotype` description>. The function must return an array of floats representing the probability for each row. The available columns and their meanings are provided as key value pairs in the following dictionary: '<`variable dict`>'. You may only use the features whose names appear in this dictionary.

4. Cohort

Our experiments were conducted using EHR data from 1200 patients who received longitudinal primary care in the University of Pennsylvania Healthcare System (UPHS) and had chart review. The study data was previously reported in La Cava et al. (2023). Subject inclusion criteria were: (a) at least five outpatient visits in at least three separate years between 2007 and 2017, (b) at least two encounters at a single primary care practice sites, and (c) age 18 years or older. Table 1 describes the population comprising the data, including distribution of the three diagnoses we study here.

The data contains a total of 331 features extracted from EHR data. The features include demographic and clinical features such as age, sex, race, hospital proximity, weight, BMI,

Table 1: Demographic and diagnosis statistics of the study population. Values are counts (%) unless otherwise noted.

			Grouped by aTRH	
		Overall	False	True
n		1199	1023	176
Age, mean (SD)		57.2 (18.5)	55.0 (18.2)	70.4 (14.0)
Race	Black	338 (28.2)	251 (24.5)	87 (49.4)
	Other	108 (9.0)	100 (9.8)	8 (4.5)
	White	753 (62.8)	672 (65.7)	81 (46.0)
Sex	F	737 (61.5)	631 (61.7)	106 (60.2)
	M	462 (38.5)	392 (38.3)	70 (39.8)
HTN	False	591 (49.3)	591 (57.8)	0 (0)
	True	608 (50.7)	432 (42.2)	176 (100)
HTN	False	1027 (85.7)	924 (90.3)	103 (58.5)
HypoK	True	172 (14.3)	99 (9.7)	73 (41.5)

blood pressure, occurrences of elevated BP (systolc/diastolic $\geq 140/90$ mmHg) during encounters. Longitudinal features were aggregated as minimum, maximum, median, standard deviation, and skewness.

Laboratory results from 34 common tests (*e.g.*, metabolic panel, blood count, lipids, TSH, HbA1c) were aggregated using quartiles and medians. Diagnosis codes for hypertension and related comorbidities were summarized as annual medians and totals. Medication prescriptions were summarized as the number of days prescribed for each antihypertensive class and the count of encounters while prescribed 1, 2, 3, or 4 or more anti-hypertensive medications, described as sum, median, standard deviation, and skewness. Clinical notes were scanned for mentions of hypertension using regular expressions.

Features with physiologically implausible values, low variance (< 0.05), or sparse data ($< 5\%$ non-zero counts) were excluded. Missing values were imputed using the median. A detailed description is available in Appendix B.

4.1. Target computable phenotypes

Three clinical phenotypes were targeted, in order of increasing complexity: hypertension (HTN), hypertension with hypokalemia (HTN-HypoK), and apparent treatment-resistant hypertension (aTRH). For each phenotype, two labeling methods were used: "diagnosis" (Dx) based on expert chart reviews and "heuristic", an expert-designed clinical protocol refined iteratively using training data to work as a simpler version of the phenotypic labels. Heuristics were initially developed on the basis of clinical data expertise and iteratively refined based on evaluation in subsets of the training data in this study. By evaluating performance on both heuristics and diagnosis labels, we can assess how well the LLM captures simpler heuristics and, subsequently, how well it represents more complex phenotypes.

The simple HTN heuristic consisted of the presence of two or more diagnosis codes for hypertension[1]. For HTN-HypoK, the heuristic was the presence of the HTN heuristic plus

[1] The data were recorded, pre-processed, and labeled, under the ICD-9 classification, which was in use at the time.

two of one of the following: diagnosis codes for hypokalemia, outpatient encounters with low blood potassium results, or ambulatory prescriptions for an oral potassium supplement. For aTRH, we refined a previously reported CP to label patients. It consisted of inclusion criteria: (a) with documentation of at least two outpatient encounters with elevated blood pressure while on antihypertensive medications from 3 distinct classes or (b) prescribed four or more distinct classes of antihypertensive medications. The exclusion criteria for aTRH included diagnosis codes for heart failure, heart transplant, or moderate to severe chronic kidney disease prior to meeting the above inclusion criteria.

For diagnosis labels ascertained by chart review, a study physician reviewed clinical charts and classified subjects with respect to three phenotypes. The classification was based on JNC7 Guidelines on Prevention, Detection, Evaluation, and Treatment of High Blood Pressure (Chobanian et al., 2003).

4.2. Prompt construction with phenotype and data description

We start by describing our definitions for each hypertension phenotype, used when constructing prompts with detailed phenotype descriptions. They were designed to provide precise context to the model, when cross-referenced with the provided dictionary for available EHR features.

Hypertension: 2 or more hypertension Dx codes;

Hypertension with hypokalemia: 2 or more hypertension Dx codes and either 2 or more low potassium test results, 2 or more potassium supplementation prescriptions, or 2 or more hypokalemia diagnosis codes;

Treatment resistant hypertension: 2 or more high blood pressure measurements while prescribed 3 or more hypertension medications or 2 or more encounters while prescribed 4 or more hypertension medications

We identified a small feature set sufficient for either precisely building or approximating the heuristics definitions, described in Table 2. To ensure the LLMs appropriately interpret the features' derivations, we provide precise feature descriptions. When not using the subset of features from Table 2, then the model is prompted with all features and their respective descriptions, potentially allowing it to capture more complex dependencies but creating a much larger prompt.

By using the prompt presented in Section 3 under "User prompt", we replace <`phenotype`> by the description. A simple prompt will contain only the name of the phenotype (i.e. "hypertension"), while the rich prompt is formatted as follows: <`phenotype`>, which we will define as <`description of the heuristic`>.

5. Experiments

We evaluated LLMs' capabilities in generating CPs for hypertension and subphenotypes thereof, and evaluated their performance on both the expert-defined heuristics and chart-reviewed patient diagnoses. The biggest knowledge gaps include understanding how prompts

Table 2: The "expert" features used to define the phenotypes in the expert-curated heuristics. Under the 'expert features' experiment setting, rather than receiving a data dictionary of all available features (n=311), the LLM receives this focused set.

Feature name	Feature description
mean/median_systolic	Mean/median of systolic blood pressure (SBP) measured
mean/median_diastolic	Mean/median of diastolic blood pressure (DBP) measured
bp_n	Total number of blood pressure (BP) measurements
high_bp_n	Number of high blood pressure measurements (SBP >= 140 or DBP >= 90)
high_BP_during_htn_meds_X	Number of high BP measurements (SBP >=140 or DBP >=90) while prescribed X hypertension medications (X=1, 2, 3)
high_BP_during_htn_meds_4_plus	Number of high BP measurements (SBP >=140 or DBP >=90) while prescribed four or more hypertension medications
sum_enc_during_htn_meds_4_plus	Total encounters while prescribed 4 or more hypertension medications
low_K_N	Total number of low potassium test results
test_K_N	Total number of potassium test results
Med_Potassium_N	Total number of potassium supplement prescriptions
Dx_HypoK_N	Total number of hypokalemia diagnoses
re_htn_sum	Sum of regex counts for hypertension in clinical notes

Table 3: LLM models considered in our experiments, accessed via the OpenAI API.

Model	Checkpoint date (yy-mm-dd)	Knowledge cut-off date (yy-mm)	Max tokens
`gpt-3.5-turbo`	24/04/09	21/10	16K
`gpt-4o-mini`	24/7/18	23/10	128K
`gpt-4o`	24/08/06	23/10	128K

influence phenotype generation, whether adding more features improves accuracy or unnecessarily increases complexity, and how LLM-composition compares with existing ML methods for the same task.

In our first set of experiments, we perform an extensive evaluation of different settings for the LLMs with each phenotype. We investigate how different LLM models respond to prompt richness and feature quantity, and whether the SEDI loop improves performance over ten iterations. Table 3 reports the LLMs evaluated in this study. We set the temperature to 0.5 and nucleus sampling (`top-p`) to 1.0, based on preliminary experiments, to allow some variability while keeping the generated programs concise.

Our second set of experiments focuses on comparing the best setting for prompt richness and feature set with other interpretable ML methods, namely decision trees (DT), logistic regression with L1 regularization (LR L1), random forests (RF), and a symbolic regression algorithm named Feature Engineering Automation Tool (FEAT) (La Cava et al., 2019). Although random forests are not inherently interpretable, they are powerful models and serve as a competitive accuracy benchmark for this task.

Methods were compared using 5-fold cross-validation (CV) over 75% (n=899) of the study subjects. Each fold was tested with ten different random seeds, as some methods (*e.g.*, RF, LLMs, FEAT) are stochastic. Non-LLM based method underwent additional hyperparameter tuning on each fold (*i.e.*, nested CV), using the hyperparameters in Table 6 in Appendix C.

We first compared the cross-validation performance across all 50 runs per phenotype and modeling strategy. Next, we focused on the most complex phenotype, aTRH, and selected from these runs the LLM-based CP with the best cross-validation performance and evaluated it on a held-out test set of 300 patients. The selected CPs went through an additional parameter optimization step aiming to maximize AUPRC, and the tuned version was also included in the comparisons.

Metrics To evaluate the performance of the LLM-generated CPs and ML models, we report the area under the precision-recall curve (AUPRC) and the area under the receiver operating characteristic curve (AUROC).

In addition to manually interpreting a small set of generated CPs, we measured the size of each model as a scalable proxy for potential interpretability. In short, we approximate this model complexity by counting the number of computational components each model contains. DT model size is measured as the total number of nodes of the tree. RF is measured as the sum of number of nodes for every tree. LR L1 is measured by the number of features and arithmetic operations needed to build a linear combination of features with non-zero coefficients. FEAT is measured as the number of operations, constants, and features used in the mathematical function. To estimate the size of LLM-generated CPs, we first load the generated function into Python's abstract syntax tree library (`ast`) and then count the number of nodes in progam tree. Although imperfect, this size comparison gives us a valuable, if rough, sense of the potential interpretability of models returned by different methods.

6. Results

Fig. 2 reports the cross-validated AUPRC for the settings study. We compare the performance of each LLM (rows) in generating CPs for the six different phenotypes (columns). For each plot on the grid, we sort in the y axis each combination of the level of detail provided for the phenotype and the provided feature set, in order of increasing design complexity. For each combination, the bars represent the different prompting strategies. A boxplot version of this result is presented in Appendix D to better visualize the distribution.

In subsequent comparisons to other ML methods, for simplicity, we compare to LLM results using the SEDI strategy, as it appeared on average to be associated with the best overall performance for all LLM models. As a proxy for interpretability, we measure and plot the model sizes against the cross-validation AUPRC using the best llm settings and the SEDI strategy for all phenotypes in Fig. 3, also including the models found by other ML methods. In addition, Appendix E compares the distribution of validation AUPRC for aTRH across different LLM-generated CPs and interpretable machine learning methods.

Finally, we compared the performance of the best GPT-generated CP for aTRH to the FEAT CP reported in prior work, using 300 held-out test subjects. A single final model was selected for `gpt-4o` variants as follows: the best GPT-generated CPs were identified by selecting, from all CPs generated across cross-validation folds and iterations, the one with the highest performance.

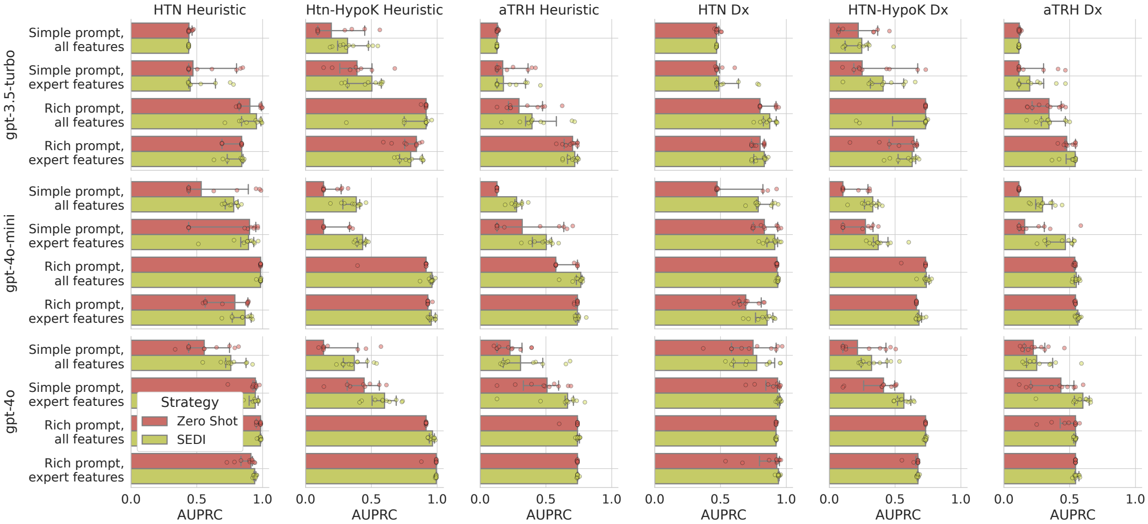

Figure 2: Average AUPRC on each held-out fold of the LLM-generated CPs across different prompting approaches. Prompts were varied to exclude/include detailed descriptions of the phenotype (*i.e.*, Simple, Rich) and include the entire set of features or only expert-selected features used in the heuristic (*i.e.*, all features, expert features). Whiskers depict the 95% confidence interval across different seeds.

We compare the performance of FEAT and `gpt-4o`-generated models at different levels of prompt richness and expert feature inclusion for aTRH in Table 4. The validation columns reports the cross-validation performance which also corresponds to the maximum observed performance on the train-validation data. The held-out column reports the performance on exclusive data with different prevalence. Additional parameter tuning was performed using a black-box optimization framework on the LLM-generated function generated with the SEDI strategy (See details in Appendix F), as it iteratively refines the program, to further assess whether the generated CPs holds reasonable parameters.

The `gpt-4o+SEDI` model with rich prompt and expert features (Fig. 4) achieves an average AUPRC of 0.79 and an AUROC of 0.90, with overlapping confidence intervals when comparing to FEAT. The parameter optimization appeared to increase the `gpt-4o+SEDI` final model's performance to AUPRC of 0.85 and AUROC of 0.94. The results show that the FEAT-trained model (Fig. 5) achieves an AUPRC of 0.80 and an AUROC of 0.94; overlapping confidence intervals suggest the `gpt-4o` methods are as good as the prior CP on this cohort. We also observe improvements on AUPRC but not on AUROC between `gpt-4o+SEDI` without parameter optimization and it's correspondent `zero-shot` setting, with increase from 0.61 to 0.86 AUPRC with simple prompt and all features, and from 0.71 to 0.79 with rich prompt and expert features. Appendix G presents the initial versions of the program, illustrating how the iterative SEDI strategy incrementally refined the model based on feedback.

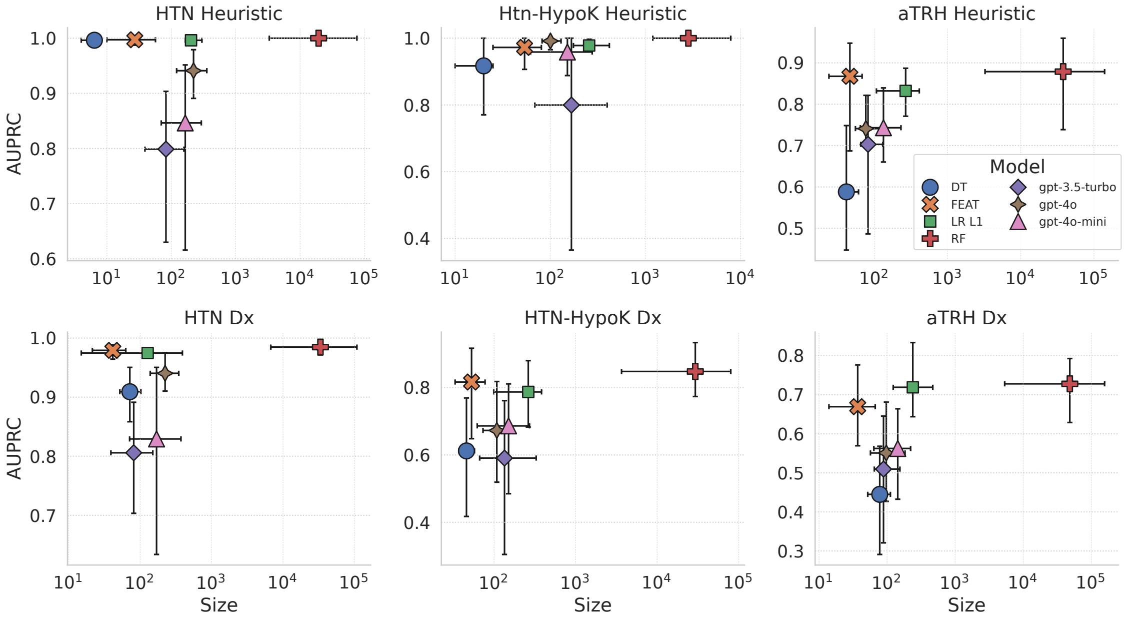

Figure 3: Trade-off between validation AUPRC (higher is better) and model size (smaller is better) for all phenotypes, using the SEDI strategy, rich prompts, and expert features. The error bars denote the 95% confidence intervals.

Table 4: Comparison between FEAT and best LLM-generated models from cross-validation for different levels of prompt richness and expert features for the most complex phenotype aTRH Dx. We performed parameter tuning of the final models with a black-box optimization framework. Values in parentheses represent 90% confidence intervals. Highlight denotes the best-performing method across all the rows and those whose confidence intervals overlap with it. Validation partition prevalence is 0.11. Held-out partition prevalence is 0.24.

Model	Strategy	Rich prompt	Expert features	Param. tuning	Model size	Held-out test	
						AUROC	AUPRC
FEAT	-	-	✗	✓	**44**	**0.94 (0.91 – 0.96)**	0.80 (0.71 – 0.87)
RF	-	-	✗	-	5539	**0.96 (0.95 – 0.98)**	**0.90 (0.84 – 0.94)**
GPT-4o	Zero-Shot	✗	✗	✗	68	0.86 (0.86 – 0.86)	0.61 (0.61 – 0.61)
		✓	✓	✗	8	0.93 (0.90 – 0.95)	0.71 (0.64 – 0.79)
	SEDI	✗	✗	✗	57	**0.95 (0.93 – 0.97)**	**0.86 (0.79 – 0.91)**
		✓	✓	✗	59	**0.95 (0.92 – 0.96)**	0.79 (0.72 – 0.87)
		✗	✗	✓	57	0.90 (0.86 – 0.92)	0.74 (0.65 – 0.82)
		✓	✓	✓	59	**0.94 (0.91 – 0.96)**	**0.85 (0.78 – 0.91)**

LLM-generated aTRH computable phenotype

```
def predict_aTRH(*features):
  prob = 0.0
  if high_BP_during_htn_meds_3 >= 2:
    prob += 0.31  # 0.4
  if sum_enc_during_htn_meds_4_plus
      ↪>=2:
    prob += 0.28  # 0.4
  if mean_diastolic > 80:
    prob += 0.07  # 0.1
  if mean_systolic > 140:
    prob += 0.12  # 0.1
  if high_BP_during_htn_meds_2 > 5:
    prob += -0.16 # 0.1
  if high_BP_during_htn_meds_3 > 5:
    prob += 0.02  # 0.1

  prob = min(1.0, prob)

  if Med_Potassium_N > 0 and
      ↪Dx_HypoK_N > 0:
    prob *= 0.45  # 0.5

  if mean_systolic<130 and
      ↪mean_diastolic<75:
    prob *= 0.77  # 0.5

  return prob
```

Figure 4: Final aTRH CP generated by `gpt-4o+SEDI` strategy with expert features and rich prompts. Constants were obtained using a parameter optimizer. Original constants are shown as comments. Held-out performance AUPRC of 0.85 and AUROC of 0.94.

FEAT aTRH computable phenotype

```
def predict_aTHR_FEAT(*features):
  acc = 0
  if sum_enc_during_htn_meds_3 > 1:
    acc += 1.33
  if median_enc_during_htn_meds_4_plus
      ↪ >1.25:
    acc += 0.49
  if mean_systolic >= 128.64:
    acc += 0.95
  if max_CALCIUM >= 10.15:
    acc += 0.4
  if re_htn_spec_sum > 40:
    acc += 0.42

  acc += -0.52*
      ↪sd_enc_during_htn_meds_2
  offset = -3.96

  logit = 1/(1+exp(-(offset+acc)))

  return logit
```

Figure 5: Final aTRH CP generated by FEAT, adapted to Python from Figure 5b of La Cava et al. (2023). Held-out performance AUPRC of 0.80 and AUROC of 0.94.

7. Discussion

From Fig. 2, we can see that the rich phenotype description strongly impacted LLM-derived CP performance for most phenotypes. This is expected, particularly for the heuristic phenotypes, as providing a precise specification in the prompt narrows the complexity of the task for the LLM. If the LLMs could perform well using the much simpler prompt, this would be a much more scalable and generalizable approach. Notably, `gpt-4o` appeared to perform much better than other LLMs using simple prompts, particularly using the iterative strategy and/or with an expert-curated feature set. In fact, from Table 4 we see that the best average performance of `gpt-4o+SEDI` with simple prompts outperformed its counterpart with rich prompts.

While the rich prompts contain the natural language definitions of the heuristics, they do not provide explicit code. As a result, the LLM is not always able to perfectly replicate the heuristic phenotypes as executable programs. At the same time, it reflects a real-world scenario in which no expert-driven feature selection is performed.

The SEDI strategy appeared to improve CP performance for many of the underperforming LLM-generated CPs. For the most complicated outcome, aTRH by chart review, SEDI appeared to improve all `gpt-4o-mini` CPs and both `gpt-4o` CPs provided with only the simple phenotype description. In fact, for the most minimal prompt designs (*i.e.*, simple prompts using all features), SEDI appeared to improve CP performance for `gpt-4o` and `gpt-4o-mini` across all phenotypes, and in cases where it did not, it caused no harm. However, for several of the outcomes, these SEDI-associated improvements did not yield equivalent performance to that of using the richer prompts. We expect future improvements to the SEDI process will lead to further improvements. It is also worth noting that `gpt-3.5-turbo` only showed substantial improvements with SEDI for the HTN-HypoK and aTRH heuristics.

Across experiments, the impact of providing an expert-curated feature set was inconsistent, when a rich prompt is presented to the model. For `gpt-4o` and the simple phenotype descriptions, it appeared overall associated with improved model performance. However, there were situations in which it appeared associated with worse performance, including for `gpt-3.5-turbo` and HTN-HypoK. The set of expert features had the greatest impact on `gpt-4o-mini` with simple prompts, suggesting that a smaller search space can compensate for the lack of a detailed phenotype description.

Additionally, the full feature set — comprising more than 300 features — expands the search space and often includes features with overlapping semantics. Using expert features appears to help by narrowing the search space and reducing prompt length. However, this is not a general rule, as competitive performance is still observed with the full feature set in several cases. This suggests that while curated features can reduce complexity, the method remains effective even when such features are not available. Model capability also plays a significant role — for example, `gpt-3.5` consistently underperforms compared to `gpt-4` variants.

Anecdotally, we observe that LLMs sometime tried to “cheat” on the task by, *e.g.*, creating random weights, or importing external libraries to fit a classifier, despite not having access to labels. However, the occurrence of these programs is drastically reduced when using SEDI strategy. In fact, when using SEDI, we observe that LLMs will, in some instances, create and refine a list of weights for the features, but without SEDI occasionally will generate weights as `weights=np.random.rand(len(features))`. Occurrences of programs with random weights or shortcuts are more common with zero-shot strategies for `gpt-3.5-turbo` and `gpt-4o-mini`.

From Fig. 3 we observe that ML-based strategies (except DT) tend to outperform LLM-based strategies on average. However, the performance differences between the best LLM models (`gpt-4o+SEDI`) and FEAT are typically small and `gpt-4o+SEDI` was superior for the HTN-hypoK heuristic. Notably, `gpt-4o+SEDI` underperforms on the simplest tasks (HTN Heuristic/Dx) where other ML methods excel, but is competitive on the more complex HTN-HypoK heuristic. The benefits of the SEDI strategy were apparent for `gpt-4o-mini` across both HTN and aTRH phenotypes and for `gpt-4o` for the HTN heuristic.

In Fig. 3 we see that `gpt-4o+SEDI` outperforms the other LLM variants, although their performance is close, with overlapping confidence intervals. `gpt-4o+SEDI` performance falls between the simplicity of the DT model and the complexity of the RF model, demonstrating a good balance between AUPRC and model size.

Even though gpt-4o+SEDI is outperformed by FEAT in individual trials, selecting the LLM final model from across the cross-validation experiment yields a final LLM model with performance comparable to that of the FEAT model, as reported in Table 4. If we consider model size as a proxy for interpretability, we see that gpt-4o+SEDI achieves better results than the black-box model generated by RF, whose ensemble exceeds five thousand nodes. In classification metrics, it achieves an AUPRC of 0.85, falling short of the best-performing method, RF, which reaches 0.90. However, compared to the traditional RF approach, our proposed SEDI method combined with the most recent LLM offers a better balance between performance and interpretability.

We also note from Table 4 that the zero-shot strategy without a rich prompt and expert features generates a larger model than that with SEDI, yet with decreased performance. When using a rich prompt and expert features, it generated an oversimplified model that likewise failed to achieve competitive performance. This indicates that SEDI may play an important in refining models to improve size and performance. The final CP for the gpt-4o using the zero-shot strategy has low variability, likely due to confidence intervals being computed by bootstrapping performance of a single selected program on a held-out test set.

The final CP model found by gpt-4o+SEDI, depicted in Fig. 4, reveals a concise set of intuitive, interpretable rules. We argue that interpretability comes mainly from the fact that the program can be inspected, and all instructions are clear enough to be used in clinical practice.

Taking as an example the final CP model generated by gpt-4o+SEDI, the program clearly assigns cumulative probabilities based on observed conditions. The program begins by checking for high blood pressure in patients prescribed three hypertension medications and correspondingly increases the cumulative probability for aTRH. The program then contin-ues to adjust the probability by interrogating other features more deeply and incorporating additional information. After calculating the final cumulative probability, it compensates for potential false negatives by reducing the probability based on number of potassium supplement prescriptions (Med_Potassium_N) and hypokalemia diagnoses (Dx_HypoK_N), which likely relates to bias in the subject sampling in the training set.

Overall, this final CP is interpretable, intuitively represents predictor interactions, and achieves competitive performance. Moreover, we demonstrated that supervised parameter optimization has the potential to considerably improve the performance of the LLM-trained models by tuning the parameters of the model, which are close to the actual parameters the LLM generated (represented as comments in the code).

We assess potential biases in the final model generated by gpt-4o+SEDI by evaluating its performance on each subpopulation defined by gender and race. The final model does not incorporate race or gender, but it shows worse AUPRC performance in Black women and white women. A detailed analysis is presented in Appendix H.

Limitations We note several limitations. First, we focused exclusively on hypertension and subphenotypes thereof; it is possible that different methods or settings would excel when applied to different phenotypes. Second, we did not evaluate the performance of other, potentially more advanced LLMs like gpt-o1, Claude or llama. Third, there may be variations of the SEDI strategy that would provide more effective feedback to the LLMs;

a full experimental study of these many options is beyond the current scope of this paper. The generated models are not calibrated to mitigate bias across subpopulations, which could potentially be addressed by incorporating fairness objectives in the prompts. Finally, our experiment is confined to a limited set of patients from a single (albeit large, multi-site) health system, partially due to the intensive nature of chart review. Because of this we cannot rule-out differences in the performance of these CPs across distinct patient populations.

8. Conclusions and Future Work

State-of-the-art LLMs generate reasonably accurate and concise CPs for hypertension phenotypes, even when a simple prompt is presented. When given detailed and focused prompts and equipped with data-driven, iterative feedback (*i.e.*, SEDI), LLM-generated CPs are competitive with those trained using supervised ML. We observe in general that traditional supervised ML approaches leveraging chart-reviewed examples still outperform LLM-derived CPs, but yield models that are much larger and require access to a much larger set of expert-labeled data.

We have also established the potential for LLMs to iteratively learn CPs expressed as intelligible Python code.

Our work presented a novel approach for using LLMs to generate CPs automatically, aiming to reduce manual feature engineering and leaning towards scalable solutions. We have made our SEDI framework publicly available so that it can be readily adapted to developing CPs for other conditions (*e.g.*, diabetes complications), given a small set of training examples.

Future work could extend the SEDI-based LLM approach used here, consider an expanded set of phenotyping tasks, or look at the performance of an iterative computable phenotyping system in prospective clinical settings.

Data and Code Availability Study data was extracted from the University of Pennsylvania Healthcare System (UPHS) electronic health record and cannot be shared publicly to protect the privacy of the subjects. However, it can be shared upon request and subject to relevant approvals. All our methods, experiments, and post-processing analysis are publicly available at https://github.com/cavalab/htn-phenotyping-with-llms.

Institutional Review Board (IRB) This study was reviewed and approved by University of Pennsylvania Institutional Review Board (#827260), which approved a waiver of informed consent. This study followed all relevant ethical regulations.

Acknowledgments

This work was supported by Patient-Centered Outcomes Research Institute (PCORI) Award (ME-2020C1D-19393). W.G. La Cava was supported by National Institutes of Health (NIH) grant R01-LM014300. G.S.I.A. is supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) finance Code 001. The statements in this work are solely the responsibility of the authors and do not necessarily represent the views of PCORI, its Board of Governors or Methodology Committee, or the NIH.

References

Emily Alsentzer, Matthew J. Rasmussen, Romy Fontoura, Alexis L. Cull, Brett Beaulieu-Jones, Kathryn J. Gray, David W. Bates, and Vesela P. Kovacheva. Zero-shot interpretable phenotyping of postpartum hemorrhage using large language models. npj Digital Medicine, 6(1), November 2023. ISSN 2398-6352. doi: 10.1038/s41746-023-00957-x. URL http://dx.doi.org/10.1038/s41746-023-00957-x.

Anthropic. Claude 3: Next-generation ai model, 2024. URL https://claude.ai/. Accessed: 2025-02-05.

Juan M Banda, Yoni Halpern, David Sontag, and Nigam H Shah. Electronic phenotyping with aphrodite and the observational health sciences and informatics (ohdsi) data network. AMIA Summits on Translational Science Proceedings, 2017:48, 2017.

Juan M. Banda, Martin Seneviratne, Tina Hernandez-Boussard, and Nigam H. Shah. Advances in Electronic Phenotyping: From Rule-Based Definitions to Machine Learning Models. Annual Review of Biomedical Data Science, 1:53–68, July 2018. ISSN 2574-3414. doi: 10.1146/annurev-biodatasci-080917-013315.

Cristiana Catena, GianLuca Colussi, Roberta Lapenna, Elisa Nadalini, Alessandra Chiuch, Pasquale Gianfagna, and Leonardo A. Sechi. Long-Term Cardiac Effects of Adrenalectomy or Mineralocorticoid Antagonists in Patients With Primary Aldosteronism. Hypertension, 50(5):911–918, November 2007. ISSN 0194-911X, 1524-4563. doi: 10.1161/HYPERTENSIONAHA.107.095448. URL https://www.ahajournals.org/doi/10.1161/HYPERTENSIONAHA.107.095448.

Shan Chen, Jack Gallifant, Mingye Gao, Pedro Moreira, Nikolaj Munch, Ajay Muthukkumar, Arvind Rajan, Jaya Kolluri, Amelia Fiske, Janna Hastings, Hugo Aerts, Brian Anthony, Leo Anthony Celi, William G. La Cava, and Danielle S. Bitterman. Cross-care: Assessing the healthcare implications of pre-training data on language model bias, 2024. URL https://arxiv.org/abs/2405.05506.

Aram V Chobanian, George L Bakris, Henry R Black, William C Cushman, Lee A Green, Joseph L Izzo Jr, Daniel W Jones, Barry J Materson, Suzanne Oparil, Jackson T Wright Jr, et al. Seventh report of the joint national committee on prevention, detection, evaluation, and treatment of high blood pressure. hypertension, 42(6):1206–1252, 2003.

Jordana B. Cohen, Debbie L. Cohen, Daniel S. Herman, John T. Leppert, James Brian Byrd, and Vivek Bhalla. Testing for Primary Aldosteronism and Mineralocorticoid Receptor Antagonist Use Among U.S. Veterans: A Retrospective Cohort Study. Annals of Internal Medicine, pages M20–4873, December 2020. ISSN 0003-4819, 1539-3704. doi: 10.7326/M20-4873. URL https://www.acpjournals.org/doi/10.7326/M20-4873.

Food and Drug Administration. Clinical decision support software - guidance for industry and food and drug administration staff, 2022. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/clinical-decision-support-software.

John W Funder. Primary aldosteronism as a public health issue. The Lancet Diabetes & Endocrinology, 4(12):972–973, December 2016. ISSN 22138587. doi: 10.1016/S2213-8587(16)30272-8. URL https://linkinghub.elsevier.com/retrieve/pii/S2213858716302728.

John W. Funder, Robert M. Carey, Franco Mantero, M. Hassan Murad, Martin Reincke, Hirotaka Shibata, Michael Stowasser, and William F. Young. The Management of Primary Aldosteronism: Case Detection, Diagnosis, and Treatment: An Endocrine Society Clinical Practice Guideline. The Journal of Clinical Endocrinology & Metabolism, 101 (5):1889–1916, May 2016. ISSN 0021-972X, 1945-7197. doi: 10.1210/jc.2015-4061. URL https://academic.oup.com/jcem/article/101/5/1889/2804729.

Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L. Beam. The false hope of current approaches to explainable artificial intelligence in health care. The Lancet Digital Health, 3(11):e745–e750, Nov 2021. ISSN 2589-7500. doi: 10.1016/S2589-7500(21)00208-9. URL https://doi.org/10.1016/S2589-7500(21)00208-9.

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision making and a "right to explanation". AI Magazine, 38(3):50–57, 2017. doi: https://doi.org/10.1609/aimag.v38i3.2741. URL https://onlinelibrary.wiley.com/doi/abs/10.1609/aimag.v38i3.2741.

Dejan Grubisic, Chris Cummins, Volker Seeker, and Hugh Leather. Priority sampling of large language models for compilers, 2024. URL https://arxiv.org/abs/2402.18734.

Kavi Gupta, Peter Ebert Christensen, Xinyun Chen, and Dawn Song. Synthesize, execute and debug: Learning to repair for neural program synthesis. Advances in Neural Information Processing Systems, 33:17685–17695, 2020.

Ting He, Anas Belouali, Jessica Patricoski, Harold Lehmann, Robert Ball, Valsamo Anagnostou, Kory Kreimeyer, and Taxiarchis Botsis. Trends and opportunities in computable clinical phenotyping: A scoping review. Journal of Biomedical Informatics, 140:104335, 2023.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 30016–30030. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

HRDUK Phenotype Library. HRDUK Phenotype Library, 2024. URL https://phenotypes.healthdatagateway.org/. Accessed 6 Feb 2025.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation, 2024. URL https://arxiv.org/abs/2406.00515.

Lavender Yao Jiang, Xujin Chris Liu, Nima Pour Nejatian, Mustafa Nasir-Moin, Duo Wang, Anas Abidin, Kevin Eaton, Howard Antony Riina, Ilya Laufer, Paawan Punjabi, Madeline Miceli, Nora C. Kim, Cordelia Orillac, Zane Schnurman, Christopher Livia, Hannah Weiss, David Kurland, Sean Neifert, Yosef Dastagirzada, Douglas Kondziolka, Alexander T. M. Cheung, Grace Yang, Ming Cao, Mona Flores, Anthony B. Costa, Yindalon Aphinyanaphongs, Kyunghyun Cho, and Eric Karl Oermann. Health system-scale language models are all-purpose prediction engines. Nature, 619(7969):357–362, July 2023. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-023-06160-y.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What Disease does this Patient Have? A Large-scale Open Domain Question Answering Dataset from Medical Exams, September 2020.

Nikhil Khandekar, Qiao Jin, Guangzhi Xiong, Soren Dunn, Serina S Applebaum, Zain Anwar, Maame Sarfo-Gyamfi, Conrad W Safranek, Abid A Anwar, Andrew Zhang, Aidan Gilson, Maxwell B Singer, Amisha Dave, Andrew Taylor, Aidong Zhang, Qingyu Chen, and Zhiyong Lu. Medcalc-bench: Evaluating large language models for medical calculations, 2024. URL https://arxiv.org/abs/2406.12036.

Sabine C. Käyser, Tanja Dekkers, Hans J. Groenewoud, Gert Jan van der Wilt, J. Carel Bakx, Mark C. van der Wel, Ad R. Hermus, Jacques W. Lenders, and Jaap Deinum. Study Heterogeneity and Estimation of Prevalence of Primary Aldosteronism: A Systematic Review and Meta-Regression Analysis. The Journal of Clinical Endocrinology & Metabolism, 101(7):2826–2835, July 2016. ISSN 0021-972X, 1945-7197. doi: 10.1210/jc.2016-1472. URL https://academic.oup.com/jcem/article-lookup/doi/10.1210/jc.2016-1472.

William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason H. Moore. Learning concise representations for regression by evolving networks of trees. In International Conference on Learning Representations, ICLR, 2019.

William G. La Cava, Paul C. Lee, Imran Ajmal, Xiruo Ding, Priyanka Solanki, Jordana B. Cohen, Jason H. Moore, and Daniel S. Herman. A flexible symbolic regression method for constructing interpretable clinical prediction models. npj Digital Medicine, 6(1):1–14, June 2023. ISSN 2398-6352. doi: 10.1038/s41746-023-00833-8.

Jinqi Lai, Wensheng Gan, Jiayang Wu, Zhenlian Qi, and Philip S. Yu. Large language models in law: A survey. AI Open, 5:181–196, 2024. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2024.09.002. URL https://www.sciencedirect.com/science/article/pii/S2666651024000172.

Yen-Hung Lin, Lian-Yu Lin, Aaron Chen, Xue-Ming Wu, Jen-Kuang Lee, Ta-Chen Su, Vin-Cent Wu, Shih-Chieh Chueh, Wei-Chou Lin, Men-Tzung Lo, Pa-Chun Wang, Yi-Lwun Ho, and Kwan-Dun Wu. Adrenalectomy improves increased carotid intima-media thickness and arterial stiffness in patients with aldosterone producing adenoma. Atherosclerosis, 221(1):154–159, March 2012. ISSN 00219150. doi: 10.1016/

j.atherosclerosis.2011.12.003. URL https://linkinghub.elsevier.com/retrieve/pii/S0021915011011336.

Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak Patel. Large language models are few-shot health learners. arXiv preprint arXiv:2305.15525, 2023.

Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. Datasets for large language models: A comprehensive survey. arXiv preprint arXiv:2402.18041, 2024.

Vadim Liventsev, Anastasiia Grishina, Aki Härmä, and Leon Moonen. Fully Autonomous Programming with Large Language Models. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 1146–1155, Lisbon Portugal, July 2023. ACM. ISBN 9798400701191. doi: 10.1145/3583131.3590481.

Josh Magnus Ludan, Qing Lyu, Yue Yang, Liam Dugan, Mark Yatskar, and Chris Callison-Burch. Interpretable-by-design text understanding with iteratively generated concept bottleneck, 2024. URL https://arxiv.org/abs/2310.19660.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Paul Milliez, Xavier Girerd, Pierre-François Plouin, Jacques Blacher, Michel E. Safar, and Jean-Jacques Mourad. Evidence for an increased rate of cardiovascular events in patients with primary aldosteronism. Journal of the American College of Cardiology, 45(8):1243–1248, April 2005. ISSN 07351097. doi: 10.1016/j.jacc.2005.01.015. URL https://linkinghub.elsevier.com/retrieve/pii/S0735109705002184.

Huan Mo, William K Thompson, Luke V Rasmussen, Jennifer A Pacheco, Guoqian Jiang, Richard Kiefer, Qian Zhu, Jie Xu, Enid Montague, David S Carrell, Todd Lingren, Frank D Mentch, Yizhao Ni, Firas H Wehbe, Peggy L Peissig, Gerard Tromp, Eric B Larson, Christopher G Chute, Jyotishman Pathak, Joshua C Denny, Peter Speltz, Abel N Kho, Gail P Jarvik, Cosmin A Bejan, Marc S Williams, Kenneth Borthwick, Terrie E Kitchner, Dan M Roden, and Paul A Harris. Desiderata for computable representations of electronic health records-driven phenotype algorithms. Journal of the American Medical Informatics Association, 22(6):1220–1230, November 2015. ISSN 1067-5027. doi: 10.1093/jamia/ocv112.

Lincoln Murr, Morgan Grainger, and David Gao. Testing llms on code generation with varying levels of prompt specificity. arXiv preprint arXiv:2311.07599, 2023.

Andrew NG et al. Sparse autoencoder. CS294A Lecture notes, 72(2011):1–19, 2011.

OpenAI. Chatgpt, 2024. URL https://chatgpt.com/. Accessed: 2025-02-05.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35:27730–27744, 2022.

Jesse E Passman, Jasmine Hwang, Justin Tang, Madeline Fagen, Mika Epps, MaryAnne Peifer, John T Howell, Jordana B Cohen, M Kit Delgado, Heather Wachtel, and Daniel S Herman. Active Choice Nudge to Increase Screening for Primary Aldosteronism in At-Risk Patients. Journal of the American College of Surgeons, 240(1):46–59, January 2025. ISSN 1072-7515, 1879-1190. doi: 10.1097/XCS.0000000000001221. URL https://journals.lww.com/10.1097/XCS.0000000000001221.

Inioluwa Deborah Raji, Roxana Daneshjou, and Emily Alsentzer. It's Time to Bench the Medical Exam Benchmark. NEJM AI, 2(2):AIe2401235, January 2025. doi: 10.1056/AIe2401235.

J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad, 2018.

Shauli Ravfogel, Yoav Goldberg, and Jacob Goldberger. Conformal nucleus sampling. arXiv preprint arXiv:2305.02633, 2023.

Martin Reincke, Evelyn Fischer, Sabine Gerum, Katrin Merkle, Sebastian Schulz, Anna Pallauf, Marcus Quinkler, Gregor Hanslik, Katharina Lang, Stefanie Hahner, Bruno Allolio, Christa Meisinger, Rolf Holle, Felix Beuschlein, Martin Bidlingmaier, and Stephan Endres. Observational Study Mortality in Treated Primary Aldosteronism: The German Conn's Registry. Hypertension, 60(3):618–624, September 2012. ISSN 0194-911X, 1524-4563. doi: 10.1161/HYPERTENSIONAHA.112.197111. URL https://www.ahajournals.org/doi/10.1161/HYPERTENSIONAHA.112.197111.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1(5):206–215, May 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0048-x. URL http://dx.doi.org/10.1038/s42256-019-0048-x.

Chandan Singh, Jeevana Priya Inala, Michel Galley, Rich Caruana, and Jianfeng Gao. Rethinking interpretability in the era of large language models. arXiv preprint arXiv:2402.01761, 2024.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Abubakr Babiker, Nathanael Schärli, Aakanksha Chowdhery, Philip Mansfield, Dina Demner-Fushman, Blaise Agüera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. Nature, 620(7972):172–180, August 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06291-2.

Robert C. Tasker. Why Everyone Should Care About "Computable Phenotypes". Pediatric Critical Care Medicine: A Journal of the Society of Critical Care Medicine and the World

Federation of Pediatric Intensive and Critical Care Societies, 18(5):489–490, May 2017. ISSN 1529-7535. doi: 10.1097/PCC.0000000000001115.

Ramya Tekumalla and Juan M. Banda. Towards automated phenotype definition extraction using large language models. Genomics & Informatics, 22(1):21, Oct 2024. ISSN 2234-0742. doi: 10.1186/s44342-024-00023-2. URL https://doi.org/10.1186/s44342-024-00023-2.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. Nature Medicine, 29(8):1930–1940, July 2023. ISSN 1546-170X. doi: 10.1038/s41591-023-02448-8. URL http://dx.doi.org/10.1038/s41591-023-02448-8.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

Michael Wornow, Yizhe Xu, Rahul Thapa, Birju Patel, Ethan Steinberg, Scott Fleming, Michael A. Pfeffer, Jason Fries, and Nigam H. Shah. The shaky foundations of large language models and foundation models for electronic health records. npj Digital Medicine, 6(1):1–10, July 2023. ISSN 2398-6352. doi: 10.1038/s41746-023-00879-8.

Chao Yan, Henry H Ong, Monika E Grabowska, Matthew S Krantz, Wu-Chen Su, Alyson L Dickson, Josh F Peterson, QiPing Feng, Dan M Roden, C Michael Stein, V Eric Kerchberger, Bradley A Malin, and Wei-Qi Wei. Large language models facilitate the generation of electronic health record phenotyping algorithms. Journal of the American Medical Informatics Association, 31(9):1994–2001, September 2024. ISSN 1527-974X. doi: 10.1093/jamia/ocae072.

Hongjian Zhou, Fenglin Liu, Boyang Gu, Xinyu Zou, Jinfa Huang, Jinge Wu, Yiru Li, Sam S Chen, Peilin Zhou, Junling Liu, et al. A survey of large language models in medicine: Progress, application, and challenge. arXiv preprint arXiv:2311.05112, 2023.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. arXiv preprint arXiv:2310.01405, 2023.

Appendix A. Prompts for the SEDI strategy

In this section, we provide a detailed description of the prompts used in the synthesize-execute-debug-inspect (SEDI) strategy.

The iteration begins with the same message for both SEDI and the zero-shot strategy, which includes a description of the phenotype and the available features. Then, after receiving the response from the LLM, we parse it to extract the generated program and evaluate it for execution errors.

If execution errors occur, the following message is used as a prompt:

Debug prompt Python encountered an error when trying to execute the function. Error Message: <error traceback as formatted text>. Please try again. **MAKE ABSOLUTELY SURE TO RETURN A SYNTACTICALLY VALID PYTHON FUNCTION.

If the program executes without errors, we evaluate its performance by checking for false positives and false negatives. For each case, we randomly select up to 10 examples from the training data and format them as input dictionaries. Notice that, in the worst case scenario, the model will see only 20 training samples each iteration, meaning that the training uses less data than a supervised ML method.

The updated instruction prompt consists of three components: the *performance report* message, the *false positive examples* (if applicable), and the *false negative examples* (if applicable). The second and third components are used only if there is any false positive or negative case, otherwise the model does not receive specific examples. During the first iteration, no improvement message is provided to the LLM. After the first iteration, the performance report message includes either a *reinforcement* message if the changes improve the model or a negative message otherwise.

Performance report message We evaluated the prediction function you provided on a set of < training dataset size> patients. <reinforcement message, if applicable>. Using the performance feedback below, please refine the Python function.
# Overall Performance
Area Under the Receiver-Operating Curve (AUROC): <AUROC, with 3 decimal places>
Area under the precision-recall curve (AUPRC): <AUPCR, with 3 decimal places>
The False Positive Rate is <FP rate, as percentage>
The False Negative Rate is <FN rate, as percentage>

False positive examples # Analysis of False Positives
Please refine the function so that the <number of false positives> False Positives have lower predicted probabilities
Below you will find <number of FP examples> example patients with false positive assessments to assist you in prescribing changes to the predict_hypertension function:
<List containing the FP examples, formatted as dictionaries>

False negative examples # Analysis of False Negatives
Please refine the function so that the <number of false negatives> False Negatives have higher predicted probabilities

Below you will find <number of FN examples> example patients with false negative assessments to assist you in prescribing changes to the predict_hypertension function:
<List containing the FN examples, formatted as dictionaries>

Finally, the prompt concludes with a *summary* message that reiterates the task description.

Summary # Summary of Request
Please create an updated Python function named ‘predict_hypertension‘ that achieves fewer false positives and fewer false negatives than the one you previously provided.
The function should assess whether each patient (represented as rows) has evidence of <phenotype>.
As before, the function takes a pandas DataFrame named ‘df‘ as input.
Recall that the available columns and their meanings are provided as key value pairs in a dictionary previously provided.
As before, you may only use the features whose names appear in this dictionary.
As before, your response must contain only a Python function, with no comments or explanations, that strictly follows the given description.

Appendix B. Feature construction

As previously described, 331 features were extracted from the EHR. Demographic and encounter features included age, race, sex, binned distance from zip code 19104, weight, BMI, blood pressure, and the number of encounters with elevated blood pressure (systolic/diastolic BP >= 140/90 mmHg). Longitudinal features were aggregated as minimum, maximum, median, standard deviation, and skewness.

The 34 most common ambulatory laboratory test results (*i.e.*, complete metabolic panel, complete blood count with differential, lipids, thyroid stimulating hormone, and hemoglobin A1c) with missingness in fewer than one-third of subjects were summarized as minimum, maximum, median, 1st quartile, and 3rd quartile. Diagnosis codes for hypertension, associated comorbidities, and other indications for anti-hypertensive medications were aggregated and summarized as median per year and sum. Medication prescriptions were summarized as both the number of days prescribed for each antihypertensive class and the count of encounters while prescribed 1, 2, 3, or 4 or more anti-hypertensive medications, described as sum, median, standard deviation, and skewness, as well as the sum of encounters with elevated blood pressure. Regular expressions were applied to clinical notes to identify mentions of ‘hypertension’ and variants thereof, summarized as counts.

Features with values outside of physiologically reasonable ranges, with fewer than 5% non-zero counts, or with variance < 0.05 were excluded. Missing values were median imputed.

In the original paper from which the data were extracted (La Cava et al., 2019), the diagnoses were manually ascertained through chart review. A study physician (a MD graduate with experience as a medical officer) reviewed clinical charts and classified subjects according to three phenotypes. Classification was based on the JNC7 Guidelines on Prevention, Detection, Evaluation, and Treatment of High Blood Pressure (Chobanian et al., 2003). The heuristics were proposed by expert clinicians and iteratively refined using the training data to assess how many cases were labeled as true positives. Unclear cases were reviewed by one additional study physician. The chart review was documented in a single Redcap form, with the reviewer's conclusion and the underlying evidence.

Table 5 summarizes all features present in the dataset. All features can be considered potential risk factors for different hypertension phenotypes. Some features represent descriptive statistics of lab tests, with a suffix indicating the corresponding statistic. When multiple statistics are available, they are separated by a forward slash in the feature name.

Appendix C. ML hyper-parameters

Table 6 presents the machine learning methods compared in the second set of experiments. The hyperparameters for each method were optimized using a gridsearch with a stratified 5-fold cross-validation. For each method, the table lists the possible values considered during the gridsearch process to identify the optimal configuration.

Appendix D. Boxplot visualization for the settings study

Fig. 6 reports the same data from the settings study (Figure 2), using boxplots instead of barplots. While the barplot provides a cleaner view of average performance, making it easier to compare two methods, the boxplot gives us distribution estimates.

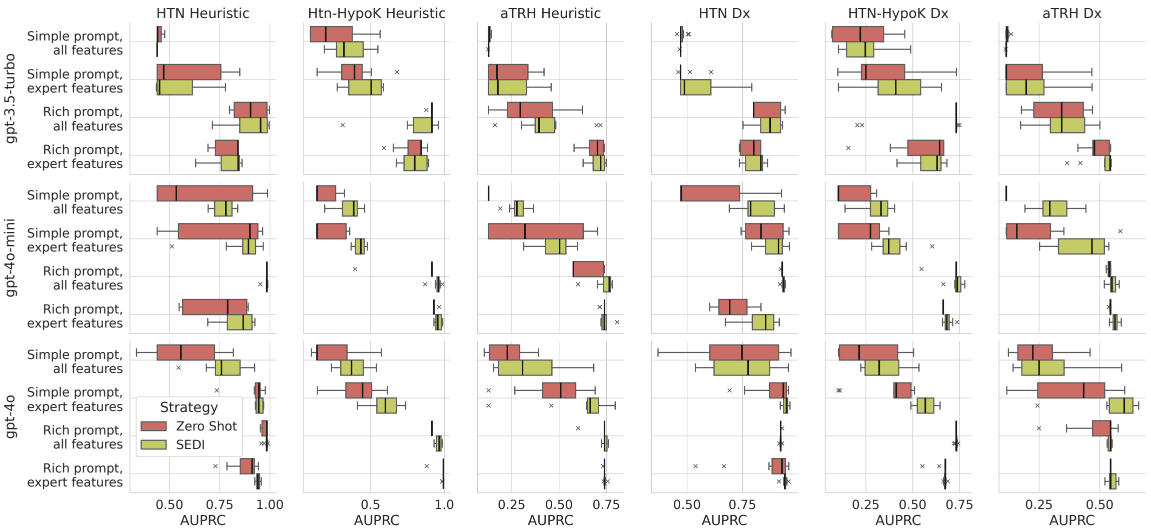

Figure 6: Average AUPRC on each held-out fold of different LLM-generated CPs, compared to other interpretable machine learning methods. LLMs results are shown using rich prompts and expert features, with and without SEDI training.

Appendix E. AUPRC comparison for aTRH

Fig. 7 reports a comparison of LLM-generated CPs held-out validation performance in 5-fold cross-validation, comparing the LLMs to conventional interpretable ML methods

Table 5: All features in the dataset. Features with multiple variations (indicated by words separated by a forward slash) mean that all options exist as features

Group	Feature	Feature Description
Demo	Age	Patient's age at right-censoring date
	Sex	Patient's documented Sex (1 = male, 0 = female)
	Race (Black)	1 = black, 0 = non-black
	Race (Other)	1 = Asian, other, mixed, Native American, Pacific Islander, 0 = black or white
	Race (White)	1 = white, 0 = non-white
	ZIP_CAT	Distance from patient's home to 19104, binned
Encounter	Practice Site	Code for healthcare site (not one-hot encoded), common service for all source systems.
	Practice type	1 = internal medicine practice, 0 = family medicine practice
BMI/Weight	weight_{statistic}	Min/max/median/sd/skewness of weights
	bmi_{statistic}	Min/max/median/sd/skewness of BMI
BP	bp_n	Total number of blood pressure (BP) measurements
	{statistic}_systolic/diastolic	Max/mean/median/sd/skew of systolic/diastolic BP measured
	high_bp_n	Number of high blood pressure cases (SBP >= 140 or DBP >= 90)
	{statistic}_high_bp_systolic/diastolic	Mean/median/sd/skew of systolic/diastolic BP of all high blood pressure measurements (SBP >=140 or DBP >= 90)
	{statistic}_high_bp_n_yr	Median/sd/skew of systolic/diastolic BP for high blood pressure measurements
Labs Dx	{statistic}.lab_XXX	Maximum/minimum/median/q1/q3 of XXX lab test
	{statistic}_ICD_XXX (Dx)	Median/sum XXX ICD-9 and ICD-10 codes, by year
	{statistic}_XXX (disease name)	Median/sum XXX disease name, by year
	Dx_N	Number of total ICD-9 and ICD-10 codes (PX_DX_ID)
	enc_N	Number of OUTPATIENT (including INFUSION VISIT) encounters
	dx_days_x	Days from 1st Dx to last Dx in system
Medication	HTN_MED_days_XXX	Days on med XXX (including anti-HTN and Potassium Supplement)
	MED_N	Number of medication prescriptions total
	high_bp_during_htn_meds_{meds}	Number of high BP measurements during 1/2/3/4+ anti-HTN meds
	{statistic}_enc_during_htn_meds_{meds}	Number of OUTPATIENT encounters during 1/2/3/4+ meds

Continued on next page

DT and LR L1, a representative non-interpretable ML method RF, and a representative symbolic regression method FEAT. We observe that the SEDI approach appeared associated higher performance for some LLM, phenotype combinations, particularly for `gpt-4o-mini`

Table 5: All features in the dataset. Features with multiple variations (indicated by words separated by a forward slash) mean that all options exist as features (Continued)

Group	Feature	Feature Description
Medication	N_med_k_chlo_enc	Number of encounters on POTASSIUM_CHLORIDE/POTASSIUM_GLUCONATE
	sd_med_k_chlo_enc	Standard deviation of number (by year) of encounters on POTASSIUM_CHLORIDE/POTASSIUM_GLUCONATE
	skewness_med_k_chlo_enc	Skewness of number (by year) of encounters on POTASSIUM_CHLORIDE/POTASSIUM_GLUCONATE
Heuristic Features	low_K_N	Number of low potassium test results
	test_K_N	Number of potassium test results
	Med_Potassium_N	Number of potassium supplement medication subscriptions
	Dx_HypoK_N	Number of Hypokalemia Dx
HTN Score Features	ICD_hyp_sum	HTN ICD codes
	Med_HTN_N	Anti-HTN meds prescriptions
	bp_hyp_norm	high_bp_n/bp_n
	icd_hyp_sum_norm	ICD_hyp_sum/Dx_N
	MED_HTN_N_norm	MED_HTN_N/MED_N
	re_hyp_spec_norm	re_hyp_spec/words_n
Regex	re_htn_{statistic}	Sum/max/mean/median/sd/skewness of regex counts in clinical notes for hypertension
	re_htn_spec_{statistic}	Sum/max/mean/median/sd/skewness of regex counts in clinical notes for hypertension (specific, excluding preliminary negations)
	re_htn_teixeira_{statistic}	Sum/max/mean/median/sd/skewness of regex counts in clinical notes for hypertension (regex used in Teixeira paper)
	re_word_count_{statistic}	Sum word counts in clinical notes

for the more complicated aTRH phenotypes. The average performance of CPs generated by gpt-4o for HTN-HypoK heuristic outperformed FEAT, however in all other cases the LLM-generated CPs showed slightly worse average performance. In despite of that, particular runs generated good performing CPs.

Appendix F. Using optimizers to fine-tune the final model

We performed supervised parameter optimization on the final gpt-4o+SEDI model using a black-box optimizer nevergrad (Rapin and Teytaud, 2018). The optimizer aimed to maximize AUPRC by exploring different combinations of hyperparameters in the training data. The approach was implemented by manually adjusting the python function to include all numeric parameters as arguments and then using an gradient-free, adaptative optimization algorithm, to iteratively evaluate different combinations of values. The process can be applied to our generated methods as it does not rely on prior knowledge of the model's internal structure, but we also notice that it has limitations depending on the initial starting point.

In this proof-of-concept, we demonstrate that LLM-constructed models can be considerably refined by leveraging available outcome data, searching for sub-optimal parameter configurations that further improves the performance of our generated method. The down-

Table 6: Machine learning methods compared in the second set of experiments, with hyperparameters optimized via gridsearch in a stratified 5-fold cross-validation. Each list represents the possible values set to optimize using gridsearch.

Method	Parameter

Decision Tree (DT)	max_features = ['auto', 'sqrt']
	max_depth = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
	min_samples_split = [2, 5, 10]
	min_samples_leaf = [1, 2, 4]

Logistic Regression
with L1 norm (LR L1)
	Cs = [1.e-06, 1.e-05, 1.e-04, 1.e-03,
	1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03]
	penalty=['l1']
	solver = ['liblinear']

Random Forests (RF)	n_estimators = [100, 500, 900, 1300, 1700, 2100]
	max_features = ['auto', 'sqrt']
	max_depth = [10, 30, 50, 70, 90, 110]

Feature Engineering
Automation Tool (FEAT)
	max_depth=[6]
	max_dim = [10]
	objectives= [["fitness","size"]]
	sel=['lexicase']
	gens = [200]
	pop_size = [1000]
	stagewise_xo = [True]
	scorer=['log']
	ml=['LR']
	fb=[0.5]
	classification=[True]
	functions= ["split","and","or","not","b2f"]
	split=[0.8]
	normalize=[False]
	corr_delete_mutate=[True],
	simplify=[0.005]

side of this approach is that it requires a lot of outcome data in order to perform a parameter optimization of the generated program in a supervised learning fashion.

Appendix G. Dialog for generating the final aTRH model

Figure 8 displays the first three generated programs before the final aTRH model was obtained by `gpt-4o+SEDI` strategy. Unlike the reported model in Section 6, the outputs here have not been processed for clarity. We show the first three programs just for illustrative purposes.

The `Initial model` was evaluated and achieved an AUPRC of 0.54, AUROC of 0.91, a FP rate of 8.0% and FN rate of 9.8%. The feedback prompt provided 10 FP and FN examples.

The LLM then presented the `Model after first iteration of SEDI`, which included more features and if-else statements, but kept the core instructions from the previous model. The metrics changed to 0.56 AUPRC, 0.91 AUROC, 1.1% FP, and 84.1% FN. We can see that the LLM performed some adjustments, improving some metrics but presenting an

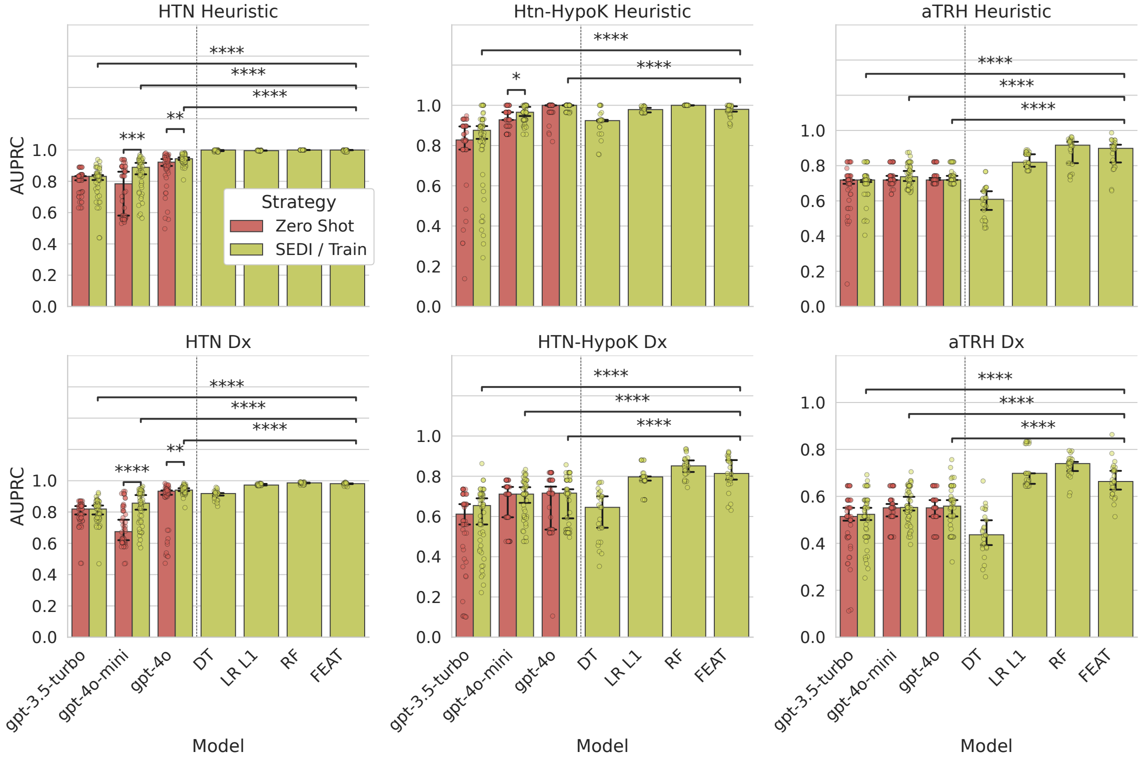

Figure 7: Average AUPRC cross-validation of different LLM-generated CPs, compared to other interpretable machine learning methods from La Cava et al. (2023). LLMs results are shown using rich prompts and expert features, with and without SEDI training. Error bars depict the 95% confidence interval across different seeds. In addition to the bars, the performance of each run is depicted as a dot overlaying the bars. Statistical comparisons are conducted using Mann-Whitney-Wilcoxon two-sided test with Holm-Bonferroni corrections. *: 1.00e-02 $< p \leq$ 5.00e-02; **: 1.00e-03 $< p \leq$ 1.00e-02 ***: 1.00e-04 $< p \leq$ 1.00e-03 ****: $p \leq$ 1.00e-04. Appendix D has a boxplot visualization of the same data.

extremely high FN rate. The prompt states to create a new model that achieves fewer false positive and fewer false negatives.

The next output updated the model with a new set of conditionals, reported as `Model after second iteration of SEDI`, while still keeping the core of the previous programs. It was evaluated to 0.60 AUPRC, 0.93 AUROC, 4.2% FP, and 40.2% FN.

Appendix H. Performance across different subgroups

We assess potential biases in the final model generated by `gpt-4o+SEDI` by evaluating its performance across different subgroups within the test dataset. We considered bias relative to subject's documented gender and race. These variables were selected because they are commonly studied in fairness assessments in machine learning models.

To evaluate bias, we report how the final model performs on each subpopulation defined by gender and race, reporting classification metrics for each subpopulation and comparing them to the overall performance of the model. This allows us to determine if the model exhibits disproportionate error rates or other disparities across different demographic groups. Table 7 reports the metrics for the intersecting subpopulations when using the final model generated by the `gpt-4o+SEDI` strategy, using the rich prompt and expert features set, evaluated for aTRH diagnosis. Table 8 reports case counts and sample counts.

Table 7: Average AUPRC and AUROC on the held-out test partition on each subpopulation, with race and gender as the distinguishing variables, evaluated for aTRH diagnosis.

Race	Gender	AUPRC	AUROC
Black	Female	0.822	0.929
	Male	0.862	0.917
White	Female	0.824	0.966
	Male	0.903	0.981
Other	Female	0.456	0.965
	Male	0.329	0.823

We see differences in AUPRC and AUROC across race and gender. The model's discriminative performance appears highest for White men and lowest for Black women and white women. Other ethnic groups shows a low AUPRC, and prevalence is also low for these subgroups.

Table 8: Case counts and sample counts for each subpopulation, with race and gender as the distinguishing variables, evaluated for aTRH diagnosis.

Race	Gender	# Cases	# Samples	Prevalence
Black	Female	26	70	0.37
	Male	13	37	0.35
White	Female	17	97	0.17
	Male	15	69	0.21
Other	Female	1	16	0.06
	Male	1	11	0.09

There are several explanations for this apparent bias. LLMs are known to recapitulate biases from their training corpuses. However, for this application a likely explanation is bias in training and testing data. While the final model does not directly incorporate race or gender, there are associations between these factors and incorporated features. As shown in Tables 7 and 8, Black patients are more likely to be documented as having apparent treatment-resistent hypertension. Differences in AUPRC measures are explained in part by the sensitivity of this metric to differences in case prevalence. Some of the differences in AUROC are also due to have single case counts/few patients within subgroups (*e.g.*, the 'Other' category). In addition, there are many differences in how women and men interact with the healthcare system, which were not considered in the upstream feature engineering. These findings warrant follow-up to compare for bias in the ML methods and to mitigate bias in the LLM-based approach. Approaches for mitigating bias could include accounting for subject factors when samping for FP and FN subjects in the SEDI approach.

In addition, prompts could use classification metrics evaluated across subpopulations and ask the LLM to explicitly consider mitigating bias.

Iterative program refinement using the SEDI strategy

```python
# Initial model
def predict_hypertension(df: pd.DataFrame) -> np.ndarray:
    probabilities = []

    for _, row in df.iterrows():
        if row['high_BP_during_htn_meds_3'] >= 2 or row['
            ↪sum_enc_during_htn_meds_4_plus'] >= 2:
            probabilities.append(1.0)
        else:
            probabilities.append(0.0)

    return np.array(probabilities)

# Model after first iteration of SEDI
def predict_hypertension(df: pd.DataFrame) -> np.ndarray:
    probabilities = []

    for _, row in df.iterrows():
        prob = 0.0
        if row['high_BP_during_htn_meds_3'] >= 2:
            prob += 0.5
        if row['sum_enc_during_htn_meds_4_plus'] >= 2:
            prob += 0.5
        if row['high_BP_during_htn_meds_3'] >= 2 and row['
            ↪sum_enc_during_htn_meds_4_plus'] >= 2:
            prob = 1.0
        if row['mean_diastolic'] < 75 or row['Med_Potassium_N'] == 0:
            prob *= 0.5
        probabilities.append(prob)

    return np.array(probabilities)

# Model after second iteration of SEDI
def predict_hypertension(df: pd.DataFrame) -> np.ndarray:
    probabilities = []

    for _, row in df.iterrows():
        prob = 0.0
        if row['high_BP_during_htn_meds_3'] >= 2:
            prob += 0.4
        if row['sum_enc_during_htn_meds_4_plus'] >= 2:
            prob += 0.4
        if row['mean_diastolic'] > 80:
            prob += 0.1
        if row['mean_systolic'] > 140:
            prob += 0.1
        if row['high_BP_during_htn_meds_2'] > 5:
            prob += 0.1
        prob = min(1.0, prob)
        if row['Med_Potassium_N'] > 0 and row['Dx_HypoK_N'] > 0:
            prob *= 0.5
        probabilities.append(prob)

    return np.array(probabilities)
```

Figure 8: Initial programs generated by GPT-4o with the SEDI strategy.