Multimorbidity-Anchored ICU Phenotypes (MAIP)

Hari S. Sreedeth

2025-12-23

Introduction

Critically ill patients admitted to intensive care units (ICUs) exhibit substantial heterogeneity in both underlying chronic disease burden and acute physiological derangement. This heterogeneity complicates prognostication and the identification of clinically meaningful subgroups. Although unsupervised learning has been widely used to derive data-driven ICU phenotypes, many approaches are dominated by the strong signal of multimorbidity and age. As a result, clusters can collapse into broad severity or comorbidity gradients rather than isolating distinct *acute* clinical presentations within comparable background risk.

The Multimorbidity-Anchored ICU Phenotypes (MAIP) project[1] addresses this limitation by explicitly separating chronic disease burden from acute physiology when deriving phenotypes in the SUPPORT-II cohort of approximately 9,000 adult ICU admissions[2]. The pipeline implements two complementary unsupervised pathways: (i) multimorbidity-stratified phenotyping (MMSP) and (ii) a simplified multi-view similarity network fusion approach (SNF-lite), designed to identify phenotype structure that is not merely a reflection of comorbidity burden. Cluster solutions are assessed for stability and separability, and downstream outcome information is reserved for validation rather than used to construct the phenotypes.

A central design goal of MAIP is auditability: translating high-dimensional clustering output into transparent, reviewable definitions suitable for clinical communication. To do this, a sparse decision-tree surrogate model is trained to approximate final cluster assignments, and its decision paths are exported as canonical JSON rules. These machine-readable rules are then converted into clinician-facing rulecards and ASCII flowcharts using a retrieval-augmented generation (RAG) pattern grounded in a frozen corpus of variable definitions, phenotype summaries, and style guidance. Rule translation is performed with a remote general-purpose chat model (GPT-4.1-mini, accessed via API) under deterministic constraints that prohibit new logic, and is coupled with programmatic fidelity checks to ensure that the textual rulecards preserve the original rule content. The resulting output is an end-to-end reproducible workflow from raw ICU data to stable phenotypes and an auditable, human-readable decision artifact.

The primary objective of the MAIP project is therefore twofold:

[1] See Github repositories Project-MAIP

[2] The support dataset is a random sample of 1000 patients from Phases I & II of SUPPORT, Study to Understand Prognoses Preferences Outcomes and Risks of Treatment

1. To derive stable, clinically interpretable ICU phenotypes that are not merely reflections of comorbidity gradients, and
2. To demonstrate a robust, auditable pathway from unsupervised clustering to a clinician-facing rule representation, implemented via a RAG-assisted small-model translation pipeline.

Methods

Study design and data source

This is a retrospective cohort study using the SUPPORTII dataset, which contains detailed information on approximately 9,000 critically ill adult patients admitted to ICUs in multiple North American centres during the late 1980s and early 1990s. SUPPORT-II includes baseline demographics, comorbidities, physiological measurements near ICU admission, prognostic scores and follow-up outcomes such as vital status, survival time and length of stay. The target population for the phenotyping analyses comprises adult ICU admissions with sufficient data in the comorbidity and acute physiology domains to support cluster assignment. Outcome variables are reserved for validation analyses and are not used to build the phenotypes themselves.

Feature engineering and view construction

The analysis distinguishes between a feature matrix (X), which is used for unsupervised learning and surrogate modelling, and an outcome matrix (Y), which is held out for external validation. The outcome matrix includes 2 vital status at one or more pre-specified horizons (for example in-hospital and 5-year mortality), time-to-event variables, and measures of ICU and hospital length of stay and resource utilisation. No outcome variable is included among the clustering features.

The feature matrix is organised into three conceptual "views". The comorbidity view contains variables that describe chronic disease burden: the SUPPORT-II comorbidity count, indicator variables for specific chronic conditions, and broader diagnostic categories where available. The physiology view contains acute-state variables measured around ICU admission, including vital signs, arterial blood gas values, laboratory measurements, and composite severity indices. The socio-contextual view includes demographic characteristics such as age and sex, socio-economic proxies where available, and care-context variables such as advance directives. Continuous variables in the physiology view are transformed if necessary to mitigate extreme outliers and are standardised to have mean zero and unit variance. Categorical

variables are encoded in a way that preserves interpretability (for example, one-hot encodings with clear level labels).

Missing data in (X) are handled using a consistent, pre-specified strategy. A pragmatic approach is to apply a single imputation model that is fitted without reference to the outcomes, optionally coupled with missingness indicators for variables with substantial gaps. The imputation recipe is documented in sufficient detail (including model type, predictors, and random seed) to support reproduction. The key design principle is that the imputation must not "peek" at outcomes and must preserve the link between the imputed features and the original clinical variables.

Phenotyping strategy: ***multimorbidity-stratified (MMSP) and multi-view fusion (SNF-lite)***

The phenotyping stage proceeds along two conceptual pathways, which are later compared and reconciled.

In the multimorbidity-stratified pathway, patients are first divided into strata defined by chronic disease burden, as measured by the comorbidity count. For example, one stratum may contain patients with 0–1 chronic conditions, a second those with 2–3, and a third those with four or more, although the exact cut-points are decided a priori based on the distribution of comorbidity and clinical plausibility. Within each stratum, the clustering algorithm then focuses chiefly on acute physiology and selected contextual variables, effectively asking: among patients with broadly similar multimorbidity, what patterns in acute state and context differentiate subgroups?

Because the data contain both continuous and categorical variables, dimensionality reduction within each stratum is performed using Factor Analysis of Mixed Data (FAMD). FAMD is a principal-component-type method designed for tables that contain both quantitative and qualitative variables and can be understood as an extension of principal component analysis (PCA) that incorporates multiple correspondence analysis (MCA) for categorical variables. (Wikipedia) A small number of FAMD components capturing a substantial proportion of the variance are retained. Distances between patients in this reduced space are then computed using a metric suitable for mixed data, such as Gower distance, which accounts for both numerical and categorical dimensions. Clustering is performed with a partitioning algorithm such as k-medoids, which is less sensitive to outliers than k-means and is naturally defined on arbitrary dissimilarity matrices.

A range of candidate values for the number of clusters (K) is explored within each multimorbidity stratum. For each candidate (K), the stability of the clustering solution is evaluated using bootstrap

resampling and the Adjusted Rand Index to quantify concordance between re-fitted partitions. Internal cluster quality indices such as the average silhouette width, the Calinski–Harabasz index and the Davies–Bouldin index are also calculated. The final choice of (K) balances statistical stability and separation with the requirement that the resulting clusters be clinically interpretable.

The multi-view fusion pathway starts from the three views of the feature matrix described above. For each view, a similarity matrix between patients is constructed. For the comorbidity and socio-contextual views, Gower similarities are appropriate, as they can accommodate both binary and categorical variables. For the acute physiology view, where the variables are scaled and continuous, a radial basis function (RBF) kernel is used so that pairs of patients that are close in physiological space have high similarity, and those that are far apart have similarity close to zero. The result is three view-specific similarity matrices, each reflecting a different aspect of patient resemblance.

These matrices are then combined using a simplified implementation of Similarity Network Fusion (SNF). SNF iteratively diffuses information across networks so that the fused network emphasises similarities that are supported by multiple data types. (PubMed) In practice, each similarity matrix is transformed into an affinity matrix, and a diffusion process is carried out in which each view's affinity is updated using a weighted average of its own structure and the structures of the other views. After a fixed number of iterations, the fused affinity matrix represents a consensus notion of similarity that integrates comorbidities, physiology and context. Spectral clustering is then applied to the fused matrix: the graph Laplacian is constructed, its leading eigenvectors are used to embed patients into a low-dimensional space, and a clustering algorithm is run in that embedded space. The choice of (K) is informed by the eigengap heuristic together with the same stability and internal validity metrics used in the stratified pathway.

The two pathways yield alternative sets of phenotypes. Their performance is compared both quantitatively, via stability and internal validity indices, and qualitatively, through clinical inspection of cluster profiles. The final phenotyping solution is selected from among these candidates, with a preference for solutions that are stable, parsimonious, and offer interpretable distinctions that are not reducible to simple gradients in age or multimorbidity.

External validation and phenotype characterisation

Once a final clustering solution has been chosen, it is examined in relation to the held-out outcomes. Survival analyses compare phenotypes using Kaplan–Meier curves and log-rank tests, followed by Cox proportional hazards models in which cluster membership is entered as a categorical covariate, adjusted for key baseline confounders. Similar models or regression frameworks appropriate to the outcome distribution are used to compare ICU and hospital length of stay and cost across clusters. These analyses assess whether the phenotypes convey prognostic information that is not fully captured by standard severity scores or comorbidity counts. At the same time, descriptive profiles of each phenotype are assembled that summarise their comorbidity patterns, physiological characteristics and socio-demographic composition.

Surrogate decision-tree model and rule extraction

To render the phenotyping solution in a form that is directly usable at the bedside or in downstream studies, a sparse surrogate model is trained to approximate the mapping from the feature matrix (X) to the final cluster labels. The surrogate is a decision tree fitted with constraints on depth and minimum node size to avoid over-fitting and to keep the tree compact and interpretable. The input features can be restricted to a subset of clinically intuitive variables (for example, a curated set of comorbidities and physiological measures) to simplify the resulting rules.

The fitted tree is evaluated by comparing its predicted cluster labels against the original phenotyping assignments, using accuracy and macro-averaged F1 scores as measures of fidelity. A confusion matrix highlights any phenotypes that are intrinsically difficult to separate with a shallow tree. A target fidelity threshold is set a priori, recognising that the surrogate purposefully trades some fine-grained accuracy for interpretability.

Once an acceptable tree is obtained, its structure is exported into a structured JSON representation. Each path from the root to a leaf corresponds to a decision rule that assigns patients to a specific phenotype based on a conjunction of conditions on individual variables. The JSON representation records, for each rule, the feature names, comparison operators, threshold values, and the resulting phenotype label. A schematic example is:

```
{
  "ruleset_id": "maip_v1",
  "phenotypes": ["P1", "P2", "P3", "P4"],
  "rules": [
```

```
    {
      "id": "R1",
      "if": [
        {"feature": "num_co", "op": "<", "value": 2},
        {"feature": "aps", "op": ">", "value": 18}
      ],
      "then": {"phenotype": "P2"}
    }
  ]
}
```


This structured representation becomes the canonical source of truth for the rule logic used in the subsequent LLM translation phase.

Retrieval-augmented LLM translation of rules

The final stage translates the JSON-encoded surrogate tree rules into clinician-facing "rulecards" and ASCII flowcharts using a small, retrieval-augmented prompting pipeline. Conceptually, this stage treats the decision-tree rules as ground truth and uses a large language model (LLM) purely as a controlled re-writer that produces human-readable text without altering the underlying logic.

The retrieval component is deliberately simple and fully local. A small corpus is maintained in the analysis repository: (i) a variable dictionary (`variable_dictionary.json`), (ii) per-phenotype markdown summaries (`phenotype_<label>.md`), and (iii) a markdown style guide (`style_guide.md`). The variable dictionary associates each model feature with a canonical code, a display name, measurement units, optional notes, and optional additional metadata (e.g. ordinal scales or value maps). The phenotype summaries are short, manually written descriptions of each multimorbidity phenotype, derived analytically from the clustering solution (e.g. dominant comorbidities, typical physiological profile, outcome gradients) rather than generated by the LLM. The style guide encodes high-level constraints on phrasing, such as requiring explicit units for numeric thresholds, favouring simple "if/then" constructions, and avoiding qualitative labels such as "severe" unless anchored in an explicit cut-point.

For each surrogate stratum (High_MM, Mid_MM, Low_MM), the `rag_translate_rules.py` script reads the corresponding `rule_ruleset.json` file produced by the surrogate tree training step. Rules are grouped by their outcome label, and, for each label, the script: (i) collects all rules whose `outcome` equals that label; (ii) identifies the set of features used in those rules by inspecting the `path` conditions; and (iii) extracts only the relevant entries from the variable dictionary. If a markdown summary file `phenotype_<label>.md` is present, it is loaded; otherwise,

an explicit placeholder text is inserted indicating that no summary is available. These ingredients (style guide, phenotype summary, and phenotype-specific subset of the variable dictionary) are then concatenated with the raw JSON rules into a single markdown user prompt. A short system prompt describes the LLM's role as a deterministic translator and emphasises that every condition (feature, operator, threshold) from the JSON must appear in the textual output in an equivalent form, and that no new criteria or clinical recommendations may be introduced. The script writes one prompt JSON file per phenotype label (e.g. `prompt_High_MM_0.json`), containing the stratum, label, system prompt, user prompt, the original rules, and the list of variables used.

Rulecards are generated by a separate script (`run_rulecards_remote_canonical.py`) that consumes these prompt files. For each phenotype, the script constructs a canonical system prompt that enforces a fixed output structure and calls a chat-completion model via API (default: `gpt-4.1-mini`, OpenAI). The model is run at near-zero temperature and with a fixed token budget to promote deterministic, reproducible outputs. The user message contains all retrieved context (style guide text, phenotype summary, variable snippets, and the JSON rules), and the system message specifies a strict format: the first line must be of the form `Phenotype {label} - <short clinical title>`, followed by three sections headed exactly `Key idea:`, `Rulecard:`, and `ASCII flowchart:`. The `Key idea` section provides a brief narrative description of the phenotype using only information from the supplied phenotype summary and variable dictionary. The `Rulecard` section lists each decision rule as a bullet-point "IF … THEN phenotype {label}" statement, preserving the surrogate tree logic. The `ASCII flowchart` section renders the same branching structure as a simple text tree for quick visual inspection. Illustrative schematic output is shown below; in practice, the headings and layout are enforced by the system prompt, and the content is constrained to the information passed in the user message:

```
Phenotype P2 - "Hemodynamically unstable with limited chronic burden"

Key idea:
Patients with relatively few chronic conditions but marked acute physiologic derangement as reflected b

Rulecard:
- IF comorbidity_count < 2 AND aps > 18 THEN classify as Phenotype P2.
- IF comorbidity_count < 2 AND aps  18 AND pao2_fio2_ratio < 200 THEN classify as Phenotype P2.
- …
```

```
ASCII flowchart:
[Start]
  |
  |-- Is comorbidity_count < 2? -- No --> [Other phenotypes]
  | Yes
  |
  |-- Is aps > 18? -- Yes --> [P2]
  | No
  |
  |-- Is pao2_fio2_ratio < 200? -- Yes --> [P2]
                           No --> [Other phenotypes]
```

The LLM output is not accepted blindly. A dedicated validation script (`rulecard_validate.py`) applies three families of checks. First, a rule-coverage check parses each markdown rulecard, extracts lines that begin with "- IF", and verifies that every JSON condition in the surrogate ruleset has a textual counterpart. For each condition, the script requires that the feature name appear as a whole word in at least one "IF" line and that one of several plausible string renderings of the numeric threshold (e.g. raw value and rounded formats) co-occurs in the same line. Any missing feature or threshold at this string-matching level is flagged in a per-phenotype summary table. Second, a variable-dictionary alignment check confirms that all features used in the surrogate rules are mentioned by their canonical name in the rulecard text and, where possible, appear in parentheses or code formatting (e.g. "… (aps)" or "`aps`"), indicating that the textual rulecard remains tied to the underlying feature definitions. Third, an optional synthetic-profile check (enabled when the fitted surrogate tree model is available) evaluates the internal consistency of the JSON rules with respect to the tree they were distilled from: for each path in the tree, a synthetic feature vector is generated that lies slightly inside all of the path's inequalities, the surrogate model predicts a phenotype label for this vector, and the predicted label is compared to the intended outcome of the rule. These checks produce CSV reports summarising rule coverage, naming alignment, and (optionally) agreement between the JSON rules and the surrogate model.

All artefacts in this stage—including the JSON rulesets, the assembled prompts, and the generated markdown rulecards—are stored as plain-text files under a fixed directory structure. Given a particular version of the code and the surrogate tree outputs, the rulecard generation process can therefore be rerun end-to-end, and any individual rulecard can be reconstructed from the saved prompts and ruleset if manual audit is required.