

# Validating an open source electronic health record pipeline to phenotype pneumonia

*ehrapy: an open source Python framework for EHR data*

Shauna Heron 

*sheron@laurentian.ca*

*Laurentian University*

April 7, 2025

# EHR data contains a wealth of information

*However working with it is extremely challenging*

- EHRs = rich but messy (hierarchical, sparse, missing)
- Need better tooling for data processing, joining, organization, cohort tracking, analysis + phenotyping
- ehrapy: Python framework designed for EHR analysis [1]

# Phenotyping pneumonia unspecified

*Learn to process a large EHR dataset for clustering and modelling*

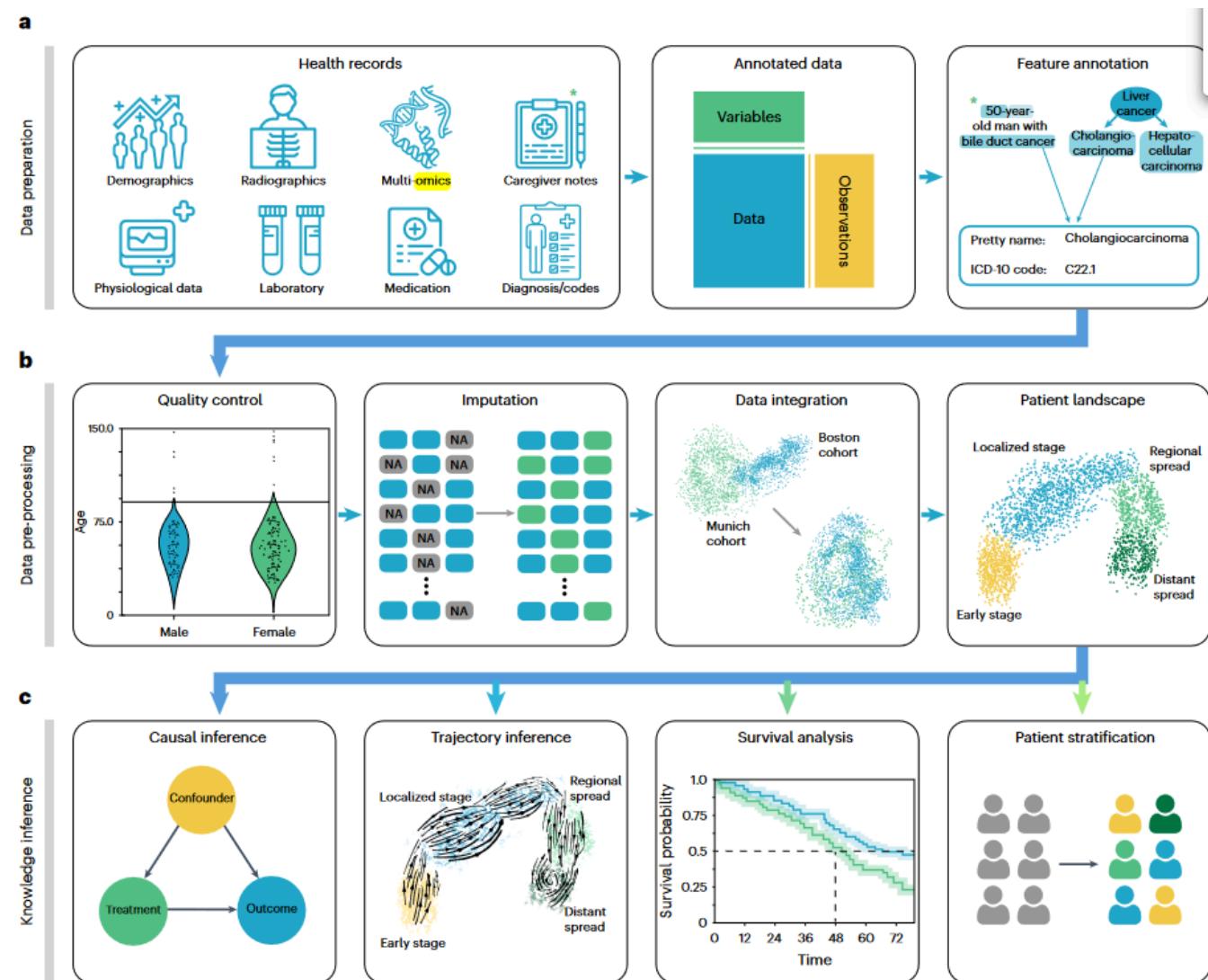
1.  Validate ehrapy with MIMIC-IV
2.  Replicate Gupta and colleagues original paper that phenotype pediatric pneumonia [2]
  - i.  Stratify pneumonia patients into subtypes (unsupervised)
  - ii.  Extend with supervised ML to predict hospital mortality



## Tip

The process of identifying and grouping patients based on shared clinical characteristics, patterns, or outcomes – often using unsupervised learning to uncover hidden subtypes within a population [3].

# ehrapy Workflow



ehrapy EHR data pipeline. Retrieved from: [2]

# Dataset

# MIMIC-IV

## *Adult pneumonia unspecified cohort*

- ~70,000 ICU and non-ICU adult patient admissions (2008–2019)
- Structured: vitals, labs, meds, diagnoses, procedures
- Credentialed access to MIMIC-IV can be obtained via PhysioNet [4]

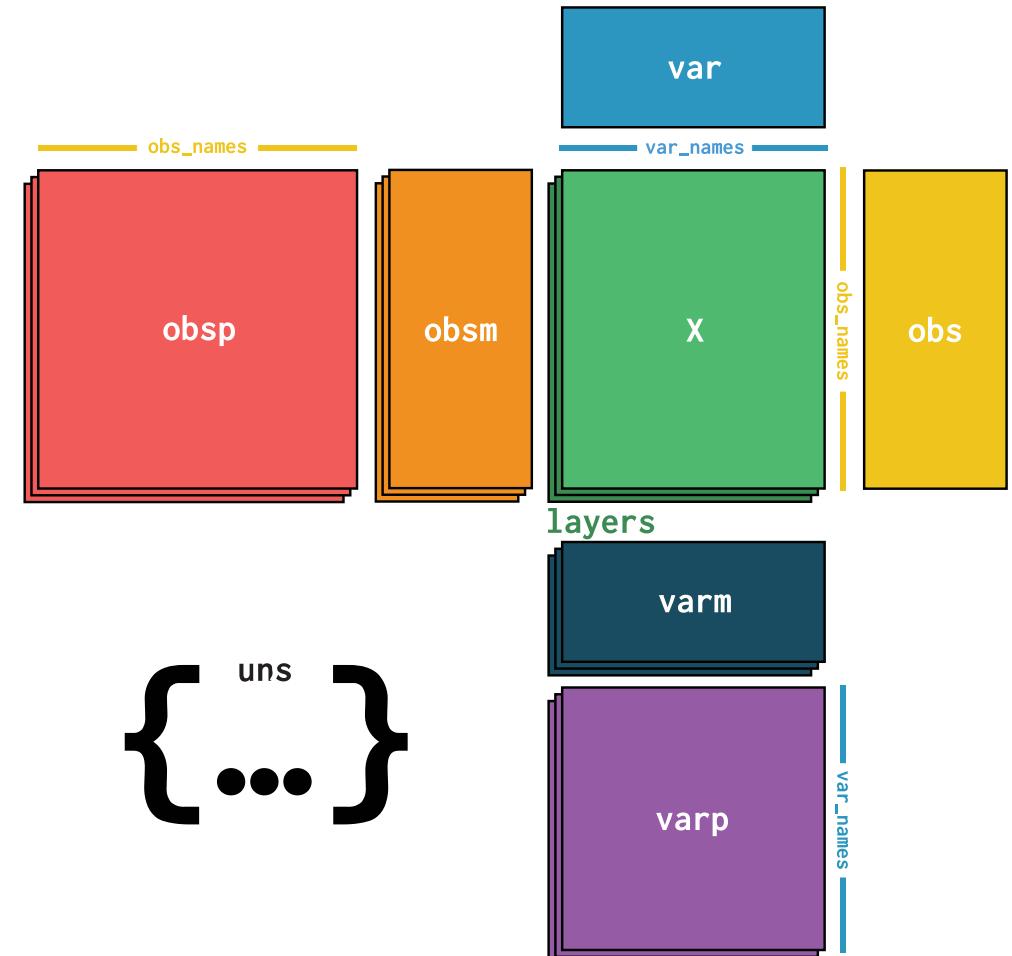
<b>Characteristic</b>	<b>Value</b>
Average Age	63.04
Average Hospital Mortality	7.14%
Average ICU Stay (days)	3.63
Average ICU Visits per Admission	1.11
Patients with >1 ICU Visit (%)	9.03%
Male (%)	55.98%
Female (%)	44.02%

# The plan

# Phase 1 - Framework Validation

## *ehrapy data processing pipeline*

- Wrangle data into vectors for converting to anndata object [5]
- Use ehrapy's built-in clustering pipeline to prepare and preprocess data
- Identify pneumonia subgroups (phenotypes)
- Compare with original paper (PIC) subgroups [2]
- Use new subgroups to predict 30 day in hospital mortality



Anndata Structure. Retrieved from [5]

# Reality...

# Trials and tribulations

```
File Edit Selection View Go Run Terminal Help
New Open ... 03-Ehrapy-Prep.ipynb - MIMIC-FINAL - Positron
EXPLORER paper.ipynb preAnndataProcessing.ipynb 03-Ehrapy-Prep.ipynb ML_Analysis ...
MIMIC-FINAL
mimiciv\2.0
icu
model
  _pycache_
.ipynb_checkpoints
behrt_model.py
behrt_train.py
calibrate_output.py
data_generation_icu.py
data_generation.py
di_train.py
evaluation.py
fairness.py
goofin.py
mimic_model.py
ml_models.py
model_utils.py
parameters.py
readme.md
tokenization.py
notebooks
  clean
  cohort_dfs
  ehrapy_data
  for_descriptives
  model_ready
  tables
  icd_chart_filtered.csv
  paper.ipynb
notes
  preprocessing_overview
preprocessing
  day_intervals_preproc
  hosp_module_preproc
readme.md
quarto_docs
OUTLINE
TIMELINE
Code + Markdown | Run All | Clear All Outputs | Restart Kernel ...
SESSION HELP VIEWER CONNECT ...
03-Ehrapy-Prep.ipynb
VARIABLES
cols_to_add
cols_to_convert
demographics_cat
demographics_fea
diagnoses_feature
end
feature
has_features
join_keys
lab_measurements
measurement_gro
medications_featu
metrics
microbiology_feat
proc_dict
procedures_featur
start
vars_to_normalize
CLASSES
TableOne
PLOTS
CONSOLE
Python 3.12.9 (Venv: venv) ~\Documents\MIMIC-FINAL
IPython 9.0.2 -- An enhanced Interactive Python. Type '?' for help.
Tip: The `%timeit` magic has an '-o' flag, which returns the results, making it easy to See `%timeit?`.
```

2 demographics\_features = ['Age', 'gender', 'ethnicity', 'insurance']  
3 demographics\_cat\_features = ['gender', 'ethnicity', 'insurance']  
4  
5 # Diagnoses – explicit list  
6 diagnoses\_features = ['ICD\_E87', 'ICD\_I50', 'ICD\_N17', 'ICD\_J96']  
7 has\_features = adata.var\_names[adata.var\_names.str.startswith("has\_")].tolist()  
8  
9 # Combine them  
10 diagnoses\_features += has\_features  
11  
12 # Lab measurements – range between two variable names  
13 start = adata.var\_names.get\_loc("ALT\_max")  
14 end = adata.var\_names.get\_loc("WBC\_min")  
15 lab\_measurements\_features = adata.var\_names[start:end + 1].tolist()  
16  
17 # Medications – same deal  
18 start = adata.var\_names.get\_loc("Acetaminophen-IV")  
19 end = adata.var\_names.get\_loc("Vecuronium")  
20 medications\_features = adata.var\_names[start:end + 1].tolist()  
21  
22 # Procedures – startswith filter  
23 procedures\_features = adata.var\_names[adata.var\_names.str.startswith("proc\_")].tolist()  
24  
25 # Microbiology – same as labs/meds  
26 start = adata.var\_names.get\_loc("PROBABLE ENTEROCOCCUS\_positive")  
27 end = adata.var\_names.get\_loc("ACINETOBACTER BAUMANNII COMPLEX\_positive")  
28 microbiology\_features = adata.var\_names[start:end + 1].tolist()

# Phase 1 - Technical Challenges

*Did I bite off more than I can chew?*



## Caution

Wrangling 9.8 GB worth of datasets and > 30 individual datasets into a cohesive, single cohort fit for analysis is HARD on your AND on your RAM

- What to include or not include in terms of chart data, medication data, icu versus hospital data, etc.
- Iterative pipeline testing to get it right
- Lots of late nights & hair pulling

# Phase 1 - Data Wrangling

*ICU data - All features with > 60% coverage*

```
import ehrapy as ep

① adata = ep.io.read_csv(pic_data_path)          # Read in the dataset into an AnnData object
② ep.pp.winsorize(adata)                         # Winsorize extreme values
③ ep.pp.summarize_measurements(adata)            # Summarize measurements to obtain min, average and max values
④ ep.pp.knn_impute(adata)                        # Impute missing values with k-nearest-neighbors imputation
⑤ ep.pp.log_norm(adata)                          # Log normalize numerical values
⑥ ep.pp.encode(adata, encodings="one-hot")        # One-hot encode categorical values
    ep.pp.pca(adata)                            # Reduce the dimensionality of the data using PCA
    ep.pp.neighbors(adata)                      # Compute the neighborhood graph
⑦ ep.tl.umap(adata)                            # Embed the graph in two dimensions using UMAP
    ep.pl.umap(adata, color="LOS")              # Plot the UMAP colored by the length of stay
```

ehrappy basic data pipeline functions. Retrieved from [2]

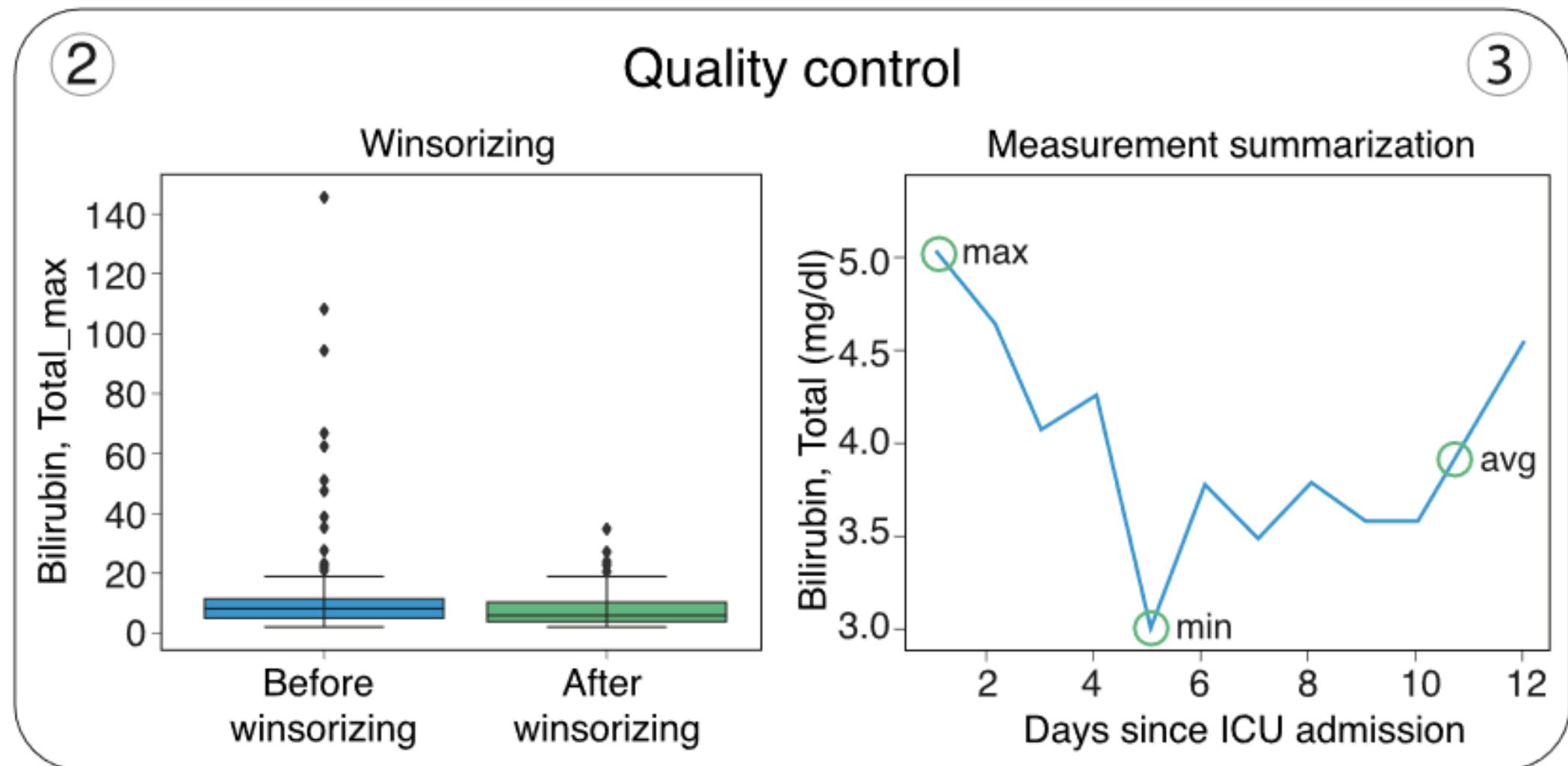
# The unspecified pneumonia cohort

ICD-J18

Characteristic	Value
Unique Patients	7518
Average Age	65.56
Average Hospital Mortality	13.69%
Average ICU Stay (days)	4.98
Average ICU Visits per Admission	1.20
Patients with >1 ICU Visit (%)	16.23%
Male (%)	55.00%
Female (%)	45.00%
<i>TOTAL OBSERVATIONS</i>	9864
<i>TOTAL FEATURES</i>	180

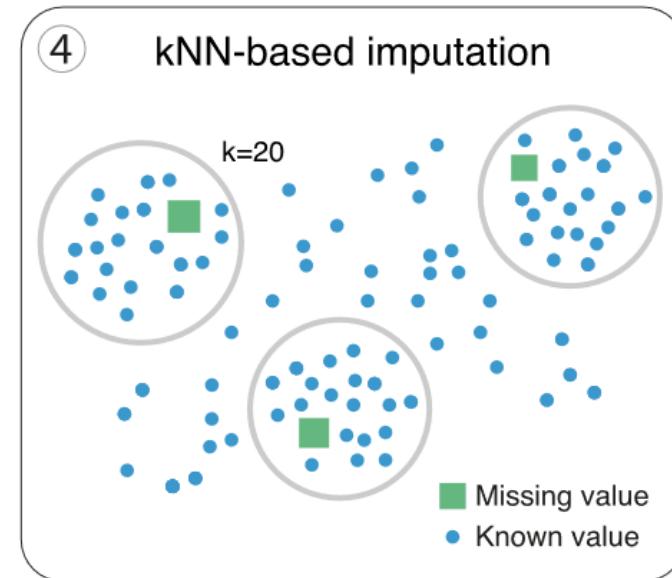
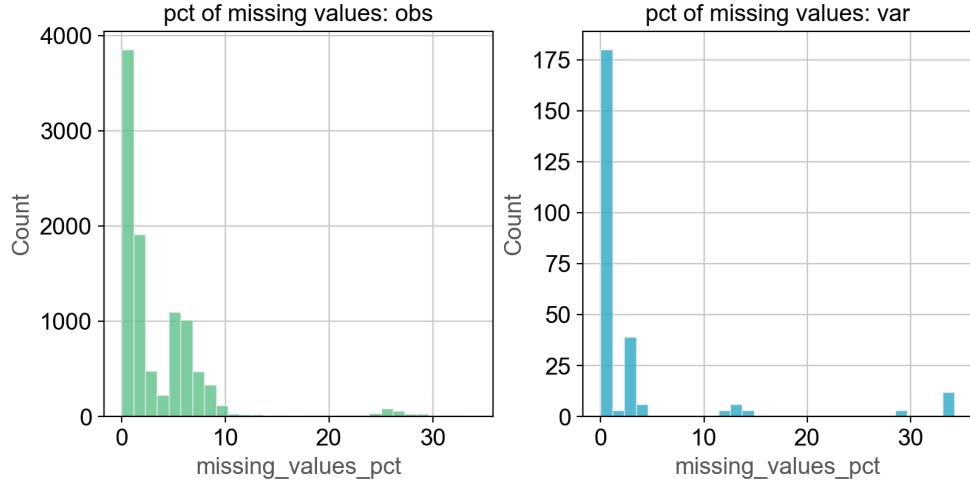
# Winsorizing & summarization

 Helps reduce outlier influence while keeping data shape



# Missing Data

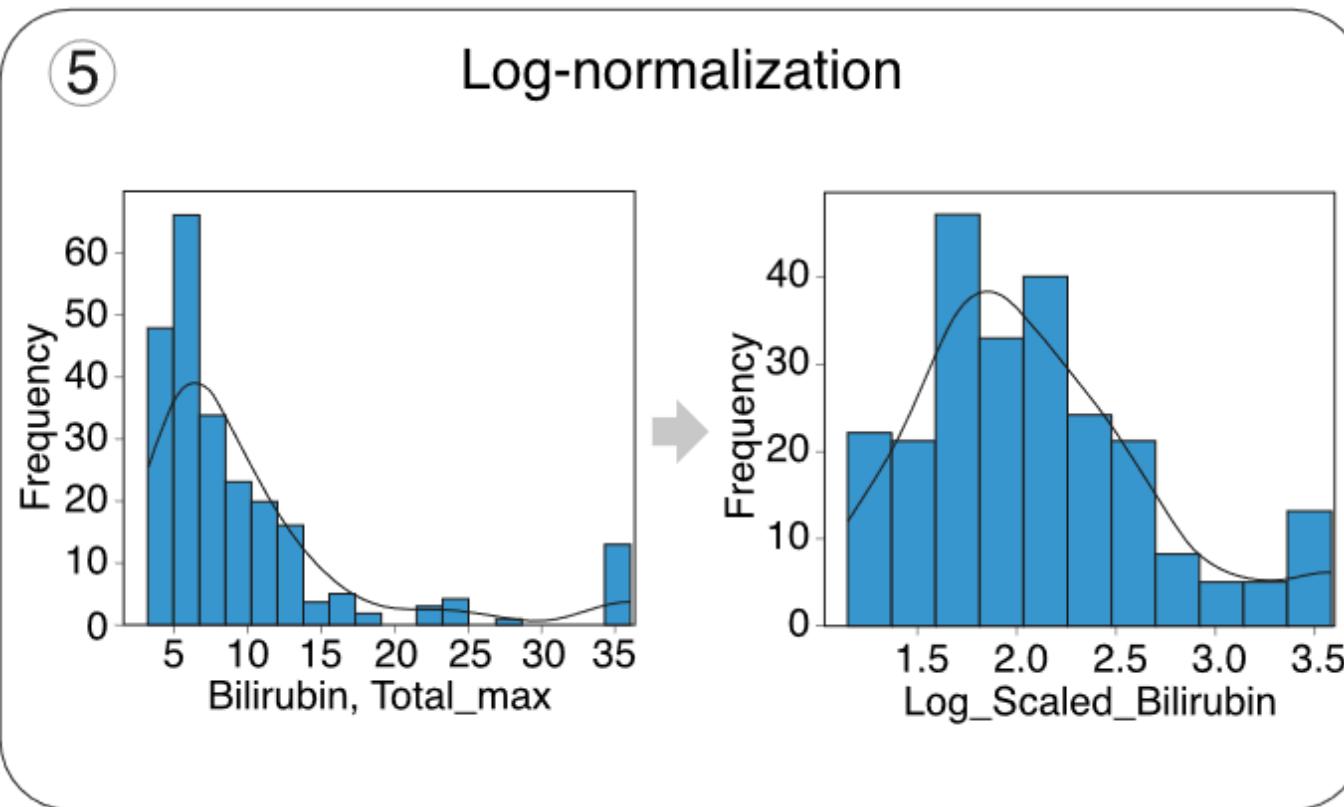
## KNN - 20 neighbors



the percentage of missing values in all features did not exceed 40%, however, some features were not complete. We used K-nearest neighbors imputation (with  $K = 20$ ) to fill in missing values. Each missing value was estimated by averaging the same feature across its 20 nearest neighbors.

# Normalization

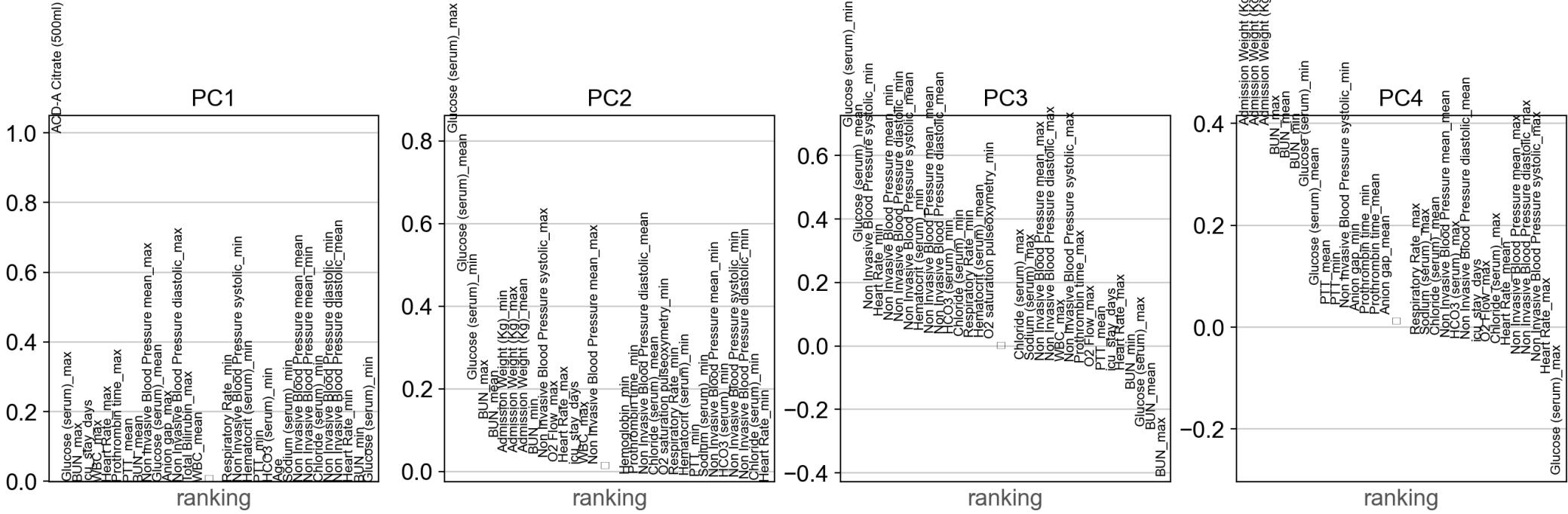
## *Log normalization*



```
1 adata.var["coefficient.variation"] =  
2     adata.var["standard_deviation"] / adata.var["mean"]  
3 ) * 100  
4 adata.var.loc[(adata.var["coefficient.variation"] > 50) & (adata.var["mean"] > 50),]  
5 ep.pp.log_norm(adata, vars=vars_to_normalize, offset=1)
```

# Principal Component Analysis (PCA)

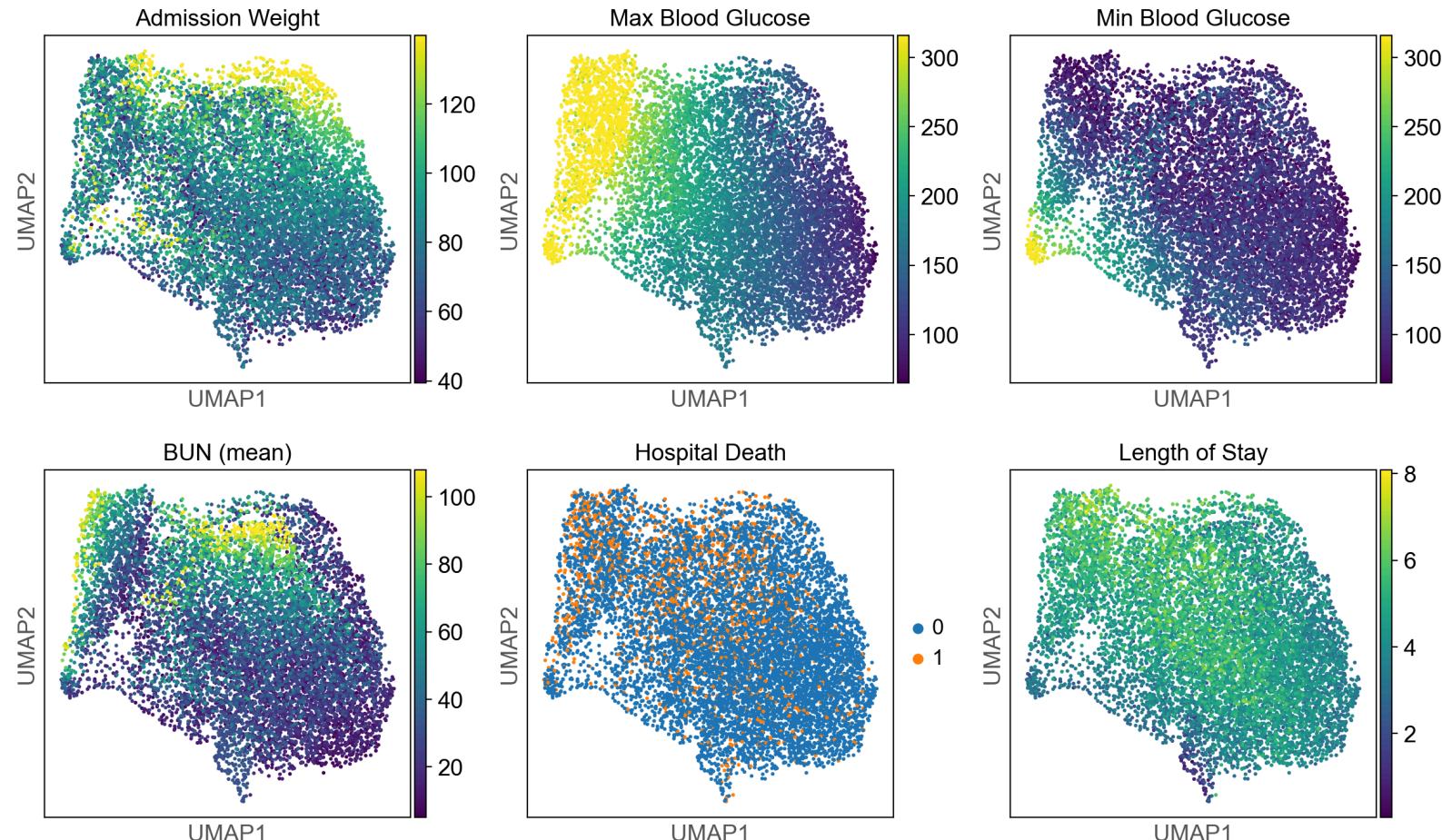
## *Dimensionality reduction*



PCA was used to transform our high-dimension data into a smaller set of uncorrelated components. The reduced representation was then used as input for the neighbors graph calculation.

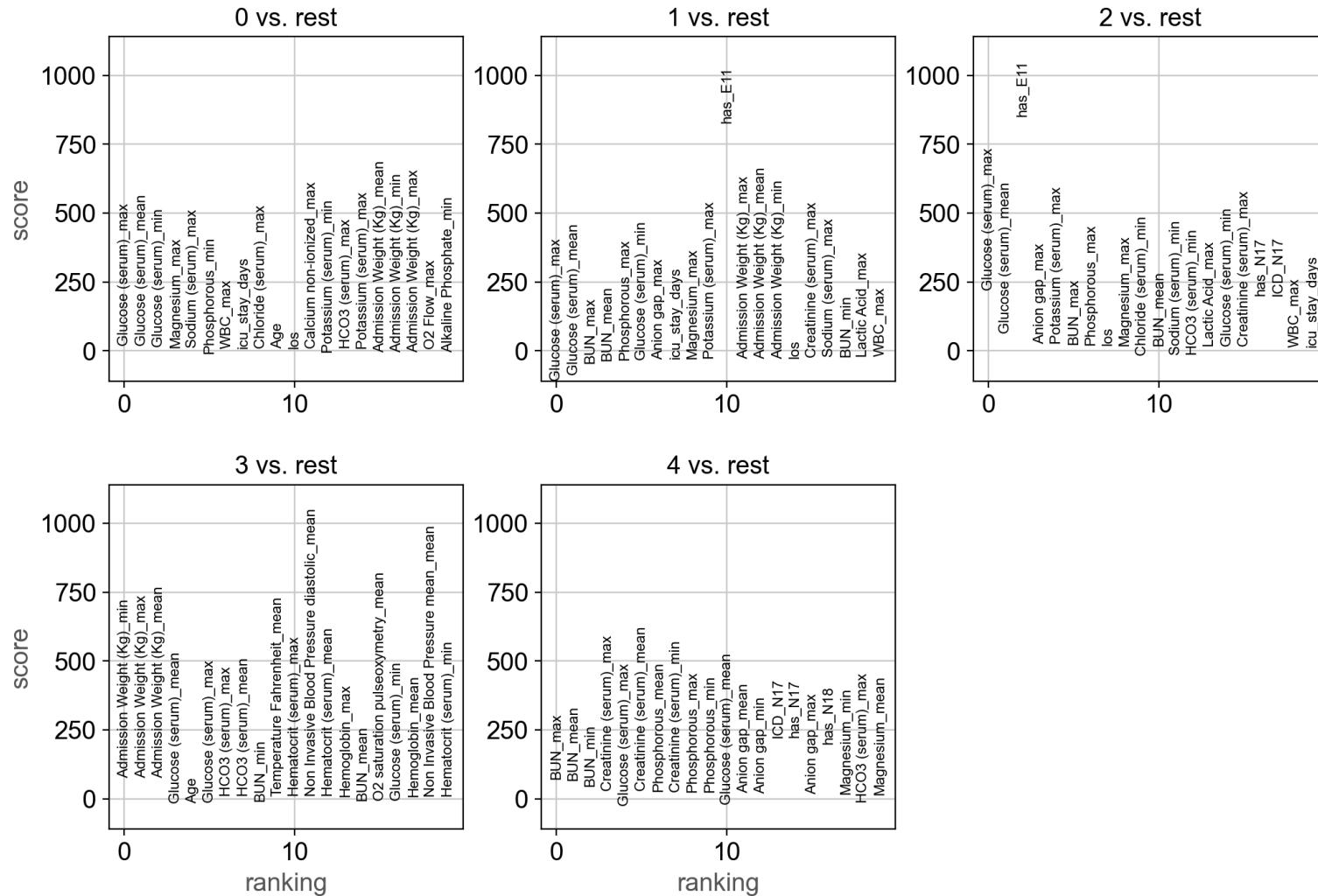
# UMAP

## *Uniform Manifold Approximation and Projection (UMAP)*



This figure shows a UMAP embedding of our dataset after dimensionality reduction with PCA. Each dot is an ICU stay, positioned based on similarity across high-dimensional vitals, labs, and demographics.

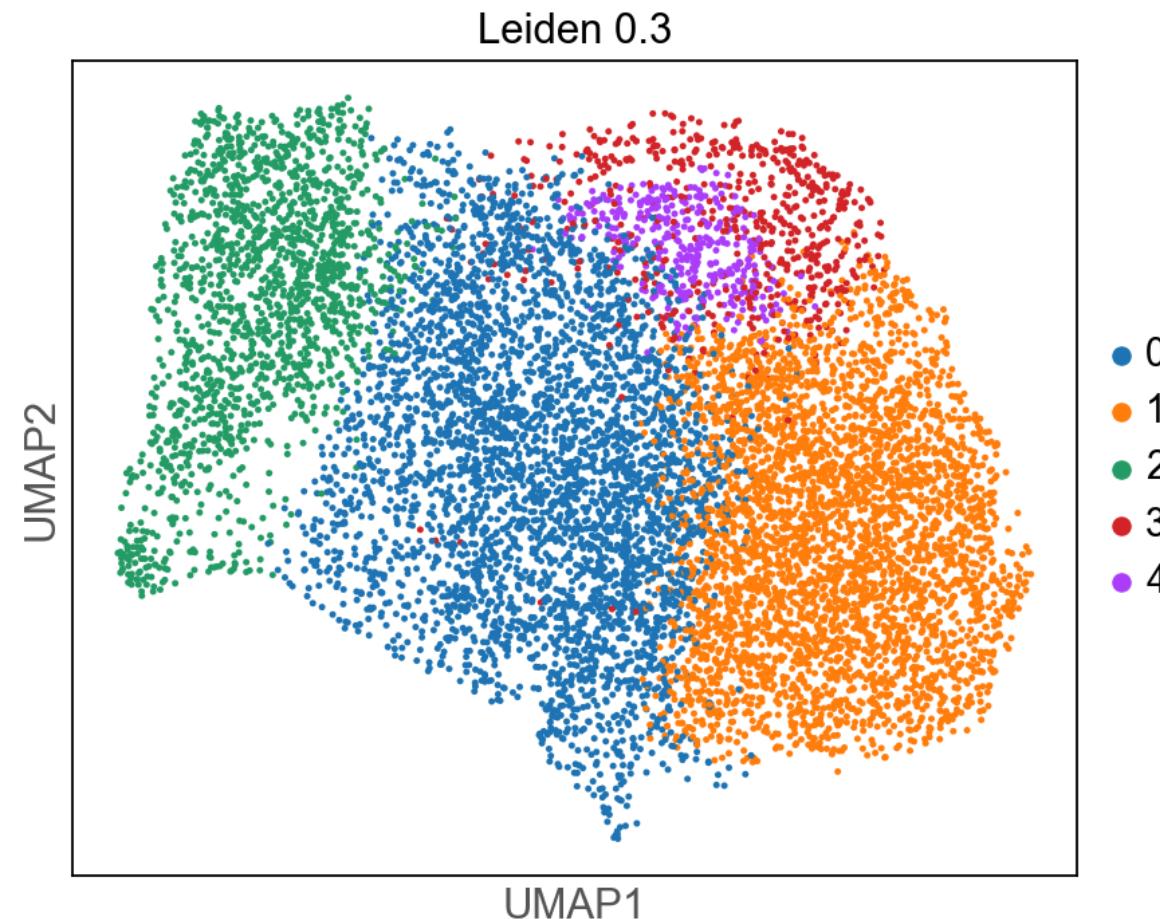
# Leiden Clusters



After projecting patient data into UMAP space, we applied Leiden clustering to identify subgroups. The y-axis represents importance scores, while the x-axis ranks features from most to least informative for each cluster. How strongly this variable distinguishes this cluster from the rest, based on the Wilcoxon test statistic.

# Cluster identification

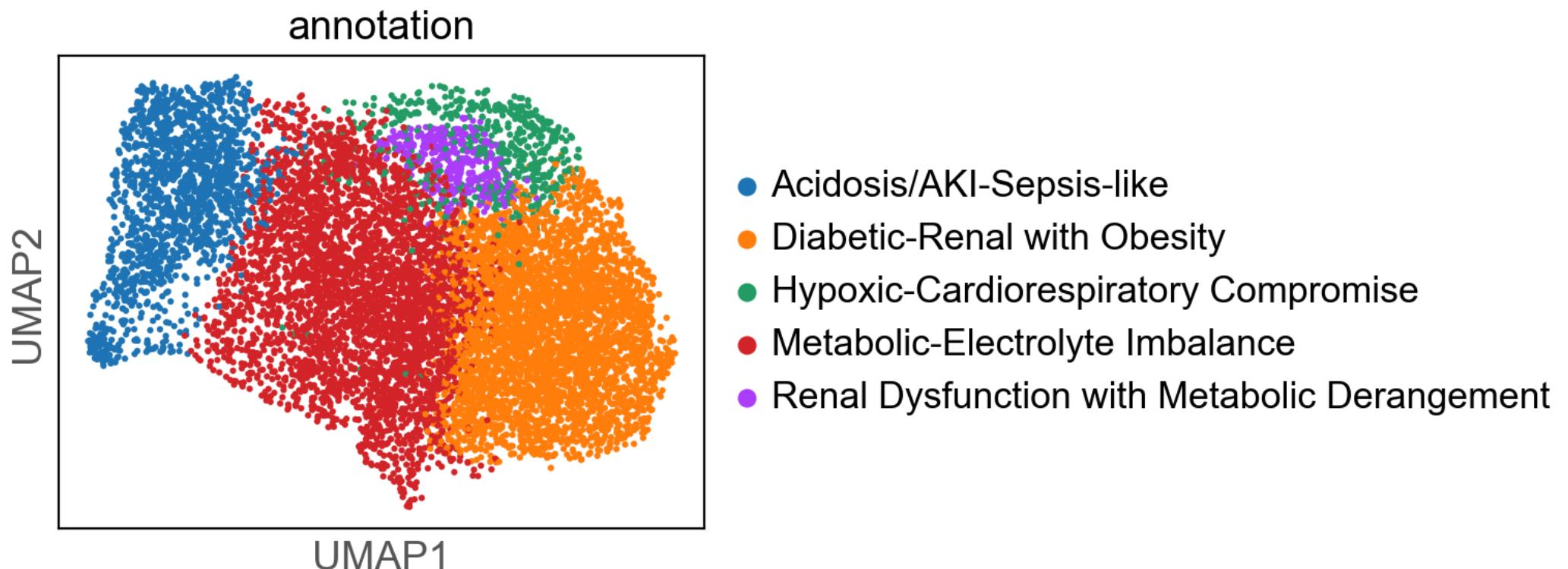
*Leiden 0.4*



Leiden is a fast, community detection algorithm that improves on Louvain by guaranteeing well-connected, high-quality clusters. Often used in network and single cell data. Optimizes the modularity of Louvain where clusters can become disconnected.

# Phase 1 - Results

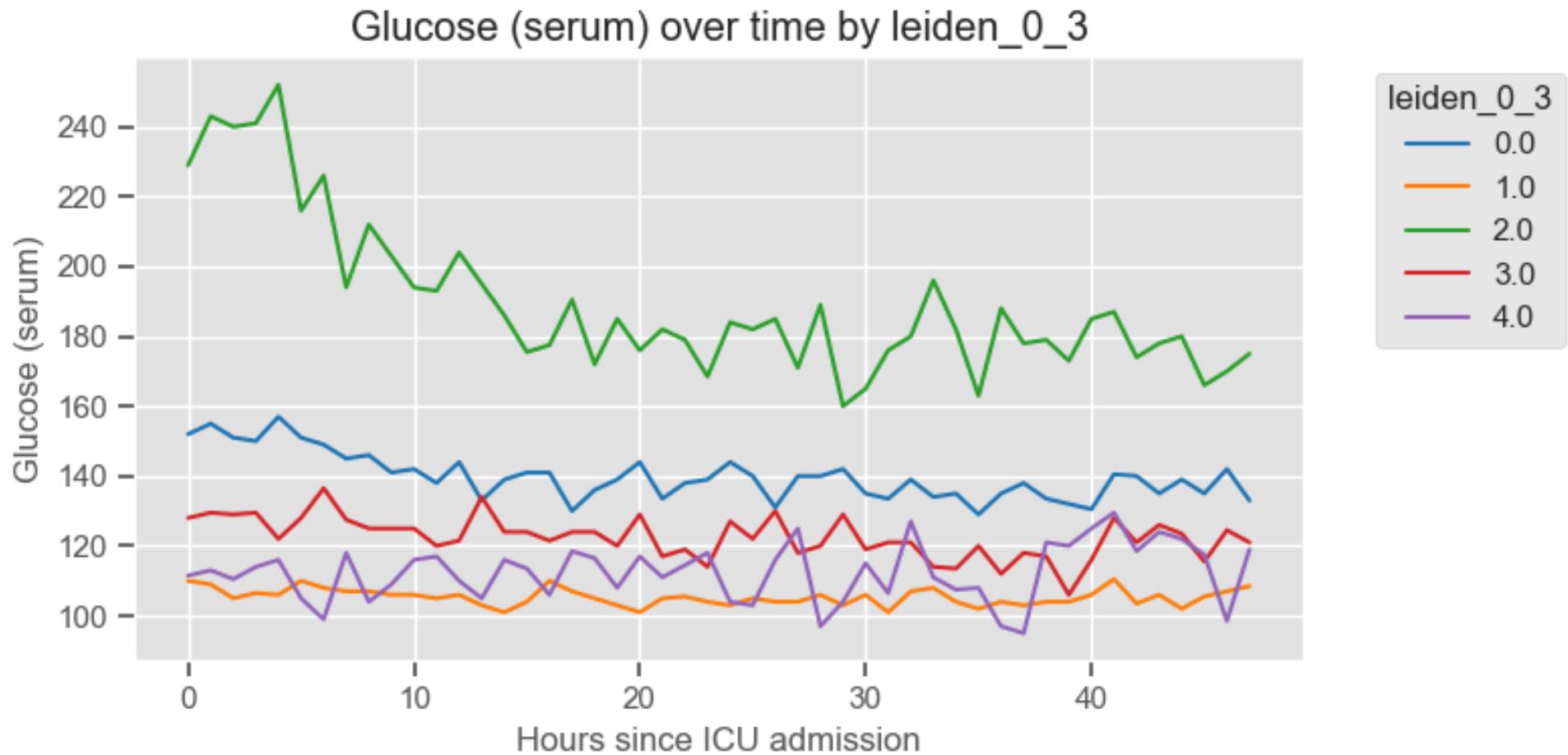
*GPT-Assisted annotation of clusters for disease phenotypes*



Gupta and colleagues had a group of physicians evaluate and annotate their subgroups; we used ChatGPT for assistance [6].

# Glucose levels measured over time

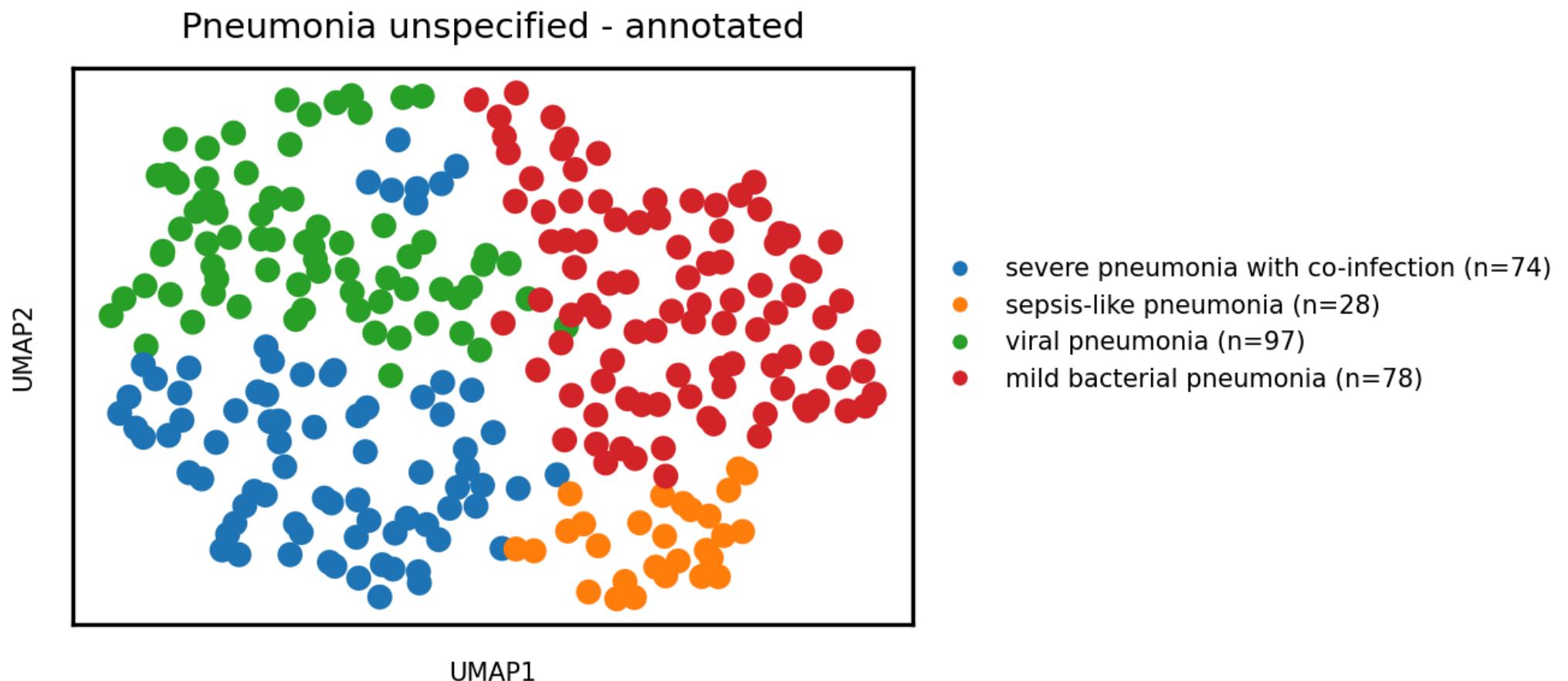
*By cluster*



This is an example of how the groups differ in one specific measure (glucose serum) over time. Note the most severe cluster w/diabetic sepsis enters the ICU with very high sugars.

# Comparison with ehrapy paper

*Pediatric cohort [2]*



Pediatric clusters. Retrieved from [2]

# Comparing Adult & Pediatric Phenotypes

*Validation with original pediatric cohort*

Pediatric Phenotype	Adult Cluster	Shared Signals
<b>Sepsis-like</b>	Cluster 0 / 2	↑ Creatinine, Hypotension, Acidosis, possibly ↑ mortality
<b>Severe w/ Co-infection</b>	Cluster 2 / 4	↑ LOS, AKI, Electrolyte imbalance
<b>Mild bacterial</b>	Cluster 1	Low inflammation, metabolic comorbidity (e.g., diabetes/obesity)
<b>Viral pneumonia</b>	Cluster 3	Stable BP, ↑ O2 sat, less renal involvement

Adult & pediatric clusters overlap but also reflect comorbidity burden  
(e.g., diabetes) not observed in pediatric patients.

# Phase 2: predicting in-hospital mortality

# Phase 2 - ML Extension

*Goal: Build temporal ML model using pneumonia cluster labels*

- Predict in-hospital mortality from early ICU data
  - Binary target: **hospital\_death** (0 = survived, 1 = died)
- Vitals, labs, and structured demographic features (first 48h)
  - Cluster info (Leiden groups) included as one-hot encoded features
- Stratified dataset for train-test split of (80/20)
  - Class imbalance handled by undersampling survivors (only 24% mortality in df)

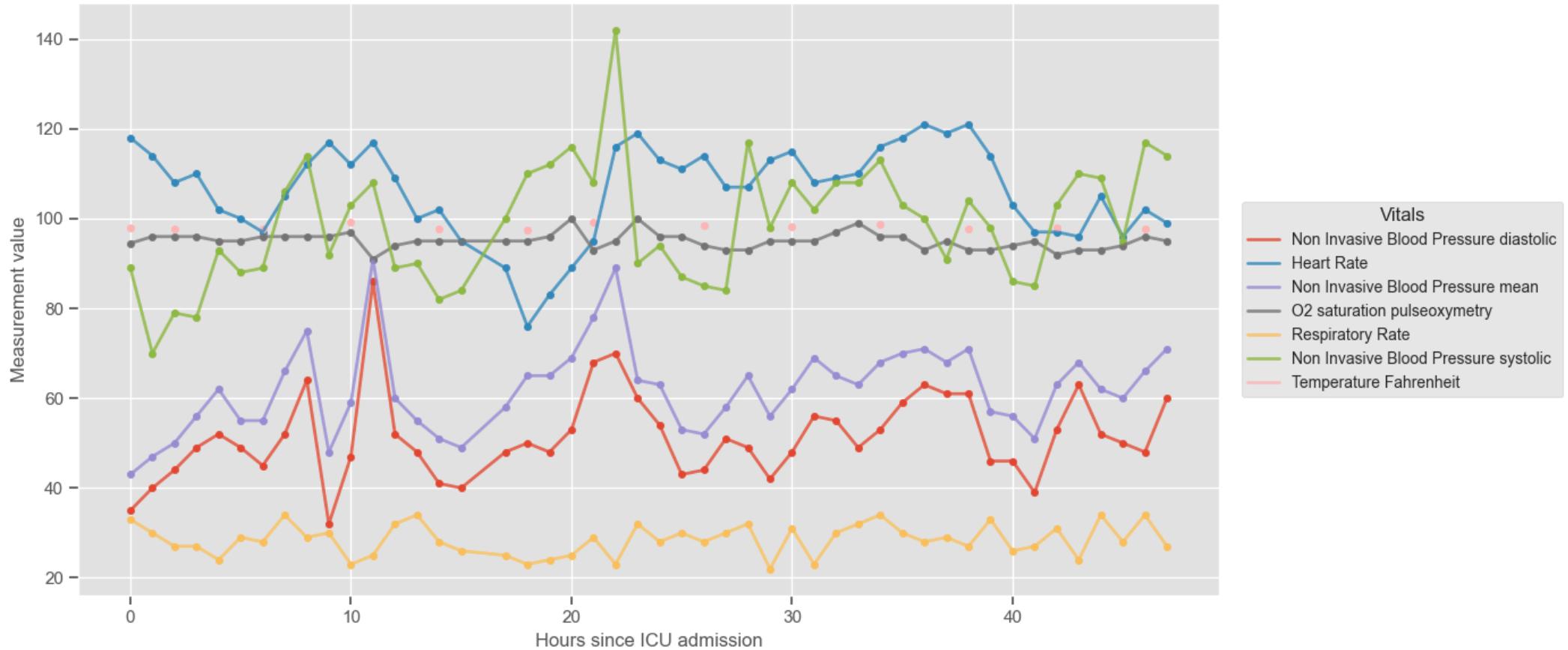
# Phase 2 - Model Architecture

*Predicting in-hospital mortality from first 48 hours*

- Random Forest (GridSearchCV, hyperparameter tuning)
- XGBoost (tree boosting baseline)
- Neural Net (MLP-Classifier, small feedforward network)

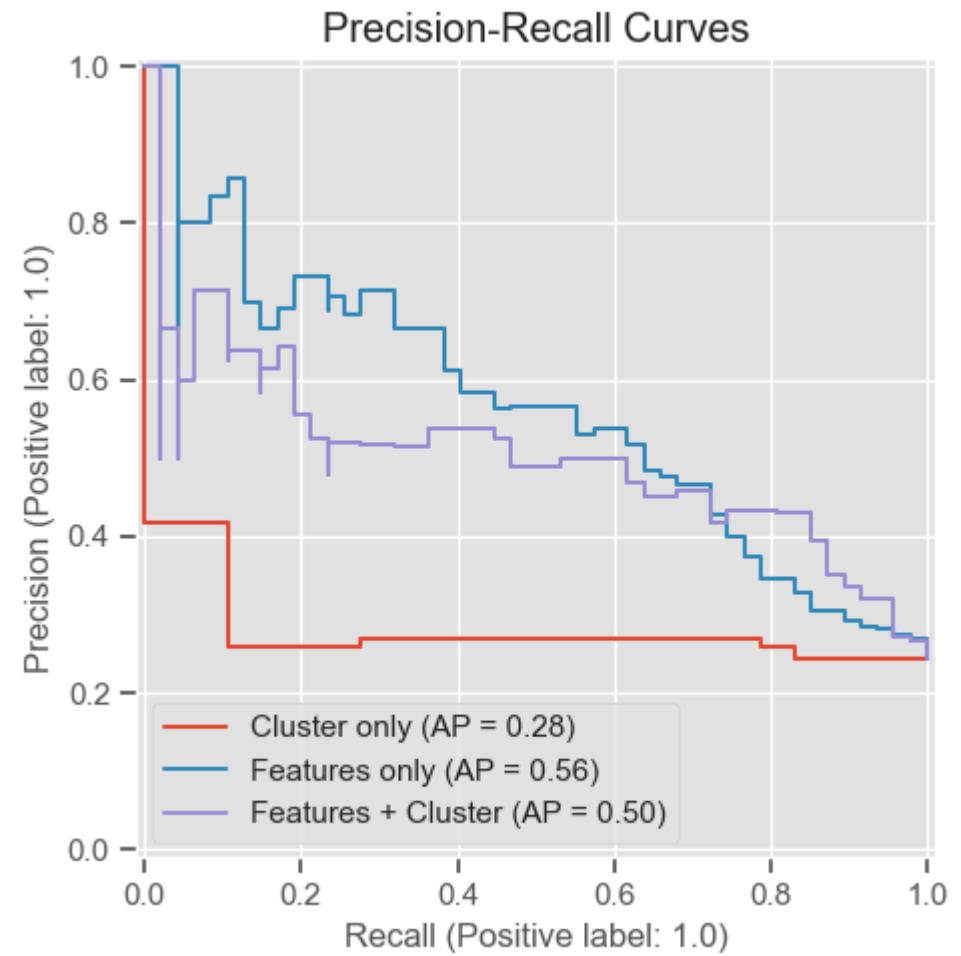
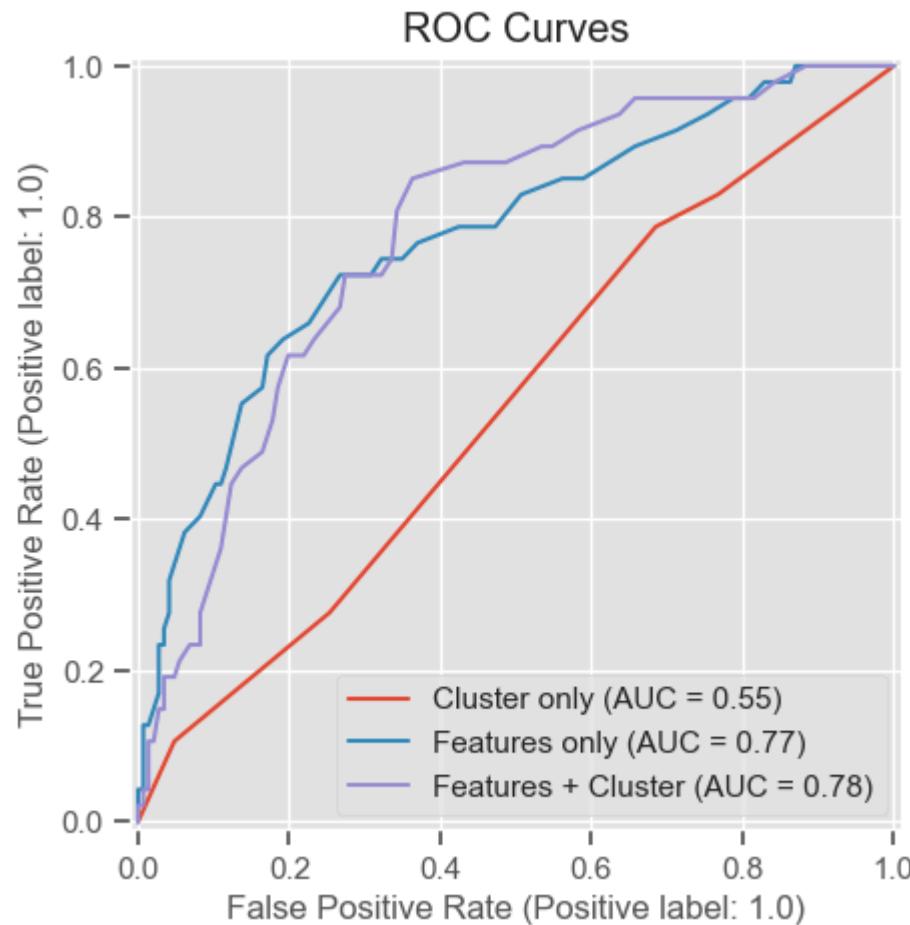
# Time Series Extraction

Vitals were aggregated (min, mean, max) every hour for first 48 hrs



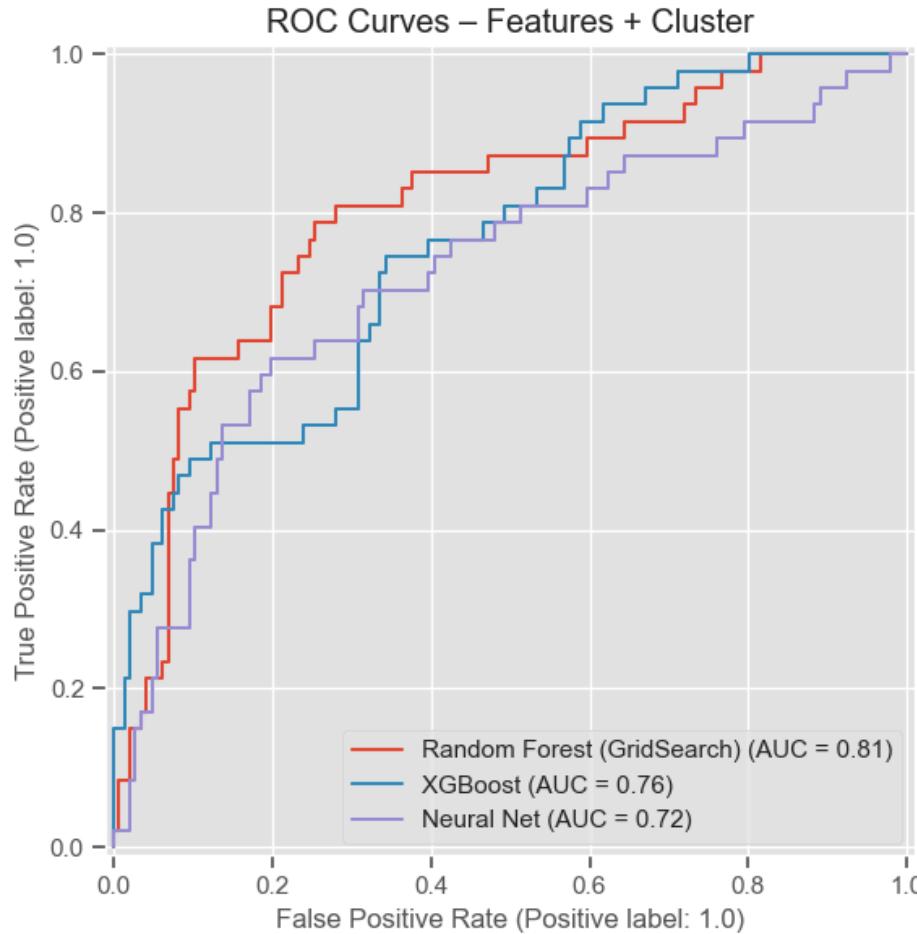
# Results: RF Feature Comparison

*With and without leiden clusters*



# Results: RF, XGB & NN Race

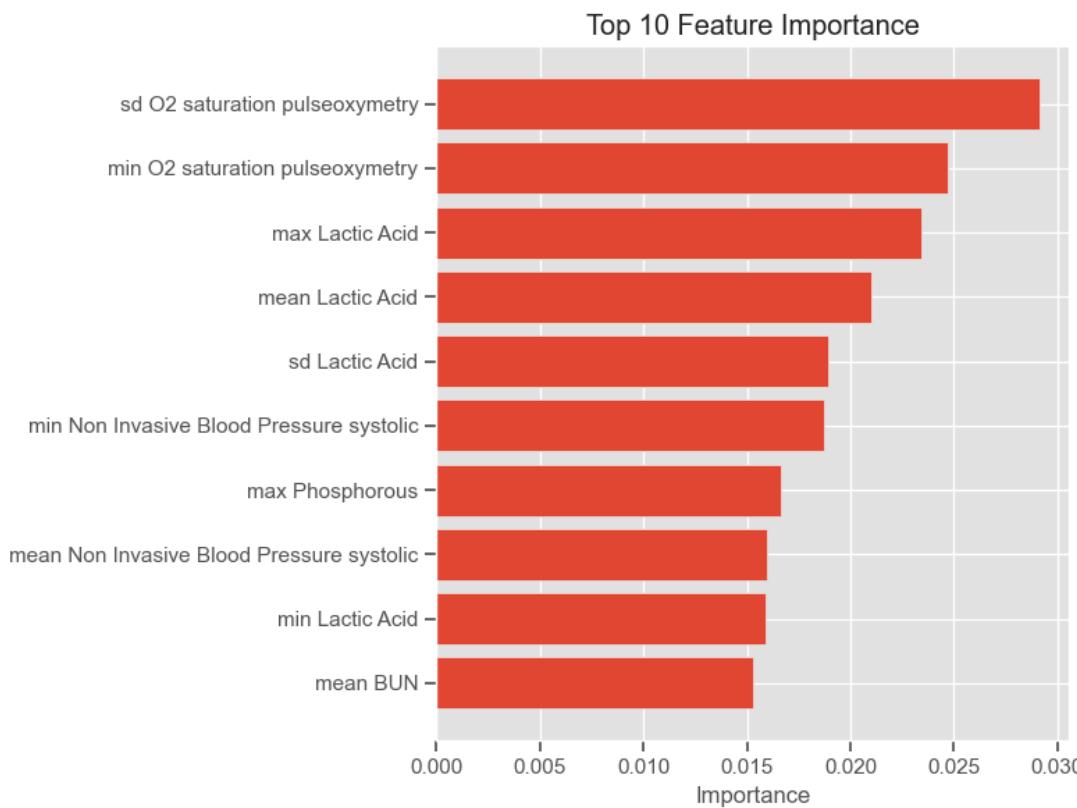
*Random Forest + clusters for the win*



The model does well at identifying survivors but struggles with catching true deaths (low recall). High AUC suggests the model still ranks high-risk patients reasonably well, which could be improved with better threshold tuning or resampling

# Features that contribute most to predicting in-hospital death in pneumonia patients

SHAP



$O^2$  sat., lactic acid levels, and BP stability are  contributors to mortality prediction. These align well with clinical expectations around respiratory failure, sepsis, and hemodynamic instability in high-risk pneumonia patients.



# Next Steps

- **Robust Modeling & Tuning**

Explore advanced ML pipelines with deeper hyperparameter tuning, model ensembles, and uncertainty quantification.

*Limited by time and compute.*

- **Temporal Modeling**

Incorporate trends across time using time-series models or RNNs to better capture clinical trajectories.

- **Validation on External Cohorts**

Evaluate model generalizability and phenotype stability on new datasets or hospital systems.

- **Interpretability & Explainability**

Use SHAP or similar tools to explore feature importance and clinical relevance of phenotypes.



## Clinical Collaboration

It would be important to work with actual clinicians to validate clusters and explore practical applications in care delivery.

# Key Takeaways

## *Conclusion*

- ehrapy has a steep learning curve; but may be worth it
- EHR phenotyping benefits from hybrid unsupervised + supervised approaches
- Open-source workflows like ehrapy + PyTorch are promising for real-world EHR work

# Trials and tribulations

```
2 demographics_features = ['Age', 'gender', 'ethnicity', 'insurance']
3 demographics_cat_features = ['gender', 'ethnicity', 'insurance']
4
5 # Diagnoses - explicit list
6 diagnoses_features = ['ICD_E87', 'ICD_I50', 'ICD_N17', 'ICD_J96']
7 has_features = adata.var_names[adata.var_names.str.startswith("has_")].tolist()
8
9 # Combine them
10 diagnoses_features += has_features
11
12 # Lab measurements - range between two variable names
13 start = adata.var_names.get_loc("ALT_max")
14 end = adata.var_names.get_loc("WBC_min")
15 lab_measurements_features = adata.var_names[start:end + 1].tolist()
16
17 # Medications - same deal
18 start = adata.var_names.get_loc("Acetaminophen-IV")
19 end = adata.var_names.get_loc("Vecuronium")
20 medications_features = adata.var_names[start:end + 1].tolist()
21
22 # Procedures - starts-with filter
23 procedures_features = adata.var_names[adata.var_names.str.startswith("proc_")].tolist()
24
25 # Microbiology - same as labs/meds
26 start = adata.var_names.get_loc("PROBABLE ENTEROCOCCUS_positive")
27 end = adata.var_names.get_loc("ACINETOBACTER BAUMANNII COMPLEX_positive")
28 microbiology_features = adata.var_names[start:end + 1].tolist()

# Assign the measurement groups to features in .var
measurement_group = []

for feature in adata.var_names:
    if feature in demographics_features:
        measurement_group.append('demographics')
    elif feature in diagnoses_features:
        measurement_group.append('diagnoses')
    elif feature in lab_measurements_features:
        measurement_group.append('lab_measurement')
    elif feature in medications_features:
        measurement_group.append('medications')
    elif feature in procedures_features:
        measurement_group.append('procedures')
    elif feature in microbiology_features:
        measurement_group.append('microbiology')

# Add the measurement group to the feature
adata.var['measurement_group'] = measurement_group
```

SESSION HELP VIEWER CONNECT

VARIABLES

03-Ehrapy-Prep.ipynb

- > cols\_to\_add
- > cols\_to\_convert
- > demographics\_cat
- > demographics\_fea
- > diagnoses\_feature
- > lab\_measurement
- > measurement\_group
- > medications\_feat
- > metrics
- > microbiology\_feat
- > proc\_dict
- > procedures\_featur
- > start
- > vars\_to\_normalize

CLASSES

TableOne <class 'tableone.tableone.TableOne'>

PLOTS

CONSOLE

Python 3.12.9 (Venv: venv) ~\Documents\MIMIC-FINAL

IPython 9.0.2 -- An enhanced Interactive Python. Type '?' for help.

Tip: The `%timeit` magic has an '-o` flag, which returns the results, making it easy to See `%timeit?`.

# Math

## *Winsorization*

For a value  $x_i$  in feature  $X$ :

$$x_i^{\text{win}} = \begin{cases} P_1(X), & \text{if } x_i < P_1(X) \\ x_i, & \text{if } P_1(X) \leq x_i \leq P_{99}(X) \\ P_{99}(X), & \text{if } x_i > P_{99}(X) \end{cases}$$

Where:

- $P_1(X)$  = 1st % of  $X$
- $P_{99}(X)$  = 99th % of  $X$
- $x_i^{\text{win}}$

# KNN

For a missing value  $x_{i,j}$  in row  $i$ , column  $j$ :

$$\hat{x}_{i,j} = \frac{1}{K} \sum_{k \in \mathcal{N}_K(i)} x_{k,j}$$

# References

- [1] M. Gupta, B. Gallamoza, N. Cutrona, P. Dhakal, R. Poulain, and R. Beheshti, “An Extensive Data Processing Pipeline for MIMIC-IV,” in *Proceedings of the 2nd machine learning for health symposium*, in Proceedings of machine learning research, vol. 193. PMLR, 2022, pp. 311–325. Available: <https://proceedings.mlr.press/v193/gupta22a.html>
- [2] M. Gupta, B. Gallamoza, N. Cutrona, P. Dhakal, R. Poulain, and R. Beheshti, “An extensive data processing pipeline for MIMIC-IV,” *Proceedings of machine learning research*, vol. 193, pp. 311–325, Nov. 2022, Available:  
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9854277/>
- [3] S. Yang, P. Varghese, E. Stephenson, K. Tu, and J. Gronsbell, “Machine learning approaches for electronic health records phenotyping: A methodical review,” *Journal of the American Medical Informatics Association*, vol. 30, no. 2, pp. 367–381, Feb. 2023, doi: [10.1093/jamia/ocac216](https://doi.org/10.1093/jamia/ocac216). Available: <https://doi.org/10.1093/jamia/ocac216>
- [4] A. E. W. Johnson *et al.*, “MIMIC-IV, a freely accessible electronic health record dataset,” *Scientific Data*, vol. 10, no. 1, p. 1, Jan. 2023, doi: [10.1038/s41597-022-01899-x](https://doi.org/10.1038/s41597-022-01899-x). Available: <https://www.nature.com/articles/s41597-022-01899-x>
- [5] I. Virshup, S. Rybakov, F. J. Theis, P. Angerer, and F. A. Wolf, “anndata: Annotated data,” 2021, doi: [10.1101/2021.12.16.473007](https://doi.org/10.1101/2021.12.16.473007)
- [6] OpenAI, “ChatGPT (april 2024 version).” <https://chat.openai.com>, 2024.