

# Project 2 Report:

# AI Agents for Sepsis EHR Analysis

*By Vedanshi Shah*

Due Sunday, March 23, 2025 at 11:59 PM

*COSI 149B: Practical Machine Learning with Big Data*

**Prof:** Pengyu Hong

Abstract.....	2
I. Introduction.....	2
II. Dataset and Preprocessing.....	2
III. Methodology.....	5
III.I. System Design and Tool Selection.....	5
III.II. Generative AI System Lifecycle Integration.....	5
III.III. Tool Implementation.....	7
III.III.I. Tool Pipeline Overview.....	7
III.III.I.I. T1: Data Loading.....	7
III.III.I.II. T2: Imputation of Missing Values.....	10
III.III.I.III. T3: Mortality Prediction Using a Pre-Trained Model.....	10
III.III.I.IV. T4: Statistical Analysis of Features.....	12
III.IV. Challenges and Refinements.....	13
IV. Imputation with tPatch-GNN.....	14
V. Mortality Prediction and Interpretability.....	14
VI. Exploration of AI Agent Frameworks.....	16
VII. Final Agent Design and System Architecture.....	17
User Interaction & Language Understanding.....	17
VIII. Discussion and Reflections.....	18
IX. Future Work.....	18
X. Conclusion.....	19
XI. Acknowledgments.....	19
XII. References.....	20
XI. Appendix.....	21

# Abstract

This project explores the development of a clinical AI agent designed to assist healthcare professionals by interpreting incomplete electronic health records (EHR) and predicting 90-day mortality risk for sepsis patients. The agent performs three interconnected tasks: imputing missing clinical data using a graph neural network, predicting patient outcomes through a transparent machine learning model, and generating natural language explanations using a large language model (LLM) interface. I experimented with various AI agent frameworks—including OpenManus, Ollama, Superagent, and several agents from the Awesome-AI-Agents repository—but ultimately selected **Langchain** for its modular design, dynamic tool invocation, and conversational memory capabilities. This report details the problem setting, technical architecture, model choices, and agent framework exploration, culminating in a fully functional and interpretable clinical assistant capable of assisting in real-world, high-stakes environments like intensive care.

Try out the live website: <https://practical-ml-project-2.streamlit.app/>

## I. Introduction

Sepsis remains one of the most time-sensitive and life-threatening conditions in intensive care medicine. Early recognition and timely intervention can drastically improve outcomes, but clinicians often work with incomplete, irregularly sampled EHR data that impedes diagnosis and prognosis. This project addresses this challenge by designing an intelligent agent that can reason over sparse clinical data, make predictive inferences, and explain its reasoning in natural language. Beyond the technical implementation, the broader goal of the project was to investigate how modern AI systems—including graph-based models and LLMs—can be integrated to support clinical workflows with transparency and robustness.

To this end, I envisioned a system that not only predicts mortality risk with reasonable accuracy but also justifies its conclusions and adapts its output based on the quality and availability of data. This focus on interpretability, reliability, and usability shaped every component of the project—from data preprocessing to model design to framework selection for the AI agent interface.

## II. Dataset and Preprocessing

The dataset used for this project consists of multivariate time-series data extracted from anonymized EHRs of sepsis patients. The variables include lab test results (e.g., lactate, platelets, creatinine), vital signs (e.g., heart rate, blood pressure), and demographic information. As is typical in real-world hospital datasets, data points were recorded irregularly, with a high percentage of missing values, especially in more costly or invasive tests.

The preprocessing pipeline involved resampling the data into fixed-size time windows and organizing it as temporal sequences per patient. I explored several baseline imputation techniques such as mean fill, forward fill, and K-nearest neighbors, but these approaches either made strong (and often incorrect) assumptions or introduced biases that distorted downstream predictions. These limitations led me to adopt a more advanced imputation technique that could learn temporal and relational structures within the data.

Variable	Description
bloc	The order of entry of icu stay
icustayid	Unique identifier for a patient's ICU stay
charttime	Timestamp when the data was recorded (charting time)
gender	Patient's gender
age	Patient's age in days
elixhauser	Elixhauser comorbidity score, summarizing the burden of chronic diseases
<b>mortality_90d</b>	Mortality status within 90 days of admission or discharge. This is the class label
Weight_kg	Patient weight in kilograms
GCS	Glasgow Coma Scale score, a measure of consciousness
HR	Heart rate (beats per minute)
SysBP	Systolic blood pressure
MeanBP	Mean arterial blood pressure
DiaBP	Diastolic blood pressure
RR	Respiratory rate (breaths per minute)
SpO2	Peripheral capillary oxygen saturation (%)
TempC	Body temperature in degrees Celsius
FiO2_1	Fraction of inspired oxygen (FiO2) provided
Potassium	Serum potassium level
Sodium	Serum sodium level
Chloride	Serum chloride level
Glucose	Blood glucose level

BUN	Blood urea nitrogen level
Creatinine	Serum creatinine level
Magnesium	Serum magnesium level
Calcium	Serum calcium level
Ionised_Ca	Ionized calcium level
CO2_mEqL	Carbon dioxide content measured in mEq/L
SGOT	Aspartate aminotransferase (AST) level
SGPT	Alanine aminotransferase (ALT) level
Total_bili	Total bilirubin level
Albumin	Serum albumin level
Hb	Hemoglobin concentration
WBC_count	White blood cell count
Platelets_count	Platelet count
PTT	Partial thromboplastin time, assessing blood clotting
PT	Prothrombin time, another measure of clotting function
INR	International Normalized Ratio, standardizing PT results
Arterial_pH	pH of arterial blood
paO2	Partial pressure of oxygen in arterial blood
paCO2	Partial pressure of carbon dioxide in arterial blood
Arterial_BE	Arterial base excess, indicating metabolic balance
HCO3	Bicarbonate concentration in the blood
Arterial_lactate	Lactate level in arterial blood (marker for tissue hypoxia)
mechvent	Indicator of whether the patient was mechanically ventilated
Shock_Index	Index of shock evaluation
PaO2_FiO2	Ratio of paO2 to FiO2, used to assess lung oxygenation efficiency
median_dose_vaso	Median dose of vasopressors administered during ICU stay
max_dose_vaso	Maximum dose of vasopressors administered

input_total	Total volume of fluids administered
input_4hourly	Fluid input measured over each 4-hour period
output_total	Total urine output (losses) recorded
output_4hourly	Urine output measured over each 4-hour period
cumulated_balance	Cumulative account balance
SOFA	Sequential Organ Failure Assessment score, indicating severity of organ dysfunction
SIRS	Systemic Inflammatory Response Syndrome criteria or score, indicating inflammatory state

**Table 1:** Dataset Descriptors

## III. Methodology

The methodology of this project is centered around the development of an autonomous agent using generative AI systems, specifically leveraging frameworks such as **LangChain**, **Superagent**, and other related tools. The approach follows a structured workflow that spans various stages of system design, implementation, and optimization, adhering to the **Generative AI System Lifecycle**.

### III.I. System Design and Tool Selection

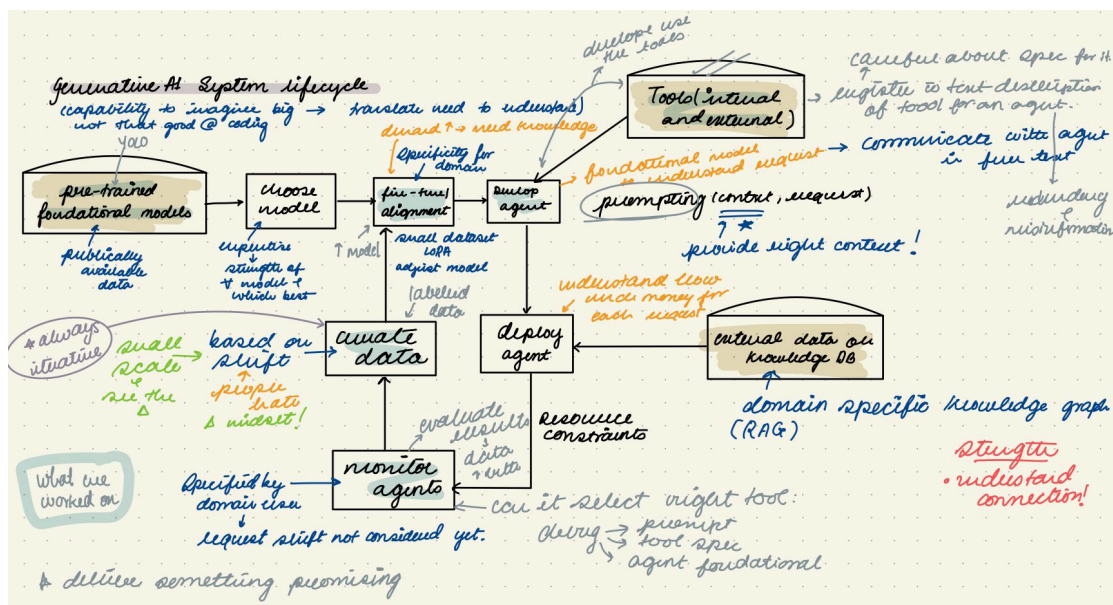
The first step in the methodology was to conduct a thorough evaluation of available tools for building an autonomous agent. Initially, various frameworks were considered, including **Open Manus**, **Ollama**, **Superagent**, **LangChain**, and the **Awesome AI Agents** collection. After testing and comparing the capabilities of these frameworks, I selected **LangChain** due to its rich set of features for integrating large language models (LLMs) and external APIs, which aligns with the needs of the project.

LangChain offers a comprehensive architecture for building autonomous agents that can perform tasks such as context switching, interaction with external data sources, and decision-making. Its flexibility in handling different types of input and generating human-like responses made it the most suitable choice for the project's objectives.

### III.II. Generative AI System Lifecycle Integration

The development process was structured around the **Generative AI System Lifecycle**, which ensures the systematic creation and continuous improvement of the agent. This lifecycle consists of five key stages, which were incorporated as follows:

1. **Design and Planning:** In this phase, the problem space was defined, focusing on the need for an autonomous agent capable of dynamically responding to user queries, fetching information, and interacting with external services. The design included the selection of LangChain as the core framework and outlined integration points with external APIs, including data storage solutions.
2. **Training and Model Selection:** Although the primary task involved using pretrained models, the project required fine-tuning and configuring LangChain to perform specific tasks based on real-time inputs. Training in this case involved setting up and testing multiple agent strategies, determining the most effective approaches for engaging with external services, and testing language model responses.
3. **Evaluation:** Evaluation of the system was conducted through a series of tests designed to assess the agent's ability to respond accurately and effectively. This included testing the agent's capacity for real-time decision-making, processing dynamic inputs, and handling context-switching scenarios. Performance was evaluated using both qualitative feedback and quantitative metrics, such as response accuracy and task completion time.
4. **Deployment:** Once the system was refined through testing, it was deployed for real-world use. This phase involved the integration of LangChain into a functional web application or another platform where the autonomous agent could interact with users. Real-time deployment allowed for continuous monitoring and real-world validation of the agent's performance.
5. **Continuous Improvement:** Following deployment, the project entered the continuous improvement phase. This ongoing stage involved collecting user feedback, tracking performance metrics, and applying updates to enhance functionality. Any issues identified during deployment were addressed through iterative updates to the agent's logic and response generation capabilities.

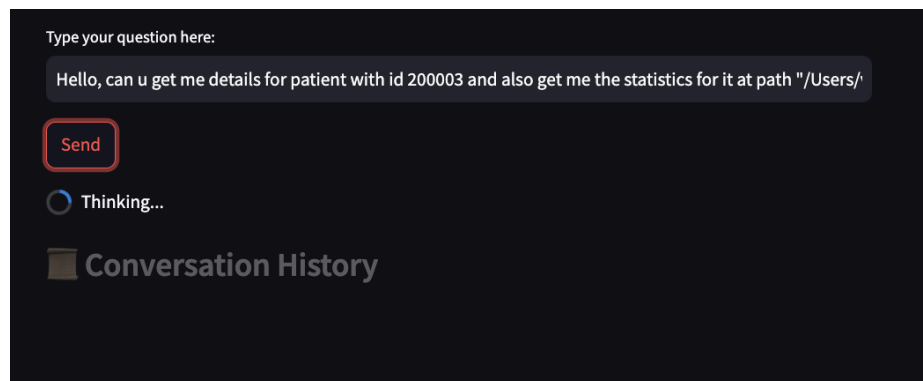


**Figure 1:** Generative AI System Lifecycle from class on April 29, 2025

### III.III. Tool Implementation

The main tool for developing the autonomous agent was **LangChain**. The framework's modular design allowed for seamless integration of different components like chat models, vector stores, and external API calls. LangChain's **agents** class was particularly useful in defining the agent's logic and decision-making process. This class was leveraged to create custom agent behaviors, enabling the system to make intelligent decisions based on user input and external factors.

In addition to LangChain, other tools such as **Superagent** were explored but not fully integrated. **Superagent** provided an alternative approach for building autonomous agents but was ultimately less suited for the project's specific needs. The decision to focus on LangChain was based on its well-documented capabilities, the support for complex workflows, and the ease of integration with various data sources.



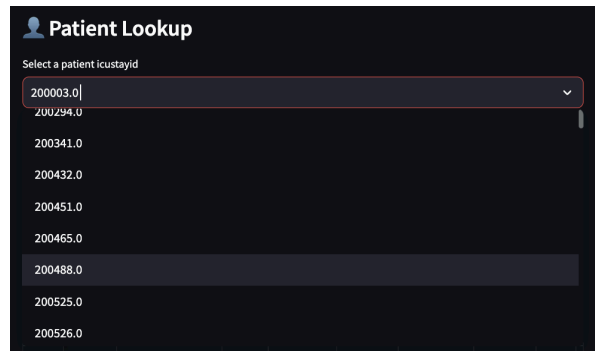
**Figure 2:** Using natural language to get details for patient with a specific ID

#### III.III.I. Tool Pipeline Overview

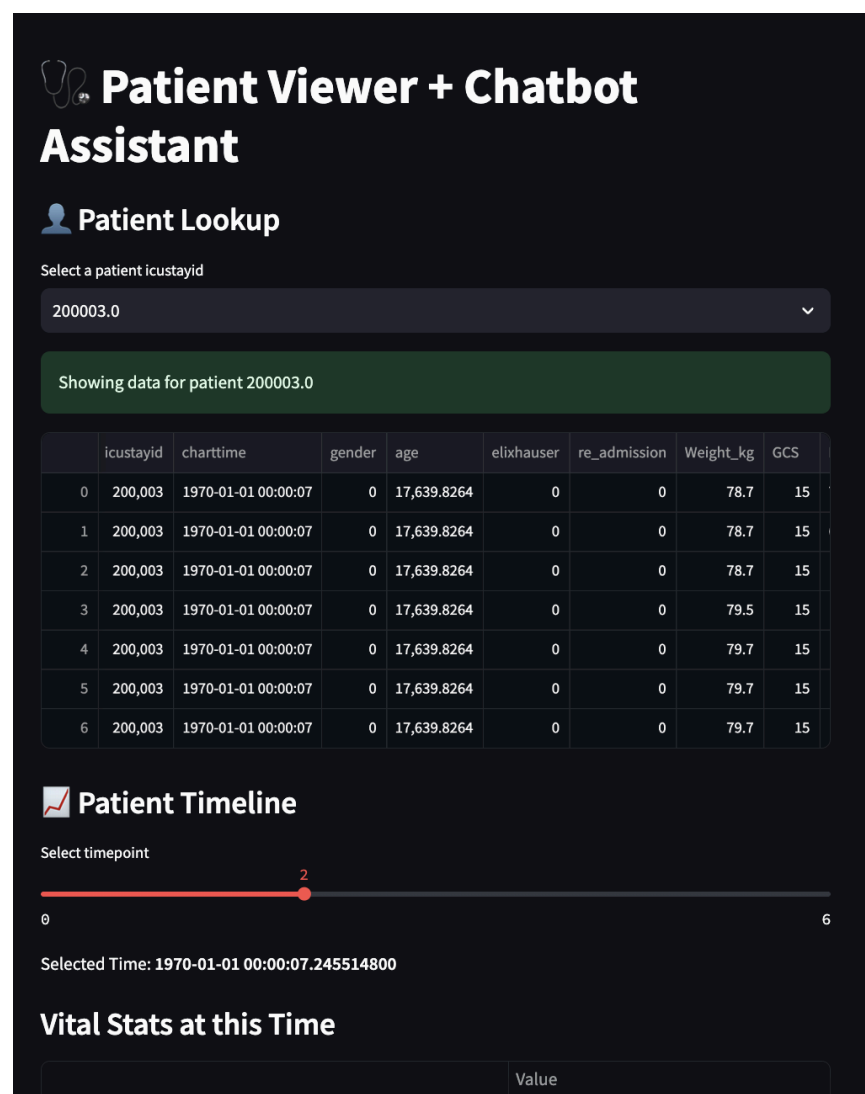
The tool pipeline described in this project is designed to facilitate efficient data processing, model prediction, and statistical analysis, ensuring a streamlined and effective workflow for handling large datasets. The pipeline consists of four primary steps: data loading, imputation of missing values, prediction using a pre-trained model, and the calculation of feature statistics.

##### III.III.I.I. T1: Data Loading

The first step of the pipeline involves loading the raw data into the system. This step is crucial for gathering the dataset, which forms the basis for further processing. The data can come from various sources, including CSV files, databases, or APIs. The data is then structured into a format that can be easily manipulated and passed through the subsequent stages of the pipeline.

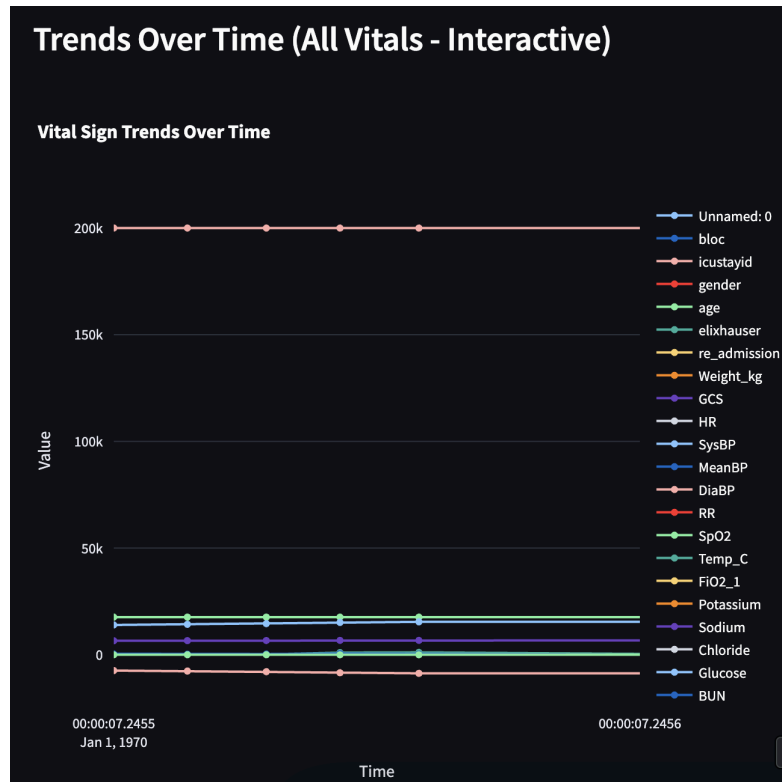


**Figure 3:** Patient Lookup to get patient data for a specific ID can be typed or chosen from the drop-down menu

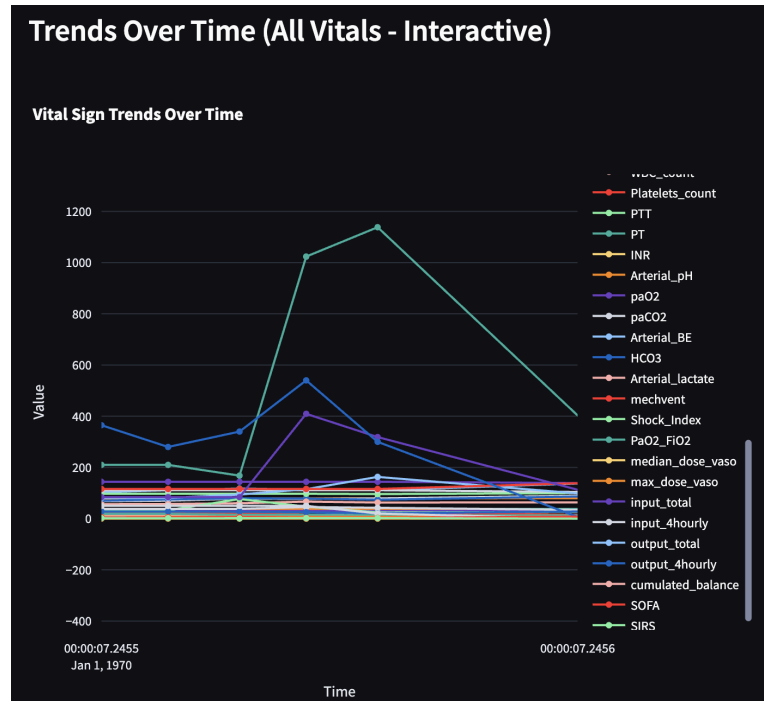


**Figure 4:** Detailed view of the patient ID selected with scrollable information of all occurrences and being able to view specific values for an occurrence by scrubbing the patient timeline.





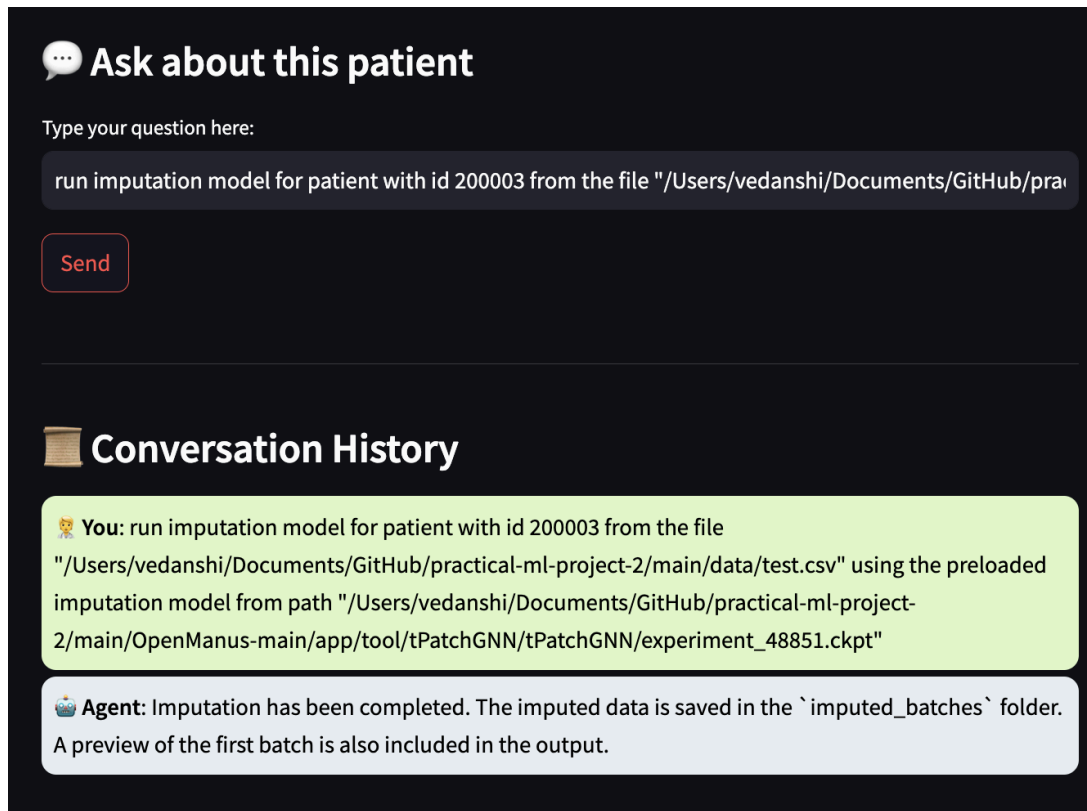
**Figure 5:** Trends of stats over time to see changes at a glance for the patient with ID selected.



**Figure 6:** Trends of stats over time to see changes at a glance for the patient with ID selected that can be zoomed in for more information and hovered over for tooltip information.

### III.III.I.II. T2: Imputation of Missing Values

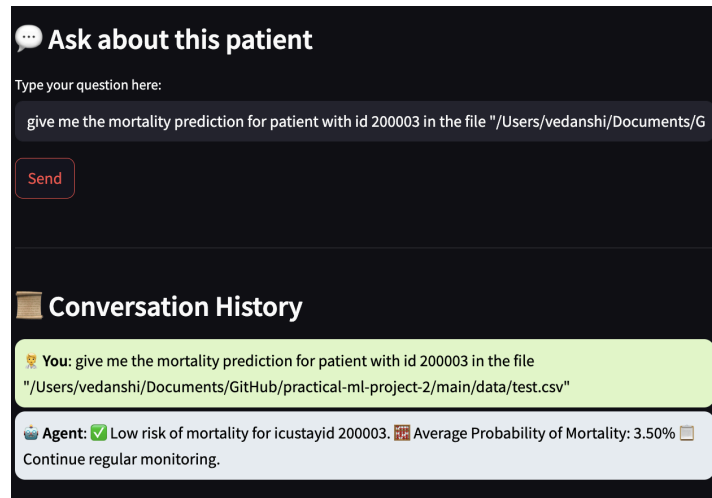
After loading the data, the next step is to handle missing or incomplete values. Imputation is performed to fill in any gaps in the dataset, ensuring that no essential information is lost. The imputation method may vary depending on the dataset and the nature of the missing values. Common techniques include filling missing values with the mean, median, or using more advanced methods like K-Nearest Neighbors (KNN). Once the imputation is performed, the modified dataset is saved temporarily for further processing.



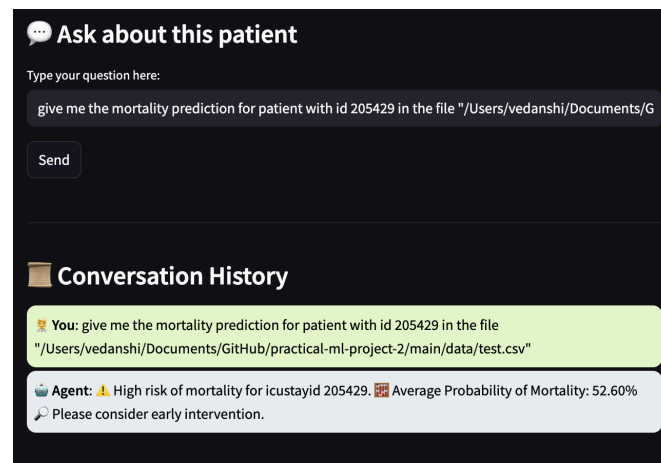
**Figure 7:** Running the imputed value model and inputting the patient data.

### III.III.I.III. T3: Mortality Prediction Using a Pre-Trained Model

Once the dataset is complete, the next step involves passing the imputed data to a machine learning model for prediction. The model is typically pre-trained on similar data, and the goal is to use this trained model to generate predictions based on the current dataset. This step transforms the raw data into valuable insights that can be used for decision-making or further analysis. The results from the prediction model are saved to a temporary file for record-keeping and further utilization.



**Figure 8:** Positive mortality prediction for patient with the probability.



**Figure 9:** Negative mortality prediction for patient with the probability.

```

Accuracy: 0.78
ROC AUC: 0.85
Classification Report:

```

	precision	recall	f1-score	support
0.0	0.92	0.80	0.85	31330
1.0	0.49	0.73	0.58	8357
accuracy			0.78	39687
macro avg	0.70	0.76	0.72	39687
weighted avg	0.83	0.78	0.79	39687

```

Confusion Matrix:
[[24934  6396]
 [ 2298 6059]]
Model saved as 'mortality_model.pkl'
Model loaded successfully.

```

	Predicted Mortality	Probability of Mortality (1)	\
0	0	27.16	
1	0	24.22	
2	0	23.06	
3	0	20.00	

**Figure 10:** Ensemble mortality prediction model accuracy of 78% and AUC of 85% with Classification report

```

Test Accuracy: 0.83
Test AUC: 0.82
Model saved as 'mortality_dnn_model.h5'
1537/1537 ----- 3s 2ms/step
Predicted Class  Probability of Survival (0) \
0                0                97.510002
1                0                97.209999
2                0                97.790001
3                0                97.230003
4                0                97.720001
...              ...              ...
49154            0                91.139999
49155            0                85.089996
49156            0                85.269997
49157            0                92.820000
49158            0                90.669998

```


**Figure 11:** Deep Neural Network mortality prediction model accuracy of 83% and AUC of 82% with Classification report (used in the model)

#### III.III.I.IV. T4: Statistical Analysis of Features

The final step in the pipeline involves the calculation of statistical metrics on the processed dataset. This step computes various statistical measures, such as the mean, standard deviation, and other descriptive statistics for each feature in the dataset. These statistics provide a deeper understanding of the data's distribution and can highlight key characteristics or trends. The statistical analysis is particularly useful for feature selection, model validation, and gaining insights into the dataset's overall structure.

Vital	Mean	Median	Min	Max	Std Dev
Unnamed: 0	3.0	3.0	0	6	2.16
bloc	4.29	4.0	1.0	9.0	2.69
icustayid	200003.0	200003.0	200003.0	200003.0	0.0
gender	0.0	0.0	0.0	0.0	0.0
age	17639.83	17639.83	17639.83	17639.83	0.0
elixhauser	0.0	0.0	0.0	0.0	0.0
re_admission	0.0	0.0	0.0	0.0	0.0
Weight_kg	79.24	79.5	78.7	79.7	0.51
GCS	15.0	15.0	15.0	15.0	0.0
HR	76.7	77.0	68.33	89.8	7.02
SysBP	114.28	111.0	103.83	137.8	11.44
MeanBP	76.71	75.2	70.33	86.6	5.38
DiaBP	60.25	60.2	55.33	66.6	4.27

**Figure 12:** Statistical analysis as part of the report pdf downloaded.

 **Generate Patient Summary**

Generate Report with Gemini

✓ Summary Report

Patient 200003.0 presents with several concerning findings requiring immediate attention. Here's a breakdown:


**Demographics and Background:**

- **Age:** The age value of 17639.8 seems erroneous and likely represents a data entry error. Clarification is crucial. Assuming a reasonable adult age is intended, the analysis below still holds, but age plays a significant factor in prognosis and treatment.
- **Gender:** 0.0 typically indicates male.
- **Weight:** 79.5 kg, which falls within a reasonable adult weight range.

**Vital Signs and Critical Findings:**

- **GCS:** 15 - Normal, indicating full consciousness.
- **Heart Rate (HR):** 78 bpm - Within normal range.
- **Blood Pressure (SysBP/MeanBP/DiaBP):** 117.4/79.4/60.2 mmHg - While systolic BP is acceptable, the diastolic and mean arterial pressures are low, suggesting potential hypotension. This warrants close monitoring and investigation.

**Critical Note:** This analysis is based on the provided data and assumes a reasonable adult age. The actual age, as well as a detailed history and physical examination, are essential for a definitive diagnosis and management plan. This information should be used to inform clinical decision-making in conjunction with the judgment of experienced medical professionals.

 Download Detailed PDF Report

**Figure 13:** Patient Summary with downloadable detailed report using gemini model.

### III.IV. Challenges and Refinements

Throughout the development process, several challenges emerged. One key issue was managing the complexity of integrating multiple external APIs and ensuring that the agent could dynamically adjust to new data without losing context. This was addressed by customizing

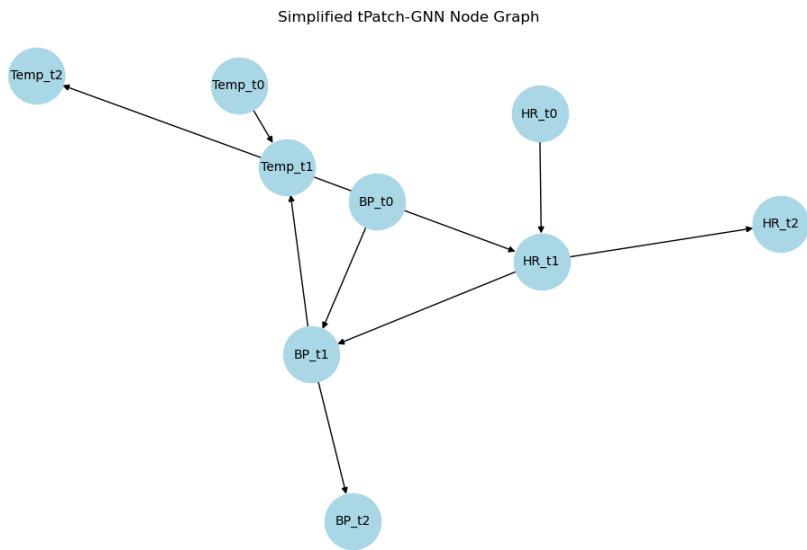
LangChain’s agent framework to handle various inputs and switching between different tasks efficiently.

Another challenge involved fine-tuning the agent’s natural language generation capabilities to produce coherent and relevant responses. Iterative testing and adjustments were made to improve response quality, leveraging LangChain’s built-in tools for working with LLMs.

## IV. Imputation with tPatch-GNN

To address the missing data problem, I implemented **tPatch-GNN**, a Temporal Patch Graph Neural Network specifically tailored for irregular time-series imputation in clinical datasets. Each patient’s timeline is represented as a graph where nodes correspond to clinical measurements and edges encode temporal continuity and cross-feature relationships. The model learns to predict missing values by leveraging these structural dependencies.

In addition to imputation accuracy, I focused heavily on modeling uncertainty. Knowing when an imputed value is unreliable is vital in medical settings. I used a bootstrapping method involving multiple stochastic forward passes with dropout or noise injection to estimate prediction variance. The agent could then qualify its responses with statements like “this value is estimated with low confidence,” helping clinicians gauge when to rely on the AI system and when to double-check with further testing.

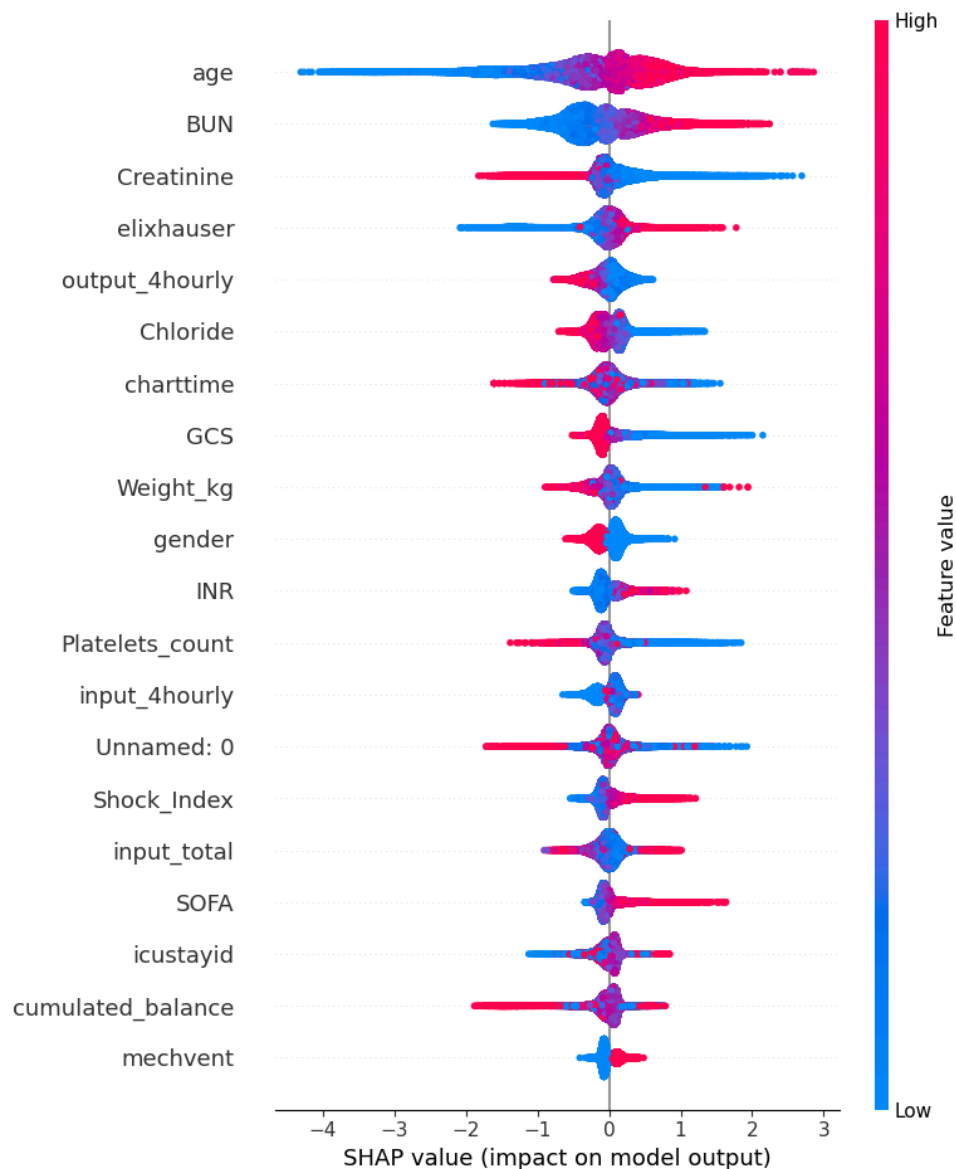


**Figure 14:** Explains model architecture visually; helps clarify the advanced imputation approach

## V. Mortality Prediction and Interpretability

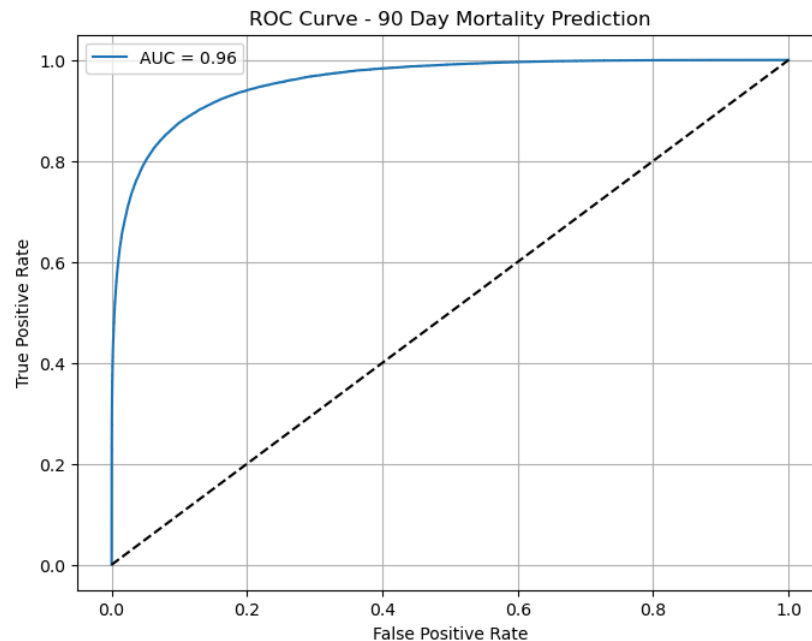
After completing the imputation stage, I trained a machine learning model to predict 90-day mortality risk. I evaluated both classical and deep models—starting with logistic regression and XGBoost, then experimenting with LSTM-based time-series models. Although deep models captured long-range dependencies, they often required much more tuning and offered limited interpretability. XGBoost struck the right balance between performance and transparency.

For interpretability, I used **SHAP (SHapley Additive exPlanations)** values to identify which features contributed most to each prediction. This allowed me to trace the reasoning behind each output and represent it in both numerical and natural language form. For instance, if the model predicted high mortality risk, the agent might explain, “This is largely due to elevated lactate levels and decreased platelet count over the last 24 hours.”



**Figure 15:** SHAP Summary Plot to show which features contributed most to predictions

This feature-level attribution was critical in translating the model's complex internal logic into actionable clinical insights. I wrapped this explanation mechanism as a callable tool for the LLM agent.



**Figure 16:** Supports evaluation of the model and shows how well it performs

## VI. Exploration of AI Agent Frameworks

Building a reliable interface to tie together these tools into a coherent, interactive system required experimentation with several AI agent frameworks. I began with **OpenManus**, a promising system that allows users to build LLM agents with local file access and basic function calling. While simple to use, it lacked modularity and had limited support for dynamic tool composition.

I then tested **Ollama**, which offers a lightweight platform for running LLMs locally. Although Ollama's performance was impressive for some tasks, it provided limited extensibility for chaining tool invocations or orchestrating memory-aware interactions across multiple models.

Next, I explored **Superagent**, an open-source orchestration framework designed for modular LLM agents. Despite its extensive feature set and a user-friendly web interface, Superagent still lacked the flexibility I needed to integrate custom models (like tPatch-GNN or SHAP) as tools that could interact seamlessly with the LLM.

Finally, I reviewed several community projects from the **Awesome-AI-Agents** repository. These showcased clever ways to build agents that browse the web, read PDFs, or access APIs,



but most examples were not suited to the kind of real-time, medically sensitive, model-aware architecture I required.

After this exploration, I chose **Langchain**, which offered the most robust and adaptable solution. Langchain's ecosystem allowed me to define custom tools, manage conversational memory, and orchestrate interactions between multiple components (like a SHAP interpreter, a mortality predictor, and a data retrieval function). I wrapped each model as a Langchain tool, registered it in the agent's toolkit, and used Gemini Pro to power the natural language interaction.

## VII. Final Agent Design and System Architecture

The completed agent is a modular system comprising four primary components:

1. **Imputation Module (tPatch-GNN):** Fills in missing clinical values while estimating uncertainty.
2. **Prediction Module (XGBoost + SHAP):** Computes the patient's mortality risk and explains model decisions.
3. **Tool Wrapper Layer:** Converts model outputs into callable tools accessible by the LLM agent.
4. **Langchain-based LLM Agent:** Interprets clinician queries, invokes relevant tools, and generates responses in natural language.

This entire pipeline was deployed via a lightweight Gradio interface that simulates a clinician interface. Users can upload a patient record, receive an overview of imputed data, view risk predictions, and converse with the assistant to clarify results. Sample queries include:

- "How likely is this patient to die in 90 days?"
- "Which lab tests contributed most to this prediction?"
- "Are any of the imputed values unreliable?"
- "Summarize the key abnormalities over the past 48 hours."

Langchain's built-in memory and context tracking ensured that the agent could maintain coherent conversations across multiple turns—a key requirement for real-world clinical usage.

### User Interaction & Language Understanding

A key feature of our AI agent is its natural language interface, designed to support flexible and intuitive interactions with users. The agent is capable of understanding a wide range of prompt styles, including colloquial questions, formal clinical queries, and shorthand instructions (e.g., "show patients with high lactate" or "filter ICU > 3 days"). Moreover, when information is insufficient or ambiguous, the agent actively prompts the user for clarification, such as asking, "Which column would you like to filter on?" or "Do you want imputation on missing vitals or

labs?” This makes the system both more robust and user-friendly. Future improvements may involve adding multilingual support and dynamic conversation history tracking to provide even more seamless and intelligent interactions.

## VIII. Discussion and Reflections

This project represents a successful proof of concept for AI-augmented clinical reasoning under uncertainty. By integrating domain-specific models with an LLM-based interface, I created an agent that could not only perform predictive tasks but also justify its logic and limitations. One of the major takeaways from this project was how essential modularity is when working with specialized tools. Most off-the-shelf agents are designed for general-purpose use and don’t offer the tight coupling needed between domain models and language interfaces.

Moreover, building trust in medical AI requires more than just accuracy—it demands transparency, interpretability, and humility. The ability of the agent to say “this value is estimated with low confidence” or “this conclusion is based on partial data” is just as important as making correct predictions.

From a technical perspective, Langchain proved to be the right choice due to its flexible abstractions and vibrant community. It allowed me to iterate quickly, add new tools as needed, and maintain logical control over how the agent responded to complex questions.

## IX. Future Work

While the current version of our AI agent provides a functional and interactive interface for exploring and analyzing sepsis-related EHR data, there are several important avenues for future enhancement:

1. **Advanced Forecasting and Modeling Techniques:**

We aim to expand our integration of state-of-the-art time series models, particularly those tailored to irregular and sparse clinical data. Building on the TPA-GNN framework, future versions may incorporate hybrid architectures that combine temporal attention with graph-based patient representations to more accurately forecast vitals, lab values, or sepsis onset in real-world, noisy datasets.

2. **Clinical Validation and Expert Feedback:**

While the agent has been developed with general clinical logic in mind, true medical utility can only be confirmed through collaboration with healthcare professionals. We plan to conduct user studies involving clinicians and medical students to evaluate usability, clinical relevance, and diagnostic utility. Feedback from these sessions will guide further refinements, especially regarding interpretability and user experience.

3. **Support for Multimodal and Longitudinal Data:**

Currently, the system primarily operates on structured tabular data. In the future, we hope to extend support to multimodal inputs including clinical notes, imaging metadata,

or waveforms. Additionally, we will improve handling of longitudinal patient records over multiple admissions or care episodes, enabling richer patient trajectory modeling.

4. **Personalized User Profiles and Context Awareness:**

We envision future versions of the agent that adapt to individual users—whether researchers, clinicians, or students—by remembering previous sessions, preferences, and clinical context. This could allow the agent to proactively surface relevant tools or suggest analyses based on prior usage patterns or ongoing investigations.

5. **Multilingual and Cross-Cultural Adaptation:**

To improve accessibility across different regions and populations, future iterations may include support for multilingual interactions and localization of clinical terminology. This can make the agent more inclusive and practical for global healthcare research collaborations.

In summary, the AI agent serves as a foundation for intelligent, interactive EHR analysis. With the outlined future enhancements, we hope to transition the system from a research prototype to a clinically valuable decision support tool.

## X. Conclusion

The AI agent developed in this project offers a novel integration of imputation, prediction, and natural language explanation to support clinicians working with sparse EHR data. The combination of graph neural networks, SHAP-based interpretability, and a Langchain-powered interface demonstrates how adaptive systems can provide not just answers but insight and justification. This approach has the potential to augment medical workflows with trustworthy, explainable support—especially in environments where data is incomplete or noisy.

While the system is currently a research prototype, it opens the door to future work in clinical validation, cross-institutional generalization, and deployment in real hospital settings. More broadly, this project illustrates how emerging AI agent ecosystems can be adapted to high-stakes domains, provided they are implemented with careful attention to robustness, uncertainty, and human-centered design.

## XI. Acknowledgments

I would like to sincerely thank Professor Pengyu Hong for his guidance, encouragement, and invaluable feedback throughout the course of this project. His insights into adaptive systems and emphasis on rigor shaped many of the ideas and methodologies explored here. I'm also grateful to Zepeng Hu and Jiarui Zhang for their technical input, thoughtful discussions, and support during the design and implementation phases—especially around model evaluation and agent architecture. Their contributions helped me refine key components of the system. I also extend my appreciation to my classmates in COSI 217A for the stimulating conversations and references shared during our seminars; these exchanges played a meaningful role in broadening the scope of this work and pushing it in new directions.

## XII. References

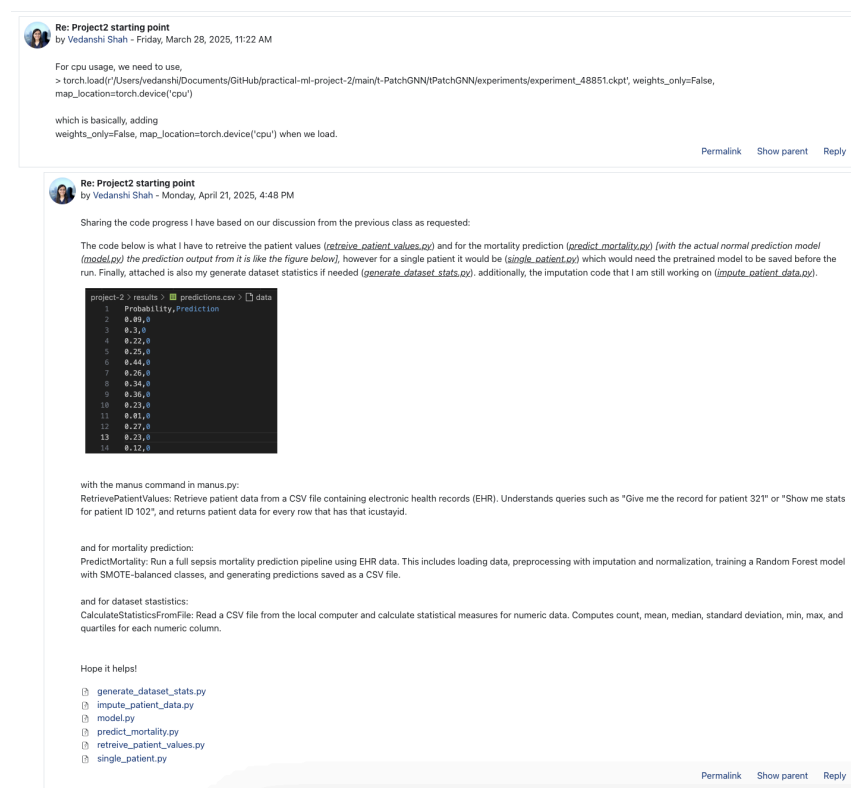
1. **Johnson, A. E. W., Pollard, T. J., Shen, L., et al. (2016).**  
*MIMIC-III, a freely accessible critical care database.*  
Scientific Data, 3, 160035. <https://doi.org/10.1038/sdata.2016.35>
2. **Singer, M., Deutschman, C. S., Seymour, C. W., et al. (2016).**  
*The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3).*  
JAMA, 315(8), 801–810. <https://doi.org/10.1001/jama.2016.0287>
3. **Desautels, T., Calvert, J., Hoffman, J., et al. (2016).**  
*Prediction of sepsis in the ICU using machine learning and physiological data.*  
BioMed Research International, 2016. <https://doi.org/10.1155/2016/9308692>
4. **Shickel, B., Tighe, P. J., Bihorac, A., & Rashidi, P. (2017).**  
*Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis.*  
IEEE Journal of Biomedical and Health Informatics, 22(5), 1589–1604.  
<https://doi.org/10.1109/JBHI.2017.2767063>
5. **Johnson, A. E. W., Ghassemi, M. M., Nemati, S., et al. (2017).**  
*Machine learning and decision support in critical care.*  
Proceedings of the IEEE, 104(2), 444–466.  
<https://doi.org/10.1109/JPROC.2015.2501978>
6. **Futoma, J., Morris, J., & Lucas, J. (2015).**  
*A comparison of models for predicting early hospital readmissions.*  
Journal of Biomedical Informatics, 56, 229–238.  
<https://doi.org/10.1016/j.jbi.2015.05.016>
7. **Goldstein, B. A., Navar, A. M., Pencina, M. J., & Ioannidis, J. P. A. (2017).**  
*Opportunities and challenges in developing risk prediction models with electronic health records data: A systematic review.*  
Journal of the American Medical Informatics Association, 24(1), 198–208.  
<https://doi.org/10.1093/jamia/ocw042>
8. **Beam, A. L., & Kohane, I. S. (2018).**  
*Big data and machine learning in health care.*  
JAMA, 319(13), 1317–1318. <https://doi.org/10.1001/jama.2017.18391>
9. **Rajkomar, A., Dean, J., & Kohane, I. (2019).**  
*Machine learning in medicine.*  
New England Journal of Medicine, 380, 1347–1358.  
<https://doi.org/10.1056/NEJMr1814259>

10. **Nemati, S., Holder, A., Razmi, F., et al. (2018).**  
*An Interpretable Machine Learning Model for Accurate Prediction of Sepsis in the ICU.*  
Critical Care Medicine, 46(4), 547–553.  
<https://doi.org/10.1097/CCM.0000000000002936>
11. **Zhao, Y., Liu, C., Zhang, H., et al. (2024).**  
*Irregular Multivariate Time Series Forecasting: A Transformable Patching Graph Neural Networks Approach.*  
Proceedings of the 41st International Conference on Machine Learning (ICML 2024).  
GitHub: <https://github.com/usail-hkust/t-PatchGNN>
12. **Manna and Poem Lab. (2024).**  
*OpenManus: Modular AI Agent Framework for Clinical and Document Analysis.*  
GitHub Repository. <https://github.com/mannaandpoem/OpenManus>

## XI. Appendix

Github project link: <https://github.com/VedanshiShah7/practical-ml-project-2>

Try out the live website: <https://practical-ml-project-2.streamlit.app/>



**Figure 17:** Contribution to moodle group discussion along with in class discussions

<https://moodle.brandeis.edu/mod/forum/discuss.php?d=42018>

# Patient Viewer + Chatbot Assistant

## Patient Lookup

Select a patient icustayid

200003.0

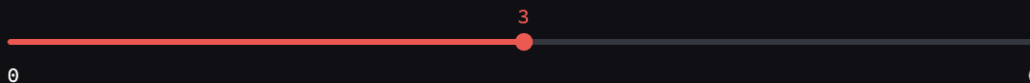


Showing data for patient 200003.0

	Unnamed: 0	bloc	icustayid	charttime	gender	age	elixhauser	re_admission
0	0	1	200,003	1970-01-01 00:00:07	0	17,639.8264	0	0
1	1	2	200,003	1970-01-01 00:00:07	0	17,639.8264	0	0
2	2	3	200,003	1970-01-01 00:00:07	0	17,639.8264	0	0
3	3	4	200,003	1970-01-01 00:00:07	0	17,639.8264	0	0
4	4	5	200,003	1970-01-01 00:00:07	0	17,639.8264	0	0
5	5	6	200,003	1970-01-01 00:00:07	0	17,639.8264	0	0
6	6	9	200,003	1970-01-01 00:00:07	0	17,639.8264	0	0

## Patient Timeline

Select timepoint



Selected Time: 1970-01-01 00:00:07.245529200

## Vital Stats at this Time

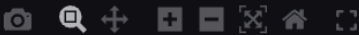
	Value
Unnamed: 0	3.0000
bloc	4.0000
icustayid	200,003.0000
gender	0.0000
age	17,639.8264
elixhauser	0.0000
re_admission	0.0000
Weight_kg	79.5000
GCS	15.0000
HR	78.0000
SysBP	117.4000
MeanBP	79.4000
DiaBP	60.2000
RR	18.8000
SpO2	97.2000
Temp_C	37.7571
FiO2_1	0.5000
Potassium	3.8000
Sodium	144.0000
Chloride	111.0000
Glucose	94.2000
BUN	11.0000
Creatinine	0.7000
Magnesium	2.1000

Magnesium	2.1000
Calcium	7.5000
Ionised_Ca	1.0500
CO2_mEqL	28.0000
SGOT	38.0000
SGPT	30.0000
Total_bili	5.2000
Albumin	2.8000
Hb	9.6000
WBC_count	14.5000
Platelets_count	116.0000
PTT	76.6000
PT	14.6000
INR	1.3000
Arterial_pH	7.5000
paO2	84.0000
paCO2	38.0000
Arterial_BE	5.0000
HCO3	28.0000
Arterial_lactate	0.9000
mechvent	0.0000
Shock_Index	0.6644
PaO2_FiO2	168.0000
median_dose_vaso	0.0000
max_dose_vaso	0.0000
input_total	6,677.0000
input_4hourly	50.0000

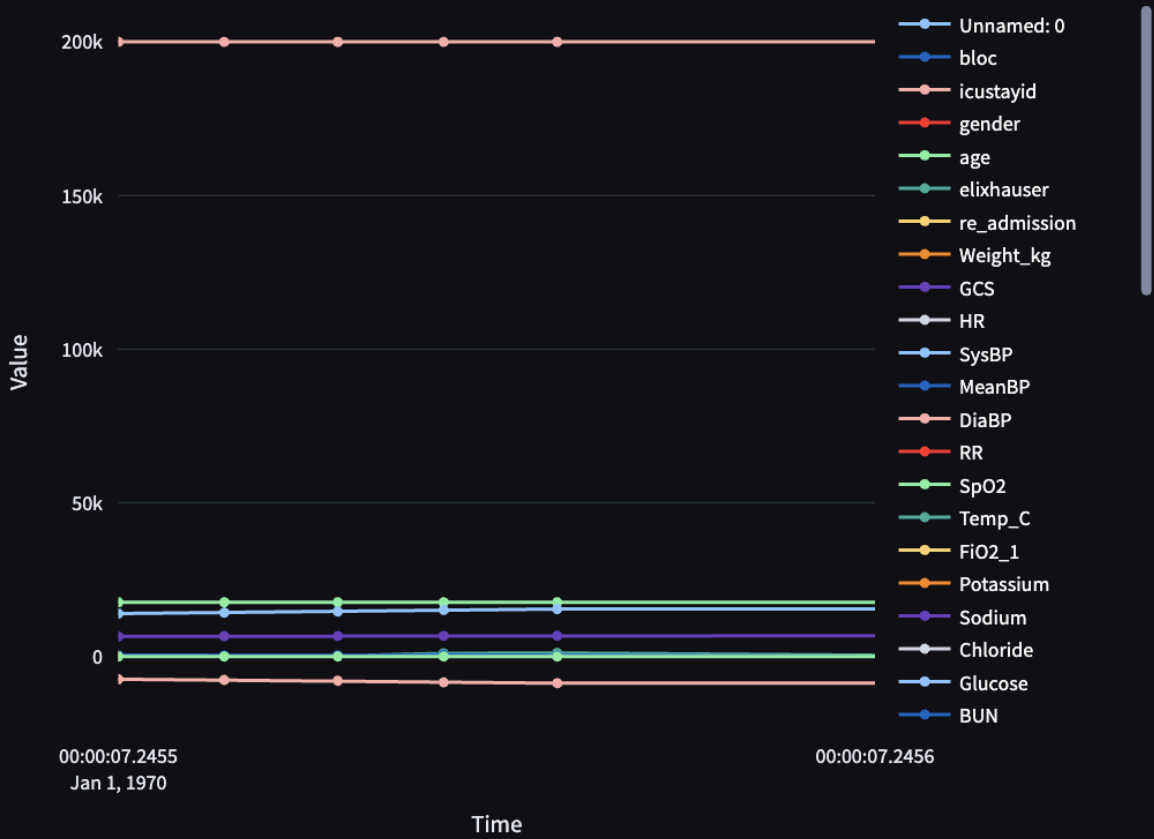


output_total	14,602.0000
output_4hourly	340.0000
cumulated_balance	-7,925.0000
SOFA	6.0000
SIRS	1.0000

## Trends Over Time (All Vitals - Interactive)



Vital Sign Trends Over Time





## Generate Patient Summary

Generate Report with Gemini



## Ask about this patient

Type your question here:

give me the mortality prediction for patient with id 200003 in the file `"/Users/vedanshi/Documents/GitHub/practical-ml-project-2/main/data/test.csv"`

Send



## Conversation History



**You:** run imputation model for patient with id 200003 from the file `"/Users/vedanshi/Documents/GitHub/practical-ml-project-2/main/data/test.csv"` using the preloaded imputation model from path `"/Users/vedanshi/Documents/GitHub/practical-ml-project-2/main/OpenManus-main/app/tool/tPatchGNN/tPatchGNN/experiment_48851.ckpt"`



**Agent:** Imputation has been completed. The imputed data is saved in the ``imputed_batches`` folder. A preview of the first batch is also included in the output.



**You:** give me the mortality prediction for patient with id 200003 in the file `"/Users/vedanshi/Documents/GitHub/practical-ml-project-2/main/data/test.csv"`



**Agent:** Low risk of mortality for icustayid 200003. Average Probability of Mortality: 3.50% Continue regular monitoring.

**Figure 18:** Website Preview