AI Agent for Mortality Prediction Using EHR Data

Parthiv Ganguly

May 6, 2025

GitHub Link

https://github.com/parthiv2048/AI-Agent-For-Mortality-Prediction

Abstract

This project presents the development of an AI agent designed to predict patient mortality within 90 days based on Electronic Health Records (EHR). The system incorporates data preprocessing, missing value imputation, machine learning models, tool calling, and interpretability techniques to enhance decision-making support in clinical settings. The model primarily leverages Random Forest for classification and SHAP (SHapley Additive exPlanations) for explainability. Furthermore, the predictive results are translated into natural language using a large language model (LLM), offering clinicians an intuitive understanding of patient risk profiles. The final product is an AI agent accessible through natural language queries, implemented using Langchain and Google's Gemini-2.0.

1 Introduction

Electronic Health Records (EHR) are an essential component of modern healthcare, capturing detailed, time-series data on patient vitals, lab tests, medications, and clinical observations. Mining such data for insights, especially for early mortality prediction, can significantly impact patient outcomes and resource planning. Predictive modeling of EHR data, however, presents several challenges, including missing values, irregular sampling, and class imbalance.

This project addresses these challenges by developing a comprehensive pipeline that includes:

- Cleaning and preprocessing of time-series EHR data.
- Imputation strategies to handle missing values.
- Machine learning models such as Random Forest [1] and XGBoost [2] for prediction.
- SMOTE (Synthetic Minority Over-sampling Technique) [3] for class balancing.

- SHAP [4] for model interpretability.
- Tool Calling by AI Agent using Langchain
- Natural language generation for human-readable explanations

The outcome is an intelligent agent capable of interpreting queries, retrieving relevant patient data, and providing evidence-based, explainable mortality predictions.

2 Data Preparation

2.1 Initial Pre-Processing

Patient data was grouped using the `icustayid`, a unique identifier for each ICU stay which effectively acts an unique id for each patient. Within each group, rows were ordered by `charttime` to maintain the temporal sequence. There are 52 features available for the model to train on, all of which are of the data type: float.

2.2 Missing Value Identification and Imputation

Certain features used 0 to indicate missing values. These were identified and replaced with NaN to facilitate appropriate imputation. Three datasets were constructed:

1. **Non-imputed dataset**: Original dataset with NaNs.
2. **Strategy 1 Imputed Dataset**:
    - Group Forward Fill
    - If backwards entry is NaN: Group Backward Fill
    - If both surrounding entries are NaN: Group Mean
    - If all values in group are NaN: Global Mean
3. **Strategy 2 Imputed Dataset**: Linear Interpolation for each feature.

3 Feature Summarization

Each `icustayid` group was compressed into a single row. For each feature, the following statistical measures were calculated:

- Mean
- Standard Deviation
- Minimum
- Maximum
- Last Recorded Value

4 Model Training and Evaluation

4.1 Modeling

Two machine learning models were employed:

- Random Forest Classifier [1]
  - Number of Estimators = 100
  - Max Tree Depth = 10
  - Class Weight = Balanced
- XGBoost Classifier [2]
  - Number of Estimators = 100
  - Max Tree Depth = 10
  - Learning Rate = 0.1
  - Evaluation Metric = Log Loss

4.2 Handling Class Imbalance

Due to the imbalanced nature of the class labels (mortality within 90 days), Synthetic Minority Over-sampling Technique (SMOTE) [3] was applied to improve F1-scores for the positive class.

4.3 Model Evaluation

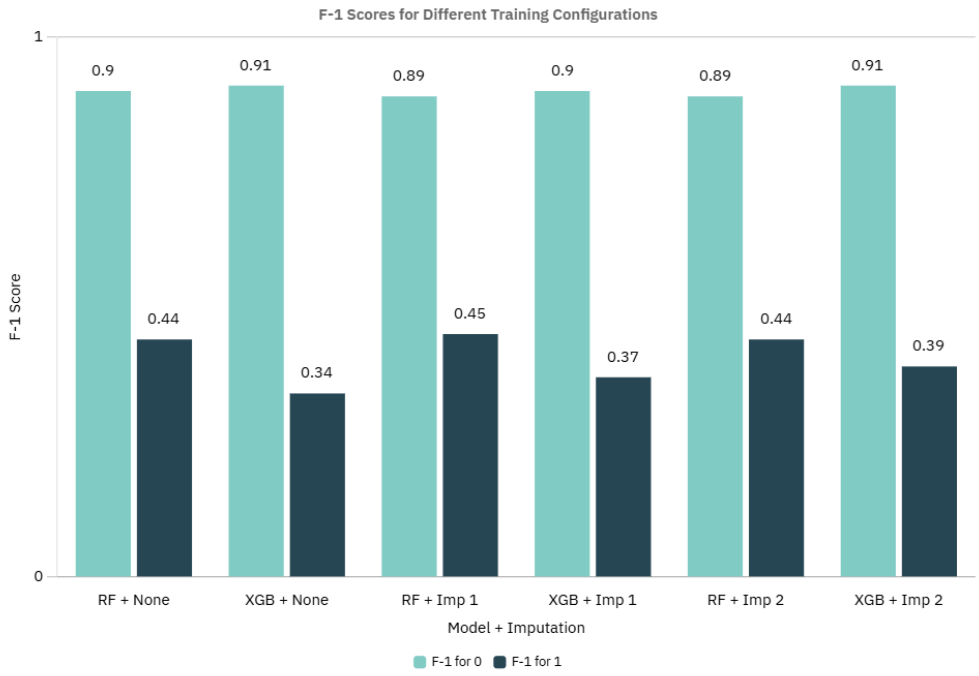

Figure 1: X-Axis represents the different training configurations, where RF stands for Random Forest, XGB stands for XGBoost, Imp 1 stands for Strategy 1 Imputed Dataset as described in Section 2.2, and Imp 2 stands for Strategy 2 Imputed Dataset. The light green bars on the left are F-1 Scores for mortality = 0 i.e. the negative class label. The dark green bars on the right are F-1 Scores for mortality = 1 i.e. the positive class label.

All the different training configurations produce similar results when it comes to the F-1 Scores. RF + Imp 1 has the best F-1 Score for the positive class label, and XGB + Imp 2 has the best F-1 Score for the negative class label. Overall, the best performance was observed using Random Forest on the Strategy 1 imputed dataset. This model was selected for final predictions.

5 Prediction and Explainability

The selected model was applied to the test dataset without class labels to generate mortality predictions. Feature importance was determined using SHAP (SHapley Additive exPlanations) [4].

5.1 SHAP Summary Plot for General Model

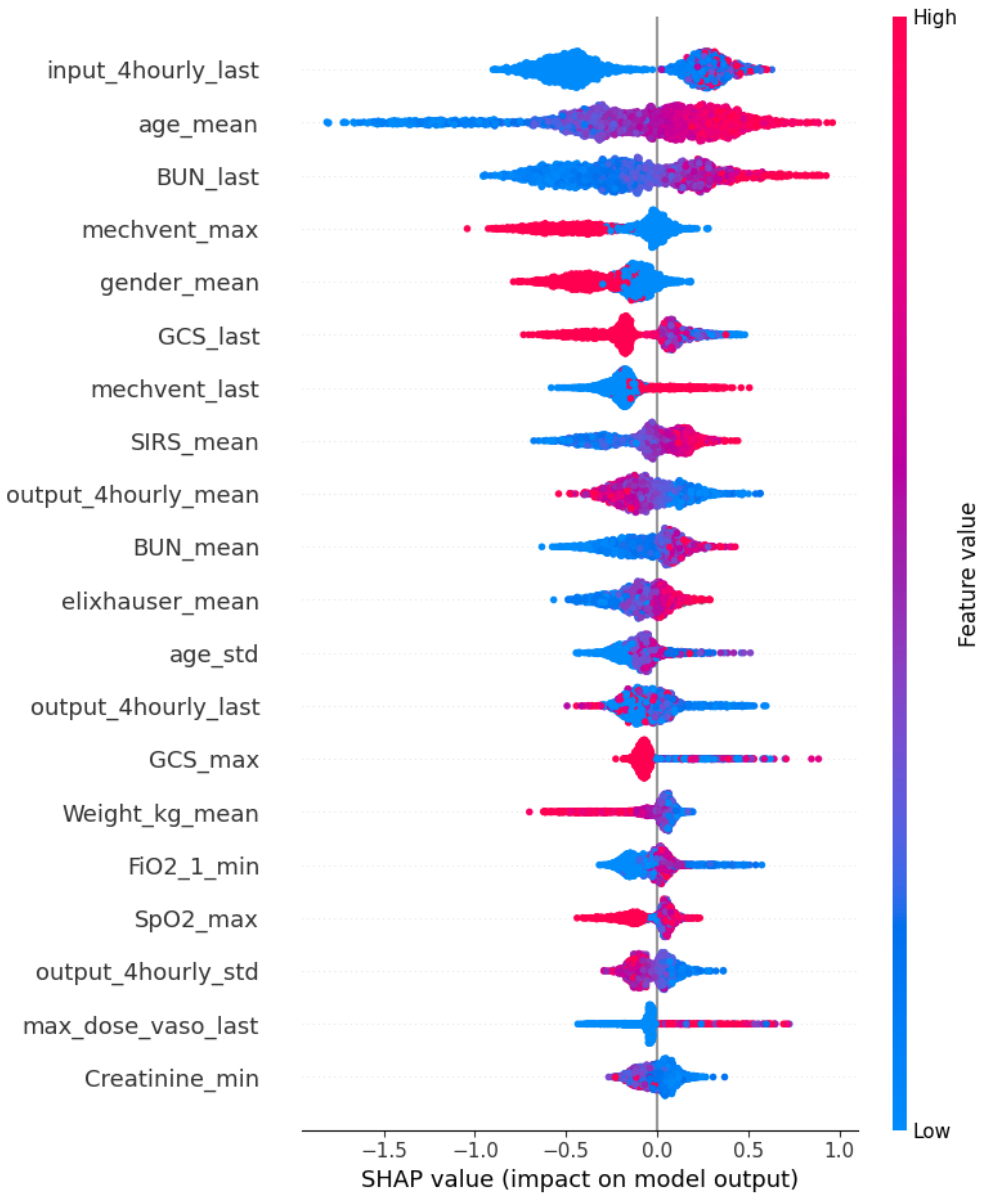

Figure 2: SHAP Summary Plot showing global feature importance.

5.2 SHAP Force Plot for Example Patient

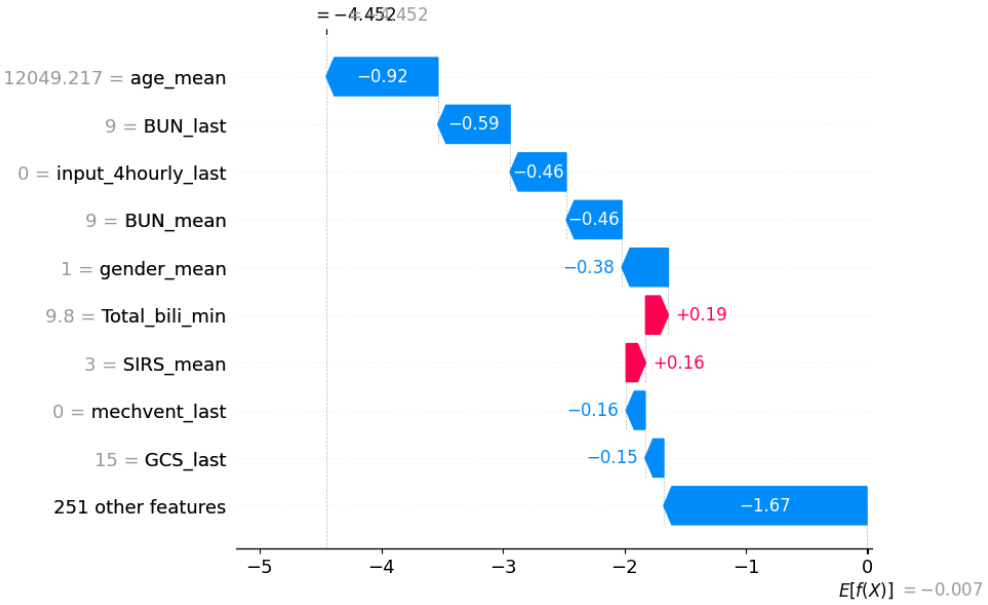

Figure 3: SHAP Force Plot explaining individual prediction for a single patient.

6 Creating Agent using Langchain LLM

6.1 Functionality

Two core functions (tools) were developed:

1. Fetch rows corresponding to a specific `icustayid`.
2. Predict patient mortality and provide:
   - Raw mortality prediction
   - Mortality prediction after imputation
   - Top 10 most influential features (according to SHAP values)
   - Explain all this data in natural language using a LLM

6.2 Langchain and VertexAI Integration

These functions were wrapped as tools and connected to a Langchain agent powered by Gemini-2.0 (via Google VertexAI). This enabled natural language queries to invoke these tools and receive comprehensible explanations.

7 Future Work

The F-1 Scores for the positive class label were very low (0.45), especially for medical applications. There are many steps that can be taken to improve the mortality prediction model

- Use more advanced missing value imputation techniques like ML-based imputation models
- Impute missing values using t-Patch GNN for irregular time-series forecasting
- Modify feature extraction to include additional statistics beyond mean, std, etc.
- Use feature selection to remove unnecessary features
- Experiment with other ML models or Neural Networks like the LSTM architecture

On the AI Agent side, we can try building other tools to add even more interaction options between the agent and the user.

8 Conclusion

This project demonstrates a complete pipeline to build an AI agent for patient mortality prediction using EHR data. It incorporates preprocessing, imputation, model training, explainability, and natural language interaction. The system can assist clinicians by providing interpretable predictions, thus enhancing transparency and trust in AI-assisted healthcare.

References

[1] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5-32, 2001.

[2] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 785-794.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, vol. 16, pp. 321-357, 2002.

[4] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in neural information processing systems, 2017, pp. 4765–4774.