MIMIC-III Mortality Prediction – Summary

Arun Krishnasamy

Objective

The goal of this task was to predict whether a patient would die during their hospital stay using data from the MIMIC-III dataset. My approach was to start simple and gradually improve the model step-by-step using meaningful features and smarter models.

Data Review

I began by loading the training and test data, and then joined the outcome column to the training set using `icustay_id`. I quickly checked for missing values using `vis_miss()`—surprisingly, there were none. Then I used `skim()` to get a quick summary of the variables, including date columns, ID fields, and diagnosis codes.

A major obstacle early on was the ICD-9 diagnosis codes. There were hundreds of unique values, which would be too much for a model. To deal with that, I wrote a helper function that converted the codes into broader medical chapters like "Infectious", "Neoplasms", and "Respiratory". This grouped the codes meaningfully and reduced the number of levels in the categorical variable.

Feature Engineering

To strengthen the model, I looked for extra clinical insights. I used the `comorbidity` package to calculate Charlson and Elixhauser scores—both are well-known ways to summarize a patient's disease burden. I also created a count of how many unique ICD codes each patient had, which served as a rough indicator of complexity.

Later, I tried something more advanced. I used `text2vec` to build GloVe embeddings for the ICD-9 codes. For each patient, I collected their diagnosis codes in order and treated them like words in a sentence. After training the embedding model, I averaged the vectors per patient to get a dense representation of their diagnoses. This gave me 50 new numerical columns that captured more subtle patterns.

Preprocessing

With all the new features added, I built a preprocessing pipeline using the `recipes` package. I started by creating a new column for age, based on the difference between date of birth and admission time. I removed unnecessary date fields and ID columns that weren't useful for prediction. Then I one-hot encoded categorical variables, imputed any missing numeric values

(just in case), and normalized all numeric predictors. This pipeline ensured that the data was clean, consistent, and ready for modeling.

Model Building

I trained a range of models to compare performance. First, I used logistic regression as a baseline—no tuning needed. Then I moved to more complex models. For Random Forest, I set the number of trees to 1500 and tuned `mtry` and `min_n`. For XGBoost, I tuned learning rate, tree depth, sample size, and `mtry`, using a maximum entropy grid.

For the neural network, I used `keras` and tuned the number of hidden units, dropout rate, and training epochs. This model was a bit slower but performed well. Once I had results from all models, I stacked the XGBoost and Neural Net models using the `stacks` package. This allowed the ensemble to learn which model to trust more in different situations.

Evaluation

Each model was evaluated using 5-fold cross-validation and ROC-AUC as the main metric. Logistic regression gave a decent starting point. Random Forest and Neural Net both performed well, but XGBoost consistently gave better results. The stacked model outperformed all individual models, achieving a final ROC-AUC around 0.95. This showed that both the feature engineering and the model combination were working.

Predictions

After selecting the best stacked model, I used it to make predictions on the test dataset. These predictions were saved to a file named `submission_stack.csv`, ready for submission or downstream use.

Reflection

This project started off with just basic features like age and diagnosis chapters. But by adding comorbidity indices, diagnosis count, and embeddings, I was able to improve performance significantly. Careful preprocessing, model tuning, and stacking pushed the results even further. What started as a simple logistic regression turned into a strong ensemble model capable of making accurate mortality predictions. Each step built on the previous one, and the end result is something I feel confident in submitting.

AI usage Acknowledgement:

I predominantly used AI for researching different methods, to improve my feature engineering strategies and efficiency of my codes. All the methods that was suggested: click here