ICU Mortality and Length of Stay Prediction using EHR Data

Table of Contents

Problem Setting

Critically ill patients in ICUs require continuous monitoring, as mortality risk and length of stay (LOS) impact patient outcomes and resource utilization. Machine learning offers a data-driven approach to enhance predictive precision, aiding early interventions and personalized care. Accurate mortality and length of stay predictions can optimize ICU resource allocation, reduce costs, and improve patient management. Additionally, identifying key predictive variables can enhance risk stratification and treatment strategies.

Problem Definition

This project focuses on the following major areas:

- **ICU Mortality Prediction**: Predicting the likelihood of in-hospital mortality based on the first 24 hours of clinical data.
- **Length of Stay Prediction:** Predicting length of ICU stays (for shorter stay periods).

OKRs (Objectives and Key Results)

- **Objective**: Develop accurate models for ICU mortality prediction.
  - **Key Result 1**: Achieve an Area Under the Curve (AUC) score of at least 0.85 for mortality prediction on the test dataset.
  - **Key Result 2**: Identify key features affecting mortality rate.
- **Objective:** Build a robust model to predict short ICU stays.
  - **Key Result 1:** Define "short stay" clearly (e.g., ≤ 7 days) and develop a regression model with MAE <= 1 day for short length of stay (LOS).
  - **Key Result 2**: Identify key features contributing to short ICU stays.

KPIs (Key Performance Indicators)

- **Model Performance Metrics:** Quantitatively assess prediction reliability using different performance metrics.
- **Feature Importance Ranking:** Identify and rank the top contributing variables using the model's internal feature importance scores.
- **AUC Score >= 0.85 on test dataset for ICU Mortality Prediction:** Indicates strong discriminatory power between survivors and non-survivors.
- **Regression Evaluation Metrics on test dataset:** Ensures less prediction error.

Data Sources

The data is taken from the publicly available **MIMIC-IV Clinical Database**, which provides detailed clinical and demographic information about ICU patients.

Data Description

The **MIMIC-IV Clinical Database** is structured into multiple tables, with key features to support this analysis. It contains about 94458 ICU admission records. The table below is a summary of some important tables in the dataset that were used in our analysis.

Table Name	**Brief Description**	**No. of Columns**	**Key contributing columns**
icustays	Primary ICU admission data and needed for LOS calculation	10	• *stay_id, subject_id, hadm_id* – Primary keys to link to other tables • *intime, outtime* – ICU admission/discharge timestamps • ***los* – Pre-calculated ICU Length of Stay**
patients	Basic patient information	5	• *gender* – Risk stratification factor • *dob, dod* – Mortality calculation
admissions	Hospital admission records	9	• *admittime, dischtime* – Length of stay • ***hospital_expire_flag* – Indicates mortality** • *ethnicity* – Social determinants
chartevents	Vital signs & bedside monitoring data.	12	• *itemid* – Measurement Type • *charttime* – Timestamp • *valuenum* – Measurement
diagnoses_icd	Comorbidities & primary diagnosis data which drive risk factors for mortality and longer LOS	6	• *icd_code* – Diagnosis code • *icd_version* – ICD -9/ICD-10
labevents	Lab test results	6	• *icd_code* – Procedure code • *icd_version* – ICD -9/ICD-10

Data Mining Tasks

a. Data Loading

As mentioned above, the data in MIMIC IV database are arranged in multiple tables. For the analysis, we selectively load essential tables like ICU stays, admissions, patients, lab events, chart events, and diagnosis data, with an emphasis on memory efficiency using only relevant columns and date parsing. To manage large-scale data, particularly the lab events file and chart events file, a chunked loading strategy is implemented, processing 500,000 records per iteration. Each chunk is appended to a list, and all chunks are concatenated into a single DataFrame.

b. Dataset Integration: Exploration and Preparation

Our next step is to merge the datasets into one file. To understand the data better we start by exploring each table separately. Using *info()* we check the column data types, non-null counts and amount of data in each table. For categorical columns, we first explored the various categories using *value_counts().* To make the categorical columns easier to work with, we grouped similar options into broader categories. For example, for column "first_careunit" under ICU table, we grouped them into broader categories like Medical ICU, Surgical ICU, Cardiac ICU, Neuro ICU, and Stepdown/Other; to make the data easier to work with and analyze. The other categorical columns which were handled similarly are: "race", "admission type", and "admission location" from admissions table.

Next, we identified and removed ICU readmissions that happened within 2 days of a previous discharge. This was done by sorting ICU stays by admission time and calculating if a patient had been in the ICU shortly before. If so, that stay was marked as readmission and excluded from our main dataset. Thus, we filtered the data to keep only the first ICU stay for each patient during a hospital admission.

For lab events, certain lab events were identified based on research from papers and domain knowledge. The dataset was merged with ICU stay information (*icustays*) based on *hadm_id* and enriched with descriptive labels from the mapping table. Events were filtered to include only those occurring within the first 24 hours of ICU admission. Subsequently, lab measurements were aggregated by computing the mean *valuenum* for each *stay_id* and *itemid*, followed by pivoting to create a wide-format table with each lab test as a separate feature. Similar processing was done with chart events data.

The diagnosis_icd table was used to extract comorbidity information by categorizing ICD codes into clinically meaningful groups. A mapping function was implemented to classify ICD-9 and ICD-10 codes into categories such as Heart Failure, Trauma, Sepsis/Septic Shock, Metastasis, and others, based on their code patterns. This categorization was applied to all diagnosis entries, and entries without a mapped clinical category were removed. A total of 341,798 categorized records were retained. Finally, a binary pivot table was created with one row per hospital admission (hadm_id) and binary indicators for the presence of each clinical condition, enabling its use as comorbidity features in downstream modeling. Table 1 shows the variable details and corresponding table in MIMIC IV database from where they were extracted.

Finally, the processed datasets were merged into one file for further analysis. We started my merging the "icustays" and "admissions" tables on both *hadm_id* and *subject_id*, followed by incorporating demographic information from the patients table via *subject_id*. Next, vital signs and lab report features were added by merging pivoted "chartevents" and "labevents" data on *stay_id*. Finally, comorbidity data from the diagnosis pivot table was merged using a left join on *hadm_id* to preserve ICU stays even when diagnosis data was missing. This sequential merging ensured alignment of clinical, demographic, and diagnostic information at the ICU stay level.

Table1: Features used in our analysis

Feature	**Description**	**Table Name**
Demographic Data		
Age	Patient's Age	patients
Gender	Patient's gender	patients
Ethnicity	Patient's race/ethnicity	patients
Patient Stay Data		
First Care Unit	Name of the ICU where the patient was first admitted during the ICU stay	icu_stays
Admission type	Admission type of the patient	admissions
Admission Location	Where the patient was admitted from	admissions
Intime	Date and time the patient was admitted	icu_stays
Outtime	Date and time the patient was discharged	icu_stays
Diagnosis on Admission and Comorbidity		
Trauma	Comorbidity Information data and common diagnosis data on admission were grouped under these categories	diagnosis_icd
Hematological diseases		
Heart failure		
Immunosuppression		
Dialysis		
Metastasis		
Sepsis/Septic shock		
Respiratory failure		
Stroke		
Acute coronary syndrome		
Cirrhosis		
Lymphoma		

AIDS		
Liver failure		
Post cardiac arrest syndrome		
Lab events		
Albumin	Albumin	labevents
ALP	Alkaline phosphatase (IU/L)	
ALT	Alanine transaminase (IU/L)	
Anion Gap	Anion Gap	
AST	Aspartate transaminase (IU/L)	
Bicarbonate	HCO3 (Bicarbonate)	
Bilirubin	Bilirubin	
BUN	Blood Urea Nitrogen (Max/Min)	
Chloride	Chloride	
Creatinine	Creatinine	
Glucose	Glucose	
Lactate	Lactate	
LDH	Lactate Dehydrogenase (LDH)	
Mg	Serum magnesium (mmol/L)	
pH	pH	
Potassium	Potassium	
Sodium	Sodium	
Urea Nitrogen	Urea Nitrogen	
Hematocrit	Hematocrit	
Hemoglobin	Hemoglobin	
MCH	Mean Corpuscular Hemoglobin	
MCHC	Mean Corpuscular Hemoglobin Concentration	
MCV	Mean Corpuscular Volume	
Platelet Count	Platelet Count	
RDW	Red Cell Distribution Width	
Red Blood Cells	Red Blood Cells (RBC)	
White Blood Cells	White Blood Cells (WBC)	
Chart Events		
Heart Rate	Heart Rate	chartevents
Respiratory Rate	Respiratory Rate (Total)	
O2 saturation	O2 saturation pulseoxymetry	
GCS eye	Glasgow Coma Scale (Eye Response)	
GCS verbal	Glasgow Coma Scale (verbal)	
GCS motor	Glasgow Coma Scale (Motor)	
PCO2	Arterial CO2 Pressure	
PO2	Arterial O2 pressure	
Temperature	Temperature in Fahrenheit	

FiO2	Fraction of Inspired Oxygen	
Outcome variables		
hospital_expire_flag	Indication for mortality	admissions
los	Length of Stay	icu_stays

c. Data Visualization

The columns representing comorbidity features are binary in nature. Below is the distribution of these binary features. Conditions such as **Respiratory failure**, **Hematological diseases**, and **Stroke** show a **higher prevalence**, with noticeably more 1s compared to others. Other features like **AIDS, Acute coronary syndrome, Cirrhosis, Dialysis, Lymphoma, and Trauma** have very low positive counts, suggesting they occur in only a **small fraction of patients**.

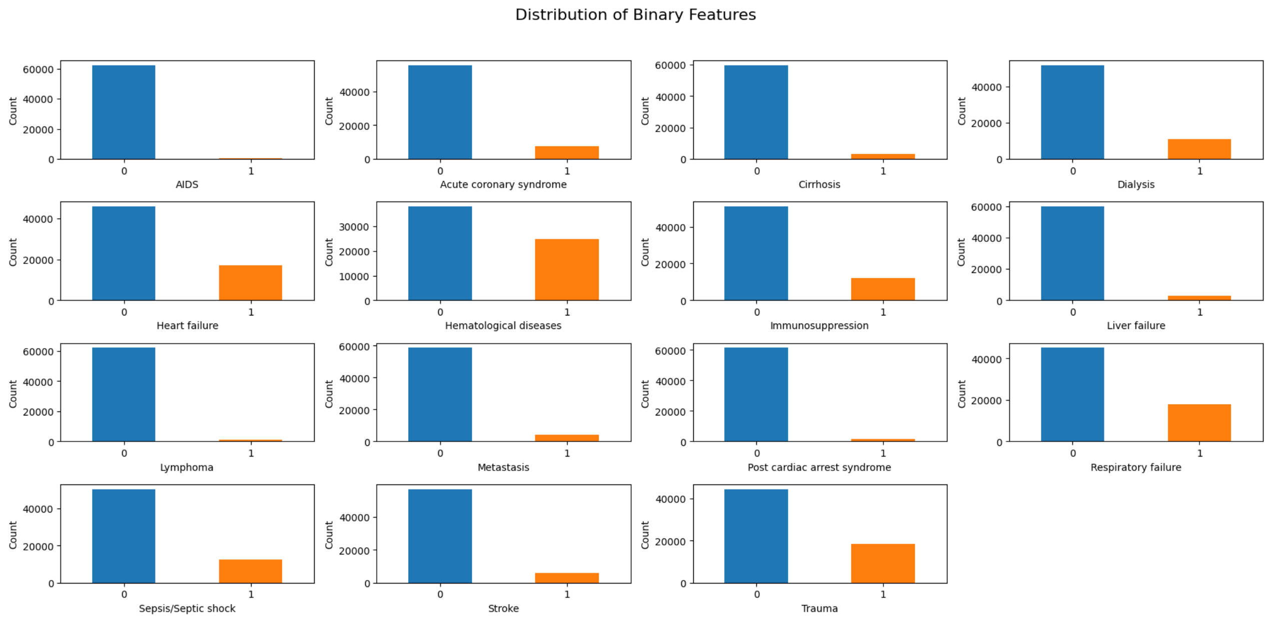


Figure 1: Visualizing the distribution of binary clinical features

We used **histogram plot** to show the frequency distribution of each **numerical variable**. They help to identify the range, central tendency, and spread of the data.

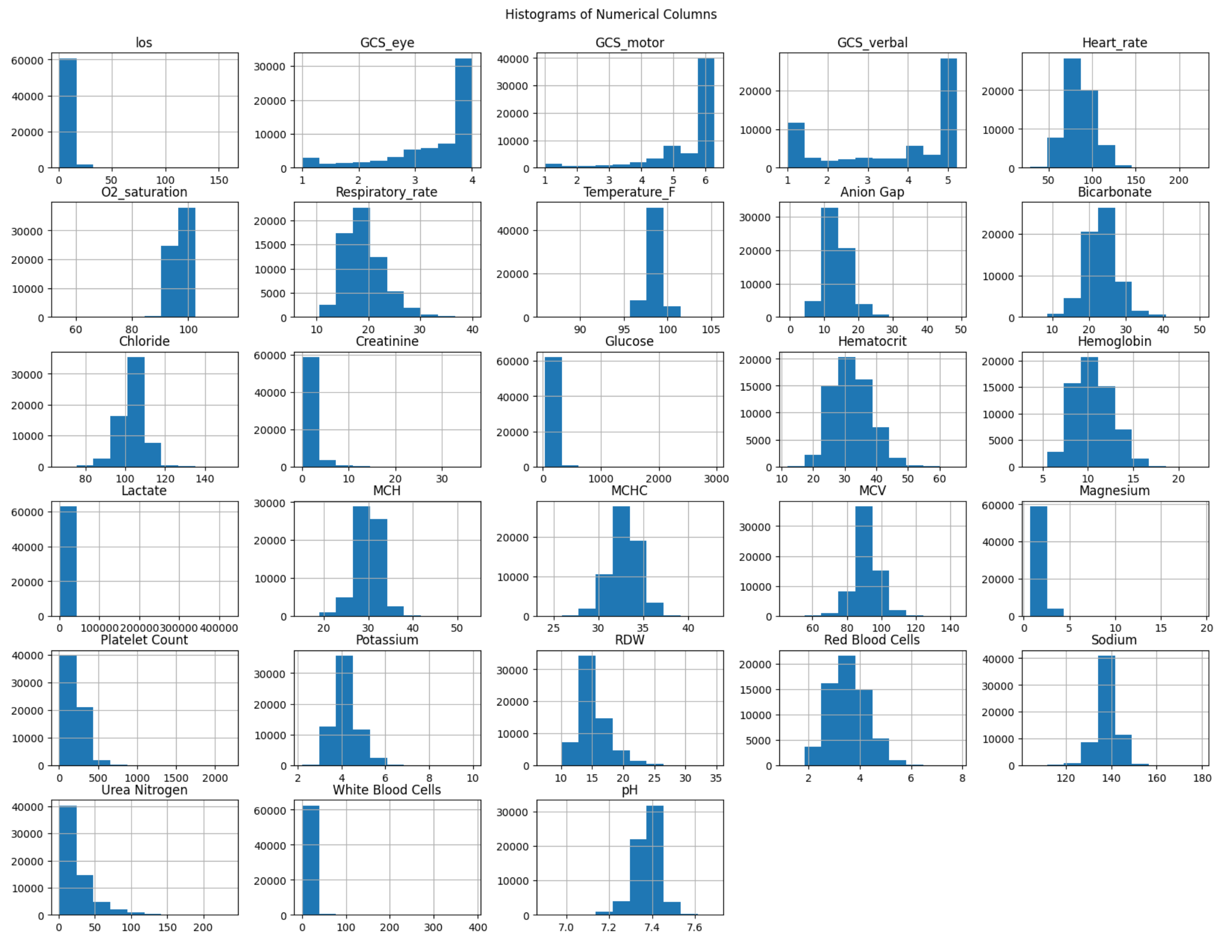


Figure 2: Visualizing the distribution of numerical features

For bivariate analysis we used stacked bar plots to show the proportion of each category within categorical variables for each class of the target variable. They help to understand how different categories are associated with the outcome. For example, we can see if certain “admission_type” or “first_careunit” have a higher proportion of expired patients.

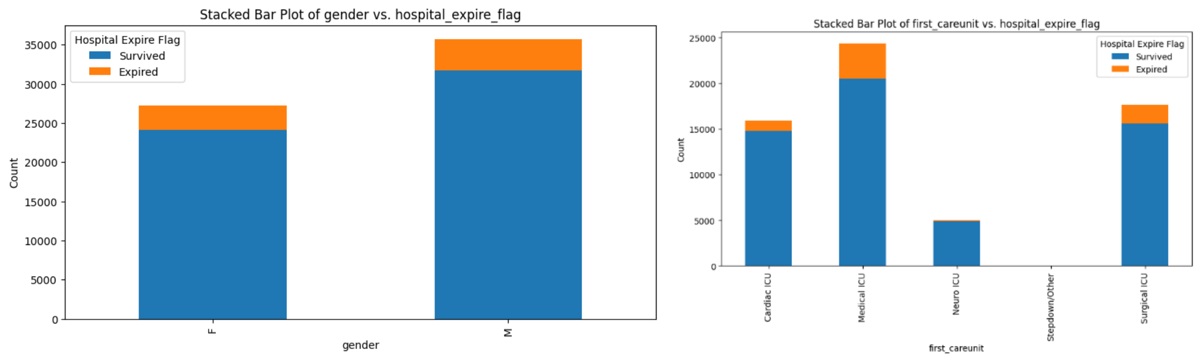


Figure 3: Visualization using Stacked Bar Plots for Bivariate Analysis

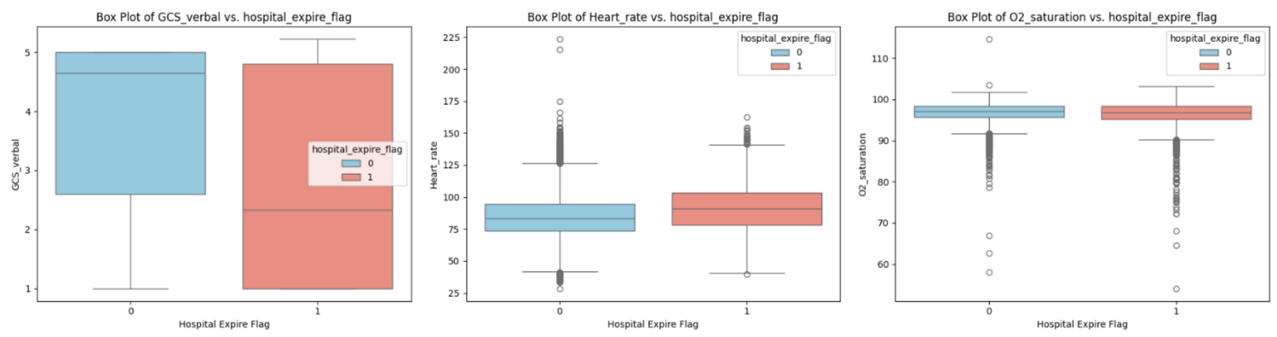


Figure 4: Visualization using Box Plots for Bivariate Analysis

d. Data Pre-processing

Below are the data pre-processing steps performed in our analysis:

- Duplicate Removal – Checked for and removed any duplicate records using "*duplicated().sum()*", although no duplicates were found.
- Filtering Based on ICU Stay Length – Retained only patients with an ICU length of stay greater than 1 day since we have used first 24 hours data from lab events and chart events to focus on more clinically significant cases.
- Data Cleaning Rules – Applied clinical plausibility filters to remove physiologically unrealistic values as below:
  - Age between 18 and 90
  - Heart rate between 25–225 bpm
  - Respiratory rate between 7–40
  - O_2 saturation between 50–120%
  - Temperature between 86–113°F
- Handling "NaN" for Binary Clinical Categories – For 18 clinical comorbidity columns (e.g., AIDS, Heart failure) missing values ("NaN") were replaced with "0".
- Dropping Non-Predictive Identifiers – Removed columns like "subject_id", "hadm_id", "stay_id", "intime", "outtime", and "deathtime", which are not helpful predictors for further analysis.
- Handling Missing Values – Identified columns with missing values and dropped features with more than 50% missing data. Then we applied **Iterative Imputer** from "sklearn" library on all numeric columns. The categorical columns did not have any missing value.

e. Data Preparation

Finally, for modelling the "**gender**" column was mapped from **categorical ("M", "F") to binary numeric format**: "M → 0", "F → 1". The **categorical columns** were also encoded and **changed to numeric using one-hot encoding**.

f. Data Partitioning

For classification task of ICU Mortality prediction, the dataset was **split into training and testing subsets using 90:10 ratio** with a fixed random state (random_state=42) to ensure reproducibility (the training set was kept large since we are down sampling later). All features are considered as predictors except "hospital_expire_flag" which is the response variable of categorical nature. Figure 5 shows that there is a class imbalance; patients who did **not** expire in the ICU (class 0) significantly outnumbers patients who **did** expire (class 1). Machine learning models trained on this data may become biased toward predicting the majority class (no expiry), leading to poor sensitivity/recall for the minority class (actual mortalities). Hence, we used "**RandomUnderSampler"** from "**sklearn.utils.resample"** on the training set to randomly down sample the majority class to match the size of the minority class (label 1, expired). It then combines the down sampled majority class with the minority class forming a balanced training set as shown in Figure 5.

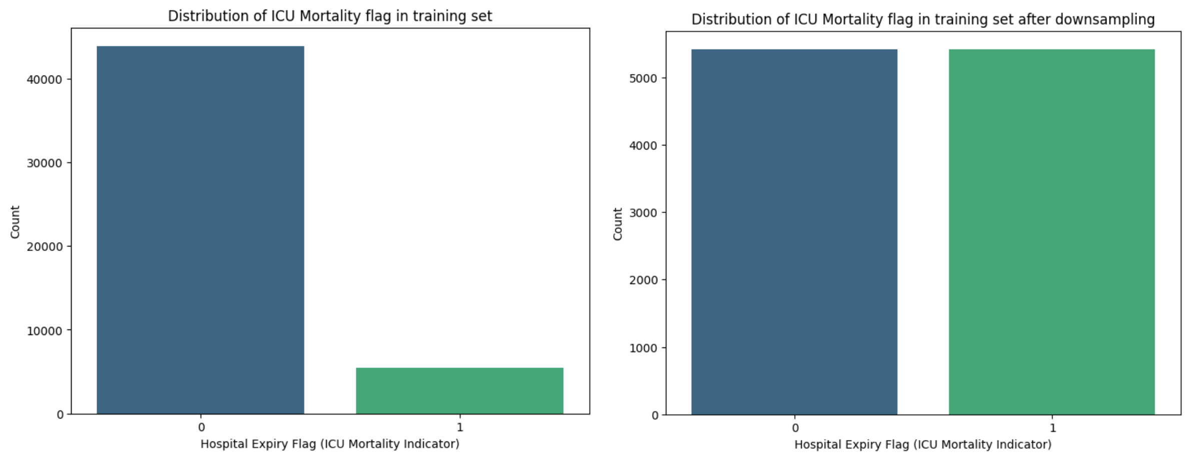


Figure 5: Distribution of Hospital Expiry flag before and after down sampling

For the regression task of ICU length of stay prediction, all features are considered as predictors except "los" which is the numeric response variable. For this project we are restricting our analysis only for short ICU stays and hence the dataset is **filtered to keep records where LOS <= 7days.** We also **excluded admissions resulting in death** as they would bias the LOS since

LOS would be shorter for this group**.** Analysing the distribution of "los" revealed that it is highly skewed and hence we decided to go with log transformation of the response variable (los). Figure 6 shows the distribution of los before and after log transformation.

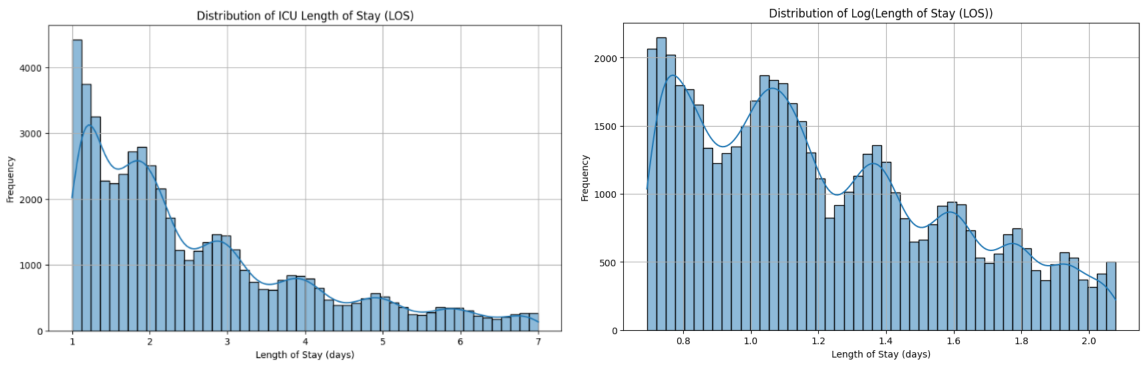


Figure 6: Distribution of Length of Stay before and after log transformation

The dataset was then split into training and testing subsets using an 80:20 ratio with a fixed random state (random_state=42) using the log transformed "y" column.

<u>Data Mining Models</u>

a. Classification Modelling for ICU Mortality Prediction

<u>Feature Selection</u>

We used feature selection technique to reduce the dimensionality by effectively choosing features that contribute more towards prediction. We used Random Forest algorithm for this classification task. A Random Forest Classifier was trained on the complete set of input features using default hyperparameters with a fixed random state. The feature importance scores were extracted from the model. These scores reflect the relative importance of each feature in contributing to the prediction of mortality. The features were ranked based on their importance scores, and a threshold of 0.01 was used to select features. The selected features, as shown in Figure 7, were then used for further modelling.

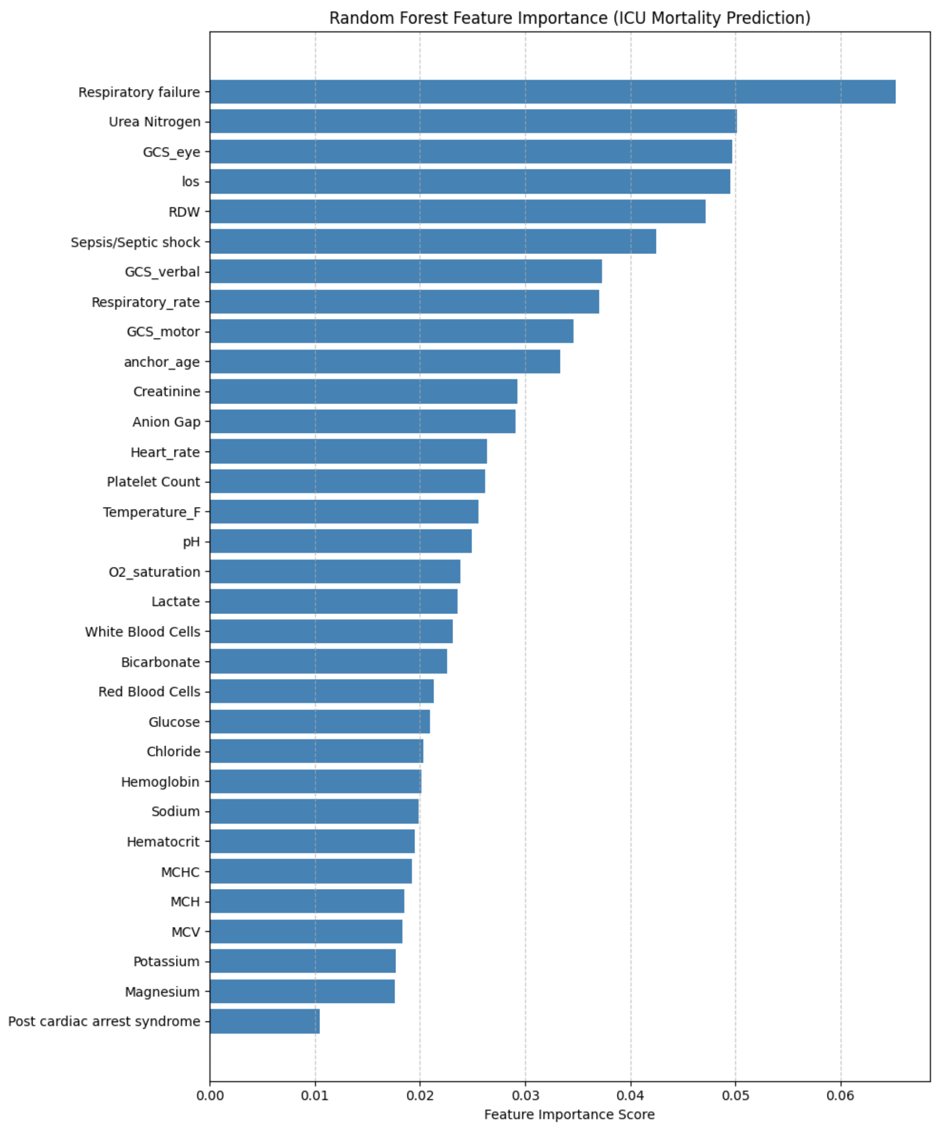


Figure 7: Feature selection using Random Forest Feature Importance feature

Modelling

For the prediction mortality risk in ICU patients, the below classification models are selected:

1. Logistic Regression

Logistic regression is a simple model used for binary classification tasks, predicting the probability of an outcome based on input features. It applies the logistic (sigmoid) function to model the relationship between independent variables and a categorical dependent variable (e.g., ICU mortality: survived or not and ICU length of stay: long or short).

Advantages:

- It is easy to implement and computationally efficient.

- It can handle multiple variables and hence can analyze multiple clinical and demographic factors.
- It provides probability estimation, so the output probabilities can help assess risk levels.

Disadvantages:

- It assumes a linear relationship between independent variables and the log odds, which may not always hold true. Thus, it struggles with non-linear dependencies in ICU stay and mortality predictions.
- It is sensitive to outliers – Performance may be affected by extreme values in clinical data.

Implementation:

For implementing Logistic Regression, hyperparameter selection is important. Hence, we used **Grid Search method** to choose the best hyperparameters, along with **5-fold cross-validation and roc-auc as the scoring metric**. For ICU Mortality prediction, the AUC achieved is 0.877 on the training set

2. Random Forest Classifier

The Random Forest (RF) Classifier is an ensemble learning method that constructs multiple decision trees during training and outputs the mode (majority vote) of their predictions. Unlike a single Decision Tree, RF reduces overfitting by averaging multiple models, making it more robust for ICU prediction tasks.

Advantages:

- Handles high-dimensional data well and is effective for both classification and regression.
- More robust to noise and overfitting compared to a single decision tree.
- Can handle missing values and unbalanced datasets effectively.

Disadvantages:

- Computationally intensive, requiring significant memory and processing power for large datasets.
- Model interpretability is lower compared to a single Decision Tree.
- Requires hyperparameter tuning for optimal performance.

Implementation:

The RF model was trained with *n_estimators=250, max_depth=10, and min_samples_split=5*, after tuning the hyperparameters using **Grid Search method**, along with **5-fold cross-validation** and **roc-auc as the scoring metric**. The prediction algorithm using RF had the highest predictive value for ICU mortality among the training cohort with AUC 0.998.

3. XGBoost Classifier

The XGBoost (Extreme Gradient Boosting) Classifier is an advanced gradient boosting algorithm known for its speed and accuracy. It works by building sequential decision trees, where each tree corrects errors made by the previous one. XGBoost is widely used in healthcare predictive modelling due to its ability to handle imbalanced data and its built-in feature selection.

Advantages:

- Highly efficient and scalable due to parallel processing.
- Can handle missing values internally without imputation.
- Provides feature importance scores, aiding interpretability in ICU prediction tasks**.**

Disadvantages:

- Requires careful hyperparameter tuning to achieve optimal performance.
- More complex and computationally demanding than Random Forest.
- Less interpretable than simpler models like Decision Trees and Logistic Regression.

Implementation:

The XGBoost classifier was trained using *n_estimators=300*, *max_depth=5*, *gamma=1, min_child_weight=5, scale_pos_weight=1, subsample=0.08* and *learning_rate=0.05*. Hyperparameter tuning was performed using **Grid Search method**, along with **5-fold cross-validation** and **roc-auc as the scoring metric**. This model demonstrated superior accuracy and precision in predicting ICU Length of Stay with AUC score of 0.961.

4. Neural Network with MLPClassifier

A Multi-Layer Perceptron (MLPClassifier) is a type of feedforward artificial neural network used for classification tasks. It consists of an input layer, one or more hidden layers, and an output layer, where each neuron applies an activation function (commonly ReLU or sigmoid) to learn complex patterns in data.

Advantages:

- Captures Complex Relationships, thus can model non-linear interactions between variables, making it more effective for complex ICU predictions.

- Because of its high predictive power it performs well when trained on large, high-quality datasets.
- Learns relevant patterns without the need for extensive feature engineering.

Disadvantages:

- Requires more resources (time and memory) compared to traditional models like logistic regression.
- Performance depends on choosing the right network architecture, learning rate, and regularization.

Implementation:

Model was trained using *MLPClassifier* from *sklearn.neural_network*. For hyperparameter tuning, different learning rates, L2 regularization, early_stopping and hidden layer sizes were used. Activation function used was *ReLU* (rectified linear unit) for hidden layers for faster training. Solver (optimizer) used was *sgd* (stochastic gradient descent) for efficient learning. The AUC score achieved on the training set is 0.903.

Performance Evaluation

We then evaluated classifier models (using the best models selected during training after hypertuning their parameters) on a test set. Since our goal is to accurately identify mortality, we prioritized **sensitivity or recall** (which measures how many actual deaths were correctly predicted). To improve detection of actual deaths, we **lowered the prediction threshold from 0.5 to 0.35**. Below is the summary of the performance of the models on test set:

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC-AUC**
Logistic Regression	0.72	0.28	0.9	0.42	0.8717
Random Forest	0.66	0.25	0.95	0.39	0.8921
XGBoost	0.73	0.29	0.92	0.44	0.8994
MLP Classifier	0.72	0.28	0.91	0.43	0.8928

Table 2: Performance of Classifier Models on Test set

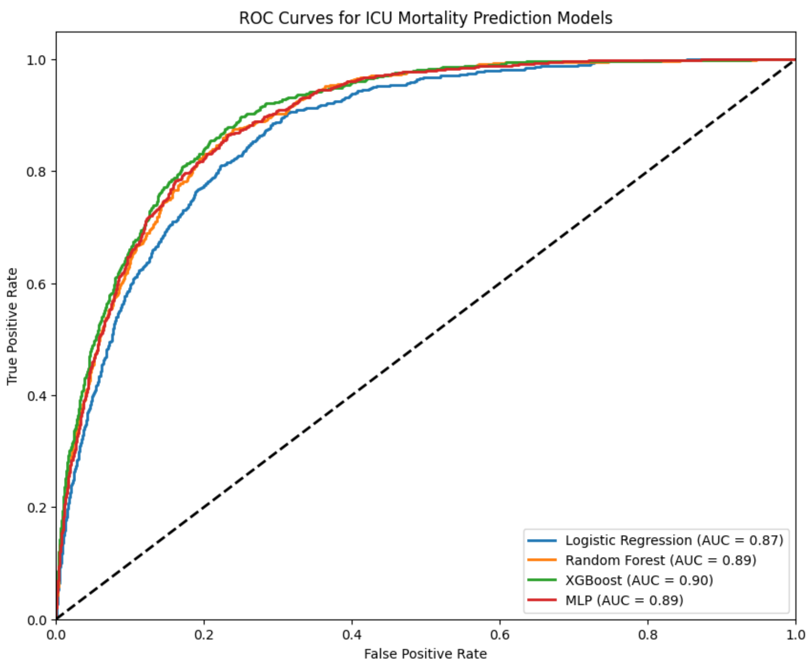


Figure 7: ROC Curves for different classifiers

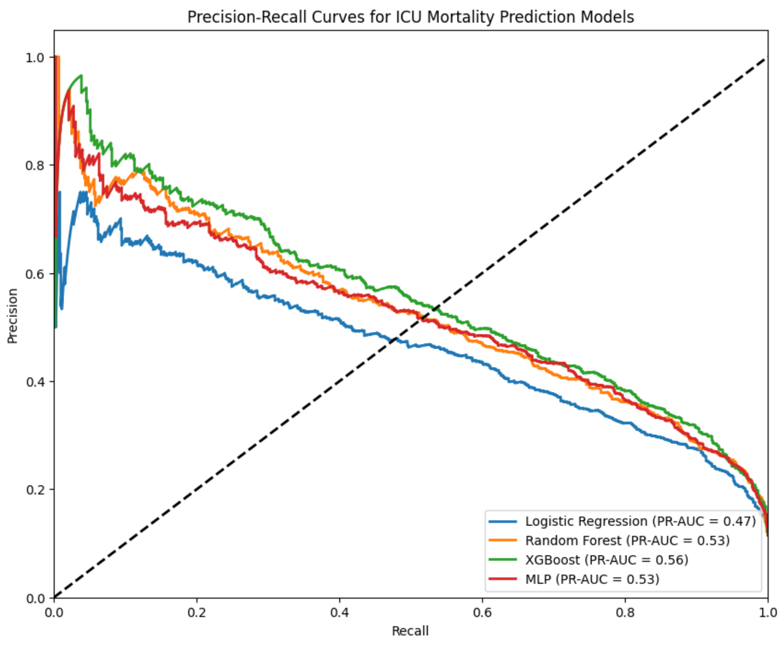


Figure 8: Precision-Recall curve for different classifiers

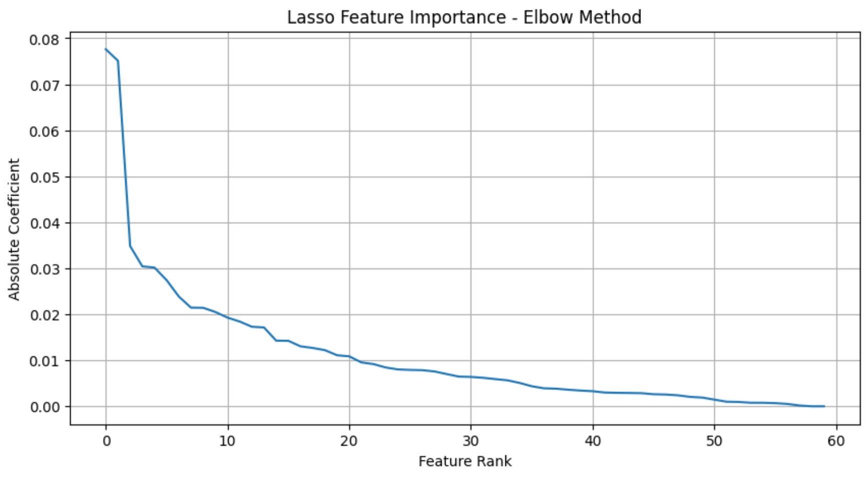
The final model selected for classification task of ICU Mortality prediction is **XGBoost Classifier.** XGBoost Classifier gave best ROC-AUC score among all models and the classification summary also shows that it gives better score compared to other models used.

b. Regression Modelling for ICU Length of stay prediction

Feature Selection

For the feature selection process of the regression task, we used **Lasso regression with cross-validation (LassoCV)** to identify the most relevant features. This method helps in reducing dimensionality and focusing the model on the most impactful predictors.

A **Pipeline** is created with "**StandardScaler"** for normalization and "LassoCV" for feature selection using 5-fold cross-validation to find the best regularization strength (alpha). Features are then sorted by the absolute value of the **Lasso model's coefficients**, which reflects their importance. Using the Elbow plot, as in Figure 9, we decided to keep the top 15 features. The training and test datasets are then filtered to include only the selected features for further modeling.



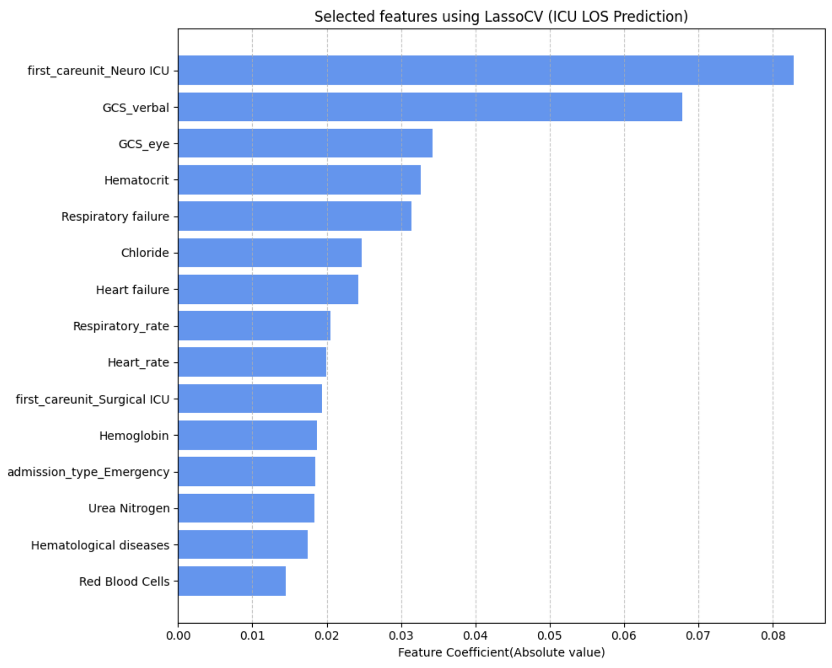


Figure 9: Feature Selection for ICU Length of stay prediction modelling

Modelling

1. **Elastic Net Regression** (Regularized version of Linear Regression)

Elastic Net Regression is a **linear regression model** that **combines both L1 (Lasso) and L2 (Ridge) regularization techniques**. It is particularly useful when there are multiple correlated features, as it can perform feature selection (like Lasso) while maintaining model stability (like Ridge).

Advantages:

- Handles **multicollinearity** better than Lasso or Ridge alone.
- Performs **automatic feature selection** and **regularization**.
- Help prevent **overfitting**.

Disadvantages:

- It can be computationally intensive for large datasets since it requires tuning of **two hyperparameters**: alpha (regularization strength) and l1_ratio (mixing parameter).
- Interpretation of coefficients can be harder than in plain linear regression.

Implementation:

Initially the model was trained using **LinearRegression()** from **sklearn.linear_model.** To further reduce the error, we chose to use ElasticNet() along with hyperparameter tuning. **Grid Search technique** is used with **5-fold cross-validation** and **negative mean absolute error** as the scoring metric. A parameter grid was defined to explore different values of **alpha** and **l1_ratio.** With the best estimator and best estimator, the best model was used on training dataset. The predicted output and actual value of y was then changed to original scale for calculating evaluation metrics. We achieved an R-square score of 0.14, MAE of 0.982, MAPE of 42.15% and RMSE of 1.32.

2. **K-Nearest Neighbors (KNN) Regressor**

KNN Regressor is a **non-parametric, instance-based learning algorithm** that predicts the target value for a data point based on the average (or weighted average) of the k-nearest training samples in the feature space. It makes predictions purely based on the proximity of data points, without learning an explicit model.

Advantages:

- **Simple and intuitive** to understand and implement.
- No training phase is needed and stores all data and computes predictions on demand.
- Performs well with **well-separated** data.

Disadvantages:

- **Computationally expensive** at prediction time for large datasets.
- Performance highly depends on the choice of **k** and the **distance metric**.

Implementation:

The model was trained using **KNeighborsRegressor()** from **sklearn.Neighbors.** We performed hyperparameter tuning using Grid Search technique, with **5-fold cross-validation** and **negative mean absolute error** as the scoring metric. A parameter grid was defined to explore different values of **k or n_neighbors** and **weights** as uniform or distance**.** With the best estimator and best estimator, the best model was used on training dataset to achieve a perfect R-square score of 1.0, MAE of 0.0001, MAPE of 0% and RMSE of 0.01. Evaluation metrics are calculated on original scale and not log scale. This might suggest overfitting of the model on the training dataset.

3. RandomForest Regressor

Random Forest Regressor is an **ensemble learning method** that builds multiple decision trees and averages their predictions to **improve accuracy and reduce overfitting**. It introduces randomness by bootstrapping the training data and selecting a random subset of features at each split, making it **robust and powerful for regression tasks**.

Advantage:

- **High accuracy** and robust performance on many types of data.
- Can handle **non-linear relationships** and **interactions** between variables.
- Works well with both **numerical** and **categorical** data.
- Automatically provides **feature importance**.

Disadvantages:

- **Less interpretable** than a single decision tree.
- It can be **computationally intensive** and memory-hungry with large datasets.
- Slower prediction time compared to simpler models.

Implementation:

The model was trained using **RandomForestRegressor()** from **sklearn.ensemble.** We then performed hyperparameter tuning to improve performance further. Grid Search technique, with **5-fold cross-validation** and **negative mean absolute error** as the scoring metric was used. A parameter grid was defined to explore different values of **estimators, max_depth, min_samples_split, min_samples_leaf, max_features** and **bootstrap** (as

True or False**).** With the best estimator and best estimator, the best model was used on the training dataset to achieve a R-square score of 0.85, MAE of 0.38, MAPE of 14.92% and RMSE of 0.54. Evaluation metrics are calculated on original scale and not log scale.

4. XGBoost Regressor

XGBRegressor is an implementation of gradient boosting from the XGBoost library, optimized for **speed and performance**. It builds an **ensemble of decision trees**, where each tree attempts to correct the errors of the previous ones, using gradient descent on a custom loss function. XGBoost includes regularization to prevent overfitting.

Advantages:

- **High predictive accuracy** and robustness.
- Efficient handling of **missing values** and **sparse data**.
- **Parallel and distributed computing** support.
- Includes **regularization (L1 & L2)** to reduce overfitting.

Disadvantages:

- **Complex model** – less interpretable than simpler algorithms.
- May overfit on **noisy** datasets if not properly regularized.
- Requires **careful hyperparameter tuning** for optimal performance.

Implementation:

The model was trained using XGBRegressor() from sklearn. Hyperparameter tuning was performed to improve performance further. Grid Search technique, with **5-fold cross-validation** and **negative mean absolute error** as the scoring metric was used. A parameter grid was defined to explore different values of estimators, max_depth, min_samples_split, min_samples_leaf, max_features and bootstrap (as True or False). With the best estimator and best estimator, the best model was used on training dataset to achieve a R-square score of 0.19, MAE of 0.94, MAPE of 40.24% and RMSE of 1.27. Evaluation metrics are calculated on original scale and not log scale.

Model	**RMSE**	**MAE**	**MAPE (%)**	**R2**
Elastic Net Regression	1.32	0.98	42.15	0.13
KNN Regressor	0.00	0.00	0	1
Random Forest	0.54	0.38	14.92	0.85
XGBoost	1.27	0.94	40.24	0.19

Table 3: Performance of Regression Models on Training set

Performance Evaluation

We then evaluated regression models (using the best models selected during training after hypertuning their parameters) on a test set where the target variable was log-transformed. For each model predictions were made on log scale, performed **inverse transform** of predictions and true values to original LOS scale using "np.expm1()" and then evaluation metrics were computed on original scale. Below is the summary of the performance of the models on test set:

Model	**RMSE**	**MAE**	**MAPE (%)**	**R2**	**CV Score**
Elastic Net Regression	1.29	0.96	41.15	0.15	0.99
KNN Regressor	1.31	0.96	39.38	0.13	0.98
Random Forest	1.23	0.95	41.10	0.17	0.99
XGBoost	1.27	0.94	40.36	0.18	0.98

Table 4: Performance of Regression Models on Test set

From the above regression modelling we find that KNN Regressor overfitted on the training set and hence performed comparatively poorly on the test set. On the test set the metrics of all the models are almost similar. But Elastic Net Regressor performed similarly on both training and test set, which might indicate that it can generalize better than other models. Also, it is easy to train since it requires less hyperparameter to be tuned. **Hence, we choose Elastic Net Regressor for predicting short ICU stays.**

Though the models fitted the data poorly, indicated by the poor R-squared score, our main aim was to achieve low Mean Absolute Error (MAE). Since the length of stay data is highly skewed, hence we chose MAE as the prime evaluation metrics. The calibration curve in Figure 10 shows that the predictions of the models are very close to true value.

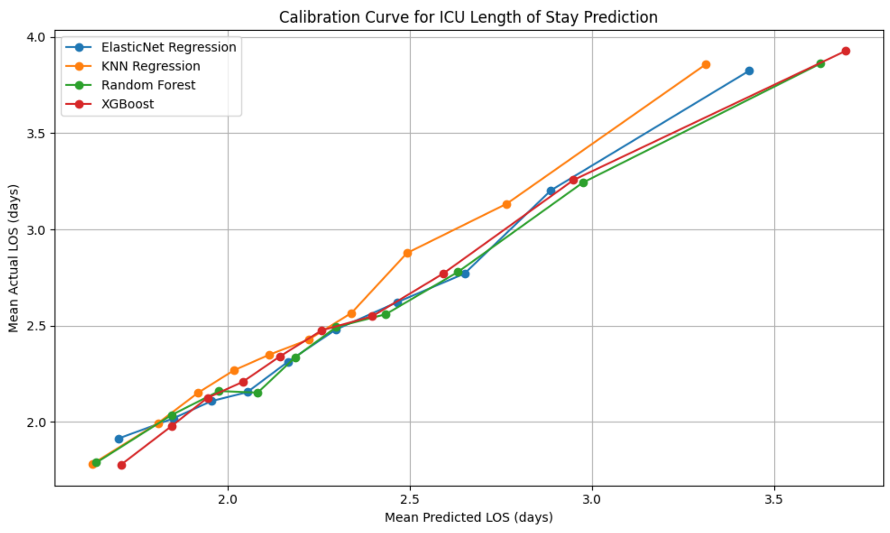
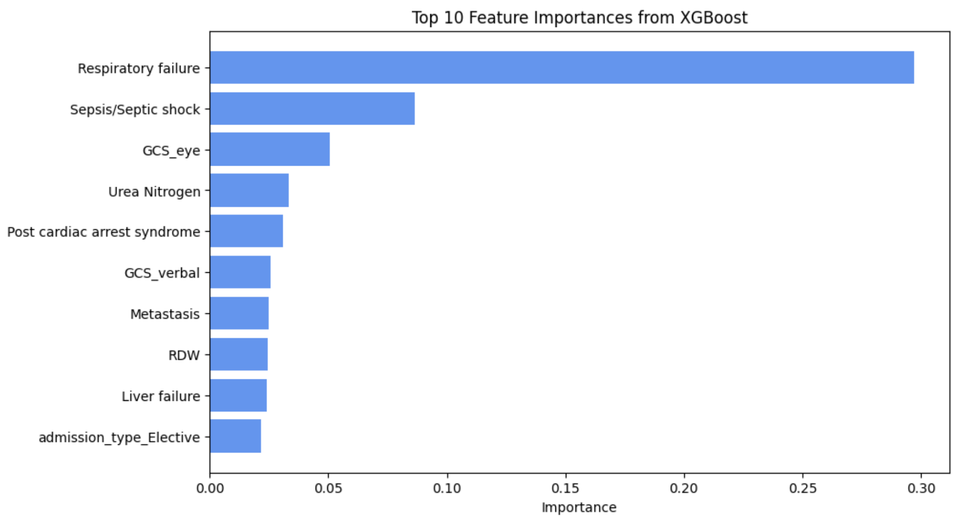


Figure 10: Calibration Curve of Regression Models on Test set

Project Outcomes

Our study reinforces the high predictive potential of machine learning models for ICU mortality. The continuous monitoring of critically ill patients generates extensive data streams, providing an ideal foundation for machine learning-driven risk assessment. Recent studies have increasingly explored this approach, reporting strong predictive performance (AUC: 0.85–0.89). Our findings align closely with these results, demonstrating similarly high accuracy in mortality prediction.

We used XGBoost's built-in feature importance method to rank variables based on their contribution to model performance. The most important features contributing to mortality are: Respiratory failure, Sepsis/Septic shock, Glasgow Coma Scale (GCS) – eye, Urea Nitrogen level, post cardiac arrest syndrome, Glasgow Coma Scale (GCS) – verbal, Metastasis, Liver failure and admission type of elective.



For ICU length of stay prediction, our chosen model, **Elastic Net Regressor**, achieved **a MAE of 0.96 days and a MAPE of 41%**, indicating moderate prediction accuracy. While this may not be too desirable for short stays ($\leq$7 days), the results surpass random prediction. The limited performance suggests that the first 24-hour lab events alone may not be adequate for LOS prediction. We focused only on survivors, aligning with prior studies that found LOS predictors differ between survivors and non-survivors. A more effective strategy could involve classifying LOS into short, average, and long stays, followed by regression within each group.

In conclusion, our analysis further established the fact that the machine learning algorithm could predict ICU mortality and length of stay prediction. **Respiratory failure, GCS eye and GCS verbal** are some of the key variables in predicting both mortality and length of stay.

Future Work

Electronic Health Records (EHR) offer a rich source of features for advanced healthcare analytics. The following approaches can be further explored to enhance the current analysis:

1. **Patient Risk Stratification through Clustering:**
   Implement clustering techniques to classify patients into distinct risk subgroups based on predictors of ICU mortality and length of stay. This stratification aids in identifying high-risk cohorts and personalizing care strategies.

2. **Improved Missing Data Handling:**
   Address missing values using imputation strategies that leverage the mean values within clinically relevant patient cohorts or subgroups. This ensures more reliable model inputs while preserving cohort-level characteristics.

3. **Categorical Preprocessing for ICU Stay Prediction:**
   To improve ICU length of stay predictions, first classify patients into short, average, and long stay groups. Tailoring feature selection and regression models to each subgroup can significantly enhance prediction accuracy and clinical relevance.

Citations

❖ *Johnson, A.E.W., Pollard, T.J., Shen, L., Lehman, L.-w. H., Feng, M., et al. (2021). MIMIC-IV, a freely accessible critical care database. Nature Scientific Data. https://physionet.org/content/mimiciv/2.2/*

(Data was obtained from MIMIC-IV and is available at https://physionet.org/content/mimiciv/1.0/ , with the permission of PhysioNet. Here is the link to only the extracted datasets used in our analysis)

❖ *Iwase, S., Nakada, Ta., Shimada, T. et al. Prediction algorithm for ICU mortality and length of stay using machine learning. Sci Rep 12, 12912 (2022). https://doi.org/10.1038/s41598-022-17091-5*

❖ *Pang, K.; Li, L.; Ouyang,W.;Liu, X.; Tang, Y. Establishment of ICU Mortality Risk Prediction Models with Machine Learning Algorithm Using MIMIC-IV Database. Diagnostics 2022,12, 1068. https://doi.org/10.3390/diagnostics12051068*