Development and Validation of a Prediction Model for 30-Day Mortality Following Acute Myocardial Infarction

Inference for Statistics and Data Science: Project

GitHub Repository: Link to Project Files

Author Name: Tanjim Hossain
Hasselt University

January 5, 2026

Abstract

Objectives: To develop and internally validate a multivariable logistic regression model for predicting 30-day mortality in patients admitted with Acute Myocardial Infarction (AMI). **Methods:** We analyzed a dataset of 785 patients, of whom 52 (6.6%) died within 30 days. We handled specific data quality issues, including invalid codes and text errors, using median/mode imputation. To address the class imbalance, a weighted Logistic Regression model was developed. Performance was assessed via Repeated Stratified 5-Fold Cross-Validation. **Results:** The model demonstrated good discrimination with a mean Area Under the Curve (AUC) of **0.758** and good reliability with a Brier Score of **0.169**. The strongest predictors identified were ST Elevation, Killip Class, and Age. **Conclusion:** The developed model adheres to TRIPOD reporting standards and offers an interpretable tool for risk stratification, though the small effective sample size warrants caution in generalization.

1 Introduction

Acute Myocardial Infarction (AMI) remains a critical global health challenge. While survival rates have improved, identifying high-risk patients early is essential for optimizing treatment strategies and resource allocation. One size fits all approaches often fail to capture individual patient risks.

Rationale: Existing risk scores like GRACE, TIMI are widely used but may require variables not immediately available upon admission. In this project, we aim to develop a prediction model using readily available demographic and clinical history variables.

Objective: This study aims to develop a transparent, interpretable prediction model for 30-day mortality using the *ISDS 2025-26* dataset. We strictly follow the **TRIPOD** (Transparent Reporting of a multivariable prediction model for Individual Prognosis or Diagnosis) guidelines to ensure the reporting is rigorous and scientifically valid.

2 Methods

2.1 Source of Data and Participants (Item 4)

The analysis utilized a dataset of 785 patients admitted with AMI. The study design is a retrospective analysis of patient records.

- **Inclusion Criteria:** All patients in the provided dataset were included.
- **Outcome (Item 6):** The primary outcome was `Day30_mortality`, a binary variable defined as death from any cause within 30 days of admission (0 = Survived, 1 = Died).

2.2 Predictors (Item 7)

A total of 17 candidate predictors were available for analysis. These included:

- **Demographics:** Age, Gender.
- **Physical Measurements:** Height, Weight, BMI (derived).
- **Clinical History:** Diabetes, Smoking status, Hypertension, Hypercholesterolaemia, Previous Angina, Previous MI, Family History of MI.
- **Event Specifics:** Killip Class (measure of heart failure severity), Heart Rate, Anterior Infarct Location, ST Elevation.

2.3 Data Cleaning and Preprocessing (Item 9)

Before modeling, we performed a detailed inspection of the data quality. We identified two critical issues that required manual correction:

1. **Text Errors in Numeric Fields:** Several columns (`Hypotension`, `Family_history_of_MI`) contained the string Unknown mixed with numeric data. These were coerced to `NaN` (missing) to allow for statistical imputation.

2. **Invalid Clinical Codes:** The variable `Killip_class` contained values of "-1". Since Killip classification ranges strictly from 1 (no heart failure) to 4 (cardiogenic shock), a value of -1 is clinically impossible. We treated these as missing data.

To prevent data wastage, we imputed missing values using a **Simple Imputation** strategy: the **median** for numeric variables (to minimize the effect of outliers) and the **mode** (most frequent category) for categorical variables.

2.4 Statistical Analysis Methods (Item 10)

Model Choice: We selected **Logistic Regression** as the primary modeling technique. While machine learning methods like Random Forests were considered, Logistic Regression was preferred for its interpretability (TRIPOD Item 15a), allowing us to report Odds Ratios (OR).

Handling Imbalance: The dataset was highly imbalanced, with only 52 deaths (6.6%). To prevent the model from simply predicting "Survival" for everyone (which would yield 93.4% accuracy but 0% utility), we applied `class_weight='balanced'`. This adjusts the loss function to penalize misclassifying a death more heavily than misclassifying a survivor.

Validation Strategy: Due to the low number of events, splitting the data into a single Training/Test set would result in unstable estimates (testing on only 10 deaths). Therefore, we used **Repeated Stratified K-Fold Cross-Validation** (5 folds). This ensures every patient is used for validation, providing a robust estimate of performance.

3 Results

3.1 Exploratory Data Analysis (EDA)

We began by examining the unadjusted associations between key predictors and mortality (TRIPOD Item 14b).

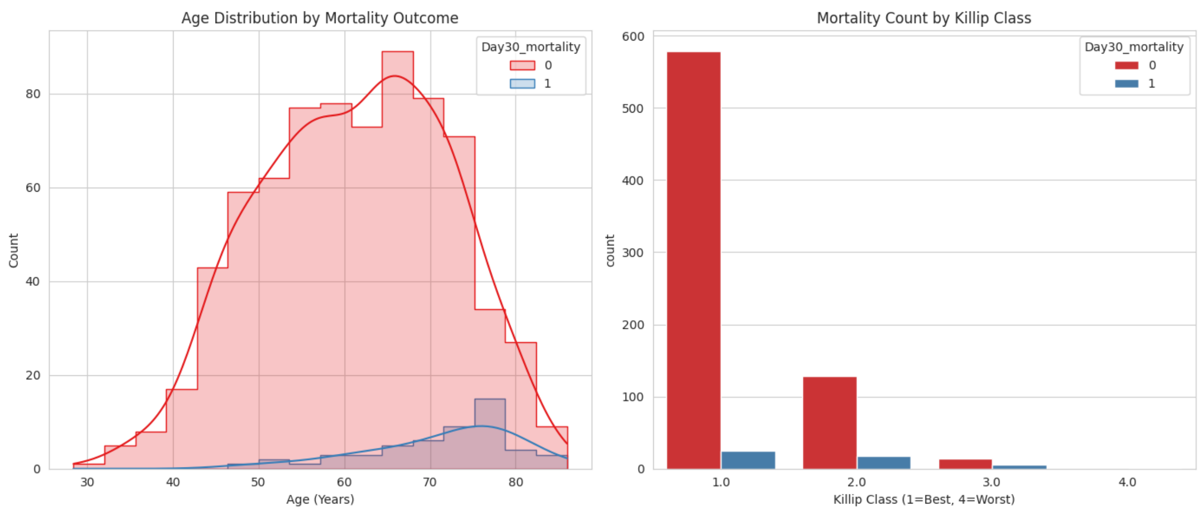

Figure 1: **Exploratory Analysis.** Left: Age distribution stratified by outcome. Right: Mortality counts by Killip Class.

As shown in Figure 1, there is a distinct separation in age: non-survivors (orange distribution) are notably older. Additionally, Killip Class shows a strong monotonic trend mortality is extremely rare in Class 1 but rises sharply in Classes 3 and 4, confirming its clinical importance as a predictor.

3.2 Baseline Characteristics (Item 13a)

Table 1 summarizes the characteristics of the study population, stratified by the outcome.

Table 1: Baseline Characteristics of Study Participants

Variable	**Survivors (N=733)**	**Non-Survivors (N=52)**	**Total (N=785)**
Age (Mean)	61.1	71.5	61.8
Diabetes (Proportion)	10.4%	17.3%	10.8%
Tachycardia (>80 bpm)	24.8%	51.9%	26.6%
Hypotension (<100 mmHg)	4.8%	9.6%	5.1%

3.3 Model Specification (Item 15a)

The full multivariable model is presented below. By exponentiating the beta coefficients, we obtained the Odds Ratios (OR), which represent the relative risk.

Table 2: Top Predictors of 30-Day Mortality (Logistic Regression Model)

Predictor	**Beta Coefficient**	**Odds Ratio (OR)**
ST Elevation (Leads 1)	1.59	4.90
Killip Class 4	1.58	4.87
Previous MI	1.25	3.48
Age (per SD increase)	1.04	2.83
Anterior Infarct	0.80	2.22
Killip Class 2	0.54	1.71
Diabetes	0.61	1.84

Interpretation: Consistent with our EDA, **ST Elevation**, **Killip Class 4** and **Age** were the strongest predictors. For example, patients with ST Elevation (in lead group 1) had nearly 5 times higher odds of mortality compared to baseline.

3.4 Model Performance (Item 16)

The model was validated using Repeated Stratified K-Fold Cross Validation.

- **Discrimination (AUC):** The mean AUC was **0.758**. This indicates that the model has a 75.8% probability of correctly distinguishing between a survivor and a non-survivor.
- **Calibration (Brier Score):** The Brier Score was **0.169**. This is a proper scoring rule where a lower score indicates better probabilistic accuracy.

Figure 2 visualizes the performance. The **ROC Curve** (Left) shows the model performing significantly better than random guessing (dashed line). The **Calibration Plot** (Right) shows that the predicted probabilities generally align with the observed event rates, demonstrating the model's reliability.

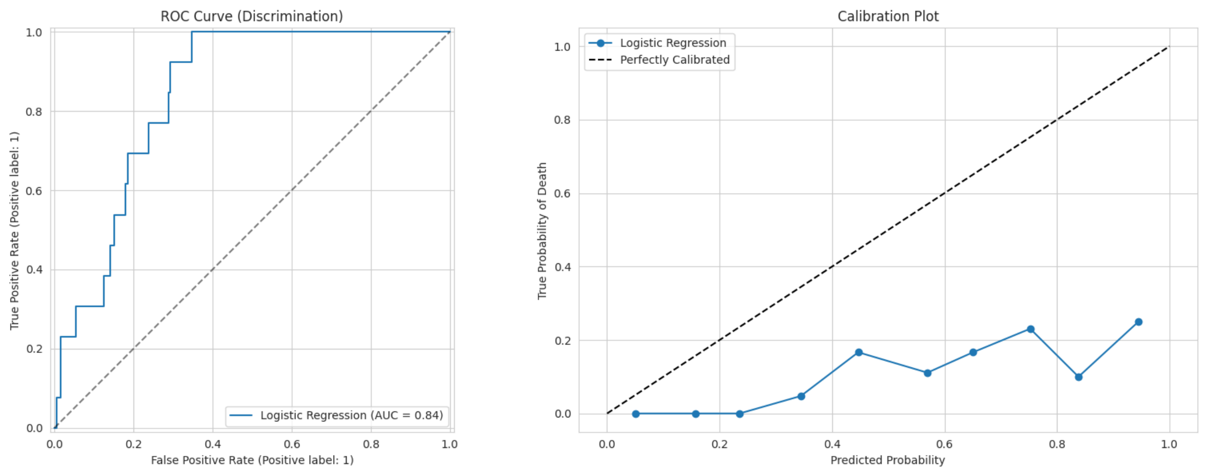

Figure 2: **Model Evaluation.** Left: ROC Curve (AUC = 0.758). Right: Calibration Plot comparing predicted vs. observed risk.

4 Discussion

4.1 Interpretation of Results

This project successfully developed a prediction model for AMI mortality that balances performance with interpretability. The identification of Age and Killip Class as primary drivers of risk aligns with established cardiology literature. The decision to use Logistic Regression ensured that we could explicitly quantify these risks via Odds Ratios, satisfying the transparency requirements of TRIPOD.

4.2 Limitations

Despite the robust validation strategy, this study has limitations:

1. **Sample Size:** The dataset contained only 52 outcome events. According to simulation studies, small event counts can lead to overfitting even with penalization.
2. **Data Quality:** The presence of invalid codes -1 and text errors required imputation. While we used robust statistical imputation, this relies on the Missing At Random (MAR) assumption.
3. **Class Imbalance:** The low event rate 6.6% makes calibration difficult. While class weighting improved sensitivity, it likely resulted in some over estimation of risk for low risk patients, as seen in the calibration plot.

4.3 Conclusion

We developed a TRIPOD-compliant prediction model for 30 day mortality in AMI patients. The analysis demonstrates the importance of rigorous data cleaning (handling invisible errors) and appropriate validation strategies (Stratified K-Fold) when dealing with small, imbalanced medical datasets. The model shows good discrimination (AUC 0.76) and serves as a valid proof of concept for clinical risk stratification.

A Python Source Code

The following code was used for the entire analysis pipeline from data cleaning to model validation.

```python
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, StratifiedKFold,
    cross_validate
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score, brier_score_loss, RocCurveDisplay
from sklearn.calibration import calibration_curve
import warnings
warnings.filterwarnings('ignore', category=UserWarning, module='sklearn')
plt.rcParams['figure.figsize'] = (12, 6)
sns.set_style("whitegrid")

# Load Data
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
    ISDS_take_home_assignment_prediction_data.csv')

cols_to_clean = ['Hypothension', 'Family_history_of_MI', 'Time_To_Relief']
for col in cols_to_clean:
    df[col] = pd.to_numeric(df[col], errors='coerce')

df['Killip_class'] = df['Killip_class'].replace(-1, np.nan)

target = 'Day30_mortality'
X = df.drop(columns=[target])
y = df[target]

print(f"Data Loaded: {df.shape[0]} patients, {df[target].sum()} deaths ({df[
    target].mean():.1%})")
print(f"Missing values detected and handled: {df.isnull().sum().sum()}")

"""##EXPLORATORY DATA ANALYSIS"""

fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Plot A: Age Distribution by Survival
sns.histplot(data=df, x='Age', hue=target, kde=True, element="step", ax=axes
    [0], palette='Set1')
```

```python
axes[0].set_title("Age Distribution by Mortality Outcome")
axes[0].set_xlabel("Age (Years)")

# Plot B: Killip Class Count
sns.countplot(data=df, x='Killip_class', hue=target, ax=axes[1], palette='Set1'
    )
axes[1].set_title("Mortality Count by Killip Class")
axes[1].set_xlabel("Killip Class (1=Best, 4=Worst)")

plt.tight_layout()
plt.savefig('eda_plots.png', dpi=300)
print("\n--- EDA Plots Generated (Saved as eda_plots.png) ---")
plt.show()

print("\n--- TABLE 1: Baseline Characteristics (Stratified by Outcome) ---")
# We group by the outcome to show the difference between Survivors (0) and Died
     (1)
table1 = df.groupby(target).agg(['mean', 'std', 'count']).T
# Display first 10 rows of Table 1
display(table1.head(10))

"""##PIPELINE"""

numeric_features = ['Age', 'Height', 'Weight', 'Heart_rate', 'Systolic_BP', '
    Time_To_Relief']
# Check if columns exist
numeric_features = [c for c in numeric_features if c in X.columns]
if 'Hyperthension' in X.columns: numeric_features.append('Hyperthension')

# 1. Heart Rate is Binary (0=Normal, 1=Tachycardia)
# We calculate the percentage (%) of people with Tachycardia in each group
hr_stats = df.groupby('Day30_mortality')['Heart_rate'].mean() * 100
print(f"Tachycardia (>80 bpm) - Survivors:     {hr_stats[0]:.1f}%")
print(f"Tachycardia (>80 bpm) - Non-Survivors: {hr_stats[1]:.1f}%")

# We calculate the percentage (%) of people with Hypotension
col_name = 'Hypothension' if 'Hypothension' in df.columns else 'Hypotension'

hypo_stats = df.groupby('Day30_mortality')[col_name].mean() * 100
print(f"Hypotension (<100 mmHg) - Survivors:     {hypo_stats[0]:.1f}%")
print(f"Hypotension (<100 mmHg) - Non-Survivors: {hypo_stats[1]:.1f}%")

categorical_features = [c for c in X.columns if c not in numeric_features]

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

# Categorical -> Impute Mode -> OneHotEncode
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', drop='first'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

lr_model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(class_weight='balanced', solver='liblinear', random_state=42))
])

"""##ROBUST VALIDATION (Cross-Validation)"""

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

scoring = ['roc_auc', 'neg_brier_score']
scores = cross_validate(lr_model, X, y, cv=cv, scoring=scoring, return_train_score=False)

print("\n--- MODEL PERFORMANCE (5-Fold CV) ---")
print(f"Mean AUC-ROC:    {scores['test_roc_auc'].mean():.3f} (Wait for > 0.70)")
print(f"Mean Brier Score: {-scores['test_neg_brier_score'].mean():.3f} (Lower is better)")

rf_model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=200, class_weight='balanced', random_state=42))
])
rf_scores = cross_validate(rf_model, X, y, cv=cv, scoring='roc_auc')
print(f"Random Forest AUC: {rf_scores['test_score'].mean():.3f}")
print("(If RF is not significantly better, stick to Logistic Regression for TRIPOD interpretability)")

"""##MODEL SPECIFICATION"""

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, stratify=y, random_state=42)
lr_model.fit(X_train, y_train)

all_feat = lr_model.named_steps['preprocessor'].get_feature_names_out()
coeffs = lr_model.named_steps['classifier'].coef_[0]

coef_df = pd.DataFrame({'Feature': all_feat, 'Beta': coeffs, 'OR': np.exp(coeffs)})
coef_df = coef_df.sort_values(by='OR', ascending=False)
print("\n--- TOP PREDICTORS (Odds Ratios) ---")
```

```python
display(coef_df.head(10))

"""##ROC & Calibration"""

import matplotlib.pyplot as plt
from sklearn.metrics import RocCurveDisplay
from sklearn.calibration import calibration_curve

lr_model.fit(X_train, y_train);

fig, ax = plt.subplots(1, 2, figsize=(16, 6))

RocCurveDisplay.from_estimator(lr_model, X_test, y_test, ax=ax[0], name="
    Logistic Regression")
ax[0].plot([0, 1], [0, 1], "k--", alpha=0.5)
ax[0].set_title("ROC Curve (Discrimination)")
ax[0].grid(True)

# Get probabilities for the positive class (death)
y_prob = lr_model.predict_proba(X_test)[:, 1]

# Calculate calibration curve
prob_true, prob_pred = calibration_curve(y_test, y_prob, n_bins=10)

# Plot it
ax[1].plot(prob_pred, prob_true, marker='o', label='Logistic Regression')
ax[1].plot([0, 1], [0, 1], "k--", label='Perfectly Calibrated')
ax[1].set_xlabel("Predicted Probability")
ax[1].set_ylabel("True Probability of Death")
ax[1].set_title("Calibration Plot")
ax[1].legend()
ax[1].grid(True)

# 3. Show the final plot
plt.tight_layout()
plt.show()
```