Predicting patient survival rates in critically ill

Arash Daneshvar
Politecnico di Torino
Student id: s314415
s314415@studenti.polito.it

***Abstract*—In this project, a pipeline is proposed to accurately predict the binary outcome ”death” for patients within a specified time frame. Initially, various missing values are handled, and feature extraction is performed using appropriate techniques. These processed features are then used as inputs for the hyperparameter tuning of three classification models: random forest, XGBoost, and gradient boosting classifier. The proposed approach achieves high accuracy and robustness in predicting patient survival, with performance evaluated using the F1-score, and it outperforms the baseline models established for this problem.**

I. Problem overview

The project consists of predicting survival rates in critically ill patients, which is seen as crucial in modern healthcare by significantly influencing clinical decision-making and patient management. The challenge to be addressed is the building of a classification model capable of accurately predicting the binary outcome ”death,” indicating whether a patient survives or dies within a specified time frame.

A dataset of 9,105 critically ill patients from five U.S. medical centers, collected between 1989-1991 and 1992-1994, is used for this project. The records, containing physiological, demographic, and disease severity information, are aimed at improving end-of-life care through accurate survival predictions. The dataset is divided into two parts:

- a development set, consisting of 7,285 patients, is used for training the model to predict the ”death” outcome;
- an evaluation set, consisting of 1,820 patients, is used for testing the model.

The model will be trained and validated using the development set and tested using the evaluation set. The development set is the initial focus. Through data exploration and visualization, it is understood that the data need to be preprocessed to handle several issues. The preprocessing step includes handling missing values, dropping correlated features, applying encoding for categorical data, normalization, balancing the dataset, and performing feature selection using PCA.

II. Proposed approach

The initial phase involves a comprehensive exploration of the dataset through various visualizations. This exploratory data analysis (EDA) is crucial for identifying the appropriate preprocessing techniques required for the dataset. Subsequently, five primary methods are employed to preprocess the data:

- Assessing correlations using a correlation matrix
- Handling missing values

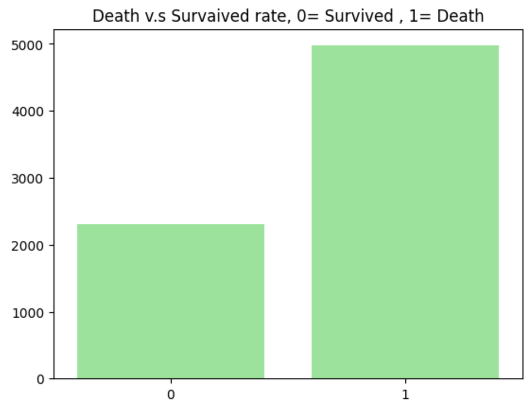

Fig. 1. Distribution of the target rate

- Encoding categorical variables
- Normalization
- Balancing the dataset

Each of these steps will be elaborated upon in the following sections.
Prior to preprocessing, an analysis of the data exploration will be presented.
To gain insights into the dataset, the analysis initially excludes the ’death’ column, which serves as the target variable, and the ’Id’ column, identified as non-informative. Subsequently, the distribution of the target variable is examined, focusing on the count of deceased and surviving patients. Figure 1 illustrates this comparison, revealing a disparity where the number of deceased patients is nearly twice that of survivors. Out of a total of 7,284 patients, 4,974 were recorded as deceased, while 2,310 patients were documented as survivors. This observation highlights an imbalance in the dataset, indicating the need for suitable balancing techniques.

Subsequently, the dataset is checked for missing values, revealing that 20 features contain incomplete data. To manage this, the data types of these features are identified to determine suitable handling methods. The dataset includes 31 columns of type float64, 7 columns of type object, and 4 columns of type int64. Figure 2 provides a visual representation of the distribution of data types across the dataset.

Next, categorical and numerical features within the dataset is identified . For the numerical columns, histograms and box plots is employed to visualize their statistical properties. Figure 3 showcases an example plot for the ”age” column.

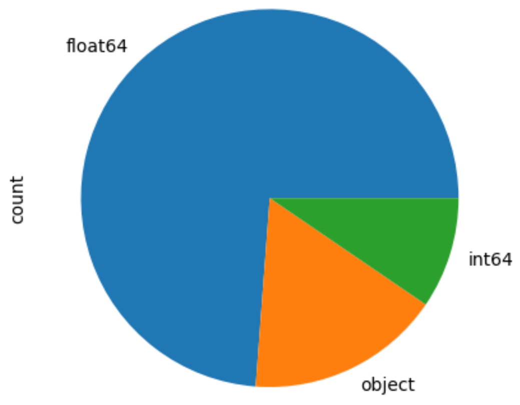

Fig. 2. Data types of the columns

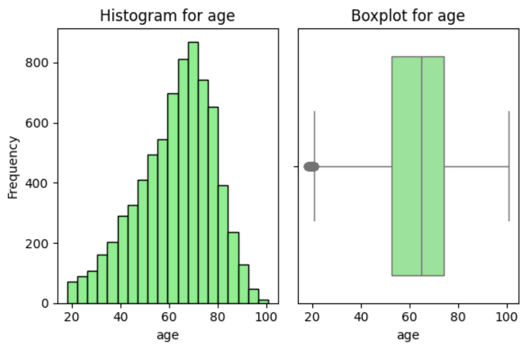

Fig. 3. Statistical properties of numerical features

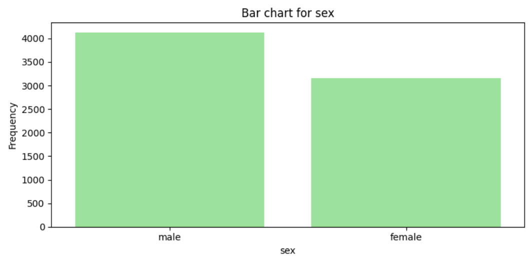

Fig. 4. Statistical properties of categorical features

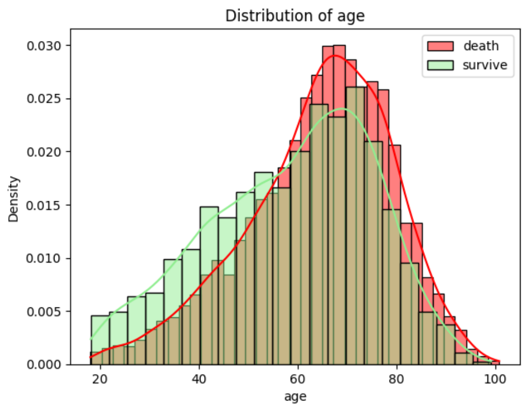

Fig. 5. Relationship between age feature and the target variable

The histogram provides a distribution overview, indicating that patients' ages range from 18 to 100 years, with an average age of 62 years. This graphical representation is complemented by the numerical summary from the .describe() function.

Meanwhile, the box plot illustrates the data's spread and skewness. It delineates the interquartile range (IQR), showing that the middle 50% of ages fall between 52 and 74 years. The whiskers extend to indicate the full range of data points, excluding outliers. Notably, the median age (64.88) is positioned within the box, indicating a right-skewed distribution towards older individuals.

These visualizations provide valuable insights into the distribution and characteristics of the numerical data, facilitating a deeper understanding of the dataset's demographic information and its implications. For categorical columns, bar charts is used . Figure 4 shows an overview of the gender of the patients, where male are more than female.

Next, the relationship between each feature and the target variable is examined. Histograms are utilized for numerical features, and heatmaps are employed for categorical features. Upon analysis, it is observed that the mortality rate among males is higher compared to females. Individuals diagnosed with Acute Respiratory Failure (ARF), Multiple Organ System Failure (MOSF) with Sepsis, Acute Kidney Injury (AKI), or Multiple Organ Dysfunction Syndrome (MODS) with sepsis exhibit a higher likelihood of death. An annual income of less than \$11K is associated with a higher likelihood of death, potentially due to financial constraints in affording hospital bills and medical care. The data indicates that the White race has a higher probability of death. Additionally, patients with metastatic cancer, characterized by the spread of cancer throughout the body, are more likely to die. Figure 5 illustrates that younger individuals under the age of 60 are more likely to survive, whereas older individuals have a higher probability of death.

A. *Preprocessing*

The first step involves identifying the correlation between the columns using the correlation matrix. Based on the insights derived from the correlation matrix and an understanding of the data, the following columns were removed: ['totcst', 'totmcst','sps', 'surv2m', 'prg2m', 'dnrday', 'adls', 'adlsc', 'dzclass']. This step is essential to reduce redundancy. Figure 6 illustrates this correlation matrix.

The second step involves addressing missing values within the dataset, which is observed to contain a significant amount of incomplete data. To address this issue, the 'SimpleImputer' function is utilized, employing statistical measures to impute missing values. Specifically, mean imputation is applied for numerical features, while the most frequent value imputation is used for categorical features.

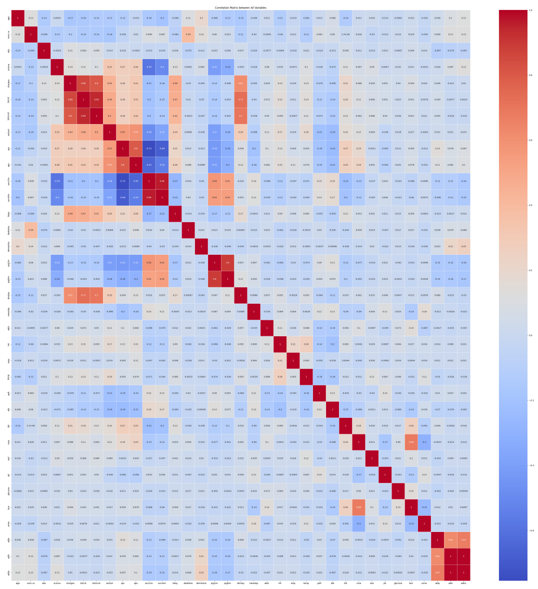

Fig. 6. Correlation matrix

The third step focuses on encoding categorical features. The categorical variables are encoded according to their specific characteristics. The categorical columns are: ['sex', 'dzgroup', 'income', 'race', 'ca', 'dnr']. For the 'income' column, since the values have an inherent order, they are converted to 0, 2, 3, 3 based on their range. The 'sex' variable, which only takes two distinct values, is encoded using OrdinalEncoding. For the remaining columns, one-hot encoding is performed using the get_dummies function.

The next step focuses on normalization. Data normalization is performed using 'StandardScaler', a method that standardizes features by centering them around their mean and scaling them to have unit variance. This process ensures that each feature contributes equally to the analysis, mitigating the influence of variables with larger scales and facilitating more effective model training and evaluation.

The next step examined is the imbalance of the target variable, a critical consideration in classification tasks. As previously noted, the target variable exhibits imbalance: the deceased population numbered 4,974, while the survived population numbered 2,310. To address this imbalance, an oversampling technique is implemented to balance the dataset, thereby preventing bias towards the majority class. Following oversampling, both classes comprise 4,974 individuals.

The final aspect is using Principal Component Analysis (PCA) method for feature selection. In this step, feature selection is performed, an iterative process aimed at identifying a subset of relevant variables from the dataset features. The method utilized is PCA. The PCA is then fitted, and 'Explained Variance' and 'cumulative variance' are calculated and plotted. Figure 7 shows the percentage of explained variance. The plot on the right side displays the percentage of cumulative variance, which helps determine the optimal number of features. Based on the plot, 35 features account for more than 99% of the data.

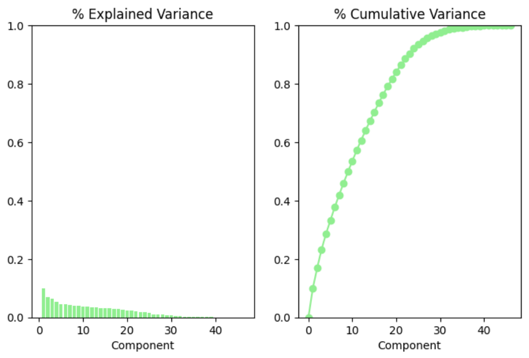

Fig. 7. PCA

B. Model selection

For training moedl the following classifier have been uesd:

- Random Forest classifier: The Random Forest classifier is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) of the individual trees. It improves predictive accuracy and controls overfitting by averaging multiple decision trees. Each tree is trained on a bootstrap sample of the data, and during the splitting process, a random subset of features is considered. This randomness ensures that the trees are decorrelated. Random Forest is robust to noise and can handle both numerical and categorical data.
- XGBoost classifier: XGBoost (Extreme Gradient Boosting) classifier is an optimized gradient boosting algorithm known for its speed and performance in machine learning competitions. It sequentially builds an ensemble of weak learners (typically decision trees) and combines them to improve prediction accuracy. [1]
- Gradient Boosting classifier: Gradient Boosting Classifier is an advanced machine learning technique that sequentially builds an ensemble of decision trees to enhance predictive accuracy. By iteratively training weak learners (typically decision trees) to correct errors from previous models, it combines them into a strong learner capable of handling complex data interactions and providing reliable predictions.

All three classifiers demonstrate optimal performance during the training model.

C. Hyperparameters tuning

GridSearch was utilized to optimize hyperparameters for the models employed in this study. In Table I, the best performing hyperparameter configurations identified through this process are displayed. A predefined set of hyperparameter values was systematically explored by GridSearch using cross-validation to maximize model performance metrics such as accuracy, precision, recall, and F1-score. The optimal configuration found during the tuning process is represented by the selected hyperparameter settings, ensuring that the models are fine-tuned for optimal predictive performance on the evaluation dataset.

TABLE I
BEST HYPERPARAMETER CONFIGURATIONS FOR MODELS

Model	Hyperparameters
Random Forest	`max_depth`: 11 `max_features`: auto `min_samples_leaf`: 1 `min_samples_split`: 2 `n_estimators`: 120 `random_state`: 39
XGBoost	`colsample_bytree`: 0.8 `gamma`: 0.05 `learning_rate`: 0.05 `max_depth`: 5 `min_child_weight`: 3 `n_estimators`: 500 `reg_alpha`: 0.01 `reg_lambda`: 0.01 `subsample`: 0.8
Gradient Boosting Classifier	`learning_rate`: 0.01 `n_estimators`: 200 `subsample`: 0.8 `random_state`: 39

III. RESULTS

To perform predictions on the evaluation dataset, it is necessary to preprocess the data in the same manner as the training set. This involves dropping correlated columns, handling missing values, encoding categorical columns and normalize the features.

After training and hyperparameter tuning, the classifiers were evaluated on an independent evaluation dataset. The performance metrics, F1-scores, for each classifier are detailed below:

- An F1-score of 0.746 was achieved by the Random Forest Classifier.
- An F1-score of 0.746 was achieved by the XGBoost Classifier.
- An F1-score of 0.738 was achieved by the Gradient Boosting Classifier.

IV. DISCUSSION

The evaluation of classifiers on the independent dataset revealed the following F1-scores: 0.746 for both the Random Forest and XGBoost classifiers, and 0.738 for the Gradient Boosting Classifier. Comparatively, the baseline F1-score was determined to be 0.74. Notably, both the Random Forest and XGBoost classifiers slightly outperformed the baseline, demonstrating their capability in predicting outcomes on unseen data. In contrast, the Gradient Boosting Classifier, while slightly below the baseline, still exhibited competitive performance.

Despite extensive preprocessing efforts, including feature engineering and normalization, the overall impact on model performance was minimal. For instance, applying outliers detection using boxplots as a technique resulted in the removal of significant samples from the dataset. This reduction in dataset size did not lead to improved performance; in some cases, it even led to a decrease in predictive accuracy. This outcome show that while preprocessing techniques are essential for data preparation, their indiscriminate application without careful consideration of the dataset's characteristics can adversely affect model training and performance.

REFERENCES

[1] Chen, Tianqi Guestrin, Carlos. (2016). XGBoost: A Scalable Tree Boosting System. 785-794. 10.1145/2939672.2939785.