Data Science Lab: Process and Methods
Politecnico di Torino
Project Assignment Summer Call, A.Y. 2023/2024

Mozhdeh Hajiani S330007

Abstract - In this project we created a model using machine learning to forecast whether a patient will survive or pass away within a period.

I. Introduction

Predicting survival rates in critically ill patients is crucial for modern healthcare, as it influences clinical decisions and patient management. Accurate predictions allow for timely interventions, improve patient quality of life, and optimize resource allocation. This project aims to create a machine learning model to predict whether critically ill patients will survive or die within a specified timeframe.

The dataset which is used includes records of 9105 critically ill patients from five medical centers in the United States, collected between 1989-1991 and 1992-1994. Each object represents a hospitalized patient who met the criteria related to nine disease categories: acute respiratory failure, chronic obstructive pulmonary disease, congestive heart failure, liver disease, coma, colon cancer, lung cancer, multiple organ system failure with malignancy, and multiple organ system failure with sepsis. The goal is to determine if these patients survived based on several physiologic, demographics, and disease severity information. This problem is important because the world is dealing with a growing concern which patients lose their controls over their lives as they get close to the end of life. Therefore, our aim is to facilitate decision making and planning processes in order to minimize instances of prolonged suffering during end-of-life stages.

II. Methodology

Data Loading and Exploration

We analyzed the dataset to understand its structure, summary statistics, and the presence of missing values. To visualize the extent of missing data, we created a heat map of missing values. The heat map of missing values helped us to identify which features had significant amounts of missing values, guiding our strategy for handling these missing values during preprocessing.

However, we tried over and over to make a progress by deleting the entire rows or columns which have a great deal of missing values according to an arbitrary threshold but no improvement resulted. After that, we decided to impute the missing values using some approaches which we explain in the following.

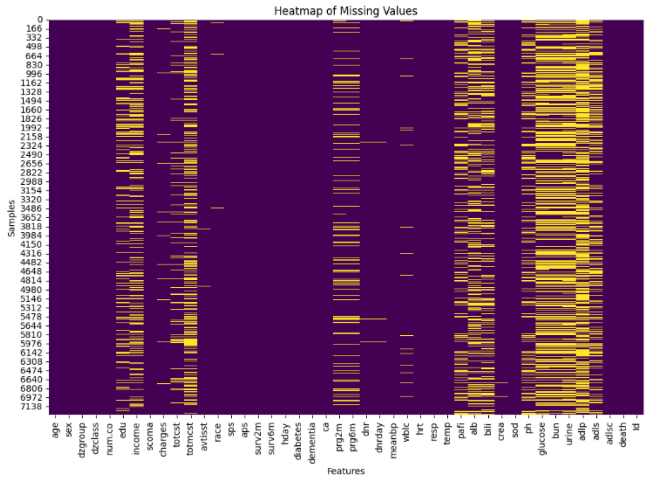

Fig. 1. heat map of missing values

We also examined the correlation matrix for numerical features to understand the relationships between different variables. This step helps in identifying highly correlated features, which may need to be considered during model training to avoid multicollinearity.

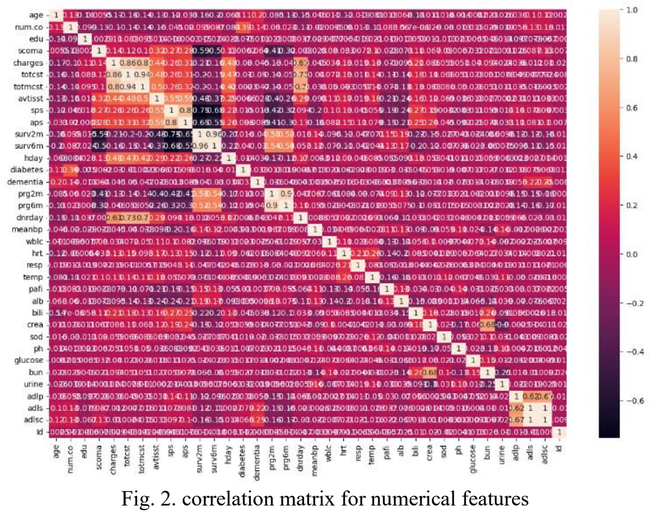

Fig. 2. correlation matrix for numerical features

III. Data Preprocessing

Data preprocessing steps were implemented to handle missing values, encode categorical variables, and scale numerical features. This was done using scikit-learn's `ColumnTransformer` and `Pipeline`.

Handling Missing Values:

- **Numerical features:** Missing values were imputed using the K-Nearest Neighbors (KNN) imputer. The KNN imputer estimates missing values by finding the nearest neighbors and averaging their values. This method is chosen for its ability to provide more accurate imputations by considering the similarity between data points.
- **Categorical features:** Missing values were imputed using the most frequent category. This method replaces missing values with the most common value in the column, ensuring that the imputed values are reasonable for categorical data.

Encoding Categorical Variables:

- Categorical features were encoded using one-hot encoding to convert them into numerical format suitable for machine learning models. One-hot encoding creates binary columns for each category, allowing the model to handle categorical data effectively.

Scaling Numerical Features:

- Numerical features were scaled using `StandardScaler` to ensure that they have a mean of 0 and a standard deviation of 1, improving model performance. Scaling is essential to ensure that features with larger ranges do not dominate the model training process.

Model Building

We divided the dataset into a train and test set. For the model selection we tried different models and different configurations. We started with a `RandomForestClassifier` which was chosen for its robustness and effectiveness in binary classification tasks. The model was trained and evaluated using the F1 score. Also, we used an optimized gradient boosting framework that improves performance and execution speed.
Moreover, Ensemble methods were employed to enhance prediction accuracy. The following classifiers were used:

1. **Random Forest Classifier:**
An ensemble of decision trees that reduces overfitting by averaging multiple trees.

2. **Gradient Boosting Classifier:**
It builds trees sequentially to reduce errors, focusing on misclassified samples from previous trees.

3. **Voting Classifier:**
Combines predictions from multiple models (Random Forest and Gradient Boosting) to improve overall performance.

It seemed that this setting is the best model for our dataset. We divided the dataset into train and test sets and used the train set for training the model and the test set for evaluating the model.

Hyperparameter Tuning

To improve the model's performance, hyperparameter tuning was performed using `GridSearchCV`. We tried different values for the number of estimators and the maximum depth of the trees. We also tried different values for the minimum number of samples required to split an internal node and the minimum number of samples required to be at a leaf node.

After achieving the best hyperparameters, we finally created our pipeline which consists of a preprocessor, then we will normalize the features and finally we will train our model.

Predict on the Evaluation Set

The best model was used to predict the outcomes for the evaluation dataset.

IV. Results

The model was evaluated using the F1 score, which balances precision and recall, making it suitable for imbalanced datasets. Hyperparameter tuning improved the model's performance, and the final predictions were prepared for submission.

F1 Score (Random Forest):
0.8436080467229073
F1 Score (Gradient Boosting):
0.8446760013025072
F1 Score (Voting Classifier):
0.8364611260053619

I. Conclusion

We successfully constructed our proposed machine learning model to predict patient survival rates in critically ill populations and observed the predicted values along with the F1 Score. The process involved data preprocessing, model training, hyperparameter tuning, and evaluation. The resulting model can assist healthcare providers in making informed decisions, ultimately improving patient care and resource management.