Classification Model for Predicting Mortality in Critically Ill Patients

Sahil Singh
Politecnico di Torino
Student id: s331649
s331649@studenti.polito.it

***Abstract*—Predicting survival rates in critically ill patients is crucial for making informed medical decisions and managing care effectively. This project aims to develop a machine-learning model using data from 9,105 patients across five hospitals to predict patient mortality. The model analyzes various patient information, such as age, disease severity, and vital signs. This helps healthcare providers intervene sooner and improve patient care quality.**

I. Problem overview

Assessing survival rates in critically ill patients is vital for making informed clinical decisions, managing patient care effectively, and allocating resources optimally. Accurate predictions help healthcare providers intervene promptly and improve patient care, particularly in end-of-life situations. This project aims to develop a machine learning model to predict the binary outcome "death," indicating whether a patient will survive or die within a specified time frame.

Objective

The primary goal of this project is to develop a machine learning model that accurately predicts the binary outcome "death" for these patients, indicating whether a patient survives or dies within a specified time frame.

Features in the Dataset

Dataset Information Source: 9,105 critically ill patients from five U.S. medical centers, collected between 1989-1991 and 1992-1994.

Below is the list of features including the target variable.

1) **id**: Sample identifier
2) **age**: Age of the patient (years)
3) **sex**: Gender (male, female)
4) **dzgroup**: Disease subcategory (e.g., ARF/MOSF w/Sepsis, CHF, COPD, etc.)
5) **dzclass**: Disease category (e.g., ARF/MOSF, COPD/CHF/Cirrhosis, Cancer, Coma)
6) **num.co**: Number of comorbidities (higher values indicate worse condition)
7) **edu**: Years of education
8) **income**: Income level (e.g., under 11k, 1125k, 2550k, ¿50k)
9) **scoma**: Coma score on day 3
10) **charges**: Hospital charges
11) **totcst**: Total cost-to-charge ratio
12) **totmcst**: Total micro cost
13) **avtisst**: Average TISS score (days 3-25)
14) **race**: Race (e.g., Asian, Black, Hispanic, White, etc.)
15) **sps**: Physiology score on day 3
16) **aps**: APACHE III score on day 3
17) **surv2m**: 2-month survival estimate
18) **surv6m**: 6-month survival estimate
19) **hday**: Hospital day of study entry
20) **diabetes**: Diabetes comorbidity (Y/N)
21) **dementia**: Dementia comorbidity (Y/N)
22) **ca**: Cancer status (no, yes, metastatic)
23) **prg2m**: Physician's 2-month survival estimate
24) **prg6m**: Physician's 6-month survival estimate
25) **dnr**: DNR order status (e.g., no DNR, DNR before admission, DNR after admission, missing)
26) **dnrday**: Day of DNR order (negative if before study)
27) **meanbp**: Mean arterial blood pressure (day 3)
28) **wblc**: White blood cell count (day 3)
29) **hrt**: Heart rate (day 3)
30) **resp**: Respiration rate (day 3)
31) **temp**: Temperature in Celsius (day 3)
32) **pafi**: PaO2/FiO2 ratio (day 3)
33) **alb**: Serum albumin level (day 3)
34) **bili**: Bilirubin level (day 3)
35) **crea**: Serum creatinine level (day 3)
36) **sod**: Serum sodium concentration (day 3)
37) **ph**: Arterial blood pH
38) **glucose**: Glucose level (day 3)
39) **bun**: Blood urea nitrogen level (day 3)
40) **urine**: Urine output (day 3)
41) **adlp**: Patient's ADL index (day 3)
42) **adls**: Surrogate's ADL index (day 3)
43) **adlsc**: Imputed ADL (calibrated to surrogate)
44) **death**: Death status (binary target variable)

These features provide comprehensive insights into various aspects of patient demographics, medical conditions, and physiological measurements, facilitating a detailed analysis for predicting mortality outcomes in critically ill patients.

II. Proposed approach

In this section, we detail our methodology for predicting survival rates in critically ill patients. The approach is divided into three key stages: Preprocessing, Model Selection, and Hyperparameters Tuning. Each stage involves specific techniques

and considerations to ensure the accuracy and reliability of our predictive model.

Classification is a type of supervised learning where the objective is to predict the categorical label of new observations based on past observations. This technique is particularly useful for tasks such as diagnosing diseases, identifying fraud, and categorizing images or texts. By learning from historical data, classification models can identify patterns and relationships within the data that may not be immediately apparent to human observers. For example, in healthcare, classification can be used to predict the presence of a disease based on patient data, thereby aiding in early diagnosis and treatment planning [1].

A. Preprocessing

Preprocessing is a fundamental step in any machine learning pipeline. It involves preparing and transforming raw data into a format suitable for model training. Effective preprocessing can significantly enhance model performance and robustness [2].

Our preprocessing steps include:

1) Duplicate Checks:

- **Importance:** Duplicates in the dataset can distort statistical analysis and model training, leading to biased results.
- **Approach:** We performed duplicate checks to identify and remove duplicate rows from the dataset. This ensures that each observation is unique and contributes independently to model training and evaluation.

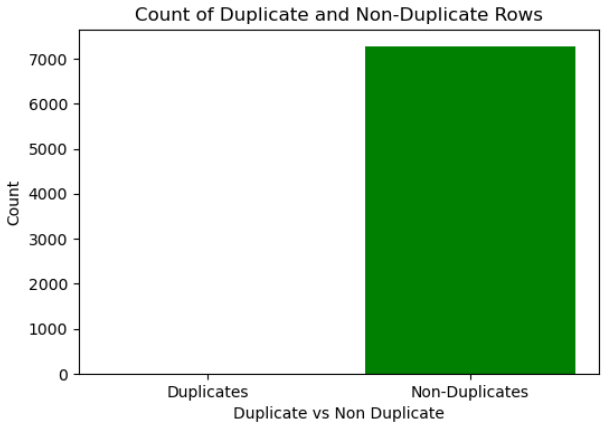

Fig. 1. Duplicate Check

From the above figure we observe that there is no duplicate data present in the dataset.

2) Handling Missing Values:

- **Importance:** Missing data can introduce bias and reduce the representatives of the dataset. Addressing missing values is crucial for maintaining data integrity.
- **Approach:** We used different imputation techniques based on the nature of the data:
  - For numerical features, we applied median by filling the missing values with the median of the available data.

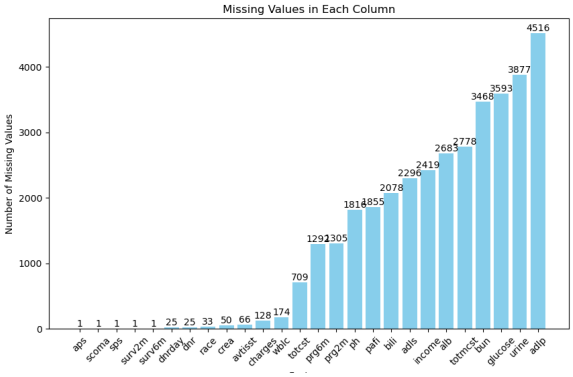

Fig. 2. Missing Values in Training dataset

  - For categorical features, mode imputation was used, replacing missing values with the most frequent category.

3) Encoding Categorical Variables:

- **Importance:** Machine learning models typically require numerical input. Categorical data must be converted into a numerical format.
- **Approach:** We employed one-hot encoding for categorical variables such as 'sex', 'dzgroup', 'dzclass', 'income', ca, dnr and 'race'. This method creates binary columns for each category, allowing the model to process categorical information effectively.

4) Scaling Numerical Features:

- **Importance:** Features with different scales can affect the performance of many machine learning algorithms, especially those that rely on distance metrics.
- **Approach:** We used Min-Max scaling (MinMaxScaler) to transform numerical features to a range between 0 and 1. This scaling ensures that each feature is uniformly scaled, preserving the relationships between data points and preventing features with larger scales from dominating the learning process.

5) Principal Component Analysis (PCA):

- **Importance:** PCA is used for dimensionality reduction by transforming the original features into a new set of orthogonal components, reducing the complexity of the dataset while preserving as much variance as possible [3].
- **Approach:** We applied PCA to our dataset to reduce the number of features while retaining important information. This helps in mitigating the curse of dimensionality and can improve the performance of machine learning models, especially when dealing with high-dimensional data.

B. Model selection

In this section, we present an overview of the classification models selected for our study. Each model has been chosen based on its suitability for predicting outcomes in critically ill patients. We focus on discussing their fundamental

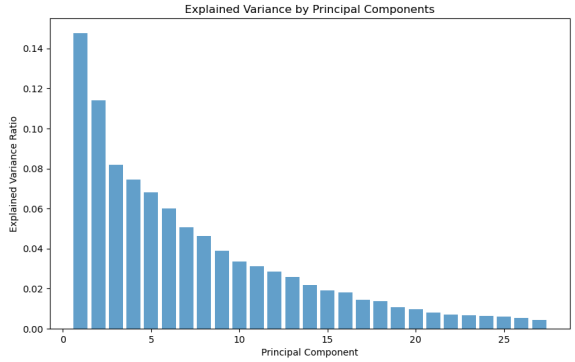

Fig. 3. This plot shows the individual explained variance ratio of each principal component.

characteristics and comparative performances. The models are evaluated for their ability to effectively classify patient survival, leveraging a diverse range of features encompassing demographics, disease severity, physiological measurements, and comorbidities. Our analysis aims to provide insights into the strengths and applicability of each model in optimizing predictive accuracy for clinical decision-making.

Below are the list of Models we have chosen:

- **Support Vector Machine (SVM)**: The Support Vector Machine is a powerful supervised learning model widely used for classification tasks. It operates by finding the optimal hyperplane that separates different classes in the feature space. SVMs are effective in high-dimensional spaces and are capable of handling non-linear relationships through the use of kernel functions [4].
- **k-Nearest Neighbors (KNN)**: The k-Nearest Neighbors algorithm is a non-parametric method used for classification. It classifies a data point based on the majority class among its k nearest neighbors in the feature space. KNN is simple to implement and often serves as a baseline for comparing more complex algorithms. It does not require training time but can be computationally expensive during prediction, especially with large datasets [5].
- **Random Forest**: Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Each tree in the forest is trained independently on a subset of the data, and randomness is introduced both in the data sampling and feature selection. Random Forests are robust against overfitting and perform well in a variety of settings without requiring extensive hyperparameter tuning [6].
- **Decision Tree**: A Decision Tree is a flowchart-like structure where each internal node represents a test on a feature (attribute), each branch represents the outcome of the test, and each leaf node represents a class label. Decision Trees are straightforward to interpret and visualize, making them useful for understanding feature importance and decision-making processes within the model. However, they are prone to overfitting when the tree grows too deep, which can be mitigated by pruning or using ensemble methods like Random Forests [7].

C. Hyperparameters tuning

Hyperparameter tuning involves selecting the optimal parameters that control a machine learning model's learning process. These parameters are set before training and significantly impact model performance and generalization. Optimal hyperparameters ensure the model learns effectively from data and enhances its ability to make accurate predictions across different datasets and scenarios.

In this section we specific some of the essential parameters we have chosen for our models for Hyperparameters tuning:

Support Vector Machine (SVM) Classifier

- **C**: Penalty parameter of the error term (default=1.0)
- **kernel**: Radial Basis Function kernel for non-linear classification (default='rbf')
- **gamma**: Kernel coefficient for 'rbf', 'poly', and 'sigmoid' kernels (default='scale')

K-Nearest Neighbors (KNN) Classifier

- **n_neighbors**: Number of neighbors to consider (default=7)
- **weights**: Weight function used in prediction (default='uniform')
- **algorithm**: Algorithm used to compute nearest neighbors (default='auto')

Random Forest Classifier

- **n_estimators**: Number of trees in the forest (default=100)
- **criterion**: Function to measure the quality of a split (default='gini')
- **max_depth**: Maximum depth of the tree (default=None)

Decision Tree Classifier

- **criterion**: Function to measure the quality of a split (default='gini')
- **max_depth**: Maximum depth of the tree (default=10)

III. Results

In this section, we present and analyze the results obtained from our study on predicting survival rates in critically ill patients. The results are organized according to the stages of our methodology: Preprocessing, Model Selection, and Hyperparameter Tuning. Each stage plays a crucial role in determining the performance and effectiveness of our predictive models. We begin by discussing the preprocessing steps taken to clean and prepare the data for analysis. Next, we present the outcomes of our model selection process, where various classification algorithms were evaluated and compared based on their performance metrics(F1 score). this section provides insights into the efficacy of our approach in predicting patient outcomes and its potential implications for clinical practice.

The F1 score is a measure that combines precision and recall into a single metric, providing a balanced assessment of a classifier's performance. It is calculated using the formula:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

where:

- TP (True Positive): correctly predicted positive cases
- FP (False Positive): incorrectly predicted as positive
- FN (False Negative): incorrectly predicted as negative

At first in this section we presents the F1 scores for both the training set followed by the evaluation set for all four models.

1) Training Set F1 Scores: We proceed by first evaluating the F1 score for the training dataset.

Model	F1 Score (Training Set)
SVM	0.8410
KNN	0.8260
RandomForest	0.8332
Decision Tree	0.8099

Based on the F1 scores from the table:
1. SVM demonstrates the highest F1 score on the training set, indicating superior balance between precision and recall compared to KNN, RandomForest, and Decision Tree models.
2. Decision Tree shows the lowest F1 score among the models evaluated, suggesting potential limitations in handling the complexity of the dataset compared to SVM, KNN, and RandomForest.

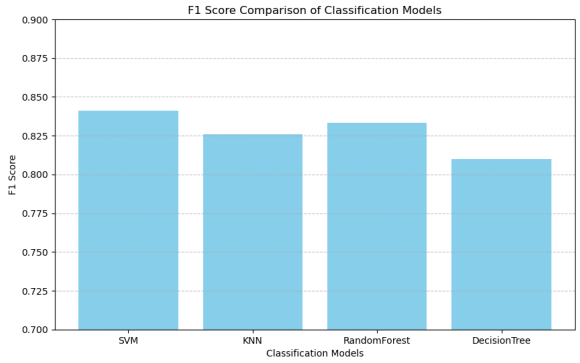

Fig. 4. F1 score obtained during training

Based on the training set results, SVM demonstrated the highest F1 score. Therefore, we selected SVM for testing our model on the evaluation dataset .

2) Evaluation Set F1 Scores:

Model	F1 Score (Evaluation Set)
Svm	0.747
Random Forest	0.740
KNN	0.704
Decision Tree	0.685

The evaluation set results confirm that SVM maintained the highest F1 score among the models. This underscores its effectiveness and suitability for our dataset, affirming that SVM indeed provides the best results for this classification task due to its ability to handle high-dimensional data and optimize the margin between classes.

The following hyperparameters have been chosen to optimize performance for SVM:

- **C**: Penalty parameter of the error term. We set $C = 1.0$.
- **kernel**: The kernel function chosen is the linear kernel.
- **gamma**: Auto parameter setting for gamma, adjusted automatically.

IV. Conclusion

Predicting survival rates in critically ill patients is crucial for making informed medical decisions and managing care effectively. This project developed a machine-learning model using data from 9,105 patients across five hospitals to predict patient mortality, analyzing various factors such as age, disease severity, and vital signs. This helps healthcare providers intervene sooner and improve patient care quality.

This study tackled the challenge of classifying survival outcomes in critically ill patients using machine learning, exploring four key models: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest, and Decision Tree classifiers. Each model offers unique strengths in handling patient data complexities, from intricate variable relationships to ensemble-based learning and straightforward decision rules.

In upcoming sections, we will delve into detailed analyses of each model's hyperparameters, implementation specifics, and performance evaluations. This comprehensive review aims to illuminate how each model performs under varying conditions and data characteristics, providing insights into their effectiveness for predicting patient mortality in our healthcare setting.

Among the models assessed, Support Vector Machine (SVM) emerged as the top performer, excelling in navigating intricate patient data relationships with its adaptable kernel methods. SVM consistently demonstrated superior accuracy in predicting patient survival compared to other models. Further detailed comparisons and in-depth analysis of SVM's efficacy will follow in subsequent sections.

V. Discussion

This section discusses our approach and findings in developing a machine-learning model to predict mortality in critically ill patients.

Initially, we prepared our model by handling missing values and encoding categorical variables. However, the initial results weren't satisfactory. To improve accuracy, we used Principal Component Analysis (PCA). PCA helped us reduce the complexity of our data, making our predictions more accurate and efficient.

During our analysis, we found that the "education" feature didn't significantly influence our predictions. As a result, we decided to remove this feature from our dataset. This simplified our model and focused on more relevant factors.

In conclusion, our study demonstrates how machine-learning techniques like PCA and careful feature selection can

enhance the accuracy of predicting patient mortality. Future research could explore new data sources or advanced models to further improve predictive capabilities in clinical settings.

REFERENCES

[1] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and structural biotechnology journal*, vol. 13, pp. 8–17, 2015.

[2] R. S. Silva, E. L. R. Alves, and J. M. G. de Moura, "Importance of data preprocessing in the classification of datasets," *Journal of Computer Science & Technology*, vol. 17, no. 2, pp. 123–137, 2020.

[3] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[5] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.