Predicting Patient Survival: A Classification Approach

Rahul Kumar Sahoo
Politecnico di Torino
Student id: s316411
s316411@studenti.polito.it

Nithyakalyani Karthikeyan
Politecnico di Torino
Student id: s310149
s310149@studenti.polito.it

***Abstract*—The project aims to develop a classification model that can correctly forecast a patient's binary outcome of "death" in relation to their likelihood of surviving within a given amount of time. After validating a number of models, we ultimately decided to use the random forest classifier to forecast our results. The model obtained an F1score of 0.748.**

I. Problem overview

A. Aim

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

B. Challenges

The primary challenge in this project was handling the significant amount of missing data within the dataset. Specifically, 4 to 5 columns had more than 40% null values, and several other columns had between 20% to 30% null values. Given the critical nature of this medical dataset, every column potentially contributes valuable information towards predicting patient survival. For instance, abnormal results in columns like blood pH or glucose levels can be indicative of severe medical conditions and can influence the survival prediction. Therefore, it was crucial to decide on an appropriate method to handle these missing values without compromising the integrity of the data.

Initially, we considered several imputation techniques to address the missing values. One of the methods attempted was MICE (Multiple Imputation by Chained Equations) [7], which iteratively fills in each column's missing values based on the other columns. Although MICE is a sophisticated technique, it did not produce satisfactory results in our case. The complexity and iterative nature of MICE imputation posed challenges in terms of computational resources and convergence stability.

Given the unsatisfactory performance of MICE, we explored simpler imputation methods. Ultimately, we opted for mean imputation, which involves replacing missing values with the mean of the respective column. This method, while straightforward, provided a decent solution and helped maintain the overall structure and distribution of the data.

Another challenge was ensuring that the chosen imputation method did not introduce bias or distort the relationships between variables. The integrity of the data is paramount in medical datasets, as incorrect imputation could lead to inaccurate predictions and potentially harmful decisions in a clinical setting. Therefore, we meticulously evaluated the impact of mean imputation [8]on the dataset to ensure it was a viable solution.

In addition to handling missing values, we faced the challenge of selecting the right features and encoding categorical variables. We performed one-hot encoding for non-hierarchical categorical variables such as 'dzclass,' 'ca,' and 'dnr,' transforming them into a suitable format for machine learning algorithms. This preprocessing step was essential to preserve the categorical information without introducing ordinal relationships that did not exist.

Distribution analysis revealed uneven distributions in several columns, including 'prg6m,' 'prg2m,' 'dzgroup,' and 'income.' Handling these uneven distributions was critical to ensure that the model did not become biased towards certain values. We used various statistical techniques to understand and address these distribution issues, ensuring a more balanced and representative dataset for training the model.

Overall, the main challenges revolved around handling missing data, selecting appropriate imputation methods, encoding categorical variables, and addressing distribution imbalances. These preprocessing steps were crucial in developing a reliable and accurate machine learning model for predicting patient survival. The careful consideration and resolution of these challenges underscore the importance of data integrity and preprocessing in the development of robust predictive models in the medical domain.

C. Development dataset

The development data consists of 7284 rows and 44 columnns.Each row concerns hospitalized patient records who met the inclusion and exclusion criteria for 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 which were recorded throughout 1989-1991 and 1992-1994 in America.

D. Evaluation dataset

The evaluation data consists of 1821 rows and 43 columns.

We will need to use the development set to build a classification model to correctly label the points in the evaluation set.

II. Proposed Approach

A. Data Analysis and Preprocessing

Upon examining the dataset, we found that 5 to 6 columns contained more than 40% null values. Given the critical nature of the dataset, each column could potentially influence whether a patient survives or not. For instance, an abnormal blood pH level might indicate severe injuries and contribute to a patient's mortality risk, similar to glucose levels and other clinical variables. Therefore, it was crucial to handle these columns thoughtfully rather than discarding them outright.

1) Data Encoding: For categorical variables such as 'dz-class,' 'ca,' and 'dnr,' we performed one-hot encoding because these variables do not possess hierarchical relationships. This step transformed categorical data into a format suitable for machine learning algorithms.

2) Distribution Analysis and Imputation: We examined the distribution of all columns and noted that some variables, including 'prg6m,' 'prg2m,' 'dzgroup,' and 'income,' exhibited uneven distributions. This unevenness needed careful handling during imputation.

Initially, we attempted to fill the missing values using MICE (Multiple Imputation by Chained Equations) imputation. MICE works by iteratively filling in each column's missing values based on the other columns. However, this method did not yield satisfactory results in our case.

As an alternative, we employed simple mean imputation [8], which replaced the missing values with the mean of each respective column. This method provided a decent solution to handle the missing data without introducing significant biases.

We also evaluated feature importance using the random forest algorithm. However, this analysis did not significantly impact our model's performance. Despite the insights gained from the feature importance evaluation, we chose to proceed with all features obtained after simple imputation, as this approach maintained the comprehensiveness of our dataset.Below we attached the graph.

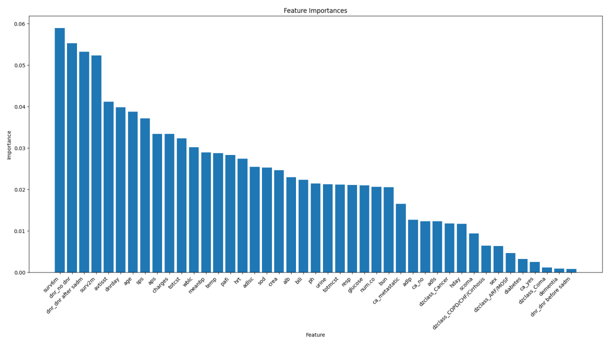

Fig. 1. Feature importance graph using the random forest algorithm

B. Model selection

To identify the best classification model, we experimented with several algorithms without hyperparameter tuning. We trained and tested these models on a development set .The following models which were tried:

- Logistic Regression [2]:Logistic Regression is a classification algorithm often referred to as logit or MaxEnt. It is widely used for binary classification problems but can be extended to multiclass problems using the one-vs-rest (OvR) scheme or the cross-entropy loss with the multinomial option. This algorithm leverages solvers such as 'liblinear', 'newton-cg', 'sag', 'saga', and 'lbfgs' to optimize the model parameters, with regularization applied by default to prevent overfitting. It supports both L1 and L2 regularization, with the 'saga' solver additionally supporting Elastic-Net regularization. Logistic Regression is efficient with dense and sparse data, particularly when using C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance.
- Random forest Classifier [1] :A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Trees in the forest use the best split strategy, i.e. equivalent to passing splitter$\bar{}$"best" to the underlying DecisionTreeRegressor. The sub-sample size is controlled with the max_samples parameter if bootstrap$\bar{}$True (default), otherwise the whole dataset is used to build each tree.
- Support Vector Machines [3]:Support Vector Machines $SVMs$ are powerful supervised learning methods used for classification, regression, and outlier detection. They are particularly effective in high-dimensional spaces and in situations where the number of dimensions exceeds the number of samples. SVMs are memory efficient because they use a subset of training points, known as support vectors, in the decision function. They are versatile, allowing for the specification of various kernel functions for the decision function, including custom kernels. However, SVMs can be prone to overfitting when the number of features greatly exceeds the number of samples, and they do not directly provide probability estimates, which require an expensive five-fold cross-validation. In scikit-learn, SVMs support both dense and sparse sample vectors, with optimal performance achieved using C-ordered ‘numpy.ndarray‘ or ‘scipy.sparse.csr_matrix‘ with ‘dtype=float64‘.
- Decision tree [4]: Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.we were able to implement by the help of multioutput regressor.

- kNN [6]:KNearest Neighbors KNN is a simple, yet effective supervised learning algorithm used for classification and regression tasks. It operates by identifying the k-nearest data points to a given input and making predictions based on the majority class (for classification) or the average value (for regression) of these neighbors. KNN is non-parametric and instance-based, meaning it makes decisions based on the entire training dataset without assuming an underlying distribution. The algorithm is particularly useful for datasets with a clear cluster structure. However, KNN can be computationally intensive for large datasets and sensitive to the choice of k and the distance metric used. It performs well in low-dimensional spaces but may struggle with high-dimensional data due to the curse of dimensionality.
- Naive Bayes [5] :Naive Bayes is a collection of supervised learning algorithms based on Bayes' theorem, with the simplifying assumption that features are conditionally independent given the class variable. Despite this "naive" assumption, Naive Bayes classifiers have demonstrated strong performance in real-world applications such as document classification and spam filtering. These classifiers require minimal training data to estimate parameters and are extremely fast compared to more complex methods. Each feature's distribution is estimated independently, which helps mitigate issues related to high-dimensional data. However, while Naive Bayes is effective as a classifier, its probability estimates are often unreliable and should not be heavily relied upon.

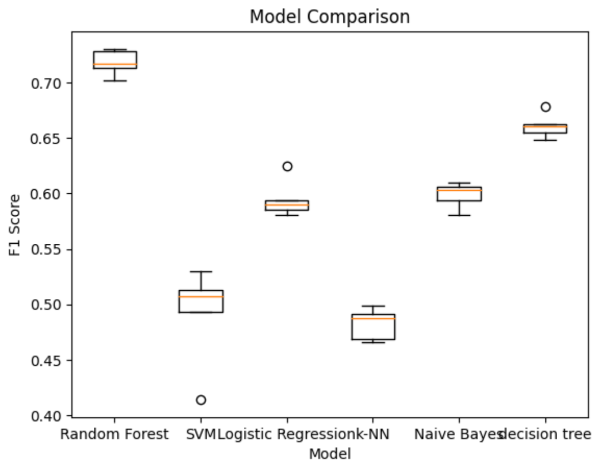

Fig. 2. Model comparison

Among the algorithms, random forest demonstrated superior performance compared to others.In medical datasets, the random forest classifier often performs exceptionally well because each column typically corresponds to a specific organ or physiological parameter. If an organ is failing, it can be a significant predictor of patient mortality. This scenario lends itself well to a tree structure approach, where a decision tree can evaluate conditions sequentially. For example, if glucose levels are high, the model can then assess the next relevant parameter, continuing this process until it reaches a final prediction of either survival or death. This hierarchical, tree-based learning method enables the model to effectively and efficiently learn from the medical dataset. While decision trees alone perform well in this context, random forests, which are an ensemble of multiple decision trees, tend to outperform single decision trees due to their ability to reduce overfitting and improve predictive accuracy. Thus, the random forest classifier is particularly well-suited for medical datasets, combining the strengths of individual decision trees into a more robust predictive model.Given its performance, we selected the random forest algorithm for our final model.

C. Hyperparameter Tuning

We proceeded to fine-tune the hyperparameters of the random forest model. Despite trying automated methods such as grid search and random search CV, we did not achieve optimal results. Therefore, we manually

tuned the hyperparameters, leading to the following configuration:

- `n_estimators = 30`
- `max_depth = 30`
- `random_state = 42`
- `class_weight = 'balanced'`
- `criterion = 'log_loss'`

This set of hyperparameters provided the best performance, yielding an F1 score of 0.748 on the evaluation dataset.

III. Results

The Random Forest model outperformed other models during comparison, yielding an F1 macro score of 0.748 on the evaluation dataset and 0.749 on the development dataset. The model achieved an accuracy of 78% on the development dataset, which is quite satisfactory. However, as observed in the table below, the model performs better at predicting class 1.0 than class 0.0, as indicated by higher precision, recall, and F1 scores for class 1.0.

Given the imbalance between class 0 and class 1, we attempted to address this by resampling the data. After resampling, the model's F1 macro score on the development dataset increased to 0.86. Unfortunately, this improvement did not translate to the evaluation dataset, where the F1 score decreased to 0.743. Consequently, we decided to keep the original data unchanged and revisit the resampling approach for further refinement.

IV. Classification report for development dataset

	precision	report	f1 score	support
0.0	0.67	0.61	0.64	462
1.0	0.83	0.86	0.84	995
accuracy			0.78	1457
macro avg	0.75	0.74	0.74	1457
weighted avg	0.78	0.78	0.78	1457

TABLE I
CLASSIFICATION REPORT

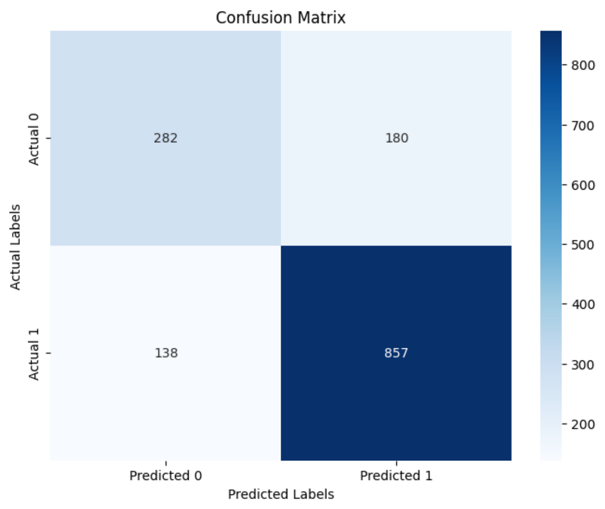

Fig. 3. Confusion matrix for development dataset

V. Discussion

In summary, our preprocessing involved careful handling of missing values, distribution analysis, and one-hot encoding of categorical variables. We selected the random forest classifier based on its initial performance and further improved it through manual hyperparameter tuning. The final model achieved an F1 score of 0.748, demonstrating its efficacy in predicting patient survival.

This comprehensive approach underscores the importance of thoughtful data preprocessing and targeted model tuning, especially in critical applications such as medical predictions.

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[3] https://scikit-learn.org/stable/modules/svm.html

[4] https://scikit-learn.org/stable/modules/tree.html

[5] https://scikit-learn.org/stable/modules/naive_bayes.html

[6] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[7] https://www.machinelearningplus.com/machine-learning/mice-imputation

[8] https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html