Patient Survival Rate Prediction

Abstract—**In this report, we show a possible approach to predict the patient survival rate to facilitate informed decisions. The proposed approach uses a Random Forest Classifier with smote to treat the imbalanced data, after removing outliers, encoding categorical features with dummy encoding. The results have been evaluated using Macro f1 score, and it is observed that the random forest classifier model works better than the SVM and the logistic regressor.**

I. Problem overview

The primary goal of this project is to develop a machine learning model that accurately predicts the binary outcome ”death” for critically ill patients, signifying whether a patient survives or dies within a specified time frame. The dataset comprises 9105 individual critically ill patients from five United States medical centers, accessioned between 1989-1991 and 1992-1994. The patient records include several physiologic, demographic, and disease severity information. The task is to build a binary classification pipeline to predict the target variable ”death.” The submissions will be evaluated based on the f1 score.

II. Proposed approach

A. Preprocessing

As a starting point for the development of this project, we analyzed each feature to understand their meanings and to determine if any of them might be insignificant to our objective.

From this initial logical analysis, the following features were found to be of little significance:

- **id**: The identification number of each patient does not provide any useful information for our prediction.
- **dzclass**: Correlated with ”dzgroup,” which contains more information, making it redundant and thus eliminable.
- **totcost**, **totmcst**: Probably correlated with ”charge,” making them redundant and eliminable.
- **edu**: The level of education is not useful for our purposes.
- **income**: Irrelevant; while one might think that higher-cost care could be better and guarantee a higher survival rate, this information is already encoded in ”charge.”
- **adls**, **adlsc**: Useful for medical purposes only when ”adlp” is missing.
- **dnrday**: Does not present any significant information compared to ”dnr.”

At this point, we proceeded to examine the dataset. To do this, we unified the ”evaluation” and ”development” datasets to get a clearer and more comprehensive view of the problem. Subsequently, the two datasets will be separated as they were originally.

During the exploration of the dataset, we found several missing values.

Finally, we checked whether the problem was balanced or not, in order to choose the most appropriate direction to find a solution to our problem.

We discovered that we are dealing with an imbalanced dataset, where the majority class is represented by the label 1.

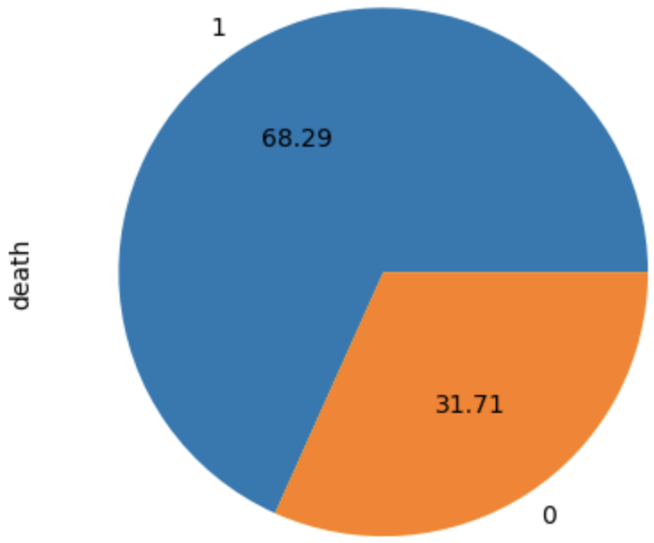


Fig. 1. Imbalanced dataset

During data exploration, we wondered if the features regarding measurements (glucose, urea...) were linked in some way to the dzgroup. For this reason, for each peature we tried to replace the missing values with the average value calculated by grouping the data of that feature by dzgroup. However, this worsened the performance of the models for which the missing values had been replaced with simple average values, without consideration of the dzgroup they belonged to. As a consequence, the missing values were replaced by their corresponding mean values, or by their mode, if it was a categorical field.

Distinction was made for:

- **adlp**: As mentioned earlier, when not present, ”adls” is used in the medical field. Since, as we will see later, ”adls” is highly correlated with ”adlsc,” the missing values of ”adlp” were replaced with the corresponding values of ”adlsc,” which are always present.
- **pgr2m**, **pgr6m**: Following the same reasoning as the previous case, these missing values were replaced by the corresponding ”surv2m” and ”surv6m” values.

At this point, we proceeded to test the correlation between the various features. In fact, the features to be used in a classification model should not be correlated with each other. A first look trough the following matrix helped us understand the more correlated features:

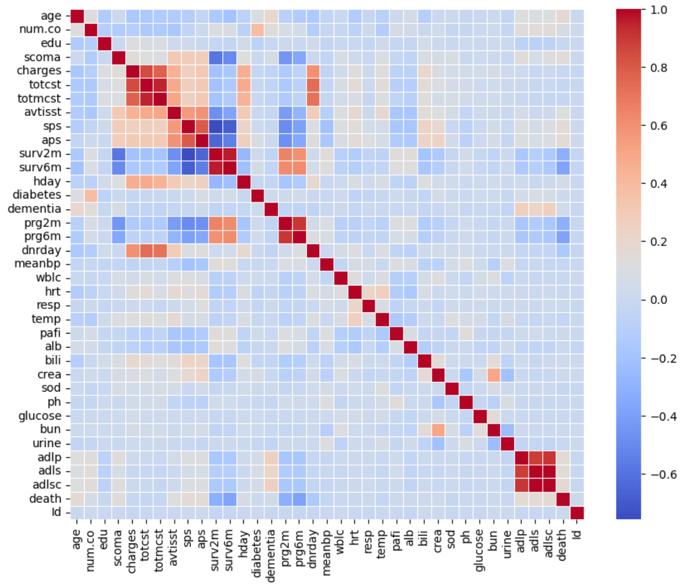
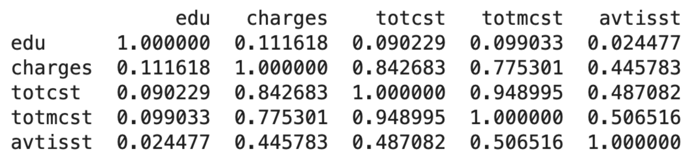


Fig. 2. Feature Correlation

First, we tested the correlation between all the ”monetary” features.



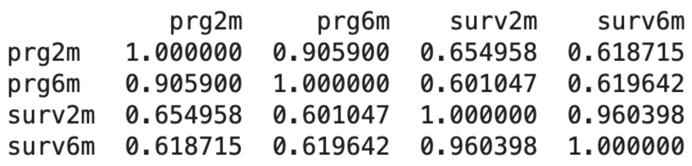
```
              edu   charges    totcst   totmcst   avtisst
edu      1.000000  0.111618  0.090229  0.099033  0.024477
charges  0.111618  1.000000  0.842683  0.775301  0.445783
totcst   0.090229  0.842683  1.000000  0.948995  0.487082
totmcst  0.099033  0.775301  0.948995  1.000000  0.506516
avtisst  0.024477  0.445783  0.487082  0.506516  1.000000
```

Fig. 3. Correlation Matrix

As can be seen from the above matrix, some of these features are highly correlated with each other. For this reason, the columns ”totcst” and ”totmcst” will be dropped from the dataset. This result confirms what was observed in the preliminary feature analysis phase.

At this point, we tested the correlation between ”dzgroup” and ”dzclass”; finding this to be quite high as well, we decided to eliminate ”dzclass” as it is less informative compared to ”dzgroup.”

Finally, the following correlations was evaluated:



```
           prg2m     prg6m    surv2m    surv6m
prg2m   1.000000  0.905900  0.654958  0.618715
prg6m   0.905900  1.000000  0.601047  0.619642
surv2m  0.654958  0.601047  1.000000  0.960398
surv6m  0.618715  0.619642  0.960398  1.000000
```

Fig. 4. prg and surv correlation

Since ”prg2m” and ”prg6m” and ”surv2m” and ”surv6m” were found to be highly correlated, we tested which of these were less correlated with the target variable, in order to eliminate the appropriate feature.

The result showed that ”prg2m” and ”surv2m” were less correlated with ”death” and were therefore removed from the dataset.

An important part of data preprocessing is removing noise, as it can compromise the predictive capabilities of our model. For this reason, we used the IQR technique to remove outliers from all numerical features. Finally, to better utilize classification models, categorical variables were discretized using ”dummy encoding,” and numerical features were standardized.

B. *Model selection*

At this stage, the dataset has been restored the two initial dataset: ”development (dev)” and ”evaluation (eval).”

Given the task’s nature as a binary classification problem with an imbalanced dataset, a decision was taken to evaluate the performance of the following models:

- Logistic regression
- Random forest
- Support Vector Machine (SVM)

Starting with the dataset obtained from the initial stages, a traditional workflow was followed to determine the $f1\_macro$ score for the predictions produced by the aforementioned three models:

- Dividing the dataset into training and testing sets
- Training the models using the training data
- Computing the $f1\_macro$ score via cross-validation

Considering the notably lower performance demonstrated by the SVM model when compared to the other two models, a strategic decision was made to exclude the SVM model and focus on improving the effectiveness of the remaining models.

Logistic Regression

The performance of the model was assessed based on the ”dev” dataset initially to determine the potential impact of future strategies on enhancing the model’s predictive capabilities.

As noted earlier, an imbalanced dataset was utilized, prompting an investigation into the effectiveness of various balancing methods on performance enhancement. Three balancing strategies were examined, namely:

- Upsampling of the minority class
- Downsampling of the majority class
- Employment of SMOTE technique

Despite implementing these techniques, none of them resulted in a performance enhancement from the f1_macro perspective.

Subsequently, leveraging the outcomes from the initial assessment, we proceeded to evaluate feature importance and eliminate those features that contributed minimally to the model's predictive abilities. It was determined that eliminating the 20 least significant features proved to be advantageous in improving the model's performance measured by the $f1\_macro$ metric.

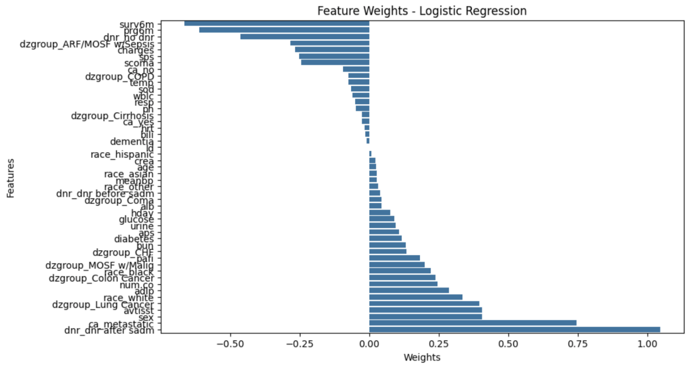


Fig. 5. Feature Importances by Logistic Regression

In the next phase, the trained model's performance was evaluated using the refined set of features, leading to an enhancement in the $f1\_macro$ score.

Random Forest The initial performance evaluation was conducted using the "dev" dataset. This evaluation is crucial for understanding whether future strategies will genuinely enhance the model's predictive power. From this assessment, we observed that our model outperformed the logistic regressor, indicating a promising start.

Two different dataset balancing techniques were tested to improve the model's performance:

- We used the class_weight='balanced' parameter when creating the RandomForestClassifier. This method adjusts the weights inversely proportional to the class frequencies in the input data, ensuring that each class contributes equally to the learning process.
- SMOTE (Synthetic Minority Over-sampling Technique) generates synthetic samples for the minority class, aiming to balance the class distribution in the training dataset.

Both approaches resulted in an increased f1_score, unlike what was observed for the logistic regressor. However, SMOTE yielded the best results, significantly enhancing the prediction quality.

By calculating the importance of each feature in the prediction, we ranked the features from most to least significant. Using a greedy approach, we recursively selected the top n most significant features, incrementally increasing n. The selection of the 26 most significant features yielded an f1_macro score of 0.757, representing the best result achieved on the online platform.

As a different approach to the previous feature selection, we applied PCA to reduce the dimensionality of our data. The configuration which obtaied the best score on the "dev" dataset was the following: Best Parameters: rf max depth: None, rf n estimators: 200, pca components: 40. Best Cross-validation Accuracy: 0.830.

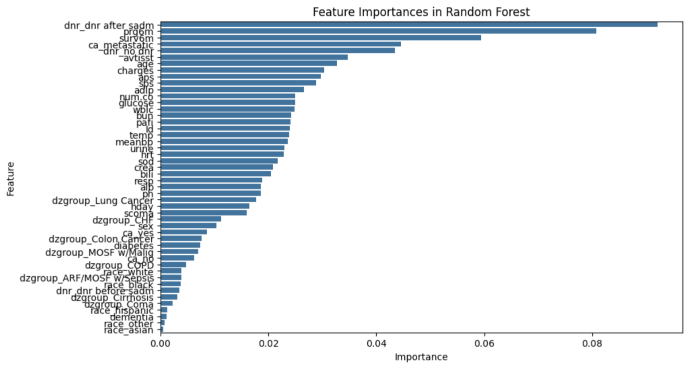
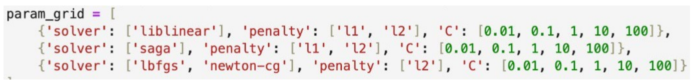


Fig. 6. Feature Importances by Random Forest

C. Hyperparameters tuning

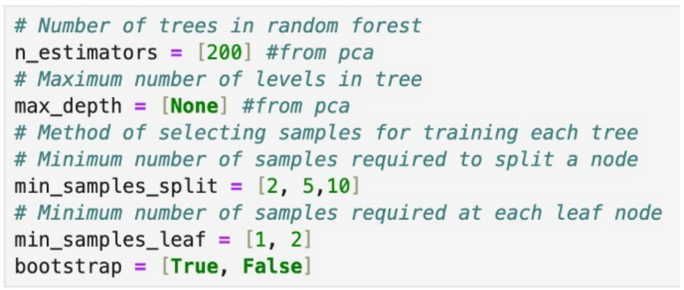
Logistic regressor Utilizing the subset of optimized features identified earlier, and some of the hyperparameters found by the exploration of the best PCA, an exploration of the best hyperparameter configurations was conducted through GridSearchCV coupled with 5-fold cross-validation. The culmination of this process yielded a notable result of $f1\_macro = 0.844$ on the "dev" dataset.



```
param_grid = [
    {'solver': ['liblinear'], 'penalty': ['l1', 'l2'], 'C': [0.01, 0.1, 1, 10, 100]},
    {'solver': ['saga'], 'penalty': ['l1', 'l2'], 'C': [0.01, 0.1, 1, 10, 100]},
    {'solver': ['lbfgs', 'newton-cg'], 'penalty': ['l2'], 'C': [0.01, 0.1, 1, 10, 100]}
```

Fig. 7. Logistic Regression parameters

Random forest Upon adjusting the hyperparameters of the random forest, we observed notable enhancements in the local f1_macro score. However, this improvement did not translate to the online platform. Despite employing cross-validation to minimize overfitting risk, the discrepancy in outcomes is attributed to the variance in parameter optimization criteria between the local environment and the online platform.



```
# Number of trees in random forest
n_estimators = [200] #from pca
# Maximum number of levels in tree
max_depth = [None] #from pca
# Method of selecting samples for training each tree
# Minimum number of samples required to split a node
min_samples_split = [2, 5,10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2]
bootstrap = [True, False]
```

Fig. 8. Initial parameters for Random Forest

III. Results

After all the hyperparameter tuning, we found the best parameters for both logistic regression and Random Forest as:

TABLE I
PARAMETERS OF THE TRAINED MODELS

Model	**Parameters**
Logistic Regression	C: 0.1, penalty: l2, solver: liblinear
Random Forest	bootstrap: False, criterion: entropy, max depth: None, max features: None, min samples leaf: 2, min samples split: 2, n estimators: 20

IV. Discussion

The evaluation the random forest classifier has shown superior performance compared to the logistic regressor. The use of SMOTE as a balancing technique significantly improved the prediction quality of the evaluation dataset. Despite the challenges with PCA and hyperparameter tuning, the careful selection of significant features was the only way to improve the score on the platform. Despite the number of components in combination with cross-validation, we failed to improve the score on the ”eval” dataset. This can either indicate overfitting in the local environment or a different data distribution from the ”dev”.

References

[1] Xie, L., Xing, E. P. (2020). Explainable Deep Learning: A Field Guide for the Uninitiated. Journal of Machine Learning Research, 21, 1-27.