Prediction of Mortality Rate of Heart Failure Patients Admitted to ICU

Name:	**Siddhi Pravin Lipare**
Registration No./Roll No.:	21262
Institute/University Name:	IISER Bhopal
Program/Stream:	DSE
Problem Release date:	August 17, 2023
Date of Submission:	November 19, 2023

1 Introduction

The objective of this Binary Classification project is to predict the factors of hospital mortality for the patients who are admitted to ICUs (Intensive Care Units) due to heart failure. The data is collected from MIMIC-III database (version 1.4, 2016), containing de-identified data on 46,520 patients and 58,976 admissions to the ICU of the Beth Israel Deaconess Medical Center, Boston, USA, between 1 June, 2001 and 31 October, 2012. The dataset consists of 48 features (columns) and 1058 data points. There are a total of 1723 missing values. Further, the dataset is imbalanced, having 915 samples corresponding to class 0 and 143 samples corresponding to class 1. 10 features in the dataset are categorical (0's and 1's). 1.

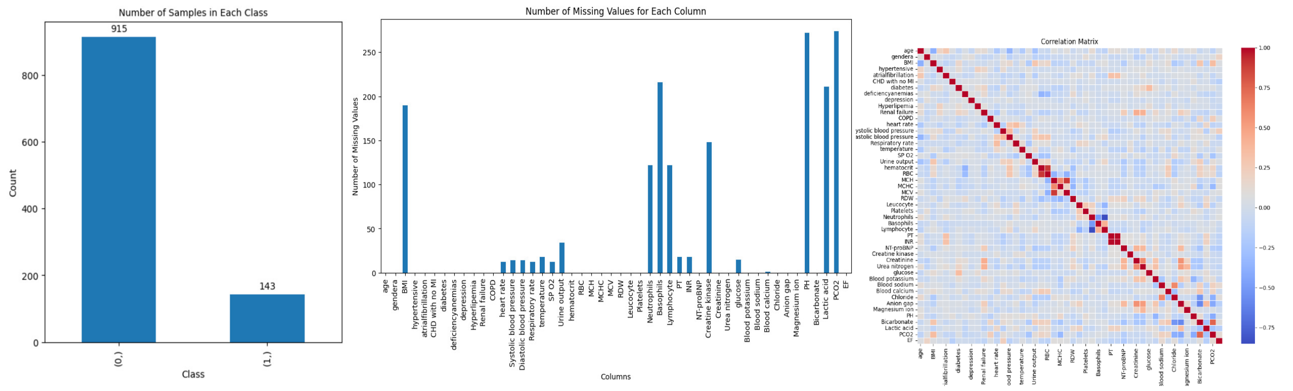

Figure 1: Overview of Data Set

2 Methods

2.1 Preprocessing phase:

- **Handling Missing Values:** Mean, Median, Mode and kNN Imputation: These methods involve replacing missing values with the mean, median, mode or the k Nearest neighbour of the respective feature. Since the 10 categorical features did not have any missing values, there was no need to apply the mode imputation. kNN Imputation gave the best results at k=3.

- **Feature Selection:** Feature selection was conducted using Mutual Information, ANOVA (Analysis of Variance) F1 Score, Chi-Squared, and Recursive Feature Elimination using Cross Validation. ANOVA F1 Score is applied to assess the variance between group means and F1 Score is used to evaluate feature importance for classification tasks. Mutual Information measures the dependency between two variables and is used for feature selection by identifying the most informative features. Chi-Squared is a statistical test applied to categorical features to evaluate the independence between features and the target variable. Recursive Feature Elimination (RFE) with Cross-Validation is an iterative method that recursively removes less important features, optimizing model performance through cross-validation.

- **Handling Categorical Features:** One Hot Encoding is a method to convert categorical variables into binary vectors, allowing the classifiers to interpret and utilize them effectively. I have applied One Hot Encoding function to the 10 categorical features in my data.

2.2 Model Training:

- **Classification:** Decision Tree, Random Forest, Support Vector Machine (SVM), kNN, Linear SVC, Logistic Regression, AdaBoost, and XGBoost classifiers were chosen to capture different aspects of the data, each with their own strengths and weaknesses.
- **Data Splitting:** Training and Validation Sets with Stratification: The dataset is divided into training and validation sets while ensuring that the class proportions are maintained (stratification), preventing biased model training.
- **Addressing Imbalanced Data:** Synthetic Minority Oversampling Technique (SMOTE) was employed to address imbalanced data, proving more effective compared to other oversampling methods such as ADASYN and Random Oversampling.
- **Hyperparameter Tuning:** Grid Search CV was used to perform an exhaustive search over a specified hyperparameter grid, optimizing the model's hyperparameters for better performance. This comprehensive approach ensures a robust data preprocessing and model training pipeline, considering different aspects such as missing data, feature importance, categorical variables, diverse classifiers, class imbalance, and hyperparameter tuning.

3 Experimental Setup

I have used numpy, pandas, matplotlib, seaborn and scikit-learn libraries for the project. I have evaluated the model using standard metrics such as F1-Score, Precision, Recall and Confusion Matrix. Accuracy is not a valid method of evaluation for a binary classification especially when classes are imbalanced since it does not provide how the model has dealt with the False Positives and Negatives. F1-Score provides the best insights especially when the classes are imbalanced. I have considered the macro averaged F1-Score since it considers the overall score for both the classes. For Hyper Parameter Tuning, GridSearchCV library was used and parameter grids were made for each classifier to check all the combinations and select the best one based on the maximum F1-Score (macro averaged). Feature Selection was done using SelectKBest function where score function was chosen from ANOVA F1 score, Chi2 and Mutual Information Score. Apart from Imputatations were done using the SimpleImputer and kNNImputer function to implement mean, median, mode and kNN imputation. kNN imputation gave the best results for most of the classifiers. There was no imputation required in any of the 10 categorical features. 'Age' feature was dropped since scaling would have made the feature continuous which is not practically possible in the case of age. Further, for feature selection, I plotted a Correlation heat map to analyse how all the pairs of features are correlated. There are only 5-6 pairs of features havig a coorelation greater than 0.6. This made it slightly difficult to select features of importance in the dataset.

4 Results and Discussion

Based on the results upon trying various imputers, classifiers and feature selectors, I've got the best F1-Score of 0.68 for the Adaboost Classifier using the kNN imputer. Similar results are achieved using the rest of the classifiers as shown in the table below. Only the best results for the best feature selector, SMOTE Oversampler and kNN Imputer have been displayed in the table after finding the best results for each classifier.

Table 1: Performance Of Different Classifiers Using All Features

Classifier	Feature Selector	Precision	Recall	F-measure
Adaptive Boosting	Mutual Information	0.80	0.64	0.68
Decision Tree	ANOVA F value	0.61	0.67	0.63
K-Nearest Neighbor	ANOVA F value	0.63	0.71	0.64
Logistic Regression	ANOVA F value	0.62	0.70	0.64
Random Forest	Mutual Information	0.69	0.59	0.61
Support Vector Machine	ANOVA F Value	0.63	0.65	0.64
XGBoost	Mutual Information	0.60	0.58	0.59

Table 2: Confusion Matrices of Different Classifiers

Actual Class	Predicted Class	
	1	0
1	179	4
0	20	9

Adaptive Boosting

Actual Class	Predicted Class	
	1	0
1	161	22
0	17	12

Decision Tree

Actual Class	Predicted Class	
	1	0
1	144	39
0	12	17

K-Nearest Neighbor

Actual Class	Predicted Class	
	1	0
1	142	41
0	12	17

Logistic Regression

Actual Class	Predicted Class	
	1	0
1	177	6
0	23	6

Random Forest

Actual Class	Predicted Class	
	1	0
1	168	15
0	22	7

XGBoost

Actual Class	Predicted Class	
	1	0
1	161	22
0	17	12

SVM

5 Conclusion

In conclusion, our study on predicting mortality rates for heart failure patients admitted to the ICU has provided valuable insights. The Adaboost classifier stood out with a solid F1-Score of 0.68, especially when paired with kNN imputation and SMOTE for handling imbalanced data. I learned that dealing with missing values and class imbalance is crucial. Techniques like imputation and oversampling played a key role in improving model performance. For future research, exploring advanced feature engineering, utilizing larger datasets, and incorporating domain-specific knowledge could further refine predictive models. With a higher computational power, we can use other methods for hyper parametric tuning as well to check more parameters and values at a time.

6 References

[1] GitHub Repository For the Project GitHub

[2] XGBoost: A Scalable Tree Boosting System Tianqi Chen

[3] Simple techniques for missing data imputation: Kaggle

[4] XGBoost Documentation: XGBoost

[5] An Efficient Probabilistic Ensemble Classification Algorithm for Diabetes Handling Class Imbalance Missing Values: IEEE

[6] Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results IEEE

[7] Research and Application of AdaBoost Algorithm Based on SVM IEEE

[8] Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study IEEE