SUPPORT2: Study to Understand Prognoses Preferences Outcomes and Risks of Treatment Dataset Analysis Final Report

Jo Gasior-Kavishe
Brown University Class of 2025
Project GitHub Link

Introduction

Purpose

In this final project, the goal is to predict whether a terminally ill patient would have died or not in the hospital based on their demographics and health vitals. This question is important because in the United States of America, healthcare is by no means equal, and the quality of care given to patients can vary greatly across race, income bracket, gender, and various physiological factors. Moreover, the United States does not have a centralized healthcare system, and healthcare can differ across all fifty states. Because of these vastly different systems across the country, it is essential to categorize and recognize the different demographic and health signs that could predict whether a patient would die in a hospital or not. Knowing these predictors could greatly influence earlier decision-making in hospital care to prevent death.

SUPPORT2: Study to Understand Prognoses Preferences, Outcomes, and Risks of Treatment Dataset Analysis Final Report[1]

SUPPORT2 (Study to Understand Prognoses, Preferences, Outcomes and Risks of Treatment) is a dataset composed of 9105 individual critically ill patients across 5 United States medical centers, accessed throughout 1989-1991 and 1992-1994. Each patient's demographics and vitals were measured only once in the dataset, as there were 9105 unique patient IDs, and there was no record in the dataset indicating what year that patient's information was collected nor which hospital they came from.

Target Variable and Features

SUPPORT2 is a classification dataset, and the problem of predicting whether a patient would have died in the hospital is a binary classification problem. The target variable, 'hospdead,' indicates whether the patient died in the hospital (0 for lived and 1 for died). The dataset has 9105 instances and forty-seven features: five discrete, thirty-three continuous, eight categorical, and one binary feature. The SUPPORT2 dataset for this project was acquired from the University of California, Irvine Machine Learning Repository, which was donated to the repository by the Vanderbilt University Department of Biostatistics.

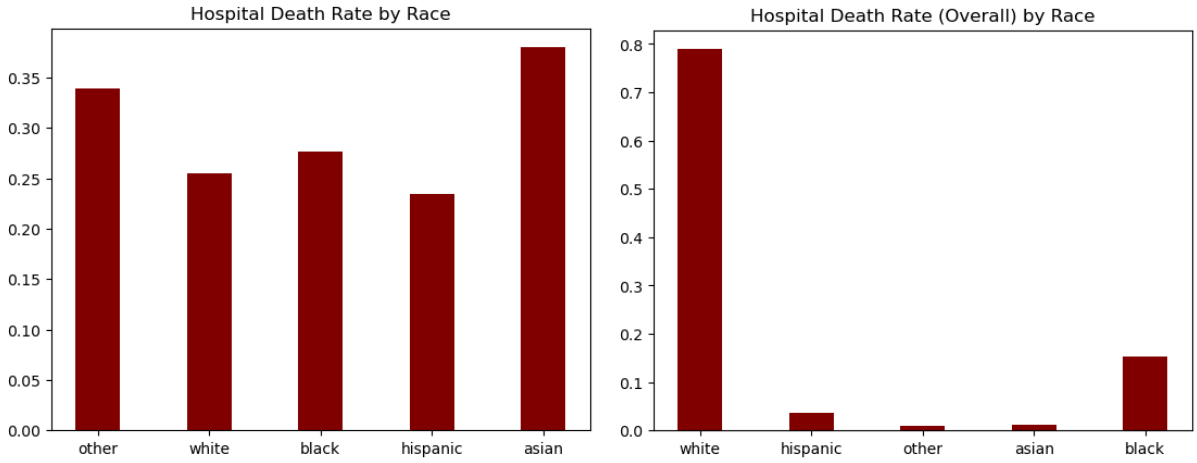
[1] Professor Frank Harrell 2022, url: https://hbiostat.org/data/

Current Research

Previous work with this dataset included predicting whether a patient would die overall (not died in the hospital) or determining these patients' 2- and 6-month survival rates. However, no published work has yet been found about previous results when predicting hospital death.

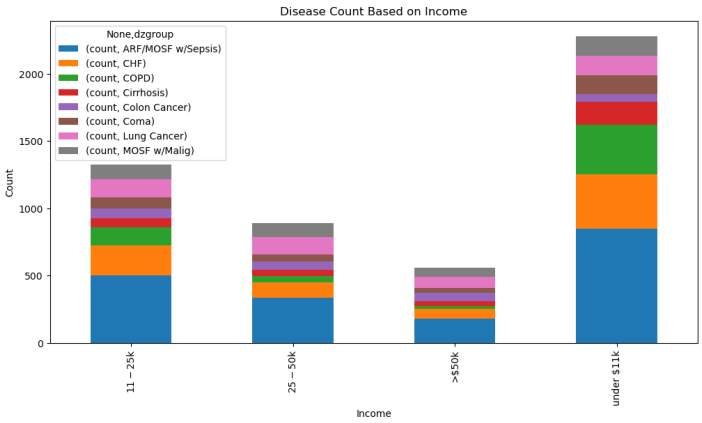
Exploratory Data Analysis

In my Exploratory Data Analysis (EDA), I decided to examine various hospital death rates across race, gender, and income and explore any factors that could influence hospital death prediction.

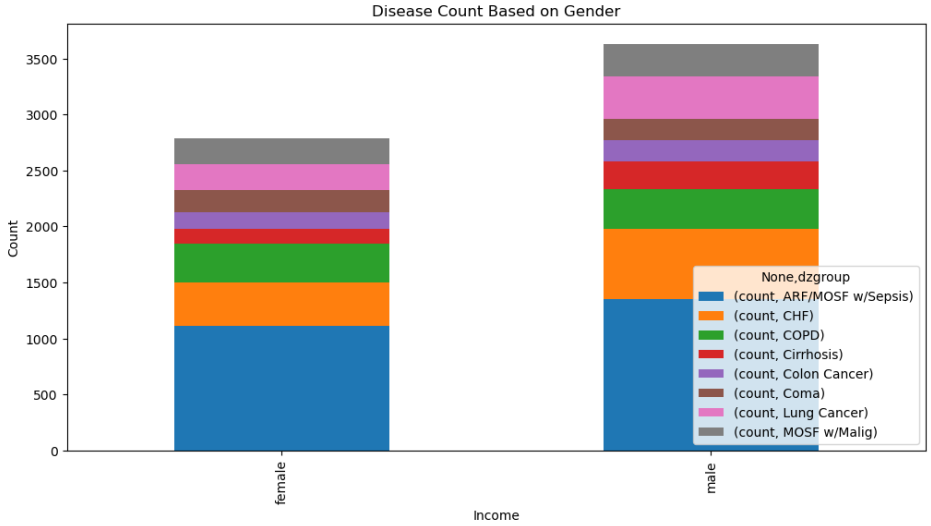


The bar graphs above analyze the various death rates across races; here, in the first graph, the hospital death rate by race is calculated proportionally by race (total number of patients by race who died divided by the total number of patients of that specific race in the dataset. Out of all of the races patients who were White, Hispanic, Black, and Other identifying had similar death rates (~25%, ~26%, ~27% and ~28%, respectively); however, what was surprising was that patients who were Asian had a higher hospital death rate of ~38%. The bar graph to the right analyzes the overall death rate (total number of patients by race who died divided by the total number of deaths). It is seen that out of all of the patients who had died, ~78% of them were White, ~3.6% were Hispanic, 0.9~ were marked as Other, 1.1~ were Asian identifying, and ~15.3% were Black identifying.

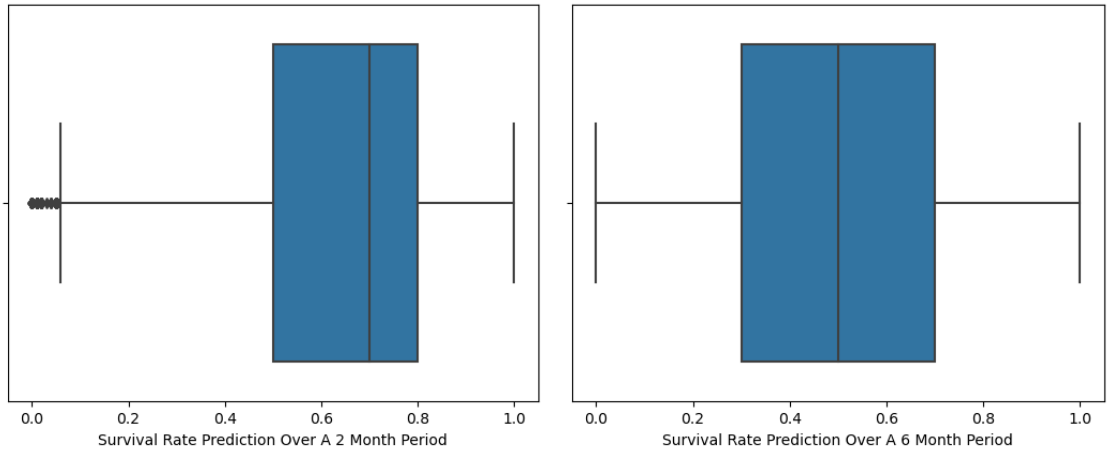
Next, when looking at the disease distribution across income levels, we can see a difference in the income brackets regarding the amount of diseases each has.



There also is a slight difference in the disease counts based on gender.



Next, I wanted to see how the distribution of the predicted two-month versus six-month survival rate was, and we can see that generally, patients were predicted to survive at a higher rate over two months versus over six months.



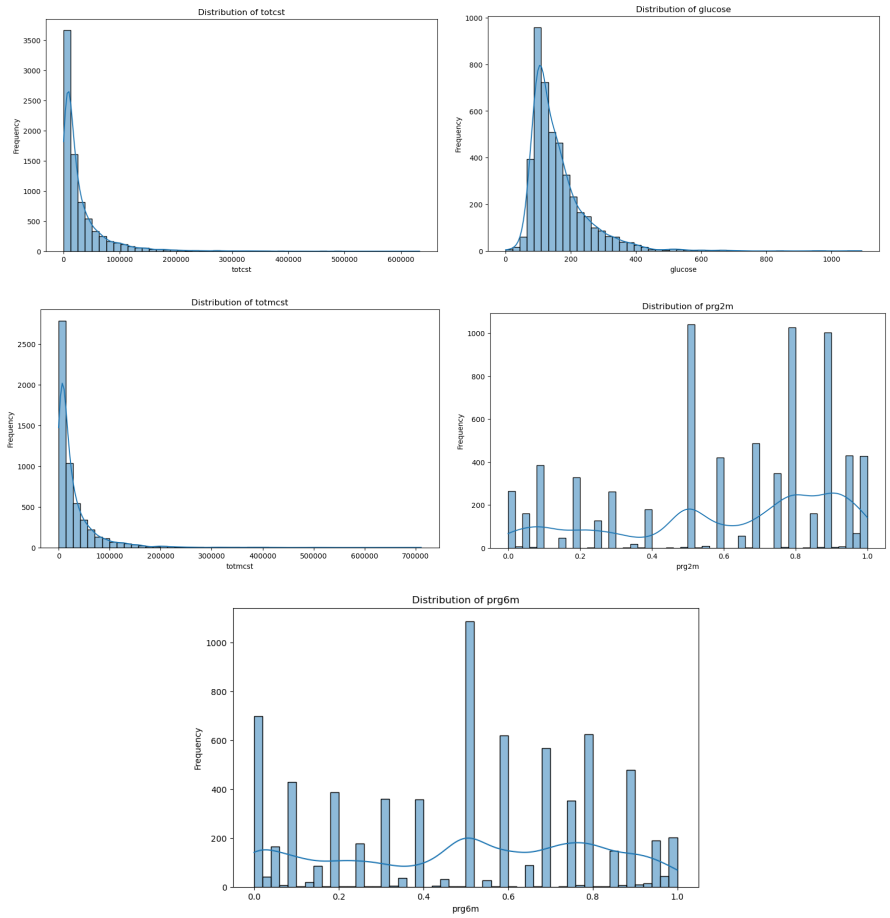
Methods

Splitting Strategy

I trained five models for the binary classification problem: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Classifier, and XGBoost Classifier. Because the SUPPORT2 dataset had different amounts of data points in terms of gender, race, income, and disease group, the dataset was relatively imbalance, so I decided to train one set of the models using a stratified shuffle split so that the data could be shuffled around. However, according to the UCI Machine Learning Repository, where I got the dataset, the authors stated that using a normal train test split was also a sufficient strategy. So, I trained one set of the five models using a Stratified Shuffle Split for my own curiosity, and I trained another five of them using a normal train test split. The Stratified Shuffle Split models were split into three datasets: other and test, and then, the training and validation data was split from the other dataset. The normal train test split was also split into the four datasets.

Data Preprocessing

One of the main challenges of this dataset was the amount of missing data present within it. About 96% of the rows in the entire dataset had at least one missing value, so I needed to evaluate what my imputation strategies would truly be. If I were to drop all the rows with missing data, the dataset would go from 9105 rows to 306 rows, a significant drop I did not want to train my models on. First, I dropped all rows in features with less than 100 missing values. Then, my biggest challenge with the data imputation was the education and income features; because medical data is so sensitive, I did not feel comfortable imputing any demographic data, so I dropped all those rows with missing education and income values. This brought my dataset size down to about 5049 rows. I was lucky that the UCI Machine Learning Repository gave baseline values to fill in various NaN features. For other features, I had to look at the distribution of their values, and since they were numerical, I decided to use either median or mode imputation.



Here above are a couple of feature distributions. If there was a skewed distribution, I used a median; if there was a normal distribution, I used a mean.

I used a column transformer to handle all of my transformations for my data preprocessing since I had many features. I had an ordinal encoder, one hot encoder, and a standard scaler within this column transformer. My ordinal encoder was used on the income column, my one hot encoder was used on features with string values such as sex, disease group, race, cancer, and resuscitation. Then, I used a standard scaler for the rest of the continuous or integer features since I knew the models I would be using would be sensitive to features with various distributions. I put the preprocessor into a pipeline and then fit transformed my training data and transformed my validation and testing data.

I used various parameters for each of my models. I wanted to encapsulate a wide range.

Model	Tuned Parameters
Logistic Regression	'penalty': ['l1', 'l2', 'elasticnet', 'none'], 'C': logspace(-4, 4, 20), 'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], 'l1_ratio': linspace(0, 1, 10)
Decision Tree Classifier	'criterion': ['gini', 'entropy', 'log_loss'], 'splitter': ['best', 'random'], 'max_depth': [None, 10, 30, 50], 'min_samples_split': range(2, 10), 'min_samples_leaf': range(1, 5), 'max_features': ['sqrt', 'log2', None], 'max_leaf_nodes': [None, 10, 30, 50], 'min_impurity_decrease': [0, 0.01, 0.1]
Random Forest Classifier	'n_estimators': [100, 300, 4500], 'max_features': ['sqrt', 'log2', 0.2, 0.5, 0.8], 'max_depth': [None, 10, 30, 50], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False], 'criterion': ['gini', 'entropy']
Support Vector Classifier	'C': [0.1, 1, 10, 100, 1000], 'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'degree': [2, 3, 4], 'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1], 'coef0': [0, 0.5, 1]
XGBoost Classifier	'learning_rate': [0.01, 0.1, 0.2, 0.3], 'n_estimators': [100, 200, 300], 'max_depth': [3, 4, 5, 6, 7], 'min_child_weight': [1, 3, 5], 'subsample': [0.6, 0.7, 0.8, 0.9], 'colsample_bytree': [0.6, 0.7, 0.8, 0.9], 'gamma': [0, 0.1, 0.2, 0.3, 0.4]

<Table 1. Parameter Values for each Machine Learning Algorithm>

The metric used to evaluate my models' performances was accuracy. Each model went through 3-5 random training states, depending on how computationally expensive it was to run that model. All of the test scores for all the parameters were then averaged out, and a standard deviation was calculated. My models for my stratified shuffle split and regular train test split have similar results.

Results

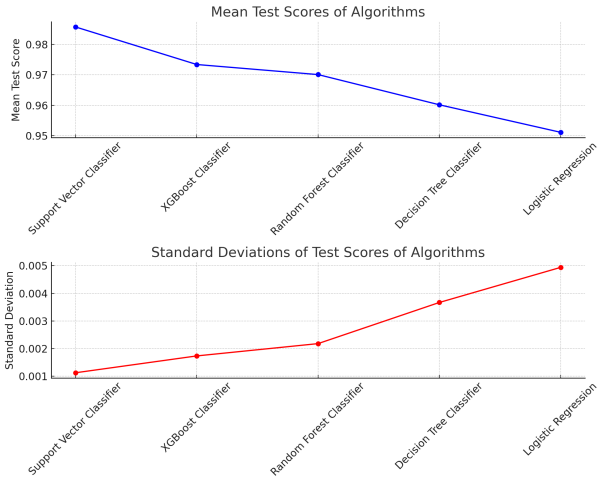
Model	Mean Test Score Result (Accuracy)	St.D. of test scores
Support Vector Classifier	0.9856985698569857	0.0011218964826386654
XGBoost Classifier	0.9720572057205721	0.002178106696724257
Random Forest Classifier	0.9689768976897689	0.001943181708762996
Decision Tree Classifier	0.9610561056105611	0.0024302224460258
Logistic Regression	0.9572277227722772	0.005220784140801375

<Table 2. Stratified Shuffle Split Model Accuracies for each Machine Learning Algorithm>

Model	Mean Test Score Result (Accuracy)	St.D. of test scores
Support Vector Classifier	0.9856985698569857	0.0011218964826386654
XGBoost Classifier	0.9733773377337734	0.0017324549777803656
Random Forest Classifier	0.9700770077007701	0.002178106696724223
Decision Tree Classifier	0.9601760176017602	0.0036684998901062717
Logistic Regression	0.9511551155115512	0.004939481698711486

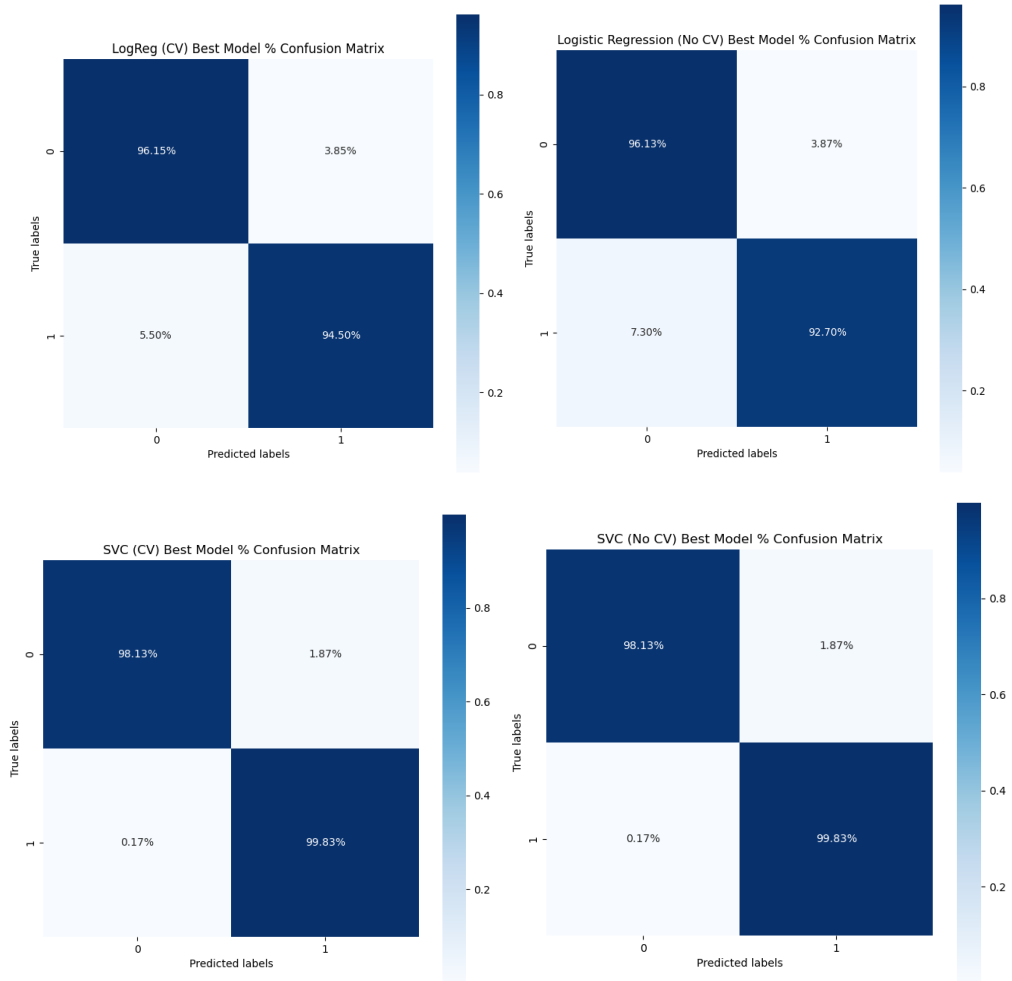
<Table 3. Regular train test split Model Accuracies for each Machine Learning Algorithm>

I was surprised to see that most of my models all performed slightly better if not the same with no stratified shuffle splitting compared to stratified shuffle splitting. My support vector classifier had the highest predictive power of around 98.5% accuracy across both the stratified and regular splitting methods.



Above are the models' mean test scores and standard deviations without stratified shuffle splitting. To interpret my models, I calculated a confusion matrix for my best and worst models, which were the support vector classifier and the logistic regression model.

<Table 4 : Confusion Matrices using the worst and best-performing models>



Outlook

Overall, my support vector classifier was the model that consistently performed the best, and I am quite happy with its predictive performance. I would have tried to do more feature engineering to improve my model's performance. Since there were 47 features originally, it could have potentially been beneficial to combine any of those features into a general overall health/quality of life score; this would have significantly decreased the dimensionality. Moreover, I would have used more advanced data imputation techniques. Since my models' predictive power was already high with this median and mode imputation, I decided not to do any more imputation; however, should they have had low accuracy at the beginning, I would have chosen different imputation techniques.

Since medical data is extremely sensitive ethically, I could have trained an XGBoost model on the missing data points and not done any imputation. Should I want to improve on this model, that would be another approach I would have taken.

References

Professor Frank Harrell 2022, url: https://hbiostat.org/data/