Task 1 - Data science course

1
General Description of the Proposed Pipeline

The data analysis pipeline can be broken down into the following main steps:

1. **Loading the Dataset:** The data is loaded from a CSV file into a Pandas DataFrame. This data includes both training and test datasets. An initial examination of the target variable 'hospital_death' is performed with a histogram to understand its distribution.

2. **Preliminary Data Analysis:** The data is analyzed to understand its structure, missing values, and the types of variables (numerical and categorical) present.

3. **Data Cleaning**: The missing values in the data are handled through imputation. Depending on the feature distribution, either mean or median imputation is applied. Any remaining missing values are filled using backward and forward filling methods. Additionally, irrelevant columns are dropped.

4. **Feature Engineering**: New features are created based on existing features which are indicative of certain health conditions.

5. **Encoding and Binning**: Categorical features are one-hot encoded, and numerical features (excluding those with "id" in their names) are binned into three categories using a uniform strategy. One-hot encoding is used to convert categorical data into a format that could be provided to ML algorithms to improve prediction results. Binning is a data pre-processing technique used to reduce the effects of minor observation errors.

6. **Feature Selection**: We used SHAP (SHapley Additive exPlanations), for feature selection and importance visualization.
   The process starts by training an XGBoost classifier on the data. SHAP values, representing each feature's contribution to the prediction for each data instance, are then computed. Features are ranked by their mean absolute SHAP value, and the top 50% are retained for further processing. This approach focuses on the most impactful features and reduces the data's dimensionality, potentially improving model performance.

7. **Model Training and Evaluation**: Various classifiers are trained on the dataset, including AdaBoost, Random Forest, Naive Bayes, XGBoost, and Decision Tree. For each model, hyperparameters are tuned to get the best performance. The models are evaluated using cross-validation, and various metrics like AUC, accuracy, recall, and precision are calculated. Confusion matrices and running times for each classifier are also calculated and displayed.

8. **Unlabeled Data Processing**: The same preprocessing steps (cleaning, feature engineering, encoding, and binning) are applied to the unlabeled (test) data. This ensures that the test data is in the same format as the training data.

9. **Prediction on Unlabeled Data**: The best-performing model makes predictions on the unlabeled data. The prediction results are saved and can be used for further analysis.

10. **Result Analysis and Visualization**: The results are analyzed and visualized using histograms and other suitable methods. The accuracy of the model's predictions is assessed, and important insights are drawn from the work.

The data preprocessing consisted of several steps designed to clean, transform, and reduce the dimensionality of the data, preparing it for modeling.

2
The results of the preliminary analysis

In the initial phase of our exploration, we executed several important steps to understand our data.

Data Structure and Quality: We started by examining the structure of the training data, which comprises a certain number of instances and features. To assess the completeness and quality of our data, we identified and visualized the number of missing values across different columns. A bar plot was constructed to clearly depict the columns with missing values and the extent of these missing values.

Target Variable Analysis: We then turned our attention to the target variable, hospital_death. By generating a histogram, we were able to understand its distribution. Interestingly, the histogram showed that a majority of instances in the dataset correspond to patients who did not experience a hospital death. This observation provides insights into the nature of the problem (classification or regression) and indicates a potential class imbalance problem that we talk about in the next sector.

Class Imbalance Consideration: Our preliminary examination of hospital_death suggests the possibility of class imbalance. We are considering using techniques such as Random Over-Sampling to balance the dataset. This technique would allow us to increase the representation of the minority class by creating copies of those instances.

Handling Missing Data: We identified the columns with more than 10% missing data and generated histograms to explore their distributions. We found some normally distributed features. In these cases, it's likely that we will use mean imputation, as it is a suitable strategy for normally distributed data. For other variables, which do not appear to follow a normal distribution, we anticipate using the median to impute missing values, as the median is less affected by outliers and is a suitable measure for non-normally distributed data.

3
A description of the logic of the pre-processing of the data you performed.

Data Cleaning and Imputation: The first step involved dealing with missing data. Two strategies were used to impute missing values: mean imputation and median imputation. The strategy used was dependent on the distribution of the data. Features with normal distributions were imputed with the mean, while the rest were imputed with the median. After imputation, any remaining missing values were dropped.

One-Hot Encoding: All categorical features were one-hot encoded to convert them into a format that can be understood by the machine learning algorithms. This step transformed categorical data into binary vectors, which are easier for algorithms to process.

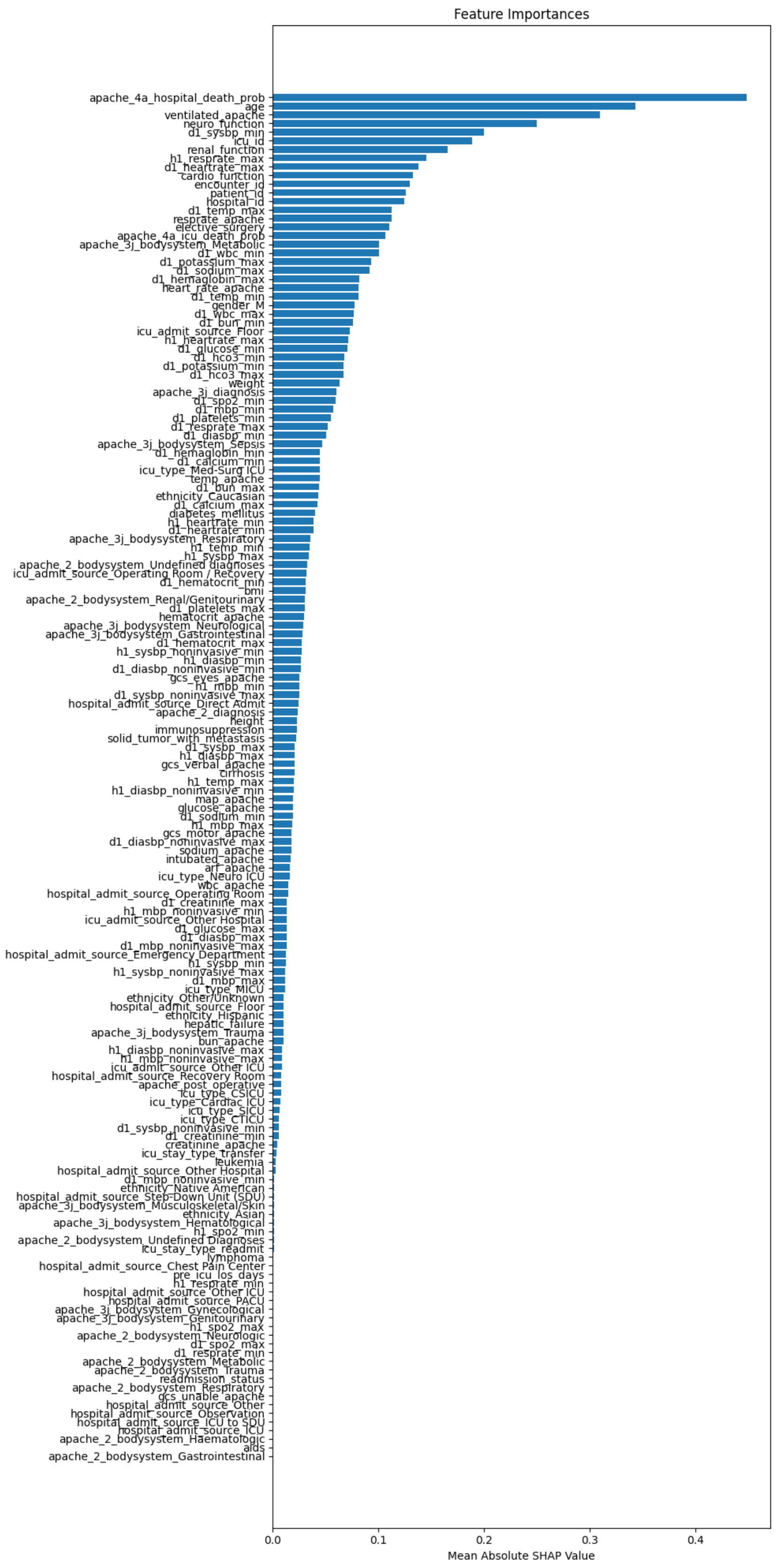
Feature Engineering: Three new features were created based on existing ones: 'renal_function', 'neuro_function', and 'cardio_function'. These features were created by performing operations on existing features, providing more insightful information about the patient's renal, neurological, and cardiovascular functions.

Data Binning: Numerical features (excluding those with "id" in their names) were binned into three categories using a uniform strategy. This step was done to convert continuous data into categorical data, reducing the potential effect of outliers and noise on the models.

Unnecessary Feature Removal: Any features in the unlabeled data that were not present in the training data were dropped. This was done to ensure that the training and test data had the same features.

Features ranking

As described above, we used SHAP (SHapley Additive exPlanations), for feature selection and importance visualization. The process starts by training an XGBoost classifier on the data. SHAP values, representing each feature's contribution to the prediction for each data instance, are then computed. Features are ranked by their mean absolute SHAP value, and the top 50% are retained for further processing. This approach focuses on the most impactful features and reduces the data's dimensionality, potentially improving model performance.



Classification models pipeline, each with its own specific hyperparameters

AdaBoost Classifier: The AdaBoost Classifier is a boosting ensemble method that works by combining multiple weak learners to create a strong learner. The hyperparameters used for this model were algorithm='SAMME.R', learning_rate=1.0, and n_estimators=200.

Random Forest Classifier: Random Forest is an ensemble method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes or mean prediction of the individual trees. The hyperparameters used for this model were max_depth=7 and n_estimators=200.

Gaussian Naive Bayes: Gaussian Naive Bayes is a probabilistic classifier based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. The hyperparameter used for this model was var_smoothing=1e-09.

XGBoost Classifier: XGBoost is a framework that uses decision trees as base learners. The hyperparameters used for this model were learning_rate=0.1, max_depth=7, and n_estimators=500.

Decision Tree Classifier: Decision Tree is a non-parametric supervised learning method used for classification. The algorithm creates a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The hyperparameter used for this model was max_depth=7.

The models were evaluated using cross-validation on several performance metrics, including the area under the ROC curve (AUC-ROC), accuracy, recall, precision, and the time taken for evaluation.

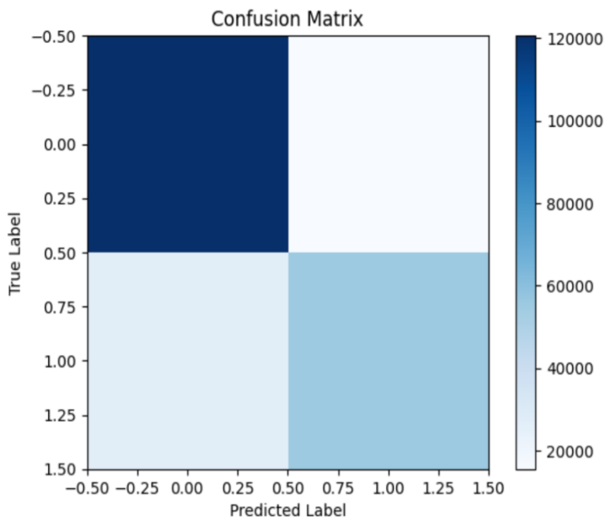
6

Results of the internal validation according to the cross-validation protocol

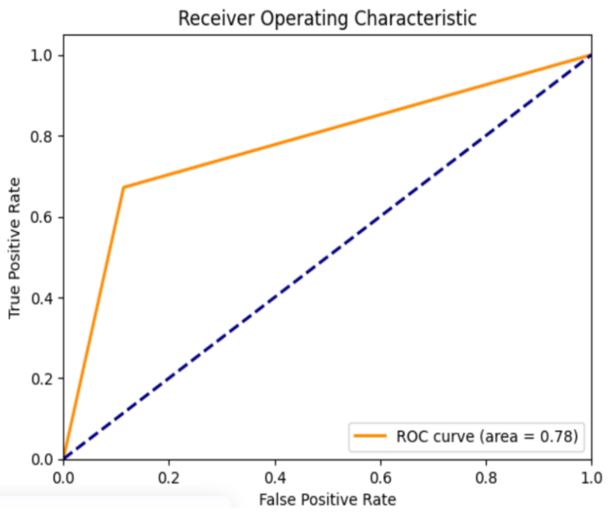
Classifier	AUC-ROC	Accuracy	Recall	Precision	Evaluation Time (seconds)
AdaBoost	0.775	0.802	0.668	0.773	294.01
Random Forest	0.751	0.794	0.579	0.815	139.57
Gaussian Naive Bayes	0.674	0.739	0.415	0.79	2.68
XGBoost	0.958	0.956	0.966	0.92	1605.53
Decision Tree	0.757	0.79	0.621	0.775	9.14

ADABoost

```
Confusion matrix:
```

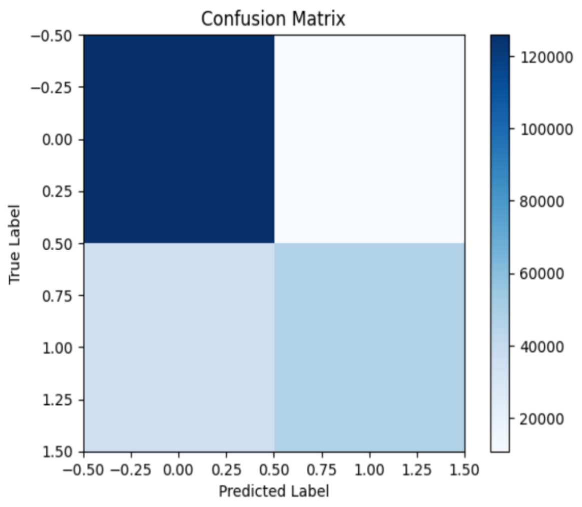


```
Roc curve:
```

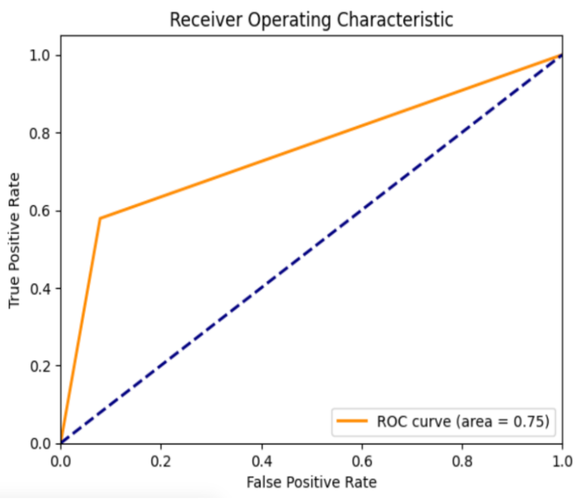


Random Forest

```
Confusion matrix:
```

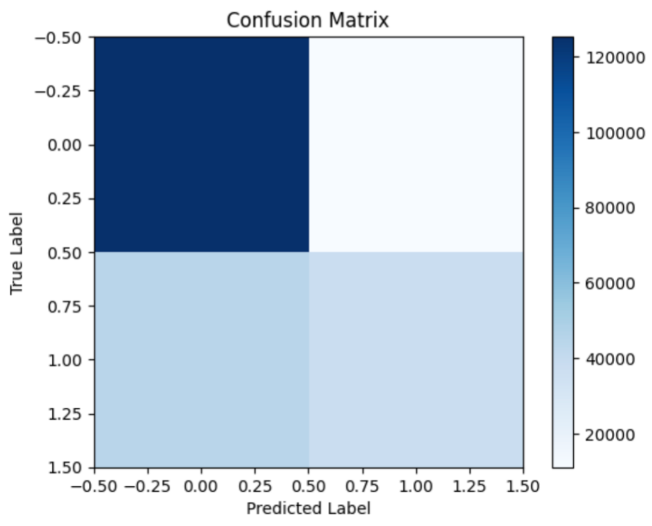


```
Roc curve:
```

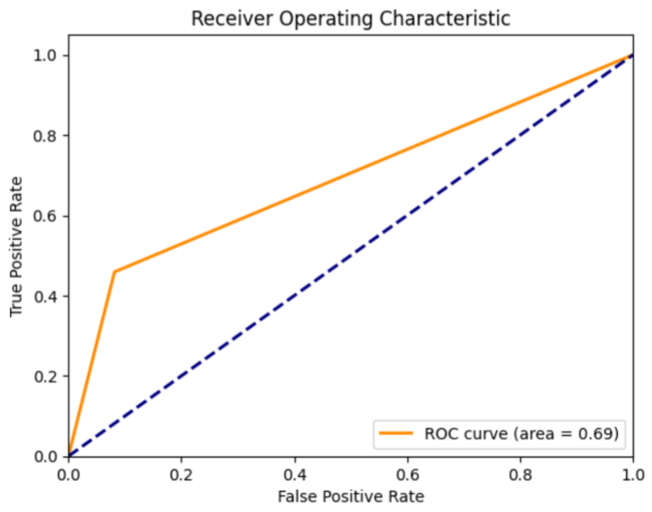


Naive Bayas

Confusion matrix:

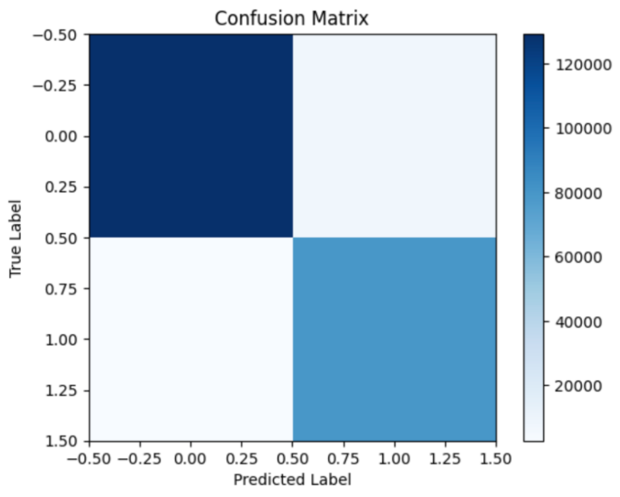


Roc curve:

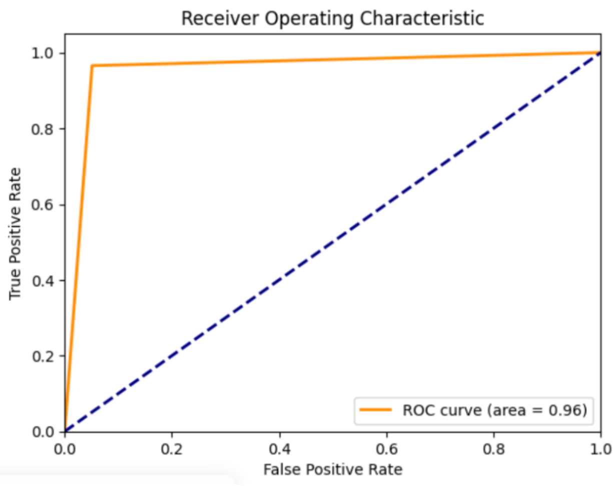


XGBoost

Confusion matrix:

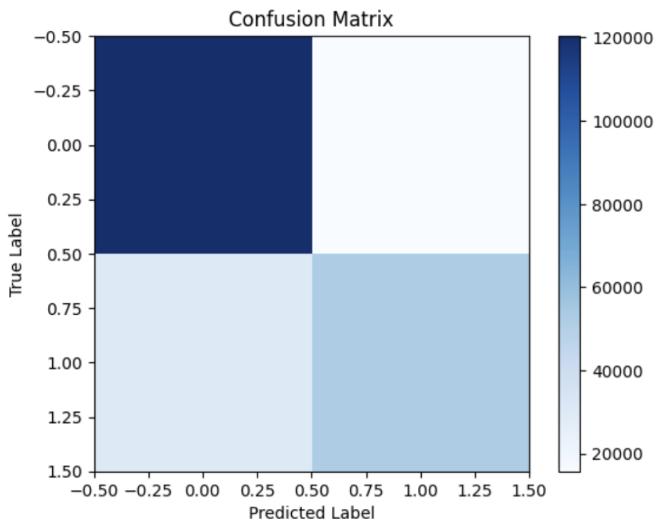


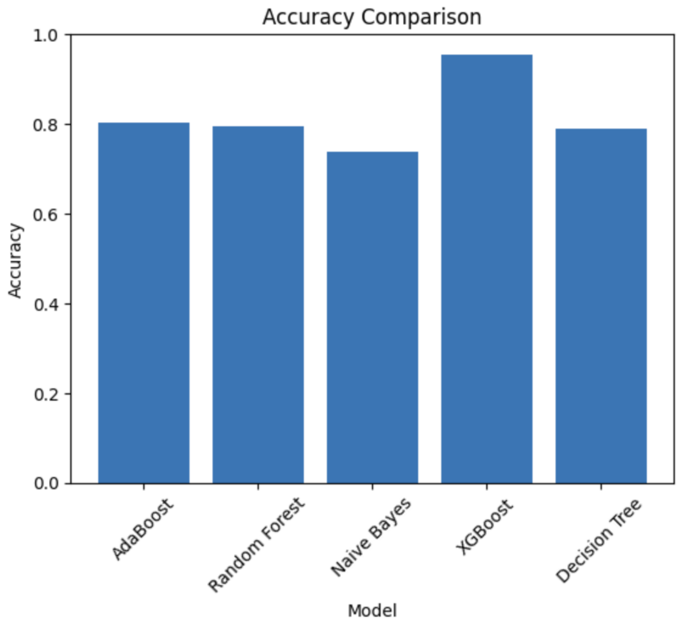
Roc curve:



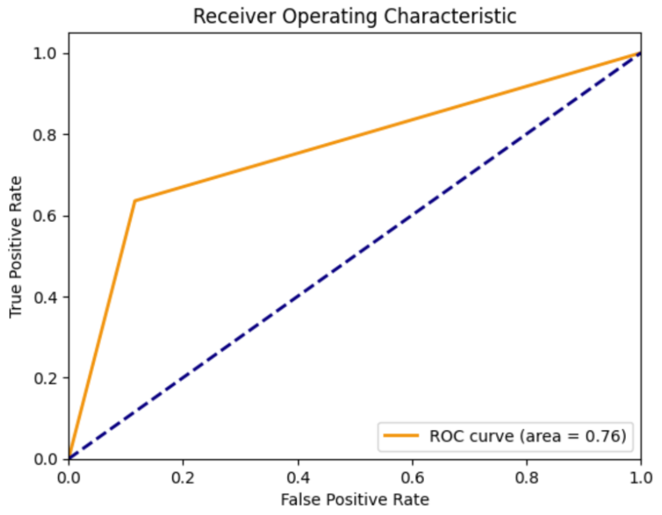
```
Confusion matrix:
```

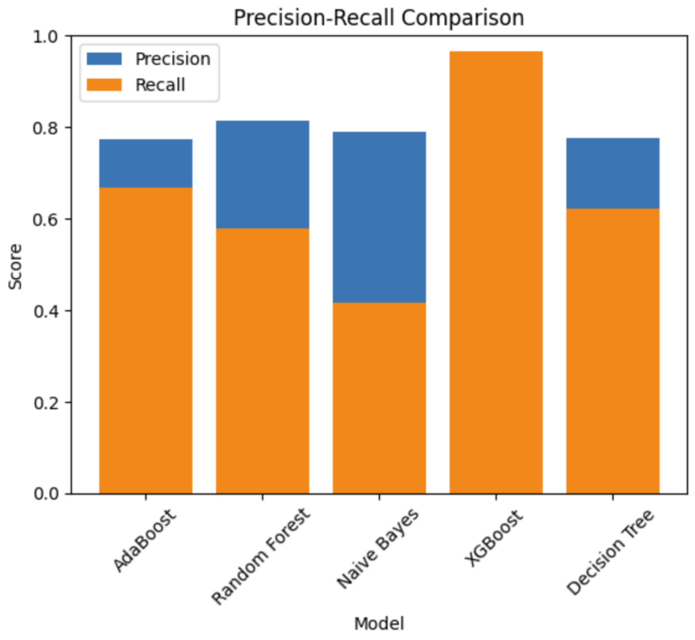
Decision Tree

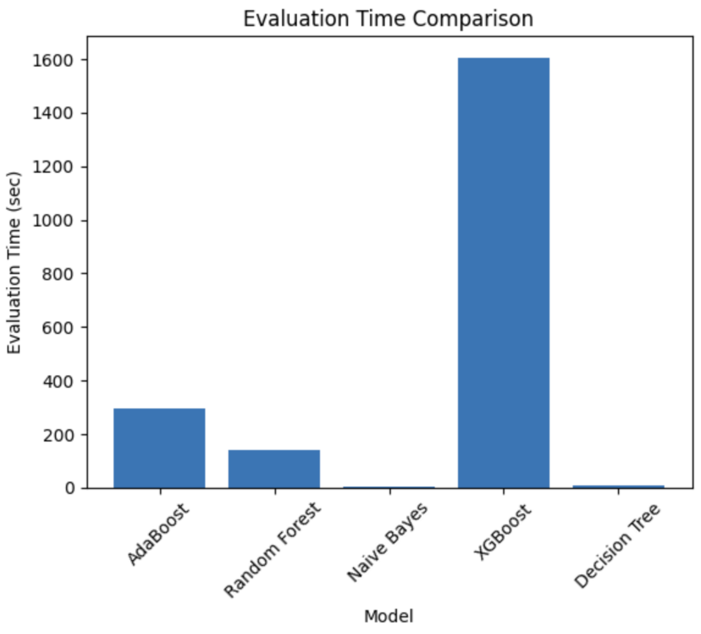




```
Roc curve:
```







7
Summary

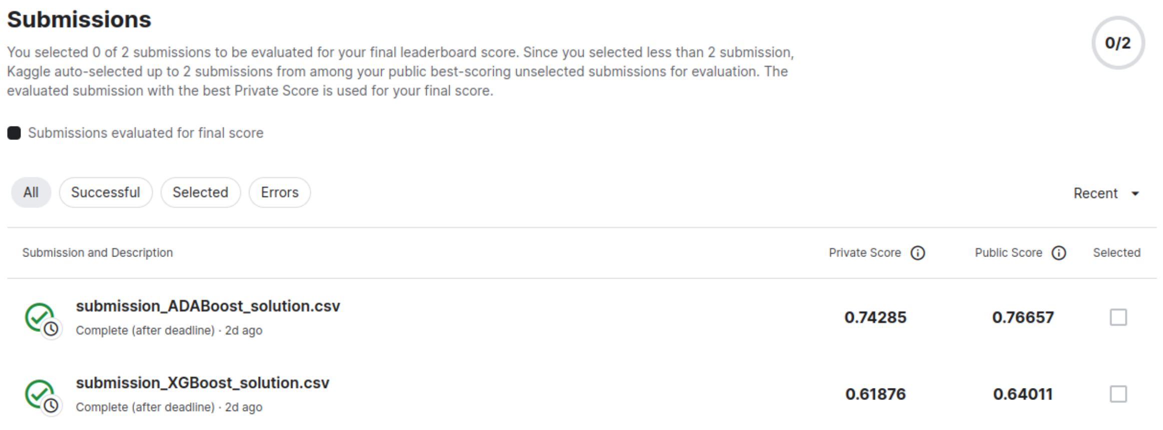
This project aimed to predict hospital deaths based on a range of clinical, demographic, and hospital-related features. We followed a structured pipeline that involved data preprocessing, feature engineering, feature selection, model selection, hyper-parameter tuning, and model evaluation, using several machine learning techniques.

We had though a few difficulties during the project, At first, we tried to remove a lot of features, cutting out a bunch of features from our dataset using SHAP for features that were in the lower 75% of the score.
It turns out that even the features we thought weren't important had some useful info. When we took them out, our models didn't do so well. That's when we knew we had to rethink our strategy. So, instead of removing these features, we kept the top 50%, making the model scores much better.

Another thing we learned was about data balancing. Our dataset had way more survivors than fatalities. We didn't think much of it at first, but it became a problem when our models started predicting way more survivors because of this imbalance. To fix this, we had to use techniques like Random Oversampling to balance out the data. This made our models better at predicting both survivors and fatalities.

8
kaggle results

For our final estimations we chose to focus on 2 models: AdaBoost and XGBoost.
We uploaded the unlabeled dataset with the predicted results and those are the result via kaggle:



Submissions

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

0/2

● Submissions evaluated for final score

All | Successful | Selected | Errors — Recent

Submission and Description	Private Score	Public Score	Selected
submission_ADABoost_solution.csv Complete (after deadline) · 2d ago	0.74285	0.76657	☐
submission_XGBoost_solution.csv Complete (after deadline) · 2d ago	0.61876	0.64011	☐

AdaBoost model score: 0.76657, XGBoost model score: 0.64011
AdaBoost model has better score and we can assume that our XGBoost model suffers from overfitting.