Heart Failure Prediction:

Nancy Akoum 202200112

Mohammad Al Izzi 202104792

8 Dec,2023

Contents

Problem Statement:

In light of the abundant medical data available and the increasing prominence of Data science, numerous startups are embracing the challenge of developing predictive indicators for potential diseases. Globally, cardiovascular diseases (CVDs) stand as the leading cause of death, claiming approximately 17.9 lives annually, constituting 31% of all overall deaths. Heart failure, a common consequence of CVDs, necessitates early detection and management, particularly for individuals with cardiovascular disease or those at high risk due to factors like hypertension, diabetes, hyperlipidemia, or pre-existing conditions. Leveraging machine learning models can prove instrumental in automating the detection and management of these conditions, demonstrating how AI techniques can be employed to address and mitigate health challenges, ultimately paving the way for tackling the next set of issues.

Aim:

The primary objective is to develop a binary classification model capable of predicting whether a patient is prone to heart failure. This involves the utilization of multiple numerical and categorical features, including age, sex, chest pain type, resting blood pressure, cholesterol levels, and more.

Dataset & Code:

The Google Colab notebook used in this machine learning project, along with the dataset in CSV format, is accessible through the following links:

1. Google Colab Notebook: https://colab.research.google.com/drive/1E_oAf2WZzDObJ459RIU1c78TlE1Rncsi?usp=sharing

2. Dataset in CSV format: https://drive.google.com/file/d/16kmwxoMrt4QnsS3ESSOXej1cG1UcGKKd/view?usp=sharing

Dataset Attributes:

- Age: Age of patients (Years)
- Sex: Sex of patients (M: Male, F: Female)
- ChestPainType: chest pain types are classified as follows:
    - TA: Typical Angina
    - ATA: Atypical Angina

    - NAP: Non-Anginal Pain
    - ASY: Asymptomatic
- RestingBP: resting blood pressure (mm Hg)
- Cholesterol: serum cholesterol (mm/dl)
- FastingBS: fasting blood sugar (mg/dl)
- RestingECG: resting electron cardiogram result:
    - Normal: Normal
    - ST: having ST-T wave abnormality (T wave inversions and/or ST elevation and/or depression ST elevation or depression of > 0.05 mv
    - LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria
- MaxHR: maximum heart rate achieved (numeric value between 60 and 202)
- ExerciseAngina: exercise-induced angina (Y: Yes; N: No)
- Oldpeak: numeric value measured in depression
- ST_Slope: the slope of the peak exercise ST segment
    - Up: up sloping
    - Flat: flat
    - Down: down sloping
- HeartDisease: output class
    - 1: heart disease
    - 0: Normal

Methodology:

1. Data Analysis and Visualization:

1.1. Loading and Previewing the Data:

In this phase, we loaded the heart health dataset, which contains essential information about patients' health attributes.

Using the Pandas library in Python, we imported the data and previewed the first few rows to gain an initial understanding of the dataset's structure.

This exploration provided insights into the types of features available, such as age, sex, cholesterol levels, and more.

1.2. Dataset Overview:

To obtain a holistic overview of the dataset, we conducted a thorough examination of its basic characteristics. This included determining the number of instances (individual patients' records) and attributes (distinct health-related features). By using Pandas

functions, we accessed information about data types, the presence of missing values, and the overall shape of the dataset. Understanding these aspects is crucial for informed decision-making throughout the machine-learning process.

1.3. Handling Missing Values:

Data quality is a critical consideration in any analysis. We implemented a heatmap visualization to identify missing values within the dataset. The heatmap visually represents missing values, allowing us to pinpoint specific attributes that might require attention during the data preprocessing stage. The identification of missing data is essential for deciding on appropriate strategies, such as imputation or exclusion, to ensure the reliability of our models.

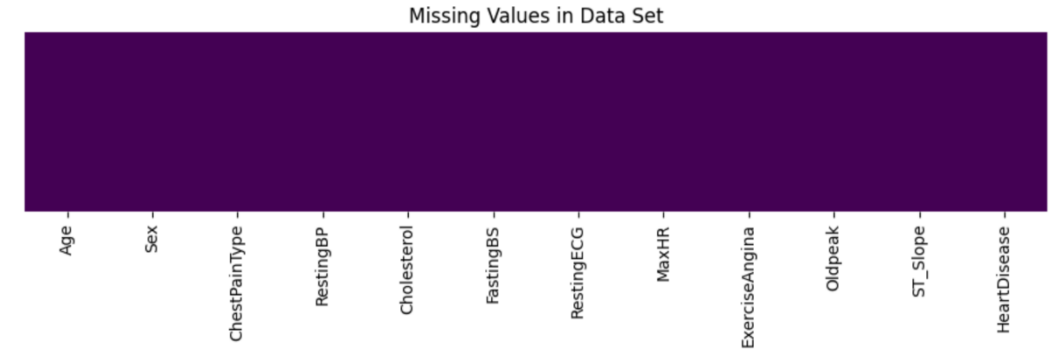


Figure1.1: Missing Values heatmap

1.4. Correlation Matrix:

To unravel relationships between different health attributes, we calculated and visualized a correlation matrix. This matrix illustrates the degree of association between pairs of numerical variables. By leveraging the Seaborn library, we created a heatmap that visually represents the correlation coefficients. This exploration is valuable for feature selection, providing insights into potential redundancies or strong associations between attributes. Understanding feature correlations informs our choices during model development and enhances the interpretability of the final machine-learning model.

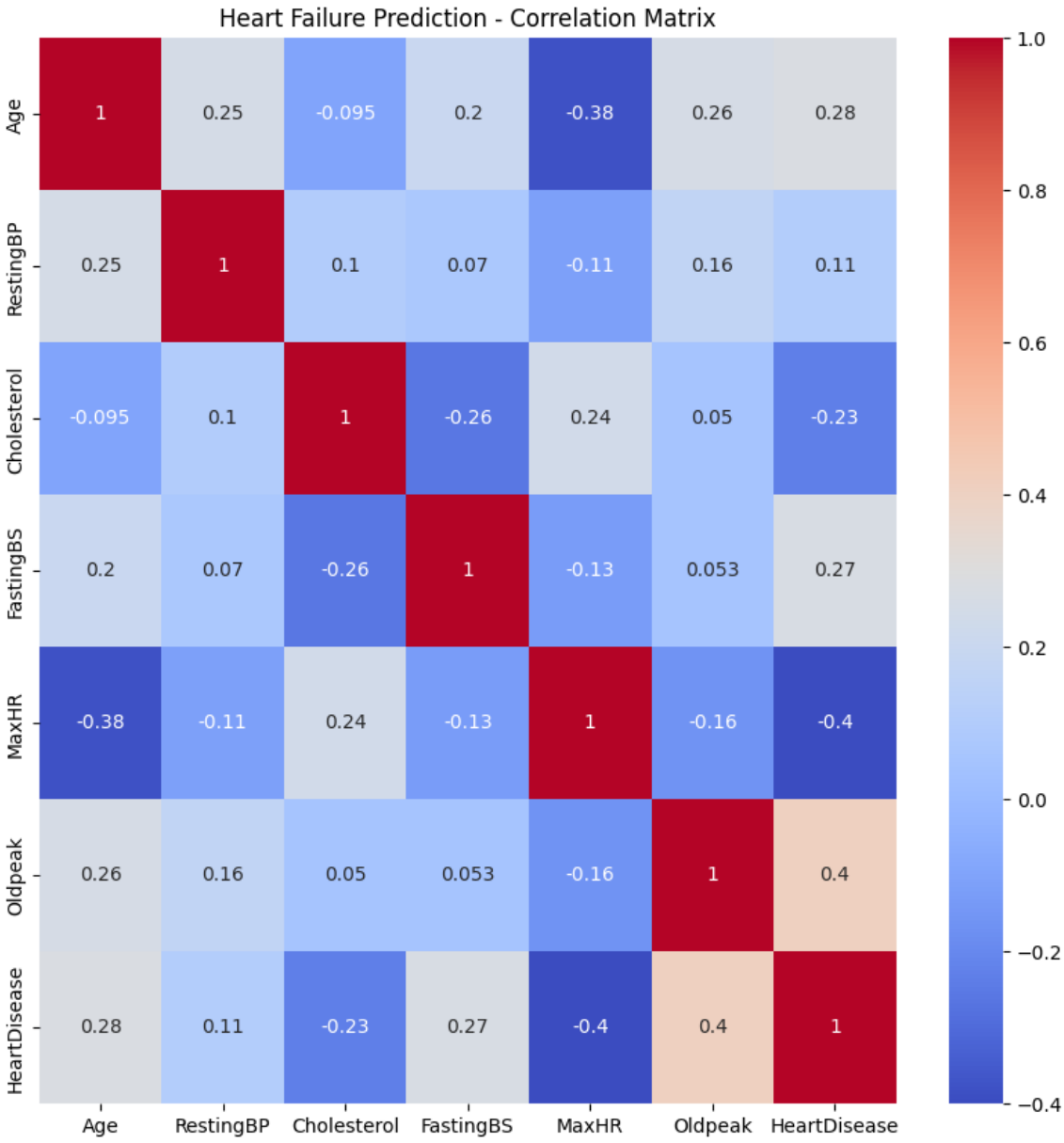


figure1.2: Correlation Matrix

2. Data Preprocessing:

Data preprocessing is a crucial step in the data analysis pipeline, and it involves cleaning, organizing, and transforming raw data into a format suitable for analysis.

2.1. Handling Categorial Data:

The dataset includes categorial features such as “Sex”, “ChestPainType”, “RestingECG”, “ExerciseAngina”, and “ST_Slope”. To facilitate model training, we convert the categorial variables into numerical representations using one-hot encoding (Label encoding). This ensures that the machine learning algorithms can effectively interpret and learn from these features.

2.2. Outlier detection and removal:

After identifying the missing values during the data analysis phase, we implemented a strategy to address them that includes the following:

- Outlier Identification:

  Skewness analysis was conducted to identify potentially skewed numerical features with the dataset

  A threshold of 0.5 was set to classify features as positively skewed, negatively skewed, and not skewed

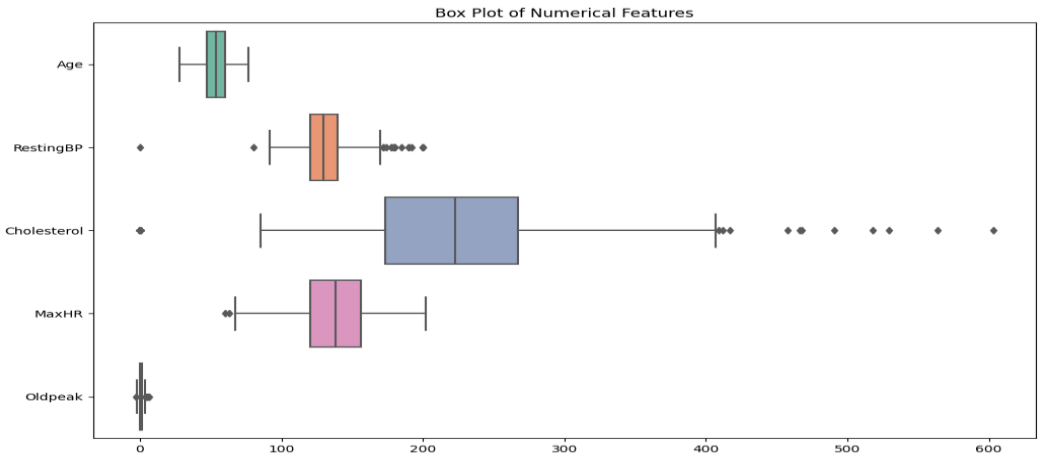


Figure1.3: Box-Plot

- Outlier Replacement Strategy:

  For positively and negatively skewed features exceeding the skewness threshold, a replacement strategy was implemented

  Outliers were replaced with either the median (if positively skewed) or the mean (if negatively skewed or not skewed)

- Implementation:

  The code iterates through each numerical column in the dataset, calculating the skewness and determining the skewness type.

  Outliers were replaced using the interquartile Range (IQR) method, ensuring a robust and effective approach to handling extreme values.

  The replacement was performed in a column-wise manner, and the details of the replacement, including the central tendency used (mean or median) were printed for transparency.

2.3. Data splitting:

Once the data preprocessing steps were completed, the next crucial step in our methodology was to split the dataset into training and testing sets. This division is essential to assess the model's performance on unseen data and avoid overfitting. We adopted a common split ratio, allocating a certain percentage of the dataset for training the model and the remaining portion for testing. The sklearn library's train_test_split function was utilized for a randomized and representative split. This separation ensures that the model is trained on one subset and evaluated on another, providing an unbiased evaluation of its predictive capabilities.

3. Choosing Models:

In pursuit of developing an accurate heart failure prediction model, we explored three diverse machine learning algorithms: Logistic Regression, k-nearest Neighbors (KNN), and Naive Bayes. Each of these models brings unique strengths and characteristics to the predictive task, allowing us to evaluate their performance comprehensively.

3.1. Logistic Regression:

Logistic Regression is a well-established algorithm for binary classification tasks, making it an ideal candidate for our heart failure prediction model. We instantiated the Logistic Regression model using the sci-kit-learn library and trained it on the preprocessed training dataset. The model learned the relationships between the input features and the binary target variable, capturing the probability of a patient experiencing heart failure. The logistic regression model provides interpretable coefficients, enabling us to understand the impact of each feature on the predicted outcome.

3.2. K-Nearest Neighbors:

K-Nearest Neighbors is a non-parametric algorithm that makes predictions based on the majority class of its k-nearest neighbors in the feature space. We implemented KNN using sci-kit-learn, adjusting the hyperparameter n_neighbors to optimize the model's performance. The KNN algorithm is particularly suited for our dataset.

3.3. Naïve Bayes:

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. We opted for the Bernoulli Naive Bayes variant from sci-kit-learn, Bernoulli Naive Bayes is particularly useful in situations where feature vectors represent binary data

or data that can be converted into binary data. It's well-suited for binary features, such as the presence or absence of a certain term, or the occurrence of a certain event.

4. Training the Model and Evaluating:

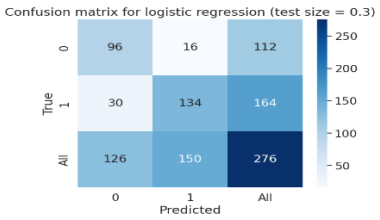
4.1. Logistic Regression:

In this section, we assess the performance of a logistic regression model trained on a given dataset. The evaluation is conducted across different test sizes (30%, 20%, 10%) to explore how the model generalizes to varying proportions of training and testing data. The dataset is divided into training and testing sets using the train_test_split function from scikit-learn. Different test sizes (30%,20%, 10%) are considered to observe the impact on model performance.

The model is evaluated using various metrics, including accuracy, confusion matrix, and classification report. The confusion matrix provides insights into true positives, false positives, true negatives, and false negatives.

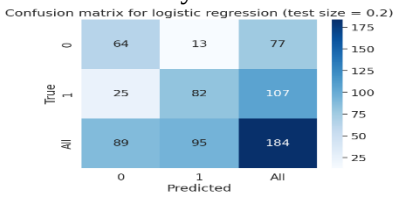
- Test Size 30%

  Accuracy: 83%



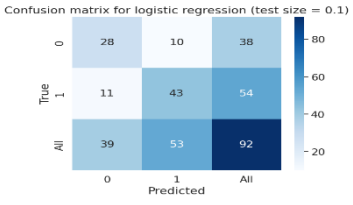
- Test Size 20%

  Accuracy:79%



- Test Size 10%

  Accuracy 77%



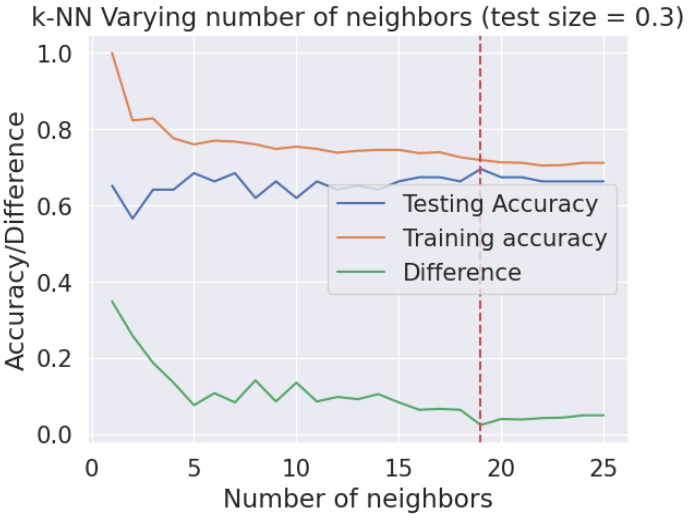
The model exhibited commendable accuracy across all test sizes, showcasing its ability to generalize well to different proportions of training and testing data. It is noteworthy to mention that the exploration of different test sizes revealed understanding how the model responds to varying amounts of test data is crucial for robust model assessment.

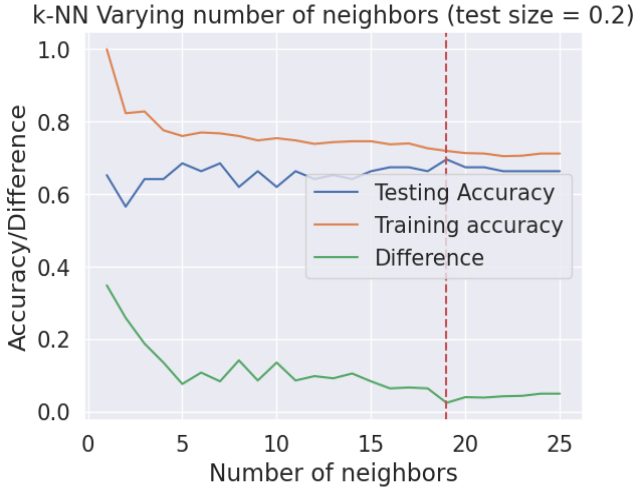
4.2. KNN:

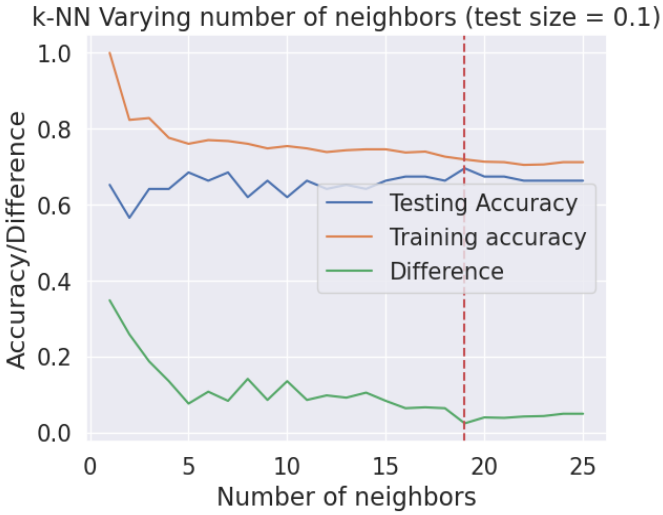
This section focuses on the evaluation of the k-NN (k-Nearest Neighbors) classifier, aiming to determine the optimal number of neighbors for model performance. The assessment is conducted across different test sizes (30%, 20%, and 10%) to observe the influence of varying training and testing data proportions. The goal is to identify the number of neighbors that maximizes testing accuracy while maintaining a small difference with training accuracy, thus avoiding overfitting or underfitting.
The model is tested with a range of neighbors, from 1 to 25, to identify the number of neighbors that yields the highest testing accuracy while maintaining a small difference between testing and training accuracy. The intention is to avoid overfitting (large difference) or underfitting (small difference).

The k-NN classifier demonstrates robust performance across various test sizes when selecting an appropriate number of neighbors. The detailed evaluation, including the exploration of different numbers of neighbors and the examination of confusion matrices, provides a comprehensive understanding of the model's strengths and limitations. These insights can guide further refinement and application of the k-NN classifier in practical scenarios.

The results are as follows:


k-NN Varying number of neighbors (test size = 0.3)
Accuracy/Difference
Number of neighbors
Testing Accuracy
Training accuracy
Difference
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25


k-NN Varying number of neighbors (test size = 0.2)
Accuracy/Difference
Number of neighbors
Testing Accuracy
Training accuracy
Difference
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25


k-NN Varying number of neighbors (test size = 0.1)
Accuracy/Difference
Number of neighbors
Testing Accuracy
Training accuracy
Difference
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25

As can be seen, before tuning the parameters we had the following:

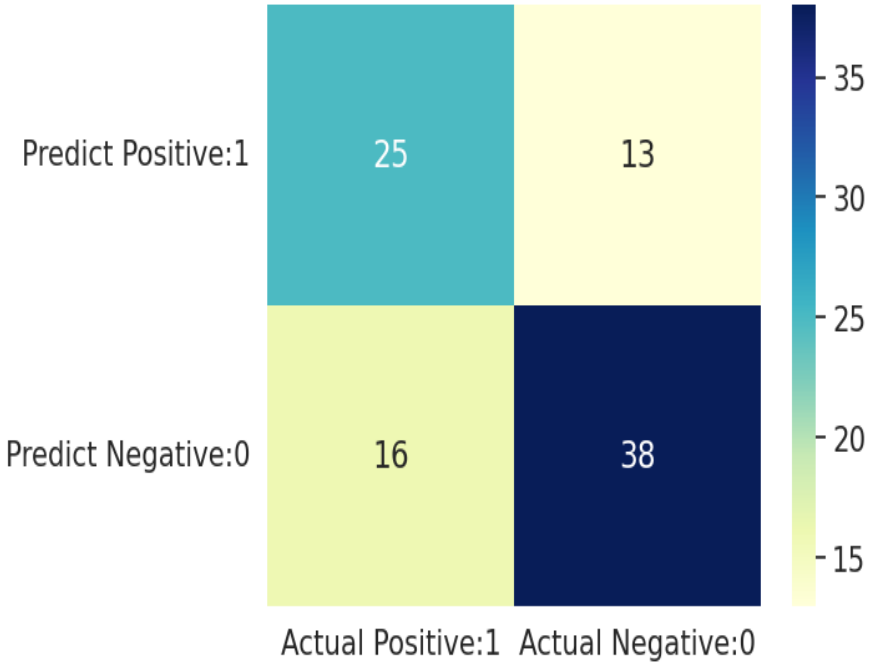
	LR testing accuracy	**k-NN testing accuracy**
30% testing	0.883	0.7
20% testing	0.79	0.7
10% testing	0.77	0.7

Best model: Logistic regression with 30% test size

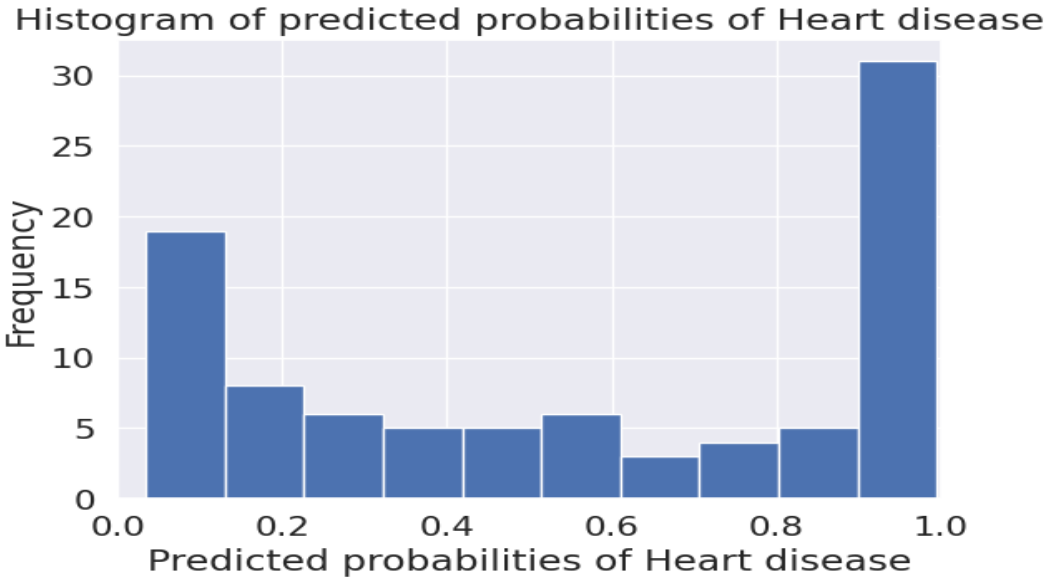
4.3. Naïve Bayes:

The Bernoulli Naive Bayes model is applied to a binary classification problem, where the target variable, Y, is divided into two classes. The training process involves splitting the data into training and testing sets using a test size of 10%. The model is then initialized and trained on the training set using the fit method.

Feature Scaling: To ensure uniformity and comparability of features, the data undergoes feature scaling using the Robust Scaler. This step is crucial, especially when working with Naive Bayes classifiers, to mitigate the impact of feature scale variations.

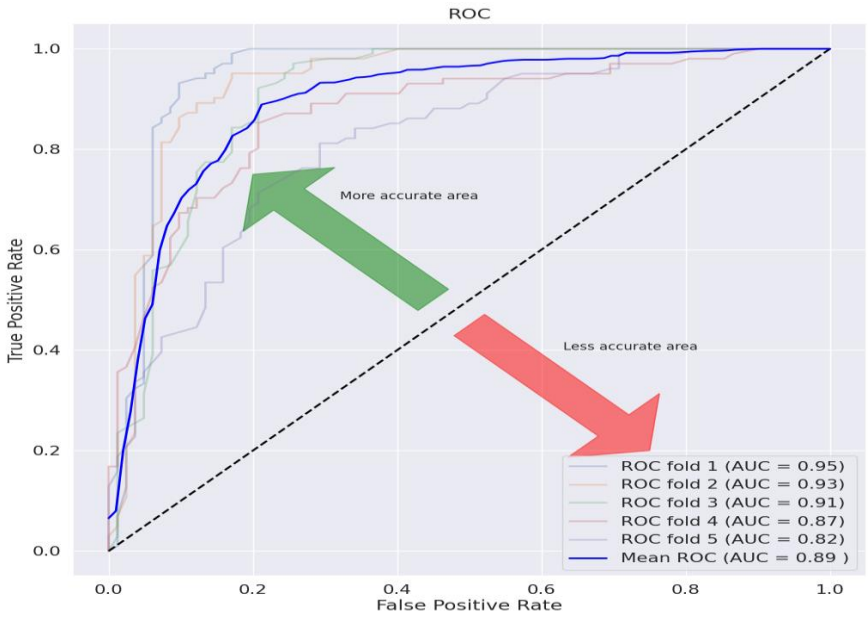


Results shows that the Classification accuracy: 0.6848 & Classification error: 0.3152



ROC:

The Receiver Operating Characteristic (ROC) Curve is a powerful tool for visually assessing the performance of a classification model. It plots the True Positive Rate (Recall) against the False Positive Rate at various classification threshold levels, providing insights into the model's discrimination ability.



5. Tuning the parameters:

As it can be seen, after tuning the parameters we had the following: KNN did not improve, so we will compare only the Logistic regression progress

	Before tuning	After tuning
30% testing	0.883	0.833
20% testing	0.79	0.81
10% testing	0.77	0.78

Results confirm that the best model: Logistic regression with a 30% test size

6. Making Predictions:

   To make predictions on the new enhanced models, we use the LR model with 30% testing data size as it gave the highest testing accuracy among all the logistic regression models and the other models. We provide the models with the best parameters that we got while tuning the parameters. We then predict the first value of the testing data set shown in the notebook.

   Finally, we print the value in the y_test set to find out that is the same as the predicted value. 😊