Faculty of Engineering - CSE Department
Semester: Fall 2023 - Academic Year: 2023-2024
Module Title: Machine Learning
Module Code: CSE 5634

Instructor Name: Dr. Ahmed Alenany
TA Name(s): Eng. Dina Magdy

Project Title: Heart Failure Clinical Records Dataset

Prepared by:

Student Name: Islam Gomaa	**Student ID: 212495**
Student Name: Andrew Hany	**Student ID: 212179**
Student Name: Mahmoud Bahaa	**Student ID: 210733**
Student Name: Ramez George	**Student ID: 195905**

Table of Contents

Introduction:

Cardiovascular diseases are among the most lethal health issues globally, responsible for approximately 17 million deaths annually. Heart failure, a critical condition where the heart cannot pump sufficient blood to sustain other organs, highlights the heart's vital role in our bodies. When the heart struggles or fails in its function, the consequences can be severe, including immediate death or heart failure. Common causes of heart failure include diabetes, hypertension, other heart-related disorders, and conditions such as HIV, thyroid abnormalities, alcohol misuse, and congenital diseases. In 2019, cardiovascular diseases caused an estimated 17.9 million deaths, representing 32% of all global fatalities, with strokes and heart attacks accounting for 85% of these deaths. Consequently, there's an urgent need for enhanced prevention and early detection of heart diseases, especially in children. Early diagnosis allows for more effective treatment and counseling, improving patient outcomes. In recent decades, machine learning and data mining have emerged as prominent tools in various fields, including healthcare. Machine learning, a branch of artificial intelligence, involves algorithms and statistical models that enable computers to perform tasks effectively without explicit programming, relying instead on pattern recognition and inference. Data analytics significantly contributes to deciphering large datasets for diagnostics, disease prevention, prediction, and policymaking in the healthcare sector. While it is advancing steadily, challenges persist in effectively modeling survival rates for heart failure, both in attaining high predictive accuracy and pinpointing key influencing factors. Most models designed for this purpose have so far achieved only limited success. However, more recent models are beginning to show encouraging progress, particularly when integrated with additional objectives. The primary goal of the research paper is to utilize machine learning algorithms for predicting the survival of heart failure patients. This involves tackling the challenge of imbalanced datasets, which have not been adequately addressed previously, and implementing feature selection strategies to enhance prediction accuracy.

Data description:

To conduct a research, of course, the dataset is the main material that will be processed in such a way using an algorithm. The dataset used in this research is Heart Failure Clinical Records data taken from the UCI repository website. The dataset was published in 2020 that contains medical records of 299 heart failure patients that was collected at the Faisalabad Institute of Cardiology and at the Allied Hospital in Faisalabad (Punjab, Pakistan) in 2015. The patients consisted of 105 women and 194 men with the age bracket of 40 to 95 years. The dataset contains 13 features, which present clinical, body, and lifestyle information. Based on the dataset collected, 6 of the attributes are Boolean including the target class (Anemia, High blood pressure, Diabetes, Sex, Smoking and death event) were converted into two categories, in order to make the dataset usable for classification task. Diabetes, smoking, anemia and high blood pressure 0 and 1 attributes were converted to false and true respectively, while death event 0 and 1 were converted to alive and died respectively also sex 0 and 1 were converted to female and male respectively.

S/N	Feature	Explanation	**Measurement**	**Range**
1	Age	Age of the patient	Years	[40,…95]
2	Anemia	Decrease of red blood cells or haemoglobin	Boolean	0,1
3	High blood pressure	If a patient has hypertension	Boolean	0,1
4	Creatinine phosphokinase (CPK)	Level of the CPK enzyme in the blood	mcg/L	[23,…,7861]
5	Diabetes	If the patient is diabetic	Boolean	0,1
6	Ejection fraction	The percentage of blood that leaves that heart after each contraction	Percentage	[14,..,80]
7	Sex	Male or Female	Binary	0,1
8	Platelets	Platelets in the blood	kiloplatelets/mL	[25.01,…,850.00]
9	Serum creatinine	Level of creatinine in the blood	mg/dL	[0.50,...,9.40]
10	Serum sodium	Level of sodium in the blood	mEq/L	[114,...,148]
11	Smoking	If the patient is a smoker	Boolean	0,1
12	Time	Follow-up period	Days	[4,...,285]
13	Death Event (target))	If the patient died during the follow-up period	Boolean	0,1

fig1. Dataset Features

Data understanding:

In this phase**,** python programming language have been used in the Anaconda environment. According to used libraries, Numpy: for standard library for math operations. Pandas: to manipulate data inside data frames and for basic computations. Sklearn: used to apply different ML models. Pyplot: to plot visualizations and Seaborn: for built on top of Pyplot.

First of all, we will import the needed libraries and read the dataset CSV file.

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import warnings
import seaborn as sns
from colorama import Fore, Back, Style
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from mlxtend.plotting import plot_confusion_matrix
from plotly.offline import plot, iplot, init_notebook_mode
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import plotly.express as px
from statsmodels.formula.api import ols
import plotly.graph_objs as gobj
from sklearn.preprocessing import StandardScaler
import time
%matplotlib inline
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

Fig.2 import libraires

```
data=pd.read_csv('heart_failure_clinical_records_dataset.csv')
```

Fig.3 read dataset file

The heart failure clinical records dataset includes both numeric and categorical variables. Let us get overall statistics about dataset by using the following code.

```
df.describe().transpose()
```

fig.4 finds overall statistics

	count	mean	std	min	25%	50%	75%	max
age	299.0	60.833893	11.894809	40.0	51.0	60.0	70.0	95.0
anaemia	299.0	0.431438	0.496107	0.0	0.0	0.0	1.0	1.0
creatinine_phosphokinase	299.0	581.839465	970.287881	23.0	116.5	250.0	582.0	7861.0
diabetes	299.0	0.418060	0.494067	0.0	0.0	0.0	1.0	1.0
ejection_fraction	284.0	38.137324	11.851204	14.0	30.0	38.0	45.0	80.0
high_blood_pressure	299.0	0.351171	0.478136	0.0	0.0	0.0	1.0	1.0
platelets	299.0	263358.029264	97804.236869	25100.0	212500.0	262000.0	303500.0	850000.0
serum_creatinine	299.0	1.393880	1.034510	0.5	0.9	1.1	1.4	9.4
serum_sodium	299.0	136.625418	4.412477	113.0	134.0	137.0	140.0	148.0
sex	299.0	0.648829	0.478136	0.0	0.0	1.0	1.0	1.0
smoking	299.0	0.321070	0.467670	0.0	0.0	0.0	1.0	1.0
time	299.0	130.260870	77.614208	4.0	73.0	115.0	203.0	285.0
DEATH_EVENT	299.0	0.321070	0.467670	0.0	0.0	0.0	1.0	1.0

fig.5 Overall Statistics

Data Visualization:

Visualizing data is indeed a powerful method to gain insights from datasets. By transforming numbers and statistics into graphical formats, complex relationships and patterns can become more understandable and accessible.

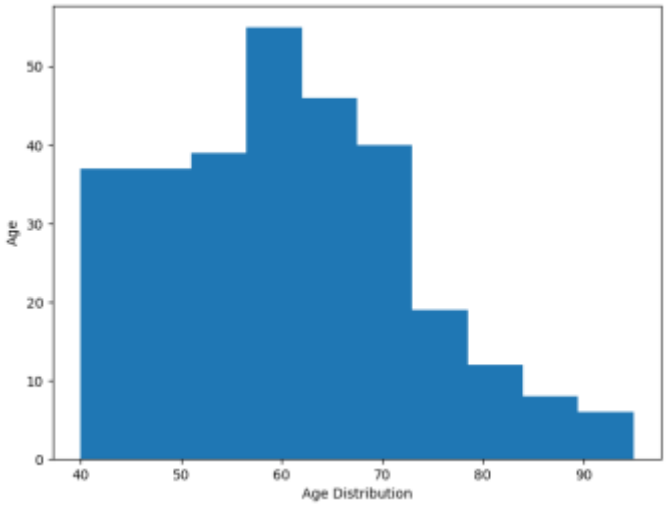

Fig.6

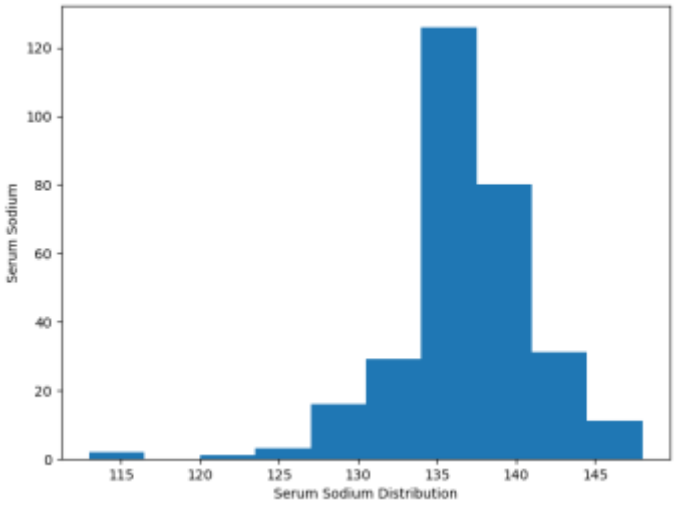

Fig.7

The histogram is a graph that shows the distribution of a continuous variable by breaking it into bins and showing the frequency of observations within each bin.

The resulting histogram in fig.6 shows that the most common age range among the patients in the dataset is between 50 to 70 years old, with a smaller number of patients in the younger and older age ranges**.** Also, the smallest age is 40 years old. This information can help healthcare professionals to understand the age distribution of patients with heart failure and to tailor their treatment plans accordingly.

The histogram in fig.7 shows the distribution of serum sodium levels in the dataset. Serum sodium is an important electrolyte that helps regulate fluid balance in the body. The plot shows that serum sodium levels are normally distributed around a mean of approximately 137 mEq/L. Abnormal levels of serum sodium can be an indicator of underlying health issues.

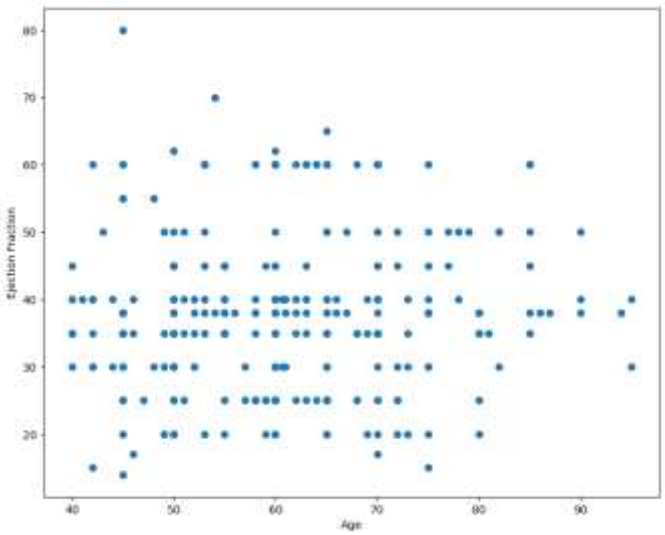

Fig.8

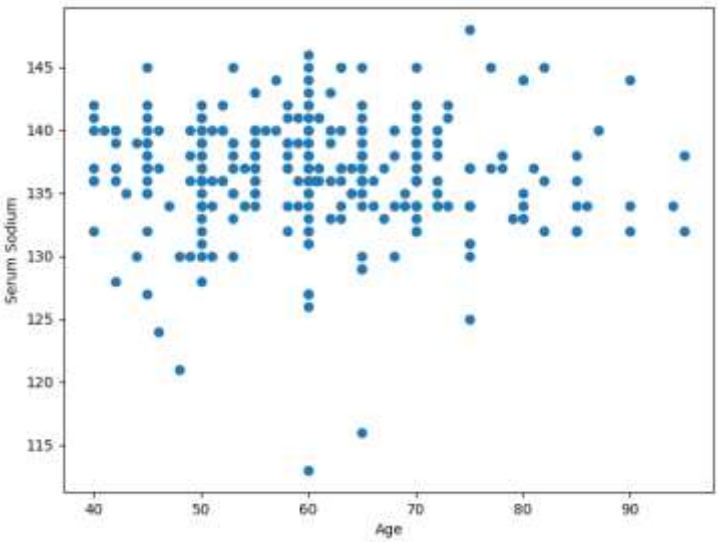

Fig.9

The scatter plot in fig.8 helps us understand the relationship between age and ejection fraction, which is the percentage of blood leaving the heart each time it contracts. The plot shows that as age increases, ejection fraction tends to decrease. This could indicate that older individuals may be at a higher risk of heart failure due to a decrease in heart function.

The scatter plot in fig.9 shows the relationship between age and serum sodium levels. The plot shows that there is no clear relationship between the two variables, with serum sodium levels distributed evenly across all ages. However, there are some outliers with extremely low serum sodium levels, which could indicate underlying health issues.

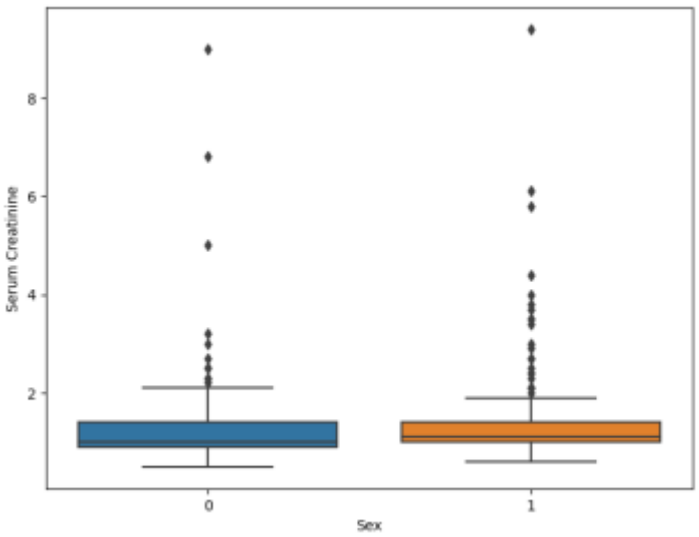

fig.10

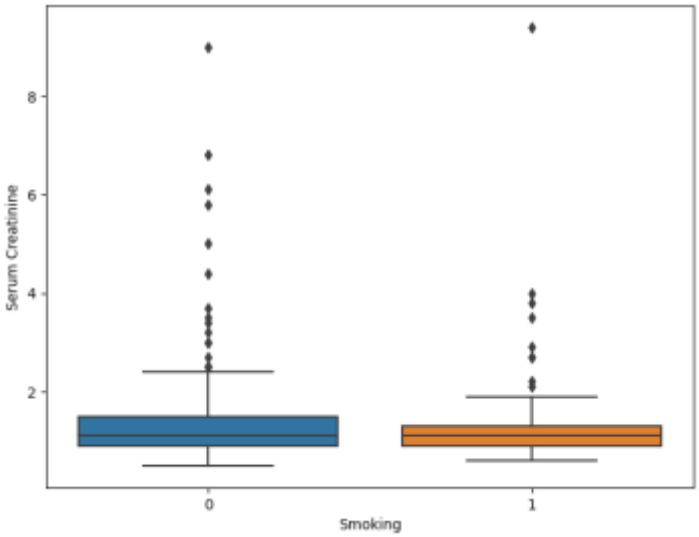

fig.11

The box plot in fig.10 showed that there was a slightly higher median serum creatinine level in males compared to females.

The box plot in fig.11 shows the distribution of serum creatinine levels for smokers and non-smokers. Serum creatinine is a measure of kidney function, and high levels can be a sign of kidney damage. The plot shows that on average, smokers have higher serum creatinine levels than non-smokers. This suggests that smoking may have a negative impact on kidney function.

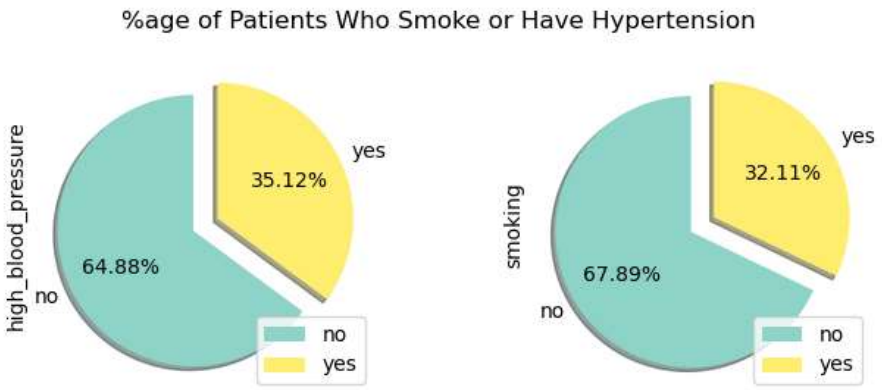

Fig.12

In fig.12 the first graph on the left shows the percentage of patients with high blood pressure. According to the data, 64.88% of patients do not have high blood pressure, while 35.12% do. The second graph on the right details the percentage of patients who smoke. It indicates that 67.89% of patients do not smoke, and 32.11% of the patients are smokers.

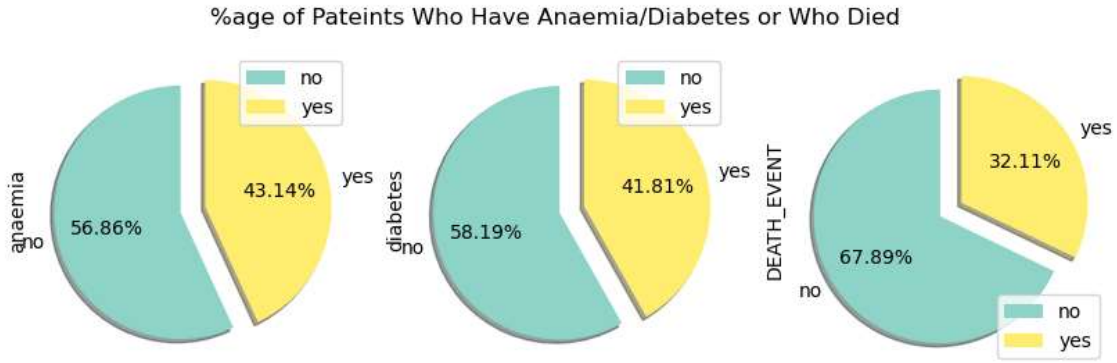

Fig.13

In fig.13 the first graph on the left shows the percentage of patients with anemia, with 56.86% not having anemia and 43.14% having it.

The middle graph provides information on the prevalence of diabetes among patients. It indicates that 58.19% of the patients do not have diabetes, while 41.81% do.

The third graph on the right represents the percentage of patients who had a death event. According to this graph, 67.89% of the patients did not experience a death event, while 32.11% did

Data Preprocessing:

Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. So, before running any data mining algorithms preprocessing is necessary.

In this crucial step, it is imperative to meticulously identify and detect all null values in the dataset. Ensuring that every missing or undefined data point is detected forms the foundation of a robust data preprocessing phase. Once this is achieved, implementing min-max normalization is the next significant stride. This technique adjusts the data values to a common scale, enhancing the overall accuracy of the model. By carefully executing these processes, we lay the groundwork for a more reliable and effective machine learning model.

We can see that there are 15 null values in “ejection_fraction” column

```
df.isnull().sum()
```

Fig.14 detect null values

```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction          15
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

Fig.15 Null values

Here we will fill the null values with the mean value of that column “ejection_fraction” by using these lines of code.

```
df.replace(" ", np.nan, inplace = True)
avg_ejection_fraction=df['ejection_fraction'].astype('float').mean(axis=0)
print("Average of ejection_fraction:", avg_ejection_fraction)
df["ejection_fraction"].replace(np.nan, avg_ejection_fraction, inplace=True)
```

Fig.16 filling null vales

```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

Fig.17 Null Values After Filling

In the following, we applied some codes to build Heatmap to present the correlation between all the dataset attributes

```
corrMatt = df.corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sns.heatmap(corrMatt, mask=mask, vmax=.8, square=True, annot=True, cmap='inferno')
```

Fig.18 implementation of Heatmap

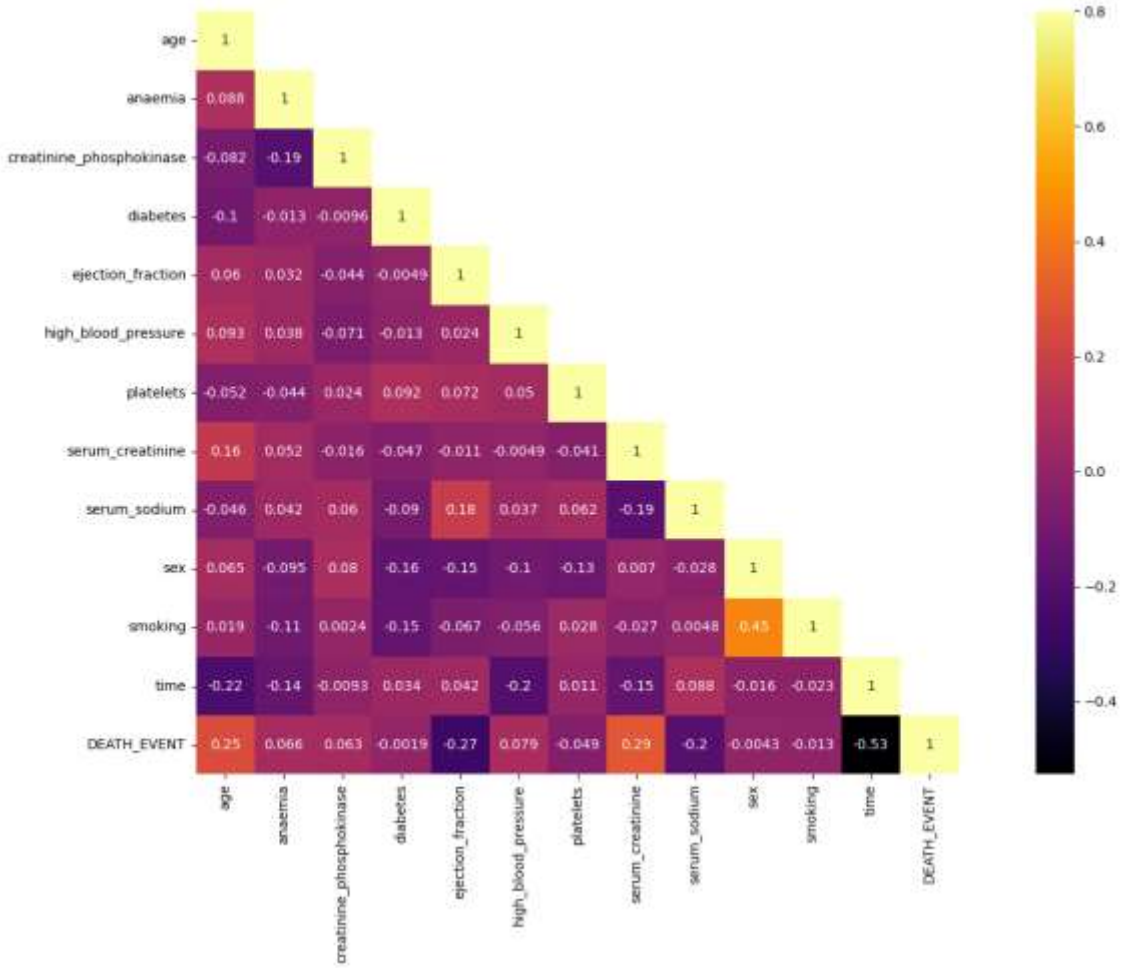

Fig.19 Heatmap

Correlation heatmaps are a type of plot that visualize the strength of relationships between numerical variables. Correlation plots are used to understand which variables are related to each other and the strength of this relationship.

The heatmap in fig.19 shows that the largest correlation is between time (Follow-up period) and DEATH EVENT, with a value of -0.53(negative correlation). The serum creatinine, ejection fraction, and age follow, with correlation values of 0.29, 0.26, and 0.25 with the death event, respectively.

Now, we have to make very important step that is separate the dataset into train and test which is a crucial step in many data science and machine learning projects.

Firstly, we will split the inputs and the output. where the output is:" Death_event" column

```
data_1 = df.drop('DEATH_EVENT', axis=1)
data_1 = np.array(data_1, dtype=int)
target_1 = df['DEATH_EVENT']
target_1 = np.array(target_1, dtype=int)
```

Fig.20 Split the inputs and output

Secondly, Split the dataset to train and test data where: 70% for Training & 30% for Testing.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
x_train, x_test, y_train, y_test = train_test_split(data_1, target_1, test_size=0.3, random_state=4)
```

Fig.21 Split the to training and testing

And here as a final step in data preprocessing, we will normalize the dataset. The normalization has been done to make all the attribute values between zero and one (0–1) to reach better accuracy.

```
x_train = normalize(x_train)
x_test = normalize(x_test)
```

Fig.22 apply normalization

Modeling

Given that our target variable is categorical in nature, employing classification models stands out as the most effective approach for prediction. In this section, we will delve into the application of a variety of classification algorithms. These include Naive Bayes, k-nearest neighbors (k-NN), Logistic Regression, Decision Trees, and Random Forest. Each of these methods will be meticulously applied and evaluated to ascertain their efficacy in achieving our predictive objectives.

K-Nearest Neighbors (KNN): is a simple and versatile supervised machine learning algorithm used for classification and regression tasks. The main idea behind KNN is to classify a data point based on the majority class or average of its k nearest neighbors in the feature space.

Initially, we constructed a k-nearest neighbors (k-NN) model, selecting **K=5**. Upon testing, this configuration yielded a promising accuracy rate of **0.80%**.

```
k = 5
neigh = KNeighborsClassifier(n_neighbors = k).fit(x_train,y_train)
neigh
yhat = neigh.predict(x_test)
yhat[0:5]
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(x_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))
```

```
Train set Accuracy:  0.7894736842105263
Test set Accuracy:  0.8
```

Fig.23 KNN model construction

Following the initial test, we determined that the achieved accuracy was suboptimal. Consequently, we implemented a specialized code designed to identify the optimal value of K from the graph that would yield the highest accuracy for our k-nearest neighbors (k-NN) model.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

Ks = 10
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))

for n in range(1, Ks):
    # Train Model and Predict
    neigh = KNeighborsClassifier(n_neighborsn).fit(x_train, y_train)
    yhat = neigh.predict(x_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)
    std_acc[n-1] = np.std(yhat == y_test) / np.sqrt(yhat.shape[0])

# Plot the model accuracy for different number of neighbors
plt.plot(range(1, Ks), mean_acc, 'g')
plt.fill_between(range(1, Ks), mean_acc - 1 * std_acc, mean_acc + 1 * std_acc, alpha=0.10)
plt.fill_between(range(1, Ks), mean_acc - 3 * std_acc, mean_acc + 3 * std_acc, alpha=0.10, color="green")
plt.legend(('Accuracy ', '+/- 1xstd', '+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Neighbors (K)')
plt.tight_layout()
plt.show()
```

Fig.24 KNN graph code

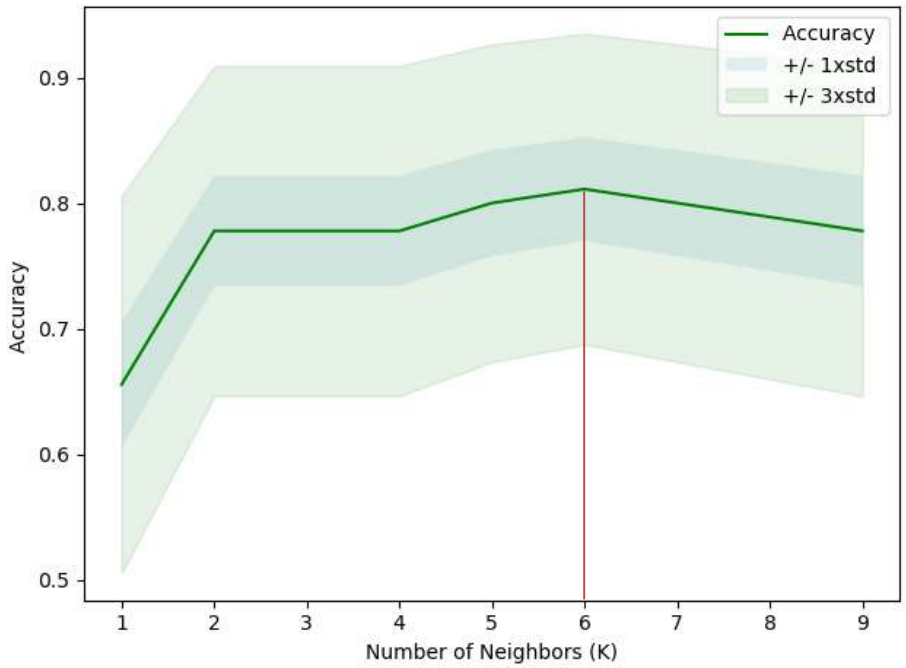

Fig.25 KNN Graph

```
k = 6
neigh6 = KNeighborsClassifier(n_neighbors = k).fit(x_train,y_train)
yhat6 = neigh6.predict(x_test)
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh6.predict(x_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat6))
```

```
Train set Accuracy:  0.7799043062200957
Test set Accuracy:  0.8111111111111111
```

Fig.26 Accuracy after choosing optimal k

The graph presented above offers a clear visual representation, demonstrating that a K value of 6 achieves the highest level of accuracy. Specifically, it indicates an accuracy rate of **81.11%** for our k-nearest neighbors (k-NN) model.

Now, we will proceed to represent the results using a **confusion matrix**. This matrix will provide a comprehensive view of the model's performance, illustrating the accuracy of predictions in relation to actual outcomes.

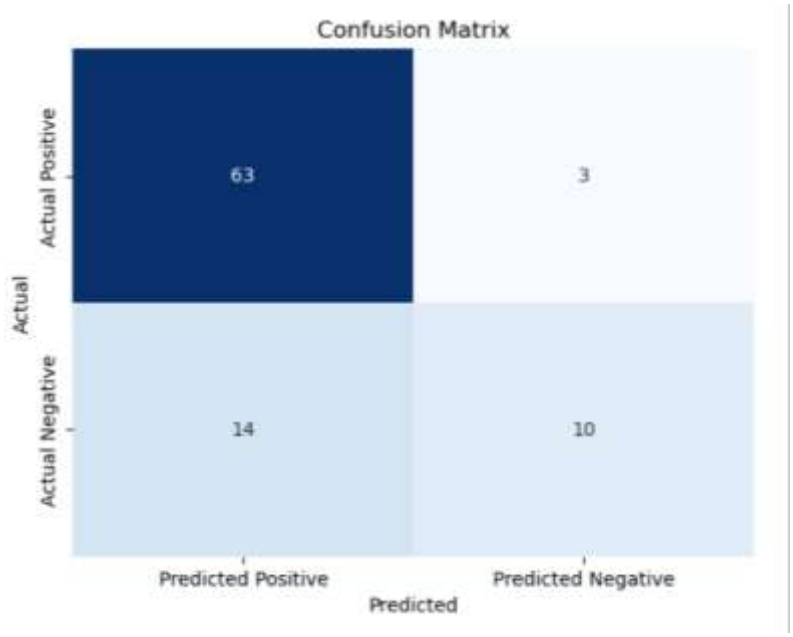

Fig.27 KNN confusion matrix

From confusion matrix graph we got classification report and results as follows:

	precision	recall	f1-score	support
0	0.82	0.95	0.88	66
1	0.77	0.42	0.54	24
accuracy			0.81	90
macro avg	0.79	0.69	0.71	90
weighted avg	0.81	0.81	0.79	90

```
True Positives (TP): 63
False Positives (FP): 14
True Negatives (TN): 10
False Negatives (FN): 3
```

Fig.28 Classification report for KNN model

Naïve Bayes (NB) Model: The Naive Bayes (NB) model is a simple but effective machine learning algorithm used for classification tasks. It's based on Bayes' theorem, which calculates the probability of a hypothesis given the evidence. In the context of classification, Naive Bayes assumes that the features are conditionally independent, given the class label.

The NB model calculates the probability of each class for a given set of features and assigns the class with the highest probability to the input data point. Despite its "naive" assumption of independence, the NB model often performs well, especially in situations where this assumption holds reasonably well.

Firstly, we constructed a Naive Bayes model by using the needed library.

```
# Initialize the Naive Bayes model
naive_bayes_model = GaussianNB()

# Train the model
naive_bayes_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = naive_bayes_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

Fig.29 Naive Bayes construction

Now, it is the time to get all results of Naive Bayes model using confusion matrix.

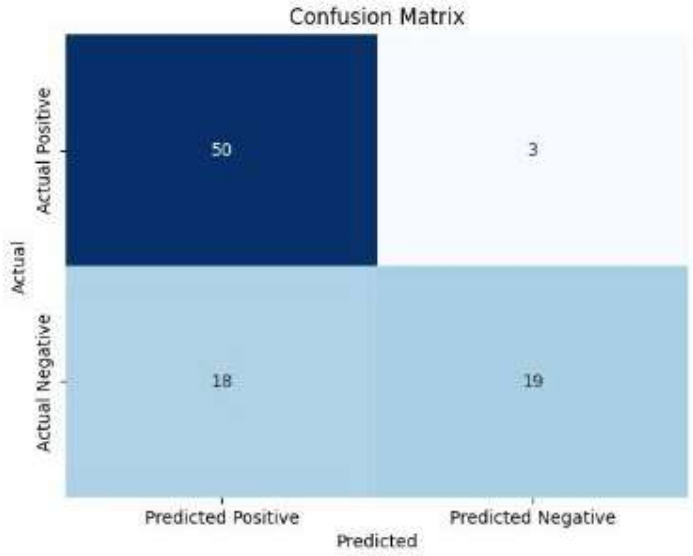

Fig.30 Naive Bayes confusion matrix

```
Accuracy: 0.77
Confusion Matrix:
[[50  3]
 [18 19]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.94      0.83        53
           1       0.86      0.51      0.64        37

    accuracy                           0.77        90
   macro avg       0.80      0.73      0.74        90
weighted avg       0.79      0.77      0.75        90
```

```
True Positives (TP) : 50
False Positives (FP) : 18
True Negatives (TN) : 19
False Negatives (FN) : 3
```

Fig.31 Classification report for Naïve Bayes model

Decision Tree Model (DT): A Decision Tree is a machine learning algorithm that recursively splits a dataset based on features, forming a tree-like structure with decision and leaf nodes. It is used for classification and regression tasks, with nodes representing decisions and leaves indicating final outcomes.

The algorithm selects features to maximize homogeneity in resulting nodes, and stopping criteria prevent overfitting. Despite simplicity and interpretability, Decision Trees may need pruning to avoid overfitting.

Moreover, we constructed a Decision Tree model by using the following code.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

model = DecisionTreeClassifier(random_state=4)
model.fit(x_train, y_train)

# Making predictions
y_pred = model.predict(x_test)

# Evaluating the model
conf_matrix = confusion_matrix(y_test, y_pred)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", conf_matrix)
```

Fig.32 Decision Tree construction

Now, we will proceed to represent the results of Decision Tree using a confusion matrix and classification report.

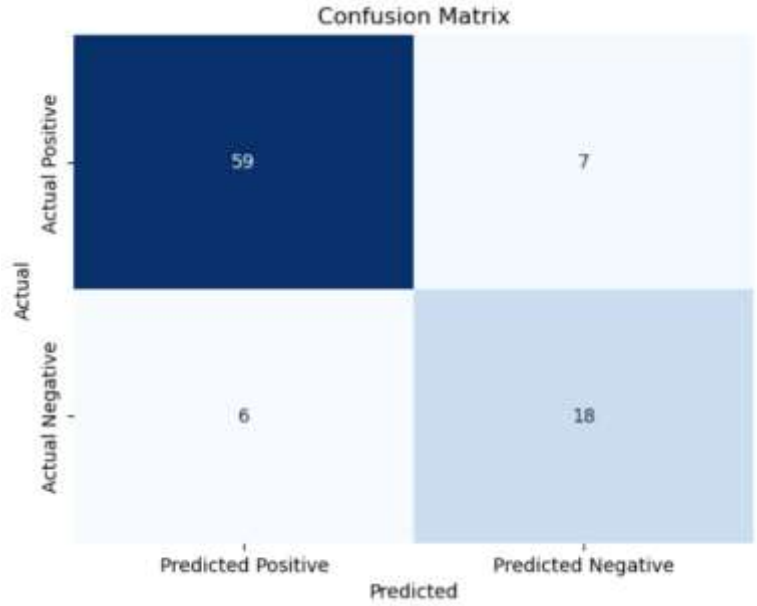

Fig.33 Decision Tree confusion matrix

Classification Report:

	precision	recall	f1-score	support
0	0.91	0.89	0.90	66
1	0.72	0.75	0.73	24
accuracy			0.86	90
macro avg	0.81	0.82	0.82	90
weighted avg	0.86	0.86	0.86	90

True Positives (TP): 59
False Positives (FP): 6
True Negatives (TN): 18
False Negatives (FN): 7

Fig.34 Classification report for Decision Tree model

Random Forest (RF): Random Forest is a commonly-used machine learning algorithm which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

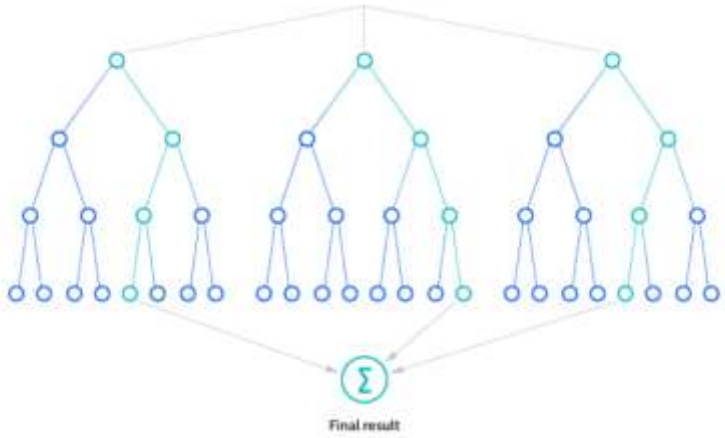

Fig.35 Random Forest Algorithm

Moreover, we constructed a Random Forest Algorithm by using the following code.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# Create a Gaussian Classifier
clf = RandomForestClassifier(n_estimators=100)

# Train the model using the training sets
clf.fit(x_train, y_train)

# Predict the response for test dataset
y_pred = clf.predict(x_test)

# Model Accuracy
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(conf_matrix)

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Fig.36 Random Forest model construction

The next step, we will proceed to represent the results of Random Forest model using a confusion matrix and classification report.

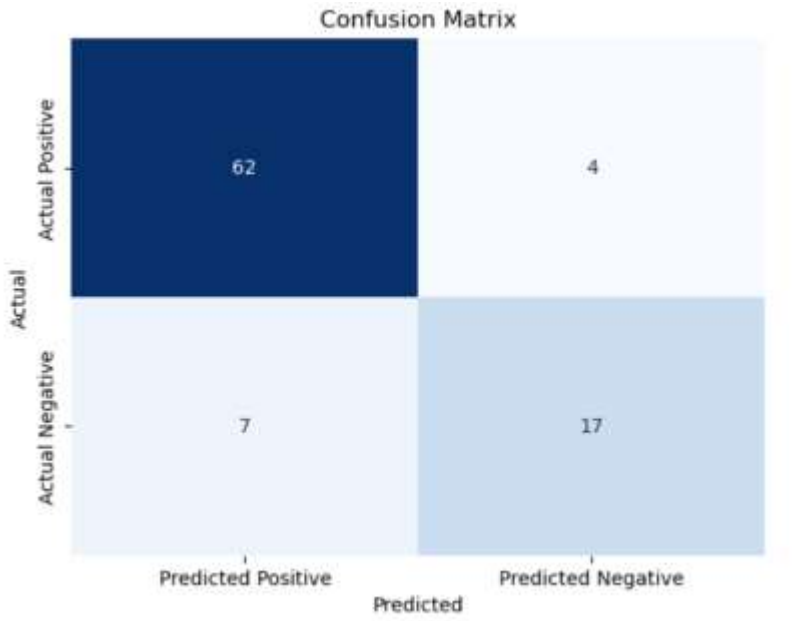

Fig.37 Random Forest model confusion matrix

```
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.94      0.92        66
           1       0.81      0.71      0.76        24

    accuracy                           0.88        90
   macro avg       0.85      0.82      0.84        90
weighted avg       0.87      0.88      0.88        90


True Positives (TP): 62
False Positives (FP): 7
True Negatives (TN): 17
False Negatives (FN): 4
```

Fig.38 Classification report for Random Forest model

Results

Initially, we will present our findings, followed by a comparative analysis between our work and three other research studies. This will include a detailed comparison of our results with sample methodologies documented in the literature, specifically focusing on the same dataset.

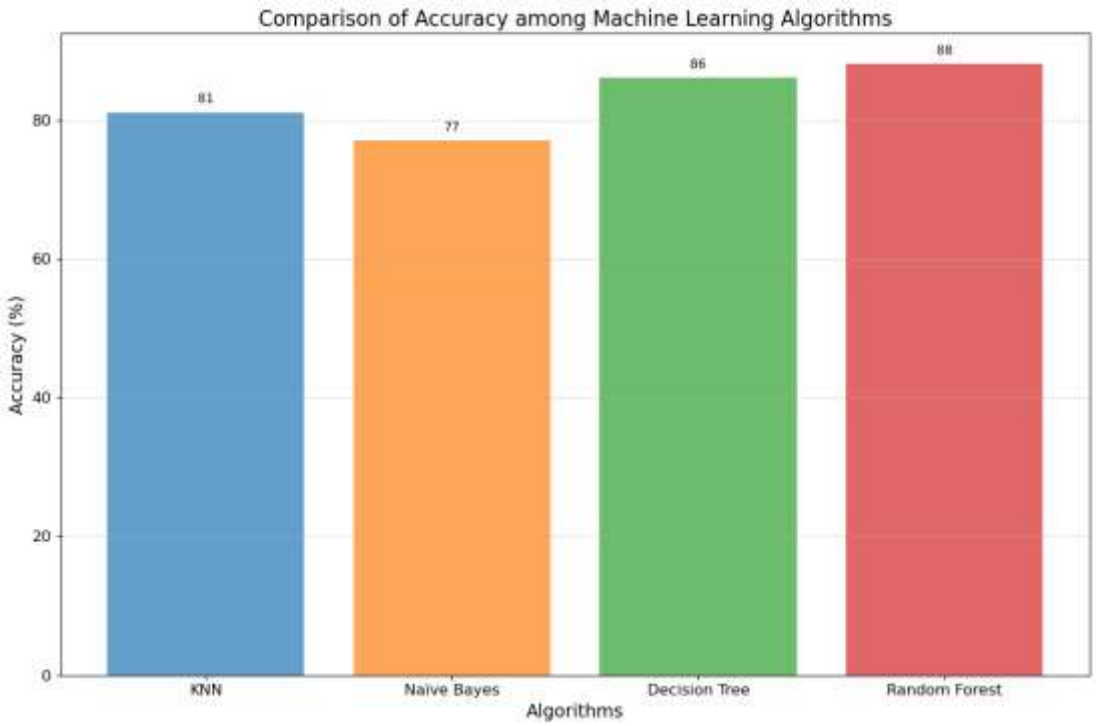

Fig.38 Comparison between used algorithms

The bar graph shows our study's accuracy results for various machine learning algorithms: k-Nearest Neighbors with 81%, Naive Bayes with 77%, Decision Tree with 86%, and Random Forest with 88%. Random Forest leads in performance, while Naive Bayes has the most room for improvement.

In the following graph, we present a comparative analysis of our findings alongside the results from the research paper titled '**Prediction of Survival of Heart Failure Patients Using Random Forest'.**

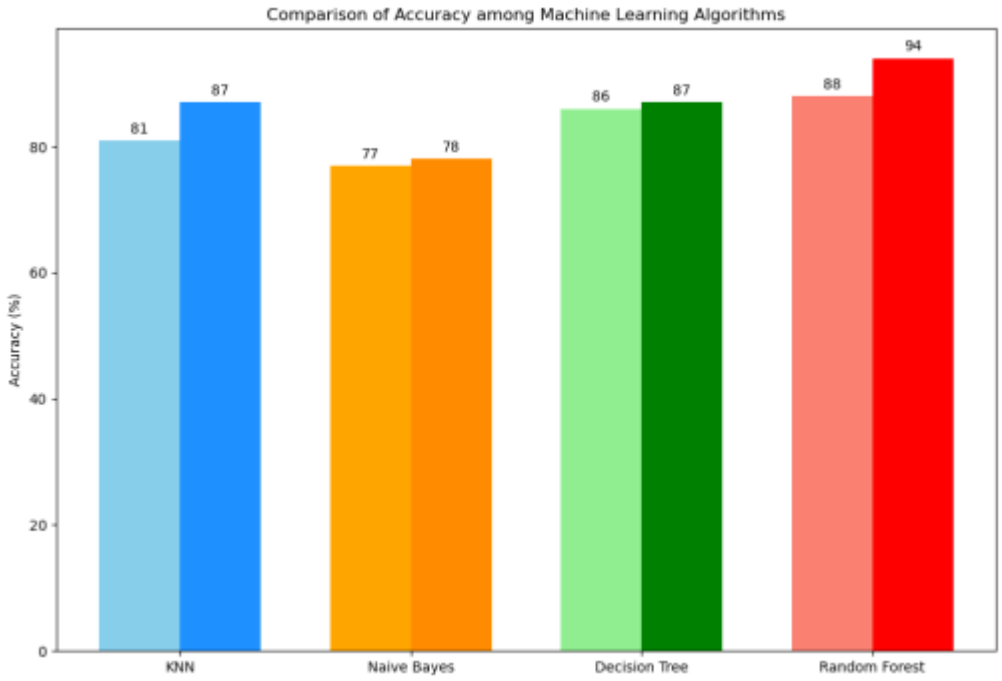

Fig.39 Comparison between our results and other research

From previous graph: k-Nearest Neighbors algorithm, our model shows an 81% accuracy compared to the 87% reported in the research paper. In the case of Naive Bayes, our results achieving 77% accuracy against the paper 78%. The Decision Tree algorithm presents a result with 86% and the paper 87%. Notably, the Random Forest algorithm reach in our model reaches an 88% while 94% in research paper.

In the following graph, we show a comparison of our results with the research paper **'A Soft Voting Ensemble Classifier to Improve Survival Rate Predictions of Cardiovascular Heart Failure Patients'.**

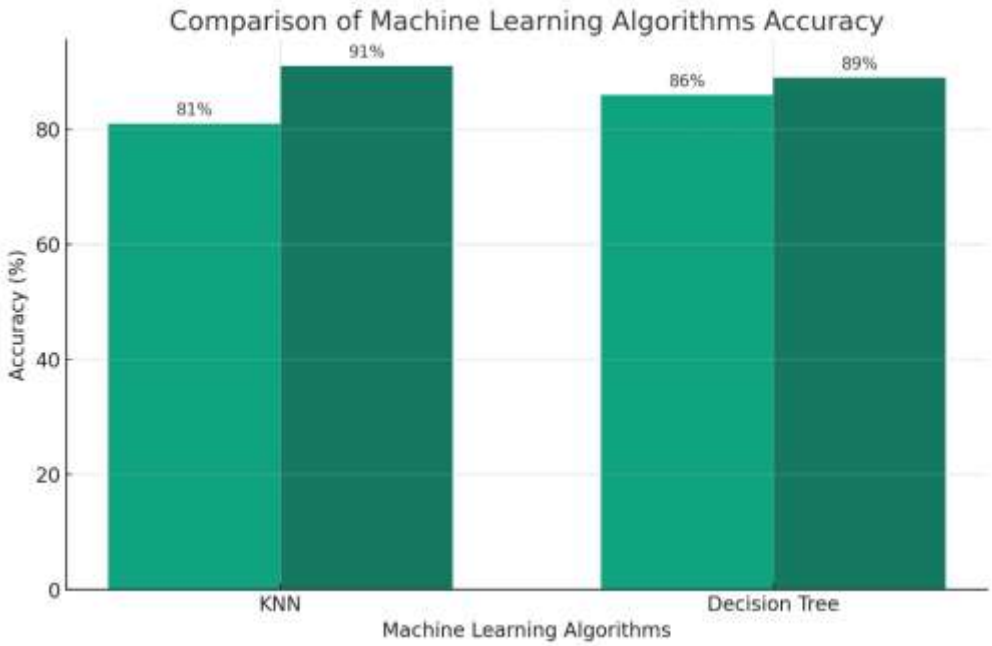

Fig.40 Comparison between our results and other research

The previous graph compares the accuracy of two machine learning algorithms: k-Nearest Neighbors and Decision Tree. For KNN, our results show an 81% accuracy, while the research paper reports have accuracy of 91%. In the case of the Decision Tree algorithm, our study achieved an 86% accuracy while 89% in the research paper.

This graph represents a compression between our results and the paper titled '**Predicting Survival of Heart Failure Patients Using Classification Algorithms**'.

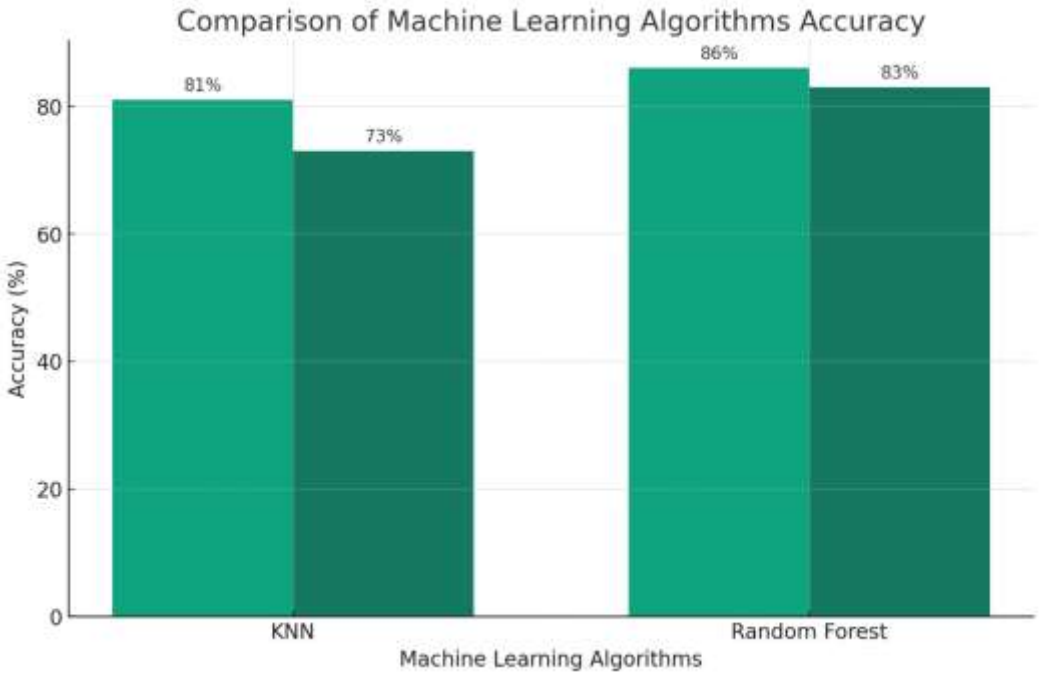

Fig.41 Comparison between our results and other research

The previous graph depicts a comparison of accuracy between our machine learning k-Nearest Neighbors and Random Forest. Our KNN model shows an accuracy of 81%, compared to 73% reported in the research paper. On the other hand, for the Random Forest algorithm, our model has an accuracy of 86%, compared with 83% accuracy cited in the paper.

Conclusion

Human existence is deeply intertwined with the intricate and harmonious functioning of various bodily organs, with the heart playing a pivotal role. The malfunctioning of this vital organ, leading to heart failure, represents a critical and potentially fatal stage in cardiac health issues. In the realm of medical science and data analysis, data mining methods have shown significant potential in accurately predicting patient outcomes, especially in cases of heart failure. Our comprehensive research delved into this area, utilizing the advanced Random Forest algorithm. This approach was further refined by incorporating resampling preprocessing techniques, which were applied meticulously to a Clinical Records dataset. This dataset is notably comprehensive, encompassing 12 distinct attributes and one class, providing a robust foundation for analysis. In our investigative journey, we explored and compared a spectrum of machine learning algorithms. This lineup included the k-Nearest Neighbors, Naive Bayes, Decision Tree, and, notably, the Random Forest algorithm. Through rigorous testing and analysis, the Random Forest algorithm stood out as the most accurate and reliable method. Our current research endeavors are primarily focused on enhancing the predictive accuracy of these algorithms. The aim is to improve the life expectancy predictions for patients suffering from cardiovascular heart failure, utilizing these sophisticated machine learning techniques. This endeavor is not just a technical challenge but also a significant step towards better patient care and health management in the field of cardiology.

References

- Munandar, A., Maulana Baihaqi, W. and Nurhopipah, A. (2023) 'A soft voting ensemble classifier to improve survival rate predictions of cardiovascular heart failure patients', *ILKOM Jurnal Ilmiah*, 15(2), pp. 344–352. doi:10.33096/ilkom.v15i2.1632.344-352.
- Oladimeji, O.O. and Oladimeji, O. (2020) 'Predicting survival of heart failure patients using classification algorithms', *JITCE (Journal of Information Technology and Computer Engineering)*, 4(02), pp. 90–94. doi:10.25077/jitce.4.02.90-94.2020.
- Newaz, A., Haq, F.S. and Ahmed, N. (2021) 'A case study on risk prediction in heart failure patients using random survival forest', *2021 5th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)* [Preprint]. doi:10.1109/iceeict53905.2021.9667933.
- *What is Random Forest?* (no date) *IBM*. Available at: https://www.ibm.com/topics/random-forest
- Keita, Z. (2022) *Classification in machine learning: A guide for beginners*, *DataCamp*. Available at: https://www.datacamp.com/blog/classification-machine-learning