

# PROJECT REPORT

## HEART FAILURE PREDICTION



Submitted By: -

Name: Shabista Mehrish

USN: 3GN18CS091

Department: Computer Science

Submission Date: 13/10/2021

# UNDERTAKING

I declare that the work presented in this project titled “Heart Failure Prediction”, submitted to the All India Council of Robotics and Automation, for the award of the Internship in Data Science, is my original work. I have not plagiarized or submitted the same work for the award of any other Internship. In case this undertaking is found incorrect, I accept that my Project may be unconditionally withdrawn.

13 October,2021

Shabista Mehrish  
3GN18CS091

# CERTIFICATE

This is to certify that the work contained in the project titled “Heart Failure Prediction”, by Shabista Mehrish, has been carried out under my supervision and that this work has not been submitted elsewhere for internship.

All India Council of Robotics and Automation  
Data Science  
Delhi-110020

# PREFACE

Heart failure — sometimes known as congestive heart failure — occurs when the heart muscle doesn't pump blood as well as it should. When this happens, blood often backs up and fluid can build up in the lungs, causing shortness of breath.

Certain heart conditions, such as narrowed arteries in the heart (coronary artery disease) or high blood pressure, gradually leave the heart too weak or stiff to fill and pump blood properly.

Proper treatment can improve the signs and symptoms of heart failure and may help some people live longer. Lifestyle changes — such as losing weight, exercising, reducing salt (sodium) in your diet and managing stress — can improve your quality of life. However, heart failure can be life-threatening. People with heart failure may have severe symptoms, and some may need a heart transplant or a ventricular assist device (VAD).

One way to prevent heart failure is to prevent and control conditions that can cause it, such as coronary artery disease, high blood pressure, diabetes and obesity.

So, the main goal here is to predict whether the individual has any heart problems, further leading towards heart failure. People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

# ACKNOWLEDGEMENT

I take upon this opportunity to acknowledge the many people whose prayers and support meant a lot to me.

I am deeply indebted to my mentor Mr. Sumit Chatterjee who motivated me along the way.

I would like to thank all my teachers who help me in this project.

I further thank my friends.

My heartfelt thanks to parents who supported me a lot. I owe my sincere gratitude towards the almighty God.

Finally, I would like to wind up by paying my heartfelt thanks to AICRA institute who provided me with this great opportunity.

Shabista Mehrish

# CONTENTS

1) Dataset description
2) Methodology
3) Architecture of Methodology
3) Implementation
4) Import Libraries
5) Load Dataset
6) Summarize the Dataset
7) Data Pre-processing
8) Data Visualization
9) Create a Validation Dataset
10) Build Models
13) Conclusion
14) References

# DATASET DESCRIPTION

This dataset contains the medical records of 299 patients who had heart failure, collected during their follow-up period, where each patient profile has 13 clinical features.

These 13 clinical attributes are pre-processed and trained to predict that there is the presence or absence of the heart disease. The attributes are in the form of numeric type which represents the age of the patient in the numeric values, Age is the most important risk factor in growing the heart diseases; it doubles the risk in “ADOLESCENCE”.

Dataset Attributes: -

Thirteen (13) clinical features: -

- 1)age: age of the patient (years)
- 2)anaemia: decrease of red blood cells or haemoglobin (boolean) 3)high blood pressure: if the patient has hypertension (boolean)
- 4)creatinine phosphokinase (CPK): level of the CPK enzyme in the blood (mcg/L)
- 5)diabetes: if the patient has diabetes (boolean)
- 6)ejection fraction: percentage of blood leaving the heart at each contraction (percentage)
- 7)platelets: platelets in the blood (kiloplatelets/mL)
- 8)sex: woman or man (binary)
- 9)serum creatinine: level of serum creatinine in the blood (mg/dL)
- 10)serum sodium: level of serum sodium in the blood (mEq/L)
- 11)smoking: if the patient smokes or not (boolean)
- 12)time: follow-up period (days)
- 13)[target] death event: if the patient deceased during the follow-up period (boolean)

# METHODOLOGY

## **Logistic Regression: -**

Logistic regression is the supervised learning algorithm, which is used to predict the categorical variables or discrete values. It can be used for the classification problems in machine learning, and the output of the logistic regression algorithm can be either Yes or NO, 0 or 1, Red or Blue, etc.

## **K-Nearest Neighbour (KNN): -**

K-Nearest Neighbour is a supervised learning algorithm that can be used for both classification and regression problems. This algorithm works by assuming the similarities between the new data point and available data points. Based on these similarities, the new data points are put in the most similar categories.

## **Decision Tree Algorithm: -**

A decision tree is a supervised learning algorithm that is mainly used to solve the classification problems but can also be used for solving the regression problems. It can work with both categorical variables and continuous variables. It shows a tree-like structure that includes nodes and branches and starts with the root node that expand on further branches till the leaf node.

## **Random Forest Algorithm: -**

Random forest is the supervised learning algorithm that can be used for both classification and regression problems in machine learning. It is an ensemble learning technique that provides the predictions by combining the multiple classifiers and improve the performance of the model.

## **Gradient Boosting Algorithm: -**

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.



# SYSTEM ARCHITECTURE

1. The initial procedure for this project is the collection of data. We have collected the data set from Kaggle, which is available; it is an open-source.
2. The next step after data collection is data pre-processing. In this step, the data is cleansed by removing unnecessary values. It also removes the missing/ null/ corrupted values.
3. After Data is cleansed, the next step is dividing the data, we divide the data into two sets, Training data and testing data. The values must deal before we start to construct the training model. By using training data, we build a model for prediction.
4. Now, we must calculate the accuracy of the model.
5. The final step is predicting the disease.

# IMPLEMENTATION

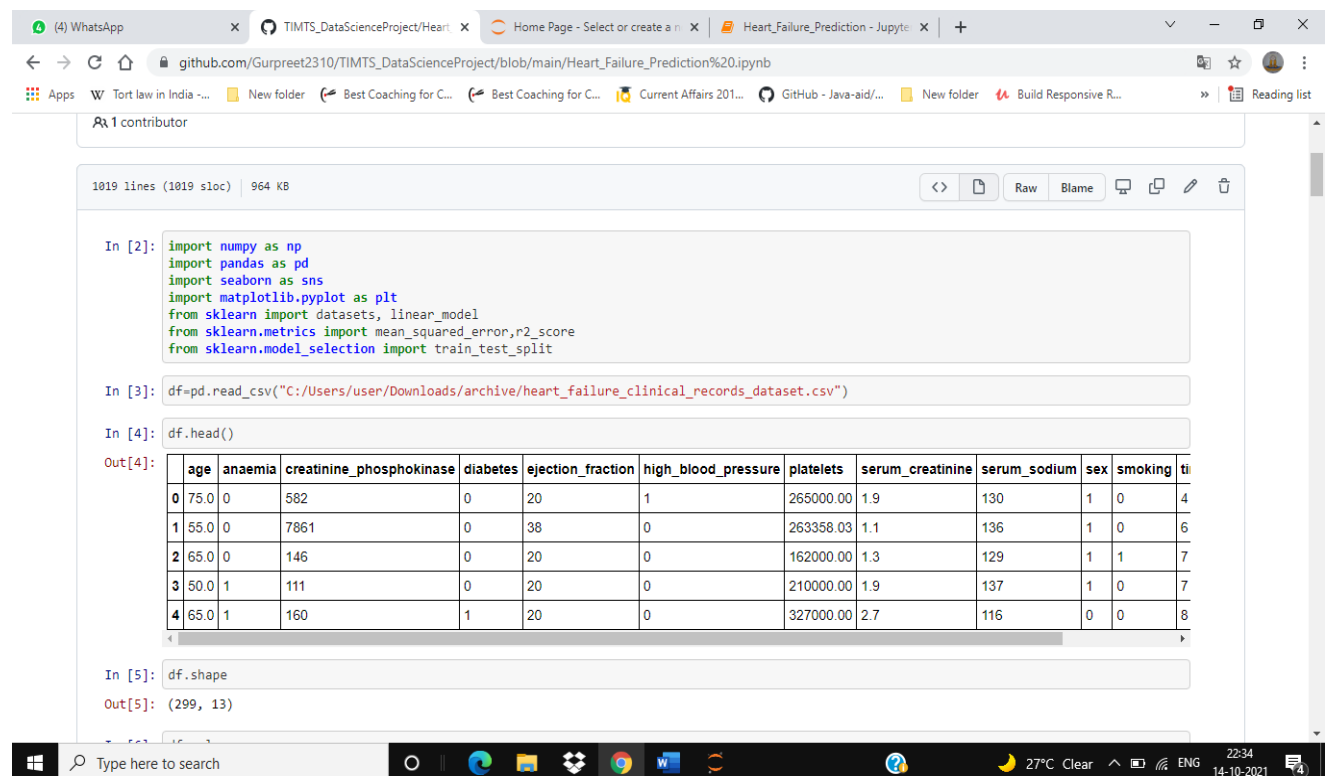
This work used Python programming for this project, as it is a high-level programming language and it has vast libraries and Python automates tasks and makes it efficient. Firstly, we need to install Python then we need to import some libraries, they are:

- 1) Numpy: Numpy is used for multi-dimensional arrays, it does element to element operations and it also has different methods for processing arrays.
- 2) Panda: Pandas is one of the highly used python libraries, it provides high performance. It manipulates data and it makes data analysis fast and easy.
- 3) Sklearn: It is most useful library, this library contains lot of efficient tools, it is used to build models like statistical modelling including classification, regression, clustering. After loading required packages, we divide dataset as training and testing as follows, here 80 % of dataset is taken as training and remaining 20 % as to perform test.

# IMPORT LIBRARIES

First, let's import all the modules, functions and objects we are going to use

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [2]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```

```
In [3]: df=pd.read_csv("C:/Users/user/Downloads/archive/heart_failure_clinical_records_dataset.csv")
```

```
In [4]: df.head()
```

Out[4]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8

```
In [5]: df.shape
Out[5]: (299, 13)
```

The interface also shows a browser window at the top with the URL [github.com/Gurpreet2310/TIMTS\\_DataScienceProject/blob/main/Heart\\_Failure\\_Prediction%20.ipynb](https://github.com/Gurpreet2310/TIMTS_DataScienceProject/blob/main/Heart_Failure_Prediction%20.ipynb).

# Load Dataset

We can load the data directly from the UCI Machine Learning repository. We are using pandas to load the data. We will also use pandas next to explore the data both with descriptive statistics and data visualization.

```
df=pd.read_csv("C:/Users/user/Downloads/archive/heart_failure_clinical_records_dataset.csv")
```

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [2]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```

```
In [3]: df=pd.read_csv("C:/Users/user/Downloads/archive/heart_failure_clinical_records_dataset.csv")
```

```
In [4]: df.head()
```

Out[4]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8

```
In [5]: df.shape
Out[5]: (299, 13)
```

The bottom of the image shows a Windows taskbar with the date 14-10-2021 and time 22:34.

# SUMMARIZE THE DATASET

Now it is time to look at the data.

In this step we are going to look at the data a few different ways.

```
print(df.shape)
print(df.head(10))
```

The screenshot shows a Jupyter Notebook interface with the following content:

**In [4]:** `df.head()`

**Out[4]:**

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8

**In [5]:** `df.shape`

**Out[5]:** (299, 13)

**In [6]:** `df.columns`

**Out[6]:** Index(['age', 'anaemia', 'creatinine\_phosphokinase', 'diabetes', 'ejection\_fraction', 'high\_blood\_pressure', 'platelets', 'serum\_creatinine', 'serum\_sodium', 'sex', 'smoking', 'time', 'DEATH\_EVENT'], dtype='object')

**In [7]:** `df.isnull().sum()`

**Out[7]:**

age	0
anaemia	0
creatinine_phosphokinase	0
diabetes	0
ejection_fraction	0
high_blood_pressure	0
platelets	0
serum_creatinine	0
serum_sodium	0

df.describe()

github.com/Gurpreet2310/TIMTS\_DataScienceProject/blob/main/Heart\_Failure\_Prediction%20.ipynb

In [9]: df.describe()

Out[9]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sod
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000
mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	1.39388	136.625418
std	11.894809	0.496107	970.287881	0.494067	11.834841	0.478136	97804.236869	1.03451	4.412477
min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	0.500000	113.000000
25%	51.000000	0.000000	116.500000	0.000000	30.000000	0.000000	212500.000000	0.900000	134.000000
50%	60.000000	0.000000	250.000000	0.000000	38.000000	0.000000	262000.000000	1.100000	137.000000
75%	70.000000	1.000000	582.000000	1.000000	45.000000	1.000000	303500.000000	1.400000	140.000000
max	95.000000	1.000000	7861.000000	1.000000	80.000000	1.000000	850000.000000	9.400000	148.000000

In [10]: df.dtypes

Out[10]:

age	float64
anaemia	int64
creatinine_phosphokinase	int64
diabetes	int64
ejection_fraction	int64
high_blood_pressure	int64
platelets	float64
serum_creatinine	float64
serum_sodium	int64
sex	int64
smoking	int64
time	int64
DEATH_EVENT	int64
dtype:	object

df.info()

github.com/Gurpreet2310/TIMTS\_DataScienceProject/blob/main/Heart\_Failure\_Prediction%20.ipynb

In [7]: df.isnull().sum()

Out[7]:

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

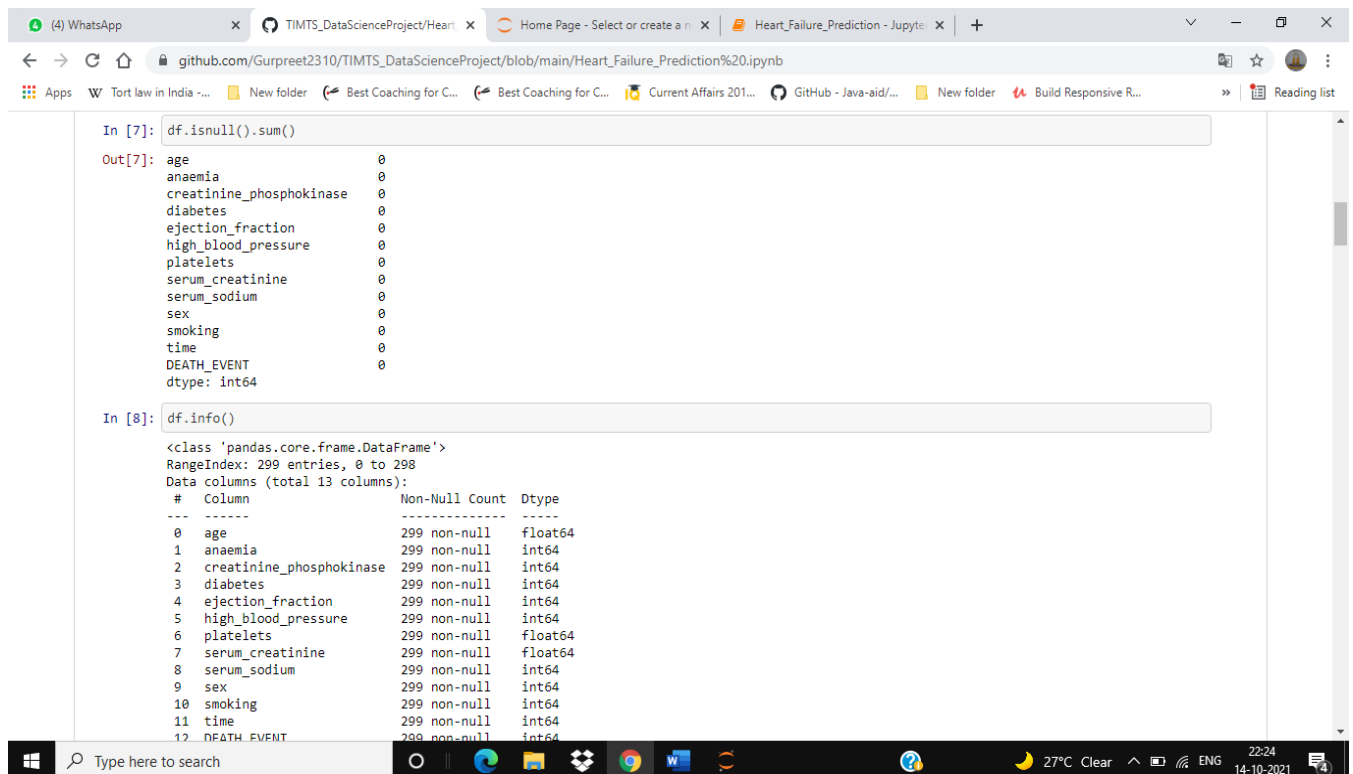
In [8]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   age                  299 non-null   float64
1   anaemia              299 non-null   int64
2   creatinine_phosphokinase  299 non-null   int64
3   diabetes              299 non-null   int64
4   ejection_fraction    299 non-null   int64
5   high_blood_pressure   299 non-null   int64
6   platelets             299 non-null   float64
7   serum_creatinine      299 non-null   float64
8   serum_sodium          299 non-null   int64
9   sex                  299 non-null   int64
10  smoking              299 non-null   int64
11  time                  299 non-null   int64
12  DEATH_EVENT           299 non-null   int64
```

# DATA PRE-PROCESSING

In data pre-processing, it is pivotal to identify and correctly handle the missing values, failing to do this, you might draw inaccurate and faulty conclusions and inferences from the data.

```
heart_data.isnull().sum()
```



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell, labeled 'In [7]:', contains the code `df.isnull().sum()`. The output, labeled 'Out[7]:', shows a series of 13 variables with their respective null counts, all of which are 0. The second cell, labeled 'In [8]:', contains the code `df.info()`. The output shows the DataFrame's structure, including the number of entries (299), the total number of columns (13), and a detailed breakdown of each column's data type and non-null count.

```
In [7]: df.isnull().sum()
Out[7]: 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

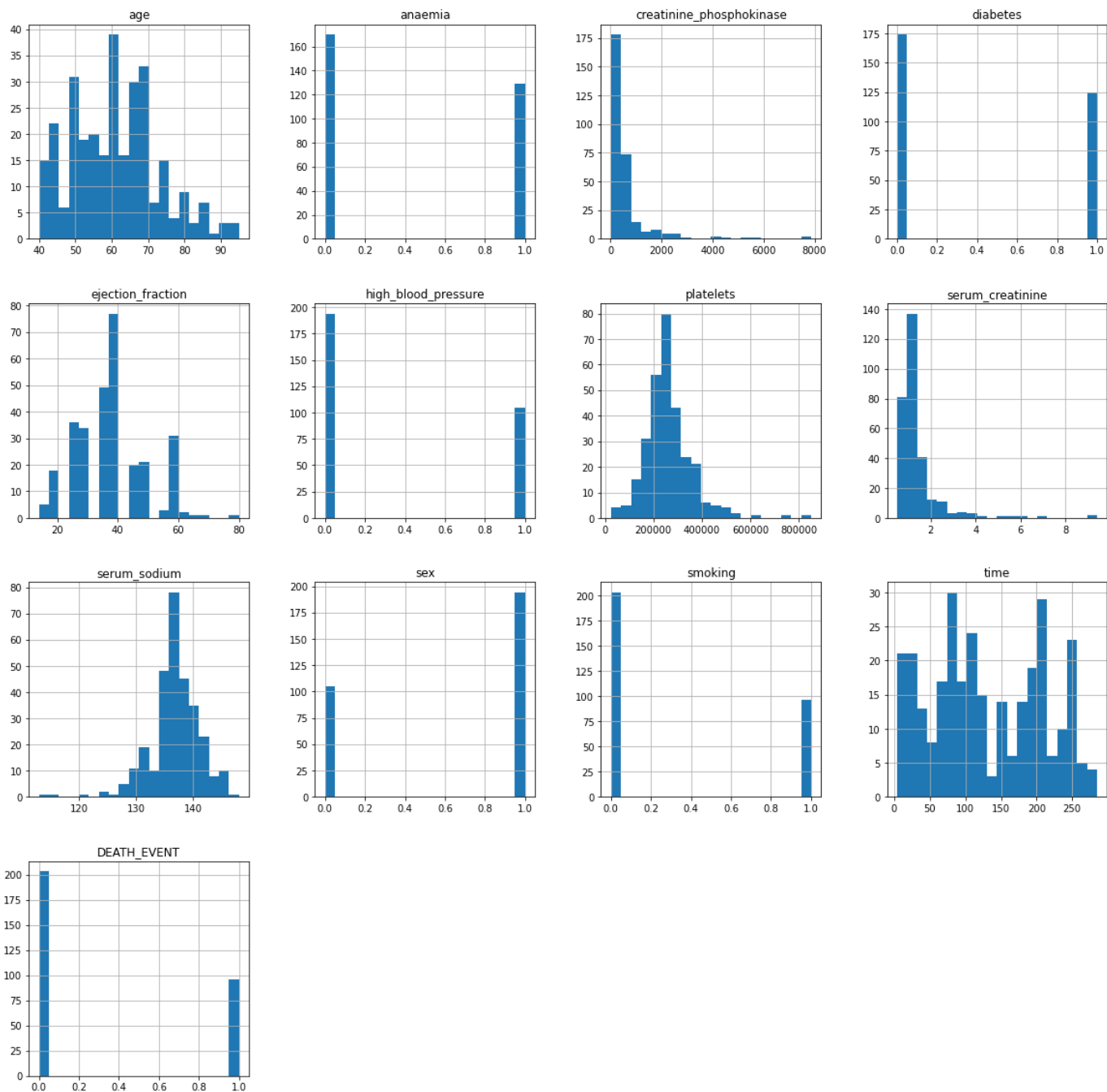
In [8]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   age                   299 non-null   float64
1   anaemia               299 non-null   int64
2   creatinine_phosphokinase  299 non-null   int64
3   diabetes              299 non-null   int64
4   ejection_fraction     299 non-null   int64
5   high_blood_pressure    299 non-null   int64
6   platelets             299 non-null   float64
7   serum_creatinine       299 non-null   float64
8   serum_sodium          299 non-null   int64
9   sex                   299 non-null   int64
10  smoking               299 non-null   int64
11  time                  299 non-null   int64
12  DEATH_EVENT           299 non-null   int64
```

As we can see from above image, there are no missing values (null values) in the dataset so, the data pre-processing techniques are not required here.

# DATA VISUALIZATION

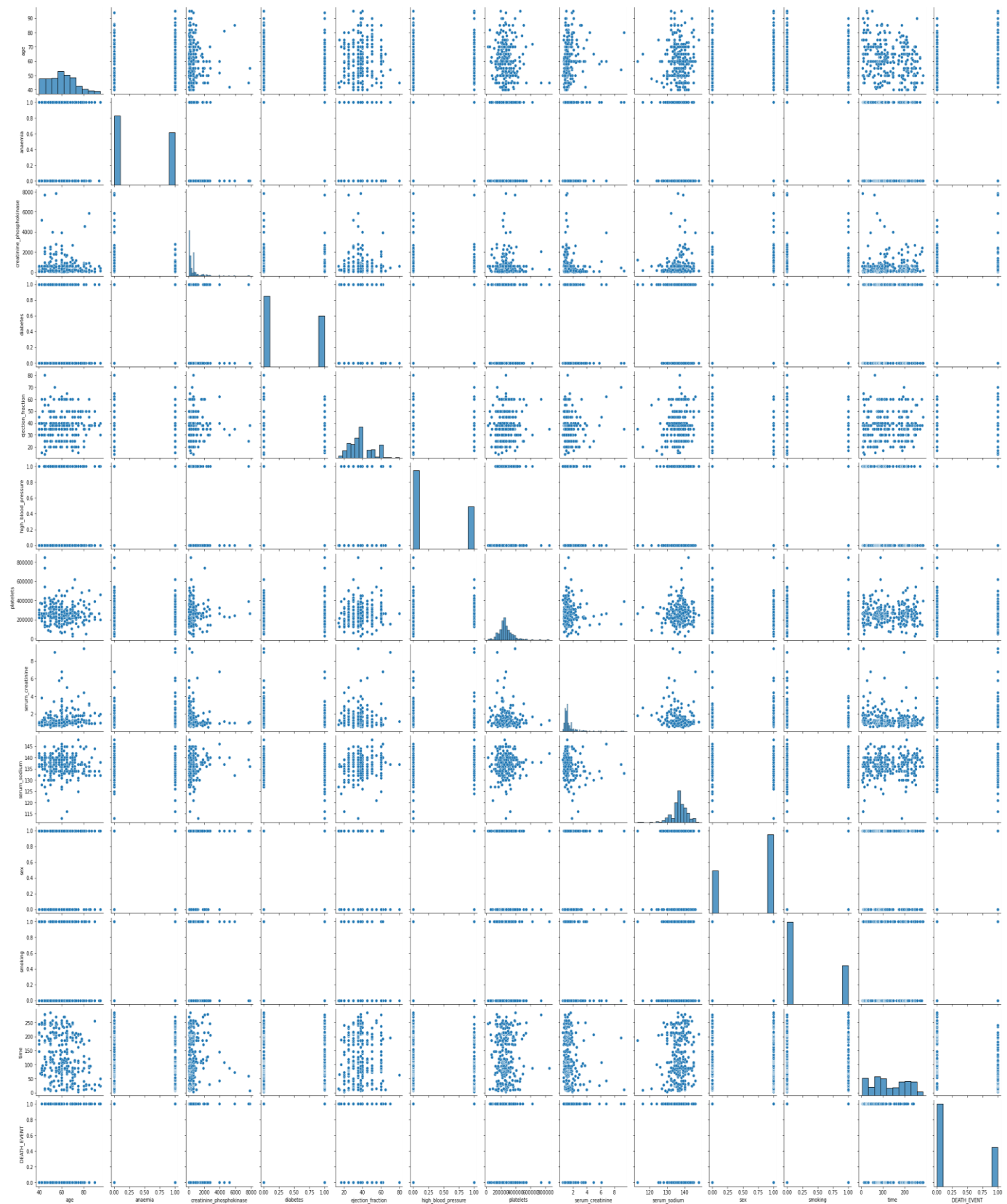
We now have a basic idea about the data. We need to extend that with some visualizations.

```
df.hist(bins = 20, figsize = (20, 20))
```

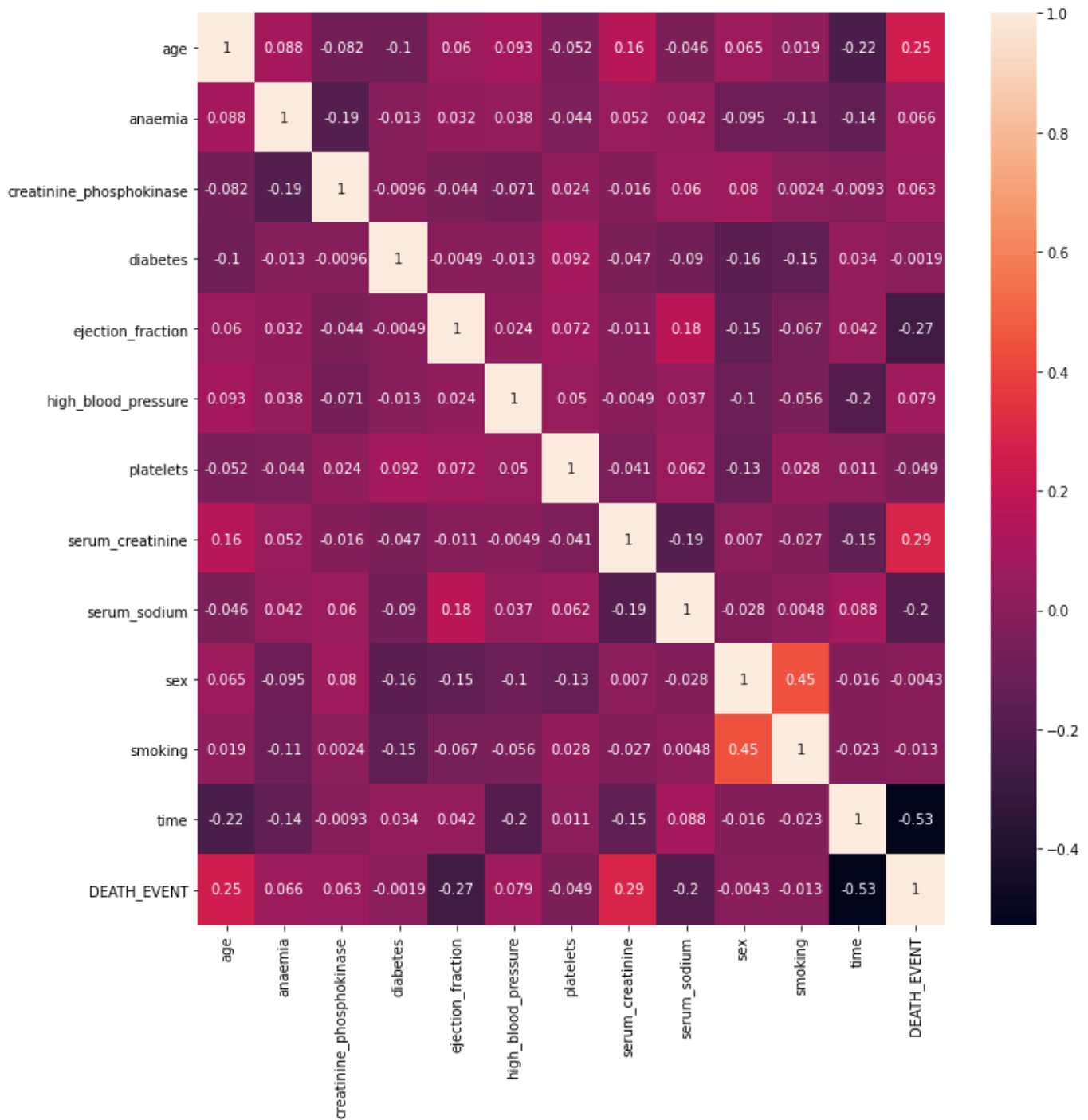




```
sns.pairplot(df)
```



```
corr_matrix = df.corr()
plt.figure(figsize=(12,12))
sns.heatmap(corr_matrix, annot=True)
plt.show()
```



# CREATE A VALIDATION DATASET

We need to know that the model we created is good.

Later, we will use statistical methods to estimate the accuracy of the models that we create on unseen data. We also want a more concrete estimate of the accuracy of the best model on unseen data by evaluating it on actual unseen data.

That is, we are going to hold back some data that the algorithms will not get to see, and we will use this data to get a second and independent idea of how accurate the best model might be.

We will split the loaded dataset into two, 85% of which we will use to train, evaluate and select among our models, and 15% that we will hold back as a validation dataset.

```
x= df.drop(['DEATH_EVENT'],axis=1)
y=df.DEATH_EVENT

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.15,random_state=25)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

# BUILD MODELS

We don't know which algorithms would be good on this problem or what configurations to use.

We get an idea from the plots that some of the classes are partially linearly separable in some dimensions, so we are expecting generally good results.

Let's test 3 different models:

- Linear Regression Model
- Multi-Variable Regression Model
- Logistic Regression Model

# Linear Regression

The screenshot shows a web browser window with a Jupyter Notebook open. The browser's address bar shows the URL: `github.com/Gurpreet2310/TIMTS_DataScienceProject/blob/main/Heart_Failure_Prediction%20.ipynb`. The notebook has two cells. The first cell, labeled 'In [219]:', contains Python code for Linear Regression using `LinearRegression` from `linear_model`. It prints the coefficients, mean squared error, and coefficient of determination. The output shows coefficients `[-0.00283787 -0.00970604 0.09525122]`, a mean squared error of `0.11`, and a coefficient of determination of `0.50`. The second cell, labeled 'In [233]:', contains Python code for Logistic Regression using `LogisticRegression` from `sklearn.linear_model`. It imports `accuracy_score` and `plot_confusion_matrix` from `sklearn.metrics`. The code prints the 'Linear Regression Success Rate' and plots a confusion matrix. The output shows a success rate of `0.50` and a confusion matrix plot.

**Linear Regression**

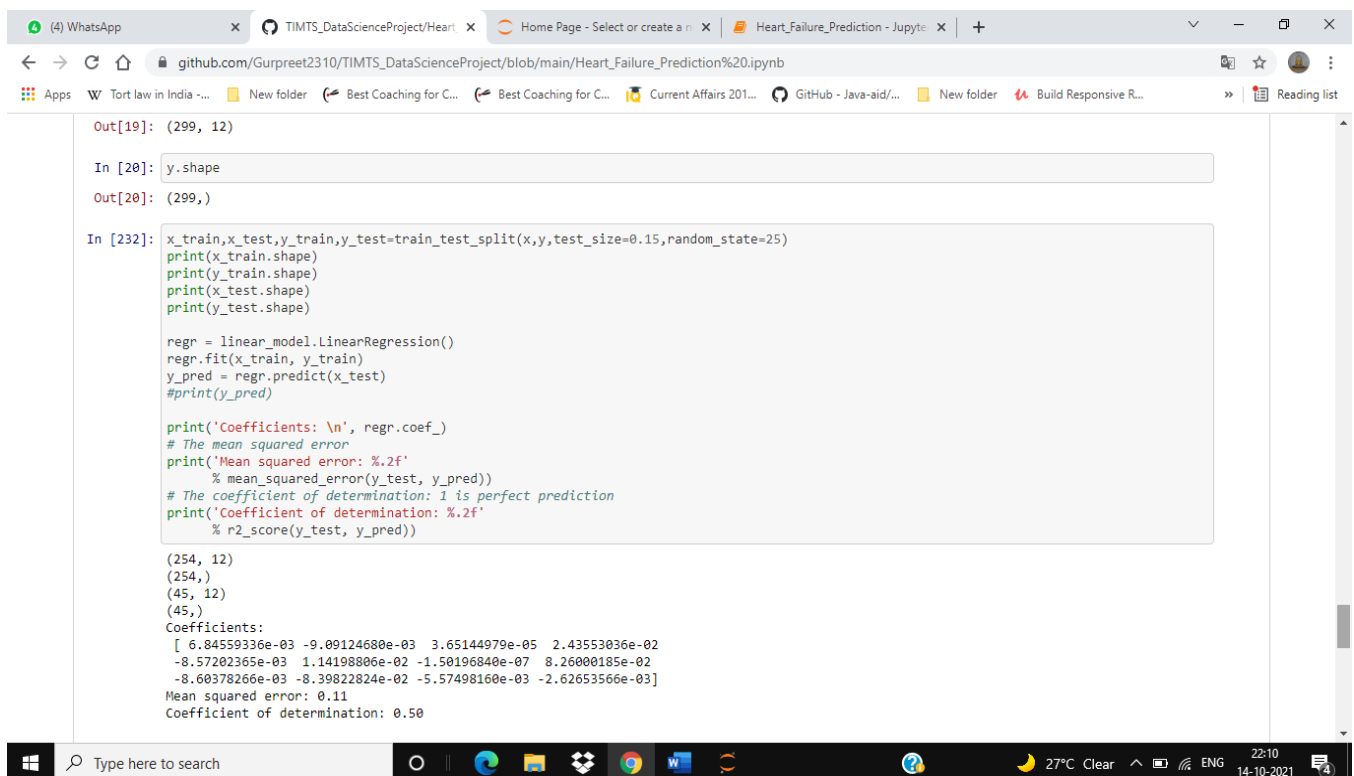
```
In [219]: x1=pd.DataFrame(df[['time','ejection_fraction','serum_creatinine']])
y1=df.DEATH_EVENT
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.1,random_state=2)
regr = linear_model.LinearRegression()
regr.fit(x1_train, y1_train)
y1_pred = regr.predict(x1_test)
#print(y1_pred)
print('Coefficients: \n', regr.coef_)
# The mean squared error
print('Mean squared error: %.2f'
      % mean_squared_error(y1_test, y1_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'
      % r2_score(y1_test, y1_pred))
```

Coefficients:  
[-0.00283787 -0.00970604 0.09525122]  
Mean squared error: 0.11  
Coefficient of determination: 0.50

**Logistic Regression**

```
In [233]: from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, plot_confusion_matrix
lr=LogisticRegression(max_iter=10000)
lr.fit(x_train,y_train)
p1=lr.predict(x_test)
s1=accuracy_score(y_test,p1)
print("Linear Regression Success Rate :", "{:.2f}%".format(100*s1))
plot_confusion_matrix(lr, x_test, y_test)
plt.show()
```

# Multi-Variable Regression



# Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, plot_confusion_matrix
lr=LogisticRegression(max_iter=10000)
lr.fit(x_train,y_train)
p1=lr.predict(x_test)
s1=accuracy_score(y_test,p1)
print("Linear Regression Success Rate :", "{:.2f}%".format(100*s1))
plot_confusion_matrix(lr, x_test, y_test)
plt.show()
```



RESULT=86.67%

# CONCLUSION

Heart is the most essential organ of the human body and day by day the loss of Human Life is increasing exponentially due to heart failure. It was experimentally found that the Global pandemic Corona Virus causes heart injury among a lot of patients. Hence there is an urgent need for research to focus into the causes for heart failure and to design a robust prediction system to detect at early stage so that loss of life can be avoided.

Even though there were many heart diseases prediction systems available at present but each one has its own limitations. The main objective of this research work is to overcome the difficulty faced by other researchers and to build a robust system which works efficiently and will be able to predict accurately the possibility of heart attack at very early stage.

This research work could be able to design a very robust and accurate model to predict the possibility of heart failure in the current scenario. By using the Logistic Regression this model could be able to predict with an accuracy of about 86.6% which is highest as compared to other algorithms.



# REFERENCES

- <https://www.kaggle.com/andrewmvd/heart-failure-clinical-data>
- <https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- <https://stackoverflow.com/questions/38963734/>
- [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.plotting.scatter\\_matrix.html](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.plotting.scatter_matrix.html)
- <https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a45178970be7>
- <https://www.datacamp.com/community/tutorials/random-forests-classifier-python>
- <https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-data-visualization-in-python/#:~:text=%20%20%20%20Bar%20Charts%20A%20charts%20are,Maps%20Temperature%20maps%20represent...%206%20Line%20Plot%20More%20>
- <https://www.geeksforgeeks.org/support-vector-machine-algorithm/>
- <https://www.educba.com/svm-algorithm/>
- <https://stackoverflow.com/questions/22533397/f1-score-in-python-code>
- <https://www.springboard.com/blog/ai-machine-learning/14-essential-machine-learning-algorithms/>
- <https://datascience.stackexchange.com/questions/28574/decision-tree-classifier-object-has-no-attribute-importances>
- [https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.subplots.html](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html)

- <https://www.bing.com/images/search?q=aicra&form=HDRSC2&first=1&tsc=ImageBasicHover>
- [https://en.wikipedia.org/wiki/Heart\\_failure](https://en.wikipedia.org/wiki/Heart_failure)
- <https://www.mayoclinic.org/diseases-conditions/heartfailure/symptoms-causes/syc-20373142>
- <https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records>