Tech Saksham

Final Project Report

Heart Failure Prediction

MIT Arts & Science college for women

Roll.no	Name
1	S. GOMATHI
2	N. HARITHA
3	S. PREETHIKA
4	C.SUBIKSHAA

Mayank Shrivastava	**Trainer Name**
Mayank Shrivastava	**Master Trainer**

ABSTRACT

- In this report, I visualize the distribution of the variables in the Heart Failure Prediction dataset and explore their relationship with the target variable **DEATH_EVENT**. Based on the result of this analysis, I define several machine learning models to compare their performance on this dataset to predict the target variable using the most correlated subset of variables from the dataset.

Discussion Regards:

1.Basic Python python packages like

pandas, numpy, matplotlib, warnings, sci-kit learn, plotly, mlxtend, knn, preprocessing,logistic regression, graph_objs.

2.Pie Charts

3.Heat Map

4.Data Modeling

5. Algorithm: Logistic Regression, Evaluation Metrics: Confusion Matrix

6.preprocesssing

7.seaborn

8.data analysis

9.train/test, split

10.Algorithm: Decision Tree Classifier

11.K Nearest Neigbors

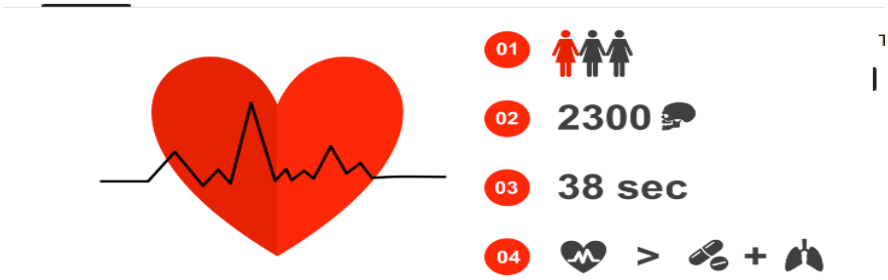
Index

CHAPTER 1

INTRODUCTION

- Heart failure is a serious medical condition that affects millions of people worldwide. It occurs when the heart is unable to pump enough blood to meet the body's needs. Predicting heart failure can be challenging but with the help of artificial intelligence and python programming, it becomes possible.
- In this presentation, we will discuss how AI and machine learning algorithms can be used to predict heart failure in patients using data analysis techniques.
- Before we dive into the details of predicting heart failure using AI, let's first understand what heart failure is and its causes. Heart failure occurs when the heart muscle

becomes weak or damaged, leading to reduced blood flow to the body's organs and tissues.



- There are several causes of heart failure, including high blood pressure, coronary artery disease, heart attack, diabetes, and obesity. Early detection and treatment of heart failure can improve outcomes and prevent complications such as stroke and kidney damage.
- **The correct prediction of heart disease can prevent life threats, and incorrect prediction can prove to be fatal at the same time.**
- A quiet Significant amount of work related to the diagnosis of Cardiovascular Heart failure using Machine Learning algorithms has motivated this work. This paper contains a brief literature survey. An efficient cardiovascular disease prediction has been made by using various algorithms some of them include Logistic Regression, KNN, Decision Tree Classifier, Etc. It can be seen in Results that each algorithm has its strength to register the defined objectives.

Data set:

We utilize a common dataset in public: https://www.kaggle.com/datasets/surendrakumar1/heart-

failure-clinical-records-datasetcsv . This data contains the following input characteristics and prediction targets. The target we want to predict is the Death Event. The features have Age, anaemia, creatinine_phosphokinase, diabetes,ejectiion_fraction, high_blood_pressure, platelets,serum_creatinine, serum_sodium,sex,smoking,time,DEATH_EVENT. I will try some machine learning algorithms later to obtain a uniform sample distribution. These datasets also have 299 rows and 13 columns.

- age: Age of the patient
- anaemia: If the patient had the haemoglobin below the normal range
- creatinine_phosphokinase: The level of the creatine phosphokinase in the blood in mcg/L
- diabetes: If the patient was diabetic
- ejection_fraction: Ejection fraction is a measurement of how much blood the left ventricle pumps out with each contraction
- high_blood_pressure: If the patient had hypertension
- platelets: Platelet count of blood in kiloplatelets/mL
- serum_creatinine: The level of serum creatinine in the blood in mg/dL
- serum_sodium: The level of serum sodium in the blood in mEq/L
- sex: The sex of the patient
- smoking: If the patient smokes actively or ever did in past
- time: It is the time of the patient's follow-up visit for the disease in months
- DEATH_EVENT: If the patient deceased during the follow-up period

Chapter 2

Required libraries:

- import warnings

warnings.filterwarnings("ignore")

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.graph_objs as go

import plotly.express as px

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix,accuracy_score

from sklearn.linear_model import LogisticRegression

from sklearn import preprocessing

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

```
# Importing the dataset
```

```
heart_data=pd.read_csv("heart_failure_clinical_records_dataset(1).csv)
```

 # To Display the data set

```
heart_data
```

 # The dataset will be appeared

In [6]:
```
heart_data
```

Out[6]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	tim
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...
294	62.0	0	61	1	38	1	155000.00	1.1	143	1	1	27
295	55.0	0	1820	0	38	0	270000.00	1.2	139	0	0	27
296	45.0	0	2060	1	60	0	742000.00	0.8	138	0	0	27
297	45.0	0	2413	0	38	0	140000.00	1.4	140	1	1	28
298	50.0	0	196	0	45	0	395000.00	1.6	136	1	1	28

299 rows × 13 columns

 # Let's look at the top 4 rows

```
heart_data.head()
```

```
In [7]: heart_data.head()
```

Out[7]:

	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

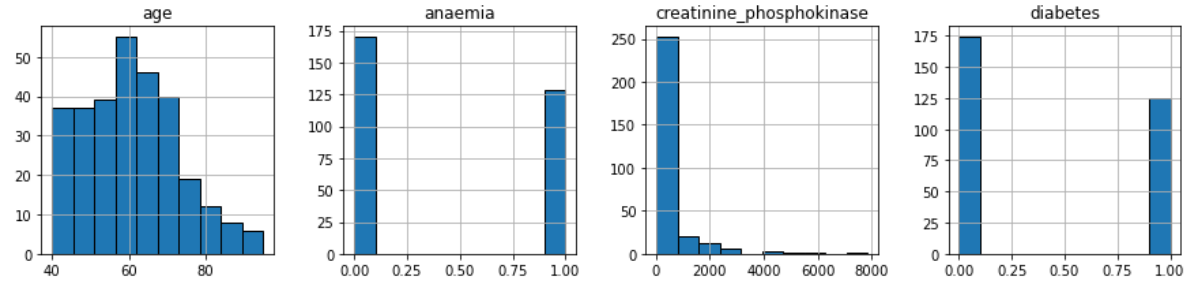
Visualization:

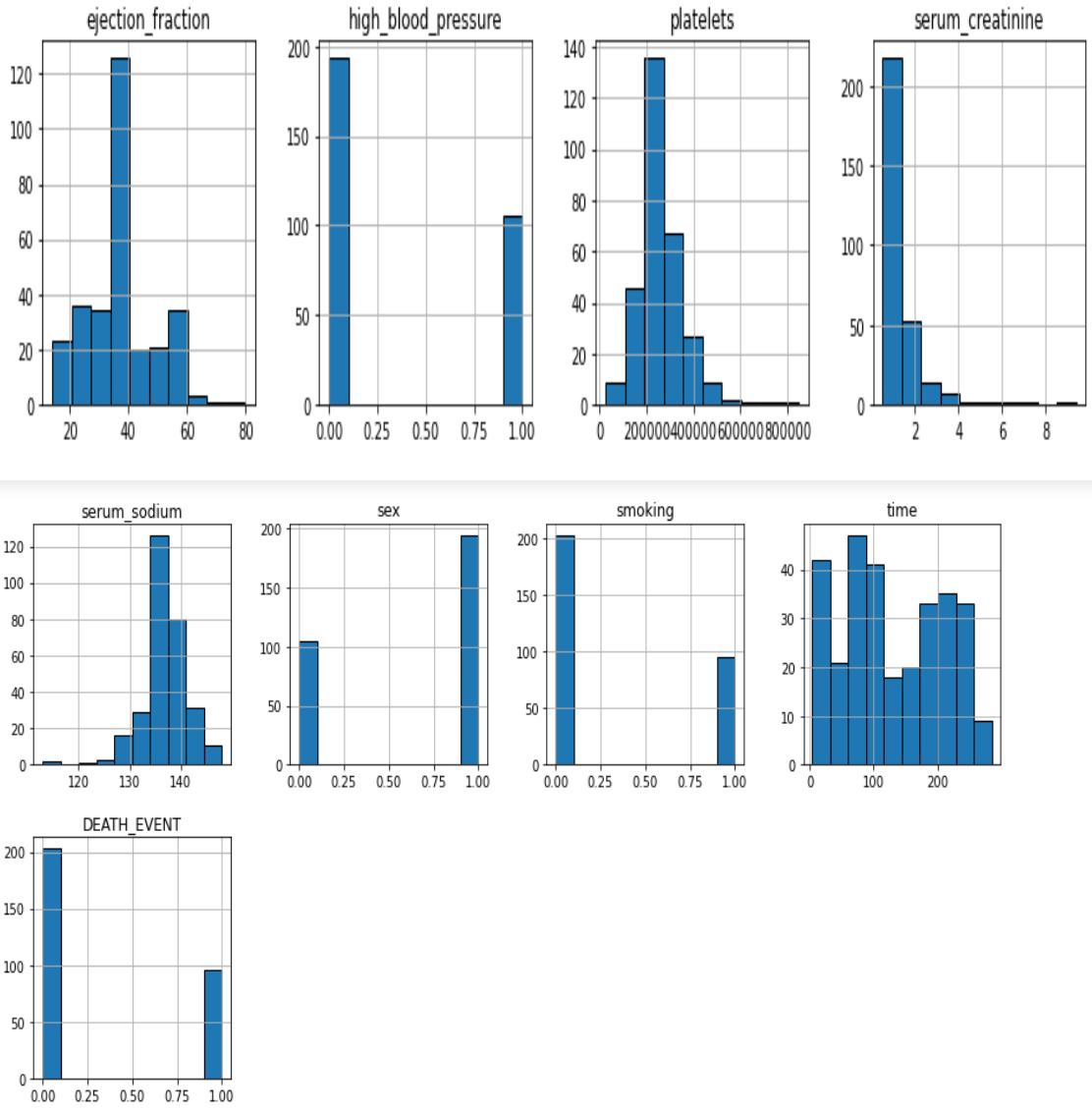
 # To visualize the dataset into graph

```
heart_data.hist(figsize=(15,15),edgecolor='black');
```

Visualization

```
In [9]: heart_data.hist(figsize=(15,15),edgecolor='black');
```





 # Checking for null values

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

```
In [10]: heart_data.isnull().sum()

Out[10]: 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
```

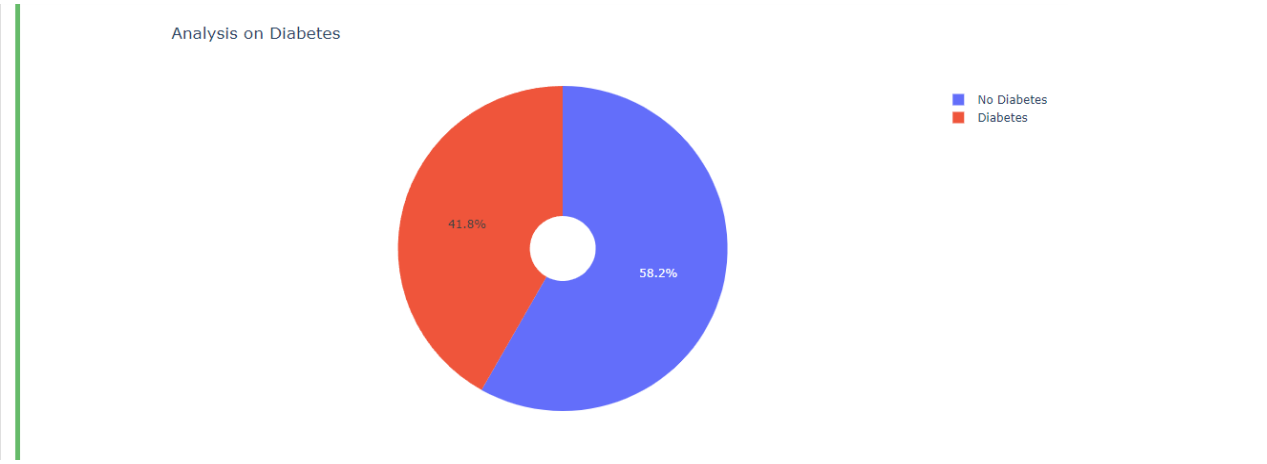
Chapter 3

Pie charts & Heat map:

 # Now create a pie chart for Death and Diabetes Analysis

Analysis on Diabetes:

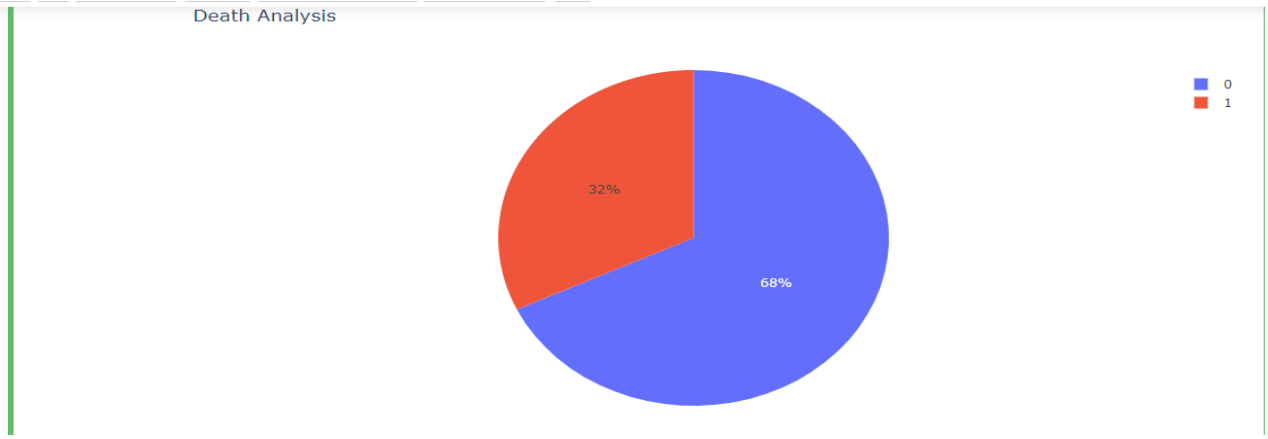
```
import plotly.graph_objs as go
labels = ['No Diabetes','Diabetes']
diabetes_yes = heart_data[heart_data['diabetes'] ==1]
diabetes_no = heart_data[heart_data['diabetes']==0]
values = [len(diabetes_no), len(diabetes_yes)]
fig = go.Figure(data=[go.Pie(labels=labels, values=values,
hole=.2)])
fig.update_layout(
   title_text="Analysis on Diabetes")
fig.show()
```



From the pie chart of Diabetes, the percentage of the No diabetes persons are **58.2%** and the diabetes persons are **41.8%**. In pie chart the colour blue describes the number of non-diabetes persons and red color describes the number of diabetes persons in dataset.

Death Analysis:

```
import plotly.express as px
fig=px.pie(heart_data,values='diabetes',names='DEATH_EVENT',title
='Death Analysis')
fig.show()
```



Gender Vs DEATH_EVENT:

From the Gender distribution of the chart and Gender Vs DEATH_EVENT chart, we have known about that how many females or males are survived and dead.

 # Create subplots: use 'domain' type for Pie subplot

 # Use `hole` to create a donut-like pie chart

 # Add annotations in the center of the donut pies.

#Code

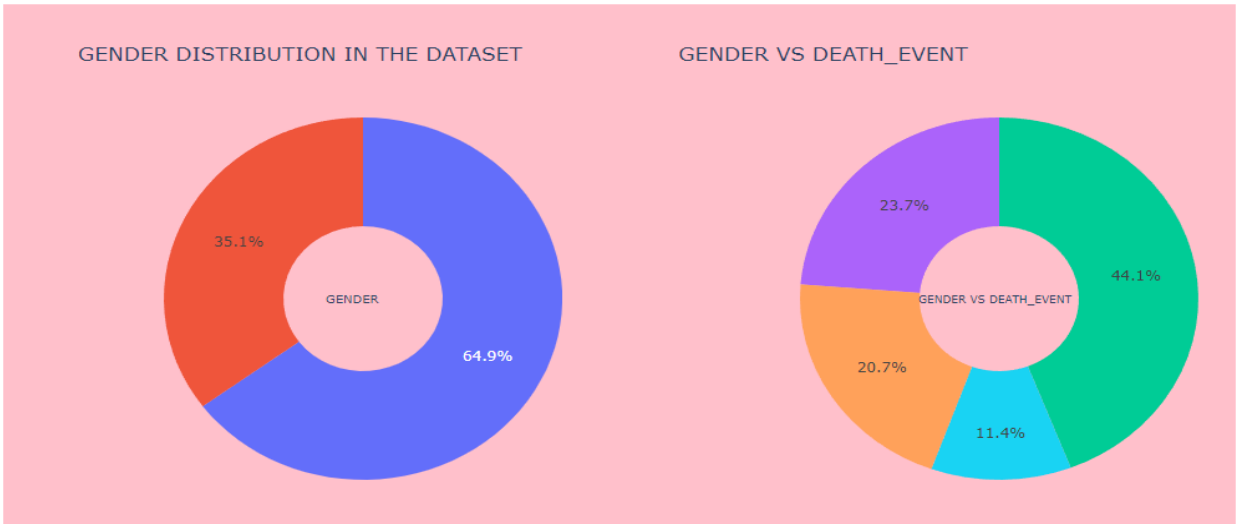
```
import plotly.graph_objects as go

from plotly.subplots import make_subplots


d1 = heart_data[(heart_data["DEATH_EVENT"]==0) &
(heart_data["sex"]==1)]

d2 = heart_data[(heart_data["DEATH_EVENT"]==1) &
(heart_data["sex"]==1)]

d3 = heart_data[(heart_data["DEATH_EVENT"]==0) &
(heart_data["sex"]==0)]

d4 = heart_data[(heart_data["DEATH_EVENT"]==1) &
(heart_data["sex"]==0)]


label1 = ["Male","Female"]
```

```
label2 = ['Male - Survived','Male - Died', "Female -  Survived", "Female - Died"]


values1 = [(len(d1)+len(d2)), (len(d3)+len(d4))]
values2 = [len(d1),len(d2),len(d3),len(d4)]


# Create subplots: use 'domain' type for Pie subplot
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=label1, values=values1, name="GENDER"),
              1, 1)
fig.add_trace(go.Pie(labels=label2, values=values2, name="GENDER VS DEATH_EVENT"),
              1, 2)


# Use `hole` to create a donut-like pie chart
fig.update_traces(hole=.4, hoverinfo="label+percent")


fig.update_layout(
    title_text="GENDER DISTRIBUTION IN THE DATASET \
                GENDER VS DEATH_EVENT",
```

 # Add annotations in the center of the donut pies.

```
    annotations=[dict(text='GENDER', x=0.19, y=0.5,
font_size=10, showarrow=False),
                 dict(text='GENDER VS DEATH_EVENT', x=0.84,
y=0.5, font_size=9, showarrow=False)],
    autosize=False,width=1200, height=500,
paper_bgcolor="pink")

fig.show()
```

#output:

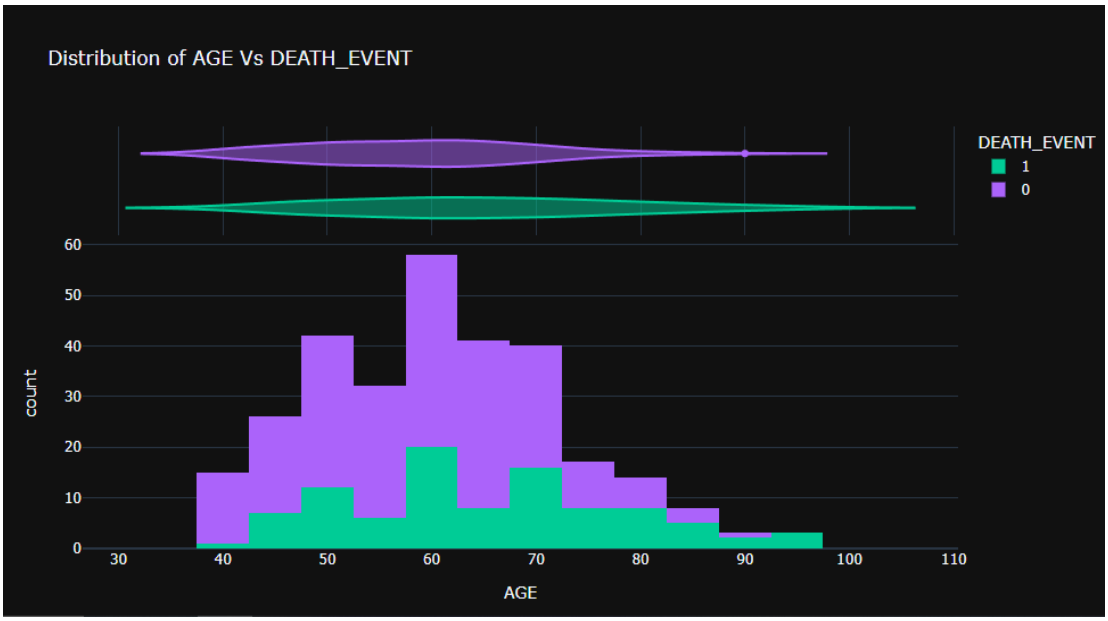


Age VS Death_count:

#Code

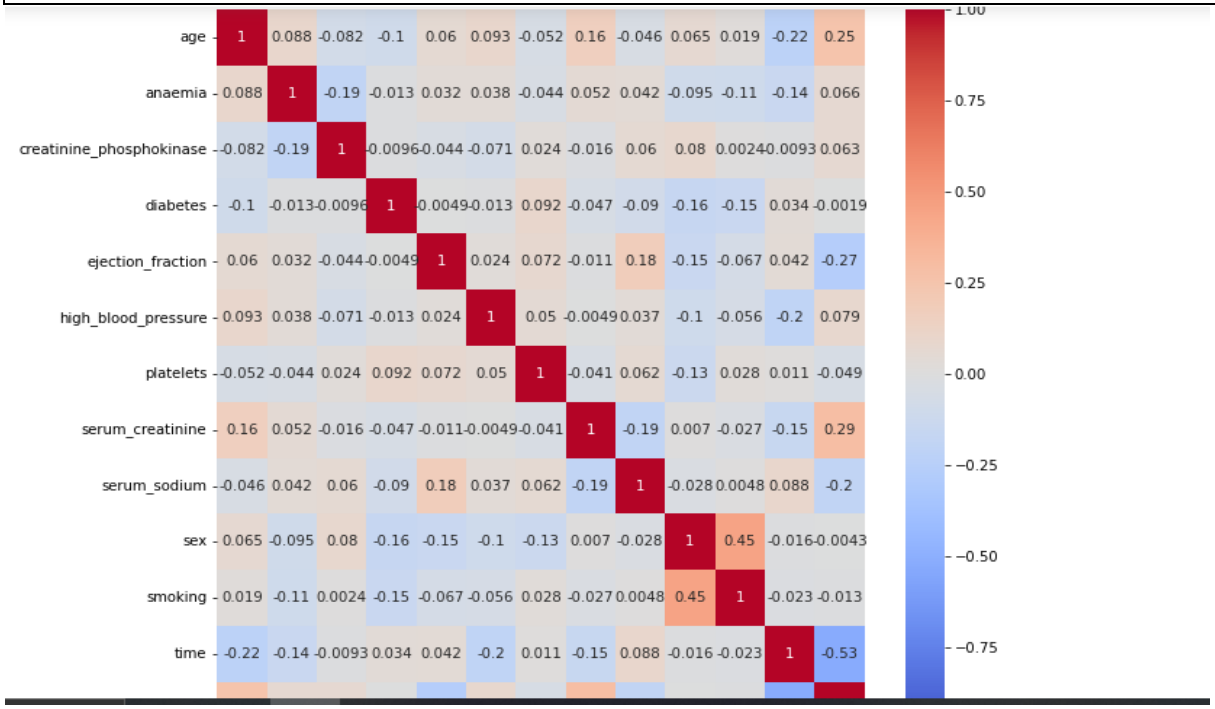
```
import plotly.express as px
fig = px.histogram(heart_data, x="age", color="DEATH_EVENT", marginal="violin", hover_data=heart_data.columns,
                   title ="Distribution of AGE Vs DEATH_EVENT",
                   labels={"age": "AGE"},
                   template="plotly_dark",
                   color_discrete_map={"0": "RebeccaPurple", "1": "MediumPurple"}
                  )
fig.show()
```

#ouput:



Heat Map:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,10))
sns.heatmap(heart_data.corr(),vmin=-1, cmap='coolwarm',annot=True);
```



Notable points:

- Time of the patient's follow-up visit for the disease is crucial in as initial diagnosis with cardiovascular issue and treatment reduces the chances of any fatality. It holds and inverse relation.
- Ejection fraction is the second most important feature. It is quite expected as it is basically the efficiency of the heart.
- Age of the patient is the third most correlated feature. Clearly as heart's functioning declines with ageing

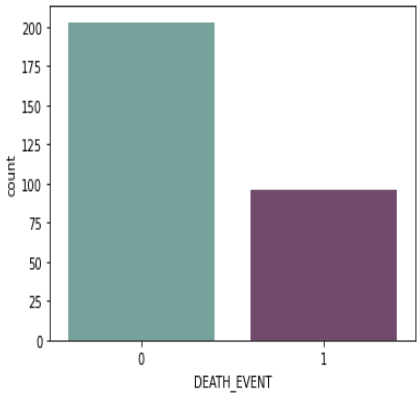
DATA ANALYSIS:

#To analyze the data of DEATH_EVENT

```
cols= ["#6daa9f","#774571"]
sns.countplot(x= heart_data["DEATH_EVENT"], palette= cols)
```

DATA ANALYSIS

```
In [27]: cols= ["#6daa9f","#774571"]
         sns.countplot(x= heart_data["DEATH_EVENT"], palette= cols)

Out[27]: <AxesSubplot:xlabel='DEATH_EVENT', ylabel='count'>
```



Chapter 4:

Data Modeling & Algorithms:

Logistic Regression:

Logistic regression is a classification model, and it is often used in dichotomy. Logistic Regression is one of algorithms of ML to solve binary (0 or 1) problems, which is used to estimate the likelihood of some things.

#import library

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
```

```
Feature=['time','ejection_fraction','serum_creatinine']
x=heart_data[Feature]
y=heart_data["DEATH_EVENT"]
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=2)
```

```
from sklearn.linear_model import LogisticRegression
```

```
log_re=LogisticRegression()
```

```
log_re.fit(xtrain,ytrain)
log_re_pred=log_re.predict(xtest)
```

```
log_re.fit(xtrain,ytrain)
log_re_pred=log_re.predict(xtest)
```

```
log_acc=accuracy_score(ytest,log_re_pred)
print("Logistic Accuracy Score: ","{:.2f}%".format(100*log_acc))
```

Logistic Accuracy Score: 90.00%

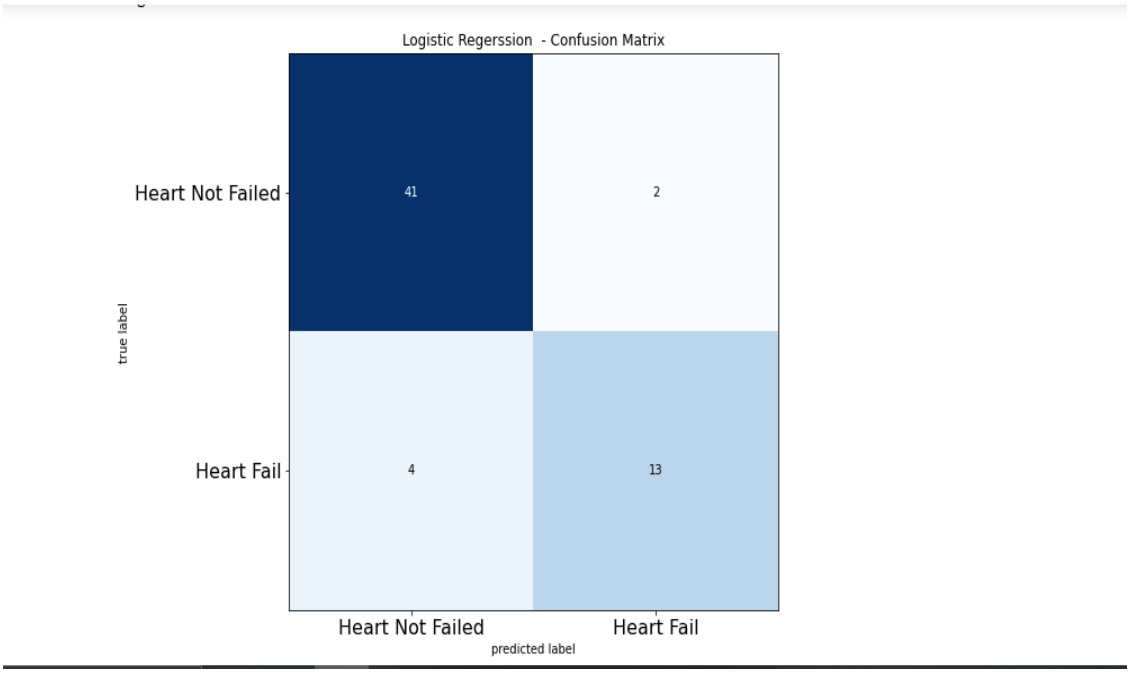
#Install mlxtend

```
%pip install mlxtend
from mlxtend.plotting import plot_confusion_matrix
```

If the mlxtend package was not in your jupyter notebook use the above command to install the mlxtend.

#Logistic Regression-Confusion Matrix

```
cm = confusion_matrix(ytest, log_re_pred)
plt.figure()
plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.title("Logistic Regerssion  - Confusion Matrix")
plt.xticks(range(2), ["Heart Not Failed","Heart Fail"], fontsize=16)
plt.yticks(range(2), ["Heart Not Failed","Heart Fail"], fontsize=16)
plt.show()
```



Preprocessing:

```
# Defining independent and dependent attributes in training and test sets
```

```
X=heart_data.drop(["DEATH_EVENT"],axis=1)
y=heart_data["DEATH_EVENT"]
```

```
#Setting up a standard scaler for the features and from sklearn import preprocessing
```

```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
col_names = list(X.columns)
s_scaler = preprocessing.StandardScaler()
X_df= s_scaler.fit_transform(X)
X_df = pd.DataFrame(X_df, columns=col_names)
X_df.describe().T
```

Out[25]:

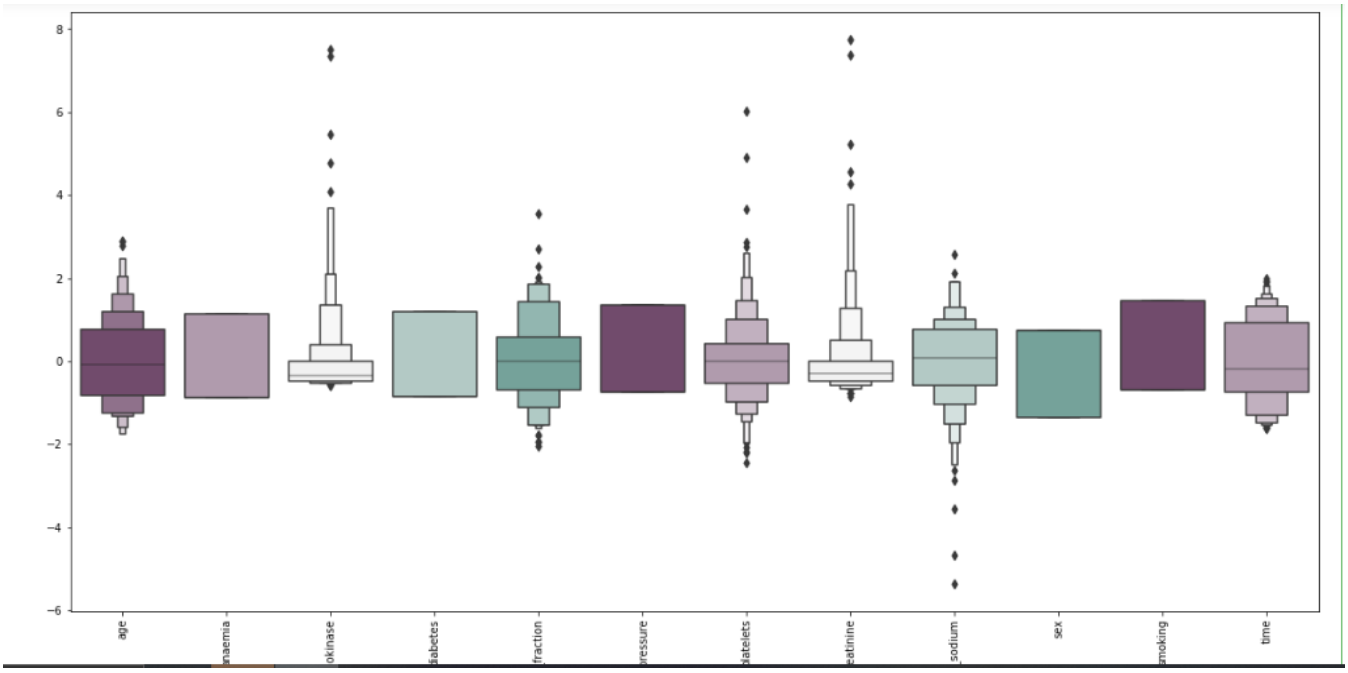
	count	mean	std	min	25%	50%	75%	max
age	299.0	5.265205e-16	1.001676	-1.754448	-0.828124	-0.070223	0.771889	2.877170
anaemia	299.0	3.594301e-16	1.001676	-0.871105	-0.871105	-0.871105	1.147968	1.147968
creatinine_phosphokinase	299.0	3.713120e-18	1.001676	-0.576918	-0.480393	-0.342574	0.000166	7.514640
diabetes	299.0	1.113936e-16	1.001676	-0.847579	-0.847579	-0.847579	1.179830	1.179830
ejection_fraction	299.0	3.341808e-18	1.001676	-2.038387	-0.684180	-0.007077	0.585389	3.547716
high_blood_pressure	299.0	-4.841909e-16	1.001676	-0.735688	-0.735688	-0.735688	1.359272	1.359272
platelets	299.0	1.009969e-16	1.001676	-2.440155	-0.520870	-0.013908	0.411120	6.008180
serum_creatinine	299.0	-2.227872e-18	1.001676	-0.865509	-0.478205	-0.284552	0.005926	7.752020
serum_sodium	299.0	-8.627435e-16	1.001676	-5.363206	-0.595996	0.085034	0.766064	2.582144
sex	299.0	-5.940993e-18	1.001676	-1.359272	-1.359272	0.735688	0.735688	0.735688
smoking	299.0	-3.861645e-17	1.001676	-0.687682	-0.687682	-0.687682	1.454161	1.454161
time	299.0	-1.069379e-16	1.001676	-1.629502	-0.739000	-0.196954	0.938759	1.997038

Seaborn:

#Plotting the scaled features using boxen plots

#import the seaborn library

```
import seaborn as sns
colours =["#774571","#b398af","#f1f1f1" ,"#afcdc7", "#6daa9f"]
plt.figure(figsize=(20,10))
sns.boxenplot(data = X_df,palette = colours)
plt.xticks(rotation=90)
plt.show()
```

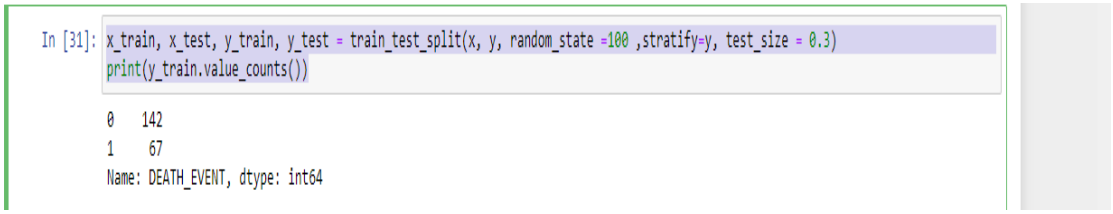


Train/Test Split:

#spliting variables and training and test sets

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

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state =100
,stratify=y, test_size = 0.3)
print(y_train.value_counts())
```

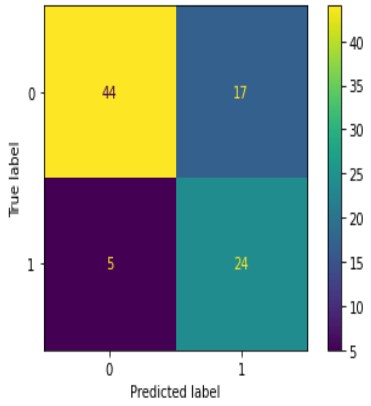


Decision Tree Classifier:

Decision Tree is a tree structure which is binary or non-binary. Each non-leaf node represents a test on a feature attribute, each branch represents the output of this feature attribute on a range of values, and each leaf node stores a category.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, plot_confusion_matrix
from sklearn.tree import DecisionTreeClassifier
list1 = []
for leaves in range(2,10):
    classifier = DecisionTreeClassifier(max_leaf_nodes = leaves, random_state=0,
criterion='entropy')
    classifier.fit(x_train, y_train)
    y_pred = classifier.predict(x_test)
    list1.append(accuracy_score(y_test,y_pred)*100)
print("Decision Tree Classifier Top 5 Success Rates:")
print([round(i, 2) for i in sorted(list1, reverse=True)[:5]])
plot_confusion_matrix(classifier, x_test, y_test)
plt.show()
```

```
Decision Tree Classifier Top 5 Success Rates:
[83.33, 83.33, 83.33, 82.22, 82.22]
```



K Nearest Neighbors:

K Nearest Neighbors is one of easiest algorithms in data analysis and it is a nonparametric statistical method for classification and regression.

```
from sklearn.metrics import accuracy_score, f1_score,confusion_matrix, recall_score, precision_score, classification_report

from sklearn.model_selection import cross_val_score, cross_val_predict

KNN = KNeighborsClassifier(n_neighbors=8)

KNN.fit(x_train, y_train)


y_test_pred_KNN = KNN.predict(x_test)

y_train_pred_KNN = KNN.predict(x_train)


test_acc_KNN = accuracy_score(y_test, y_test_pred_KNN)

train_acc_KNN = accuracy_score(y_train, y_train_pred_KNN)

scores_KNN = cross_val_score(KNN, x_train , y_train , cv = 10, scoring = 'accuracy' )
```

```
precision_score_KNN = precision_score(y_test, y_test_pred_KNN)
recall_score_KNN = recall_score(y_test, y_test_pred_KNN)
f1_score_KNN = f1_score(y_test, y_test_pred_KNN)
conf_KNN = confusion_matrix(y_test, y_test_pred_KNN)
accuracy_score_KNN = accuracy_score(y_test, y_test_pred_KNN)

print("accuracy score:", accuracy_score_KNN)
print("Train set Accuracy: ", train_acc_KNN)
print("Test set Accuracy: ", test_acc_KNN)
print("cv:  %s\n"% scores_KNN.mean())
print("**************************************************")
print("precision_score: ", precision_score_KNN)
print("recall_score: ", recall_score_KNN)
print("f1_score: ", f1_score_KNN)
print("**************************************************")
print("\nReport:\n%s\n"%classification_report(y_test, y_test_pred_KNN))
```

#output:

```
accuracy score: 0.6777777777777778
Train set Accuracy:  0.7177033492822966
Test set Accuracy:  0.6777777777777778
cv:  0.694047619047619

************************************************
precision_score:  0.5
recall_score:  0.20689655172413793
f1_score:  0.2926829268292683
************************************************

Report:
              precision    recall  f1-score   support

           0       0.71      0.90      0.79        61
           1       0.50      0.21      0.29        29

    accuracy                           0.68        90
   macro avg       0.60      0.55      0.54        90
weighted avg       0.64      0.68      0.63        90
```

In KNN the accuracy score was 0.677

Train set Accuracy was 0.7177

Test set Accuracy was 0.6777

Cross val score was 0.694

Precision_score 0.5

Recall_score 0.206

f1_score 0.292

#Decision Tree Classifier

```
print(f'Decision Tree Classifier: {round(sorted(list1, reverse=True)[0])}%')
```

Decision Tree Classifier:83%

#Logistic Regression

```
print(f'Logistic Regression: {round(100*log_acc, 2)} %')
```

Logistic Regression:90.0%

#K Nearest Neighbors

```
print(f'K Nearest  Neighbors: {(accuracy_score_KNN)}  %')
```

K Nearest Neighbors: 0.6777777777777778 %

Reference:

https://www.kaggle.com/code/surajjha101/heart-failure-prediction-svm-and-ann

https://www.kaggle.com/code/sanchitakarmakar/heart-failure-prediction-visualization

https://www.kaggle.com/code/karnikakapoor/heart-failure-prediction-ann

https://www.kaggle.com/code/midouazerty/heart-failure-prediction

Conclusion:

Al and machine learning have great potential for predicting heart failure in patients. By collecting and preprocessing patient data and using machine learning models, healthcare providers can identify patients at risk of heart failure and provide early intervention.However, there are also challenges and limitations to consider, and ethical considerations must be addressed when implementing Al in healthcare settings. With careful consideration and implementation, Al can be a powerful tool for improving patient outcomes and reducing healthcare costs. On this project I utilize the score that prediction of heart failure

using machine learning algorithm such as Decision Tree Clasifier,Logistic Regression,K Nearest Neibors accuracy score.

Heart failure prediction project link:http://localhost:8888/notebooks/Heart%20failure%20%20Prediction.ipynb#