**Prediction Of Heart Diseases Using Machine Learning**

An Engineering Project in Community Service

Phase–II Report

*Submitted by*

Team Members

1)19MIM10026 Shirish Waghmode

2)19MIM10053 Vartika Pandey

3) 19MIM10018 Ananya Saxena

4) 19MIM10013 Anant Kumar Pandey

5) 19MIM10007 Prince Nahar

6) 19BOE1007 Manav Nair

7) 19BCE10311 Ashwini Darade

8) 19BCE10294 Ayush Joshi

*in partial fulfillment of the requirements for the degree of*

*Bachelor of Engineering and Technology*

**

VIT Bhopal University

Bhopal

Madhya Pradesh

April,2022

Bonafide Certificate



Certified that this project report titled “ **Prediction Of Heart Diseases Using Machine Learning**” is the bonafide work of “Team Members Register Number and Name ( 19MIM10026 Shirish Waghmode 19MIM10053 Vartika Pandey 19MIM10013 Anant Kumar Pandey 19MIM10018 Ananya Saxena 19BOE10007 Manav Nair 19MIM10007 Prince Nahar 19BCE10311 Ashwini Darade 19BCE10294 Ayush Joshi .)” who carried out the project work under my supervision.

This project report (Phase II) is submitted for the Project Viva-Voce examination held on 20-04-22

Supervisor

Comments & Signature ( Reviewer 1)

Comments & Signature ( Reviewer 2)

Sl. No. Topic Page No.

1. Introduction 1

1.1 Motivation 2

1.2 Objective 2

2. Existing Work 3

3.Topic of the work

4)Working Principle- 11

4.1 Database selection 11

4.2 Data Pre-Processing 11

4.2.1 Cleaning: 11

a)Null values removal

b)removing outliers

c)cleaning noise in data

4.2.2 Feature Selection: 11

4.2.3 Finding outliers 11

4.3 Data analysis 11

4.4 train-test split of data set 11

4.5 Classifiers Integration 11

4.5 Creating GUI for non - technical users 12

# 

# 

# 

# **1. INTRODUCTION** - Cardiovascular diseases kill approximately 17 million people globally every year, and they mainly exhibit myocardial infarctions and heart failures. Heart failure (HF) occurs when the heart cannot pump enough blood to meet the needs of the body.

A study shows that from 1990 to 2016 the death rate due to heart diseases have increased around 34 percent from 155.7 to 209.1 deaths per one lakh population in India.

Machine learning, in particular, can predict patients’ survival from their data and can individuate the most important features among those included in their medical records.

Cardiovascular diseases kill approximately 17 million people globally every year, and they mainly exhibit as myocardial infarctions and heart failures. Heart failure (HF) occurs when the heart cannot pump enough blood to meet the needs of the body.

A study shows that from 1990 to 2016 the death rate due to heart diseases have increased around 34 percent from 155.7 to 209.1 deaths per one lakh population in India.

Machine learning, in particular, can predict patients’ survival from their data and can individuate the most important features among those included in their medical records.

## **1.1 Motivation-**

A major challenge facing healthcare organizations (hospitals, medical centers) is the provision of quality services at affordable costs. Quality service implies diagnosing patients correctly and administering treatments that are effective. Poor clinical decisions can lead to disastrous consequences which are therefore unacceptable. Hospitals must also minimize the cost of clinical tests. They can achieve these results by employing appropriate computer-based information and/or decision support systems. Most hospitals today employ some sort of hospital information systems to manage their healthcare or patient data . These systems typically generate huge amounts of data which take the form of numbers, text, charts and images. Unfortunately, these data are rarely used to support clinical decision making. There is a wealth of hidden information in these data that is largely untapped. This raises an important question: “How can we turn data into useful information that can enable healthcare practitioners to make intelligent clinical decisions?” This is the main motivation for this research.

.

**1.2 Objective-**

The objective of the study is to effectively predict if the patient suffers from heart disease. The health professional enters the input values from the patients health report the data is fed into model which predicts the probability of having heart disease.The system uses 15 medical parameters such as age, sex, blood pressure, cholesterol, and obesity for prediction. The EHDPS predicts the likelihood of patients getting heart disease. It enables significant knowledge, eg, relationships between medical factors related to heart disease and patterns, to be established. Our main aim is to provide a quick medical diagnosis to the patients living in rural areas. Nowadays it is very useful for the postcode contactless system in rural health services. The goal is to provide access to medical specialists. This system enhances the quality of health care. In the medical field, machine learning can be used for diagnosis, detection and prediction of various diseases. The main goal of this paper is to provide a tool for doctors to detect heart disease at an early stage [5]. This in turn will help to provide effective treatment to patients and avoid severe consequences.The main goal of this research is to find the best accuracy for the prediction of heart disease by using major risk factors based on different classifier algorithms such as Bayesian Optimized Support Vector Machine (BO-SVM), K-Nearest Neighbors (KNN).

**2. Existing Work / Literature Review**

The algorithm to diagnose HF in a non-acute setting is the following. First the probability of HF based on prior clinical history of the patient, the presenting symptoms, physical examination, and resting ECG is estimated.The process of diagnosis of HF can be:

(i) less time consuming,

(ii) supported and

(iii) performed with the same accuracy by the applications of machine learning techniques on the available data.

To classify HF subtypes, machine learning approaches were used.This discovery has the potential to change clinical practice by providing physicians with a new tool to help them forecast whether a heart failure patient will live or die. Indeed, medical physicians may use serum creatinine and ejection fraction to determine whether a patient will live following heart failure. There is numerous work that has been done related to disease prediction systems using different data mining techniques and machine learning algorithms in medical centers.

K. Polaraju et al, proposed Prediction of Heart Disease using Multiple Regression Model and it proves that Multiple Linear Regression is appropriate for predicting heart disease chance. The work is performed using a training data set consisting of 3000 instances with 13 different attributes which has been mentioned earlier. The data set is divided into two parts. 70% of the data are used for training and 30% used for testing. Based on the results, it is clear that the classification accuracy of Regression algorithm is better compared to other

algorithms.

Ashok Kumar Dwivedi et al, recommended different algorithms like Naive Bayes, Classification Tree, KNN, Logistic Regression, SVM and ANN. The Logistic Regression gives better accuracy compared to other algorithms.

Jayami Patel et al, suggested heart disease prediction using data mining and machine learning algorithms. The goal of this study is to extract hidden patterns by applying data mining techniques. The best algorithm J48 based on UCI data has the highest accuracy rate compared to LMT.

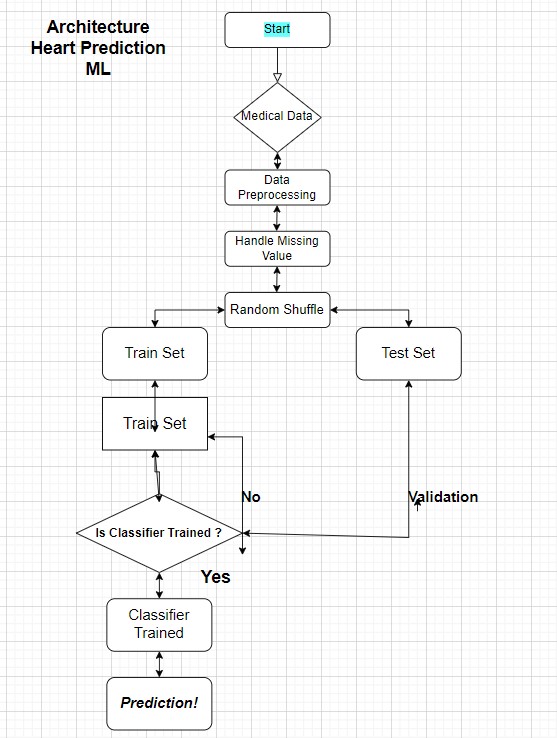
P.Sai Chandrasekhar Reddy et al, proposed Heart disease prediction using ANN algorithm in data mining. Due to increasing expenses of heart disease diagnosis disease, there was a need to develop a new system which can predict heart disease. Prediction model is used to predict the condition of the patient after evaluation on the basis of various parameters like heart beat rate, blood pressure, cholesterol etc.

Ashok Kumar Dwivedi, “Evaluate the performance of different machine learning techniques for prediction of heart disease using ten-fold cross-validation”,

Sharma Purushottam et al, proposed c45 rules and partial tree technique to predict heart disease. This paper can discover a set of rules to predict the risk levels of patients based on given parameters about their health.

# **3. Topic of the work:**

**a) System Design / Architecture**



**4) Working Principle-**

**4.1 Database selection :** after collecting the data from various sources , we need to combine all of them into a single big dataset (only the similar feachers )

**4.2 Data Pre-Processing:** Once we have our data set with us we need to move on to cleaning unnecessary data (noise) **,** Removing null values **,** selecting important features ,etc..

4.2.1 Cleaning:

a)Null values removal

b)removing outliers

c)cleaning noise in data

4.2.2 Feature Selection: we need to select important features for our classifier , we found out that time, ejection\_fraction, serum\_creatinine are important features.

4.2.3 Finding outliers : once we get out important features we need to find outliers in them and remove them if needed.

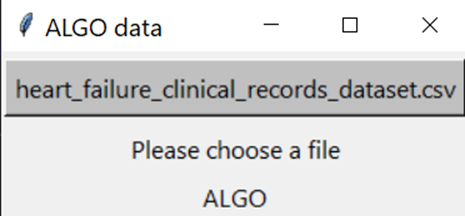
**4.3 Data analysis :** we need some insights of the dataset , distribution of age , death event , Distribution of creatinine\_phosphokinase , and many more.

**4.4 train-test split of data set** : train\_test split of dataset is 80% / 20%

Where 80% is for training and 20% is for testing.

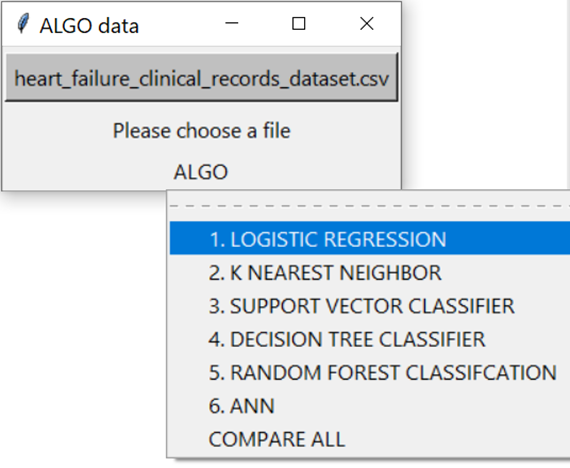
**4.5 Classifiers Integration**: moving on to classification we have used 6 different algorithms namely LOGISTIC REGRESSION , K NEAREST NEIGHBOR , SUPPORT VECTOR CLASSIFIER , DECISION TREE CLASSIFIER , Random forest , ANN.

**4.5 Creating GUI for non - technical users:** doing classification is ok but we need to generalize this so that every one canṣe use this .



Home page :

*Fig 2(Home page of application)*

Menu bar : 

*Fig 3(Menu bar with Home page of application)*

**5.)**

* **Results-**
* **Discussion**

**d) Individual Contribution by members-**

1. **Prince Nahar 19MIM10007-**  Being a part of this Team and Project I have contributed in many ways, This Project allowed me to put myself to the test by making work that showcased my abilities to the most extent possible

My contribution toward this group:

* 1st review : “Topic of work ” PPT slide no.6
* 2 nd review : ” implementation and coding ”and “1st Classifiers” slide no.20 and 21
* Final review : “Outcome and Result ” slides no 38 to 43
* Project report : “Working Principle” report page no .11-12
* Worked on coding part for all 6 of the classifier algorithm

And in the making of our final application completely.

Personal Contribution of Prince nahar 19MIM10007 towards group project in detail.

**1.Started by importing necessary libraries into the terminal**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

**2.Importing the dataset**

dataset = pd.read\_csv('heart\_failure\_clinical\_records\_dataset.csv')

**3.Checking for null values**

dataset.isnull().sum()

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

**4.Feature Selection**

To continue let us first find best Feature by plotting feat\_importance.

An extra-trees classifier.

This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

plt.rcParams['figure.figsize']=15,6

sns.set\_style("darkgrid")

x = dataset.iloc[:, :-1]

y = dataset.iloc[:,-1]

from sklearn.ensemble import ExtraTreesClassifier

import matplotlib.pyplot as plt

model = ExtraTreesClassifier()

model.fit(x,y)

print(model.feature\_importances\_)

feat\_importances = pd.Series(model.feature\_importances\_, index=x.columns)

feat\_importances.nlargest(12).plot(kind='barh')

plt.show()

We will select only 3 features : time, ejection\_fraction, serum\_creatinine

**5.Finding outliers BY plotting boxplot**

Boxplot for ejection\_fraction

sns.boxplot(x = dataset.ejection\_fraction, color = 'teal')

plt.show()

Find out that there were two outliers. And removed them

Boxplot for time

sns.boxplot(x=dataset.time, color = 'teal')

plt.show()

No outliers in time

**6. LETS GET SOME INSIGHTS OF THE DATASET**

Distribution of Age

import plotly.graph\_objects as go

fig = go.Figure()

fig.add\_trace(go.Histogram(

    x = dataset['age'],

    xbins=dict( # bins used for histogram

        start=40,

        end=95,

        size=2

    ),

    marker\_color='#e8ab60',

    opacity=1

))

fig.update\_layout(

    title\_text='AGE DISTRIBUTION',

    xaxis\_title\_text='AGE',

    yaxis\_title\_text='COUNT',

    bargap=0.05, # gap between bars of adjacent location coordinates

    xaxis =  {'showgrid': False },

    yaxis = {'showgrid': False },

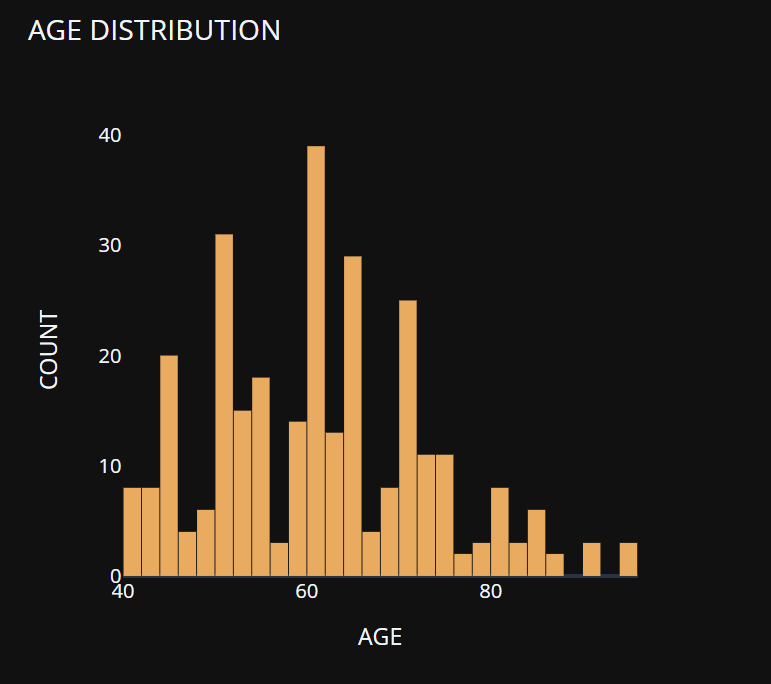
    template = 'plotly\_dark'

)

fig.show()

Output

Now lets categorize the above histogram by DEATH\_EVENT



import plotly.express as px

fig = px.histogram(dataset, x="age", color="DEATH\_EVENT", marginal="violin", hover\_data=dataset.columns,

                   title ="Distribution of AGE Vs DEATH\_EVENT",

                   labels={"age": "AGE"},

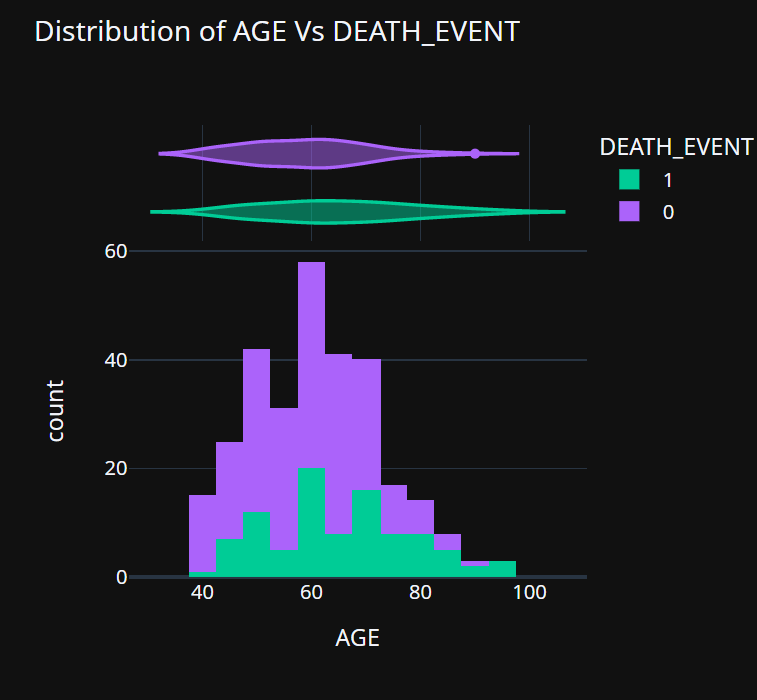
                   template="plotly\_dark",

                   color\_discrete\_map={"0": "RebeccaPurple", "1": "MediumPurple"}

                  )

fig.show()

output

Wider sections of the violin plot represent a higher probability of observations taking a given value, the thinner sections correspond to a lower probability and the value of probability is given by kde value for given x

**Similarly lets get insights of other features as well**

Distribution of creatinine\_phosphokinase

import plotly.graph\_objects as go

fig = go.Figure()

fig.add\_trace(go.Histogram(

    x = dataset['creatinine\_phosphokinase'],

    xbins=dict( # bins used for histogram

        start=23,

        end=582,

        size=15

    ),

    marker\_color='#FE6F5E',

    opacity=1

))

fig.update\_layout(

    title\_text='CREATININE PHOSPHOKINASE DISTRIBUTION',

    xaxis\_title\_text='CREATININE PHOSPHOKINASE',

    yaxis\_title\_text='COUNT',

    bargap=0.05, # gap between bars of adjacent location coordinates

    xaxis =  {'showgrid': False },

    yaxis = {'showgrid': False },

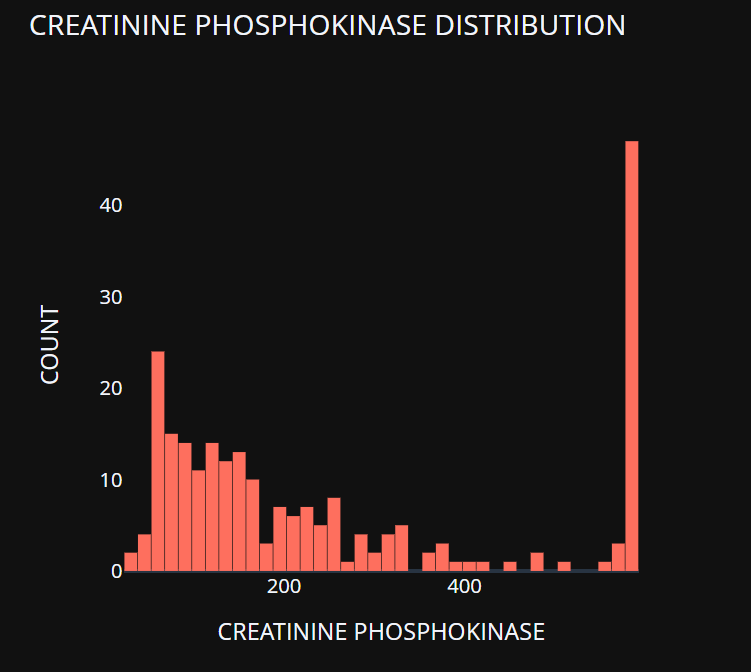
    template = 'plotly\_dark'

)

fig.show()

Output

Now lets categorize the above histogram by



DEATH\_EVENT

import plotly.express as px

fig = px.histogram(dataset, x="creatinine\_phosphokinase", color="DEATH\_EVENT", marginal="violin", hover\_data=dataset.columns,

                   title ="Distribution of CREATININE PHOSPHOKINASE Vs DEATH\_EVENT",

                   labels={"creatinine\_phosphokinase": "CREATININE PHOSPHOKINASE"},

                   template="plotly\_dark",

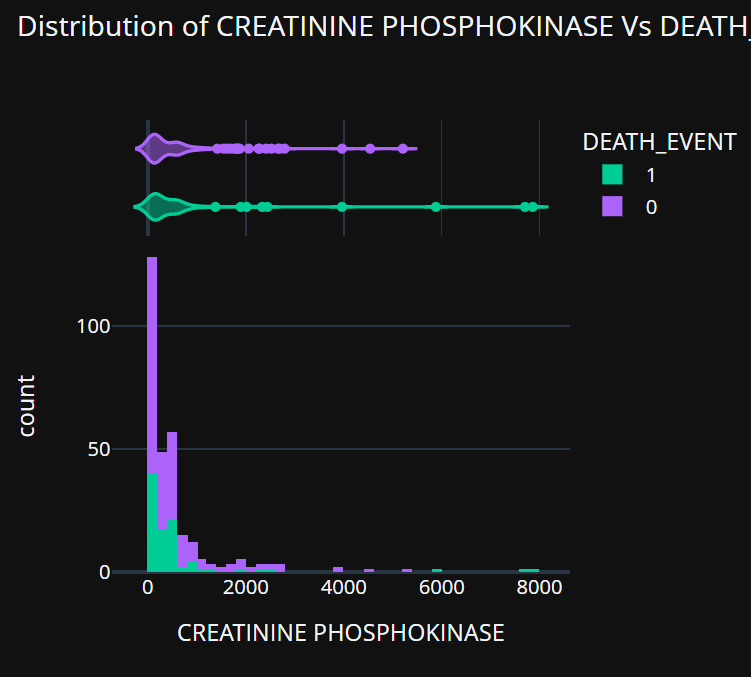
                   color\_discrete\_map={"0": "RebeccaPurple", "1": "MediumPurple"})

fig.show()

output

Continue doing this with all the category

**7.****Now its time to Splitting the dataset into training set and test set**



sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state =0)

Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.transform(x\_test)

**8.Lets start applying different algorithms and classifiers**

1. LOGISTIC REGRESSION

In [ ]:

*# Applying logistic regression on the training set*

**from** sklearn.linear\_model **import** LogisticRegression

classifier **=** LogisticRegression()

classifier**.**fit(x\_train, y\_train)

Out[ ]:

LogisticRegression()

In [ ]:

*# Predicting the test set*

y\_pred **=** classifier**.**predict(x\_test)

In [ ]:

*# Making Confusion Matrix and calculating accuracy score*

mylist **=** []

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

cm **=** confusion\_matrix(y\_test, y\_pred)

ac **=** accuracy\_score(y\_test, y\_pred)

mylist**.**append(ac)

print(cm)

print(ac)

[[40 3]

[ 4 13]]

0.8833333333333333

2.K NEAREST NEIGHBOR

In [ ]:

*# Finding the optimum number of neighbors*

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

list1 **=** []

**for** neighbors **in** range(3,10):

classifier **=** KNeighborsClassifier(n\_neighbors**=**neighbors, metric**=**'minkowski')

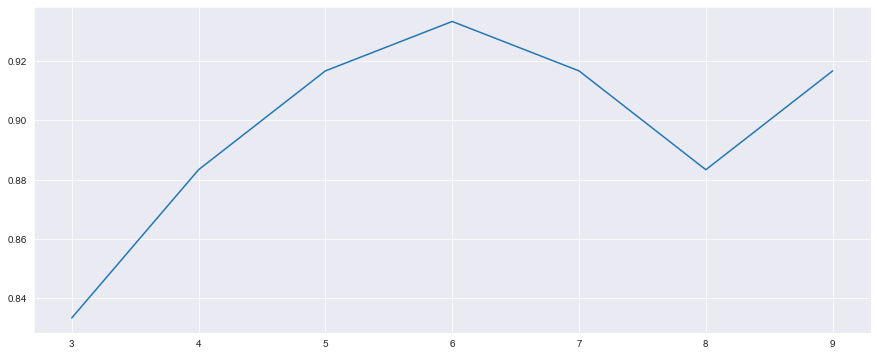
classifier**.**fit(x\_train, y\_train)

y\_pred **=** classifier**.**predict(x\_test)

list1**.**append(accuracy\_score(y\_test,y\_pred))

plt**.**plot(list(range(3,10)), list1)

plt**.**show()



In [ ]:

*# Training the K Nearest Neighbor Classifier on the Training set*

classifier **=** KNeighborsClassifier(n\_neighbors**=**6)

classifier**.**fit(x\_train, y\_train)

Out[ ]:

KNeighborsClassifier(n\_neighbors=6)

In [ ]:

*# Predicting the Test set results*

y\_pred **=** classifier**.**predict(x\_test)

print(y\_pred)

[0 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0]

In [ ]:

*# Making the confusion matrix and calculating accuracy score*

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

cm **=** confusion\_matrix(y\_test, y\_pred)

ac **=** accuracy\_score(y\_test, y\_pred)

mylist**.**append(ac)

print(cm)

print(ac)

[[42 1]

[ 3 14]]

0.9333333333333333

1. SUPPORT VECTOR CLASSIFIER

In [ ]:

**from** sklearn.svm **import** SVC

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

list1 **=** []

**for** c **in** [0.5,0.6,0.7,0.8,0.9,1.0]:

classifier **=** SVC(C **=** c, random\_state**=**0, kernel **=** 'rbf')

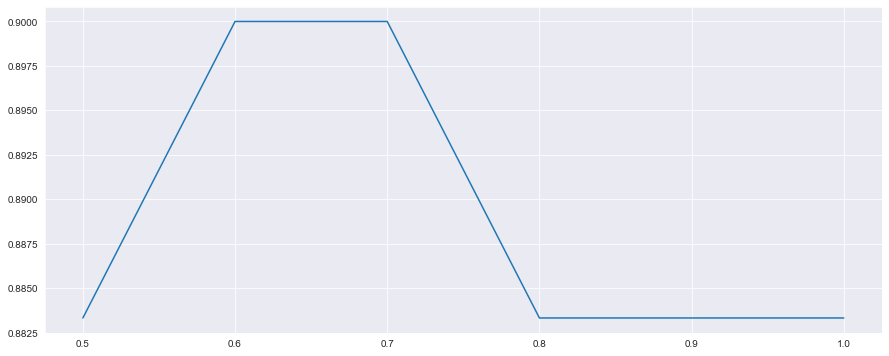
classifier**.**fit(x\_train, y\_train)

y\_pred **=** classifier**.**predict(x\_test)

list1**.**append(accuracy\_score(y\_test,y\_pred))

plt**.**plot([0.5,0.6,0.7,0.8,0.9,1.0], list1)

plt**.**show()



In [ ]:

*# Training the Support Vector Classifier on the Training set*

**from** sklearn.svm **import** SVC

classifier **=** SVC(C **=** 0.6, random\_state**=**0, kernel **=** 'rbf')

classifier**.**fit(x\_train, y\_train)

Out[ ]:

SVC(C=0.6, random\_state=0)

In [ ]:

*# Predicting the test set results*

y\_pred **=** classifier**.**predict(x\_test)

print(y\_pred)

[1 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1

0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 0]

In [ ]:

*# Making the confusion matrix and calculating accuracy score*

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

cm **=** confusion\_matrix(y\_test, y\_pred)

ac **=** accuracy\_score(y\_test, y\_pred)

print(cm)

print(ac)

mylist**.**append(ac)

[[40 3]

[ 3 14]]

0.9

1. DECISION TREE CLASSIFIER

In [ ]:

*# Finding the optimum number of max\_leaf\_nodes*

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

list1 **=** []

**for** leaves **in** range(2,10):

classifier **=** DecisionTreeClassifier(max\_leaf\_nodes **=** leaves, random\_state**=**0, criterion**=**'entropy')

classifier**.**fit(x\_train, y\_train)

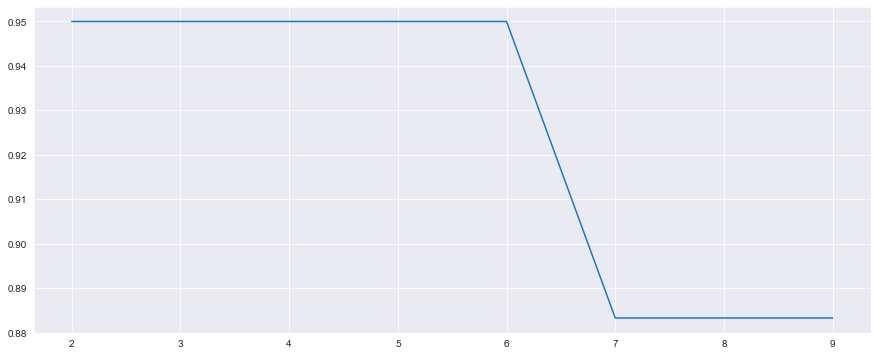
y\_pred **=** classifier**.**predict(x\_test)

list1**.**append(accuracy\_score(y\_test,y\_pred))

*#print(mylist)*

plt**.**plot(list(range(2,10)), list1)

plt**.**show()



In [ ]:

*# Training the Decision Tree Classifier on the Training set*

classifier **=** DecisionTreeClassifier(max\_leaf\_nodes **=** 3, random\_state**=**0, criterion**=**'entropy')

classifier**.**fit(x\_train, y\_train)

Out[ ]:

DecisionTreeClassifier(criterion='entropy', max\_leaf\_nodes=3, random\_state=0)

In [ ]:

*# Predicting the test set results*

y\_pred **=** classifier**.**predict(x\_test)

print(y\_pred)

[0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1

0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0]

In [ ]:

*# Making the confusion matrix and calculating accuracy score*

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

cm **=** confusion\_matrix(y\_test, y\_pred)

ac **=** accuracy\_score(y\_test, y\_pred)

print(cm)

print(ac)

mylist**.**append(ac)

[[43 0]

[ 3 14]]

0.95

1. RANDOM FOREST CLASSIFCATION

In [ ]:

*#Finding the optimum number of n\_estimators*

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

list1 **=** []

**for** estimators **in** range(10,30):

classifier **=** RandomForestClassifier(n\_estimators **=** estimators, random\_state**=**0, criterion**=**'entropy')

classifier**.**fit(x\_train, y\_train)

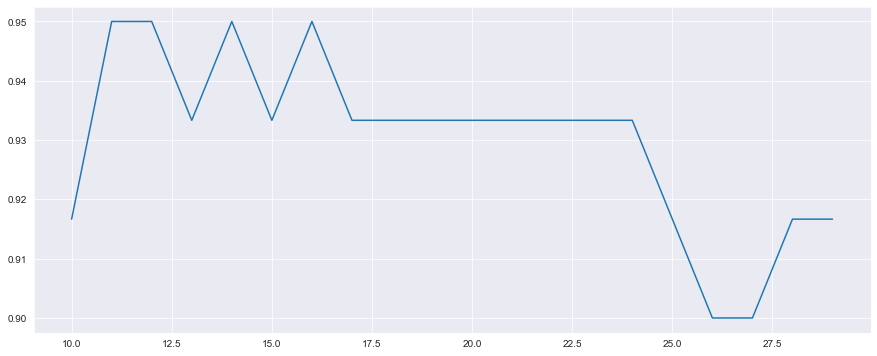
y\_pred **=** classifier**.**predict(x\_test)

list1**.**append(accuracy\_score(y\_test,y\_pred))

*#print(mylist)*

plt**.**plot(list(range(10,30)), list1)

plt**.**show()



In [ ]:

*# Training the RandomForest Classifier on the Training set*

**from** sklearn.ensemble **import** RandomForestClassifier

classifier **=** RandomForestClassifier(n\_estimators **=** 11, criterion**=**'entropy', random\_state**=**0)

classifier**.**fit(x\_train,y\_train)

Out[ ]:

RandomForestClassifier(criterion='entropy', n\_estimators=11, random\_state=0)

In [ ]:

*# Predicting the test set results*

y\_pred **=** classifier**.**predict(x\_test)

print(y\_pred)

[1 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1

0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0]

In [ ]:

*# Making the confusion matrix and calculating the accuracy score*

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

cm **=** confusion\_matrix(y\_test, y\_pred)

ac **=** accuracy\_score(y\_test, y\_pred)

mylist**.**append(ac)

print(cm)

print(ac)

[[41 2]

[ 1 16]]

0.95

1. ANN

In [ ]:

np**.**random**.**seed(0)

**import** tensorflow **as** tf

*# Initialising the ANN*

ann **=** tf**.**keras**.**models**.**Sequential()

In [ ]:

*# Adding the input layer and the first hidden layer*

ann**.**add(tf**.**keras**.**layers**.**Dense(units **=** 7, activation **=** 'relu'))

In [ ]:

*# Adding the second hidden layer*

ann**.**add(tf**.**keras**.**layers**.**Dense(units **=** 7, activation **=** 'relu'))

In [ ]:

*# Adding the third hidden layer*

ann**.**add(tf**.**keras**.**layers**.**Dense(units **=** 7, activation **=** 'relu'))

In [ ]:

*# Adding the fourth hidden layer*

ann**.**add(tf**.**keras**.**layers**.**Dense(units **=** 7, activation **=** 'relu'))

In [ ]:

*# Adding the output layer*

ann**.**add(tf**.**keras**.**layers**.**Dense(units **=** 1, activation **=** 'sigmoid'))

In [ ]:

*# Compiling the ANN*

ann**.**compile(optimizer **=** 'adam', loss **=** 'binary\_crossentropy' , metrics **=** ['accuracy'] )

In [ ]:

*# Training the ANN on the training set*

ann**.**fit(x\_train, y\_train, batch\_size **=** 32, epochs **=** 100)

**9.After applying all this on data set now lets make an app to genrlise for all data set.**

**Creating new file “HDp main app.py”**

from cgitb import text

from tkinter import \*

import tkinter as tk

top = Tk()

top.title("ALGO data ")

from tkinter import Menu

import os

global master

from tkinter.filedialog import askopenfilename

def openFile():

    global file

    file = askopenfilename(defaultextension=".txt",filetypes=[("All Files", "\*.\*"),("Text Documents", "\*.txt")])

    if file == "":

        file = None

    else:

        master.title(os.path.basename(file) + " - Notepad")

        text.delete(1.0, END)

        f = open(file, "r")

        text.insert(1.0, f.read())

        f.close()

def algo1():

    global text

    global master

    master = Tk()

    master.title("1. LOGISTIC REGRESSION")

    master.geometry("440x520")

    #master = Tk()

    master.geometry("300x400")

    labelframe1 = LabelFrame(master, text="Details")

    labelframe1.pack(fill="both", expand="yes")

    masterlabel = Label(labelframe1, text="Train test split of 80% /20%")

    masterlabel.pack()

    labelframe2 = LabelFrame(master, text = "Confusion Matrix")

    labelframe2.pack(fill="both", expand = "yes")

    bottomlabel = Label(labelframe2,text = "[[40  3]")

    bottomlabel1 = Label(labelframe2,text = "[ 4 13]]")

    bottomlabel.pack()

    bottomlabel1.pack()

    labelframe1 = LabelFrame(master, text="Accuracy\_Score")

    labelframe1.pack(fill="both", expand="yes")

    masterlabel = Label(labelframe1, text="0.8833333333333333")

    masterlabel.pack()

    master.mainloop()

    # type area

    master.mainloop()

img1 = PhotoImage(file='D:\project\_file\HDP\outputKNN.png')

def algo2():

    global text

    global master

    master = Tk()

    master.title("2. K NEAREST NEIGHBOR")

    master.geometry("440x520")

    #master = Tk()

    master.geometry("400x400")

    labelframe1 = LabelFrame(master, text="Details")

    labelframe1.pack(fill="both", expand="yes")

    masterlabel = Label(labelframe1, text="Number of neighbors used")

    masterlabel.pack()

    masterlabel = Label(labelframe1, text="KNeighborsClassifier(n\_neighbors=6)")

    masterlabel.pack()

    #img1 = PhotoImage(file='D:\project\_file\HDP\outputKNN.png')

    #masterlabel2 = Label(master, image=img1)

    #masterlabel2.pack()

    labelframe2 = LabelFrame(master, text = "Confusion Matrix")

    labelframe2.pack(fill="both", expand = "yes")

    bottomlabel = Label(labelframe2,text = "[[42  1]")

    bottomlabel1 = Label(labelframe2,text = "[ 3 14]]")

    bottomlabel.pack()

    bottomlabel1.pack()

    labelframe1 = LabelFrame(master, text="Accuracy\_Score")

    labelframe1.pack(fill="both", expand="yes")

    masterlabel = Label(labelframe1, text="0.9333333333333333")

    masterlabel.pack()

    master.mainloop()

def algo3():

    global text

    global master

    master = Tk()

    master.title("3. SUPPORT VECTOR CLASSIFIER")

    master.geometry("440x520")

    #master = Tk()

    master.geometry("300x400")

    labelframe1 = LabelFrame(master, text="Details")

    labelframe1.pack(fill="both", expand="yes")

    masterlabel = Label(labelframe1, text="Number of SV used")

    masterlabel.pack()

    masterlabel = Label(labelframe1, text="SVC(C=0.6, random\_state=0)")

    masterlabel.pack()

    #img = PhotoImage(file='D:\project\_file\HDP\outputKNN.png')

    #masterlabel = Label(master, image=img)

    #masterlabel.pack()

    labelframe2 = LabelFrame(master, text = "Confusion Matrix")

    labelframe2.pack(fill="both", expand = "yes")

    bottomlabel = Label(labelframe2,text = "[[40  3]")

    bottomlabel1 = Label(labelframe2,text = "[ 3 14]]")

    bottomlabel.pack()

    bottomlabel1.pack()

    labelframe1 = LabelFrame(master, text="Accuracy\_Score")

    labelframe1.pack(fill="both", expand="yes")

    masterlabel = Label(labelframe1, text="0.9")

    masterlabel.pack()

    master.mainloop()

def algo4():

    global text

    global master

    master = Tk()

    master.title("4. DECISION TREE CLASSIFIER")

    master.geometry("440x520")

    #master = Tk()

    master.geometry("300x400")

    labelframe1 = LabelFrame(master, text="Details")

    labelframe1.pack(fill="both", expand="yes")

    masterlabel = Label(labelframe1, text="Number of leaf node used")

    masterlabel.pack()

    masterlabel = Label(labelframe1, text="DecisionTreeClassifier(criterion='entropy', max\_leaf\_nodes=3, random\_state=0)")

    masterlabel.pack()

    #img = PhotoImage(file='D:\project\_file\HDP\outputKNN.png')

    #masterlabel = Label(master, image=img)

    #masterlabel.pack()

    labelframe2 = LabelFrame(master, text = "Confusion Matrix")

    labelframe2.pack(fill="both", expand = "yes")

    bottomlabel = Label(labelframe2,text = "[[43  0]")

    bottomlabel1 = Label(labelframe2,text = "[ 3 14]]")

    bottomlabel.pack()

    bottomlabel1.pack()

    labelframe1 = LabelFrame(master, text="Accuracy\_Score")

    labelframe1.pack(fill="both", expand="yes")

    masterlabel = Label(labelframe1, text="0.95")

    masterlabel.pack()

    master.mainloop()

def algo5():

        global text

        global master

        master = Tk()

        master.title("Count..")

        master.geometry("840x220")

        #master = Tk()

        #master.geometry("300x400")

        labelframe1 = LabelFrame(master, text="Accuracy\_Score")

        labelframe1.pack(fill="both", expand="yes")

        masterlabel = Label(labelframe1, text="Logistic Regression [0.8833333333333333], KNearestNeighbours[0.9333333333333333],SupportVector[0.9]")

        masterlabel.pack()

        masterlabel = Label(labelframe1, text="DecisionTree[0.95],RandomForest[0.95],ANN[0.9166666666666666]")

        masterlabel.pack()

        master.mainloop()

#def algo6():

# ifle name part

filename = None

def UploadAction(event=None):

    filename = askopenfilename()

    # Cut path to the file off

    filename = filename.split('heart\_failure\_clinical\_records\_dataset.csv')[len(filename.split('heart\_failure\_clinical\_records\_dataset.csv'))-1]

    print('Selected:', filename)

    label1['text'] = filename

button1 = tk.Button(text='heart\_failure\_clinical\_records\_dataset.csv', command=UploadAction, bg='#C0C0C0', fg='black')

button1.pack(padx=2, pady=5)

label1 = tk.Label(text='Please choose a file')

label1.pack(padx=2, pady=2)

#////

#algo

menubutton = Menubutton(top, text = "ALGO", relief = FLAT)

menubutton.menu = Menu(menubutton)

menubutton["menu"]=menubutton.menu

menubutton.menu.add\_command(label = "1. LOGISTIC REGRESSION",command=algo1)

menubutton.menu.add\_command(label = "2. K NEAREST NEIGHBOR",command=algo2)

menubutton.menu.add\_command(label = "3. SUPPORT VECTOR CLASSIFIER",command=algo3)

menubutton.menu.add\_command(label = "4. DECISION TREE CLASSIFIER",command=algo4)

menubutton.menu.add\_command(label = "5. RANDOM FOREST CLASSIFCATION")

menubutton.menu.add\_command(label = "6. ANN")

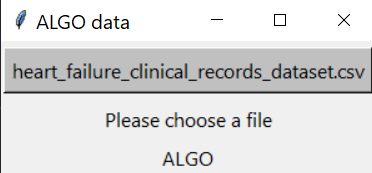
menubutton.menu.add\_command(label = "COMPARE ALL",command=algo5)

menubutton.pack()

#/////

top.mainloop()

**10.Final output/result (After we run this code)**

****HOME page

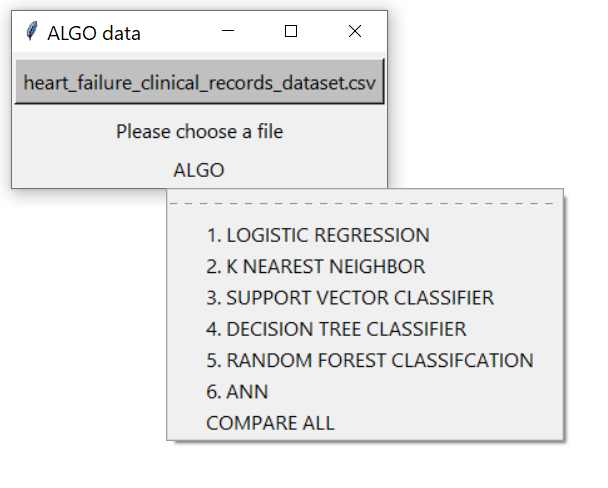
Button for FILE selecting

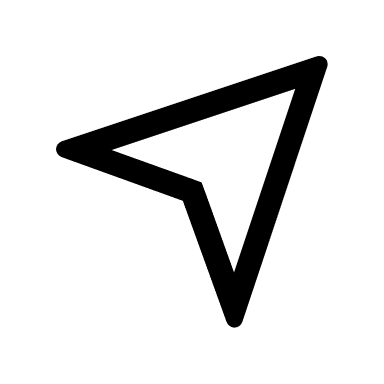
Menu button for choosing algo

Information box

Next step

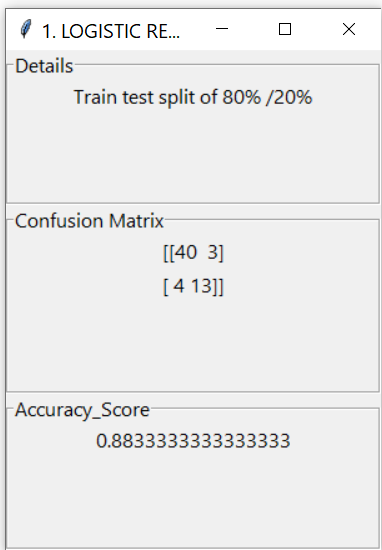
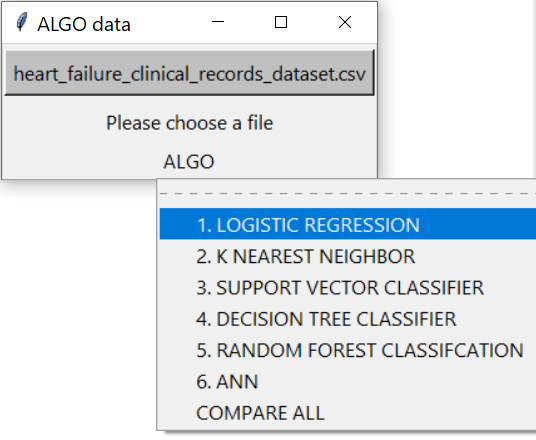
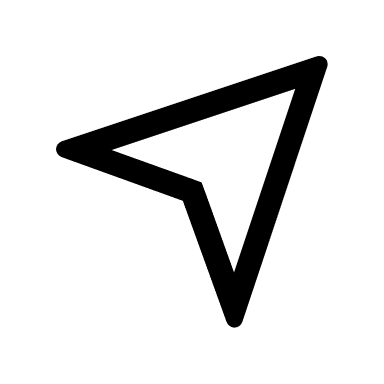
Click on algo





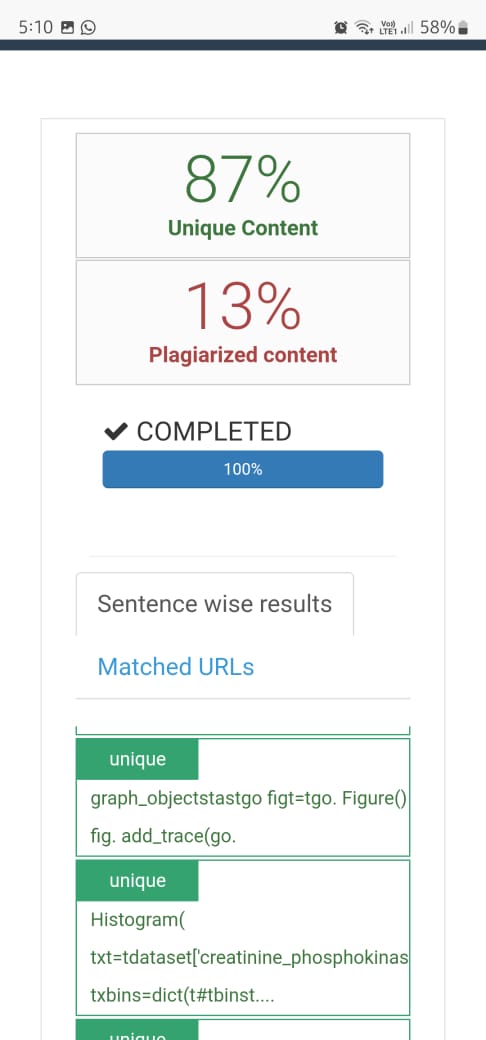
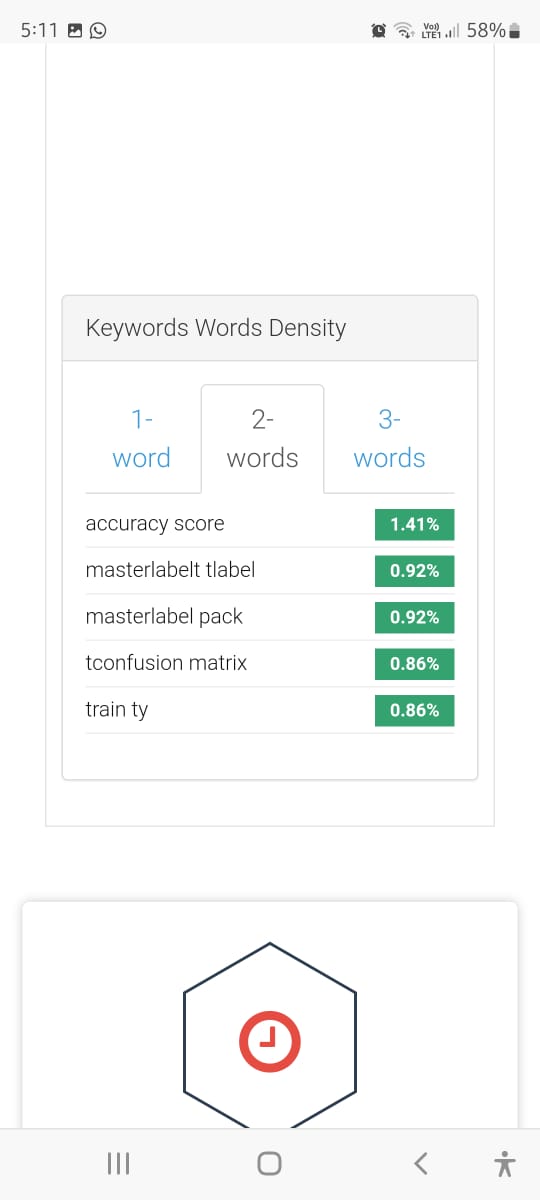
Next step

Choose your algorithm to run on data set



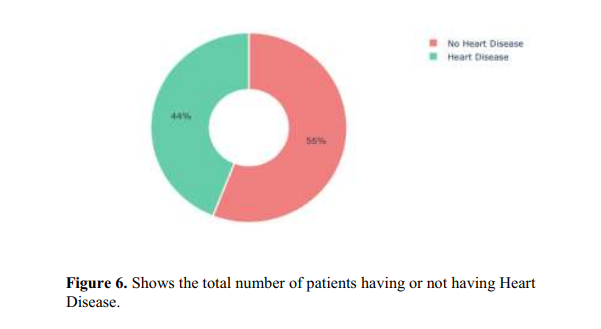


**Plagiarism test result**

****

# **6. CONCLUSION**

A cardiovascular disease detection model has been developed using three ML classification modeling techniques. This project predicts people with cardiovascular disease by extracting the patient's medical history that leads to a fatal heart disease from a dataset that includes patients’ medical history such as chest pain, sugar level, blood pressure, etc. This Heart Disease detection system assists a patient based on his/her clinical information of them being diagnosed with a previous heart disease. The algorithms used in building the given model are Logistic regression, Random Forest Classifier and KNN . The accuracy of our model is 87.5%. Use of more training data ensures the higher chances of the model to accurately predict whether the given person has a heart disease or not . By using these computer aided techniques we can predict the patient fast and better and the cost can be reduced very much. There are a number of medical databases that we can work on as these Machine learning techniques are better and they can predict better than a human being which helps the patient as well as the doctors. Therefore, in conclusion this project helps us predict the patients who are diagnosed with heart diseases by cleaning the dataset and applying logistic regression and KNN to get an accuracy of an average of 87.5% on our model which is better than the previous models having an accuracy of 85%. Also, it is concluded that accuracy of KNN is highest between the three algorithms that we have used i.e. 88.52%. ‘Figure 6’ shows 44% of people that are listed in the dataset are suffering from Heart Disease.



**7. Reference: ( Minimum 1 Page)**

1. <https://ieeexplore.ieee.org/abstract/document/8474922/>

2. <https://ieeexplore.ieee.org/document/8741465>

3. <http://sersc.org/journals/index.php/IJAST/article/download/5545/3446/>

4.<https://www.researchgate.net/publication/327722009_A_Review_on_Heart_Disease_Prediction_using_Machine_Learning_and_Data_Analytics_Approach>

10 [Decision Tree - GeeksforGeeksdecision-tree/?msclkid=b58b685bc08e11eca1d4289ca07eab76](https://www.geeksforgeeks.org/decision-tree/?msclkid=b58b685bc08e11eca1d4289ca07eab76).10 [Decision Tree - GeeksforGeeksdecision-tree/?msclkid=b58b685bc08e11eca1d4289ca07eab76](https://www.geeksforgeeks.org/decision-tree/?msclkid=b58b685bc08e11eca1d4289ca07eab76).1

9