In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data=pd.read_csv(r"C:\Users\ASUS\Downloads\archive (7)\Heart Attack.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
0	64	1	66	160	83	160.0	1.80	0.012	negative
1	21	1	94	98	46	296.0	6.75	1.060	positive
2	55	1	64	160	77	270.0	1.99	0.003	negative
3	64	1	70	120	55	270.0	13.87	0.122	positive
4	55	1	64	112	65	300.0	1.08	0.003	negative

In [4]:
```
data.tail()
```

Out[4]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
1314	44	1	94	122	67	204.0	1.63	0.006	negative
1315	66	1	84	125	55	149.0	1.33	0.172	positive
1316	45	1	85	168	104	96.0	1.24	4.250	positive
1317	54	1	58	117	68	443.0	5.80	0.359	positive
1318	51	1	94	157	79	134.0	50.89	1.770	positive

In [5]:
```
data.sample(15)
```

Out[5]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
86	40	1	76	157	93	193.0	4.66	0.003	negative
993	75	0	75	134	85	201.0	1.24	0.007	negative
231	50	1	75	142	75	122.0	6.27	0.004	negative
925	67	1	58	93	78	108.0	3.13	0.009	negative
1244	37	1	88	119	66	118.0	5.78	0.006	negative
61	90	0	58	120	69	191.0	5.22	0.015	positive
447	50	1	73	135	79	238.0	1.87	0.005	negative
761	55	0	68	116	74	143.0	1.34	0.094	positive
1142	63	1	94	105	81	168.0	1.58	0.005	negative
627	45	1	96	97	55	144.0	2.87	1.480	positive
93	60	0	60	130	56	294.0	2.13	0.103	positive
560	60	1	60	179	83	347.0	1.25	0.043	positive
71	58	0	91	120	80	177.0	18.15	0.005	positive
1067	52	0	58	120	69	97.0	5.17	0.083	positive
7	63	1	60	214	82	87.0	300.00	2.370	positive

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1319 entries, 0 to 1318
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   age            1319 non-null   int64  
 1   gender         1319 non-null   int64  
 2   impluse        1319 non-null   int64  
 3   pressurehight  1319 non-null   int64  
 4   pressurelow    1319 non-null   int64  
 5   glucose        1319 non-null   float64
 6   kcm            1319 non-null   float64
 7   troponin       1319 non-null   float64
 8   class          1319 non-null   object 
dtypes: float64(3), int64(5), object(1)
memory usage: 92.9+ KB
```

In [7]:
```
data.describe()
```

Out[7]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin
count	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000	1319.000000
mean	56.191812	0.659591	78.336619	127.170584	72.269143	146.634344	15.274306	0.360942
std	13.647315	0.474027	51.630270	26.122720	14.033924	74.923045	46.327083	1.154568
min	14.000000	0.000000	20.000000	42.000000	38.000000	35.000000	0.321000	0.001000
25%	47.000000	0.000000	64.000000	110.000000	62.000000	98.000000	1.655000	0.006000
50%	58.000000	1.000000	74.000000	124.000000	72.000000	116.000000	2.850000	0.014000
75%	65.000000	1.000000	85.000000	143.000000	81.000000	169.500000	5.805000	0.085500
max	103.000000	1.000000	1111.000000	223.000000	154.000000	541.000000	300.000000	10.300000

In [8]:
```
data.shape
```

Out[8]:
```
(1319, 9)
```

In [9]:
```
data.size
```

Out[9]:
```
11871
```

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

Out[10]:
```
age             0
gender          0
impluse         0
pressurehight   0
pressurelow     0
glucose         0
kcm             0
troponin        0
class           0
dtype: int64
```

In [11]:
```
data.age.value_counts()
```

Out[11]:
```
age
60     106
70      73
50      68
63      64
65      62
      ...
88       1
100      1
14       1
91       1
84       1
Name: count, Length: 75, dtype: int64
```

In [12]:
```
data['class'].value_counts()
```

Out[12]:
```
class
positive    810
negative    509
Name: count, dtype: int64
```

In [13]:
```
negative = data[data['class'] == "negative"]
positive = data[data['class'] == "positive"]
```

In [14]:
```
negative.shape
```

Out[14]:
```
(509, 9)
```

In [15]:
```
positive.shape
```

Out[15]:
```
(810, 9)
```

In [16]:
```
positive_sample = positive.sample(n=509)
positive_sample.shape
```

Out[16]:
```
(509, 9)
```

In [17]:
```
newdata = pd.concat([positive_sample , negative], axis=0)
```

In [18]:
```
newdata.sample(15)
```

Out[18]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
596	82	0	88	152	87	99.0	1.210	0.004	negative
1037	55	1	74	150	90	117.0	7.610	0.104	positive
860	54	1	72	154	84	127.0	2.970	0.007	negative
617	60	1	80	135	75	94.0	147.400	3.850	positive
223	63	1	119	170	107	129.0	2.610	0.005	negative
602	45	0	80	117	83	143.0	2.490	0.003	negative
760	69	0	73	135	81	69.0	4.950	0.007	negative
419	45	1	90	110	65	83.0	2.420	0.096	positive
254	67	0	69	128	70	382.0	2.330	0.007	negative
1021	61	0	93	120	71	121.0	79.620	0.007	positive
888	71	1	59	107	64	97.0	1.970	1.450	positive
983	50	0	81	124	75	114.0	0.321	0.003	negative
363	55	1	60	130	72	96.0	2.800	0.117	positive
1036	72	0	75	160	70	130.0	8.540	0.015	positive
208	49	0	67	120	55	100.0	0.676	0.005	positive

In [19]:
```
newdata['class'].value_counts()
```

Out[19]:
```
class
positive    509
negative    509
Name: count, dtype: int64
```

In [20]:
```
newdata.shape
```

Out[20]:
```
(1018, 9)
```

In [21]:
```
X=newdata.drop("class" , axis="columns")
X.head()
```

Out[21]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin
19	45	1	60	109	65	89.0	1.60	0.020
23	30	1	63	110	68	107.0	50.46	0.003
189	78	1	62	157	66	106.0	3.77	0.024
149	69	1	82	86	70	87.0	4.37	0.017
571	42	0	73	162	99	109.0	15.83	0.100

In [22]:
```
X.shape
```

Out[22]:
```
(1018, 8)
```

In [23]:
```
Y=newdata['class'].values
```

In [24]:
```
Y
```

Out[24]:
```
array(['positive', 'positive', 'positive', ..., 'negative', 'negative',
       'negative'], dtype=object)
```

In [25]:
```
Y.shape
```

Out[25]:
```
(1018,)
```

In [26]:
```
print(Y)
```

```
['positive' 'positive' 'positive' ... 'negative' 'negative' 'negative']
```

In [27]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
Y=le.fit_transform(Y)
```

In [28]:
```
Y
```

Out[28]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [29]:
```
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X , Y , test_size= 0.2)
```

In [30]:
```
len(X_train)
```

Out[30]: 814

In [31]:
```
len(X_test)
```

Out[31]: 204

In [32]:
```
len(y_train)
```

Out[32]: 814

In [33]:
```
len(y_test)
```

Out[33]: 204

In [34]:
```
X_train.shape
```

Out[34]: (814, 8)

In [35]:
```
y_train.shape
```

Out[35]: (814,)

In [36]:
```
X_train.head()
```

Out[36]:

	age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin
380	29	1	90	129	90	135.0	8.93	0.003
1093	29	1	76	157	93	242.0	4.79	0.004
449	50	1	89	162	99	100.0	1.83	0.005
1147	63	1	64	122	60	188.0	2.19	0.046
874	48	0	64	140	90	168.0	3.53	0.026

In [37]:
```
y_train
```

Out[37]:
```
array([1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0])
```

In [38]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [39]:
```
lrmodel = LogisticRegression()
lrmodel.fit(X_train , y_train)
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceW
arning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[39]:
```
▾ LogisticRegression
LogisticRegression()
```

In [40]:
```
lrmodel.predict(X_test)
```

Out[40]:
```
array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0])
```

In [41]:
```
from sklearn.metrics import accuracy_score
X_train_prediction = lrmodel.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction , y_train)
```

In [42]:
```
training_data_accuracy
```

Out[42]:
```
0.8366093366093366
```

In [43]:
```
from sklearn.metrics import accuracy_score
X_test_prediction = lrmodel.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction , y_test)
```

In [44]:
```
testing_data_accuracy
```

Out[44]:
```
0.8088235294117647
```

In [45]:
```
dtmodel = DecisionTreeClassifier(random_state=42)
dtmodel.fit(X_train, y_train)
```

Out[45]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

In [46]:
```
from sklearn.metrics import accuracy_score
X_train_prediction = dtmodel.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction , y_train)
```

In [47]:
```
training_data_accuracy
```

Out[47]:
```
1.0
```

In [48]:
```
from sklearn.metrics import accuracy_score
X_test_prediction = dtmodel.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction , y_test)
```

In [49]:
```
testing_data_accuracy
```

Out[49]:
```
0.9803921568627451
```

In [50]:
```
'''
Age
gender(0 for Female, 1 for Male)
heart rate (impulse)
systolic BP (pressurehight)
diastolic BP (pressurelow)
blood sugar(glucose)
CK-MB (kcm)
Test-Troponin (troponin)
negative refers to the absence of a heart attack, while positive refers to the presence of a heart attack
```

```
'''
```

Out[50]:
```
'\nAge\ngender(0 for Female, 1 for Male)\nheart rate (impulse)\nsystolic BP (pressurehight) \ndiastolic BP (pre
ssurelow)\nblood sugar(glucose) \nCK-MB (kcm)\nTest-Troponin (troponin) \nnegative refers to the absence of a h
eart attack, while positive refers to the presence of a heart attack\n\n'
```

In [51]:
```
dtmodel.predict([[64,1,66,160,83,160.0,1.80,0.012]])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Out[51]:
```
array([0])
```

In [52]:
```
input_data = (64,1,66,160,83,160.0,1.80,0.012)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = dtmodel.predict(input_data_reshaped)
prediction = le.inverse_transform(prediction)
prediction
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Out[52]:
```
array(['negative'], dtype=object)
```

In [53]:
```
input_data = (64,1,66,160,83,160.0,1.80,0.012)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = dtmodel.predict(input_data_reshaped)
prediction = le.inverse_transform(prediction)
if prediction == "negative":
   print("The person does not have Heart attack problem.")
else:
   print("The person has Heart attack problem.")
```

```
The person does not have Heart attack problem.
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

In [54]:
```
import pickle
with open("Heart_attack_model_pickle" , "wb") as file:
    pickle.dump(dtmodel , file)
```

In [55]:
```
import numpy as np
import pickle
with open("Heart_attack_model_pickle" , "rb") as file:
    dtmodel = pickle.load(file)
```

In [56]:
```
input_data = (64,1,66,160,83,160.0,1.80,0.012)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = dtmodel.predict(input_data_reshaped)
prediction = le.inverse_transform(prediction)
if prediction == "negative":
   print("The person does not have Heart attack problem.")
else:
   print("The person has Heart attack problem.")
```

```
The person does not have Heart attack problem.
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

In [58]:
```
#age in years
age = int(input(""))
#gender(0 for Female, 1 for Male)
gender = int(input(""))
#impluse
impluse = int(input(""))
#pressurehight
pressurehight = int(input(""))
#pressurelow
pressurelow = int(input(""))
#glucose
glucose  = float(input(""))
#kcm
kcm = float(input(""))
#troponin
troponin = float(input(""))
```

In [59]:
```
input_data = (age,gender,impluse,pressurehight,pressurelow,glucose,kcm,troponin)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = dtmodel.predict(input_data_reshaped)
prediction = le.inverse_transform(prediction)
if prediction == "negative":
   print("The person does not have Heart attack problem.")
else:
   print("The person has Heart attack problem.")
```

```
The person does not have Heart attack problem.
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

In [ ]: