HEART DISEASE PREDICTION USING ML

1.IMPORT THE DEPENDENCIES

In [5]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

data processing

In [7]:
```
heart_data=pd.read_csv("C://Users//91903//Downloads//heart_disease_data.csv")
```

In [8]:
```
heart_data
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	'
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	'
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	'
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	'
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	'
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	(
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	(
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	(
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	(
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	(

303 rows × 14 columns

In [9]:
```
heart_data.head()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [10]:
```
heart_data.tail()
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	(
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	(
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	(
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	(
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	(

In [12]:
```
heart_data.shape
```

Out[12]:
```
(303, 14)
```

In [13]:
```
heart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [17]:

```
#checking for missing values
heart_data.isnull().sum()
```

Out[17]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [18]:

```
#statistical measures of the data
heart_data.describe()
```

Out[18]:

ge	sex	cp	trestbps	chol	fbs	restecg	thalach	exar
00	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
37	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.32673
01	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.46979
00	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.00000
00	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.00000
00	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.00000
00	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.00000
00	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.00000

In [21]:

```
#checking the distribution of target variable
heart_data['target'].value_counts()
```

Out[21]:

```
1    165
0    138
Name: target, dtype: int64
```

1-->defective heart

0-->healthy heart

In [23]:
```
#splitting the features and target
X=heart_data.drop(columns="target",axis=1)
Y=heart_data['target']
```

In [27]:
```
X
```

Out[27]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

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

Out[28]:
```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

Splitting the data into training data and test data

In [31]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,ra
```

In [32]:
```
print(X.shape,X_train.shape,X_test.shape)
```

```
(303, 13) (242, 13) (61, 13)
```

Model training(LOGISTIC REGRESSION)

In [33]:
```
model=LogisticRegression()
```

In [35]:
```
#train the ML model with training data
model.fit(X_train,Y_train)
```

```
C:\Users\91903\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: ConvergenceWarning: 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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[35]:
```
LogisticRegression()
```

model evaluation accuracy score

In [39]:
```
#accuracy on training data
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
```

In [41]:
```
print("Accuracy on training data:",training_data_accuracy)
```

```
Accuracy on training data: 0.8512396694214877
```

In [42]:
```
#accuracy on test data
X_test_prediction=model.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
```

In [43]:
```
print("Accuracy on test data",test_data_accuracy)
```

```
Accuracy on test data 0.819672131147541
```

BUILDING PREDICTIVE SYSTEM

In [44]:
```
input_data=(41,0,1,130,204,0,0,172,0,1.4,2,0,2)
```

In [60]:

```
#change input_data to numpy array
input_data_as_numpy_array=np.asarray(input_data)
#reshape the numpy array as we are predicting for only one instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
prediction=model.predict(input_data_reshaped)
print(prediction)
if (prediction[0]==0):
    print("the person does not have heart disease")
else:
    print("The person has heart disease")
```

```
[1]
The person has heart disease

C:\Users\91903\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning:
X does not have valid feature names, but LogisticRegression was fitted with f
eature names
  warnings.warn(
```

In [ ]:

```

```