In [19]:
```python
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
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
heart_data = pd.read_csv("heart_disease_data.csv")
```

In [3]:
```python
heart_data.head(5)
```

Out[3]:

	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 [4]:
```python
heart_data.tail(5)
```

Out[4]:

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

In [5]:
```python
heart_data.shape
```

Out[5]:
```
(303, 14)
```

In [6]:
```python
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 [7]:

```
heart_data.describe()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000

In [8]:

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

Out[8]:

```
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 [9]:
```
heart_data.isnull().sum().sum()
```

Out[9]:
```
0
```

In [10]:
```
# Checking the distribution of Target Variable
heart_data['target'].value_counts() # 1 --> Defective heart ,0 -->Healthy heart
```

Out[10]:
```
1    165
0    138
Name: target, dtype: int64
```

In [11]:
```
X = heart_data.drop(columns="target",axis=1)
Y = heart_data['target']
```

In [12]:
```
X
```

Out[12]:

	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 [13]:
```
Y
```

Out[13]:
```
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
```

In [14]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,stratify=Y,random_s
```

In [15]:
```
X.shape,X_train.shape,X_test.shape
```

```
Out[15]: ((303, 13), (242, 13), (61, 13))
```

Model Training

Logistic Regression

```
In [16]: model = LogisticRegression()
```

```
In [20]: model.fit(X_train,Y_train)
```

```
Out[20]: LogisticRegression()
```

Model Evalution

Accuracy Score

```
In [24]: X_train_prediction = model.predict(X_train)
         training_data_accuracy = accuracy_score(X_train_prediction,Y_train)
```

```
In [25]: print ("accuracy_score :" , training_data_accuracy)
```

```
accuracy_score : 0.8512396694214877
```

```
In [28]: # accuracy on test data
         X_test_prediction = model.predict(X_test)
         test_data_accuracy = accuracy_score(X_test_prediction,Y_test)
```

```
In [29]: print('Accuracy on test data :', test_data_accuracy)
```

```
Accuracy on test data : 0.819672131147541
```

Building a predictive System

```
In [30]: input_data = (41,0,1,130,204,0,0,172,0,1.4,2,0,2)
```

```
In [32]: # change the input data to a  numpy array

         input_data_as_numpy_array = np.asarray(input_data)

         # reshape the numpy array as we are predicting for only on 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 a Heart Disesse')
         else:
             print("The person has Heart Disease")
```

```
[1]
The person has Heart Disease
```

```
In [ ]:
```