In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [2]:
```
df=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\heart_disease_data.csv")
df.head()
```

Out[2]:

	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 [3]:
```
#number of rows and columns
df.shape
```

Out[3]:
```
(303, 14)
```

In [4]:
```
#describe mathamatical data
df.describe()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.3
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.6
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.0
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.0
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.0
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.0

In [5]:
```
#information about data
df.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 [6]:
```
#to find missing value in data
df.isnull().sum()
```

Out[6]:
```
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 [7]:
```
# checking the distribution of target variable
df['target'].value_counts()
```

Out[7]:
```
1    165
0    138
Name: target, dtype: int64
```

In [22]:
```
#0-->Healthy Heart   1--> Defective Heart
df.groupby('target').mean()
```

Out[22]:

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.550725	1.585507	1.166667	1.166667	2.543478
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.139394	0.583030	1.593939	0.363636	2.121212

In [23]:
```
#separting data and labels
X=df.drop(columns='target',axis=1)
y=df['target']
print(X)
print(y)
```

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

     slope  ca  thal  
0        0   0     1  
1        0   0     2  
2        2   0     2  
3        2   0     2  
4        2   0     2  
..     ...  ..   ...  
298      1   0     3  
299      1   0     3  
300      1   2     3  
301      1   1     3  
302      1   1     2  

[303 rows x 13 columns]
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 [11]:
```
#training and test data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2,stratify=y)
print("shape of X_train= ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train=  (242, 13)
shape of X_test=  (61, 13)
shape of y_train=  (242,)
shape of y_test=  (61,)
```

In [13]:
```
#Model training
model=LogisticRegression()
model
```

Out[13]:
```
LogisticRegression()
```

In [14]:
```
model.fit(X_train,y_train)
```

```
C:\Users\Pranav\Searches\hjhkh\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to co
nverge (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[14]:
```
LogisticRegression()
```

In [16]:
```
#model evalution
#Accuracy score
#accuracy of training data
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,y_train)
print('Accuracy on training data:',training_data_accuracy)
```

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

In [17]:
```
#Accuracy score
#accuracy of testing data
X_test_prediction=model.predict(X_test)
testing_data_accuracy=accuracy_score(X_test_prediction,y_test)
print('Accuracy on testing data:',testing_data_accuracy)
```

```
Accuracy on testing data: 0.819672131147541
```

In [18]:
```
#Making a Predictive system
#changing the input _data to a numpy array
input_data1=(57,1,0,140,192,0,1,148,0,0.4,1,0,1)
input_data_numpy_array1=np.asarray(input_data1)
```

In [20]:
```
#reshape the np array as we are predicting for one instance
input_data_reshaped1=input_data_numpy_array1.reshape(1,-1)
print(input_data_reshaped1)
```

```
[[ 57.    1.    0.  140.  192.    0.    1.  148.    0.    0.4   1.    0.
    1. ]]
```

In [21]:
```
prediction=model.predict(input_data_reshaped1)
print(prediction)
```

```
[1]
C:\Users\Pranav\Searches\hjhkh\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but Lo
gisticRegression was fitted with feature names
  warnings.warn(
```

In [24]:
```
if(prediction[0]==0):
    print('The person does not hAVE a Heart Diseaes')
else:
    print('The person has Heart Diseaes ')
```

```
The person has Heart Diseaes
```

In [29]:
```
input_data2=(51,1,0,140,298,0,1,122,1,4.2,1,3,3)
input_data_numpy_array2=np.asarray(input_data2)
```

In [30]:
```
#reshape the np array as we are predicting for one instance
input_data_reshaped2=input_data_numpy_array2.reshape(1,-1)
input_data_reshaped2
```

Out[30]:
```
array([[ 51. ,   1. ,   0. , 140. , 298. ,   0. ,   1. , 122. ,   1. ,
          4.2,   1. ,   3. ,   3. ]])
```

In [31]:
```
prediction=model.predict(input_data_reshaped2)
print(prediction)
```

```
[0]
C:\Users\Pranav\Searches\hjhkh\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but Lo
gisticRegression was fitted with feature names
  warnings.warn(
```

In [32]:
```
if(prediction[0]==0):
    print('The person does not hAVE a Heart Diseaes')
else:
    print('The person has Heart Diseaes ')
```

```
The person does not hAVE a Heart Diseaes
```

In [ ]: