In [2]:
```
#Importing the libaries
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
```

In [3]:
```
"""Data collection and processing"""
```

In [4]:
```
heart_data=pd.read_csv('heart_disease_data.csv')
```

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

Out[5]:

	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 [6]:
```
heart_data.shape
```

Out[6]:
```
(303, 14)
```

In [7]:
```
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 [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]:
```
#Statistical measures about the data
heart_data.describe()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	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
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	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
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4

In [10]:
```
#checking the distribution of target variable
heart_data['target'].value_counts()
#1 means having heart disease
#0 means doesn't have heart disease
```

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

In [14]:
```
#splitting of data
x=heart_data.iloc[:,:-1]
y=heart_data.iloc[:,-1]
```

In [17]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,stratify=y,random_state=2)
```

In [19]:
```
print(x.shape,x_train.shape,x_test.shape)
```

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

In [21]:
```
#model training
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\Users\vinayak\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs f
ailed 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[21]:
```
LogisticRegression()
```

In [22]:
```
# Model evaluation
#accuracy on training data
x_train_prediction=lr.predict(x_train)
training_data_accuracy=accuracy_score(x_train_prediction,y_train)
print(training_data_accuracy)
```

```
0.8512396694214877
```

In [23]:
```
x_test_prediction=lr.predict(x_test)
test_data_accuracy=accuracy_score(x_test_prediction,y_test)
print(test_data_accuracy)
```

```
0.819672131147541
```

In [35]:
```
# Building predictive system
input_data=(41,0,1,130,204,0,0,172,0,1,4,2,0)

#change the i/p data to numpy array
input_data_np=np.asarray(input_data)

#reshaping the numpy array as we are predicting for only instances
input_data_reshape=input_data_np.reshape(1,-1)

prediction=lr.predict(input_data_reshape)


if (prediction[0]==0):
    print("The person doesn't have heart disease")
else:
    print("The person has heart disease")
```

```
The person has heart disease
```

In [ ]:

In [ ]: