In [1]:

```
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 Collection and Pre Processing

In [2]:

```
# Loading the CSV Data to a Pandas DataFrame
heart_data = pd.read_csv('B:\VSCODE\Multiple Disease Prediction System\Heart Detection System\heart.csv')
heart_data.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]:

```
heart_data.shape
```

Out[3]:

```
(303, 14)
```

In [4]:

```
heart_data.describe()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpe
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.0396
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.1610
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.0000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.0000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.8000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.6000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.2000

In [5]:

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

Out[5]:

```
age         0
sex         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 [6]:

```
# Checking the distribution of Target Variable
heart_data['target'].value_counts()
```

Out[6]:

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

1 -> Defective Heart

0 -> Healthy Heart

Splitting the Features and Target

In [7]:

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

In [8]:

```
# Splitting the data into Training Data & Testing Data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, ran
dom_state=2)
```

In [9]:

```
print(X.shape, X_train.shape, X_test.shape)
```

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

Model Training

Logistic Regression

In [12]:

```
# Model Training
model = LogisticRegression()
```

In [13]:

```
# Training the Logistic Regression Model with Training Data
model.fit(X_train, Y_train)
```

```
C:\Users\SATYAM\AppData\Roaming\Python\Python39\site-packages\sklearn\linear_model\_logis
tic.py:458: 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
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[13]:

▾ LogisticRegression

LogisticRegression()

Model Evaluation

Accuracy Score

In [15]:

```
# Accuracy on Training Data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [16]:

```
print('Accuracy on Training Data : ', training_data_accuracy)
```

```
Accuracy on Training Data :  0.8512396694214877
```

In [17]:

```
# Accuracy on Test Data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [18]:

```
print('Accuracy on Test Data : ', test_data_accuracy)
```

```
Accuracy on Test Data :  0.819672131147541
```

In [20]:

```
# Building a Predictive System
input_data = (54,1,0,140,239,0,1,160,0,1.2,2,0,2)

# Changing the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# Reshape the data as we are predicting the label 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 is not suffering from Heart Disease ')
else:
    print('The Person is suffering from Heart Disease')
```

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

```
C:\Users\SATYAM\AppData\Roaming\Python\Python39\site-packages\sklearn\base.py:439: UserWa
rning: X does not have valid feature names, but LogisticRegression was fitted with featur
e names
  warnings.warn(
```

In [ ]: