Importing Dependencies

In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [ ]:
```
```

Data Collection and Processing

In [2]:
```
data=pd.read_csv(r"C:\Users\Kishore\OneDrive\Desktop\CSV Files\heart.csv")
```

In [3]:
```
print(data)
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
\
0      52    1   0       125   212    0        1      168      0      1.0
1      53    1   0       140   203    1        0      155      1      3.1
2      70    1   0       145   174    0        1      125      1      2.6
3      61    1   0       148   203    0        1      161      0      0.0
4      62    0   0       138   294    1        1      106      0      1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
1020   59    1   1       140   221    0        1      164      1      0.0
1021   60    1   0       125   258    0        0      141      1      2.8
1022   47    1   0       110   275    0        0      118      1      1.0
1023   50    0   0       110   254    0        0      159      0      0.0
1024   54    1   0       120   188    0        1      113      0      1.4

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
...     ...  ..   ...     ...
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]
```

In [4]:
```
data
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3

1025 rows × 14 columns

In [5]:
```
data.head(10)
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	
6	58	1	0	114	318	0	2	140	0	4.4	0	3	1	
7	55	1	0	160	289	0	0	145	1	0.8	1	1	3	
8	46	1	0	120	249	0	0	144	0	0.8	2	0	3	
9	54	1	0	122	286	0	0	116	1	3.2	1	2	2	

In [6]:
```
data.tail(5)
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3

In [7]:
```
data.shape
```

Out[7]:
```
(1025, 14)
```

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
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 [10]:
```
data.describe()
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	re
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.0
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.5
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.5
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.0
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.0
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.0
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.0
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.0

In [11]:
```
data['target'].value_counts()
```

Out[11]:
```
1    526
0    499
Name: target, dtype: int64
```

In [12]:
```
#1--> Defective Heart
#0--> Healthy Heart

#according to the data set
```

In [13]:
```
X = data.drop(columns='target', axis=1)
Y = data['target']
```

In [14]:
```
print(X)
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
\
0      52    1   0       125   212    0        1      168      0      1.0
1      53    1   0       140   203    1        0      155      1      3.1
2      70    1   0       145   174    0        1      125      1      2.6
3      61    1   0       148   203    0        1      161      0      0.0
4      62    0   0       138   294    1        1      106      0      1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
1020   59    1   1       140   221    0        1      164      1      0.0
1021   60    1   0       125   258    0        0      141      1      2.8
1022   47    1   0       110   275    0        0      118      1      1.0
1023   50    0   0       110   254    0        0      159      0      0.0
1024   54    1   0       120   188    0        1      113      0      1.4

      slope  ca  thal
0         2   2     3
1         0   0     3
2         0   0     3
3         2   1     3
4         1   3     2
...     ...  ..   ...
1020      2   0     2
1021      1   1     3
1022      1   1     2
1023      2   0     2
1024      1   1     3

[1025 rows x 13 columns]
```

In [15]:
```
print(Y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1020    1
1021    0
1022    0
1023    1
1024    0
Name: target, Length: 1025, dtype: int64
```

In [16]:
```
##Splitting the Data into Training dat & testing Data
```

In [17]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, st
```

In [18]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(1025, 13) (820, 13) (205, 13)
```

In [19]:
```
## Model Training Using Logistic Regression
```

In [20]:
```
model = LogisticRegression()
```

In [21]:
```
# training the Logistic regression model with Training data
model.fit(X_train, Y_train)
```

Out[21]:
```
LogisticRegression
LogisticRegression()
```

In [22]:
```
# Model Evaluation

# Accuracy Score
```

In [23]:
```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [24]:
```
print('Accuracy on Training data : ',training_data_accuracy)
```

```
Accuracy on Training data :  0.848780487804878
```

In [25]:
```
X_test_prediction = model.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [26]:
```
print('Accuracy on Test data : ',testing_data_accuracy)
```

```
Accuracy on Test data :  0.8048780487804879
```

In [27]:
```
# Building Predictive system
```

In [28]:
```
input_data = (71,0,0,112,149,0,1,125,0,1.6,1,0,2)

# change input data to a numpy array by using reshaping

input_data_as_numpy_array = np.asarray(input_data)

# reshaping numpy array as we are predicting for one data point

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
```

In [ ]: