

Importing the Dependencies

```
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 Processing

```
In [7]: # loading the csv data to a Pandas DataFrame
heart_data = pd.read_csv('heart_disease_data.csv')
```

```
In [8]: # print first 5 rows of the dataset
heart_data.head()
```

```
Out[8]:
```

	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 [9]: # print last 5 rows of the dataset
heart_data.tail()
```

```
Out[9]:
```

	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 [10]: # number of rows and columns in the dataset
heart_data.shape
```

```
Out[10]: (303, 14)
```

```
In [11]: # getting some info about the data
heart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age          303 non-null int64
sex          303 non-null int64
cp           303 non-null int64
trestbps     303 non-null int64
chol         303 non-null int64
fbs          303 non-null int64
restecg      303 non-null int64
thalach      303 non-null int64
exang        303 non-null int64
oldpeak      303 non-null float64
slope        303 non-null int64
ca           303 non-null int64
thal         303 non-null int64
target       303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

```
In [12]: # checking for missing values
heart_data.isnull().sum()
```

```
Out[12]: 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 [13]: # statistical measures about the data
heart_data.describe()
```

```
Out[13]:
```

	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
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
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
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
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
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
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
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

```
In [32]: # checking the distribution of Target Variable
heart_data['target'].value_counts()
```

```
Out[32]: 1    165
0     138
Name: target, dtype: int64
```

1--> Defective Heart

0--> Healthy Heart

Splitting the Features and Target

```
In [33]: X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

```
In [16]: print(X)
```

```
   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
5    57   1   0    140    192   0         1    148      0      0.4
6    56   0   1    140    294   0         0    153      0      1.3
7    44   1   1    120    263   0         1    173      0      0.0
8    52   1   2    172    199   1         1    162      0      0.5
9    57   1   2    150    168   0         1    174      0      1.6
10   54   1   0    140    239   0         1    160      0      1.2
11   48   0   2    130    275   0         1    159      0      0.2
12   49   1   1    130    266   0         1    171      0      0.6
13   64   1   3    110    211   0         0    144      1      1.8
14   58   0   3    150    283   1         0    162      0      1.0
15   50   0   2    120    219   0         1    158      0      1.6
16   58   0   2    120    340   0         1    172      0      0.0
17   66   0   3    150    226   0         1    114      0      2.6
18   43   1   0    150    247   0         1    171      0      1.5
19   69   0   3    140    239   0         1    151      0      1.8
20   59   1   0    135    234   0         1    161      0      0.5
21   44   1   2    130    233   0         1    179      1      0.4
22   42   1   0    140    226   0         1    178      0      0.0
23   61   1   2    150    243   1         1    137      1      1.0
24   40   1   3    140    199   0         1    178      1      1.4
25   71   0   1    160    392   1         1    162      0      0.4
26   59   1   0    150    212   1         1    157      0      1.6
27   51   1   2    110    175   0         1    123      0      0.6
28   65   0   2    140    417   1         0    157      0      0.8
29   53   1   2    130    197   1         0    152      0      1.2
...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
273  58   1   0    100    234   0         1    156      0      0.1
274  47   1   0    110    275   0         0    118      1      1.0
275  52   1   0    125    212   0         1    168      0      1.0
276  58   1   0    145    218   0         1    105      0      2.0
277  57   1   1    124    261   0         1    141      0      0.3
278  58   0   1    136    319   1         0    152      0      0.0
279  61   1   0    138    166   0         0    125      1      3.6
280  42   1   0    136    315   0         1    125      1      1.8
281  52   1   0    128    204   1         1    156      1      1.0
282  59   1   2    126    218   1         1    134      0      2.2
283  40   1   0    152    223   0         1    181      0      0.0
284  61   1   0    140    207   0         0    138      1      1.9
285  46   1   0    140    311   0         1    120      1      1.8
286  59   1   3    134    204   0         1    162      0      0.8
287  57   1   1    154    232   0         0    164      0      0.0
288  57   1   0    110    335   0         1    143      1      3.0
289  55   0   0    128    205   0         2    130      1      2.0
290  61   1   0    125    203   0         1    161      0      0.0
291  58   1   0    114    318   0         2    140      0      4.4
292  58   0   0    170    225   1         0    146      1      2.8
293  67   1   2    152    212   0         0    150      0      0.8
294  44   1   0    120    169   0         1    144      1      2.8
295  63   1   0    140    187   0         0    144      1      4.0
296  63   0   0    124    197   0         1    136      1      0.0
297  59   1   0    164    176   1         0    90      0      1.0
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   1
1         0   2
2         2   2
3         2   2
4         2   2
5         1   1
6         1   0
7         2   3
8         2   0
9         2   0
10        2   2
11        2   2
12        2   2
13        1   0
14        2   2
15        1   0
16        2   2
17        0   2
18        2   2
19        2   2
20        1   0
21        2   0
22        2   2
23        1   0
24        2   3
25        2   2
26        2   0
27        2   0
28        2   1
29        0   2
...   ...   ...
273      2   1
274      1   1
275      2   2
276      1   1
277      2   0
278      2   2
279      1   1
280      1   0
281      1   0
282      1   1
283      2   0
284      2   1
285      1   2
286      2   2
287      2   1
288      1   1
289      1   1
290      2   1
291      0   3
292      1   2
293      1   0
294      0   0
295      2   2
296      1   0
297      1   2
298      1   0
299      1   0
300      1   2
301      1   1
302      1   1
```

[303 rows x 13 columns]

```
In [31]: print(Y)
```

```
0    1
1    1
2    1
3    1
4    1
5    1
6    1
7    1
8    1
9    1
10   1
11   1
12   1
13   1
14   1
15   1
16   1
17   1
18   1
19   1
20   1
21   1
22   1
23   1
24   1
25   1
26   1
27   1
28   1
29   1
...   ...
273   0
274   0
275   0
276   0
277   0
278   0
279   0
280   0
281   0
282   0
283   0
284   0
285   0
286   0
287   0
288   0
289   0
290   0
291   0
292   0
293   0
294   0
295   0
296   0
297   0
298   0
299   0
300   0
301   0
302   0
Name: target, Length: 303, dtype: int64
```

Splitting the Data into Training data & Test Data

```
In [18]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
In [30]: print(X.shape, X_train.shape, X_test.shape)
```

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

Model Training

Logistic Regression

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

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

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.

```
Out[21]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn',
tol=0.0001, verbose=0, warm_start=False)
```

Model Evaluation

Accuracy Score

```
In [29]: # accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
In [23]: print('Accuracy on Training data : ', training_data_accuracy)
```

Accuracy on Training data : 0.8553719008264463

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

```
In [25]: print('Accuracy on Test data : ', test_data_accuracy)
```

Accuracy on Test data : 0.8032786885245902

Building a Predictive System

```
In [27]: input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)
```

```
# 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 Disease')
else:
    print('The Person has Heart Disease')
```

[0]
The Person does not have a Heart Disease

```
In [ ]:
```