

Importing the Dependencies

```
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

```
# loading the csv data to a Pandas DataFrame
heart_data = pd.read_csv('/content/data.csv')
```

```
# print first 5 rows of the dataset
heart_data.head()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	

```
# print last 5 rows of the dataset
heart_data.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
298	57	0	0	140	241	0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	

```
# number of rows and columns in the dataset
heart_data.shape
```

(303, 14)

```
# 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):
#   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

```

```

# checking for missing values
heart_data.isnull().sum()

```

```

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

```

```

# statistical measures about the data
heart_data.describe()

```

	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000

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

```
1    165
0    138
Name: target, dtype: int64
50%    55.000000    1.000000    1.000000   130.000000   240.000000    0.000000    1.000000
```

1 --> Defective Heart

0 --> Healthy Heart

Splitting the Features and Target

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

```
print(X)
```

	age	sex	cp	trestbps	chol	...	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	...	0	2.3	0	0	1
1	37	1	2	130	250	...	0	3.5	0	0	2
2	41	0	1	130	204	...	0	1.4	2	0	2
3	56	1	1	120	236	...	0	0.8	2	0	2
4	57	0	0	120	354	...	1	0.6	2	0	2
..
298	57	0	0	140	241	...	1	0.2	1	0	3
299	45	1	3	110	264	...	0	1.2	1	0	3
300	68	1	0	144	193	...	0	3.4	1	2	3
301	57	1	0	130	131	...	1	1.2	1	1	3
302	57	0	1	130	236	...	0	0.0	1	1	2

```
[303 rows x 13 columns]
```

```
print(Y)
```

```
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
```

Splitting the Data into Training data & Test Data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=1)

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

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

Model Training

Logistic Regression

```
model = LogisticRegression()

# training the LogisticRegression model with Training data
model.fit(X_train, Y_train)

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning:
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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=100,
                    multi_class='auto', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False)
```

Model Evaluation

Accuracy Score

```
# accuracy on 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

# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

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

```
Accuracy on Test data : 0.819672131147541
```

Building a Predictive System

```
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
```