Libraries and Functions

```
import numpy as np  # To manipulate arrays, mathematical solutions
import pandas as pd # To create data into a structured formate
from sklearn.model_selection import train_test_split # To Split data into training and tes
from sklearn.linear_model import LogisticRegression # To categories data in a boolean valu
from sklearn.metrics import accuracy_score # To Find Accuracy
```

```
heart_data = pd.read_csv('/content/heart.csv') # To Load Data
```

```
heart_data.head() # First Five Rows
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
0	52	1	0	125	212	0	1	168	0	1.0	2	2	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	

```
heart_data.info() # to get information about data
```

```
<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
```

```
heart_data.describe() # it describes the dataset values
```

	age	sex	cp	trestbps	chol	fbs
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000

```
heart_data.shape # To know rows and data
```

```
(1025, 14)
```

```
heart_data['target'].value_counts
```

```
<bound method IndexOpsMixin.value_counts of 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>
```

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  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]
```

```
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
```

Spliiting data into training data and testing data

```
# it distributes the data into two parts - training and testing.
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,stratify=Y)
```

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

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

```
print(Y.shape,Y_train.shape,Y_test.shape)
```

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

```
#model Training
```

Logistic Regression = Binary Classification Model

```
model = LogisticRegression()
```

```
model.fit(X_train,Y_train) # it adjust the weight according to the data
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Conver
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()
```

Model Evolution Accuracy Score

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,Y_train)


print("The Accuracy Score of Training Data : " , training_data_accuracy)
```

```
The Accuracy Score of Training Data :  0.8414634146341463
```

Accuracy ON Test DAta

```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction,Y_test)


print("The Accuracy Score of Testing Data : " , test_data_accuracy)
```

```
The Accuracy Score of Testing Data :  0.8439024390243902
```

Building A Predictive SYstem

```
input_data = (54,1,0,122,286,0,0,116,1,3.2,1,2,2)
input_data_as_numpy_array = np.asarray(input_data)
#reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
prediction = model.predict(input_data_reshaped)
if prediction[0] == 0:
  print("Patients Heart is in Good Condition")
else:
  print("Patients Heart is in Bad Condition")
```

```
Patients Heart is in Good Condition
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not
  "X does not have valid feature names, but"
```