# Heart Disease Prediction using Machine Learning

```
importing the  Libraries
```

In [9]:

```
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 [10]:

```
#loading the dataset to a pandas Dataframe
heart_data = pd.read_csv("C:\\Users\\trylogic\\Downloads\\heart.csv")
```

In [11]:

```

# print the first five rows
heart_data.head()
```

Out[11]:

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

In [12]:

```
# print last five rows
heart_data.tail()
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
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 [13]:

```
# number of rows andcolumns
heart_data.shape
```

Out[13]:

```
(1025, 14)
```

In [14]:

```
heart_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 [15]:

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

Out[15]:

```
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 [18]:

```
heart_data.describe() #describe
```

Out[18]:

	age	sex	cp	trestbps	chol	fbs	reste
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.0000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.5297
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.5278
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.0000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.0000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.0000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.0000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.0000

In [19]:

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

Out[19]:

```
1    526
0    499
Name: target, dtype: int64
```

```
 1 ==> defective heart
 0 ==> is healthy heart
```

```
Splitting Features and Target
```

In [20]:

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

In [21]:

```
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 [22]:

```
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
```

```
Splitting the data into  training data and test data
```

In [23]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_stat
```

In [24]:

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

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

In [25]:

```
print(X_train)
print(Y_train)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
148   52    1   3       152   298    1        1      178      0      1.2   
493   55    1   0       132   353    0        1      132      1      1.2   
991   60    1   0       117   230    1        1      160      1      1.4   
52    38    1   2       138   175    0        1      173      0      0.0   
210   42    1   2       120   240    1        1      194      0      0.8   
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...   
992   50    0   0       110   254    0        0      159      0      0.0   
601   46    1   0       140   311    0        1      120      1      1.8   
356   59    1   0       164   176    1        0       90      0      1.0   
747   60    1   0       117   230    1        1      160      1      1.4   
572   34    1   3       118   182    0        0      174      0      0.0   

     slope  ca  thal  
148      1   0     3  
493      1   1     3  
991      2   2     3  
52       2   4     2  
210      0   0     3  
..     ...  ..   ...  
992      2   0     2  
601      1   2     3  
356      1   2     1  
747      2   2     3  
572      2   0     2  

[820 rows x 13 columns]
148    1
493    0
991    0
52     1
210    1
      ..
992    1
601    0
356    0
747    0
572    1
Name: target, Length: 820, dtype: int64
```

```
Model Training
```

```
logistic regression
```

In [26]:

```
model = LogisticRegression()
```

In [27]:

```
# train the Logistic regression model
model.fit(X_train,Y_train)
```

```
C:\Users\trylogic\AppData\Local\Programs\Python\Python39\lib\site-packages\s
klearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to co
nverge (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 (https://scik
it-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-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

Out[27]:

```
LogisticRegression()
```

```
model evaluation and accurcy score
```

In [28]:

```
#accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,Y_train)
```

In [29]:

```
print('Accuracy on training data : ', training_data_accuracy)
```

```
Accuracy on training data :  0.85
```

In [30]:

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

In [31]:

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

```
Accuracy on test data :  0.8390243902439024
```

```
building a predictive system
```

In [32]:

```
input_data = (62,0,0,138,294,1,1,106,0,1.9,1,3,2)

# changing the input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the np array as we are predicting for 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 does not have a heart disease')
else:
  print('The person has heart disease')
```

```
[0]
The person does not have a heart disease

C:\Users\trylogic\AppData\Local\Programs\Python\Python39\lib\site-packages\s
klearn\base.py:445: UserWarning: X does not have valid feature names, but Lo
gisticRegression was fitted with feature names
  warnings.warn(
```

In [ ]:

```

```

In [ ]:

```

```