Importing the dependencies (i.e, libraries and functions)

```
import numpy as np
import pandas as pd
import matplotlib as mplt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

data collection preprocessing

```
heart_data=pd.read_csv("/heart_disease_data.csv")  #reading csv file
###heart_data=pd.read_csv("/heart.csv",skiprows=5)  #reading csv file and skipping five ro
```

```
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	

```
heart_data.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
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

```
heart_data.shape
```

```
(303, 14)
```

```
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
```

```
#to check wheteher our dataset contains any missing value or not
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
```

```
#to check statistical data in dataset
heart_data.describe()
```

```
                age          sex          cp    trestbps        chol          fbs     restec
```

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

```
1    165
0    138
Name: target, dtype: int64
```

1----->heart disease exists

0----->heart disease does not existslink text

splitting the feature and target columns

```
X=heart_data.drop(columns='target',axis =1) #when we remove column then we write axis=1 an
Y=heart_data['target']
```

```
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
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
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   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      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 and testing data

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2
#by the use of stratify our target data is uniformly divided otherwise it might happen tha
```

```
print(X.shape)
```

```
(303, 13)
```

```
print(X_train.shape)
```

```
(242, 13)
```

```
print(X_test.shape)
```

```
(61, 13)
```

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

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

model training

logistic regression

```
model=LogisticRegression()
```

```
#training our logistic regression model with training data
```

```
model.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
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 evaluation

```
#model accuracy score on training data
```

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

```
print("accuracy score on training data is ",training_data_accuracy)
```

```
accuracy score on training data is  0.8512396694214877
```

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

```
print("accuracy score of test data is:",accuracy_x_test)
```

```
accuracy score of test data is: 0.819672131147541
```

build a predictive system

```
input_data=(52,1,2,172,199,1,1,162,0,0.5,2,0,3)
#change the input array to numpy array
input_data_to_numpy=np.asarray(input_data)
#reshape the numpy array as we are predicting for only 1 instance
reshaped_data=input_data_to_numpy.reshape(1,-1)
prediction=model.predict(reshaped_data)
print(prediction)
if(prediction[0]==0):
  print("the person is having a heart disease")
else:
  print("the person dont have aheart disease")
```

```
[1]
the person dont have aheart disease
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
```

✓ 0s completed at 11:35 AM ● ✕