In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
df = pd.read_csv('Heart_diseases.csv')
df
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

303 rows × 14 columns

In [4]:
```
df.isna()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	False	False	False	False	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	False	False	False	False	False	False	False	False	False	False	False	False	Fals
299	False	False	False	False	False	False	False	False	False	False	False	False	Fals
300	False	False	False	False	False	False	False	False	False	False	False	False	Fals
301	False	False	False	False	False	False	False	False	False	False	False	False	Fals
302	False	False	False	False	False	False	False	False	False	False	False	False	Fals

303 rows × 14 columns

In [5]:
```
df = df.fillna(df.median())
df
```

Out[5]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

303 rows × 14 columns

In [6]:
```
df.dtypes
```

Out[6]:
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [7]:
```
df = df.astype({'oldpeak':int})
```

In [8]:
```
df
```

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	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0	1	1	2	0

303 rows × 14 columns

In [9]:
```
df.dtypes
```

Out[9]:
```
age         int64
sex         int64
cp          int64
trestbps    int64
chol        int64
fbs         int64
restecg     int64
thalach     int64
exang       int64
oldpeak     int32
slope       int64
ca          int64
thal        int64
target      int64
dtype: object
```

In [10]:
```
df = df.drop_duplicates()
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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 [12]:
```
df = df.drop(['cp'], axis = 1)
df = df.drop(columns = ('thal'))
```

In [13]:
```
df.sum()
```

Out[13]:
```
age         16435
sex           206
trestbps    39744
chol        74443
fbs            45
restecg       159
thalach     45170
exang          99
oldpeak       232
slope         422
ca            217
target        164
dtype: int64
```

In [14]:
```
df = df.drop(columns = 'slope')
df
```

Out[14]:

	age	sex	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	ca	target
0	63	1	145	233	1	0	150	0	2	0	1
1	37	1	130	250	0	1	187	0	3	0	1
2	41	0	130	204	0	0	172	0	1	0	1
3	56	1	120	236	0	1	178	0	0	0	1
4	57	0	120	354	0	1	163	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	140	241	0	1	123	1	0	0	0
299	45	1	110	264	0	1	132	0	1	0	0
300	68	1	144	193	1	1	141	0	3	2	0
301	57	1	130	131	0	1	115	1	1	1	0
302	57	0	130	236	0	0	174	0	0	1	0

302 rows × 11 columns

In [15]:
```
x = df.iloc [:,0:10]
x
```

Out[15]:

	age	sex	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	ca
0	63	1	145	233	1	0	150	0	2	0
1	37	1	130	250	0	1	187	0	3	0
2	41	0	130	204	0	0	172	0	1	0
3	56	1	120	236	0	1	178	0	0	0
4	57	0	120	354	0	1	163	1	0	0
...	...	...	...	...	...	...	...	...	...	...
298	57	0	140	241	0	1	123	1	0	0
299	45	1	110	264	0	1	132	0	1	0
300	68	1	144	193	1	1	141	0	3	2
301	57	1	130	131	0	1	115	1	1	1
302	57	0	130	236	0	0	174	0	0	1

302 rows × 10 columns

In [16]:
```
y = df['target']
```

In [17]:
```
y
x_normalization = (x - np.min(x))/((np.max(x) - np.min(x)))
y_normalization = (y - np.min(y))/((np.max(y) - np.min(y)))
```

```
C:\Users\HARSH\anaconda3\Lib\site-packages\numpy\core\fromnumeric.py:84: FutureWar
ning: In a future version, DataFrame.min(axis=None) will return a scalar min over
the entire DataFrame. To retain the old behavior, use 'frame.min(axis=0)' or just
'frame.min()'
  return reduction(axis=axis, out=out, **passkwargs)
C:\Users\HARSH\anaconda3\Lib\site-packages\numpy\core\fromnumeric.py:84: FutureWar
ning: In a future version, DataFrame.max(axis=None) will return a scalar max over
the entire DataFrame. To retain the old behavior, use 'frame.max(axis=0)' or just
'frame.max()'
  return reduction(axis=axis, out=out, **passkwargs)
C:\Users\HARSH\anaconda3\Lib\site-packages\numpy\core\fromnumeric.py:84: FutureWar
ning: In a future version, DataFrame.min(axis=None) will return a scalar min over
the entire DataFrame. To retain the old behavior, use 'frame.min(axis=0)' or just
'frame.min()'
  return reduction(axis=axis, out=out, **passkwargs)
```

In [18]:
```
from sklearn.model_selection import train_test_split
```

In [19]:
```
x_train,x_test,y_train,y_test = train_test_split (x,y,test_size = 0.25)
```

In [20]:
```
from sklearn.linear_model import LogisticRegression
```

In [21]:
```
model = LogisticRegression()
```

In [22]:
```
model.fit(x_train,y_train)
```

```
C:\Users\HARSH\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[22]:

▾ LogisticRegression

LogisticRegression()

In [23]:
```
yPrediction = model.predict(x_test)
```

In [24]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,yPrediction))
```

```
0.8157894736842105
```

In [ ]: