IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/heart.csv')
df
```

	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	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	

1025 rows × 14 columns

```
df.head()
```

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

```
df.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	c
1020	59	1	1	140	221	0	1	164	1	0.0	2	
1021	60	1	0	125	258	0	0	141	1	2.8	1	
1022	47	1	0	110	275	0	0	118	1	1.0	1	
1023	50	0	0	110	254	0	0	159	0	0.0	2	
1024	54	1	0	120	188	0	1	113	0	1.4	1	

```
df.info
```

```
<bound method DataFrame.info of       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  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
...     ...  ..   ...     ...
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]>
```

FINDING MISSING VALUES

```
df.isna().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
```

SEPARATING X AND Y VARIABLES

```
x=df.iloc[:,:-1].values
x
```

```
array([[52.,  1.,  0., ...,  2.,  2.,  3.],
       [53.,  1.,  0., ...,  0.,  0.,  3.],
       [70.,  1.,  0., ...,  0.,  0.,  3.],
       ...,
       [47.,  1.,  0., ...,  1.,  1.,  2.],
       [50.,  0.,  0., ...,  2.,  0.,  2.],
       [54.,  1.,  0., ...,  1.,  1.,  3.]])
```

```
y=df.iloc[:,-1].values
y
```

```
array([0, 0, 0, ..., 0, 1, 0])
```

SPLITTING INTO TRAINING AND TESTING DATA

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
x_train
```

```
    array([[57.,  1.,  1., ...,  2.,  0.,  3.],
           [66.,  1.,  0., ...,  2.,  0.,  1.],
           [38.,  1.,  2., ...,  2.,  4.,  2.],
           ...,
           [46.,  1.,  0., ...,  2.,  0.,  3.],
           [56.,  1.,  1., ...,  2.,  0.,  3.],
           [62.,  0.,  2., ...,  1.,  1.,  3.]])
```

```
x_test
```

```
    array([[63.,  0.,  2., ...,  2.,  0.,  2.],
           [44.,  1.,  2., ...,  2.,  0.,  2.],
           [65.,  0.,  2., ...,  2.,  0.,  2.],
           ...,
           [51.,  1.,  2., ...,  1.,  0.,  2.],
           [57.,  1.,  2., ...,  1.,  1.,  3.],
           [34.,  0.,  1., ...,  2.,  0.,  2.]])
```

NORMALIZATION USING MINMAX SCALER

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
x_train
```

```
    array([[0.58333333, 1.        , 0.33333333, ..., 1.        , 0.        ,
            1.        ],
           [0.77083333, 1.        , 0.        , ..., 1.        , 0.        ,
            0.33333333],
           [0.1875    , 1.        , 0.66666667, ..., 1.        , 1.        ,
            0.66666667],
           ...,
           [0.35416667, 1.        , 0.        , ..., 1.        , 0.        ,
            1.        ],
           [0.5625    , 1.        , 0.33333333, ..., 1.        , 0.        ,
            1.        ],
           [0.6875    , 0.        , 0.66666667, ..., 0.5       , 0.25      ,
            1.        ]])
```

```
x_test
```

```
    array([[0.72340426, 0.        , 0.66666667, ..., 1.        , 0.        ,
            0.66666667],
           [0.31914894, 1.        , 0.66666667, ..., 1.        , 0.        ,
            0.66666667],
           [0.76595745, 0.        , 0.66666667, ..., 1.        , 0.        ,
            0.66666667],
           ...,
           [0.46808511, 1.        , 0.66666667, ..., 0.5       , 0.        ,
            0.66666667],
           [0.59574468, 1.        , 0.66666667, ..., 0.5       , 0.25      ,
            1.        ],
           [0.10638298, 0.        , 0.33333333, ..., 1.        , 0.        ,
            0.66666667]])
```

MODEL CREATION

KNN MODEL AND PERFORMANCE EVALUATION

```
from sklearn.neighbors import KNeighborsClassifier
knn_model=KNeighborsClassifier(n_neighbors=7)
knn_model.fit(x_train,y_train)
y_pred=knn_model.predict(x_test)
y_pred
```

```
    array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1,
           1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0,
           1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0,
           1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
           0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
           1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
           0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
           1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
           0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1,
           1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
           0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
           0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1,
           1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1])
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
result=confusion_matrix(y_test,y_pred)
result
```

```
    array([[136,  22],
           [ 24, 126]])
```

```
score=accuracy_score(y_pred,y_test)
score
```

```
    0.8506493506493507
```

NAIVE BAYES MODEL AND PERFORMANCE EVALUATION

```
from sklearn.naive_bayes import GaussianNB
nb_model=GaussianNB()
nb_model.fit(x_train,y_train)
y_pred=nb_model.predict(x_test)
y_pred
```

```
    array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
           1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
           1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
           1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0,
           1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
           0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
           1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
           0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1,
           1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
           1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,
           0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0,
           0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
           1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1])
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
result=confusion_matrix(y_test,y_pred)
result
```

```
    array([[135,  23],
           [ 16, 134]])
```

```
score=accuracy_score(y_test,y_pred)
score
```

```
0.8733766233766234
```

SVM MODEL AND PERFORMANCE EVALUATION

```
from sklearn.svm import SVC
svm_model=SVC()
svm_model.fit(x_train,y_train)
y_pred=svm_model.predict(x_test)
y_pred
```

```
array([1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1])
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
result=confusion_matrix(y_test,y_pred)
result
```

```
array([[141,  17],
       [ 25, 125]])
```

```
score=accuracy_score(y_test,y_pred)
score
```

```
0.8636363636363636
```