```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/heart(1).csv')
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1.0	0	150	0	2.3	0	0	1.0	1
1	37	1	2	130	250	0.0	1	187	0	3.5	0	0	2.0	1
2	41	0	1	130	204	0.0	0	172	0	1.4	2	0	2.0	1
3	56	1	1	120	236	0.0	1	178	0	0.8	2	0	2.0	1
4	57	0	0	120	354	0.0	1	163	1	0.6	2	0	2.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0.0	1	123	1	0.2	1	0	3.0	0
299	45	1	3	110	264	0.0	1	132	0	1.2	1	0	3.0	0
300	68	1	0	144	193	1.0	1	141	0	3.4	1	2	3.0	0
301	57	1	0	130	131	0.0	1	115	1	1.2	1	1	3.0	0
302	57	0	1	130	236	0.0	0	174	0	0.0	1	1	2.0	0

303 rows × 14 columns

```
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1.0	0	150	0	2.3	0	0	1.0	1
1	37	1	2	130	250	0.0	1	187	0	3.5	0	0	2.0	1
2	41	0	1	130	204	0.0	0	172	0	1.4	2	0	2.0	1
3	56	1	1	120	236	0.0	1	178	0	0.8	2	0	2.0	1
4	57	0	0	120	354	0.0	1	163	1	0.6	2	0	2.0	1

```
df.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	t
298	57	0	0	140	241	0.0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0.0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1.0	1	141	0	3.4	1	2	
301	57	1	0	130	131	0.0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0.0	0	174	0	0.0	1	1	

```
df.shape
```

```
(303, 14)
```

```
df.dtypes
```

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

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

```
age         0
sex         0
```

```
cp          0
trestbps    0
chol        0
fbs         5
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        6
target      0
dtype: int64
```

Filling the missing values

```
z=df['fbs'].mean()
w=df['thal'].mean()
df['fbs'].fillna(z,inplace=True)
df['thal'].fillna(w,inplace=True)
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	t
0	63	1	3	145	233	1.0	0	150	0	2.3	0	0	
1	37	1	2	130	250	0.0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0.0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0.0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0.0	1	163	1	0.6	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0.0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0.0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1.0	1	141	0	3.4	1	2	
301	57	1	0	130	131	0.0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0.0	0	174	0	0.0	1	1	

303 rows × 14 columns

```
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
```

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restec
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.151007	0.52805
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.355676	0.52586
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.00000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.00000

Separating input and output samples

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

```
(array([[63.,  1.,  3., ...,  0.,  0.,  1.],
        [37.,  1.,  2., ...,  0.,  0.,  2.],
        [41.,  0.,  1., ...,  2.,  0.,  2.],
        ...,
        [68.,  1.,  0., ...,  1.,  2.,  3.],
        [57.,  1.,  0., ...,  1.,  1.,  3.],
        [57.,  0.,  1., ...,  1.,  1.,  2.]]),
 array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]))
```

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,random_state=42)
```

```
x_train
```

```
array([[39.,  0.,  2., ...,  2.,  0.,  2.],
       [29.,  1.,  1., ...,  2.,  0.,  2.],
       [50.,  0.,  2., ...,  1.,  0.,  2.],
       ...,
       [69.,  1.,  3., ...,  1.,  1.,  2.],
       [46.,  1.,  0., ...,  2.,  0.,  3.],
       [63.,  0.,  1., ...,  2.,  2.,  2.]])
```

```
x_test
```

```
array([[57.,  1.,  0., ...,  1.,  1.,  1.],
       [59.,  1.,  3., ...,  1.,  0.,  3.],
       [57.,  1.,  2., ...,  2.,  1.,  3.],
       ...,
       [67.,  0.,  0., ...,  2.,  2.,  2.],
       [58.,  1.,  2., ...,  1.,  0.,  3.],
       [76.,  0.,  2., ...,  1.,  0.,  2.]])
```

```
y_train
```

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

```
y_test
```

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

Normalisation

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

```
x_train
```

```
array([[0.20833333, 0.        , 0.66666667, ..., 1.        , 0.        ,
        0.66666667],
       [0.        , 1.        , 0.33333333, ..., 1.        , 0.        ,
        0.66666667],
       [0.4375    , 0.        , 0.66666667, ..., 0.5       , 0.        ,
        0.66666667],
       ...,
       [0.83333333, 1.        , 1.        , ..., 0.5       , 0.25      ,
        0.66666667],
       [0.35416667, 1.        , 0.        , ..., 1.        , 0.        ,
        1.        ],
       [0.70833333, 0.        , 0.33333333, ..., 1.        , 0.5       ,
        0.66666667]])
```

```
x_test
```

```
array([[0.58333333, 1.        , 0.        , ..., 0.5       , 0.25      ,
        0.33333333],
       [0.625     , 1.        , 1.        , ..., 0.5       , 0.        ,
        1.        ],
       [0.58333333, 1.        , 0.66666667, ..., 1.        , 0.25      ,
        1.        ],
       ...,
       [0.79166667, 0.        , 0.        , ..., 1.        , 0.5       ,
        0.66666667],
       [0.60416667, 1.        , 0.66666667, ..., 0.5       , 0.        ,
        1.        ],
       [0.97916667, 0.        , 0.66666667, ..., 0.5       , 0.        ,
        0.66666667]])
```

Model creation

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

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

```
y_test
```

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

Performace evaluation

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

```
0.8241758241758241
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.80      0.80      0.80        41
               1       0.84      0.84      0.84        50

        accuracy                           0.82        91
       macro avg       0.82      0.82      0.82        91
```

weighted avg	0.82	0.82	0.82	91