In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df=pd.read_csv('heart.csv')
df.head()
```

Out[2]:

	Age	Sex	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_
0	40	M	140	289	0	Normal	172	N	0.0	
1	49	F	160	180	0	Normal	156	N	1.0	
2	37	M	130	283	0	ST	98	N	0.0	
3	48	F	138	214	0	Normal	108	Y	1.5	
4	54	M	150	195	0	Normal	122	N	0.0	

In [3]:

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

Out[3]:

```
Age               0
Sex               0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [4]:

```
x=df.drop('HeartDisease',axis=1)
y=df.HeartDisease

```

In [5]:

```
from sklearn.preprocessing import OneHotEncoder,StandardScaler
from sklearn.compose import ColumnTransformer
```

In [6]:

```
encoder=OneHotEncoder()
features=['Sex','RestingECG','ExerciseAngina','ST_Slope']
values=['Age','RestingBP','Cholesterol','MaxHR','Oldpeak']
tranformer=ColumnTransformer([
    ('one_hot',encoder,features),
    ('values',StandardScaler(),values)]
    ,remainder='passthrough')
transformed_x=tranformer.fit_transform(x)
transformed_x
```

Out[6]:

```
array([[ 0.        ,  1.        ,  0.        , ...,  1.38292822,
        -0.83243239,  0.        ],
       [ 1.        ,  0.        ,  0.        , ...,  0.75415714,
         0.10566353,  0.        ],
       [ 0.        ,  1.        ,  0.        , ..., -1.52513802,
        -0.83243239,  0.        ],
       ...,
       [ 0.        ,  1.        ,  0.        , ..., -0.85706875,
         0.29328271,  0.        ],
       [ 1.        ,  0.        ,  1.        , ...,  1.4615246 ,
        -0.83243239,  0.        ],
       [ 0.        ,  1.        ,  0.        , ...,  1.42222641,
        -0.83243239,  0.        ]])
```

In [7]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(transformed_x,y,test_size=0.2,random_state=4
```

In [8]:

```
x_train.shape , x_test.shape
```

Out[8]:

```
((734, 16), (184, 16))
```

In [9]:

```
y_train.shape , y_test.shape
```

Out[9]:

```
((734,), (184,))
```

Addign layers

In [10]:

```
from keras.models import Sequential
from keras.layers import Dense
```

In [11]:

```
model=Sequential()
model.add(Dense(128,activation='relu',input_shape=(16,)))
model.add(Dense(128,activation='relu'))
model.add(Dense(2,activation='softmax'))
model.compile(loss='binary_crossentropy',optimizer='Adam',metrics=['Accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               2176

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 2)                 258

=================================================================
Total params: 18946 (74.01 KB)
Trainable params: 18946 (74.01 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [36]:

```
from keras.utils import to_categorical
y_train_encoded=to_categorical(y_train)
y_test_encoded=to_categorical(y_test)
model.fit(x_train,y_train_encoded,epochs=100,batch_size=10,shuffle=True,validation_split=0.
```

```
59/59 [==============================] - 0s 3ms/step - loss: 6.0987e-06 - Accur
acy: 1.0000 - val_loss: 0.1680 - val_Accuracy: 0.9592
Epoch 83/100
59/59 [==============================] - 0s 3ms/step - loss: 5.7809e-06 - Accur
acy: 1.0000 - val_loss: 0.1788 - val_Accuracy: 0.9592
Epoch 84/100
59/59 [==============================] - 0s 3ms/step - loss: 5.7678e-06 - Accur
acy: 1.0000 - val_loss: 0.1748 - val_Accuracy: 0.9592
Epoch 85/100
59/59 [==============================] - 0s 3ms/step - loss: 5.3662e-06 - Accur
acy: 1.0000 - val_loss: 0.1748 - val_Accuracy: 0.9592
Epoch 86/100
59/59 [==============================] - 0s 3ms/step - loss: 5.2182e-06 - Accur
acy: 1.0000 - val_loss: 0.1791 - val_Accuracy: 0.9592
Epoch 87/100
59/59 [==============================] - 0s 3ms/step - loss: 5.0639e-06 - Accur
acy: 1.0000 - val_loss: 0.1829 - val_Accuracy: 0.9592
Epoch 88/100
59/59 [==============================] - 0s 3ms/step - loss: 5.2929e-06 - Accur
acy: 1.0000 - val_loss: 0.1790 - val_Accuracy: 0.9592
```

In [33]:

```
y_pred=model.predict(x_test)
y_preds=np.argmax(y_pred,axis=1)
y_preds
```

```
6/6 [==============================] - 0s 0s/step
```

Out[33]:

```
array([0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 1, 1, 0, 1], dtype=int64)
```

In [34]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [37]:

```
accuracy=accuracy_score(y_preds,y_test)
accuracy

```

Out[37]:

```
0.7934782608695652
```

In [38]:

```
confusion_matrix=confusion_matrix(y_preds,y_test)
confusion_matrix
```

Out[38]:

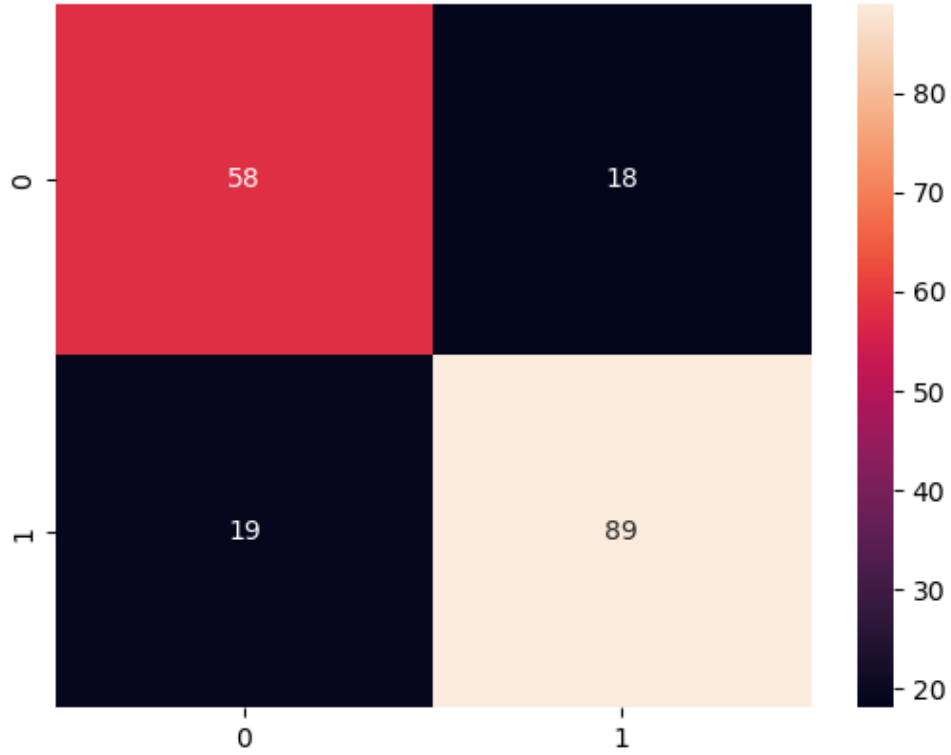
```
array([[60, 21],
       [17, 86]], dtype=int64)
```

In [31]:

```
import seaborn as sns
sns.heatmap(confusion_matrix,annot=True)
```

Out[31]:

<Axes: >



In [ ]:

```

```