In [53]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

In [54]:
```
df = pd.read_csv(r"Downloads\BostonHousingData.csv")
```

In [51]:
```
df
```

Out[51]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

In [47]:
```
x = df.drop("MEDV", axis=1).values
y = df["MEDV"].values
```

In [30]:

```
x.shape
```

Out[30]:

```
(506, 13)
```

In [31]:

```
y.shape
```

Out[31]:

```
(506,)
```

In [32]:

```
 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [33]:

```
def shape():
    print("x_train Shape :",x_train.shape)
    print("x_test Shape :",x_test.shape)
    print("y_train shape :",y_train.shape)
    print("y_test shape :",y_test.shape)

shape()
```

```
x_train Shape : (404, 13)
x_test Shape : (102, 13)
y_train shape : (404,)
y_test shape : (102,)
```

In [34]:

```
mean=x_train.mean(axis=0)
std=x_train.std(axis=0)

x_train=(x_train-mean)/std
x_test=(x_test-mean)/std
```

In [35]:

```
x_train[0]
```

Out[35]:

```
array([-0.40514967, -0.47664927, -1.30436293, -0.28828791, -0.59653159,
        2.12316802, -0.56147089, -0.26583291, -0.75855792, -1.26661874,
       -0.30023431,  0.40005185, -1.14698031])
```

In [36]:

```
y_train[0]
```

Out[36]:

```
50.0
```

In [37]:

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

In [38]:

```
model=Sequential()
model.add(Dense(128,activation='relu',input_shape=(x_train[0].shape)))
model.add(Dense(64,activation='relu'))
model.add(Dense(1,activation='linear'))
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 128)               1792

 dense_4 (Dense)             (None, 64)                8256

 dense_5 (Dense)             (None, 1)                 65

=================================================================
Total params: 10,113
Trainable params: 10,113
Non-trainable params: 0
_________________________________________________________________
```

In [39]:

```
model.fit(x_train, y_train, epochs=100, batch_size=1, verbose=1,validation_data=(x_test, y_test))
```

```
Epoch 1/100
404/404 [==============================] - 2s 2ms/step - loss: 153.2698 - mae: 8.3323 - val_loss: 21.211
5 - val_mae: 3.0602
Epoch 2/100
404/404 [==============================] - 1s 3ms/step - loss: 20.6306 - mae: 3.1361 - val_loss: 18.0010
- val_mae: 2.7586
Epoch 3/100
404/404 [==============================] - 1s 3ms/step - loss: 16.5458 - mae: 2.8533 - val_loss: 14.0782
- val_mae: 2.5288
Epoch 4/100
404/404 [==============================] - 1s 2ms/step - loss: 14.6060 - mae: 2.6943 - val_loss: 20.7477
- val_mae: 3.3892
Epoch 5/100
404/404 [==============================] - 1s 3ms/step - loss: 14.7877 - mae: 2.6624 - val_loss: 13.9981
- val_mae: 2.6260
Epoch 6/100
404/404 [==============================] - 2s 4ms/step - loss: 12.7725 - mae: 2.4977 - val_loss: 12.1307
- val_mae: 2.5411
Epoch 7/100
```

In [40]:

```
x_test[8]
```

Out[40]:

```
array([-0.40366143,  1.49412257, -1.16192331, -0.28828791, -1.03117975,
        0.61426321, -1.38093798,  1.30369772, -0.52753085, -0.05411049,
       -1.49934504,  0.36777102, -1.11383419])
```

In [41]:

```
test_input=[[-0.42101827, -0.50156705, -1.13081973, -0.25683275, -0.55572682, 0.19758953, 0.20684755, -0.342

print("ActuaOutput :",y_test[8])
print("Predicted Output :",model.predict(test_input))
```

```
ActuaOutput : 30.5
1/1 [==============================] - 0s 204ms/step
Predicted Output : [[21.41977]]
```

In [42]:

```
mse_nn,mae_nn=model.evaluate(x_test,y_test)

print('Mean squared error on test data :',mse_nn)
print('Mean absolute error on test data :',mae_nn)
```

```
4/4 [==============================] - 0s 7ms/step - loss: 8.5088 - mae: 1.9929
Mean squared error on test data : 8.508825302124023
Mean absolute error on test data : 1.9929177761077881
```

In [43]:

```
from sklearn.metrics import r2_score

y_dl=model.predict(x_test)
r2=r2_score(y_test,y_dl)

print('R2 Score :',r2)
```

```
4/4 [==============================] - 0s 2ms/step
R2 Score : 0.8749185839063687
```

In [52]:

```
df
```

Out[52]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

In [ ]: