In [1]:
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

```
C:\Users\OM\AppData\Roaming\Python\Python311\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

In [2]:
```python
import keras
from keras.models import Sequential  # To define a sequential model
from keras.layers import Dense
```

In [3]:
```python
boston = pd.read_csv("boston_house_prices.csv")
```

In [4]:
```python
boston.head()
```

Out [4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
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

In [6]:
```python
boston.columns
```

Out [6]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'PRICE'],
      dtype='object')
```

In [8]:
```python
boston.shape
```

Out [8]:
```
(506, 14)
```

In [10]:
```python
boston.describe
```

Out [10]:
```
<bound method NDFrame.describe of         CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     PTRATIO       B  LSTAT  PRICE
0       15.3  396.90   4.98   24.0
1       17.8  396.90   9.14   21.6
2       17.8  392.83   4.03   34.7
3       18.7  394.63   2.94   33.4
4       18.7  396.90   5.33   36.2
..       ...     ...    ...    ...
501     21.0  391.99   9.67   22.4
502     21.0  396.90   9.08   20.6
503     21.0  396.90   5.64   23.9
504     21.0  393.45   6.48   22.0
505     21.0  396.90   7.88   11.9

[506 rows x 14 columns]>
```

In [11]:
```python
boston.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
```

```
 12  LSTAT    506 non-null    float64
 13  PRICE    506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

In [18]:

```python
boston.isnull().sum().sum()
```

Out [18]: 0

In [20]:

```python
boston.isnull().sum()
```

Out [20]:

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

In [24]:

```python
X = boston.iloc[:, :-1]
y = boston.iloc[:,-1]
```

In [25]:

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
```

In [26]:

```python
scaler = StandardScaler()  # Initializing StandardScaler
X_train_scaled = scaler.fit_transform(X_train)  # Fit and transform training data
X_test_scaled = scaler.transform(X_test)
```

In [27]:

```python
lr_model = LinearRegression()  # Initializing Linear Regression Model
lr_model.fit(X_train_scaled, y_train)
```

Out [27]:

```
▾ LinearRegression
LinearRegression()
```

In [28]:

```python
y_pred_lr = lr_model.predict(X_test_scaled)
```

In [29]:

```python
mse_lr = mean_squared_error(y_test, y_pred_lr)  # Mean Squared Error
mae_lr = mean_absolute_error(y_test, y_pred_lr)  # Mean Absolute Error
r2_lr = r2_score(y_test, y_pred_lr)  # R² Score (Model accuracy measure)
# Displaying evaluation metrics
print("Linear Regression Model Evaluation:")
print(f"Mean Squared Error: {mse_lr}")
print(f"Mean Absolute Error: {mae_lr}")
print(f"R2 Score: {r2_lr}")
```

```
Linear Regression Model Evaluation:
Mean Squared Error: 25.41958712682183
Mean Absolute Error: 3.367790983796574
R2 Score: 0.7263451459702512
```

In [32]:

```python
model = Sequential([
Dense(128, activation='relu', input_dim=13),  # Input layer (3 features) & first hidden layer (128 neurons)
Dense(64, activation='relu'),  # Second hidden layer with 64 neurons
Dense(32, activation='relu'),  # Third hidden layer with 32 neurons
Dense(16, activation='relu'),  # Fourth hidden layer with 16 neurons
Dense(1)  # Output layer (Predicting a single value - House Price)
])
# Compiling the model
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

```
C:\Users\OM\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer
in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [33]:

```python
history = model.fit(X_train_scaled, y_train, epochs=100, validation_split=0.05, verbose=1)
```

```
Epoch 1/100
 [1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m22s[0m 207ms/step - loss: 549.5116 - mae: 21.8044 - val_loss: 402.6525 - val_mae: 19.0
Epoch 2/100
 [1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 427.1573 - mae: 18.8787 - val_loss: 276.1398 - val_mae: 15.326
Epoch 3/100
 [1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 294.9018 - mae: 14.7023 - val_loss: 100.4361 - val_mae: 9.0217
Epoch 4/100
```

```
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 26ms/step - loss: 110.3882 - mae: 8.2656 - val_loss: 52.0823 - val_mae: 6.3884
Epoch 5/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m1s[0m 21ms/step - loss: 70.3003 - mae: 6.5354 - val_loss: 25.1947 - val_mae: 4.2947
Epoch 6/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 19ms/step - loss: 36.1650 - mae: 4.3644 - val_loss: 11.9273 - val_mae: 2.8416
Epoch 7/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 24.8440 - mae: 3.8076 - val_loss: 7.8919 - val_mae: 2.2425
Epoch 8/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 22.6092 - mae: 3.4037 - val_loss: 6.6428 - val_mae: 2.1221
Epoch 9/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 24ms/step - loss: 15.0737 - mae: 2.8003 - val_loss: 6.5706 - val_mae: 2.0850
Epoch 10/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 19ms/step - loss: 16.3587 - mae: 2.9302 - val_loss: 6.4549 - val_mae: 2.1019
Epoch 11/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 21.3558 - mae: 3.0639 - val_loss: 6.4152 - val_mae: 2.0744
Epoch 12/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 18ms/step - loss: 15.0428 - mae: 2.7416 - val_loss: 6.2342 - val_mae: 1.9934
Epoch 13/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 13.7386 - mae: 2.7507 - val_loss: 6.1867 - val_mae: 2.0261
Epoch 14/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 11.9086 - mae: 2.5248 - val_loss: 6.3960 - val_mae: 1.9694
Epoch 15/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 19ms/step - loss: 14.4385 - mae: 2.7014 - val_loss: 5.8034 - val_mae: 1.8809
Epoch 16/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 10.7620 - mae: 2.3470 - val_loss: 6.0085 - val_mae: 1.9316
Epoch 17/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 18ms/step - loss: 16.5408 - mae: 2.7046 - val_loss: 6.2809 - val_mae: 1.9236
Epoch 18/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 20ms/step - loss: 11.2754 - mae: 2.3570 - val_loss: 5.7456 - val_mae: 1.8806
Epoch 19/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 11.2569 - mae: 2.4435 - val_loss: 6.6163 - val_mae: 1.9459
Epoch 20/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 11.9238 - mae: 2.5025 - val_loss: 5.5114 - val_mae: 1.8192
Epoch 21/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 14.4591 - mae: 2.4678 - val_loss: 5.7948 - val_mae: 1.8425
Epoch 22/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 9.7901 - mae: 2.2451 - val_loss: 5.7928 - val_mae: 1.8384
Epoch 23/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 9.1815 - mae: 2.2736 - val_loss: 4.9069 - val_mae: 1.7959
Epoch 24/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 18ms/step - loss: 8.8860 - mae: 2.2276 - val_loss: 6.2830 - val_mae: 1.9156
Epoch 25/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m1s[0m 14ms/step - loss: 12.9095 - mae: 2.3663 - val_loss: 4.8837 - val_mae: 1.7791
Epoch 26/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 21ms/step - loss: 13.3921 - mae: 2.3648 - val_loss: 5.5222 - val_mae: 1.8161
Epoch 27/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 10.4090 - mae: 2.1994 - val_loss: 5.2954 - val_mae: 1.8099
Epoch 28/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 16ms/step - loss: 10.3735 - mae: 2.1638 - val_loss: 4.8744 - val_mae: 1.7566
Epoch 29/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 8.3131 - mae: 2.0868 - val_loss: 5.9593 - val_mae: 1.8945
Epoch 30/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 7.9329 - mae: 2.0326 - val_loss: 5.3519 - val_mae: 1.8032
Epoch 31/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 12ms/step - loss: 9.2763 - mae: 2.1416 - val_loss: 5.2545 - val_mae: 1.8281
Epoch 32/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 8.1264 - mae: 2.0106 - val_loss: 4.9861 - val_mae: 1.7613
Epoch 33/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 12ms/step - loss: 7.6154 - mae: 2.0318 - val_loss: 4.7375 - val_mae: 1.7328
Epoch 34/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 12.0724 - mae: 2.2385 - val_loss: 4.9443 - val_mae: 1.7597
Epoch 35/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 21ms/step - loss: 7.5880 - mae: 1.9634 - val_loss: 4.8182 - val_mae: 1.7918
Epoch 36/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 7.3464 - mae: 1.9208 - val_loss: 4.9693 - val_mae: 1.7646
Epoch 37/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 19ms/step - loss: 6.8558 - mae: 1.8410 - val_loss: 4.6264 - val_mae: 1.7086
Epoch 38/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 9.1554 - mae: 2.0002 - val_loss: 4.8801 - val_mae: 1.8048
Epoch 39/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 9.5853 - mae: 2.0179 - val_loss: 5.1495 - val_mae: 1.7530
Epoch 40/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 6.5299 - mae: 1.8356 - val_loss: 5.0869 - val_mae: 1.7791
Epoch 41/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 7.2939 - mae: 1.8407 - val_loss: 4.7951 - val_mae: 1.7351
Epoch 42/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 8.8814 - mae: 1.9153 - val_loss: 5.2479 - val_mae: 1.8361
Epoch 43/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 19ms/step - loss: 8.9590 - mae: 2.0088 - val_loss: 4.9819 - val_mae: 1.8302
Epoch 44/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 6.6945 - mae: 1.8723 - val_loss: 4.3550 - val_mae: 1.6864
Epoch 45/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 20ms/step - loss: 6.3712 - mae: 1.8540 - val_loss: 4.2182 - val_mae: 1.8049
Epoch 46/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 5.5590 - mae: 1.7426 - val_loss: 5.4906 - val_mae: 1.8787
Epoch 47/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 6.4291 - mae: 1.7542 - val_loss: 5.0850 - val_mae: 1.7853
Epoch 48/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 7.7753 - mae: 1.9620 - val_loss: 4.2119 - val_mae: 1.8052
Epoch 49/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 5.9650 - mae: 1.7789 - val_loss: 5.8454 - val_mae: 1.9009
Epoch 50/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 5.1682 - mae: 1.6709 - val_loss: 5.2678 - val_mae: 1.8551
Epoch 51/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 6.4326 - mae: 1.8791 - val_loss: 4.6717 - val_mae: 1.8802
Epoch 52/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 7.5427 - mae: 1.9518 - val_loss: 4.4100 - val_mae: 1.8116
Epoch 53/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 5.6655 - mae: 1.7381 - val_loss: 4.5281 - val_mae: 1.8300
Epoch 54/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 18ms/step - loss: 6.3646 - mae: 1.8299 - val_loss: 5.4328 - val_mae: 1.9223
Epoch 55/100
```

```
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 6.2987 - mae: 1.7489 - val_loss: 5.2997 - val_mae: 1.9369
Epoch 56/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 18ms/step - loss: 4.9288 - mae: 1.5606 - val_loss: 5.6198 - val_mae: 1.9757
Epoch 57/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 16ms/step - loss: 5.8097 - mae: 1.7481 - val_loss: 5.9288 - val_mae: 1.9656
Epoch 58/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 5.1237 - mae: 1.6716 - val_loss: 5.9587 - val_mae: 2.0220
Epoch 59/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 5.1763 - mae: 1.6840 - val_loss: 4.9595 - val_mae: 1.9189
Epoch 60/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 5.0444 - mae: 1.6479 - val_loss: 5.0309 - val_mae: 1.9479
Epoch 61/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 12ms/step - loss: 4.8980 - mae: 1.6062 - val_loss: 4.8783 - val_mae: 1.9099
Epoch 62/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 20ms/step - loss: 4.6103 - mae: 1.6085 - val_loss: 4.7392 - val_mae: 1.9074
Epoch 63/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 4.7150 - mae: 1.5570 - val_loss: 5.2409 - val_mae: 1.9913
Epoch 64/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 12ms/step - loss: 4.9795 - mae: 1.5709 - val_loss: 5.4646 - val_mae: 1.9737
Epoch 65/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 4.5279 - mae: 1.5737 - val_loss: 5.8937 - val_mae: 1.9655
Epoch 66/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 3.9913 - mae: 1.5013 - val_loss: 6.7563 - val_mae: 2.1813
Epoch 67/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 11ms/step - loss: 4.4663 - mae: 1.5470 - val_loss: 4.9811 - val_mae: 1.9530
Epoch 68/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 19ms/step - loss: 4.0378 - mae: 1.4770 - val_loss: 5.4439 - val_mae: 1.9365
Epoch 69/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 4.3834 - mae: 1.5772 - val_loss: 6.0779 - val_mae: 2.1283
Epoch 70/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 3.9883 - mae: 1.4828 - val_loss: 5.8573 - val_mae: 2.0260
Epoch 71/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 18ms/step - loss: 3.9218 - mae: 1.4339 - val_loss: 5.9514 - val_mae: 2.0491
Epoch 72/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 4.1200 - mae: 1.4994 - val_loss: 5.7802 - val_mae: 2.0665
Epoch 73/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m1s[0m 32ms/step - loss: 3.9186 - mae: 1.3847 - val_loss: 5.9706 - val_mae: 2.1023
Epoch 74/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 3.3908 - mae: 1.3816 - val_loss: 6.2767 - val_mae: 2.0832
Epoch 75/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 3.0234 - mae: 1.2676 - val_loss: 6.0683 - val_mae: 2.1107
Epoch 76/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 16ms/step - loss: 3.8300 - mae: 1.4123 - val_loss: 5.6242 - val_mae: 2.0047
Epoch 77/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 3.5744 - mae: 1.3671 - val_loss: 5.6237 - val_mae: 2.0690
Epoch 78/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 18ms/step - loss: 4.2757 - mae: 1.5079 - val_loss: 5.3630 - val_mae: 1.9960
Epoch 79/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 2.9783 - mae: 1.2444 - val_loss: 5.8574 - val_mae: 2.0549
Epoch 80/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 2.9691 - mae: 1.3098 - val_loss: 5.8569 - val_mae: 2.0782
Epoch 81/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 3.1347 - mae: 1.3190 - val_loss: 5.6004 - val_mae: 2.0516
Epoch 82/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 21ms/step - loss: 3.0486 - mae: 1.3098 - val_loss: 5.7978 - val_mae: 2.0725
Epoch 83/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 3.5143 - mae: 1.3388 - val_loss: 6.4607 - val_mae: 2.1463
Epoch 84/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 3.2677 - mae: 1.3296 - val_loss: 7.1982 - val_mae: 2.2619
Epoch 85/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 2.7845 - mae: 1.2696 - val_loss: 6.3011 - val_mae: 2.1320
Epoch 86/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 3.2645 - mae: 1.3261 - val_loss: 5.9456 - val_mae: 2.0780
Epoch 87/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 3.3792 - mae: 1.3207 - val_loss: 6.5349 - val_mae: 2.1487
Epoch 88/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 3.0326 - mae: 1.3049 - val_loss: 7.2766 - val_mae: 2.3267
Epoch 89/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 2.7410 - mae: 1.2498 - val_loss: 6.5947 - val_mae: 2.1601
Epoch 90/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 12ms/step - loss: 2.6835 - mae: 1.2105 - val_loss: 7.8742 - val_mae: 2.3418
Epoch 91/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 2.8822 - mae: 1.2184 - val_loss: 6.2158 - val_mae: 2.1801
Epoch 92/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 2.8245 - mae: 1.2308 - val_loss: 5.7484 - val_mae: 2.0469
Epoch 93/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 3.0690 - mae: 1.2606 - val_loss: 5.7192 - val_mae: 2.0031
Epoch 94/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 12ms/step - loss: 2.4758 - mae: 1.1528 - val_loss: 5.7537 - val_mae: 2.0362
Epoch 95/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 14ms/step - loss: 2.8690 - mae: 1.2372 - val_loss: 6.3387 - val_mae: 2.1342
Epoch 96/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 16ms/step - loss: 2.4567 - mae: 1.1880 - val_loss: 5.9458 - val_mae: 2.1252
Epoch 97/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 3.1337 - mae: 1.2829 - val_loss: 6.4779 - val_mae: 2.1510
Epoch 98/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 17ms/step - loss: 2.2902 - mae: 1.1209 - val_loss: 6.4647 - val_mae: 2.1840
Epoch 99/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 13ms/step - loss: 2.4584 - mae: 1.1437 - val_loss: 6.4062 - val_mae: 2.1606
Epoch 100/100
[1m12/12[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 15ms/step - loss: 3.1502 - mae: 1.2778 - val_loss: 6.1874 - val_mae: 2.1853
```

In [34]:

```python
y_pred_nn = model.predict(X_test_scaled)  # Predicting house prices on test data
mse_nn, mae_nn = model.evaluate(X_test_scaled, y_test)  # Evaluating model performance
# Displaying Neural Network Evaluation Metrics
print("\nNeural Network Model Evaluation:")
print(f"Mean Squared Error: {mse_nn}")
print(f"Mean Absolute Error: {mae_nn}")
```

```
[1m4/4 [0m [32m━━━━━━━━━━━━━━━━━━━━ [0m [37m [0m [1m1s [0m 145ms/step
[1m4/4 [0m [32m━━━━━━━━━━━━━━━━━━━━ [0m [37m [0m [1m0s [0m 8ms/step - loss: 10.9387 - mae: 2.3844

Neural Network Model Evaluation:
Mean Squared Error: 9.77536678314209
Mean Absolute Error: 2.2946557998657227
```

In [35]:

```python
new_data = np.array([[0.1, 10.0, 5.0]])
# New input values: LSTAT=0.1, RM=10.0, PTRATIO=5.0
new_data_scaled = scaler.transform(new_data)
# Applying the same standardization as training data
# Predicting price using trained neural network model
prediction = model.predict(new_data_scaled)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
```

```
---------------------------------------------------------------------------ValueError                                Traceback (most recent call last)
      1 new_data = np.array([[0.1, 10.0, 5.0]])
      2 # New input values: LSTAT=0.1, RM=10.0, PTRATIO=5.0
----> 3 new_data_scaled = scaler.transform(new_data)
      4 # Applying the same standardization as training data
      5 # Predicting price using trained neural network model
      6 prediction = model.predict(new_data_scaled)
File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\_set_output.py:140, in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    145             *data_to_wrap[1:],
    146         )
File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_data.py:992, in StandardScaler.transform(self, X, copy)
    989 check_is_fitted(self)
    991 copy = copy if copy is not None else self.copy
--> 992 X = self._validate_data(
    993     X,
    994     reset=False,
    995     accept_sparse="csr",
    996     copy=copy,
    997     dtype=FLOAT_DTYPES,
    998     force_all_finite="allow-nan",
    999 )
   1001 if sparse.issparse(X):
   1002     if self.with_mean:
File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:588, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, **check_p
    585     out = X, y
    587 if not no_val_X and check_params.get("ensure_2d", True):
--> 588     self._check_n_features(X, reset=reset)
    590 return out
File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:389, in BaseEstimator._check_n_features(self, X, reset)
    386     return
    388 if n_features != self.n_features_in_:
--> 389     raise ValueError(
    390         f"X has {n_features} features, but {self.__class__.__name__} "
    391         f"is expecting {self.n_features_in_} features as input."
    392     )
ValueError: X has 3 features, but StandardScaler is expecting 13 features as input.
```

In [ ]:

```python
# Displaying the predicted house price
print("\nPredicted House Price:", prediction[0][0])
```

In [ ]:

```python

```