```
# Assignment 1: Boston House Price Prediction using Linear Regression
and Neural Network

# Step 1: Import Required Libraries
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

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

boston = pd.read_csv("boston_house_prices.csv")  # Reads CSV file into
a DataFr

boston.head(5)
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX
PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
18.7

        B  LSTAT  PRICE
0  396.90   4.98   24.0
1  396.90   9.14   21.6
2  392.83   4.03   34.7
3  394.63   2.94   33.4
4  396.90   5.33   36.2
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Correlation matrix
correlation_matrix = boston.corr()

# Display the correlation matrix
print(correlation_matrix)

# Visualize the correlation using a heatmap
plt.figure(figsize=(10, 8))
```

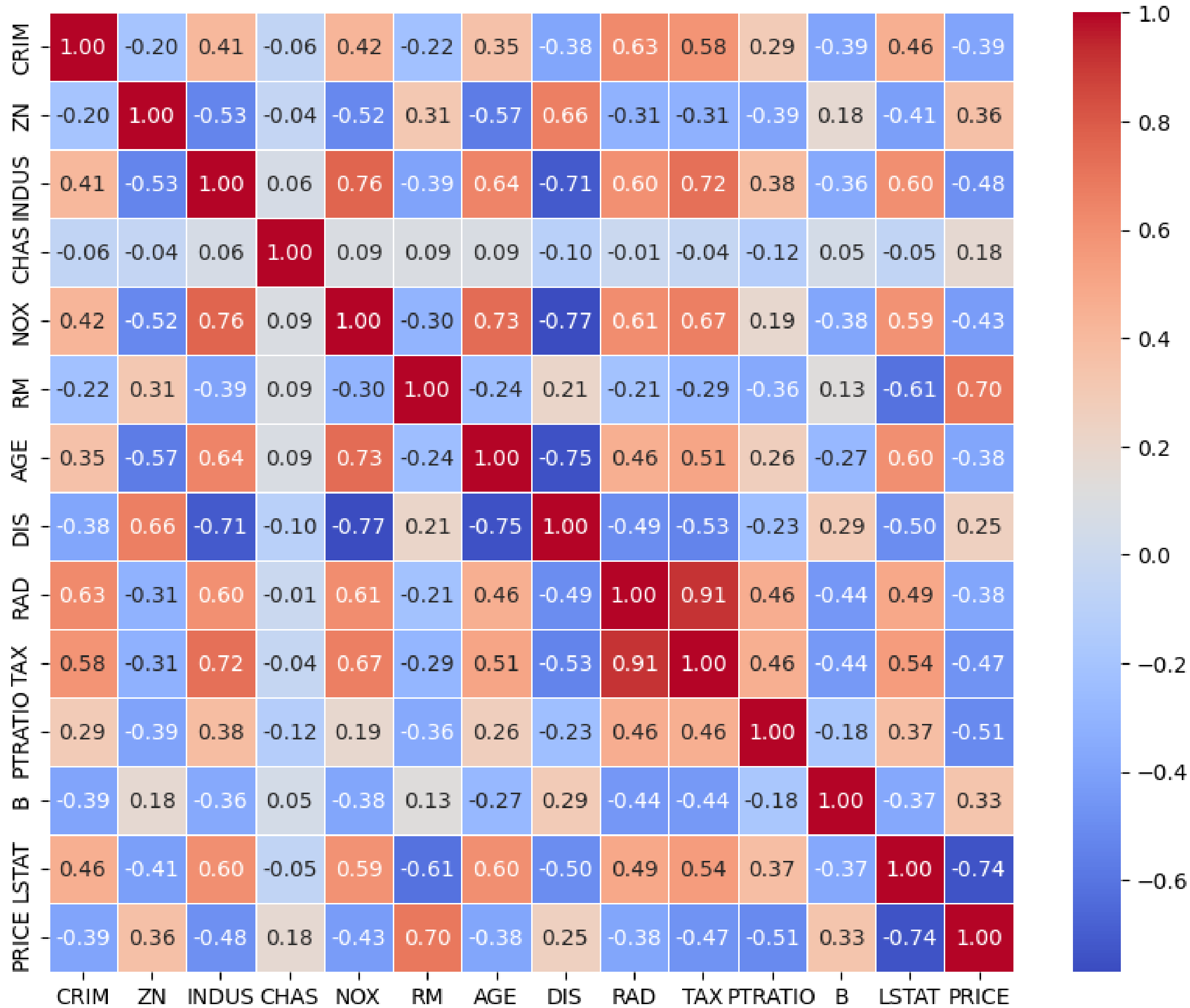
```python
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt='.2f', linewidths=0.5)
plt.show()

# Correlation of features with the target variable 'PRICE'
correlation_with_target =
correlation_matrix['PRICE'].sort_values(ascending=False)
print("\nCorrelation with target variable 'PRICE':")
print(correlation_with_target)
```

```
             CRIM        ZN     INDUS      CHAS       NOX        RM
AGE  \
CRIM     1.000000 -0.200469  0.406583 -0.055892  0.420972 -0.219247
0.352734
ZN      -0.200469  1.000000 -0.533828 -0.042697 -0.516604  0.311991 -
0.569537
INDUS    0.406583 -0.533828  1.000000  0.062938  0.763651 -0.391676
0.644779
CHAS    -0.055892 -0.042697  0.062938  1.000000  0.091203  0.091251
0.086518
NOX      0.420972 -0.516604  0.763651  0.091203  1.000000 -0.302188
0.731470
RM      -0.219247  0.311991 -0.391676  0.091251 -0.302188  1.000000 -
0.240265
AGE      0.352734 -0.569537  0.644779  0.086518  0.731470 -0.240265
1.000000
DIS     -0.379670  0.664408 -0.708027 -0.099176 -0.769230  0.205246 -
0.747881
RAD      0.625505 -0.311948  0.595129 -0.007368  0.611441 -0.209847
0.456022
TAX      0.582764 -0.314563  0.720760 -0.035587  0.668023 -0.292048
0.506456
PTRATIO  0.289946 -0.391679  0.383248 -0.121515  0.188933 -0.355501
0.261515
B       -0.385064  0.175520 -0.356977  0.048788 -0.380051  0.128069 -
0.273534
LSTAT    0.455621 -0.412995  0.603800 -0.053929  0.590879 -0.613808
0.602339
PRICE   -0.388305  0.360445 -0.483725  0.175260 -0.427321  0.695360 -
0.376955

              DIS       RAD       TAX   PTRATIO         B     LSTAT
PRICE
CRIM    -0.379670  0.625505  0.582764  0.289946 -0.385064  0.455621 -
0.388305
ZN       0.664408 -0.311948 -0.314563 -0.391679  0.175520 -0.412995
0.360445
INDUS   -0.708027  0.595129  0.720760  0.383248 -0.356977  0.603800 -
0.483725
CHAS    -0.099176 -0.007368 -0.035587 -0.121515  0.048788 -0.053929
```

```
0.175260
NOX     -0.769230  0.611441  0.668023  0.188933 -0.380051  0.590879 -
0.427321
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.613808
0.695360
AGE     -0.747881  0.456022  0.506456  0.261515 -0.273534  0.602339 -
0.376955
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.496996
0.249929
RAD     -0.494588  1.000000  0.910228  0.464741 -0.444413  0.488676 -
0.381626
TAX     -0.534432  0.910228  1.000000  0.460853 -0.441808  0.543993 -
0.468536
PTRATIO -0.232471  0.464741  0.460853  1.000000 -0.177383  0.374044 -
0.507787
B        0.291512 -0.444413 -0.441808 -0.177383  1.000000 -0.366087
0.333461
LSTAT   -0.496996  0.488676  0.543993  0.374044 -0.366087  1.000000 -
0.737663
PRICE    0.249929 -0.381626 -0.468536 -0.507787  0.333461 -0.737663
1.000000
```




```
Correlation with target variable 'PRICE':
PRICE      1.000000
RM         0.695360
ZN         0.360445
B          0.333461
DIS        0.249929
CHAS       0.175260
AGE       -0.376955
RAD       -0.381626
CRIM      -0.388305
NOX       -0.427321
TAX       -0.468536
INDUS     -0.483725
PTRATIO   -0.507787
LSTAT     -0.737663
Name: PRICE, dtype: float64
```

```python
X = boston[['LSTAT', 'RM', 'PTRATIO']]

# Target variable: House Price
y = boston['PRICE']

# LSTAT: Percentage of lower status population.
# RM: Average number of rooms per dwelling.
# PTRATIO: Pupil-teacher ratio by town.

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

scaler = StandardScaler()  # Initializing StandardScaler
X_train_scaled = scaler.fit_transform(X_train)  # Fit and transform
training data
X_test_scaled = scaler.transform(X_test)  # Transform test data using
the same scaler

# Linear Regression Model
lr_model = LinearRegression()  # Initializing Linear Regression Model
lr_model.fit(X_train_scaled, y_train)  # Training the model using
scaled training data
```

```
LinearRegression()
```

```python
# Predicting house prices on test data
y_pred_lr = lr_model.predict(X_test_scaled)

# Evaluating Linear Regression Model
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: 30.340105190234596
Mean Absolute Error: 3.5844321029226935
R2 Score: 0.6733732528519258
```

```python
# Neural Network (ANN) Model
# Creating a Deep Learning Model using Keras Sequential API
model = Sequential([
    Dense(128, activation='relu', input_dim=3),  # 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)
])
```

```
C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\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)
```

```
# Compiling the model
model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Training the Neural Network
history = model.fit(X_train_scaled, y_train, epochs=100,
validation_split=0.05, verbose=1)
```

```
Epoch 1/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 29ms/step - loss: 597.9725 - mae:
22.6633 - val_loss: 440.6859 - val_mae: 19.9912
Epoch 2/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 541.6343 - mae:
21.4712 - val_loss: 383.4800 - val_mae: 18.4633
Epoch 3/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 467.4316 - mae:
19.2414 - val_loss: 264.1100 - val_mae: 15.1874
Epoch 4/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 255.6833 - mae:
13.8029 - val_loss: 104.1311 - val_mae: 8.9425
Epoch 5/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 106.6450 - mae: 8.5953
- val_loss: 55.0628 - val_mae: 5.4499
Epoch 6/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 70.6950 - mae: 6.0300
- val_loss: 35.5970 - val_mae: 4.5327
Epoch 7/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 44.4458 - mae: 4.8824
- val_loss: 28.7060 - val_mae: 4.1877
Epoch 8/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 37.5696 - mae: 4.5097
- val_loss: 26.7047 - val_mae: 3.8280
Epoch 9/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 33.0975 - mae: 4.0658
- val_loss: 23.3646 - val_mae: 3.6589
```

```
Epoch 10/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 38.4042 - mae: 4.2916
- val_loss: 22.4100 - val_mae: 3.5545
Epoch 11/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 28.7516 - mae: 3.8734
- val_loss: 20.3089 - val_mae: 3.3905
Epoch 12/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 29.7010 - mae: 3.8923
- val_loss: 19.8083 - val_mae: 3.3503
Epoch 13/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 27.6363 - mae: 3.6715
- val_loss: 18.0534 - val_mae: 3.2142
Epoch 14/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 23.9688 - mae: 3.6526
- val_loss: 17.3083 - val_mae: 3.1416
Epoch 15/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 24.6179 - mae: 3.5567
- val_loss: 16.2637 - val_mae: 3.0573
Epoch 16/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 26.4587 - mae: 3.6649
- val_loss: 16.0502 - val_mae: 3.0069
Epoch 17/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 21.0812 - mae: 3.4753
- val_loss: 14.9949 - val_mae: 2.9223
Epoch 18/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 24.4015 - mae: 3.5063
- val_loss: 14.0938 - val_mae: 2.8334
Epoch 19/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 22.1011 - mae: 3.4099
- val_loss: 13.5602 - val_mae: 2.7907
Epoch 20/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 20.9976 - mae: 3.3606
- val_loss: 13.3791 - val_mae: 2.7748
Epoch 21/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 19.4215 - mae: 3.2039
- val_loss: 11.8821 - val_mae: 2.6630
Epoch 22/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 19.1406 - mae: 3.2789
- val_loss: 11.8897 - val_mae: 2.6224
Epoch 23/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 20.8035 - mae: 3.2380
- val_loss: 11.2603 - val_mae: 2.5930
Epoch 24/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 16.2081 - mae: 2.9831
- val_loss: 11.4103 - val_mae: 2.5892
Epoch 25/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 22.0481 - mae: 3.3207
- val_loss: 10.8912 - val_mae: 2.5622
Epoch 26/100
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 22.2064 - mae: 3.1996
- val_loss: 10.2625 - val_mae: 2.5161
Epoch 27/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 18.0137 - mae: 2.9900
- val_loss: 11.2624 - val_mae: 2.6179
Epoch 28/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 21.9304 - mae: 3.2641
- val_loss: 9.8150 - val_mae: 2.5036
Epoch 29/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 19.5846 - mae: 3.1073
- val_loss: 9.8451 - val_mae: 2.5149
Epoch 30/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 20.7555 - mae: 3.0404
- val_loss: 9.1940 - val_mae: 2.4471
Epoch 31/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 20.2241 - mae: 3.0854
- val_loss: 9.4996 - val_mae: 2.5005
Epoch 32/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 14.4073 - mae: 2.8050
- val_loss: 8.7855 - val_mae: 2.4337
Epoch 33/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 21.9740 - mae: 3.0280
- val_loss: 9.2492 - val_mae: 2.5326
Epoch 34/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 13.3788 - mae: 2.7552
- val_loss: 8.5439 - val_mae: 2.4440
Epoch 35/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 18.4987 - mae: 2.9350
- val_loss: 8.8434 - val_mae: 2.5187
Epoch 36/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 14.9023 - mae: 2.7798
- val_loss: 8.6618 - val_mae: 2.5093
Epoch 37/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 17.8659 - mae: 2.9114
- val_loss: 7.9708 - val_mae: 2.4255
Epoch 38/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 17.9758 - mae: 2.8209
- val_loss: 9.1961 - val_mae: 2.5992
Epoch 39/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 18.9060 - mae: 2.9617
- val_loss: 7.5738 - val_mae: 2.4068
Epoch 40/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 15.2602 - mae: 2.6205
- val_loss: 9.2810 - val_mae: 2.6321
Epoch 41/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 14.6993 - mae: 2.6541
- val_loss: 8.1925 - val_mae: 2.5260
Epoch 42/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 17.7217 - mae: 2.8256
```

```
- val_loss: 8.3578 - val_mae: 2.5621
Epoch 43/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 14.2149 - mae: 2.5322 -
val_loss: 7.7682 - val_mae: 2.4831
Epoch 44/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 19.6704 - mae: 2.8519
- val_loss: 8.4241 - val_mae: 2.5787
Epoch 45/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 13.8512 - mae: 2.6127
- val_loss: 7.7436 - val_mae: 2.4979
Epoch 46/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 14.3308 - mae: 2.5930
- val_loss: 8.7432 - val_mae: 2.6271
Epoch 47/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 17.7323 - mae: 2.8668 -
val_loss: 8.5786 - val_mae: 2.6171
Epoch 48/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 13.1757 - mae: 2.6023 -
val_loss: 8.0054 - val_mae: 2.5360
Epoch 49/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 14.3922 - mae: 2.6878 -
val_loss: 7.6504 - val_mae: 2.5019
Epoch 50/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.1037 - mae: 2.5659 -
val_loss: 7.5170 - val_mae: 2.4690
Epoch 51/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 18.1892 - mae: 2.8153 -
val_loss: 8.5603 - val_mae: 2.6418
Epoch 52/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 13.6342 - mae: 2.6081 -
val_loss: 7.7569 - val_mae: 2.5215
Epoch 53/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 16.7343 - mae: 2.6949 -
val_loss: 7.0923 - val_mae: 2.3990
Epoch 54/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 13.8767 - mae: 2.5624
- val_loss: 9.6079 - val_mae: 2.7623
Epoch 55/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 17.8159 - mae: 2.8933 -
val_loss: 6.7322 - val_mae: 2.2674
Epoch 56/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.3180 - mae: 2.5613 -
val_loss: 9.1954 - val_mae: 2.6965
Epoch 57/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 15.4266 - mae: 2.7325 -
val_loss: 7.6503 - val_mae: 2.5237
Epoch 58/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 18.5532 - mae: 2.7971 -
val_loss: 7.0635 - val_mae: 2.4063
```

```
Epoch 59/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 16.8370 - mae: 2.6426 - val_loss: 8.8016 - val_mae: 2.6591
Epoch 60/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 11.6949 - mae: 2.5019 - val_loss: 7.5473 - val_mae: 2.4860
Epoch 61/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 15.1934 - mae: 2.5892 - val_loss: 7.4563 - val_mae: 2.4764
Epoch 62/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 12.6252 - mae: 2.4406 - val_loss: 7.5284 - val_mae: 2.4881
Epoch 63/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 18.3914 - mae: 2.7708 - val_loss: 7.2988 - val_mae: 2.4506
Epoch 64/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.5509 - mae: 2.6321 - val_loss: 7.3331 - val_mae: 2.4637
Epoch 65/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.5423 - mae: 2.5150 - val_loss: 8.1694 - val_mae: 2.5676
Epoch 66/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 20.1376 - mae: 2.7959 - val_loss: 6.9854 - val_mae: 2.3949
Epoch 67/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 18.4924 - mae: 2.8891 - val_loss: 7.0062 - val_mae: 2.3710
Epoch 68/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 11.2424 - mae: 2.3803 - val_loss: 8.4711 - val_mae: 2.5843
Epoch 69/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.6220 - mae: 2.4766 - val_loss: 7.2923 - val_mae: 2.4210
Epoch 70/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 15.4218 - mae: 2.8238 - val_loss: 6.9128 - val_mae: 2.3555
Epoch 71/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.5296 - mae: 2.5444 - val_loss: 8.0191 - val_mae: 2.4936
Epoch 72/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 16.7625 - mae: 2.6814 - val_loss: 6.4497 - val_mae: 2.2542
Epoch 73/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 20.3179 - mae: 2.8438 - val_loss: 7.0953 - val_mae: 2.3801
Epoch 74/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.7447 - mae: 2.4665 - val_loss: 7.1953 - val_mae: 2.4042
Epoch 75/100
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 15.3910 - mae: 2.5894 -
val_loss: 7.3771 - val_mae: 2.4202
Epoch 76/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 20.5004 - mae: 2.8801 -
val_loss: 7.3425 - val_mae: 2.4527
Epoch 77/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 15.2336 - mae: 2.5878 -
val_loss: 7.0908 - val_mae: 2.3425
Epoch 78/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.3368 - mae: 2.6229 -
val_loss: 7.5718 - val_mae: 2.4836
Epoch 79/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 19.8562 - mae: 2.7653 -
val_loss: 7.2632 - val_mae: 2.4019
Epoch 80/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 17.1585 - mae: 2.6756 -
val_loss: 7.1828 - val_mae: 2.3735
Epoch 81/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 10.7822 - mae: 2.3513
- val_loss: 7.1186 - val_mae: 2.3882
Epoch 82/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 10.1218 - mae: 2.2328 -
val_loss: 6.8634 - val_mae: 2.2645
Epoch 83/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 17.2230 - mae: 2.7699 -
val_loss: 7.5084 - val_mae: 2.4459
Epoch 84/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.2912 - mae: 2.6867 -
val_loss: 6.9601 - val_mae: 2.3441
Epoch 85/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 15.1811 - mae: 2.6686 -
val_loss: 6.9161 - val_mae: 2.3426
Epoch 86/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 12.7038 - mae: 2.4260 -
val_loss: 8.6708 - val_mae: 2.5877
Epoch 87/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.6795 - mae: 2.6073 -
val_loss: 6.7757 - val_mae: 2.2749
Epoch 88/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 18.1123 - mae: 2.6755 -
val_loss: 6.4674 - val_mae: 2.2350
Epoch 89/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 12.2792 - mae: 2.4201 -
val_loss: 7.6323 - val_mae: 2.4614
Epoch 90/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.0987 - mae: 2.5773 -
val_loss: 8.2322 - val_mae: 2.4987
Epoch 91/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 12.2141 - mae: 2.4480 -
```

```
val_loss: 6.6622 - val_mae: 2.2255
Epoch 92/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 13.3556 - mae: 2.4611
- val_loss: 7.6917 - val_mae: 2.4074
Epoch 93/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.2239 - mae: 2.6453 -
val_loss: 6.9130 - val_mae: 2.3194
Epoch 94/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 13.9334 - mae: 2.5065 -
val_loss: 7.7664 - val_mae: 2.4517
Epoch 95/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 14.5630 - mae: 2.6031 -
val_loss: 7.7127 - val_mae: 2.4708
Epoch 96/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 18.9870 - mae: 2.7272
- val_loss: 7.1607 - val_mae: 2.3415
Epoch 97/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 13.2347 - mae: 2.5776 -
val_loss: 7.2873 - val_mae: 2.3944
Epoch 98/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 11.1549 - mae: 2.3638 -
val_loss: 7.1992 - val_mae: 2.3001
Epoch 99/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 11.9100 - mae: 2.3971 -
val_loss: 8.7416 - val_mae: 2.6033
Epoch 100/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 15.2612 - mae: 2.6028 -
val_loss: 7.1643 - val_mae: 2.2787
```

```python
# Evaluating the Neural Network Model
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
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 16.8934 - mae: 2.6039
```

```python
# Displaying Neural Network Evaluation Metrics
print("\nNeural Network Model Evaluation:")
print(f"Mean Squared Error: {mse_nn}")
print(f"Mean Absolute Error: {mae_nn}")
```

```
Neural Network Model Evaluation:
Mean Squared Error: 21.79648780822754
Mean Absolute Error: 2.816694736480713
```

```python
# House Price Prediction for New Data
new_data = np.array([[0.1, 10.0, 5.0]])
```

```
new_data_scaled = scaler.transform(new_data)
 # Applying the same standardization as training data
```

```
C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\
sklearn\utils\validation.py:2739: UserWarning: X does not have valid
feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

```
# Predicting price using trained neural network model
prediction = model.predict(new_data_scaled)

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step
```

```
Predicted House Price: 78.38197
```