Boston Housing Price Prediction

1. nhập thêm các thư viện cần thiết

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

- `pandas`: Để thao tác và phân tích dữ liệu.
- `numpy`: Đối với các phép toán số.
- `matplotlib`: Để trực quan hóa dữ liệu.
- `scikit-learn`:
  - `train_test_split`: Để chia dữ liệu thành các tập huấn luyện và kiểm tra.
  - `StandardScaler`: Để chuẩn hóa các tính năng.
  - `Regression models`: (Tuyến tính, Cây quyết định và Rừng ngẫu nhiên) và các chỉ số hiệu suất.
- `tensorflow`: Để tạo và huấn luyện mô hình Perceptron đa lớp (MLP).

2. Tải cơ sở dữ liệu nhà ở Bostont

- hiển thị thông tin và một số mẫu dữ liệu

```
data = pd.read_csv("BostonHousing.csv")
print(data.head())
print(data.info())
```

```
      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  medv
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
<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  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
None

#chia dữ liệu thành các tính năng và mục tiêu
X = data.drop(columns=["medv"])
y = data["medv"]
print("dataset chạy thành công")

print(data)

dataset chạy thành công
        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  medv
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]
```

- tập dữ liệu được tải xuống từ tệp file .CSV,chia dữ liệu thành các tính năng (`X`) và biến mục tiêu (`y`).
- note: `BostonHousing.csv` lưu ý nhớ coi file path trước khi chạy
- `X`: Independent variables (features).
- `y`: Dependent variable (target house price).

3. Chia dữ liệu thành các tập huấn luyện và kiểm tra

- Chúng tôi chia tập dữ liệu thành tập con huấn luyện (80%) và tập con kiểm tra (20%).

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

X_train.shape,y_train.shape

((404, 13), (404,))

X_test.shape,y_test.shape
```

```
((102, 13), (102,))
```

4. Chia tỷ lệ các tính năng

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- `StandardScaler`: các tính năng của thang đo có giá trị trung bình là 0 và độ lệch chuẩn là 1, cải thiện hiệu suất của mô hình.

5. Xây dựng và huấn luyện mô hình Perceptron đa lớp (MLP)

```
mlp_model = Sequential([
    Dense(32, activation='relu', input_dim=X_train.shape[1]),
    Dense(16, activation='relu'),
    Dense(1)
])
```

```
c:\Users\nguye\OneDrive\Desktop\bai 4\myvenv\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)
```

- **Lớp đầu vào**: Khớp với số lượng đối tượng trong tập dữ liệu.
- **Lớp ẩn**:

```
    - Lớp 1: 32 nơ-ron, kích hoạt ReLU.

    - Lớp 2: 16 nơ-ron, kích hoạt ReLU.
```

- **Lớp đầu ra**: 1 nơ-ron, kích hoạt tuyến tính (đối với các tác vụ hồi quy).

```
mlp_model.compile(optimizer='adam', loss='mean_squared_error',
metrics=['mae'])
```

- **Optimizer**: Adam (adaptive learning rate optimization).
- **Loss Function**: Mean Squared Error (MSE) for regression.
- **Metric**: Mean Absolute Error (MAE) to monitor performance during training.

```
history = mlp_model.fit(X_train_scaled, y_train, epochs=100,
batch_size=32, validation_data=(X_test_scaled, y_test))
```

```
Epoch 1/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 591.4785 - mae:
22.6219 - val_loss: 518.0832 - val_mae: 21.0749
Epoch 2/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 565.9119 - mae: 21.9997
- val_loss: 500.2781 - val_mae: 20.6557
Epoch 3/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 577.3720 - mae: 22.0793
- val_loss: 479.2048 - val_mae: 20.1422
Epoch 4/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 575.7136 - mae: 21.8891
- val_loss: 451.4072 - val_mae: 19.4446
Epoch 5/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 496.2930 - mae: 20.2373
- val_loss: 416.3414 - val_mae: 18.5309
Epoch 6/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 469.8231 - mae: 19.5559
- val_loss: 373.8247 - val_mae: 17.3859
Epoch 7/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 419.5841 - mae: 18.1760
- val_loss: 323.0823 - val_mae: 16.0044
Epoch 8/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 346.7157 - mae: 16.3279
- val_loss: 263.3647 - val_mae: 14.2482
Epoch 9/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 332.8604 - mae: 15.5809
- val_loss: 198.8700 - val_mae: 12.2463
Epoch 10/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 226.3770 - mae: 12.6517
- val_loss: 141.5460 - val_mae: 10.1347
Epoch 11/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 179.0722 - mae: 10.7820
- val_loss: 98.8399 - val_mae: 8.2668
Epoch 12/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 104.7759 - mae: 8.3402
- val_loss: 73.3067 - val_mae: 6.7907
Epoch 13/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 99.4608 - mae: 7.6946 -
val_loss: 58.7786 - val_mae: 5.8193
Epoch 14/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 81.9611 - mae: 7.0547 -
val_loss: 50.4766 - val_mae: 5.2246
Epoch 15/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 63.2249 - mae: 6.2252 -
val_loss: 44.8991 - val_mae: 4.8272
Epoch 16/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 58.1182 - mae: 5.7958 -
val_loss: 40.9281 - val_mae: 4.4934
Epoch 17/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 46.9240 - mae: 4.9943 -
```

```
val_loss: 37.9523 - val_mae: 4.2524
Epoch 18/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 42.0775 - mae: 4.7262 -
val_loss: 35.6921 - val_mae: 4.0729
Epoch 19/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 46.8618 - mae: 4.9007 -
val_loss: 34.1415 - val_mae: 3.9221
Epoch 20/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 33.1627 - mae: 4.1192 -
val_loss: 32.9539 - val_mae: 3.8070
Epoch 21/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 35.9471 - mae: 4.3090 -
val_loss: 32.0150 - val_mae: 3.7091
Epoch 22/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 31.0078 - mae: 4.1474 -
val_loss: 31.1832 - val_mae: 3.6380
Epoch 23/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 30.3726 - mae: 4.0028 -
val_loss: 30.4498 - val_mae: 3.5897
Epoch 24/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 31.6277 - mae: 3.9647 -
val_loss: 29.7636 - val_mae: 3.5465
Epoch 25/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 23.4291 - mae: 3.6134 -
val_loss: 29.1054 - val_mae: 3.4924
Epoch 26/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 30.0326 - mae: 3.8675 -
val_loss: 28.4315 - val_mae: 3.4506
Epoch 27/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 22.5733 - mae: 3.6076 -
val_loss: 28.0710 - val_mae: 3.4209
Epoch 28/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 26.8308 - mae: 3.8329 -
val_loss: 27.4058 - val_mae: 3.3899
Epoch 29/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 24.9928 - mae: 3.6685 -
val_loss: 26.9467 - val_mae: 3.3555
Epoch 30/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 25.8094 - mae: 3.6720 -
val_loss: 26.4410 - val_mae: 3.3232
Epoch 31/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 22.0818 - mae: 3.4619 -
val_loss: 26.0338 - val_mae: 3.2945
Epoch 32/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 28.2701 - mae: 3.6957 -
val_loss: 25.5140 - val_mae: 3.2549
Epoch 33/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 18.9402 - mae: 3.3710 -
val_loss: 25.1701 - val_mae: 3.2170
Epoch 34/100
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 23.0256 - mae: 3.4693 -
val_loss: 24.6666 - val_mae: 3.1768
Epoch 35/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 21.5214 - mae: 3.4895 -
val_loss: 24.3322 - val_mae: 3.1509
Epoch 36/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 22.4533 - mae: 3.5124 -
val_loss: 23.9990 - val_mae: 3.1224
Epoch 37/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 28.1240 - mae: 3.6482 -
val_loss: 23.6151 - val_mae: 3.1007
Epoch 38/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 21.5280 - mae: 3.3991 -
val_loss: 23.2533 - val_mae: 3.0657
Epoch 39/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 20.2142 - mae: 3.2412 -
val_loss: 22.9570 - val_mae: 3.0317
Epoch 40/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 20.5816 - mae: 3.3227 -
val_loss: 22.5832 - val_mae: 3.0092
Epoch 41/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 17.9459 - mae: 3.1357 -
val_loss: 22.4039 - val_mae: 2.9918
Epoch 42/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 23.3880 - mae: 3.3493 -
val_loss: 21.9680 - val_mae: 2.9560
Epoch 43/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 17.5555 - mae: 3.1315 -
val_loss: 21.6357 - val_mae: 2.9341
Epoch 44/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 16.4389 - mae: 3.0776 -
val_loss: 21.4176 - val_mae: 2.9031
Epoch 45/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 16.5153 - mae: 3.0046 -
val_loss: 21.2085 - val_mae: 2.8803
Epoch 46/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 15.0295 - mae: 2.8985 -
val_loss: 20.8324 - val_mae: 2.8604
Epoch 47/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 15.9365 - mae: 2.9306 -
val_loss: 20.5204 - val_mae: 2.8455
Epoch 48/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 17.2245 - mae: 3.0895 -
val_loss: 20.3418 - val_mae: 2.8251
Epoch 49/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 20.6628 - mae: 3.2036 -
val_loss: 20.0703 - val_mae: 2.7983
Epoch 50/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 17.0527 - mae: 2.9671 -
val_loss: 19.9336 - val_mae: 2.7697
```

```
Epoch 51/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 17.5937 - mae: 3.0976 -
val_loss: 19.5999 - val_mae: 2.7557
Epoch 52/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.6201 - mae: 2.8222 -
val_loss: 19.5453 - val_mae: 2.7303
Epoch 53/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 15.0030 - mae: 2.8959 -
val_loss: 19.2646 - val_mae: 2.7117
Epoch 54/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 15.7112 - mae: 2.9823 -
val_loss: 19.1171 - val_mae: 2.6992
Epoch 55/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 14.2441 - mae: 2.7621 -
val_loss: 18.8794 - val_mae: 2.6873
Epoch 56/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 16.6327 - mae: 2.9377 -
val_loss: 18.6485 - val_mae: 2.6723
Epoch 57/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.2746 - mae: 2.6572 -
val_loss: 18.4831 - val_mae: 2.6505
Epoch 58/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.0213 - mae: 2.5149 -
val_loss: 18.3364 - val_mae: 2.6226
Epoch 59/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.9358 - mae: 2.6397 -
val_loss: 18.1347 - val_mae: 2.6076
Epoch 60/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.6109 - mae: 2.6640 -
val_loss: 17.9849 - val_mae: 2.6070
Epoch 61/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.8071 - mae: 2.5958 -
val_loss: 17.8783 - val_mae: 2.5918
Epoch 62/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 15.2166 - mae: 2.8356 -
val_loss: 17.7531 - val_mae: 2.5812
Epoch 63/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.3902 - mae: 2.5852 -
val_loss: 17.5439 - val_mae: 2.5663
Epoch 64/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.3890 - mae: 2.5555 -
val_loss: 17.3698 - val_mae: 2.5554
Epoch 65/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.8578 - mae: 2.4872 -
val_loss: 17.2599 - val_mae: 2.5601
Epoch 66/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.0088 - mae: 2.5881 -
val_loss: 17.1295 - val_mae: 2.5415
Epoch 67/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 11.3489 - mae: 2.4578 -
```

```
val_loss: 17.0413 - val_mae: 2.5630
Epoch 68/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 14.2187 - mae: 2.7849 -
val_loss: 16.9405 - val_mae: 2.5409
Epoch 69/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.9942 - mae: 2.7468 -
val_loss: 16.7998 - val_mae: 2.5315
Epoch 70/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.9783 - mae: 2.7040 -
val_loss: 16.6794 - val_mae: 2.5235
Epoch 71/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 16.7116 - mae: 2.7602 -
val_loss: 16.6119 - val_mae: 2.5271
Epoch 72/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.2059 - mae: 2.6792 -
val_loss: 16.5347 - val_mae: 2.5383
Epoch 73/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.5971 - mae: 2.5975 -
val_loss: 16.4065 - val_mae: 2.5278
Epoch 74/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 14.4344 - mae: 2.7134 -
val_loss: 16.2947 - val_mae: 2.5065
Epoch 75/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 14.9125 - mae: 2.7022 -
val_loss: 16.2274 - val_mae: 2.5236
Epoch 76/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.8824 - mae: 2.4583 -
val_loss: 16.1280 - val_mae: 2.5161
Epoch 77/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.2429 - mae: 2.6204 -
val_loss: 16.0839 - val_mae: 2.5191
Epoch 78/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.7576 - mae: 2.6145 -
val_loss: 16.0321 - val_mae: 2.5193
Epoch 79/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 11.3173 - mae: 2.4924 -
val_loss: 15.9496 - val_mae: 2.5367
Epoch 80/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.3892 - mae: 2.4529 -
val_loss: 15.9253 - val_mae: 2.5446
Epoch 81/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 11.3074 - mae: 2.5418 -
val_loss: 15.8180 - val_mae: 2.5147
Epoch 82/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.0308 - mae: 2.3458 -
val_loss: 15.7340 - val_mae: 2.5293
Epoch 83/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 9.7760 - mae: 2.3592 -
val_loss: 15.6386 - val_mae: 2.5299
Epoch 84/100
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.1222 - mae: 2.5567 -
val_loss: 15.5581 - val_mae: 2.5347
Epoch 85/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.4539 - mae: 2.4802 -
val_loss: 15.5756 - val_mae: 2.5406
Epoch 86/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 9.9943 - mae: 2.2155 -
val_loss: 15.4744 - val_mae: 2.5307
Epoch 87/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 9.2658 - mae: 2.2810 -
val_loss: 15.3954 - val_mae: 2.5204
Epoch 88/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 11.3266 - mae: 2.3585 -
val_loss: 15.3554 - val_mae: 2.5312
Epoch 89/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 13.7288 - mae: 2.5227 -
val_loss: 15.2479 - val_mae: 2.5206
Epoch 90/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 11.9349 - mae: 2.4813 -
val_loss: 15.1939 - val_mae: 2.5331
Epoch 91/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 9.7924 - mae: 2.3456 -
val_loss: 15.0640 - val_mae: 2.5114
Epoch 92/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.0083 - mae: 2.2880 -
val_loss: 14.9861 - val_mae: 2.4936
Epoch 93/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 9.8376 - mae: 2.2408 -
val_loss: 14.9226 - val_mae: 2.5101
Epoch 94/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.4695 - mae: 2.3152 -
val_loss: 14.8522 - val_mae: 2.5046
Epoch 95/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.4659 - mae: 2.3524 -
val_loss: 14.7514 - val_mae: 2.5118
Epoch 96/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 8.7496 - mae: 2.2132 -
val_loss: 14.6933 - val_mae: 2.5065
Epoch 97/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.4339 - mae: 2.2594 -
val_loss: 14.6371 - val_mae: 2.5074
Epoch 98/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 9.2555 - mae: 2.2432 -
val_loss: 14.5216 - val_mae: 2.4861
Epoch 99/100
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.0249 - mae: 2.3572 -
val_loss: 14.4547 - val_mae: 2.4961
Epoch 100/100
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.1974 - mae: 2.4154 -
val_loss: 14.4742 - val_mae: 2.4961
```

- **Epochs**: Số lần đi qua dữ liệu huấn luyện.
- **Batch Size**: Số lượng mẫu được xử lý trước khi cập nhật mô hình.
- **Validation Data**: Theo dõi hiệu suất trên tập kiểm tra trong quá trình đào tạo.

6. Đánh giá mô hình MLP

- Chúng tôi đánh giá MLP đã được đào tạo trên tập kiểm tra và trực quan hóa các dự đoán.

```
test_loss, test_mae = mlp_model.evaluate(X_test_scaled, y_test)
print(f'Test Loss (MSE): {test_loss}')
print(f'Test Mean Absolute Error (MAE): {test_mae}')

4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 10.4216 - mae: 2.3513
Test Loss (MSE): 14.47421932220459
Test Mean Absolute Error (MAE): 2.4960906505584717
```

- Đánh giá mô hình MLP bằng MSE và MAE trên dữ liệu thử nghiệm

```
y_pred_mlp = mlp_model.predict(X_test_scaled) #prediction

4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
```

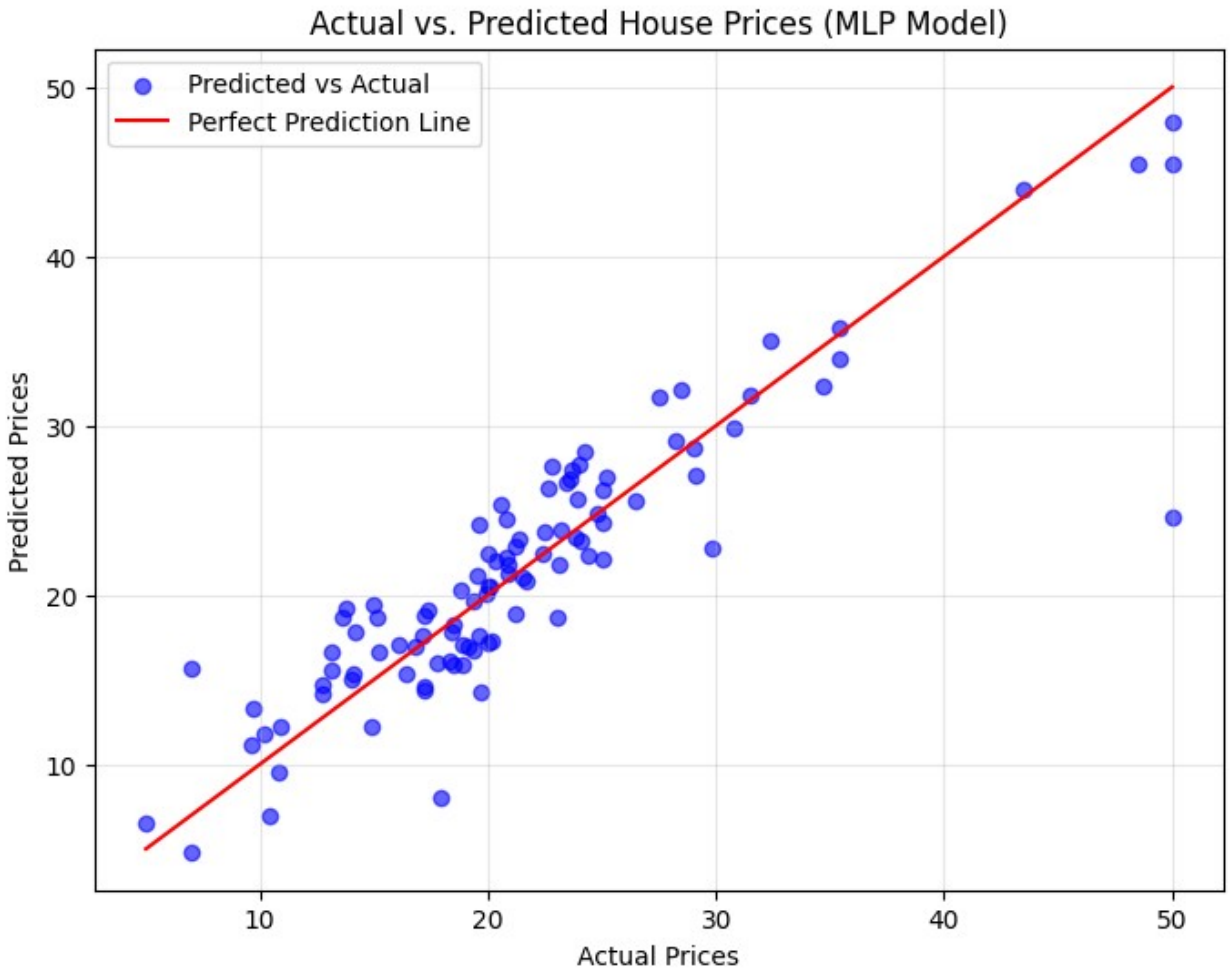
- **in kết quả**
- In ra MSE và $R2$ thể hiện mức độ tốt của mô hình MLP

```
y_pred_mlp = y_pred_mlp.flatten()#maˇng 1D

#tính toán các chiˇ sôˊ
mse_mlp = mean_squared_error(y_test, y_pred_mlp)
r2_mlp = r2_score(y_test, y_pred_mlp)

#in ra kêˊt quaˇ
print("-MLP Model Evaluation:")
print(f"-Mean Squared Error (MSE): {mse_mlp:.2f}")
print(f"-R² Score: {r2_mlp:.2f}")

-MLP Model Evaluation:
-Mean Squared Error (MSE): 14.47
-R² Score: 0.80
```

- `y_pred_mlp`: Dự đoán của mô hình MLP

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred_mlp, color='blue', alpha=0.6,
label='Predicted vs Actual')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
```

```
color='red', label='Perfect Prediction Line')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs. Predicted House Prices (MLP Model)')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```



- Vẽ biểu đồ phân tán để so sánh giá nhà thực tế (`Actual Prices`) và dự đoán (`Predicted Prices`).

7. So sánh với các mô hình hồi quy khác

- Chúng tôi so sánh mô hình MLP với hồi quy tuyến tính(`Linear Regression`), hồi quy cây quyết định(`Decision Tree`) và hồi quy rừng ngẫu nhiên(`Random Forest`).

7.1 Linear Regression

```python
#Linear Regression
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
y_pred_lr = lr.predict(X_test_scaled)
print("predicted value:")
print(y_pred_lr)

mse_lr =mean_squared_error(y_test,y_pred_lr)
r2_lr = r2_score(y_test,y_pred_lr)
print(f"Linear Regression")
print(f"- MSE: {mse_lr:.4f}")
print(f"-R²: {r2_lr:.4f}")
```

```
predicted value:
[28.99672362 36.02556534 14.81694405 25.03197915 18.76987992
23.25442929
 17.66253818 14.34119    23.01320703 20.63245597 24.90850512
18.63883645
 -6.08842184 21.75834668 19.23922576 26.19319733 20.64773313
5.79472718
 40.50033966 17.61289074 27.24909479 30.06625441 11.34179277
24.16077616
 17.86058499 15.83609765 22.78148106 14.57704449 22.43626052
19.19631835
 22.43383455 25.21979081 25.93909562 17.70162434 16.76911711
16.95125411
 31.23340153 20.13246729 23.76579011 24.6322925  13.94204955
32.25576301
 42.67251161 17.32745046 27.27618614 16.99310991 14.07009109
25.90341861
 20.29485982 29.95339638 21.28860173 34.34451856 16.04739105
26.22562412
 39.53939798 22.57950697 18.84531367 32.72531661 25.0673037
12.88628956
 22.68221908 30.48287757 31.52626806 15.90148607 20.22094826
16.71089812
 20.52384893 25.96356264 30.61607978 11.59783023 20.51232627
27.48111878
 11.01962332 15.68096344 23.79316251  6.19929359 21.6039073
41.41377225
 18.76548695  8.87931901 20.83076916 13.25620627 20.73963699
9.36482222
 23.22444271 31.9155003  19.10228271 25.51579303 29.04256769
20.14358566
 25.5859787   5.70159447 20.09474756 14.95069156 12.50395648
20.72635294
 24.73957161 -0.164237   13.68486682 16.18359697 22.27621999
24.47902364]
Linear Regression
```

```
- MSE: 24.2911
-R²: 0.6688
```

7.2 Decision Tree

```
#Decision Tree Regression
dt = DecisionTreeRegressor(random_state=42)
dt.fit(X_train_scaled, y_train)
y_pred_dt = dt.predict(X_test_scaled)
print("predicted value:")
print(y_pred_dt)

mse_dt =mean_squared_error(y_test,y_pred_dt)
r2_dt = r2_score(y_test,y_pred_dt)

print(f"Decision Tree Regression")
print(f"- MSE: {mse_dt:.4f}")
print(f"-R²: {r2_dt:.4f}")
```

```
predicted value:
[28.1 33.1 17.3 22.  23.2 18.5 16.6 16.6 22.7 22.  20.5 27.1  8.4 21.4
 18.5 23.9 18.8 10.5 46.  13.  23.1 24.4 13.6 22.  14.5 11.7 21.  13.5
 19.4 20.7 18.8 23.1 10.4 16.2 13.3 13.1 33.4 18.5 20.4 24.8 19.8 28.4
 46.  19.3 22.  13.  14.9 24.1 17.7 32.  21.7 36.1 16.7 28.4 43.1 18.5
 15.2 22.8 22.  22.5 24.5 33.  29.4 19.3 26.6 14.4 13.  22.9 22.8 14.1
 21.8 28.7  8.3 18.6 21.5 10.5 19.8 50.  13.3  8.1 21.2 16.3 19.4 10.5
 14.5 29.9 14.8 23.1 22.9 18.  23.3  8.8 19.2 17.6 16.2 19.3 50.  16.3
 11.7 16.3 19.  26.4]
Decision Tree Regression
- MSE: 10.4161
-R²: 0.8580
```

7.3 Random Forest Regression

```
#Random Forest Regression
rf = RandomForestRegressor(random_state=42, n_estimators=100)
rf.fit(X_train_scaled, y_train)
y_pred_rf = rf.predict(X_test_scaled)
print("predicted value:")
print(y_pred_rf)

mse_rf =mean_squared_error(y_test,y_pred_rf)
r2_rf = r2_score(y_test,y_pred_rf)

print(f"Random Forest Regression")
print(f"-MSE: {mse_rf:.4f}")
print(f"-R²: {r2_rf:.4f}")
```

```
predicted value:
[22.839 30.689 16.317 23.51  16.819 21.425 19.358 15.62  21.091 21.073
 20.028 19.298  8.611 21.456 19.378 25.453 19.187  8.538 46.132 14.536
 24.728 23.996 14.509 23.847 14.363 14.796 21.121 13.663 19.535 21.29
 19.45  23.392 29.3   20.338 14.596 15.594 33.835 19.129 20.915 24.376
 19.286 29.61  46.108 19.428 22.653 13.676 15.037 24.321 18.689 28.821
 21.107 33.823 16.502 25.763 44.922 21.994 15.416 32.032 22.596 20.296
 25.597 33.928 28.134 18.551 26.745 17.568 13.992 23.195 29.022 15.663
 21.064 27.426 10.06  21.569 21.956  7.084 19.905 46.154 11.274 12.981
 21.288 12.501 19.579  9.392 20.76  27.283 15.383 23.398 23.628 17.617
 21.681  8.019 19.616 18.714 22.592 19.786 41.733 12.726 12.632 13.066
 20.603 23.902]
Random Forest Regression
-MSE: 7.9127
-R²: 0.8921
```

- `Mean Squared Error`(MSE): Đo chênh lệch bình phương trung bình giữa giá trị thực tế và giá trị dự đoán.
- `R-squared`: Cho biết mô hình giải thích phương sai của biến mục tiêu tốt như thế nào.
- So sánh MLP với các mô hình truyền thống (Hồi quy tuyến tính`(Linear Regression)`, Cây quyết định`(Decision Tree)`, Rừng ngẫu nhiên`(Random Forest)`)

```python
models = {
    "MLP": y_pred_mlp.flatten(),
    "Linear Regression": y_pred_lr,
    "Decision Tree": y_pred_dt,
    "Random Forest": y_pred_rf
}

for model_name, y_pred in models.items():
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"{model_name} -> MSE: {mse:.2f}, r2-squared: {r2:.2f}")
```

```
MLP -> MSE: 14.47, r2-squared: 0.80
Linear Regression -> MSE: 24.29, r2-squared: 0.67
Decision Tree -> MSE: 10.42, r2-squared: 0.86
Random Forest -> MSE: 7.91, r2-squared: 0.89
```

source code github: https://github.com/HUyEsona/Practice-exercise-4-Multilayer-Perceptron-Regression-Exercise-Predicting-House-Prices.git