Neural Network: Regression using MLP in Keras

1. Data Loading and Preparation

Import required dependencies

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

import warnings
warnings.filterwarnings('ignore')
```

Load California Housing dataset from TensorFlow API. Use 60:20:20 ratio for the segmentation of training, validation, and testing datasets

```
(X_train_full, y_train_full), (X_test, y_test) =
tf.keras.datasets.california_housing.load_data(
    path='california_housing.npz',
    test_split=0.2,
    seed=113
)

from sklearn.model_selection import train_test_split

# 0.25 * 0.8 = 0.2
X_train, X_valid, y_train, y_valid = train_test_split(X_train_full,
y_train_full, test_size=0.25)

print("Training shape:", X_train.shape, y_train.shape)
print("Validation shape:", X_valid.shape, y_valid.shape)
print("Testing shape:", X_test.shape, y_test.shape)
```

```
Training shape: (12384, 8) (12384,)
Validation shape: (4128, 8) (4128,)
Testing shape: (4128, 8) (4128,)
```

2. Data Preprocessing

From previous assignment, the median_housing_value (our target) contains outlier at value more than 500,000. It is observed that the dataset clipped any houses with price more than 500,000 (threshold). Thus, if we plot the histogram, we will see a huge surge when the median_housing_value is more than 500,000.

Let's remove this outliers.

```
X_train = X_train[y_train < 500000]
y_train = y_train[y_train < 500000]
X_test = X_test[y_test < 500000]
y_test = y_test[y_test < 500000]
X_valid = X_valid[y_valid < 500000]
y_valid = y_valid[y_valid < 500000]
```

Standardize the features using StandardScaler.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)
```

Scale the target variable to ensure it falls within a specific range, which can help the model to learn more effectively.

```
y_train = scaler.fit_transform(y_train.reshape(-1, 1)).flatten()
y_valid = scaler.transform(y_valid.reshape(-1, 1)).flatten()
y_test = scaler.transform(y_test.reshape(-1, 1)).flatten()
```

3. Build Sequential Regression MLP Model

Model Architecture:

- Input Layer: Takes input of shape equal to the number of features (8).
- First Hidden Layer: Fully connected layer with 128 neurons and ReLU activation.
- Dropout Layer: Applies dropout to 20% of the neurons to prevent overfitting.
- Second Hidden Layer: Fully connected layer with 64 neurons and ReLU activation.
- Output Layer: Fully connected layer with 1 neuron, providing the final regression output.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(128, activation='relu',
input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dropout(0.2),  # Dropout for regularization
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
])
```

3. Compile Model

- Use MSE as loss function with Adam optimizer
- Evaluate model using MAE
- Additional: Learning rate of the adam optimizer is adjusted

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
loss='mean_squared_error', metrics=['mean_absolute_error'])
```

4. Model Training

```
history = model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_valid, y_valid))
```

```
Epoch 1/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.5337 -
mean_absolute_error: 0.5497 - val_loss: 0.3317 -
val_mean_absolute_error: 0.4237
Epoch 2/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 960us/step - loss: 0.3532 -
mean_absolute_error: 0.4398 - val_loss: 0.3108 -
val_mean_absolute_error: 0.3984
Epoch 3/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.3239 -
mean_absolute_error: 0.4156 - val_loss: 0.2990 -
val_mean_absolute_error: 0.3904
Epoch 4/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 965us/step - loss: 0.3043 -
mean_absolute_error: 0.3979 - val_loss: 0.3011 -
val_mean_absolute_error: 0.4007
Epoch 5/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.3091 -
mean_absolute_error: 0.4010 - val_loss: 0.2909 -
val_mean_absolute_error: 0.3732
Epoch 6/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2935 -
mean_absolute_error: 0.3921 - val_loss: 0.2887 -
val_mean_absolute_error: 0.3824
Epoch 7/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 981us/step - loss: 0.2795 -
mean_absolute_error: 0.3814 - val_loss: 0.2726 -
val_mean_absolute_error: 0.3670
Epoch 8/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 989us/step - loss: 0.2820 -
mean_absolute_error: 0.3823 - val_loss: 0.2712 -
val_mean_absolute_error: 0.3741
Epoch 9/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 998us/step - loss: 0.2685 -
mean_absolute_error: 0.3712 - val_loss: 0.2598 -
val_mean_absolute_error: 0.3508
Epoch 10/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 982us/step - loss: 0.2684 -
mean_absolute_error: 0.3710 - val_loss: 0.2628 -
val_mean_absolute_error: 0.3606
Epoch 11/100
```

```
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2691 -
mean_absolute_error: 0.3723 - val_loss: 0.2554 -
val_mean_absolute_error: 0.3506
Epoch 12/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2598 -
mean_absolute_error: 0.3652 - val_loss: 0.2535 -
val_mean_absolute_error: 0.3489
Epoch 13/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2577 -
mean_absolute_error: 0.3613 - val_loss: 0.2534 -
val_mean_absolute_error: 0.3446
Epoch 14/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2530 -
mean_absolute_error: 0.3572 - val_loss: 0.2559 -
val_mean_absolute_error: 0.3563
Epoch 15/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2541 -
mean_absolute_error: 0.3585 - val_loss: 0.2485 -
val_mean_absolute_error: 0.3429
Epoch 16/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2477 -
mean_absolute_error: 0.3542 - val_loss: 0.2518 -
val_mean_absolute_error: 0.3534
Epoch 17/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2572 -
mean_absolute_error: 0.3572 - val_loss: 0.2459 -
val_mean_absolute_error: 0.3400
Epoch 18/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2558 -
mean_absolute_error: 0.3582 - val_loss: 0.2455 -
val_mean_absolute_error: 0.3409
Epoch 19/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 981us/step - loss: 0.2491 -
mean_absolute_error: 0.3513 - val_loss: 0.2453 -
val_mean_absolute_error: 0.3427
Epoch 20/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2496 -
mean_absolute_error: 0.3535 - val_loss: 0.2438 -
val_mean_absolute_error: 0.3387
Epoch 21/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2405 -
mean_absolute_error: 0.3472 - val_loss: 0.2394 -
val_mean_absolute_error: 0.3417
Epoch 22/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2366 -
mean_absolute_error: 0.3442 - val_loss: 0.2474 -
val_mean_absolute_error: 0.3486
Epoch 23/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2284 -
```

```
mean_absolute_error: 0.3443 - val_loss: 0.2433 -
val_mean_absolute_error: 0.3402
Epoch 24/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2343 -
mean_absolute_error: 0.3433 - val_loss: 0.2446 -
val_mean_absolute_error: 0.3465
Epoch 25/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2484 -
mean_absolute_error: 0.3530 - val_loss: 0.2357 -
val_mean_absolute_error: 0.3365
Epoch 26/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 980us/step - loss: 0.2423 -
mean_absolute_error: 0.3477 - val_loss: 0.2406 -
val_mean_absolute_error: 0.3452
Epoch 27/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2292 -
mean_absolute_error: 0.3357 - val_loss: 0.2383 -
val_mean_absolute_error: 0.3458
Epoch 28/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 982us/step - loss: 0.2406 -
mean_absolute_error: 0.3450 - val_loss: 0.2342 -
val_mean_absolute_error: 0.3324
Epoch 29/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2399 -
mean_absolute_error: 0.3444 - val_loss: 0.2410 -
val_mean_absolute_error: 0.3357
Epoch 30/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2412 -
mean_absolute_error: 0.3474 - val_loss: 0.2472 -
val_mean_absolute_error: 0.3554
Epoch 31/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2325 -
mean_absolute_error: 0.3433 - val_loss: 0.2360 -
val_mean_absolute_error: 0.3307
Epoch 32/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2322 -
mean_absolute_error: 0.3397 - val_loss: 0.2346 -
val_mean_absolute_error: 0.3375
Epoch 33/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2249 -
mean_absolute_error: 0.3359 - val_loss: 0.2359 -
val_mean_absolute_error: 0.3366
Epoch 34/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2291 -
mean_absolute_error: 0.3394 - val_loss: 0.2313 -
val_mean_absolute_error: 0.3330
Epoch 35/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2298 -
mean_absolute_error: 0.3395 - val_loss: 0.2382 -
```

```
val_mean_absolute_error: 0.3366
Epoch 36/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2247 -
mean_absolute_error: 0.3327 - val_loss: 0.2355 -
val_mean_absolute_error: 0.3371
Epoch 37/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2211 -
mean_absolute_error: 0.3336 - val_loss: 0.2294 -
val_mean_absolute_error: 0.3316
Epoch 38/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2271 -
mean_absolute_error: 0.3370 - val_loss: 0.2277 -
val_mean_absolute_error: 0.3223
Epoch 39/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2266 -
mean_absolute_error: 0.3344 - val_loss: 0.2339 -
val_mean_absolute_error: 0.3335
Epoch 40/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2213 -
mean_absolute_error: 0.3302 - val_loss: 0.2313 -
val_mean_absolute_error: 0.3366
Epoch 41/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 993us/step - loss: 0.2148 -
mean_absolute_error: 0.3311 - val_loss: 0.2292 -
val_mean_absolute_error: 0.3325
Epoch 42/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2246 -
mean_absolute_error: 0.3336 - val_loss: 0.2300 -
val_mean_absolute_error: 0.3234
Epoch 43/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2318 -
mean_absolute_error: 0.3370 - val_loss: 0.2310 -
val_mean_absolute_error: 0.3266
Epoch 44/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2152 -
mean_absolute_error: 0.3285 - val_loss: 0.2289 -
val_mean_absolute_error: 0.3303
Epoch 45/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2279 -
mean_absolute_error: 0.3371 - val_loss: 0.2346 -
val_mean_absolute_error: 0.3381
Epoch 46/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2278 -
mean_absolute_error: 0.3350 - val_loss: 0.2318 -
val_mean_absolute_error: 0.3344
Epoch 47/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2213 -
mean_absolute_error: 0.3323 - val_loss: 0.2266 -
val_mean_absolute_error: 0.3241
```

```
Epoch 48/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2158 -
mean_absolute_error: 0.3277 - val_loss: 0.2283 -
val_mean_absolute_error: 0.3308
Epoch 49/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2141 -
mean_absolute_error: 0.3262 - val_loss: 0.2329 -
val_mean_absolute_error: 0.3278
Epoch 50/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2111 -
mean_absolute_error: 0.3232 - val_loss: 0.2318 -
val_mean_absolute_error: 0.3270
Epoch 51/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2248 -
mean_absolute_error: 0.3338 - val_loss: 0.2241 -
val_mean_absolute_error: 0.3219
Epoch 52/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2216 -
mean_absolute_error: 0.3308 - val_loss: 0.2328 -
val_mean_absolute_error: 0.3350
Epoch 53/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2095 -
mean_absolute_error: 0.3242 - val_loss: 0.2338 -
val_mean_absolute_error: 0.3376
Epoch 54/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2174 -
mean_absolute_error: 0.3312 - val_loss: 0.2268 -
val_mean_absolute_error: 0.3248
Epoch 55/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2177 -
mean_absolute_error: 0.3280 - val_loss: 0.2277 -
val_mean_absolute_error: 0.3268
Epoch 56/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2083 -
mean_absolute_error: 0.3239 - val_loss: 0.2325 -
val_mean_absolute_error: 0.3301
Epoch 57/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2176 -
mean_absolute_error: 0.3278 - val_loss: 0.2272 -
val_mean_absolute_error: 0.3246
Epoch 58/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2226 -
mean_absolute_error: 0.3303 - val_loss: 0.2299 -
val_mean_absolute_error: 0.3236
Epoch 59/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2166 -
mean_absolute_error: 0.3276 - val_loss: 0.2266 -
val_mean_absolute_error: 0.3327
Epoch 60/100
```

```
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2162 -
mean_absolute_error: 0.3271 - val_loss: 0.2271 -
val_mean_absolute_error: 0.3289
Epoch 61/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2053 -
mean_absolute_error: 0.3200 - val_loss: 0.2301 -
val_mean_absolute_error: 0.3231
Epoch 62/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2098 -
mean_absolute_error: 0.3229 - val_loss: 0.2377 -
val_mean_absolute_error: 0.3465
Epoch 63/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2282 -
mean_absolute_error: 0.3365 - val_loss: 0.2267 -
val_mean_absolute_error: 0.3237
Epoch 64/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2114 -
mean_absolute_error: 0.3216 - val_loss: 0.2256 -
val_mean_absolute_error: 0.3315
Epoch 65/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2183 -
mean_absolute_error: 0.3269 - val_loss: 0.2254 -
val_mean_absolute_error: 0.3205
Epoch 66/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2142 -
mean_absolute_error: 0.3258 - val_loss: 0.2283 -
val_mean_absolute_error: 0.3257
Epoch 67/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2084 -
mean_absolute_error: 0.3217 - val_loss: 0.2257 -
val_mean_absolute_error: 0.3187
Epoch 68/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2073 -
mean_absolute_error: 0.3199 - val_loss: 0.2240 -
val_mean_absolute_error: 0.3180
Epoch 69/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2128 -
mean_absolute_error: 0.3250 - val_loss: 0.2285 -
val_mean_absolute_error: 0.3256
Epoch 70/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2130 -
mean_absolute_error: 0.3249 - val_loss: 0.2270 -
val_mean_absolute_error: 0.3213
Epoch 71/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2127 -
mean_absolute_error: 0.3240 - val_loss: 0.2255 -
val_mean_absolute_error: 0.3188
Epoch 72/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2023 -
```

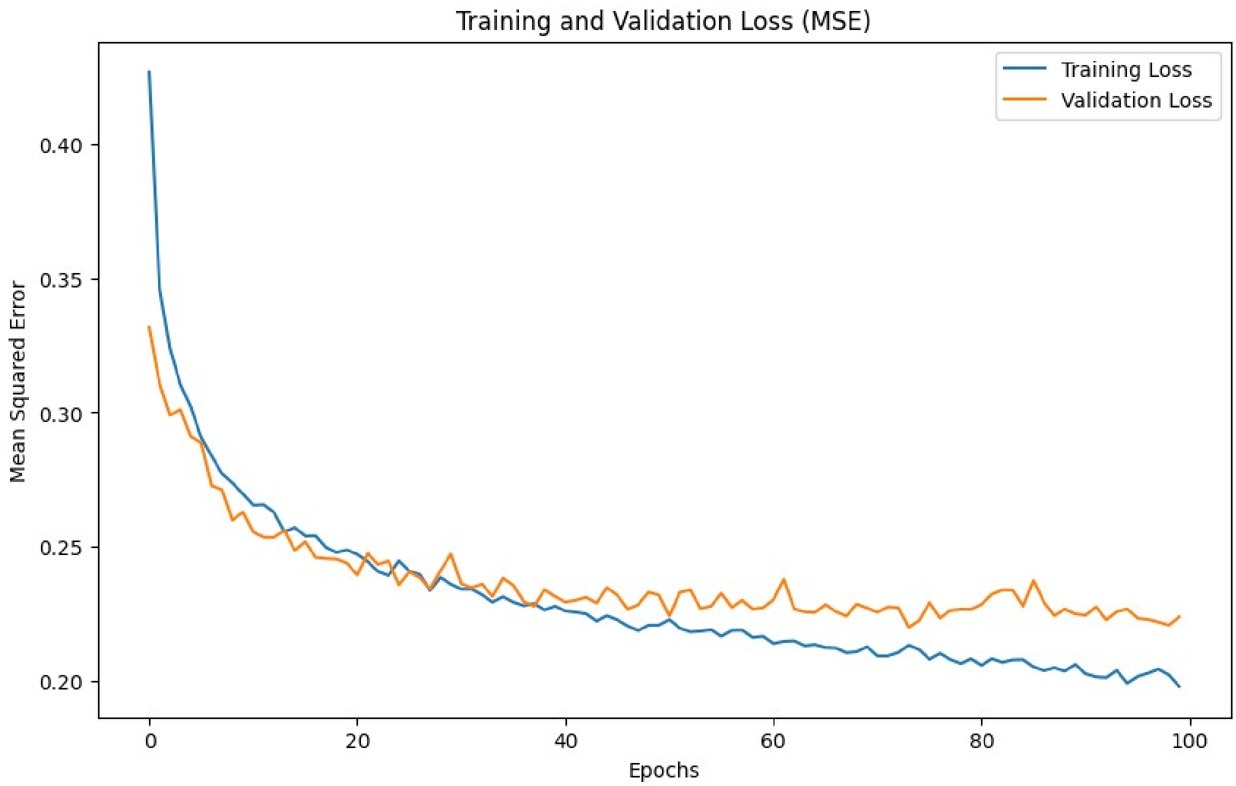
```
mean_absolute_error: 0.3163 - val_loss: 0.2274 -
val_mean_absolute_error: 0.3192
Epoch 73/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2071 -
mean_absolute_error: 0.3193 - val_loss: 0.2271 -
val_mean_absolute_error: 0.3275
Epoch 74/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2139 -
mean_absolute_error: 0.3253 - val_loss: 0.2198 -
val_mean_absolute_error: 0.3169
Epoch 75/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2183 -
mean_absolute_error: 0.3281 - val_loss: 0.2223 -
val_mean_absolute_error: 0.3219
Epoch 76/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2072 -
mean_absolute_error: 0.3192 - val_loss: 0.2291 -
val_mean_absolute_error: 0.3283
Epoch 77/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2083 -
mean_absolute_error: 0.3177 - val_loss: 0.2233 -
val_mean_absolute_error: 0.3174
Epoch 78/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2125 -
mean_absolute_error: 0.3211 - val_loss: 0.2261 -
val_mean_absolute_error: 0.3224
Epoch 79/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1989 -
mean_absolute_error: 0.3143 - val_loss: 0.2266 -
val_mean_absolute_error: 0.3201
Epoch 80/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2142 -
mean_absolute_error: 0.3219 - val_loss: 0.2265 -
val_mean_absolute_error: 0.3310
Epoch 81/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2110 -
mean_absolute_error: 0.3240 - val_loss: 0.2284 -
val_mean_absolute_error: 0.3315
Epoch 82/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2103 -
mean_absolute_error: 0.3221 - val_loss: 0.2322 -
val_mean_absolute_error: 0.3293
Epoch 83/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.2000 -
mean_absolute_error: 0.3143 - val_loss: 0.2338 -
val_mean_absolute_error: 0.3385
Epoch 84/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2034 -
mean_absolute_error: 0.3196 - val_loss: 0.2337 -
```

```
val_mean_absolute_error: 0.3224
Epoch 85/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2028 -
mean_absolute_error: 0.3204 - val_loss: 0.2277 -
val_mean_absolute_error: 0.3258
Epoch 86/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1974 -
mean_absolute_error: 0.3151 - val_loss: 0.2373 -
val_mean_absolute_error: 0.3410
Epoch 87/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1960 -
mean_absolute_error: 0.3138 - val_loss: 0.2292 -
val_mean_absolute_error: 0.3204
Epoch 88/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2029 -
mean_absolute_error: 0.3148 - val_loss: 0.2242 -
val_mean_absolute_error: 0.3181
Epoch 89/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2018 -
mean_absolute_error: 0.3156 - val_loss: 0.2266 -
val_mean_absolute_error: 0.3280
Epoch 90/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2000 -
mean_absolute_error: 0.3156 - val_loss: 0.2249 -
val_mean_absolute_error: 0.3210
Epoch 91/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1884 -
mean_absolute_error: 0.3071 - val_loss: 0.2243 -
val_mean_absolute_error: 0.3226
Epoch 92/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2035 -
mean_absolute_error: 0.3212 - val_loss: 0.2274 -
val_mean_absolute_error: 0.3282
Epoch 93/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1982 -
mean_absolute_error: 0.3155 - val_loss: 0.2226 -
val_mean_absolute_error: 0.3186
Epoch 94/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2045 -
mean_absolute_error: 0.3185 - val_loss: 0.2257 -
val_mean_absolute_error: 0.3245
Epoch 95/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1955 -
mean_absolute_error: 0.3154 - val_loss: 0.2267 -
val_mean_absolute_error: 0.3251
Epoch 96/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2055 -
mean_absolute_error: 0.3183 - val_loss: 0.2232 -
val_mean_absolute_error: 0.3238
```

```
Epoch 97/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2033 -
mean_absolute_error: 0.3175 - val_loss: 0.2227 -
val_mean_absolute_error: 0.3266
Epoch 98/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2014 -
mean_absolute_error: 0.3172 - val_loss: 0.2218 -
val_mean_absolute_error: 0.3221
Epoch 99/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1963 -
mean_absolute_error: 0.3102 - val_loss: 0.2206 -
val_mean_absolute_error: 0.3162
Epoch 100/100
369/369 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1940 -
mean_absolute_error: 0.3110 - val_loss: 0.2237 -
val_mean_absolute_error: 0.3183
```

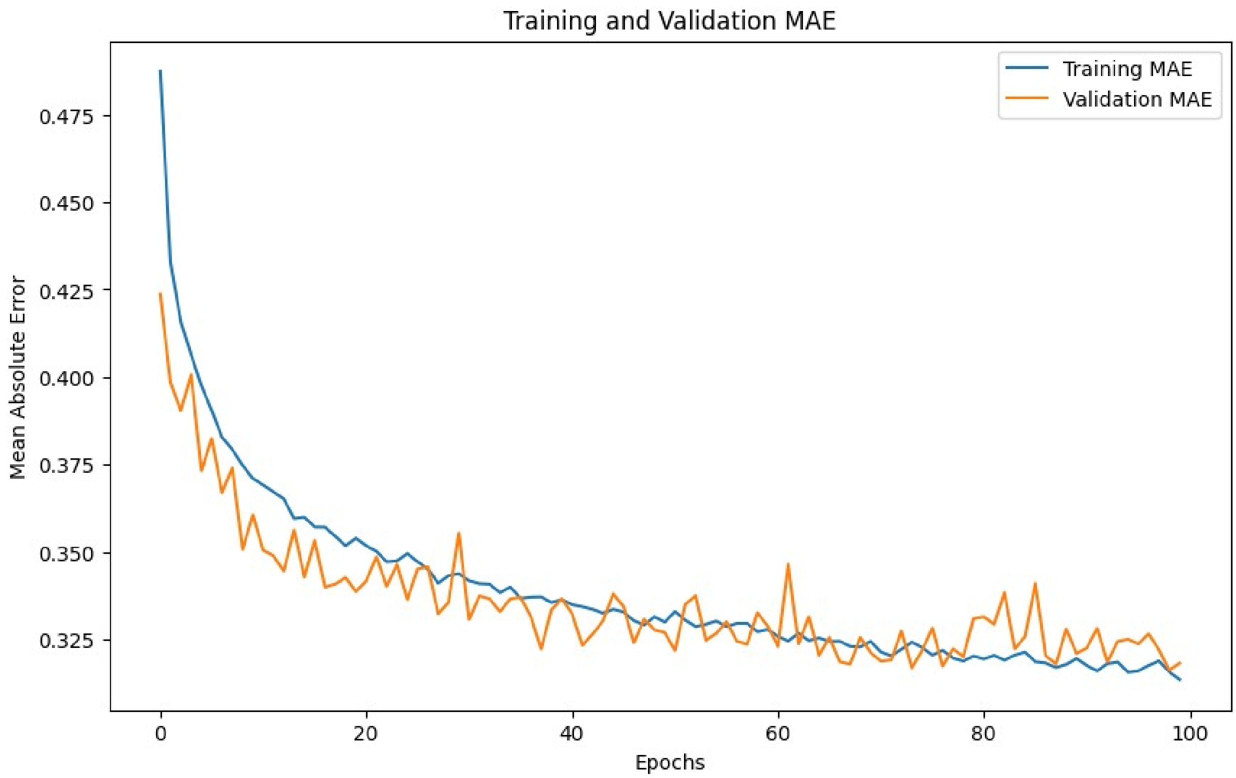
Let's plot the training history.

```
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Mean Squared Error')
plt.title('Training and Validation Loss (MSE)')
plt.legend()
plt.show()
```



Let's also plot how the MAE change during training.

```
plt.figure(figsize=(10, 6))
plt.plot(history.history['mean_absolute_error'], label='Training MAE')
plt.plot(history.history['val_mean_absolute_error'], label='Validation
MAE')
plt.xlabel('Epochs')
plt.ylabel('Mean Absolute Error')
plt.title('Training and Validation MAE')
plt.legend()
plt.show()
```



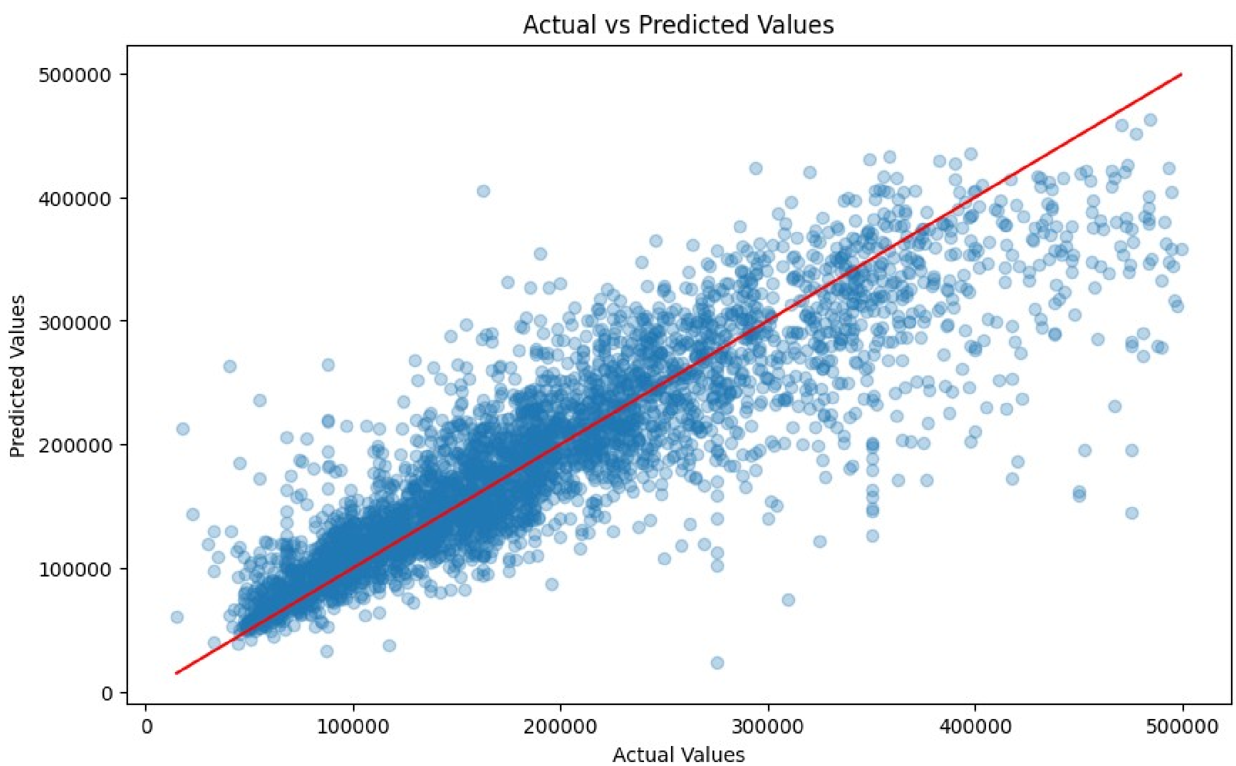
5. Model Evaluation

```
mse, mae = model.evaluate(X_test, y_test)
print(f"Mean Squared Error on test set: {mse}")
print(f"Mean Absolute Error on test set: {mae}")

123/123 ━━━━━━━━━━━━━━━━━━━━ 0s 772us/step - loss: 0.2306 -
mean_absolute_error: 0.3253
Mean Squared Error on test set: 0.2286626100540161
Mean Absolute Error on test set: 0.325112909078598
```

Oops, we need to revert back the scaling to get the true value of the predicted median_housing_value.

```
# Inverse transform the predictions and the true values
y_pred_scaled = model.predict(X_test)
y_pred = scaler.inverse_transform(y_pred_scaled).flatten()
y_test_original = scaler.inverse_transform(y_test.reshape(-1,
1)).flatten()

# Calculate the Mean Squared Error and Mean Absolute Error in the
original scale
mse_original = np.mean((y_pred - y_test_original) ** 2)
mae_original = np.mean(np.abs(y_pred - y_test_original))

print(f"Mean Squared Error on test set (original scale):
```

```
{mse_original}")
print(f"Mean Absolute Error on test set (original scale):
{mae_original}")

# Plot predictions vs actual values
plt.figure(figsize=(10, 6))
plt.scatter(y_test_original, y_pred, alpha=0.3)
plt.plot([min(y_test_original), max(y_test_original)],
[min(y_test_original), max(y_test_original)], color='red')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()

123/123 ━━━━━━━━━━━━━━━━━━━━ 0s 782us/step
Mean Squared Error on test set (original scale): 2184271616.0
Mean Absolute Error on test set (original scale): 31775.314453125
```



Observations from MAE of testing prediction:

- MAE = 31,775.
- It is quite accurate to be honest.

Observations from Actual vs Predicted plot:

- The points are more concentrated along the red line in the lower value range (below 200,000), which indicates better performance for houses in this price range.

- The spread of points increases as the actual value increases, which shows that the model is less accurate for higher-priced houses.
- The points appear to be more dispersed for actual values above 300,000, indicating higher prediction errors in this range.