Importing all the necessary packages

```
In [25]: import pandas as pd
         import numpy as np
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import MinMaxScaler
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense
```

Load the dataset

```
In [26]: data = pd.read_csv('Housing.csv')
```

Data Cleaning and Encoding

```
In [32]: binary_columns = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditic

         for column in binary_columns:
             data[column] = data[column].apply(lambda x: 1 if x == 'yes' else 0)
```

```
In [34]: data = pd.get_dummies(data, columns=['furnishingstatus'], drop_first=True)
```

Split the data into features (X) and the target variable (y)

```
In [35]: X = data.drop('price', axis=1)
         y = data['price']
```

Feature Scaling

```
In [36]: scaler = MinMaxScaler()
         X_scaled = scaler.fit_transform(X)
```

Model Building

```
In [37]: model = Sequential()
         model.add(Dense(units=64, input_dim=X.shape[1], activation='relu'))
         model.add(Dense(units=32, activation='relu'))
         model.add(Dense(units=1, activation='linear'))
```

```
In [38]: model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_absolute_err
```

Data Splitting (train and test)

```
In [39]: X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, randon
```

Model Training

```
In [77]: loss, mae = model.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 997us/step - loss: 2489828245504.0000 - mea
n_absolute_error: 1196442.0000
```

Display evaluation results

```
In [78]: print(f'Mean Squared Error (MSE): {loss}')
         print(f'Mean Absolute Error (MAE): {mae}')
```

```
Mean Squared Error (MSE): 2489828245504.0
Mean Absolute Error (MAE): 1196442.0
```

Prediction

```
In [83]: new_data = pd.DataFrame({'area': [9960], 'bedrooms': [3], 'bathrooms': [2], 'stories':
                                  'guestroom': [0], 'basement': [1], 'hotwaterheating': [0],
                         'airconditioning': [0], 'parking': [2], 'prefarea': [1]})
new_data['furnishingstatus'] = 'semi-furnished'  # Replace with the appropriate value

# One-hot encode the 'furnishingstatus' column for the new data
new_data = pd.get_dummies(new_data, columns=['furnishingstatus'], drop_first=True)

# Ensure that feature names are consistent
new_data = new_data.reindex(columns=X.columns, fill_value=0)

# Scale the new data using the same scaler
new_data_scaled = scaler.transform(new_data)
```

In [84]:
```
# for column in binary_columns:
#     if column != 'furnishingstatus':  # Skip 'furnishingstatus'
#         new_data[column] = new_data[column].apply(lambda x: 1 if x == 'yes' else 0)
# new_data_scaled = scaler.transform(new_data)
```

In [85]:
```
predictions = model.predict(new_data_scaled)
```

```
1/1 [==============================] - 0s 13ms/step
```

In [86]:
```
print(f'Predicted Price: {predictions[0][0]}')
```

```
Predicted Price: 6652320.0
```

The model is a bit off from the original values but I've tried to refine it more. Before this the model was way off but after a bit of refining I was able to achieve this. The first epoch I used was 100 for that the model was way off then I tuned it to 200 the MAE was reduced and the prediction was a bit closer to the original value and then I used 500 and then 600 which is now more closer to the original value but now I did not want to tune the epoch more because it might lead to overfit. This is my observation.