1. Problem Selection and Dataset Preparation Choose a machine learning problem that requires handling uncertainty, non-linearity, or human-understandable rules, like regression, classification, or time-series forecasting. Preprocess the dataset, normalizing it if necessary.

2. Fuzzy Inference System (FIS) Design Design the Fuzzy Inference System by:

Defining Membership Functions: Set up input and output variables using membership functions, which transform crisp inputs into fuzzy values. Common choices include triangular, Gaussian, or trapezoidal functions. Establishing Fuzzy Rules: Define a rule base for mapping inputs to outputs. E.g., in a house pricing model, rules might be "If size is large and location is good, then price is high."

3. Neural Network Integration Set up a neural network to fine-tune membership functions and rules:

Neuro-Fuzzy Architecture (e.g., ANFIS): Use an Adaptive Neuro-Fuzzy Inference System (ANFIS), which combines a neural network's backpropagation learning ability with a FIS. ANFIS has layers representing fuzzification, rule evaluation, and defuzzification, which enable it to adjust fuzzy parameters (e.g., membership function parameters) through training. Training Phase: Train the model on input-output pairs to learn how to adaptively modify membership functions and optimize the rule base.

4. Implementation Steps To implement the hybrid Neuro-Fuzzy system:

Build FIS: Code the FIS, defining membership functions and initial rules. Initialize Neural Network Layers: Structure the neural network layers to match the FIS components (e.g., fuzzification, rule aggregation). Hybrid Learning: Use forward and backward propagation to minimize the error, where the neural network's learning algorithm adjusts the membership functions and rules. Optimization: Use regularization or other optimization techniques to prevent overfitting.

5. Evaluation and Fine-Tuning Evaluate Performance: Assess model performance on a test set, using metrics like accuracy (for classification), mean squared error (for regression), or other suitable metrics. Fine-Tuning: Adjust membership functions, rules, and training parameters (e.g., learning rate, epoch count) to improve accuracy or interpretability.

6. Tools and Libraries You can implement the model using Python libraries:

scikit-fuzzy for fuzzy logic functions. TensorFlow or PyTorch for the neural network components. Matplotlib for visualizations, especially helpful to view membership functions.

```
pip install numpy scikit-fuzzy tensorflow

Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (1.26.4)
Collecting scikit-fuzzy
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl.metadata (2.6
kB)
Requirement already satisfied: tensorflow in
/usr/local/lib/python3.10/dist-packages (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.44.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-
packages (from keras>=3.2.0->tensorflow) (13.9.3)
Requirement already satisfied: namex in
/usr/local/lib/python3.10/dist-packages (from keras>=3.2.0-
```

```
>tensorflow) (0.0.8)
Requirement already satisfied: optree in
/usr/local/lib/python3.10/dist-packages (from keras>=3.2.0-
>tensorflow) (0.13.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow) (3.0.6)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.18,>=2.17->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0-
>tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0-
>tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in
/usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0-
>rich->keras>=3.2.0->tensorflow) (0.1.2)
Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl (920 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 920.8/920.8 kB 6.7 MB/s eta
0:00:00
```

```python
import numpy as np

# Generate synthetic data
np.random.seed(42)
size = np.random.uniform(500, 4000, 100)  # House size in square feet
location_quality = np.random.uniform(1, 10, 100)  # Quality rating (1-
10)
rooms = np.random.randint(1, 10, 100)  # Number of rooms
price = 50000 + (size * 30) + (location_quality * 10000) + (rooms *
```

```
5000) + np.random.normal(0, 25000, 100)

# Normalize the data (important for neural networks)
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data = np.column_stack((size, location_quality, rooms, price))
data = scaler.fit_transform(data)
X, y = data[:, :-1], data[:, -1]

import skfuzzy as fuzz
from skfuzzy import control as ctrl

# Define fuzzy variables
size_fuzzy = ctrl.Antecedent(np.arange(0, 1, 0.01), 'size')
location_fuzzy = ctrl.Antecedent(np.arange(0, 1, 0.01),
'location_quality')
rooms_fuzzy = ctrl.Antecedent(np.arange(0, 1, 0.01), 'rooms')
price_fuzzy = ctrl.Consequent(np.arange(0, 1, 0.01), 'price')

# Membership functions
size_fuzzy['small'] = fuzz.trimf(size_fuzzy.universe, [0, 0, 0.5])
size_fuzzy['medium'] = fuzz.trimf(size_fuzzy.universe, [0, 0.5, 1])
size_fuzzy['large'] = fuzz.trimf(size_fuzzy.universe, [0.5, 1, 1])

location_fuzzy['poor'] = fuzz.trimf(location_fuzzy.universe, [0, 0,
0.5])
location_fuzzy['average'] = fuzz.trimf(location_fuzzy.universe, [0,
0.5, 1])
location_fuzzy['good'] = fuzz.trimf(location_fuzzy.universe, [0.5, 1,
1])

rooms_fuzzy['few'] = fuzz.trimf(rooms_fuzzy.universe, [0, 0, 0.5])
rooms_fuzzy['moderate'] = fuzz.trimf(rooms_fuzzy.universe, [0, 0.5,
1])
rooms_fuzzy['many'] = fuzz.trimf(rooms_fuzzy.universe, [0.5, 1, 1])

price_fuzzy['low'] = fuzz.trimf(price_fuzzy.universe, [0, 0, 0.5])
price_fuzzy['medium'] = fuzz.trimf(price_fuzzy.universe, [0, 0.5, 1])
price_fuzzy['high'] = fuzz.trimf(price_fuzzy.universe, [0.5, 1, 1])

rule1 = ctrl.Rule(size_fuzzy['small'] & location_fuzzy['poor'] &
rooms_fuzzy['few'], price_fuzzy['low'])
rule2 = ctrl.Rule(size_fuzzy['large'] & location_fuzzy['good'] &
rooms_fuzzy['many'], price_fuzzy['high'])
# Add more rules based on problem understanding

price_ctrl = ctrl.ControlSystem([rule1, rule2])
price_simulation = ctrl.ControlSystemSimulation(price_ctrl)
```

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Create a neural network model
model = Sequential([
    Dense(10, activation='relu', input_shape=(X.shape[1],)),
    Dense(10, activation='relu'),
    Dense(1)  # Output layer
])

model.compile(optimizer='adam', loss='mse')

# Train the model
model.fit(X, y, epochs=100, batch_size=10, verbose=1)
```

```
/usr/local/lib/python3.10/dist-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)

Epoch 1/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - loss: 0.1421
Epoch 2/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0955
Epoch 3/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0530
Epoch 4/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0197
Epoch 5/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0142
Epoch 6/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0170
Epoch 7/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0186
Epoch 8/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0172
Epoch 9/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 0.0142
Epoch 10/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0137
Epoch 11/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0120
Epoch 12/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0113
Epoch 13/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0127
Epoch 14/100
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0108
Epoch 15/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0119
Epoch 16/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0104
Epoch 17/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0118
Epoch 18/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0123
Epoch 19/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0116
Epoch 20/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0100
Epoch 21/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0125
Epoch 22/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0105
Epoch 23/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0108
Epoch 24/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0123
Epoch 25/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0116
Epoch 26/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0130
Epoch 27/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0106
Epoch 28/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0110
Epoch 29/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0101
Epoch 30/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0108
Epoch 31/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0115
Epoch 32/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0101
Epoch 33/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0110
Epoch 34/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0089
Epoch 35/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0099
Epoch 36/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0107
Epoch 37/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0099
Epoch 38/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0131
```

```
Epoch 39/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0104
Epoch 40/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0109
Epoch 41/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0100
Epoch 42/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0118
Epoch 43/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0112
Epoch 44/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0083
Epoch 45/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0095
Epoch 46/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0119
Epoch 47/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0107
Epoch 48/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0093
Epoch 49/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0108
Epoch 50/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0100
Epoch 51/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0103
Epoch 52/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0095
Epoch 53/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0100
Epoch 54/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0098
Epoch 55/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0089
Epoch 56/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0111
Epoch 57/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0096
Epoch 58/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0111
Epoch 59/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0120
Epoch 60/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0116
Epoch 61/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0083
Epoch 62/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0088
Epoch 63/100
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0091
Epoch 64/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0114
Epoch 65/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0111
Epoch 66/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0095
Epoch 67/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0107
Epoch 68/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0116
Epoch 69/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0083
Epoch 70/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0086
Epoch 71/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0106
Epoch 72/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0102
Epoch 73/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0093
Epoch 74/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0101
Epoch 75/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0094
Epoch 76/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0099
Epoch 77/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0087
Epoch 78/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0081
Epoch 79/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0104
Epoch 80/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0123
Epoch 81/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0122
Epoch 82/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0111
Epoch 83/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0103
Epoch 84/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0093
Epoch 85/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0100
Epoch 86/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0106
Epoch 87/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0110
```

```
Epoch 88/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0090
Epoch 89/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0104
Epoch 90/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0100
Epoch 91/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0111
Epoch 92/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0098
Epoch 93/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0097
Epoch 94/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0097
Epoch 95/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0083
Epoch 96/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0085
Epoch 97/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0097
Epoch 98/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0088
Epoch 99/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0076
Epoch 100/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0102

<keras.src.callbacks.history.History at 0x7a4a12798850>
```

```python
def hybrid_inference(input_data):
    # Neural Network Prediction
    nn_output = model.predict(input_data)

    # Set up Fuzzy Inference inputs
    price_simulation.input['size'] = input_data[0, 0]
    price_simulation.input['location_quality'] = input_data[0, 1]
    price_simulation.input['rooms'] = input_data[0, 2]

    # Perform fuzzy computation
    try:
        price_simulation.compute()
        # Ensure 'price' exists in output
        if 'price' in price_simulation.output:
            fuzzy_output = price_simulation.output['price']
        else:
            print("Fuzzy output 'price' not found. Check rule
definitions and input values.")
            fuzzy_output = nn_output[0][0]  # Fallback to NN output if
fuzzy output is missing
    except Exception as e:
```

```
        print("Error in fuzzy computation:", e)
        fuzzy_output = nn_output[0][0]  # Fallback in case of error

    # Combine fuzzy and neural network output
    hybrid_output = (nn_output[0][0] + fuzzy_output) / 2
    return hybrid_output

test_data = X[:5]  # Example test data
for sample in test_data:
    prediction = hybrid_inference(sample.reshape(1, -1))
    print("Hybrid Model Prediction:", prediction)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
Fuzzy output 'price' not found. Check rule definitions and input
values.
Hybrid Model Prediction: 0.33684077858924866
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
Fuzzy output 'price' not found. Check rule definitions and input
values.
Hybrid Model Prediction: 0.6675569415092468
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
Fuzzy output 'price' not found. Check rule definitions and input
values.
Hybrid Model Prediction: 0.42293640971183777
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
Hybrid Model Prediction: 0.657500549184596
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
Fuzzy output 'price' not found. Check rule definitions and input
values.
Hybrid Model Prediction: 0.4744725227355957
```