Ex. No. 2	Regression
Date of Exercise	21/07/2025

Aim:

To implement a fully connected neural network for a regression task using a house price dataset.

Description:

This experiment involves building a feedforward neural network to predict house prices based on features such as square footage, number of rooms, and location.

Step 1: Compute the Mean and Standard Deviation

The mean (μ) and standard deviation (σ) of `X_train`:

$$\mu = \frac{1+2+3+7+8}{5} = \frac{21}{5} = 4.2$$

$$\sigma = \sqrt{\frac{(1-4.2)^2 + (2-4.2)^2 + (3-4.2)^2 + (7-4.2)^2 + (8-4.2)^2}{5}}$$

$$\sigma = \sqrt{\frac{10.24 + 4.84 + 1.44 + 7.84 + 14.44}{5}} = \sqrt{\frac{38.8}{5}} = \sqrt{7.76} \approx 2.785$$

Step 2: Standardize `X_train`

Using the formula:

$$X_{\text{scaled}} = \frac{X - 4.2}{2.785}$$

Original X	Computation	Scaled X_{scaled}
1	$\frac{1-4.2}{2.785}$	-1.1487
2	$\frac{2-4.2}{2.785}$	-0.7898
3	$\frac{3-4.2}{2.785}$	-0.4308
7	$\frac{7-4.2}{2.785}$	1.0051
8	$\frac{8-4.2}{2.785}$	1.3641

1. **Mean Quadratic Error (MQE)**
   Also known as **Mean Squared Error (MSE)**, it measures the average of the squared differences between actual and predicted values.

   Formula:

$$MQE = MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

   where:

   - n = Total number of data points
   - Y_i = Actual value
   - $\hat{Y}_i$ = Predicted value

2. **Mean Absolute Error (MAE)**
   Measures the average absolute difference between actual and predicted values.

   Formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

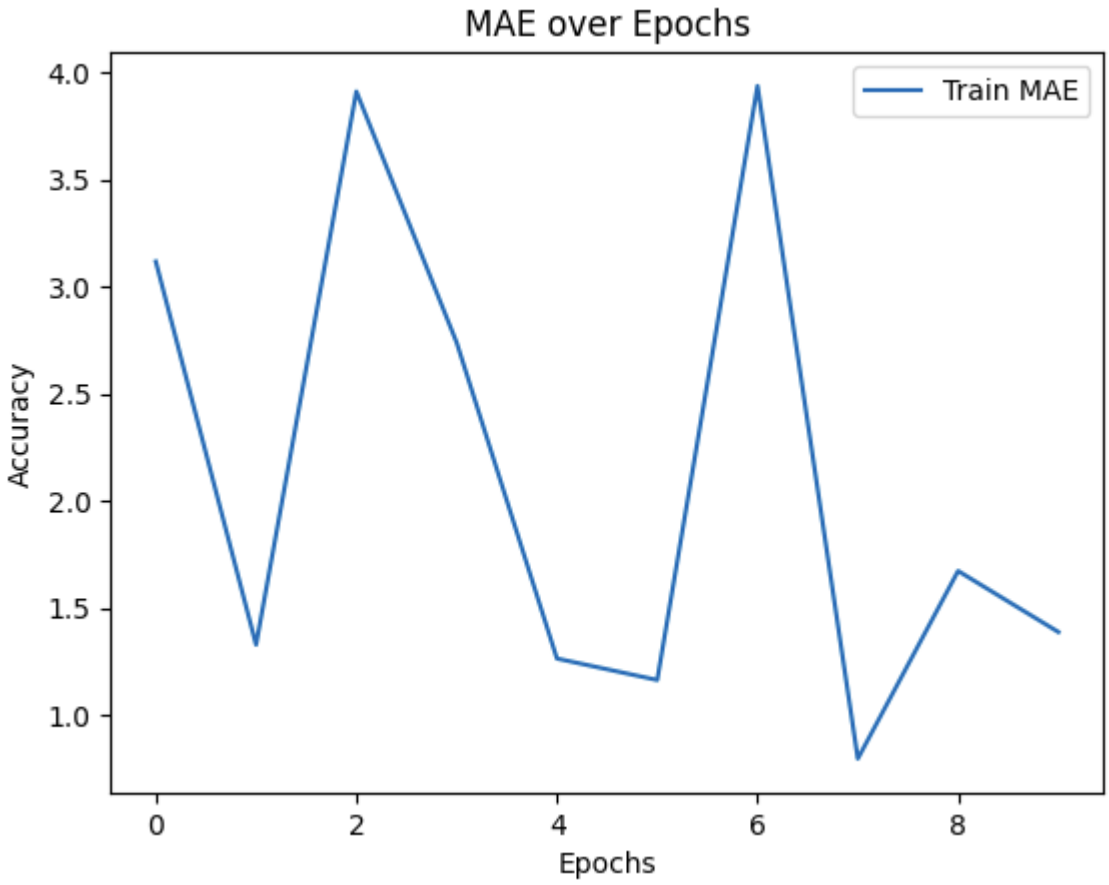
Code:

```
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Load dataset
url = "https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.csv"
data = pd.read_csv(url)

# Preprocessing
x = data[['median_income', 'total_rooms']]
y = data['median_house_value'] / 100000
```

```
# Build model
model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=[len(x.keys())]),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
])

# Compile and train
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.fit(x, y, epochs=10, batch_size=32)
```

Output

```
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m1s[0m 848us/step - loss: 793.8102 - mae: 8.8261
Epoch 2/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 861us/step - loss: 7.1922 - mae: 1.6356
Epoch 3/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 875us/step - loss: 73.7060 - mae: 3.7534
Epoch 4/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 844us/step - loss: 31.1500 - mae: 1.7702
Epoch 5/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 851us/step - loss: 5.2933 - mae: 1.4118
Epoch 6/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 869us/step - loss: 6.4163 - mae: 1.4833
Epoch 7/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 929us/step - loss: 385.2918 - mae: 8.5805
Epoch 8/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 874us/step - loss: 1.1946 - mae: 0.7941
Epoch 9/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 906us/step - loss: 2.5683 - mae: 0.8929
Epoch 10/10
[1m516/516[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m1s[0m 996us/step - loss: 23.0670 - mae: 2.4037
```



Result

The above experiment of Multi Class Classification is done successfully and the output is been obtained