```python
# Boston Housing Price Prediction - ShadowFox Internship Task
# Author: Vedant

# ==========================
# 1. Import Required Libraries
# ==========================
from sklearn.datasets import fetch_california_housing
import pandas as pd

# Load California Housing dataset
housing = fetch_california_housing(as_frame=True)

# Create DataFrame
df = housing.frame
df.rename(columns={"MedHouseVal": "MEDV"}, inplace=True)  # rename target column for consistency

# Save to CSV (optional)
df.to_csv("BostonHousing.csv", index=False)

print("Dataset shape:", df.shape)
print(df.head())


import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

```
# ==========================
# 2. Load Dataset
# ==========================
# NOTE: Place your BostonHousing.csv in the same folder as this notebook
df = pd.read_csv("BostonHousing.csv")
print("Dataset shape:", df.shape)
print(df.head())

# ==========================
# 3. Split Features and Target
# ==========================
X = df.drop("MEDV", axis=1)   # Features
y = df["MEDV"]                # Target variable (House Price)

# ==========================
# 4. Train-Test Split
# ==========================
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Train size:", X_train.shape, "Test size:", X_test.shape)

# ==========================
# 5. Data Normalization
# ==========================
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```python
# 6. Build Regression Model (Neural Network)
# ==========================
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(32, activation="relu"),
    tf.keras.layers.Dense(1)  # single output for regression
])

model.compile(optimizer="adam", loss="mse", metrics=["mae"])

# ==========================
# 7. Train Model
# ==========================
history = model.fit(X_train, y_train, epochs=100, validation_data=(X_test, y_test), verbose=0)

# ==========================
# 8. Evaluate Model
# ==========================
loss, mae = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Mean Absolute Error: {mae:.2f}")

# ==========================
# 9. Visualize Training
# ==========================
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history['val_mae'], label='Validation MAE')
plt.xlabel("Epochs")
plt.ylabel("Mean Absolute Error")
plt.legend()
plt.title("Training vs Validation Error")
plt.show()
```

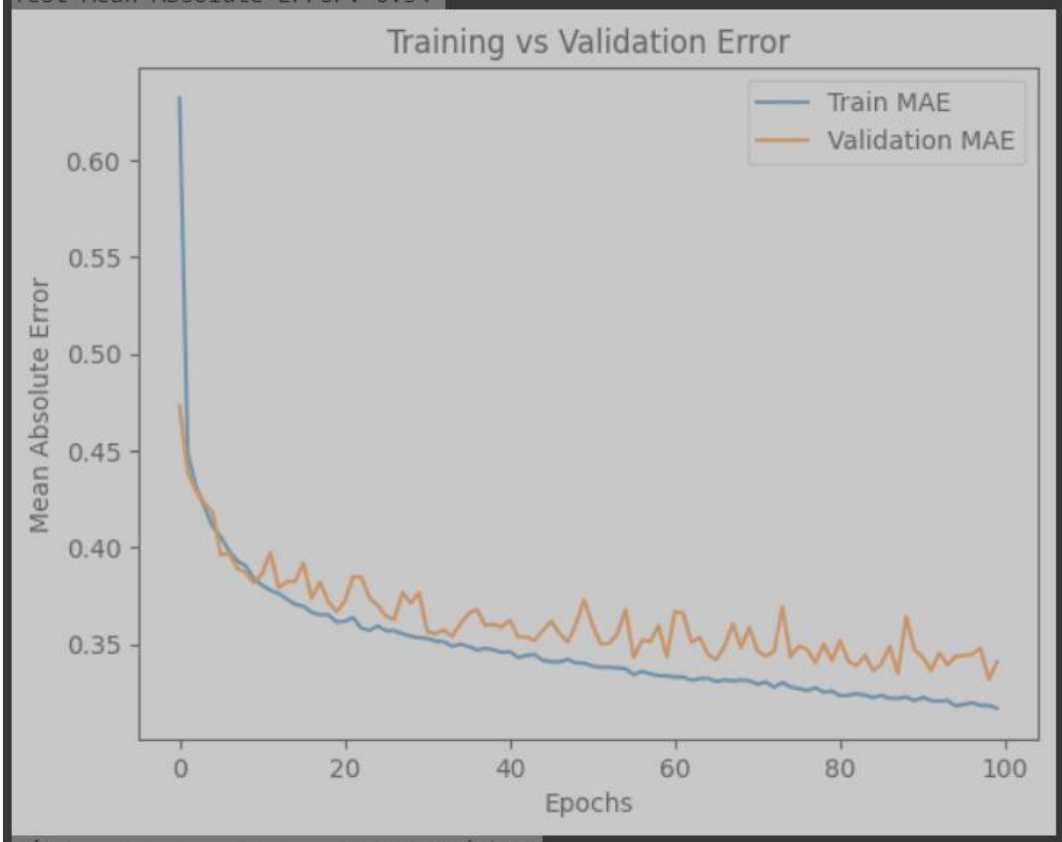
```python
# 10. Predict and Compare
# ========================
y_pred = model.predict(X_test[:10]).flatten()
print("Predicted prices:", y_pred)
print("Actual prices:", y_test[:10].values)
```

```
Dataset shape: (20640, 9)
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude   MEDV
0    -122.23  4.526
1    -122.22  3.585
2    -122.24  3.521
3    -122.25  3.413
4    -122.25  3.422
Dataset shape: (20640, 9)
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude   MEDV
0    -122.23  4.526
1    -122.22  3.585
2    -122.24  3.521
```

```
3    -122.25  3.413
4    -122.25  3.422
Train size: (16512, 8) Test size: (4128, 8)
Test Mean Absolute Error: 0.34
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step
Predicted prices: [0.4555246 0.8818768 5.2186923 2.4325957 2.9516323 1.6725786 2.128673
 1.4970808 2.2517397 4.4329834]
```