import numpy as np

import pandas as pd

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Load the dataset

data = pd.read\_csv(r"C:\Users\Aditi\Downloads\archive\HousingData.csv")

# Handle missing values by dropping rows with any NaNs

data = data.dropna()

# Split features and target

X = data.drop("MEDV", axis=1)

y = data["MEDV"]

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Build the DNN model

model = keras.Sequential([

layers.Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),

layers.Dense(32, activation='relu'),

layers.Dense(1)

])

# Compile the model

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

# Train the model

history = model.fit(X\_train\_scaled, y\_train, validation\_split=0.1, epochs=100, batch\_size=32, verbose=1)

# Evaluate the model

loss, mae = model.evaluate(X\_test\_scaled, y\_test)

print(f"\nMean Absolute Error on Test Data: {mae:.2f}")

# Predict house prices

y\_pred = model.predict(X\_test\_scaled).flatten()

# Evaluation metrics

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse:.2f}")

print(f"R² Score: {r2:.2f}")

# Print actual vs predicted prices (first 10 examples)

print("\nActual vs Predicted House Prices (First 10 rows):")

for actual, predicted in list(zip(y\_test, y\_pred))[:10]:

print(f"Actual: {actual:.2f}, Predicted: {predicted:.2f}")

# Plotting

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='green', alpha=0.6)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], '--r')

plt.xlabel('Actual Prices (in $1000s)')

plt.ylabel('Predicted Prices (in $1000s)')

plt.title('Actual vs Predicted Boston House Prices')

plt.grid(True)

plt.show()