# Correct imports

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.datasets import boston\_housing

from sklearn.preprocessing import StandardScaler

# Load data

(X\_train, y\_train), (X\_test, y\_test) = boston\_housing.load\_data()

# Standardize the features

scaler = StandardScaler()

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

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

# Build the model

model = Sequential()

model.add(Dense(1, input\_dim=X\_train.shape[1], activation='linear'))

# Compile the model

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

# Train the model

history = model.fit(X\_train, y\_train, epochs=100, validation\_split=0.2, verbose=1)

# Evaluate the model

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

print(f"Mean Squared Error (MSE) on test set: {mse}")

print(f"Mean Absolute Error (MAE) on test set: {mae}")

# Predictions

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

# Plot

plt.scatter(y\_test, y\_pred)

plt.xlabel("True Prices")

plt.ylabel("Predicted Prices")

plt.title("True vs Predicted House Prices")

plt.show()

**# Import necessary libraries**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**from tensorflow import keras**

**from tensorflow.keras import models, layers**

**# Load the IMDB dataset**

**(train\_data, train\_labels), (test\_data, test\_labels) = keras.datasets.imdb.load\_data(num\_words=10000)**

**# Function to vectorize sequences (one-hot encoding manually)**

**def vectorize\_sequences(sequences, dimension=10000):**

**results = np.zeros((len(sequences), dimension))**

**for i, sequence in enumerate(sequences):**

**results[i, sequence] = 1.0**

**return results**

**# Vectorize train and test data**

**x\_train = vectorize\_sequences(train\_data)**

**x\_test = vectorize\_sequences(test\_data)**

**# Convert labels to float32**

**y\_train = np.asarray(train\_labels).astype("float32")**

**y\_test = np.asarray(test\_labels).astype("float32")**

**# Build a simple Deep Neural Network for binary classification**

**model = models.Sequential()**

**model.add(layers.Dense(16, activation='relu', input\_shape=(10000,)))**

**model.add(layers.Dense(16, activation='relu'))**

**model.add(layers.Dense(1, activation='sigmoid')) # Output 0 or 1 (sentiment)**

**# Compile the model**

**model.compile(optimizer='rmsprop',**

**loss='binary\_crossentropy',**

**metrics=['accuracy'])**

**# Train the model**

**history = model.fit(x\_train, y\_train, epochs=10, batch\_size=512, validation\_split=0.2, verbose=1)**

**# Evaluate the model**

**test\_loss, test\_accuracy = model.evaluate(x\_test, y\_test)**

**print(f"Test Loss: {test\_loss}")**

**print(f"Test Accuracy: {test\_accuracy}")**

**# Make predictions**

**predictions = model.predict(x\_test)**

**# (Optional) Plot training and validation loss**

**plt.plot(history.history['loss'], label='Training Loss')**

**plt.plot(history.history['val\_loss'], label='Validation Loss')**

**plt.title('Training and Validation Loss')**

**plt.xlabel('Epochs')**

**plt.ylabel('Loss')**

**plt.legend()**

**plt.show()**

**# (Optional) Plot training and validation accuracy**

**plt.plot(history.history['accuracy'], label='Training Accuracy')**

**plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')**

**plt.title('Training and Validation Accuracy')**

**plt.xlabel('Epochs')**

**plt.ylabel('Accuracy')**

**plt.legend()**

**plt.show()**

import os

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt

# Main dataset folder (having 'train' and 'test' subfolders)

dataset\_dir = 'path/to/dataset' # <-- Just give the main dataset folder path

train\_dir = os.path.join(dataset\_dir, 'train')

test\_dir = os.path.join(dataset\_dir, 'test')

# Parameters

img\_size = 150

batch\_size = 32

# Data generators

train\_gen = ImageDataGenerator(rescale=1./255, rotation\_range=40, width\_shift\_range=0.2,

height\_shift\_range=0.2, zoom\_range=0.2, horizontal\_flip=True)

test\_gen = ImageDataGenerator(rescale=1./255)

train\_data = train\_gen.flow\_from\_directory(train\_dir, target\_size=(img\_size, img\_size),

batch\_size=batch\_size, class\_mode='categorical')

test\_data = test\_gen.flow\_from\_directory(test\_dir, target\_size=(img\_size, img\_size),

batch\_size=batch\_size, class\_mode='categorical')

# Build the CNN model

model = models.Sequential([

layers.Conv2D(32, (3,3), activation='relu', input\_shape=(img\_size, img\_size, 3)),

layers.MaxPooling2D(2,2),

layers.Conv2D(64, (3,3), activation='relu'),

layers.MaxPooling2D(2,2),

layers.Flatten(),

layers.Dense(512, activation='relu'),

layers.Dense(train\_data.num\_classes, activation='softmax')

])

# Compile and Train

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(train\_data, epochs=10, validation\_data=test\_data)

# Evaluate

test\_loss, test\_acc = model.evaluate(test\_data)

print(f"Test Accuracy: {test\_acc:.2f}")

# Plot Training History

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)

plt.plot(history.history['accuracy'], label='Train')

plt.plot(history.history['val\_accuracy'], label='Validation')

plt.title('Accuracy')

plt.legend()

plt.subplot(1,2,2)

plt.plot(history.history['loss'], label='Train')

plt.plot(history.history['val\_loss'], label='Validation')

plt.title('Loss')

plt.legend()

plt.show()