DL1

import numpy as npimport pandas as pdfrom sklearn.model\_selection import train\_test\_splitfrom sklearn.preprocessing import StandardScalerfrom tensorflow.keras.models import Sequentialfrom tensorflow.keras.layers import Dense, Dropoutfrom keras.optimizers import Adamfrom keras.layers import LeakyReLU# Load datadata = pd.read\_csv("Boston\_Housing.csv")# Drop unwanted column only if it existsif 'Unnamed: 14' in data.columns: data = data.drop(columns=['Unnamed: 14'])# Separate features and targetX = data.drop(columns=['MEDV']) # Featuresy = data['MEDV'] # Target (house price)# Split the datasetX\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)# Standardize featuresscaler = StandardScaler()X\_train\_scaled = scaler.fit\_transform(X\_train)X\_test\_scaled = scaler.transform(X\_test)# Build the modelmodel = Sequential([ Dense(128, input\_shape=(X\_train\_scaled.shape[1],)), LeakyReLU(alpha=0.1), Dropout(0.2), Dense(64), LeakyReLU(alpha=0.1), Dropout(0.2), Dense(32), LeakyReLU(alpha=0.1), Dropout(0.2), Dense(1) # Output layer for regression])# Compile the modeloptimizer = Adam(learning\_rate=0.0001)model.compile(optimizer=optimizer, loss='mean\_squared\_error')# Train the modelmodel.fit(X\_train\_scaled, y\_train, epochs=200, batch\_size=32, validation\_split=0.2)# Predictpredictions = model.predict(X\_test\_scaled)# Print some predictions and actual valuesprint("Some Predictions and Actual Values:")for i in range(10): print(f"Predicted Price: {predictions[i][0]:.2f}, Actual Price: {y\_test.iloc[i]}")

"""This project predicts house prices using a deep learning model

built with Keras. We used the Boston Housing dataset,

which has different features like crime rate, number of rooms, and distance from the city center,

to predict the house price (MEDV).

First, we cleaned the data, removed any extra columns, and split it into training and testing parts.

Then we scaled the features to bring them to the same range.

We created a neural network with multiple layers and used activation functions like LeakyReLU and dropout to avoid overfitting.

The model was trained using the Adam optimizer and mean squared error as the loss function.

After training, we tested the model and found that it predicts house prices quite accurately. T

his model can help in estimating house values based on location and features."""

"""

1. Dataset Description:

The Boston Housing dataset contains 506 rows and 13 main features (like crime rate, average number of rooms, property tax, etc.) that affect the house price (target value: MEDV).

2. Why StandardScaler is used:

We used StandardScaler to bring all features to the same scale so that the model can train better and converge faster.

3. Why LeakyReLU:

We used LeakyReLU instead of ReLU to avoid the "dying ReLU" problem, which can stop neurons from learning if their output becomes 0.

4. Why Dropout layers:

Dropout helps prevent overfitting by randomly turning off some neurons during training, making the model more general and stable.

5. Evaluation:

After training for 200 epochs, we tested the model using the test set and printed some predicted and actual prices. The results were close, which shows the model is working well.

6. Real-life Application:

This type of model can be used by real estate companies or property apps to suggest house prices based on location and features.

"""

"""

Here is a line-by-line explanation of your Boston Housing price prediction code in very easy language:

python

Copy

Edit

import numpy as np

import pandas as pd

👉 These two lines import NumPy and Pandas, which are used for handling numbers and datasets.

python

Copy

Edit

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

from sklearn.preprocessing import StandardScaler

👉 These import functions from sklearn to split the dataset and to scale (standardize) the features.

python

Copy

Edit

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from keras.optimizers import Adam

from keras.layers import LeakyReLU

👉 These lines import the tools needed to build a deep learning model: layers like Dense, Dropout, activation function LeakyReLU, and optimizer Adam.

python

Copy

Edit

data = pd.read\_csv("Boston\_Housing.csv")

👉 Reads the dataset file and loads it into a DataFrame called data.

python

Copy

Edit

if 'Unnamed: 14' in data.columns:

data = data.drop(columns=['Unnamed: 14'])

👉 Removes the extra column if it exists (sometimes added by mistake in CSV files).

python

Copy

Edit

X = data.drop(columns=['MEDV']) # Features

y = data['MEDV'] # Target (house price)

👉 Separates the data into input features X and the output (price) y.

python

Copy

Edit

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

👉 Splits the data into training and testing sets (80% for training, 20% for testing).

python

Copy

Edit

scaler = StandardScaler()

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

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

👉 Standardizes the features so that all values have similar ranges (important for neural networks).

python

Copy

Edit

model = Sequential([

Dense(128, input\_shape=(X\_train\_scaled.shape[1],)),

LeakyReLU(alpha=0.1),

Dropout(0.2),

Dense(64),

LeakyReLU(alpha=0.1),

Dropout(0.2),

Dense(32),

LeakyReLU(alpha=0.1),

Dropout(0.2),

Dense(1)

])

👉 This block builds the neural network model:

Dense layers are fully connected layers.

LeakyReLU is an activation function that helps the model learn better.

Dropout prevents overfitting.

The last Dense(1) outputs one value (house price).

python

Copy

Edit

optimizer = Adam(learning\_rate=0.0001)

model.compile(optimizer=optimizer, loss='mean\_squared\_error')

👉 Compiles the model using the Adam optimizer and mean squared error loss because it's a regression problem.

python

Copy

Edit

model.fit(X\_train\_scaled, y\_train, epochs=200, batch\_size=32, validation\_split=0.2)

👉 Trains the model for 200 times (epochs) on the training data, using 32 samples at a time. 20% of training data is used as validation.

python

Copy

Edit

predictions = model.predict(X\_test\_scaled)

👉 After training, this line makes predictions on the test data.

python

Copy

Edit

print("Some Predictions and Actual Values:")

for i in range(10):

print(f"Predicted Price: {predictions[i][0]:.2f}, Actual Price: {y\_test.iloc[i]}")

👉 Prints 10 sample predicted house prices and their actual prices to compare how well the model performed.

"""