Assignment 1: Linear regression by using Deep Neural network: Implement Boston housing price prediction problem by Linear regression using Deep Neural network. Use Boston House price prediction dataset

```
# Import necessary libraries
import numpy as np  # For numerical operations
import pandas as pd  # For handling datasets
from sklearn.model_selection import train_test_split  # Splitting data into train & test sets
from sklearn.linear_model import LinearRegression  # Linear Regression Model
from sklearn.preprocessing import StandardScaler  # Standardization of data
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score  # Evaluation metrics

# Importing Keras (for Neural Network)
import keras
from keras.models import Sequential  # To define a sequential model
from keras.layers import Dense  # Fully connected layers

# Importing Google Colab file handling utility
from google.colab import files

# Uploading and Loading Dataset
uploaded = files.upload()  # Opens file upload dialogue in Google Colab
boston = pd.read_csv("boston_house_prices.csv")  # Reads CSV file into a DataFrame

# Selecting Features and Target
# Selecting 3 input features:
# 1. LSTAT (Percentage of lower status population)
# 2. RM (Average number of rooms per dwelling)
# 3. PTRATIO (Pupil-teacher ratio by town)
X = boston[['LSTAT', 'RM', 'PTRATIO']]

# Target variable: House Price
y = boston['PRICE']

# Splitting the Dataset into Training and Testing Sets
# 80% of data used for training, 20% for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
```

```
# Standardizing the Dataset (Feature Scaling)
# Standardization improves model performance by normalizing feature values
scaler = StandardScaler()  # Initializing StandardScaler
X_train_scaled = scaler.fit_transform(X_train)  # Fit and transform training data
X_test_scaled = scaler.transform(X_test)  # Transform test data using the same scaler

# Linear Regression Model
lr_model = LinearRegression()  # Initializing Linear Regression Model
lr_model.fit(X_train_scaled, y_train)  # Training the model using scaled training data

# Predicting house prices on test data
y_pred_lr = lr_model.predict(X_test_scaled)

# Evaluating Linear Regression Model
mse_lr = mean_squared_error(y_test, y_pred_lr)  # Mean Squared Error
mae_lr = mean_absolute_error(y_test, y_pred_lr)  # Mean Absolute Error
r2_lr = r2_score(y_test, y_pred_lr)  # R² Score (Model accuracy measure)

# Displaying evaluation metrics
print("Linear Regression Model Evaluation:")
print(f"Mean Squared Error: {mse_lr}")
print(f"Mean Absolute Error: {mae_lr}")
print(f"R2 Score: {r2_lr}")

# Neural Network (ANN) Model
# Creating a Deep Learning Model using Keras Sequential API
model = Sequential([
    Dense(128, activation='relu', input_dim=3),  # Input layer (3 features) & first hidden
layer (128 neurons)
    Dense(64, activation='relu'),  # Second hidden layer with 64 neurons
    Dense(32, activation='relu'),  # Third hidden layer with 32 neurons
    Dense(16, activation='relu'),  # Fourth hidden layer with 16 neurons
    Dense(1)  # Output layer (Predicting a single value - House Price)
])

# Compiling the model
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
# Optimizer: Adam (Adaptive Learning Rate Optimization)
# Loss function: Mean Squared Error (MSE) - Suitable for regression problems
# Metric: Mean Absolute Error (MAE) - Helps measure performance
```

```
# Training the Neural Network
history = model.fit(X_train_scaled, y_train, epochs=100, validation_split=0.05,
verbose=1)
# Training for 100 epochs
# Using 5% of training data as validation set to monitor overfitting
# `verbose=1` displays detailed training progress
#Epoch 1/100
#12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 26ms/step - loss: 547.8306 - #mae: 21.6359 -
val_loss: 445.7750 - val_mae: 20.1572
#Epoch 2/100
#12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 550.6208 - #mae: 21.6498 -
val_loss: 403.5681 - val_mae: 19.1308
#Epoch 3/100
#12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 433.7596 -
# Evaluating the Neural Network Model
y_pred_nn = model.predict(X_test_scaled)  # Predicting house prices on test data
mse_nn, mae_nn = model.evaluate(X_test_scaled, y_test)  # Evaluating model
performance

# Displaying Neural Network Evaluation Metrics
print("\nNeural Network Model Evaluation:")
print(f"Mean Squared Error: {mse_nn}")
print(f"Mean Absolute Error: {mae_nn}")

# House Price Prediction for New Data
new_data = np.array([[0.1, 10.0, 5.0]])
# New input values: LSTAT=0.1, RM=10.0, PTRATIO=5.0

new_data_scaled = scaler.transform(new_data)
 # Applying the same standardization as training data

# Predicting price using trained neural network model
prediction = model.predict(new_data_scaled)

# Displaying the predicted house price
print("\nPredicted House Price:", prediction[0][0])

#OutPut
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step

Predicted House Price: 79.24278
```