**Practical No 1:**

import numpy as np

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

#import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from keras.optimizers import Adam

from keras.layers import LeakyReLU, Dropout

# Load the dataset

data = pd.read\_csv("C:/Users/Administrator/Documents/BE/DL Pratical/Boston\_Housing (1).csv")

# Preprocess the data

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

y = data['LSTAT']  # Target

# Drop the "Unnamed: 14" column

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

# Preprocess the data again

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

y = data['MEDV']  # Target

# Split the data into training and test sets

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

# Standardize the features

scaler = StandardScaler()

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

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

# Build the model

model = Sequential([

    Dense(128, *activation*=LeakyReLU(*alpha*=0.1), *input\_shape*=(X\_train\_scaled.shape[1],)),

    Dropout(0.2),  # Adjust dropout rate

    Dense(64, *activation*=LeakyReLU(*alpha*=0.1)),

    Dropout(0.2),  # Adjust dropout rate

    Dense(32, *activation*=LeakyReLU(*alpha*=0.1)),

    Dropout(0.2),  # Adjust dropout rate

    Dense(1)  # Linear regression output layer

])

# Adjusted Training Parameters

optimizer = Adam(*learning\_rate*=0.0001)  # Adjust learning rate

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

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

# Adjusted Training Parameters

optimizer = Adam(*learning\_rate*=0.0001)  # Adjust learning rate

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

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

# Make predictions

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

# Print some predictions and actual values

print("Some Predictions and Actual Values:")

for i in range(10):

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