### Experiment 1: Build a Deep Neural Network Model Using Linear Regression with a Single Variable

**AIM:** To Build a Deep Neural Network Model Using Linear Regression with a Single Variable

**THEORY:**

**Regression:**

A regression is a statistical technique that relates a dependent variable to one or more independent (explanatory) variables. A regression model is able to show whether changes observed in the dependent variable are associated with changes in one or more of the explanatory variables.

**Linear Regression:**

Linear regression is probably one of the most important and widely used regression techniques. It's among the simplest regression methods. One of its main advantages is the ease of interpreting results. When implementing linear regression of some dependent variable on the set of independent variables x= ($x1 \ldots x_r$), where r is the number of predictors, you assume a linear relationship between y and x: y= β0 + β1x1 + ......+ βrxr+ ϵ. This equation is the regression equation. β0, $\beta 1$, $\beta_r$ are the regression coefficients, and $\varepsilon$ is the random error. Linear regression calculates the estimators of the regression coefficients or simply the predicted weights, denoted by b1...br. They define the estimated regression function f(x) = b0 +b1x1 +....+brxr This function should capture the dependencies between the inputs and output sufficiently well.

**Simple Linear Regression:**

When implementing simple linear regression, with a given set of input-output (x-y) pairs (green circles). Which are the observations. For example, the leftmost observation (green circle) has the input x= 5 and the actual output (response) y= 5. The next one has x= 15 and y= 20, and so on. The estimated regression function (black line) has the equation f(x)= b0+b1x.Your goal is to calculate the optimal values of the predicted weights b0 and b1 that minimize SSR and determine the estimated regression function. The value of b0, also called the intercept, shows the point where the estimated regression line crosses the y axis. It is the value of the estimated response f(x) for x=0. The value of b1 determines the slope of the estimated regression line. The predicted responses (red squares) are the points on the regression line that correspond to the input values. For example, for the input x= 5, the predicted response is f (5) = 8.33 (represented with the leftmost red square). The residuals (vertical dashed gray lines) can be calculated as $y1_i$-f($x1_i$)=y1−b0−b1x1fori=1

```
# ================= STUDENT INFORMATION =================

STUDENT NAME = "A.Manas"
```

```
ROLL_NO = "23VE1A6601"
SECTION = "A"

EXPERIMENT_NO = "1"
EXPERIMENT_TITLE = "Design and Implementation of a Deep Neural Network Using Single Variable Linear Regression"

import uuid
SUBMISSION_ID = str(uuid.uuid4())[:8]

print("Student Name     :", STUDENT_NAME)
print("Roll Number      :", ROLL_NO)
print("Section          :", SECTION)
print("Experiment No    :", EXPERIMENT_NO)
print("Experiment Title :", EXPERIMENT_TITLE)
print("Submission ID    :", SUBMISSION_ID)

# ================= EXPERIMENT CODE =================

import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
observations=1000
xs=np.random.uniform(-10,10,(observations,1))
zs=np.random.uniform(-10,10,(observations,1))
generated_inputs=np.column_stack((xs,zs))
noise=np.random.uniform(-10,10,(observations,1))
generated_target=2*xs-3*zs+5+noise
np.savez('TF_intro',input=generated_inputs,targets=generated_target)
training_data=np.load('TF_intro.npz')
input_size=2
output_size=1
model = tf.keras.Sequential([tf.keras.layers.Dense(output_size)])
custom_optimizer=tf.keras.optimizers.SGD(learning_rate=0.02)
model.compile(optimizer=custom_optimizer,loss='mean_squared_error')
model.fit(training_data['input'],training_data['targets'],epochs=100,verbose=1)
model.layers[0].get_weights()
#[array([[ 1.3665189],[-3.1609795]], dtype=float32), array([4.9344487], dtype=float32)]
weights=model.layers[0].get_weights()[0]
bias=model.layers[0].get_weights()[1]
out=training_data['targets'].round(1)
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(generated_target, out)
```

```
rmse = np.sqrt(mse)
print("RMSE:", rmse)
plt.scatter(np.squeeze(model.predict_on_batch(training_data['input'])),np.squeeze(training_data['targets']),c='#88c999')
plt.xlabel('Input')
plt.ylabel('Predicted Output')
plt.show()
```

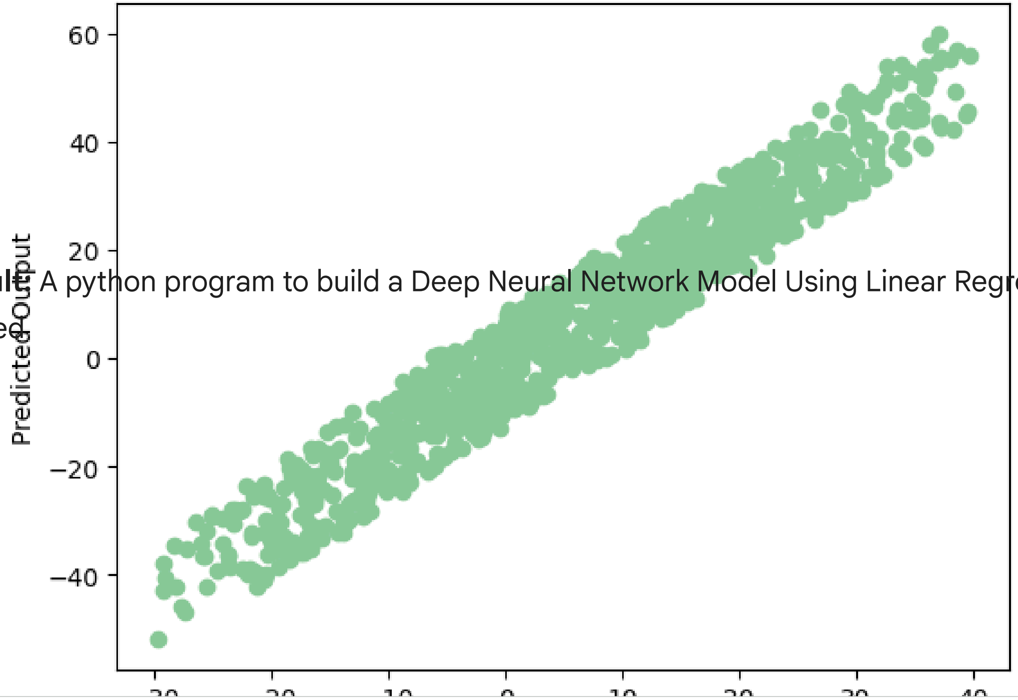
```
Student Name     : A.Manas
Roll Number      : 23VE1A6601
Section          : A
Experiment No    : 1
Experiment Title : Design and Implementation of a Deep Neural Network Using Single Variable Linear Regression
Submission ID    : f55c2418
Epoch 1/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 238.3133
Epoch 2/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 52.6434
Epoch 3/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 44.3211
Epoch 4/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 40.9329
Epoch 5/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 40.8272
Epoch 6/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 46.6195
Epoch 7/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 36.4272
Epoch 8/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 41.3001
Epoch 9/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 42.5544
Epoch 10/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 36.3560
Epoch 11/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 46.5068
Epoch 12/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.8015
Epoch 13/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 36.2086
Epoch 14/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.9005
Epoch 15/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.0008
Epoch 16/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.2541
Epoch 17/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 47.5368
Epoch 18/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.2756
Epoch 19/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 49.1476
Epoch 20/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 42.6022
```

```
Epoch 21/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 45.3735
Epoch 22/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 35.9806
Epoch 23/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 42.5034
Epoch 24/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 44.0267
Epoch 25/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.7841
Epoch 26/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 40.1876
Epoch 27/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 41.3214
Epoch 28/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 46.7956
Epoch 29/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 41.3469
Epoch 30/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.4778
Epoch 31/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 45.9277
Epoch 32/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 46.4293
Epoch 33/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.2542
Epoch 34/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.7503
Epoch 35/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 56.0627
Epoch 36/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.9679
Epoch 37/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.5688
Epoch 38/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 41.8502
Epoch 39/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 44.9113
Epoch 40/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 42.4070
Epoch 41/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.7285
Epoch 42/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 42.8847
Epoch 43/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 33.9196
```

```
Epoch 44/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 41.2457
Epoch 45/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 43.0544
Epoch 46/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.2692
Epoch 47/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 44.7310
Epoch 48/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.4734
Epoch 49/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.4081
Epoch 50/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 41.4418
Epoch 51/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.6013
Epoch 52/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 42.7128
Epoch 53/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 49.4060
Epoch 54/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 34.9074
Epoch 55/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.6249
Epoch 56/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.1805
Epoch 57/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 48.5686
Epoch 58/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 40.7187
Epoch 59/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 47.6340
Epoch 60/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.8497
Epoch 61/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 41.8874
Epoch 62/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 41.3884
Epoch 63/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 49.3072
Epoch 64/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 37.9595
Epoch 65/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 38.7266
Epoch 66/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 36.7106
```

```
Epoch 67/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 40.0880
Epoch 68/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 42.4366
Epoch 69/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 40.6139
Epoch 70/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 41.1537
Epoch 71/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 41.3927
Epoch 72/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 45.1802
Epoch 73/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 50.4067
Epoch 74/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 45.5519
Epoch 75/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.8879
Epoch 76/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.0334
Epoch 77/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 41.0297
Epoch 78/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 48.5693
Epoch 79/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.1003
Epoch 80/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.6024
Epoch 81/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 34.2116
Epoch 82/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 36.5214
Epoch 83/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.8677
Epoch 84/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.2829
Epoch 85/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.5172
Epoch 86/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.1776
Epoch 87/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 47.3886
Epoch 88/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 34.8500
Epoch 89/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 37.1187
```

```
Epoch 90/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 42.8309
Epoch 91/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.1998
Epoch 92/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.3145
Epoch 93/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.5572
Epoch 94/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 44.8120
Epoch 95/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 48.4656
Epoch 96/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 36.3916
Epoch 97/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 42.7927
Epoch 98/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 42.7941
Epoch 99/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.1964
Epoch 100/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 39.3827
RMSE: 0.02862285600991514
```




**.Result:** A python program to build a Deep Neural Network Model Using Linear Regression with a Single Variable is executed and output is verified


Input