AD18511- DEEP LEARNING LABORATORY

DATE: **LINEAR REGRESSION USING TENSORFLOW**

EX.NO: 2(a)

AIM:

To implement Linear Regression using Tensorflow from scratch

DESCRIPTION:

Linear regression is a statistical method used to model and analyze the relationship between a dependent variable and one or more independent variables. It aims to find the best-fitting straight line (a linear equation) that represents the pattern of the data, allowing for predictions and understanding the nature of the connection between the variables.

The linear regression equation can be written as: y = $\hat{\beta}_0$ + $\hat{\beta}_1$ *x

where:

y is the dependent variable (the one being predicted or explained).
x is the independent variable (the one used for making predictions).
b0 is the y-intercept.
b1 is the slope or coefficient of the independent variable.

The least squares estimates of β_0 and β_1 are:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1\bar{X}$$

The primary goal of linear regression is to estimate the values of b0 and b1 such that the line fits the data points as closely as possible. This is achieved by minimizing the sum of the squared differences between the predicted y-values and the actual y-values, known as the "method of least squares."

PROGRAM:

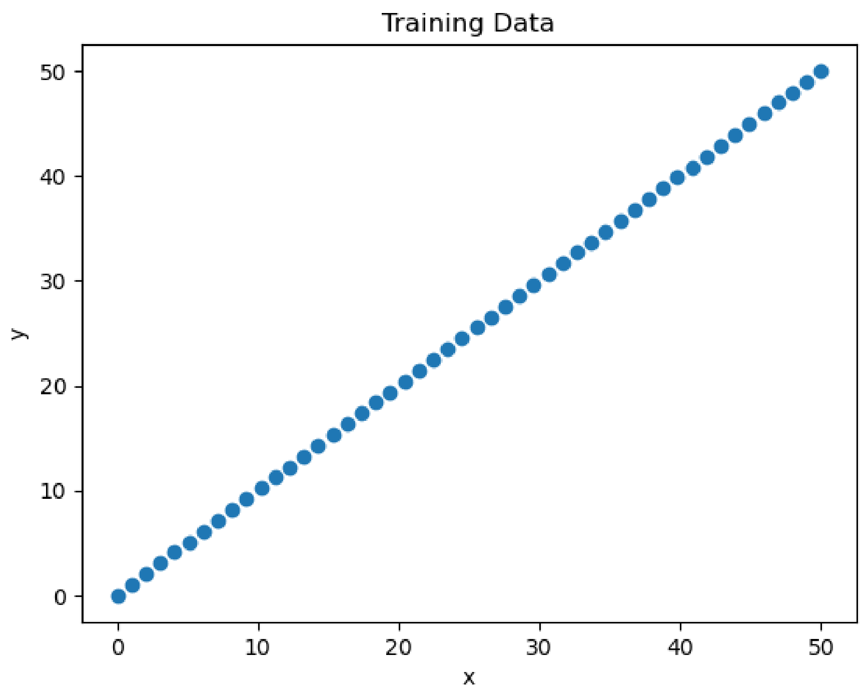
```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
np.random.seed(101)

#Generates random linear data points from 1 to 50
x = np.linspace(0,50,50) #(start,end,no of points to be generated)
y = np.linspace(0,50,50)
x,y,n
```

OUTPUT:

```
(array([ 0.        , 1.02040816, 2.04081633, 3.06122449,  4.08163265,
         5.10204082, 6.12244898, 7.14285714, 8.16326531, 9.18367347,
        10.20408163, 11.2244898 , 12.24489796, 13.26530612, 14.28571429,
        15.30612245, 16.32653061, 17.34693878, 18.36734694, 19.3877551 ,
        20.40816327, 21.42857143, 22.44897959, 23.46938776, 24.48979592,
        25.51020408, 26.53061224, 27.55102041, 28.57142857, 29.59183673,
        30.6122449 , 31.63265306, 32.65306122, 33.67346939, 34.69387755,
        35.71428571, 36.73469388, 37.75510204, 38.7755102 , 39.79591837,
        40.81632653, 41.83673469, 42.85714286, 43.87755102, 44.89795918,
        45.91836735, 46.93877551, 47.95918367,48.97959184, 50.        ]),
array([ 0.        , 1.02040816, 2.04081633, 3.06122449,  4.08163265,
         5.10204082, 6.12244898, 7.14285714, 8.16326531, 9.18367347,
        10.20408163, 11.2244898 , 12.24489796, 13.26530612, 14.28571429,
        15.30612245, 16.32653061, 17.34693878, 18.36734694, 19.3877551 ,
        20.40816327, 21.42857143, 22.44897959, 23.46938776, 24.48979592,
        25.51020408, 26.53061224, 27.55102041, 28.57142857, 29.59183673,
        30.6122449 , 31.63265306, 32.65306122, 33.67346939, 34.69387755,
        35.71428571, 36.73469388, 37.75510204, 38.7755102 , 39.79591837,
        40.81632653, 41.83673469, 42.85714286, 43.87755102, 44.89795918,
        45.91836735, 46.93877551, 47.95918367,48.97959184, 50.        ]))
```

```
plt.scatter(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.title("Training Data")
plt.show()
```

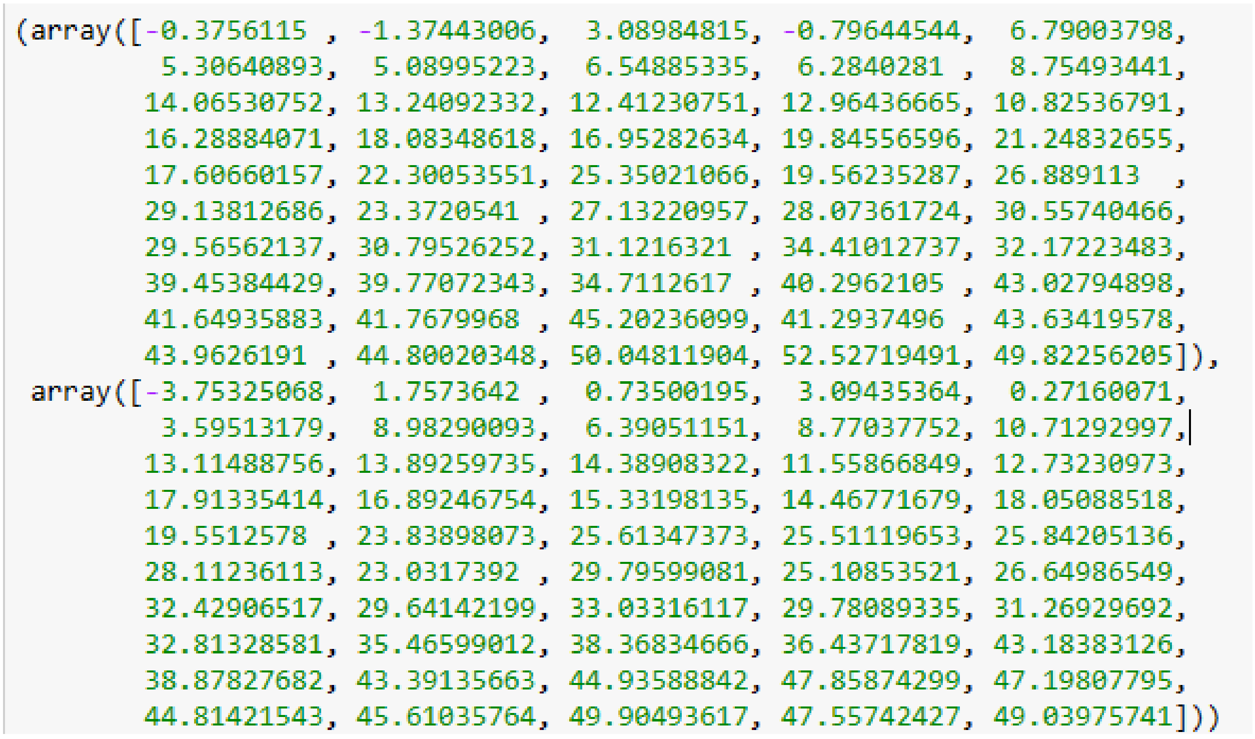
OUTPUT:



```
#Adding noise to the datapoints
x += np.random.uniform(-4,4,50)
y += np.random.uniform(-4,4,50)
n = len(x) #Number of Data Points
```

x,y

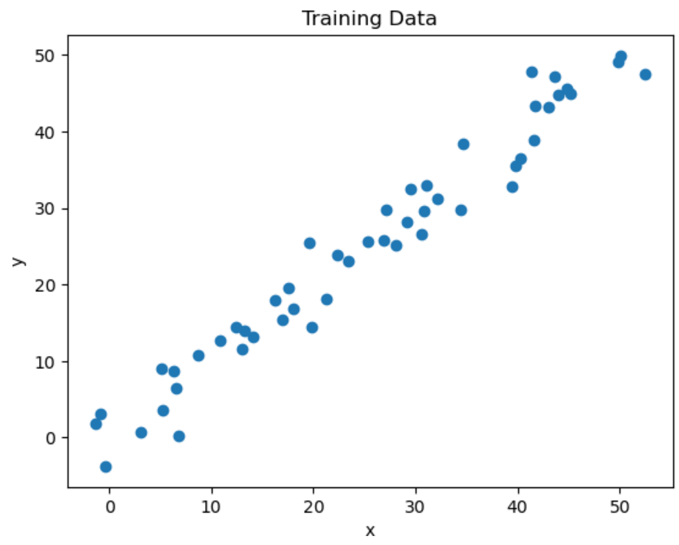
OUTPUT:



```
(array([-0.3756115 , -1.37443006,  3.08984815, -0.79644544,  6.79003798,
         5.30640893,  5.08995223,  6.54885335,  6.2840281 ,  8.75493441,
        14.06530752, 13.24092332, 12.41230751, 12.96436665, 10.82536791,
        16.28884071, 18.08348618, 16.95282634, 19.84556596, 21.24832655,
        17.60660157, 22.30053551, 25.35021066, 19.56235287, 26.889113  ,
        29.13812686, 23.3720541 , 27.13220957, 28.07361724, 30.55740466,
        29.56562137, 30.79526252, 31.1216321 , 34.41012737, 32.17223483,
        39.45384429, 39.77072343, 34.7112617 , 40.2962105 , 43.02794898,
        41.64935883, 41.7679968 , 45.20236099, 41.2937496 , 43.63419578,
        43.9626191 , 44.80020348, 50.04811904, 52.52719491, 49.82256205]),
 array([-3.75325068,  1.7573642 ,  0.73500195,  3.09435364,  0.27160071,
         3.59513179,  8.98290093,  6.39051151,  8.77037752, 10.71292997,
        13.11488756, 13.89259735, 14.38908322, 11.55866849, 12.73230973,
        17.91335414, 16.89246754, 15.33198135, 14.46771679, 18.05088518,
        19.5512578 , 23.83898073, 25.61347373, 25.51119653, 25.84205136,
        28.11236113, 23.0317392 , 29.79599081, 25.10853521, 26.64986549,
        32.42906517, 29.64142199, 33.03316117, 29.78089335, 31.26929692,
        32.81328581, 35.46599012, 38.36834666, 36.43717819, 43.18383126,
        38.87827682, 43.39135663, 44.93588842, 47.85874299, 47.19807795,
        44.81421543, 45.61035764, 49.90493617, 47.55742427, 49.03975741]))
```

```
#plot of training data
plt.scatter(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.title("Training data")
plt.show()
```

OUTPUT:



```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
X=tf.placeholder("float")
```

```
Y=tf.placeholder("float")

W=tf.Variable(np.random.randn(),name="W")
b=tf.Variable(np.random.randn(),name="b")

learning_rate = 0.01
training_epochs = 1000

#Hypothesis
y_pred = tf.add(tf.multiply(X,W),b)

#MSE Cost Function
cost = tf.reduce_sum(tf.pow(y_pred-Y,2))/(2*n)

#Gradient Descent Optimizer
optimizer =
tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

#Global Variable Initializer
init = tf.global_variables_initializer()

#Starting TensorFlow Session
with tf.Session() as sess:
    sess.run(init)
    for epoch in range(training_epochs):
        for(_x,_y) in zip(x,y):
            sess.run(optimizer, feed_dict = {X : _x, Y : _y})

        if(epoch + 1) % 50 == 0:
            c = sess.run(cost, feed_dict = {X : _x, Y : _y})
            print("Epoch",(epoch+1),": cost =", c, "W =", sess.run(W),
"b=",sess.run(b))

    #Storing necessary values to be used outside the session
    training_cost = sess.run(cost, feed_dict = {X : _x, Y : _y})
    weight = sess.run(W)
    bias = sess.run(b)
```

OUTPUT:

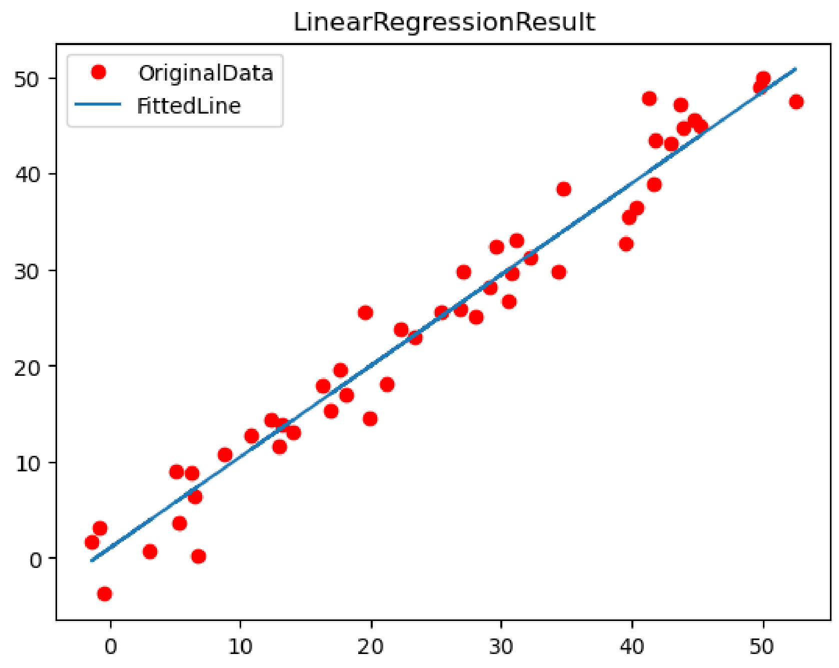
```
Epoch 800 :cost= 0.03479725 W= 0.7661446 b= 8.090939
Epoch 850 :cost= 0.03498694 W= 0.76410013 b= 8.201936
Epoch 900 :cost= 0.03515396 W= 0.76230603 b= 8.299341
Epoch 950 :cost= 0.03530045 W= 0.760731 b= 8.38486
Epoch 1000 :cost= 0.03542972 W= 0.7593485 b= 8.459916
```

```
#Calculating the predictions

predictions=weight*x + bias
```

```
print("Training cost =",training_cost,"Weight=",weight,"bias=",bias,'\n')
```

OUTPUT:

```
Training cost = 0.03542972
Weight= 0.7593485
bias= 8.459916
#Plotting the results
plt.plot(x,y,'ro',label='Original data')
plt.plot(x,prediction,label)
```



RESULT:

The given linear regression model is implemented using Tensorflow. The model is trained and tested and the line of regression is plotted for the data

DATE:

AD18511- DEEP LEARNING LABORATORY
LINEAR REGRESSION USING SCIKIT LEARN

EX.NO: 2(b)

AIM:

To implement Linear Regression using Tensorflow from scratch.

DESCRIPTION:

Linear regression is a statistical method used to model and analyze the relationship between a dependent variable and one or more independent variables. It aims to find the best-fitting straight line (a linear equation) that represents the pattern of the data, allowing for predictions and understanding the nature of the connection between the variables.
The linear regression equation can be written as:

y = �0 + �1 * x

where:

y is the dependent variable (the one being predicted or explained).
x is the independent variable (the one used for making predictions).
b0 is the y-intercept.
b1 is the slope or coefficient of the independent variable.

The least squares estimates of β_0 and β_1 are:

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \\ \hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{X}\end{aligned}$$

The primary goal of linear regression is to estimate the values of b0 and b1 such that the line fits the data points as closely as possible. This is achieved by minimizing the sum of the squared differences between the predicted y-values and the actual y-values, known as the "method of least squares."

PROGRAM:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
dataset=pd.read_csv('/content/Salary_Data.csv')
dataset.head()#in json format
```

OUTPUT:

```
[{"index":0,"YearsExperience":"1.1","Salary":"39343.0"},{"index":1,"YearsExperience":"1.3","
Salary":"46205.0"},{"index":2,"YearsExperience":"1.5","Salary":"37731.0"},{"index":3,"Years
Experience":"2.0","Salary":"43525.0"},{"index":4,"YearsExperience":"2.2","Salary":"39891.0"}
]
```

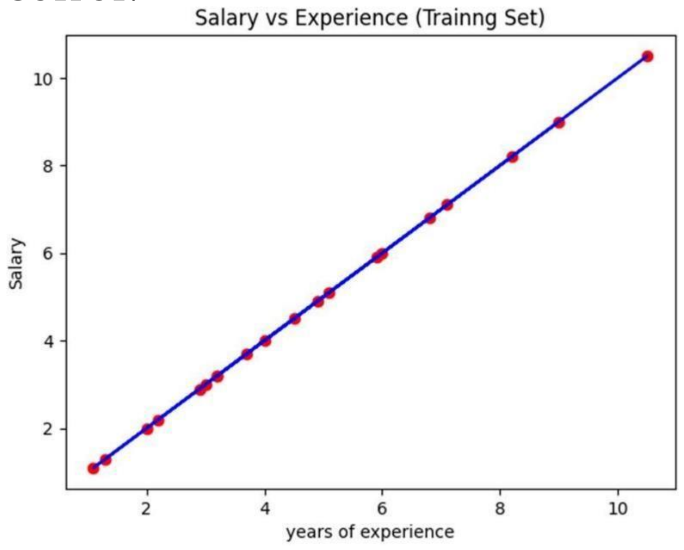
```
#data preprocessing
X=dataset.iloc[:,:-1].values #independent variable array x =all rows and columns
y=dataset.iloc[:,:1].values #dependent variable vector y= all rows and last column alone
```

OUTPUT:

```
array([[ 1.1], [ 1.3], [ 1.5], [ 1.5], [ 2. ],[ 2.2], [ 2.9], [ 3. ], [ 3.2], [ 3.2], [ 3.7], [ 3.9], [ 4. ], [ 4. ],
[ 4.1], [ 4.5], [ 4.9], [ 5.1], [ 5.3], [ 5.9], [ 6. ], [ 6.8], [ 7.1], [ 7.9], [ 8.2], [ 8.7], [ 9. ], [ 9.5], [9.6],
[10.3], [10.5]])
```

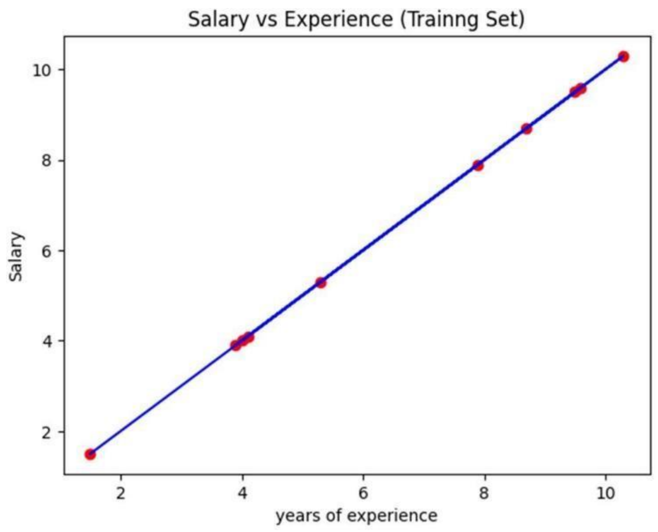
```
#splitting the dataset
from sklearn.model_selection
import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1/3,random_state=0) #fitting the regression model
from sklearn.linear_model
import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)

#Predicting the test results
y_pred=regressor.predict(X_test)

#visualizing the results
#plot for the TRAIN
get_ipython().run_line_magic('matplotlib','inline')
plt.scatter(X_train,y_train,color='red')#plotting the observation line plt.plot(X_train,regressor.predict(X_train),color='blue')
plt.title("Salary vs Experience (Trainng Set)")#stating the title of the graph
plt.xlabel("years of experience")#adding name of x-axis
plt.ylabel("Salary")
plt.show()
```

OUTPUT:



```
#visualizing the results #plot for the train
plt.scatter(X_test,y_test,color='red')#plotting the observation line
plt.plot(X_test,regressor.predict(X_test),color='blue')
plt.title("Salary vs Experience (Trainng Set)")#stating the title of the graph

plt.xlabel("years of experience")#adding name of x-axis
plt.ylabel("Salary")
plt.show()
```

OUTPUT:



RESULT:

The given linear regression model is implemented using scikit learn module. The model is trained and tested and the line of regression is plotted for the data.