Experiment 2

Shashwat Shah
60004220126
TY Btech Comps B

Aim: To perform linear regression and find errors the model is associated with.

Theory: Linear regression is one of the most popular supervised machine learning algorithms. Its a statistical method of used for predictive analysis. Linear regression makes prediction for continuous /real or numeric variables such as sales, salary, age, etc Linear regression shows a linear relationship between the variables by giving a sloped straight line.

Mathematically, we can represent a linear regression as $y = b_0 + b_1 x + \varepsilon$

y = Dependant variable

x = Independent variable

b_0 = Intercept of line

b_1 = Linear regression coefficient

ε - random error

The values for x & y variable are training datasets for linear regression model representation.

Procedure :

In the procedure we aim to simply perform Simple linear regression using the latest least square method without relying on the scikit library. To initiate the process we start by calculating the mean values X & Y. Next we calculate the deviations of each data point from their respective means. Following this we get the slope (m) of the regression line is determined by diving the sum of squared deviation from the mean of X. Subsequently, the intercept (b) is computed using mean values of X & Y along with the calculated slope. The regression slope equation is formed as Y = mx+b, providing for corresponding X values optionally, the results can be varied.

Observed / Discussed result.
The provided python code implemented without external libraries conduct linear regression on a dataset. The linear regression on a dataset. The linear regression calculate slope (m), intercept (b), mean square error (MSE) & R-squared (R^2).

Conclusion:
Hence, linear regression is a very useful predictive machine learning algorithm.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Computer Engineering Department

COURSE NAME: Machine Learning **CLASS:** TY Year B.Tech

NAME: Shashwat Shah **BATCH:** C22

EXPERIMENT NO. 2

AIM / OBJECTIVE:

To perform linear regression and find the error associated with the model.

DESCRIPTION OF EXPERIMENT:

Linear regression is one of the easiest and most popular Supervised Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.The linear regression model provides a sloped straight line representing the relationship between the variables.Cleaning Data in PythonWe will now separate the numeric columns from the categorical columns.

Mathematically, we can represent a linear regression as: $\mathbf{y= b_0+b_1x+ \varepsilon}$

Here,

y= Dependent Variable (Target Variable)

x= Independent Variable (predictor Variable)

b_0= intercept of the line (Gives an additional degree of freedom)

b_1 = Linear regression coefficient (scale factor to each input value).

ε = random error

The values for x and y variables are training datasets for Linear Regression model representation

The different values for weights or coefficient of lines (b_0, b_1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line. Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing. We can use the cost function to find the accuracy of the **mapping function**, which maps the input variable to the output variable. This mapping function is also known as **Hypothesis**

function. For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values. It can be written as:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (b_1x_i + b_0))^2$$

where,
N=Total number of observation
y_i = Actual value
$(b_1x_i+b_0)$= Predicted value.

Linear regression using Least Square Method

We have linear regression equation as **$y= b_0+b_1x$**

Using least square method,

$$\boldsymbol{b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}}$$

$$\boldsymbol{b_0 = \bar{y} - b_1\bar{x}}$$

PROCEDURE:

1. Describe the procedure that is used to perform Linear regression using Least Square Method carry out the experiment step-by-step for simple linear regression for following dataset without using scikit library. Describe every line of code with the proper interpretation of the output.

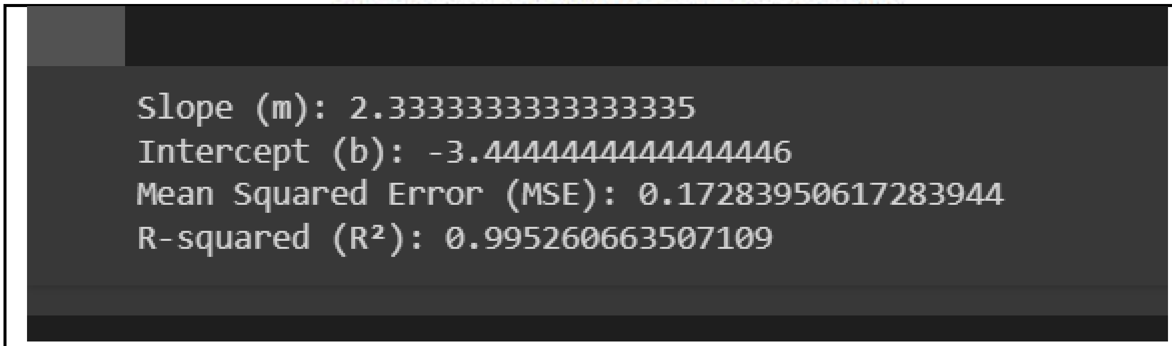
X	2	3	4	5	6	7	8	9	10
Y	1	3	6	9	11	13	15	17	20

2. Perform Regression with respect to one dataset of your choice and discuss results of all the steps.

Program

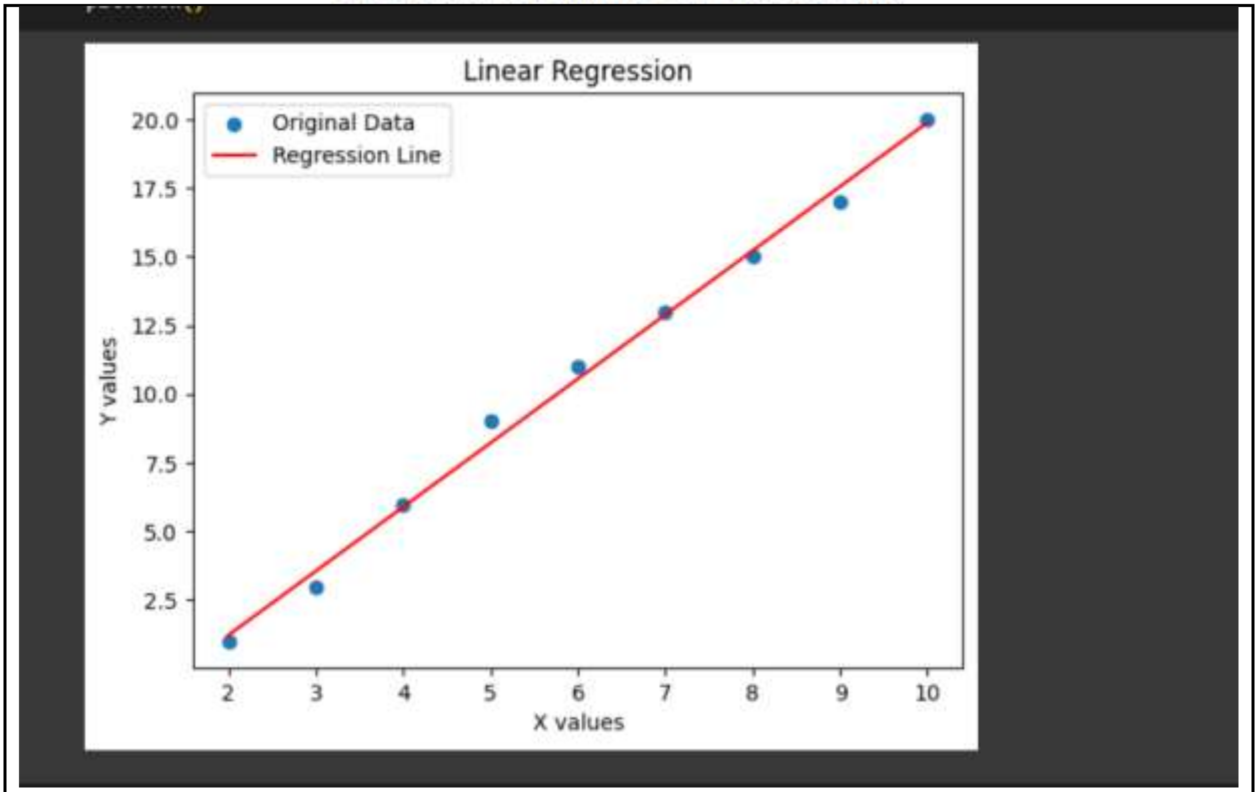
```
import matplotlib.pyplot as plt
```

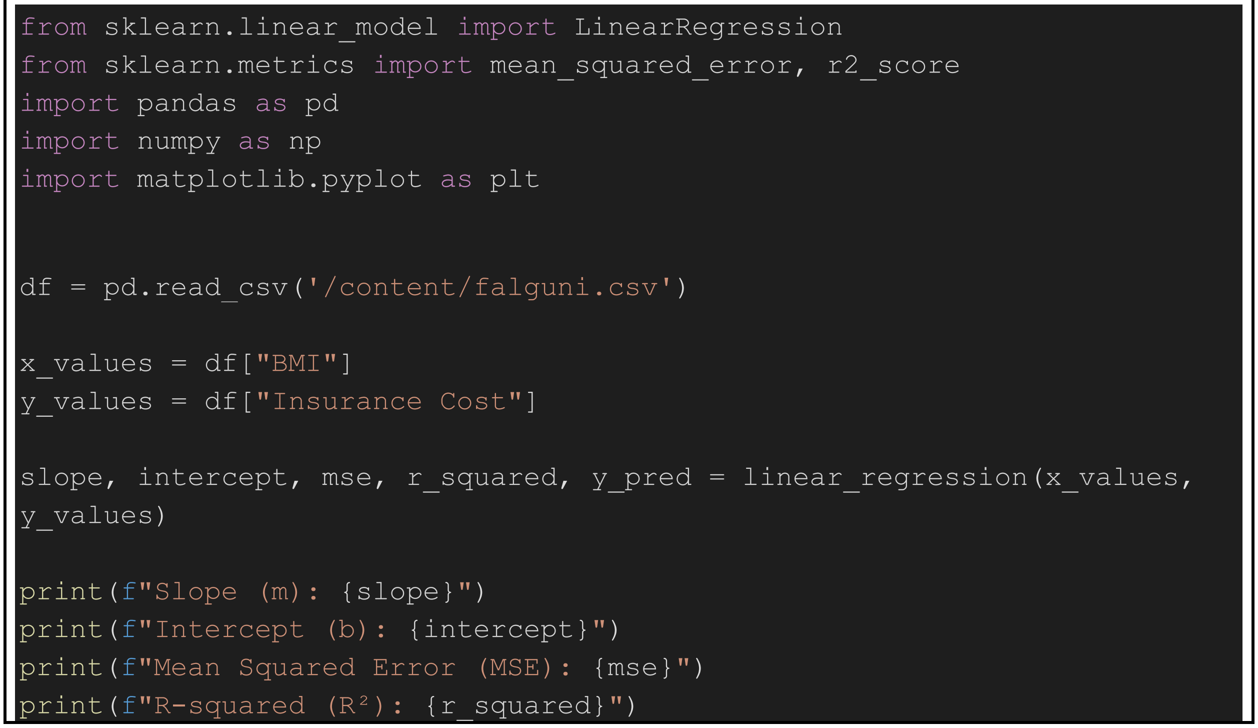
```
def linear_regression(x_values, y_values):
    n = len(x_values)
    mean_x = sum(x_values) / n
    mean_y = sum(y_values) / n
    numerator = sum((x - mean_x) * (y - mean_y) for x, y in
zip(x_values, y_values))
    denominator = sum((x - mean_x) ** 2 for x in x_values)
    m = numerator / denominator
    b = mean_y - m * mean_x
    y_pred = [m * x + b for x in x_values]
    mse = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values, y_pred))
/ n
    ss_total = sum((y - mean_y) ** 2 for y in y_values)
    ss_residual = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values,
y_pred))
    r_squared = 1 - (ss_residual / ss_total)
    return m, b, mse, r_squared, y_pred


def print_regression_results(slope, intercept, mse, r_squared):
    print(f"Slope (m): {slope}")
    print(f"Intercept (b): {intercept}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R-squared (R²): {r_squared}")


# Example usage:
x_values = [2, 3, 4, 5, 6, 7, 8, 9, 10]
y_values = [1, 3, 6, 9, 11, 13, 15, 17, 20]

slope, intercept, mse, r_squared, y_pred = linear_regression(x_values,
y_values)


print_regression_results(slope, intercept, mse, r_squared)
```



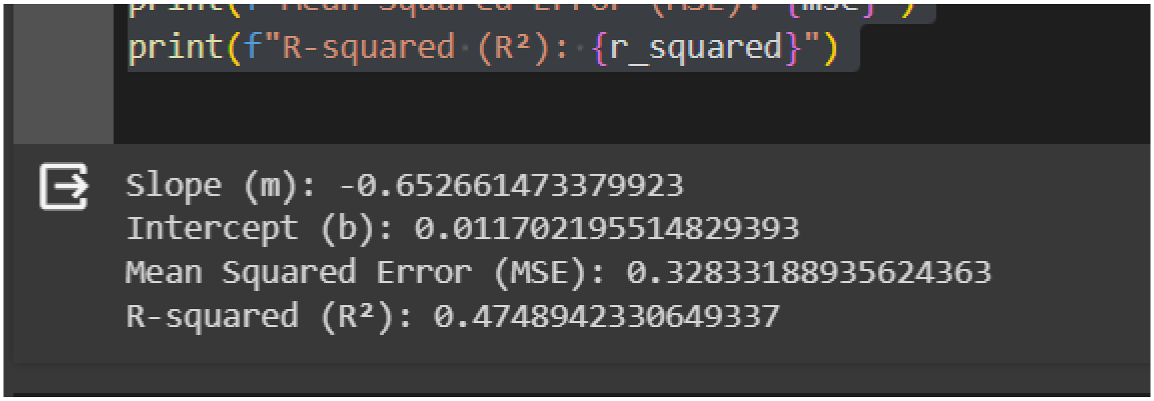
```
Slope (m): 2.3333333333333335
Intercept (b): -3.4444444444444446
Mean Squared Error (MSE): 0.17283950617283944
R-squared (R²): 0.995260663507109
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import matplotlib.pyplot as plt

def linear_regression(x_values, y_values):
    x_values = np.array(x_values).reshape(-1, 1)
    y_values = np.array(y_values)

    model = LinearRegression()
    model.fit(x_values, y_values)
    y_pred = model.predict(x_values)

    mse = mean_squared_error(y_values, y_pred)
    r_squared = r2_score(y_values, y_pred)

    return model.coef_[0], model.intercept_, mse, r_squared, y_pred

x_values = [2,3,4,5,6,7,8,9,10]
y_values = [1,3,6,9,11,13,15,17,20]

slope, intercept, mse, r_squared, y_pred = linear_regression(x_values,
y_values)

print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")
```





```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


df = pd.read_csv('/content/falguni.csv')

x_values = df["BMI"]
y_values = df["Insurance Cost"]

slope, intercept, mse, r_squared, y_pred = linear_regression(x_values,
y_values)

print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")
```



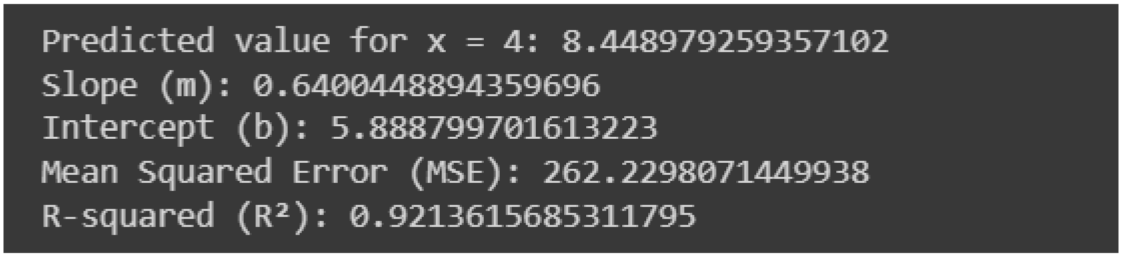
```
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")
```

```
Slope (m): -0.652661473379923
Intercept (b): 0.011702195514829393
Mean Squared Error (MSE): 0.32833188935624363
R-squared (R²): 0.4748942330649337
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


df = pd.read_csv('/content/falguni.csv')
x_values = df["BMI"]
y_values = df["Insurance Cost"]

def linear_regression(x_values, y_values):
    n = len(x_values)
    mean_x = sum(x_values) / n
    mean_y = sum(y_values) / n
    numerator = sum((x - mean_x) * (y - mean_y) for x, y in
zip(x_values, y_values))
    denominator = sum((x - mean_x) ** 2 for x in x_values)
    m = numerator / denominator
    b = mean_y - m * mean_x
    y_pred = [m * x + b for x in x_values]
    mse = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values, y_pred))
/ n
    ss_total = sum((y - mean_y) ** 2 for y in y_values)
    ss_residual = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values,
y_pred))
    r_squared = 1 - (ss_residual / ss_total)
    return m, b, mse, r_squared, y_pred
```

```
def print_regression_results(slope, intercept, mse, r_squared):
    print(f"Slope (m): {slope}")
    print(f"Intercept (b): {intercept}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R-squared (R²): {r_squared}")

    slope, intercept, mse, r_squared, y_pred =
linear_regression(x_values, y_values)


print_regression_results(slope, intercept, mse, r_squared)
```

```
# x = 4 predicted with libaries


from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


df = pd.read_csv('/content/datasetcost.csv')

x_values = df["X"]
y_values = df["Y"]
x_to_predict = 4
predicted_y = slope * x_to_predict + intercept

print(f"Predicted value for x = {x_to_predict}: {predicted_y}")

slope, intercept, mse, r_squared, y_pred = linear_regression(x_values,
y_values)

print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")
```



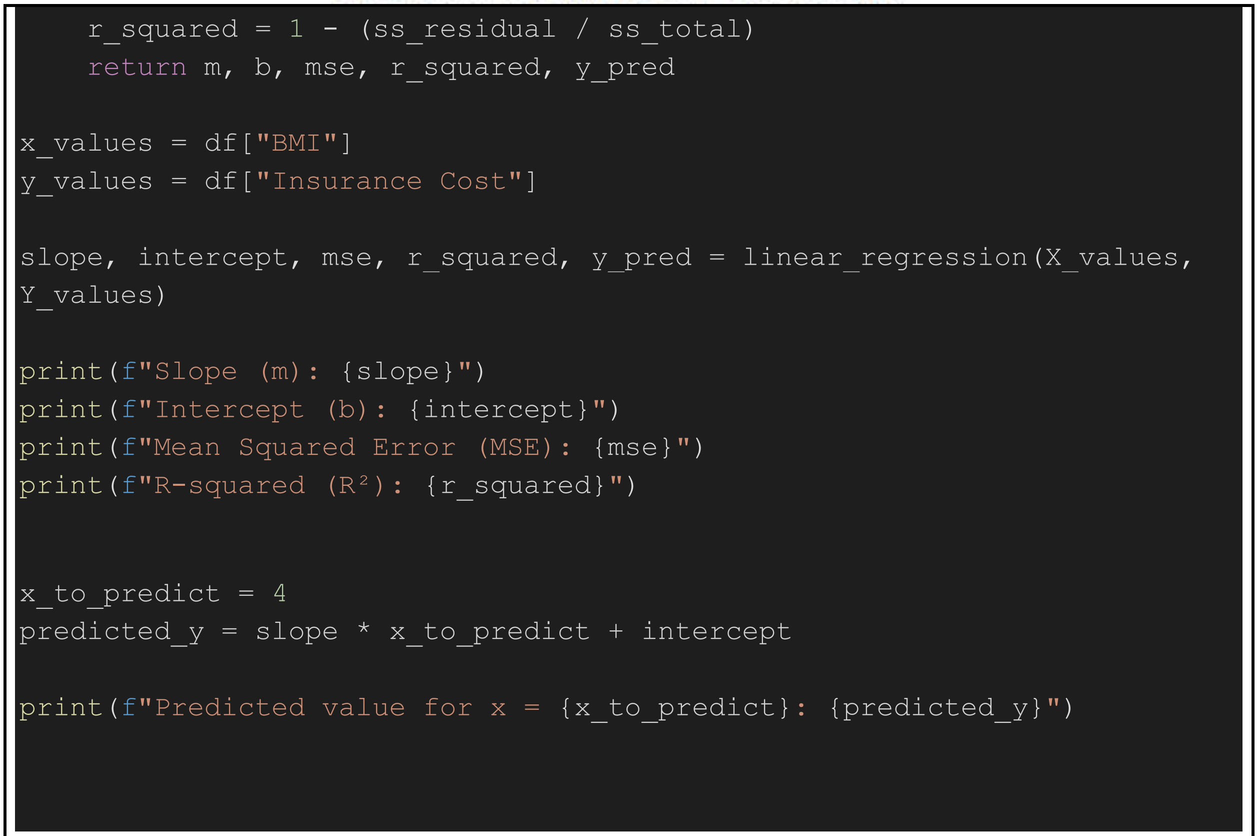
```
Predicted value for x = 4: 8.448979259357102
Slope (m): 0.6400448894359696
Intercept (b): 5.888799701613223
Mean Squared Error (MSE): 262.2298071449938
R-squared (R²): 0.9213615685311795
```

OBSERVATIONS / DISCUSSION OF RESULT:

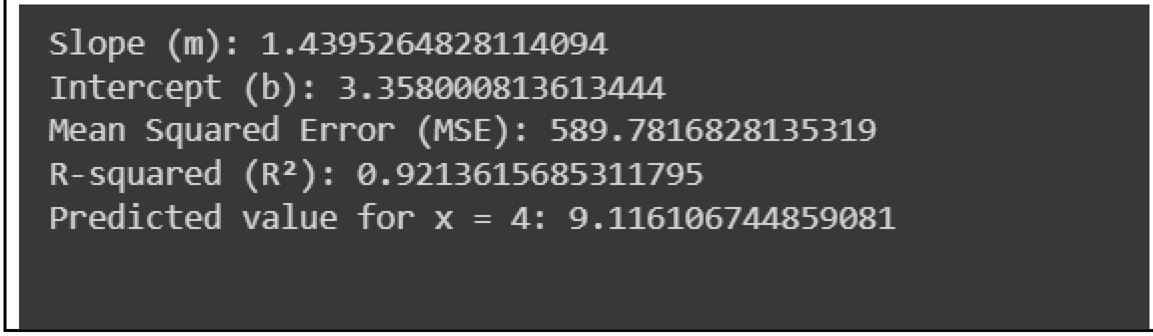
1. Find predicted value of y using Linear Regression for one epoch and RMSE for x =4.

X	2	3	4	5	6	7	8	9	10
Y	1	3	6	9	10	13	14	17	21

```
# without libaries pridicted



import pandas as pd

df = pd.read_csv('/content/Linear Regression - Sheet1.csv')
def linear_regression(x_values, y_values):
    n = len(x_values)
    mean_x = sum(x_values) / n
    mean_y = sum(y_values) / n

    numerator = sum((x - mean_x) * (y - mean_y) for x, y in
zip(x_values, y_values))
    denominator = sum((x - mean_x) ** 2 for x in x_values)

    m = numerator / denominator
    b = mean_y - m * mean_x

    y_pred = [m * x + b for x in x_values]
    mse = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values, y_pred))
/ n

    ss_total = sum((y - mean_y) ** 2 for y in y_values)
    ss_residual = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values,
y_pred))
```



```
    r_squared = 1 - (ss_residual / ss_total)
    return m, b, mse, r_squared, y_pred

x_values = df["BMI"]
y_values = df["Insurance Cost"]

slope, intercept, mse, r_squared, y_pred = linear_regression(X_values,
Y_values)

print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")


x_to_predict = 4
predicted_y = slope * x_to_predict + intercept

print(f"Predicted value for x = {x_to_predict}: {predicted_y}")
```



```
Slope (m): 1.4395264828114094
Intercept (b): 3.358000813613444
Mean Squared Error (MSE): 589.7816828135319
R-squared (R²): 0.9213615685311795
Predicted value for x = 4: 9.116106744859081
```

CONCLUSION:

Linear Regression using least squared method was implemented from scratch and using the Libraries

REFERENCES:

(List the references as per format given below and citations to be included the document)

[1] Ponniah P., “Data Warehousing: Fundamentals for IT Professionals”, 2nd Edition, Wiley India, 2013.

[2] Ageed, Z. S., Zeebaree, S. R., Sadeeq, M. M., Kak, S. F., Yahia, H. S., Mahmood, M. R., & Ibrahim, I. M. (2021), “Comprehensive survey of big data mining approaches in cloud systems”, Qubahan Academic Journal, 1(2), 29-38.

Website References:

Author's Last Name, First Initial. Middle Initial. (Date of Publication or Update). Title of work. Site name. Retrieved Month Day, Year, from URL from Homepage

[3] U.S. Census Bureau. U.S. and world population clock. U.S. Department of Commerce. Retrieved July 3, 2019, from https://www.census.gov/popclock.