DATE:

ML EXP-2

NAME: Prerna Jadhav
Sap ID: 60004220127
Batch: C22
Branch: Computer Engineering
Course: Machine Learning

AIM: To perform linear regression & find the error associated with the model.

DESCRIPTION OF EXPERIMENT: Linear Regression is one of the easiest and most popular supervised ML algorithm. 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, pages etc.

→ It shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called linear regression.

→ Since, linear regression shows linear relationship which means it finds how the value of the dependent value is changing accordingly to the value of the independent variable.

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

Here, y = Dependent variable (Target variable)
x = Independent variable (Predictor variable)
b_0 = Intercept of line
b_1 = Linear regression coefficient
ε = random error.

DATE:

→ The values for weights or coefficient of line (b_0, b_1) gives different line of regression and the cost function is used to estimate the value of the coefficient for the best fit line.

→ The cost function optimizes the regression coeffient 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 helps the i/p variable to o/p variable. This mapping function is also known as Hypothesis function.

→ for linear Regression, we use the mean square error cost function, which is the average of squared error occured between the predicted value & actual values.

→ It can be written as:-

$$MSE = \frac{1}{N} \sum_{i=1}^{n} (y - (b_1 x_i + b_0))^2$$

where, N = Total no. of observation

y = Actual value

$(b_1 x + b_0)$ = predicted value

→ Linear regression using least square Method

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

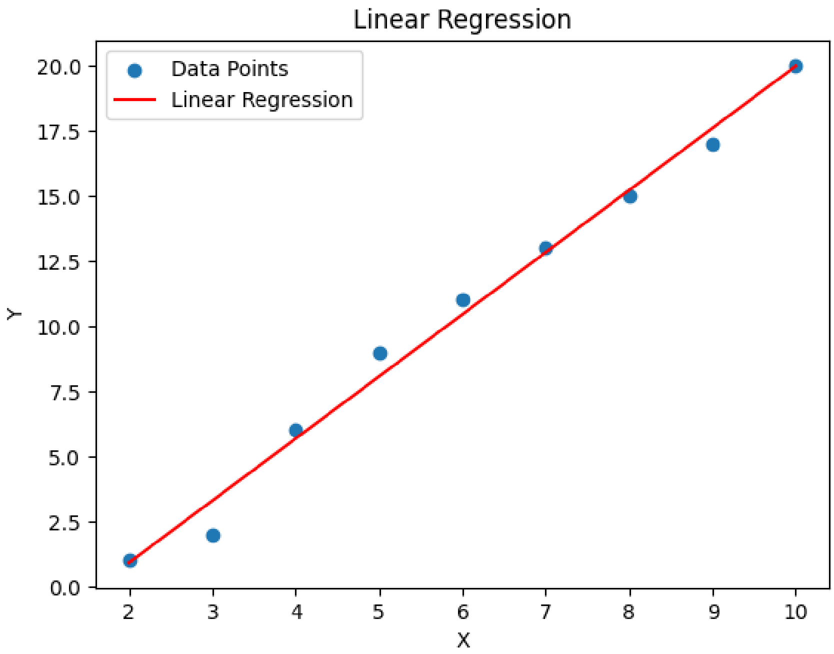
$$b_0 = \bar{y} - b_1 \bar{x}$$

CONCLUSION: Thus, we studied and implemented the Linear Regression.

- Name: Prerna Sunil Jadhav
- Sap ID: 60004220127
- Batch: C22
- Branch: Computer Engineering
- Course: Machine Learning
- Experiment 2: Linear Regression

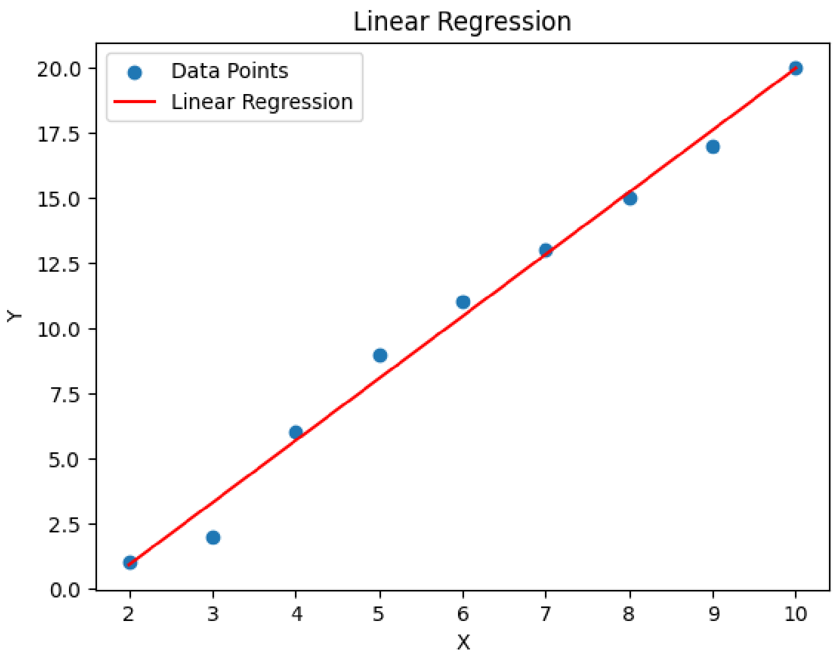
```
#part1: with lib
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

X = np.array([2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)
Y = np.array([1, 2, 6, 9, 11, 13, 15, 17, 20])

model = LinearRegression()

model.fit(X, Y)

slope = model.coef_[0]
intercept = model.intercept_

print(f"Slope: {slope}")
print(f"Intercept: {intercept}")

Y_pred = model.predict(X)

plt.scatter(X, Y, label='Data Points')
plt.plot(X, Y_pred, color='red', label='Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.title('Linear Regression')
plt.show()
```

```
Slope: 2.3833333333333337
Intercept: -3.855555555555558
```



```
#part1-without lib
import numpy as np
import matplotlib.pyplot as plt

X = np.array([2, 3, 4, 5, 6, 7, 8, 9, 10])
Y = np.array([1, 2, 6, 9, 11, 13, 15, 17, 20])

mean_X = np.mean(X)
mean_Y = np.mean(Y)

numerator = np.sum((X - mean_X) * (Y - mean_Y))
denominator = np.sum((X - mean_X) ** 2)
slope = numerator / denominator
intercept = mean_Y - slope * mean_X

Y_pred = slope * X + intercept

mse = np.mean((Y - Y_pred) ** 2)

ss_total = np.sum((Y - mean_Y) ** 2)
ss_residual = np.sum((Y - Y_pred) ** 2)
r_squared = 1 - (ss_residual / ss_total)

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

plt.scatter(X, Y, label='Data Points')
plt.plot(X, Y_pred, color='red', label='Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.title('Linear Regression')
plt.show()
```

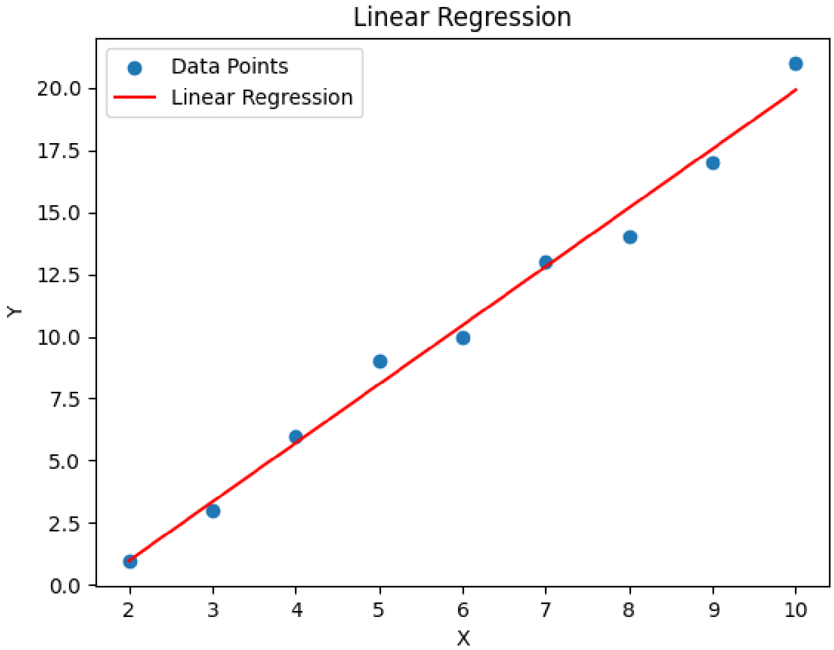
```
Slope: 2.3833333333333333
Intercept: -3.8555555555555556
Mean Squared Error: 0.3783950617283949
R-squared: 0.9901065203357005
```



```
# Part 2 : dataset
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

df = pd.read_csv("../content/train.csv")

X = df['X'].value
Y = df['Y'].value

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

model = LinearRegression()

model.fit(X_train, Y_train)

print(model.score(X_test, Y_test))

Y_pred = model.predict(X_test)

plt.scatter(X_test, Y_test, label='Test Data')
plt.plot(X_test, Y_pred, color='red', label='Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.title('Linear Regression on Test Data')
plt.show()
```

```
# Part 3-libraries

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

X = np.array([2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)
Y = np.array([1, 3, 6, 9, 10, 13, 14, 17, 21])

model = LinearRegression()

model.fit(X, Y)

slope = model.coef_[0]
intercept = model.intercept_

print(f"Slope: {slope}")
print(f"Intercept: {intercept}")

Y_pred = model.predict(X)

X_to_predict = np.array([[4]])
Y_pred_X4 = model.predict(X_to_predict)
print(f"Predicted Y for X=4: {Y_pred_X4[0]}")

mse = mean_squared_error(Y, Y_pred)
print(f"Mean Squared Error (MSE): {mse}")

r_squared = r2_score(Y, Y_pred)
print(f"R-squared: {r_squared}")

plt.scatter(X, Y, label='Data Points')
plt.plot(X, Y_pred, color='red', label='Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.title('Linear Regression')
plt.show()
```

```
Slope: 2.3666666666666667
Intercept: -3.7555555555555546
Predicted Y for X=4: 5.711111111111112
Mean Squared Error (MSE): 0.46172839506172825
R-squared: 0.9877857609405617
```




```
# Part 3-without libraries

import numpy as np
import matplotlib.pyplot as plt

X = np.array([2, 3, 4, 5, 6, 7, 8, 9, 10])
Y = np.array([1, 3, 6, 9, 10, 13, 14, 17, 21])

mean_X = np.mean(X)
mean_Y = np.mean(Y)

numerator = np.sum((X - mean_X) * (Y - mean_Y))
denominator = np.sum((X - mean_X) ** 2)
slope = numerator / denominator
intercept = mean_Y - slope * mean_X

Y_pred = slope * X + intercept

mse = np.mean((Y - Y_pred) ** 2)

ss_total = np.sum((Y - mean_Y) ** 2)
ss_residual = np.sum((Y - Y_pred) ** 2)
r_squared = 1 - (ss_residual / ss_total)

X_to_predict = 4
Y_pred_X4 = slope * X_to_predict + intercept

print(f"Slope: {slope}")
print(f"Intercept: {intercept}")
print(f"Predicted Y for X=4: {Y_pred_X4}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r_squared}")

plt.scatter(X, Y, label='Data Points')
plt.plot(X, Y_pred, color='red', label='Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.title('Linear Regression')
plt.show()
```

```
Slope: 2.3666666666666667
Intercept: -3.7555555555555546
Predicted Y for X=4: 5.711111111111112
Mean Squared Error: 0.46172839506172825
R-squared: 0.9877857609405617
```