Experiment :- 04

* Aim :- To perform linear regression in Python.

* Libraries:- Numpy, matplot lib

* Theory:-

- Regression searches for relationship among variables. In ~~and~~ other words, we need to find a function that maps some features /variables to others sufficiently well.

- The dependent features are called as dependent varables, outputs, responses.
The independent features are called as independent variables, input , predictors.

* Algorithm.

→ Least square method- finding best fit line
Least squares is a statistical method used to determine the best fit line or the regression line by minimizing the sum of squares created by a mathematical function.
The "square" function refers to squaring the distance between the data point & regression line.
The line with minimum value of the sum of square is the best fit regression line.

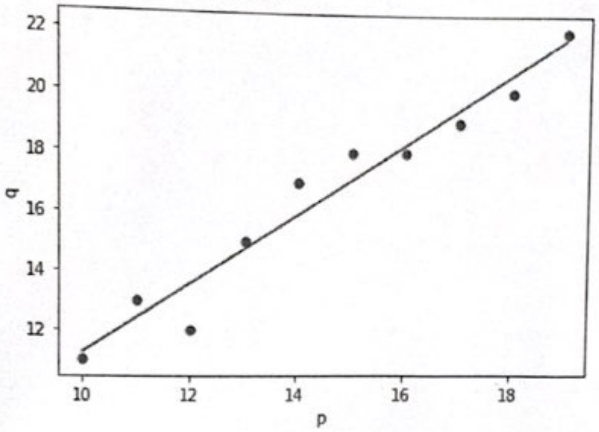
* Regression line :- $y = mx + c$
here, y = dependent variable
x = Independent variable ; c = y-intercept.

$$m = \frac{\Sigma (x-\bar{x})(y-\bar{y})}{\Sigma (x-\bar{x})^2}$$

```
import numpy as np
import matplotlib.pyplot as mtplt


def estimate_coeff(p, q):
# Here, we will estimate the total number of points or observation
    n1 = nmp.size(p)
# Now, we will calculate the mean of a and b vector
    m_p = nmp.mean(p)
    m_q = nmp.mean(q)

# here, we will calculate the cross deviation and deviation about a
    SS_pq = nmp.sum(q * p) - n1 * m_q * m_p
    SS_pp = nmp.sum(p * p) - n1 * m_p * m_p

# here, we will calculate the regression coefficients
    b_1 = SS_pq / SS_pp
    b_0 = m_q - b_1 * m_p

    return (b_0, b_1)


def plot_regression_line(p, q, b):
# Now, we will plot the actual points or observation as scatter plot
    mtplt.scatter(p, q, color = "m",
            marker = "o", s = 30)

# here, we will calculate the predicted response vector
    q_pred = b[0] + b[1] * p

# here, we will plot the regression line
    mtplt.plot(p, q_pred, color = "g")

# here, we will put the labels
    mtplt.xlabel('p')
    mtplt.ylabel('q')

# here, we will define the function to show plot
    mtplt.show()


def main():
# entering the observation points or data
    p = np.array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
    q = np.array([11, 13, 12, 15, 17, 18, 18, 19, 20, 22])

# now, we will estimate the coefficients
    b = estimate_coeff(p, q)
    print("Estimated coefficients are :\nb_0 = {} \  \nb_1 = {}".format(b[0], b[1]))

# Now, we will plot the regression line
    plot_regression_line(p, q, b)
```

```
if __name__ == "__main__":
    main()
```

```
Estimated coefficients are :
b_0 = -0.4606060606060609 \
b_1 = 1.1696969696969697
```

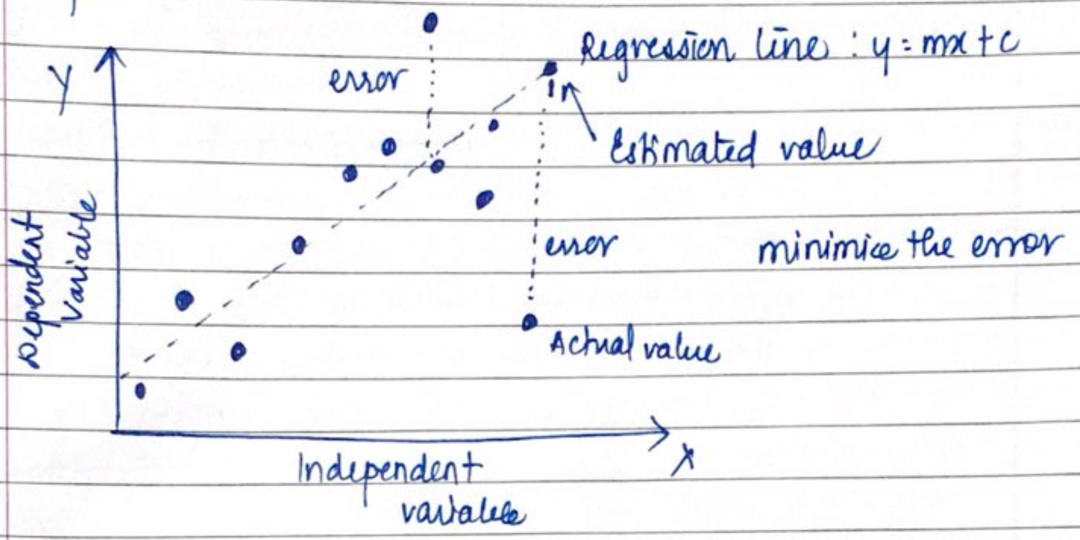


Colab paid products - Cancel contracts here

✓ 0s completed at 12:19 PM

● ×

* output



* Hance

Conclusion :- Hence, we have successfully performed linear regression in Python.