Name: Mohammed Mustafa Arif Chamadiya

Roll No : 6119051

Experiment No : 1

Experiment Name : Linear Regression

Q1 Linear Regression by manual calculation

Perform the Linear Regression by manual calculation [any example of your choice] and later implement linear Regression using Scikit_learn in python . Plot the result , print MSE, weight and coefficient .

Double-click (or enter) to edit

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable? (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable? These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

x	y	x^2	xy
1	1	1	1
2	9	4	18
3	3	9	9
4	2	16	8
5	8	25	40
6	7	36	42
$\Sigma x = 21$	$\Sigma y = 30$	$\Sigma x^2 = 91$	$\Sigma xy = 118$

$$M = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2} = \frac{6(118) - (21)(30)}{6(91) - (21)} \qquad M = 0.7428$$

$$B = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2} = \frac{(30)(91) - (21)(118)}{6(91) - (21)} \qquad B = 2.4$$

Step 01: Importing Libraries

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
```

Step 02: data for x

```
axis_X = np.array([[1],[2],[3],[4],[5],[6]])
```

Step 03: Test and Train Data

```
axis_X_train  = axis_X
axis_X_test = axis_X
```

Step 04: data for y

```
axis_y  =  np.array([[1],[9],[3],[2],[8],[7]])
```

Step 05: Splitting Data

```
axis_y_train  = axis_y
axis_y_test = axis_y
```

Step 06 : Fitting in model

```
model = linear_model.LinearRegression()
model.fit(axis_X_train, axis_y_train)

     LinearRegression()
```

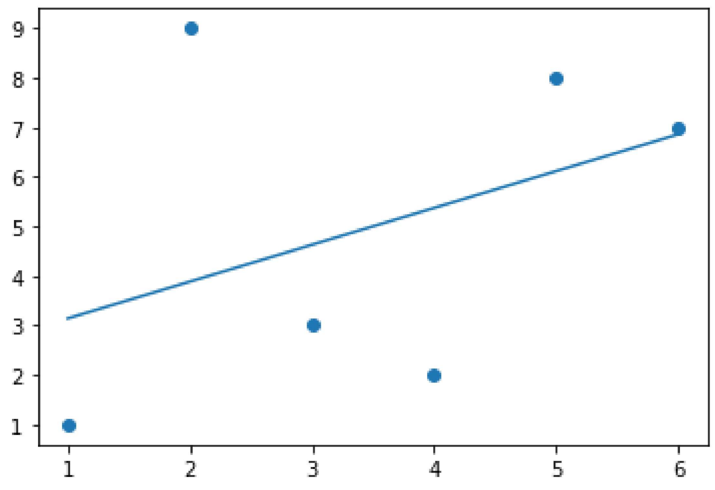
Step 07 :Predicting the Model and Printing Coefficient, Mean Squared error and Intercept

```
axis_y_predicted = model.predict(axis_X_test)
print(" ####Results######")
print(" Mean squared error is:  ", mean_squared_error(axis_y_test, axis_y_predicted))
print("Weights:  ",  model.coef_)
print(" Intercept:  ",  model.intercept_)

      ####Results######
      Mean squared error is:   8.057142857142859
     Weights:   [[0.74285714]]
      Intercept:   [2.4]
```

Step 08: Plotting Graph

```
plt.scatter(axis_X_test, axis_y_test)
plt.plot(axis_X_test, axis_y_predicted)
plt.show()
```



2) Implement Linear regression on Scikit-learn diabetes dataset and explain the code .

Step 01: Importing Libraries

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

```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn import datasets
```

Step 02: load diabetes Data

```
diabetes = datasets.load_diabetes()
print(diabetes)
```

```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990842, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06832974, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04687948,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986,  0.00306441]]), 'target': array([151.,  75., 141., 206., 135.,  97
        69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
        68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
        87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
       259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
       128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
       150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
       200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
        42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
        83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
       104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
       173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
       107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
       220.,  57.]), 'frame': None, 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\
```

Step 03: Getting basic characteristics

```
print("diabetes dataset is {}".format(diabetes.DESCR))
print("diabetes data size is {}".format(diabetes.data.shape))
print("diabetes target size is {}".format(diabetes.target.shape))
print("diabetes data has {} features, the feature names are {}".format(diabetes.data.shape[1]
print("diabetes data has {} samples, the target label names {}".format(diabetes.data.shape[1]
diabetes_x  =  diabetes.data[:,np.newaxis,2]
```

```
diabetes dataset is .. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

  :Number of Instances: 442

  :Number of Attributes: First 10 columns are numeric predictive values

  :Target: Column 11 is a quantitative measure of disease progression one year after

  :Attribute Information:
      - age     age in years
      - sex
      - bmi     body mass index
      - bp      average blood pressure
      - s1      tc, total serum cholesterol
      - s2      ldl, low-density lipoproteins
      - s3      hdl, high-density lipoproteins
      - s4      tch, total cholesterol / HDL
      - s5      ltg, possibly log of serum triglycerides level
      - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the st

Source URL:
```

```
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angl
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
diabetes data size is (442, 10)
diabetes target size is (442,)
diabetes data has 10 features, the feature names are ['age', 'sex', 'bmi', 'bp', 's1'
diabetes data has 10 samples, the target label names [151.  75. 141. 206. 135.  97. 1
 118. 171. 166. 144.  97. 168.  68.  49.  68. 245. 184. 202. 137.  85.
 131. 283. 129.  59. 341.  87.  65. 102. 265. 276. 252.  90. 100.  55.
  61.  92. 259.  53. 190. 142.  75. 142. 155. 225.  59. 104. 182. 128.
  52.  37. 170. 170.  61. 144.  52. 128.  71. 163. 150.  97. 160. 178.
  48. 270. 202. 111.  85.  42. 170. 200. 252. 113. 143.  51.  52. 210.
  65. 141.  55. 134.  42. 111.  98. 164.  48.  96.  90. 162. 150. 279.
  92.  83. 128. 102. 302. 198.  95.  53. 134. 144. 232.  81. 104.  59.
 246. 297. 258. 229. 275. 281. 179. 200. 200. 173. 180.  84. 121. 161.
  99. 109. 115. 268. 274. 158. 107.  83. 103. 272.  85. 280. 336. 281.
 118. 317. 235.  60. 174. 259. 178. 128.  96. 126. 288.  88. 292.  71.
 197. 186.  25.  84.  96. 195.  53. 217. 172. 131. 214.  59.  70. 220.
 268. 152.  47.  74. 295. 101. 151. 127. 237. 225.  81. 151. 107.  64.
 138. 185. 265. 101. 137. 143. 141.  79. 292. 178.  91. 116.  86. 122.
  72. 129. 142.  90. 158.  39. 196. 222. 277.  99. 196. 202. 155.  77.
```

Step 04: Splitting Training and testing model

```
diabetes_x_train  = diabetes_x[:-30]
diabetes_x_test = diabetes_x[-30:]
diabetes_y_train  = diabetes.target[:-30]
diabetes_y_test = diabetes.target[-30:]
```

Step 05: Creating Linear regression Object and fitting the model

```
model = linear_model.LinearRegression()
model.fit(diabetes_x_train, diabetes_y_train)
```

```
LinearRegression()
```

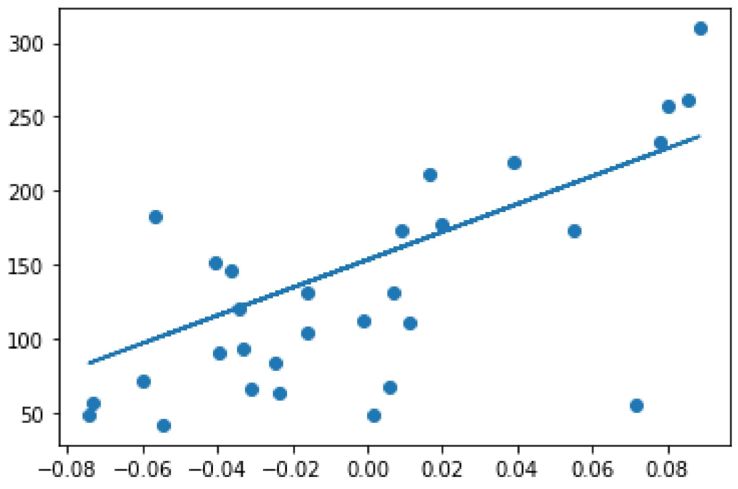
Step 06 :Predicting the model and Printing Coefficient, Mean Squared error and Intercept

```
diabetes_y_pred = model.predict(diabetes_x_test)
print("Mean Sqaured Error is:", mean_squared_error(diabetes_y_test,diabetes_y_pred))
print("Intercept:",model.intercept_)
print("Weights:",model.coef_)
```

```
Mean Sqaured Error is: 3035.0601152912686
Intercept: 153.39713623331698
Weights: [941.43097333]
```

Step 07: Plotting graph

```
plt.scatter(diabetes_x_test, diabetes_y_test)
plt.plot(diabetes_x_test, diabetes_y_pred)
plt.show()
```


300
250
200
150
100
50
−0.08
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08